Руководство по LABPP\_Automat для ARCHICAD

**Table of contents**

[Введение](#_topic_Newtopic) [33](#_topic_Newtopic)

[Особенности языка](#_topic_Newtopic2) [37](#_topic_Newtopic2)

[Моя первая программа](#_topic_Newtopic1) [37](#_topic_Newtopic1)

[Кодировка у файлов программных модулей](#_topic_Newtopic33) [38](#_topic_Newtopic33)

[Структура каталогов у LabPP\_Automat](#_topic_Structure_Folders_LabPP_Automat) [39](#_topic_Structure_Folders_LabPP_Automat)

[Конфигурационный файл config.cpp у конфигурации](#_topic_configcpp) [41](#_topic_configcpp)

[create\_button](#_topic_create_button1) [41](#_topic_create_button1)

[create\_iconbutton](#_topic_create_iconbutton1) [42](#_topic_create_iconbutton1)

[set\_palette\_size\_and\_message\_place](#_topic_set_palette_size_and_message_pl1) [42](#_topic_set_palette_size_and_message_pl1)

[Списки для выборки элементов из проекта](#_topic_Newtopic34) [42](#_topic_Newtopic34)

[Генезис свойств](#_topic_Newtopic37) [43](#_topic_Newtopic37)

[Окна LABPP](#_topic_LABPP1) [43](#_topic_LABPP1)

[Семантика языка C++ для ARCHICAD](#_topic_CARCHICAD) [44](#_topic_CARCHICAD)

[Общая структура программы](#_topic_Newtopic4) [44](#_topic_Newtopic4)

[Функция main()](#_topic_main) [44](#_topic_main)

[Идентификаторы](#_topic_Newtopic5) [45](#_topic_Newtopic5)

[Типы переменных, видимость и виды](#_topic_Newtopic6) [45](#_topic_Newtopic6)

[Пакетное объявление и обращение к переменным через идентификаторы, задаваемые текстом - var\_by\_txt()](#_topic__var_by_txt) [46](#_topic__var_by_txt)

[init](#_topic_init) [47](#_topic_init)

[is\_exist](#_topic_is_exist) [49](#_topic_is_exist)

[set](#_topic_set3) [49](#_topic_set3)

[get](#_topic_get3) [49](#_topic_get3)

[Комментарии](#_topic_Newtopic7) [50](#_topic_Newtopic7)

[Организация циклов](#_topic_Newtopic8) [50](#_topic_Newtopic8)

[Цикл for](#_topic_for) [50](#_topic_for)

[Цикл do-while](#_topic_do_while) [50](#_topic_do_while)

[Цикл while](#_topic_while) [50](#_topic_while)

[Логические операции](#_topic_Newtopic9) [50](#_topic_Newtopic9)

[Условные переходы](#_topic_Newtopic10) [51](#_topic_Newtopic10)

[Оператор if](#_topic_if) [51](#_topic_if)

[Конструкция if-else](#_topic_if_else) [51](#_topic_if_else)

[Конструкция if-else if-else](#_topic_if_elseif_else) [51](#_topic_if_elseif_else)

[Оператор switch](#_topic_switch) [51](#_topic_switch)

[Создание собственных функций (подпрограмм)](#_topic_Newtopic11) [51](#_topic_Newtopic11)

[Директива #include](#_topic_include) [52](#_topic_include)

[Особая конструкция cpp для ARCHICAD - управляемый #include](#_topic_cppARCHICAD_include) [53](#_topic_cppARCHICAD_include)

[Директива #pragma region](#_topic_pragmaregion) [53](#_topic_pragmaregion)

[Поиск по шаблону (коды REGEX)](#_topic_REGEX) [53](#_topic_REGEX)

[Логическое И и логическое ИЛИ в регулярных выражениях](#_topic_Newtopic42) [54](#_topic_Newtopic42)

[Функции для работы с оболочкой shell\_func](#_topic_shell_func) [55](#_topic_shell_func)

[get\_path](#_topic_get_path) [55](#_topic_get_path)

[set\_cur\_dir](#_topic_set_cur_dir) [55](#_topic_set_cur_dir)

[shellexecute](#_topic_shellexecute) [56](#_topic_shellexecute)

[ExecCmd](#_topic_ExecCmd) [56](#_topic_ExecCmd)

[file](#_topic_file) [57](#_topic_file)

[copy](#_topic_copy) [57](#_topic_copy)

[is\_exist](#_topic_is_exist1) [58](#_topic_is_exist1)

[delete](#_topic_delete1) [58](#_topic_delete1)

[get\_SLANG\_CODE](#_topic_get_SLANG_CODE) [58](#_topic_get_SLANG_CODE)

[Управление окнами labpp](#_topic_labpp) [59](#_topic_labpp)

[Буффер обмена Windows](#_topic_Windows) [59](#_topic_Windows)

[clipboard\_get](#_topic_clipboard_get) [59](#_topic_clipboard_get)

[clipboard\_set](#_topic_clipboard_set) [59](#_topic_clipboard_set)

[Функции автоматизации – Automate](#_topic__Automate) [59](#_topic__Automate)

[CloseCurWindow](#_topic_CloseCurWindow) [59](#_topic_CloseCurWindow)

[Zoom](#_topic_Zoom) [59](#_topic_Zoom)

[GoToView](#_topic_GoToView) [59](#_topic_GoToView)

[ChangeWindow](#_topic_ChangeWindow) [60](#_topic_ChangeWindow)

[ShowSelectionIn3D](#_topic_ShowSelectionIn3D) [60](#_topic_ShowSelectionIn3D)

[ShowAllIn3D](#_topic_ShowAllIn3D) [60](#_topic_ShowAllIn3D)

[ZoomToElements](#_topic_ZoomToElements) [60](#_topic_ZoomToElements)

[Redraw](#_topic_Redraw) [60](#_topic_Redraw)

[Rebuild](#_topic_Rebuild) [60](#_topic_Rebuild)

[Функции окружения - Environment](#_topic__Environment) [60](#_topic__Environment)

[AddLibraries](#_topic_AddLibraries) [60](#_topic_AddLibraries)

[Story\_GoTo](#_topic_Story_GoTo) [61](#_topic_Story_GoTo)

[Функции для работы с базами проекта - database](#_topic__database) [61](#_topic__database)

[GetCurrent](#_topic_GetCurrent) [61](#_topic_GetCurrent)

[Select](#_topic_Select3) [61](#_topic_Select3)

[SetTypeID](#_topic_SetTypeID) [61](#_topic_SetTypeID)

[GetTypeID](#_topic_GetTypeID) [62](#_topic_GetTypeID)

[GetLocOrigo](#_topic_GetLocOrigo) [62](#_topic_GetLocOrigo)

[get\_section\_infoidtext](#_topic_get_section_infoidtext) [63](#_topic_get_section_infoidtext)

[get\_sectionmarkers2dpositions](#_topic_get_sectionmarkers2dpositions) [63](#_topic_get_sectionmarkers2dpositions)

[GetRVMChanges](#_topic_GetRVMChanges) [64](#_topic_GetRVMChanges)

[GetRVMLayoutCurrentRevisionChanges](#_topic_GetRVMLayoutCurrentRevisionChang) [64](#_topic_GetRVMLayoutCurrentRevisionChang)

[GetTypeName](#_topic_GetTypeName) [65](#_topic_GetTypeName)

[GetSizeX, GetSizeY](#_topic_GetSizeXGetSizeY) [65](#_topic_GetSizeXGetSizeY)

[leftMargin,topMargin,rightMargin,bottomMargin](#_topic_leftMargintopMarginrightMarginbo) [66](#_topic_leftMargintopMarginrightMarginbo)

[layoutPageNumber](#_topic_layoutPageNumber) [66](#_topic_layoutPageNumber)

[layoutName](#_topic_layoutName) [66](#_topic_layoutName)

[Вывод в окно сообщений](#_topic_Newtopic3) [67](#_topic_Newtopic3)

[cout](#_topic_cout) [67](#_topic_cout)

[coutvar](#_topic_coutvar) [67](#_topic_coutvar)

[setcfg("COUT",...)](#_topic_setcfgCOUT) [67](#_topic_setcfgCOUT)

[Файловые операции](#_topic_Newtopic12) [67](#_topic_Newtopic12)

[open](#_topic_open) [67](#_topic_open)

[write](#_topic_write) [67](#_topic_write)

[read](#_topic_read1) [68](#_topic_read1)

[close](#_topic_close) [68](#_topic_close)

[SetIsUNICODE](#_topic_SetIsUNICODE) [68](#_topic_SetIsUNICODE)

[SetIsUTF16](#_topic_SetIsUTF16) [68](#_topic_SetIsUTF16)

[Строковые функции](#_topic_Newtopic13) [69](#_topic_Newtopic13)

[strcmp](#_topic_strcmp) [69](#_topic_strcmp)

[tolower](#_topic_tolower) [69](#_topic_tolower)

[toupper](#_topic_toupper) [69](#_topic_toupper)

[alltrim](#_topic_alltrim) [69](#_topic_alltrim)

[strcontains](#_topic_strcontains) [70](#_topic_strcontains)

[strreplace](#_topic_strreplace) [70](#_topic_strreplace)

[strlen](#_topic_strlen) [70](#_topic_strlen)

[substr](#_topic_substr) [70](#_topic_substr)

[strposcharset](#_topic_strposcharset) [70](#_topic_strposcharset)

[strposcharget](#_topic_strposcharget) [71](#_topic_strposcharget)

[strleft](#_topic_strleft) [71](#_topic_strleft)

[strright](#_topic_strright) [71](#_topic_strright)

[str\_regex\_search](#_topic_str_regex_search) [71](#_topic_str_regex_search)

[str\_regex\_match](#_topic_str_regex_match) [72](#_topic_str_regex_match)

[str\_regex\_replace](#_topic_str_regex_replace) [72](#_topic_str_regex_replace)

[str\_regex\_substr](#_topic_str_regex_substr) [73](#_topic_str_regex_substr)

[straddslashtoregexcodes](#_topic_straddslashtoregexcodes) [74](#_topic_straddslashtoregexcodes)

[strremovebaddelimiters](#_topic_strremovebaddelimiters) [74](#_topic_strremovebaddelimiters)

[str2threekeys](#_topic_str2threekeys) [74](#_topic_str2threekeys)

[strCompareLikeABC10\_a](#_topic_strCompareLikeABC10_a) [74](#_topic_strCompareLikeABC10_a)

[strreplace\_escape\_u](#_topic_strreplace_escape_u) [75](#_topic_strreplace_escape_u)

[strisnumber](#_topic_strisnumber) [75](#_topic_strisnumber)

[smallnums](#_topic_smallnums) [75](#_topic_smallnums)

[ts\_text\_mc](#_topic_ts_text_mc) [75](#_topic_ts_text_mc)

[Объект ac\_element\_guid (ts\_guid)](#_topic_ac_element_guidts_guid) [76](#_topic_ac_element_guidts_guid)

[Объект ac\_element](#_topic_ac_element) [76](#_topic_ac_element)

[ac\_request](#_topic_ac_request) [76](#_topic_ac_request)

[get\_floor\_name\_by\_floor\_index](#_topic_get_floor_name_by_floor_index) [76](#_topic_get_floor_name_by_floor_index)

[get\_floor\_index\_by\_floor\_name](#_topic_get_floor_index_by_floor_name) [76](#_topic_get_floor_index_by_floor_name)

[store\_cur\_element\_to\_descr](#_topic_store_cur_element_to_descr) [77](#_topic_store_cur_element_to_descr)

[set\_current\_element\_from\_descr](#_topic_set_current_element_from_descr) [77](#_topic_set_current_element_from_descr)

[set\_element\_by\_guidstr\_as\_current](#_topic_set_element_by_guidstr_as_curren) [77](#_topic_set_element_by_guidstr_as_curren)

[get\_guid\_from\_element](#_topic_get_guid_from_element) [77](#_topic_get_guid_from_element)

[load\_element\_from\_guid](#_topic_load_element_from_guid) [78](#_topic_load_element_from_guid)

[load\_elements\_list](#_topic_load_elements_list) [78](#_topic_load_elements_list)

[set\_current\_element\_from\_list](#_topic_set_current_element_from_list) [78](#_topic_set_current_element_from_list)

[add\_elements\_list](#_topic_add_elements_list) [79](#_topic_add_elements_list)

[load\_elements\_list\_from\_selection и add\_elements\_list\_from\_selection](#_topic_load_elements_list_from_selectio) [79](#_topic_load_elements_list_from_selectio)

[load\_elements\_list\_curdb](#_topic_load_elements_list_curdb) [79](#_topic_load_elements_list_curdb)

[clear\_list](#_topic_clear_list) [79](#_topic_clear_list)

[store\_current\_element\_to\_list](#_topic_store_current_element_to_list) [79](#_topic_store_current_element_to_list)

[get\_loaded\_elements\_list\_count](#_topic_get_loaded_elements_list_count) [79](#_topic_get_loaded_elements_list_count)

[select\_elements\_from\_list](#_topic_select_elements_from_list) [79](#_topic_select_elements_from_list)

[Слой - layer](#_topic__layer) [80](#_topic__layer)

[create](#_topic_create) [80](#_topic_create)

[get\_index](#_topic_get_index) [80](#_topic_get_index)

[set\_layer\_visible](#_topic_set_layer_visible) [80](#_topic_set_layer_visible)

[get\_element\_overall\_dimensions](#_topic_get_element_overall_dimensions) [80](#_topic_get_element_overall_dimensions)

[get\_quantity\_value](#_topic_get_quantity_value) [81](#_topic_get_quantity_value)

[get\_layer\_by\_substring](#_topic_get_layer_by_substring) [86](#_topic_get_layer_by_substring)

[elem\_user\_property](#_topic_elem_user_property) [87](#_topic_elem_user_property)

[get](#_topic_get) [87](#_topic_get)

[set](#_topic_set) [88](#_topic_set)

[create](#_topic_create1) [88](#_topic_create1)

[Property\_Import](#_topic_Property_Import) [89](#_topic_Property_Import)

[get\_object\_property\_value](#_topic_get_object_property_value) [89](#_topic_get_object_property_value)

[set\_object\_property\_value](#_topic_set_object_property_value) [90](#_topic_set_object_property_value)

[set\_object\_property\_value\_curdb](#_topic_set_object_property_value_curdb) [90](#_topic_set_object_property_value_curdb)

[get\_element\_value](#_topic_get_element_value) [90](#_topic_get_element_value)

[set\_element\_value](#_topic_set_element_value) [91](#_topic_set_element_value)

[assign\_element\_values](#_topic_assign_element_values) [91](#_topic_assign_element_values)

[load\_element\_default\_values](#_topic_load_element_default_values) [91](#_topic_load_element_default_values)

[set\_element\_infoidtext](#_topic_set_element_infoidtext) [91](#_topic_set_element_infoidtext)

[create\_element\_on\_project](#_topic_create_element_on_project) [92](#_topic_create_element_on_project)

[get\_element\_infoidtext](#_topic_get_element_infoidtext) [92](#_topic_get_element_infoidtext)

[project\_property](#_topic_project_property) [92](#_topic_project_property)

[get](#_topic_get4) [92](#_topic_get4)

[CalcUnitsAreaDecimals](#_topic_CalcUnitsAreaDecimals) [92](#_topic_CalcUnitsAreaDecimals)

[autotext](#_topic_autotext) [92](#_topic_autotext)

[get](#_topic_get1) [92](#_topic_get1)

[set](#_topic_set1) [93](#_topic_set1)

[create](#_topic_create2) [93](#_topic_create2)

[get\_as\_table](#_topic_get_as_table) [93](#_topic_get_as_table)

[interface\_input2point](#_topic_interface_input2point) [94](#_topic_interface_input2point)

[interface\_input2dline](#_topic_interface_input2dline) [94](#_topic_interface_input2dline)

[interface\_input3dline](#_topic_interface_input3dline) [94](#_topic_interface_input3dline)

[interface\_input2dpoly](#_topic_interface_input2dpoly) [94](#_topic_interface_input2dpoly)

[get\_levelfromprojectnull\_by\_floorindexandlevel](#_topic_get_levelfromprojectnull_by_floo) [95](#_topic_get_levelfromprojectnull_by_floo)

[get\_floorindexandlevel\_by\_levelfromprojectnull](#_topic_get_floorindexandlevel_by_levelf) [97](#_topic_get_floorindexandlevel_by_levelf)

[get\_storyinfo\_as\_table](#_topic_get_storyinfo_as_table) [97](#_topic_get_storyinfo_as_table)

[elem\_classification\_get\_item\_value](#_topic_elem_classification_get_item_val) [98](#_topic_elem_classification_get_item_val)

[elem\_classification\_get\_parent\_item\_value](#_topic_elem_classification_get_parent_i) [98](#_topic_elem_classification_get_parent_i)

[hotlinks](#_topic_hotlinks) [98](#_topic_hotlinks)

[GetHotlinkNodes](#_topic_GetHotlinkNodes) [98](#_topic_GetHotlinkNodes)

[geometry\_calc\_2d](#_topic_geometry_calc_2d) [101](#_topic_geometry_calc_2d)

[is\_point\_on\_element\_polygon](#_topic_is_point_on_element_polygon1) [101](#_topic_is_point_on_element_polygon1)

[is\_curelem\_inside\_element\_polygon](#_topic_is_curelem_inside_element_polyg1) [101](#_topic_is_curelem_inside_element_polyg1)

[is\_point\_on\_element\_refside](#_topic_is_point_on_element_refside) [102](#_topic_is_point_on_element_refside)

[where\_C\_leftorright](#_topic_where_C_leftorright) [102](#_topic_where_C_leftorright)

[getDistPointFrom2dLine](#_topic_getDistPointFrom2dLine) [102](#_topic_getDistPointFrom2dLine)

[geometry\_calc\_3d](#_topic_geometry_calc_3d) [102](#_topic_geometry_calc_3d)

[RotatePoint3dAroundVect3d](#_topic_RotatePoint3dAroundVect3d) [102](#_topic_RotatePoint3dAroundVect3d)

[GetBounds](#_topic_GetBounds) [103](#_topic_GetBounds)

[classification](#_topic_classification) [103](#_topic_classification)

[find\_item\_by\_name](#_topic_find_item_by_name) [103](#_topic_find_item_by_name)

[check\_class\_is\_inbranch](#_topic_check_class_is_inbranch) [103](#_topic_check_class_is_inbranch)

[CreateClassificationSystem](#_topic_CreateClassificationSystem) [103](#_topic_CreateClassificationSystem)

[CreateClassificationItem](#_topic_CreateClassificationItem) [104](#_topic_CreateClassificationItem)

[DeleteClassificationItem](#_topic_DeleteClassificationItem) [104](#_topic_DeleteClassificationItem)

[set\_item\_to\_curelem](#_topic_set_item_to_curelem) [104](#_topic_set_item_to_curelem)

[GetClassificationSystems](#_topic_GetClassificationSystems) [105](#_topic_GetClassificationSystems)

[GetClassificationSystem\_guid\_by\_name](#_topic_GetClassificationSystem_guid_by_) [105](#_topic_GetClassificationSystem_guid_by_)

[Classification\_Import](#_topic_Classification_Import) [106](#_topic_Classification_Import)

[GetClassifSystem\_By\_ClassTtemGuid](#_topic_GetClassifSystem_By_ClassTtemGui) [106](#_topic_GetClassifSystem_By_ClassTtemGui)

[do\_move\_rot\_and\_scale\_elemlist](#_topic_do_move_rot_and_scale_elemlist) [107](#_topic_do_move_rot_and_scale_elemlist)

[clearmessagewindow](#_topic_clearmessagewindow) [108](#_topic_clearmessagewindow)

[get\_labpp\_product\_name](#_topic_get_labpp_product_name) [108](#_topic_get_labpp_product_name)

[ac\_request\_special](#_topic_ac_request_special) [108](#_topic_ac_request_special)

[get\_pie](#_topic_get_pie) [108](#_topic_get_pie)

[load\_elements\_list](#_topic_load_elements_list1) [108](#_topic_load_elements_list1)

[add\_elements\_list](#_topic_add_elements_list1) [110](#_topic_add_elements_list1)

[load\_elements\_list\_from\_selection](#_topic_load_elements_list_from_selecti1) [110](#_topic_load_elements_list_from_selecti1)

[add\_elements\_list\_from\_selection](#_topic_add_elements_list_from_selection) [110](#_topic_add_elements_list_from_selection)

[load\_elements\_list\_curdb](#_topic_load_elements_list_curdb1) [110](#_topic_load_elements_list_curdb1)

[copy\_elements\_to\_list\_from\_list\_by\_filter](#_topic_copy_elements_to_list_from_list_) [111](#_topic_copy_elements_to_list_from_list_)

[check\_cur\_element\_by\_filter](#_topic_check_cur_element_by_filter) [111](#_topic_check_cur_element_by_filter)

[GetMaterialSurfaceSquare](#_topic_GetMaterialSurfaceSquare) [111](#_topic_GetMaterialSurfaceSquare)

[GetMaterialSurfaceSquareTable](#_topic_GetMaterialSurfaceSquareTable) [111](#_topic_GetMaterialSurfaceSquareTable)

[get\_element\_value](#_topic_get_element_value1) [112](#_topic_get_element_value1)

[buildingMaterialName\_thickness](#_topic_buildingMaterialName_thickness) [112](#_topic_buildingMaterialName_thickness)

[buildingMaterialName\_used](#_topic_buildingMaterialName_used) [113](#_topic_buildingMaterialName_used)

[set\_element\_value](#_topic_set_element_value1) [113](#_topic_set_element_value1)

[set\_element\_value\_curdb](#_topic_set_element_value_curdb) [113](#_topic_set_element_value_curdb)

[RunGDLParScript](#_topic_RunGDLParScript) [113](#_topic_RunGDLParScript)

[get\_building\_material](#_topic_get_building_material) [113](#_topic_get_building_material)

[get\_build\_material\_value](#_topic_get_build_material_value) [114](#_topic_get_build_material_value)

[get\_material](#_topic_get_material) [114](#_topic_get_material)

[load\_zonerelatedwallstolist](#_topic_load_zonerelatedwallstolist) [115](#_topic_load_zonerelatedwallstolist)

[load\_zonerelatedwallparttotable](#_topic_load_zonerelatedwallparttotable) [115](#_topic_load_zonerelatedwallparttotable)

[Attribute\_GetNum](#_topic_Attribute_GetNum) [115](#_topic_Attribute_GetNum)

[clipboard\_set](#_topic_clipboard_set1) [116](#_topic_clipboard_set1)

[clipboard\_get](#_topic_clipboard_get1) [116](#_topic_clipboard_get1)

[element\_user\_data](#_topic_element_user_data) [116](#_topic_element_user_data)

[read](#_topic_read) [116](#_topic_read)

[write](#_topic_write1) [116](#_topic_write1)

[write\_to\_file](#_topic_write_to_file) [116](#_topic_write_to_file)

[clear](#_topic_clear) [118](#_topic_clear)

[delete\_variable](#_topic_delete_variable) [118](#_topic_delete_variable)

[get\_variable\_value](#_topic_get_variable_value) [118](#_topic_get_variable_value)

[set\_variable\_value](#_topic_set_variable_value) [118](#_topic_set_variable_value)

[linkingElems](#_topic_linkingElems) [118](#_topic_linkingElems)

[uplinkBiWardByFlags](#_topic_uplinkBiWardByFlags) [118](#_topic_uplinkBiWardByFlags)

[getLinkedElemsByFlags](#_topic_getLinkedElemsByFlags) [119](#_topic_getLinkedElemsByFlags)

[set\_TSMAX\_ELEMLIST\_NUMBER](#_topic_set_TSMAX_ELEMLIST_NUMBER) [119](#_topic_set_TSMAX_ELEMLIST_NUMBER)

[get\_TSMAX\_ELEMLIST\_NUMBER](#_topic_get_TSMAX_ELEMLIST_NUMBER) [119](#_topic_get_TSMAX_ELEMLIST_NUMBER)

[set\_elemlistname](#_topic_set_elemlistname) [119](#_topic_set_elemlistname)

[Get\_Elements\_Collisions](#_topic_Get_Elements_Collisions) [120](#_topic_Get_Elements_Collisions)

[delete\_elements\_from\_list](#_topic_delete_elements_from_list) [120](#_topic_delete_elements_from_list)

[delete\_elements\_from\_project](#_topic_delete_elements_from_project) [120](#_topic_delete_elements_from_project)

[ac\_request\_attributes](#_topic_ac_request_attributes) [121](#_topic_ac_request_attributes)

[Таблица типов атрибутов](#_topic_Newtopic36) [121](#_topic_Newtopic36)

[create\_attr](#_topic_create_attr) [121](#_topic_create_attr)

[get\_num](#_topic_get_num) [122](#_topic_get_num)

[delete\_attr](#_topic_delete_attr) [122](#_topic_delete_attr)

[set\_attr\_value](#_topic_set_attr_value) [122](#_topic_set_attr_value)

[get\_attr\_value](#_topic_get_attr_value) [122](#_topic_get_attr_value)

[ac\_typeidfromstring](#_topic_ac_typeidfromstring) [123](#_topic_ac_typeidfromstring)

[ac\_getresvaluetype](#_topic_ac_getresvaluetype) [123](#_topic_ac_getresvaluetype)

[ac\_getstrvalue](#_topic_ac_getstrvalue) [123](#_topic_ac_getstrvalue)

[ac\_getnumvalue](#_topic_ac_getnumvalue) [123](#_topic_ac_getnumvalue)

[ac\_getupvaluestatus](#_topic_ac_getupvaluestatus) [123](#_topic_ac_getupvaluestatus)

[Гравитация на поверхность](#_topic_Newtopic14) [124](#_topic_Newtopic14)

[do\_elements\_landing - приземление элементов](#_topic_do_elements_landing_) [124](#_topic_do_elements_landing_)

[do\_surface\_landing - приземление по точкам](#_topic_do_surface_landing_)  [124](#_topic_do_surface_landing_)

[do\_point\_landing - приземление точки X,Y](#_topic_do_point_landing_XY)  [124](#_topic_do_point_landing_XY)

[3d сетки](#_topic_3d) [125](#_topic_3d)

[get\_lands\_combined\_contour\_coords](#_topic_get_lands_combined_contour_coord) [125](#_topic_get_lands_combined_contour_coord)

[do\_create\_contour\_around\_points](#_topic_do_create_contour_around_points) [125](#_topic_do_create_contour_around_points)

[get\_lands\_topcoords](#_topic_get_lands_topcoords) [126](#_topic_get_lands_topcoords)

[do\_create\_mesh\_by\_points\_and\_lines](#_topic_do_create_mesh_by_points_and_lin) [126](#_topic_do_create_mesh_by_points_and_lin)

[Диалоги](#_topic_Newtopic15) [127](#_topic_Newtopic15)

[Сообщения или выбор варианта - tsalert](#_topic__tsalert) [127](#_topic__tsalert)

[Поиск и выбор файла - dialog\_get\_filename](#_topic__dialog_get_filename) [128](#_topic__dialog_get_filename)

[editdoubledialog - ввод числа](#_topic_editdoubledialog_) [128](#_topic_editdoubledialog_)

[editintdialog - ввод целого числа](#_topic_editintdialog_) [129](#_topic_editintdialog_)

[editstringdialog - ввод строки](#_topic_editstringdialog_) [129](#_topic_editstringdialog_)

[editstringlistdialog - выбор строки из списка](#_topic_editstringlistdialog_) [130](#_topic_editstringlistdialog_)

[editcheckstringsdialog - выбор нескольких строк из списка](#_topic_editcheckstringsdialog_) [131](#_topic_editcheckstringsdialog_)

[editbooldialog - выбор Да/Нет](#_topic_editbooldialog_) [132](#_topic_editbooldialog_)

[editstringlistandcustomdialog - ввод строки или выбор из списка](#_topic_editstringlistandcustomdialog_) [132](#_topic_editstringlistandcustomdialog_)

[Работа с Excel](#_topic_Excel) [134](#_topic_Excel)

[excel\_attach](#_topic_excel_attach) [134](#_topic_excel_attach)

[excel\_detach](#_topic_excel_detach) [134](#_topic_excel_detach)

[excel\_putnumvalue](#_topic_excel_putnumvalue) [134](#_topic_excel_putnumvalue)

[excel\_putstrvalue](#_topic_excel_putstrvalue) [134](#_topic_excel_putstrvalue)

[excel\_select\_range](#_topic_excel_select_range) [134](#_topic_excel_select_range)

[excel\_visible](#_topic_excel_visible) [135](#_topic_excel_visible)

[excel\_speedup](#_topic_excel_speedup) [135](#_topic_excel_speedup)

[excel\_getnumvalue](#_topic_excel_getnumvalue) [135](#_topic_excel_getnumvalue)

[excel\_getstrvalue](#_topic_excel_getstrvalue) [135](#_topic_excel_getstrvalue)

[excel\_request](#_topic_excel_request) [135](#_topic_excel_request)

[set\_column\_width](#_topic_set_column_width) [135](#_topic_set_column_width)

[get\_column\_width](#_topic_get_column_width) [135](#_topic_get_column_width)

[set\_row\_height](#_topic_set_row_height) [135](#_topic_set_row_height)

[get\_row\_height](#_topic_get_row_height) [136](#_topic_get_row_height)

[set\_borders](#_topic_set_borders) [136](#_topic_set_borders)

[put\_selection\_values](#_topic_put_selection_values) [136](#_topic_put_selection_values)

[put\_selection\_fontvalues](#_topic_put_selection_fontvalues) [136](#_topic_put_selection_fontvalues)

[get\_selection\_area](#_topic_get_selection_area) [137](#_topic_get_selection_area)

[merged\_cell\_info](#_topic_merged_cell_info) [137](#_topic_merged_cell_info)

[is\_merge\_cells](#_topic_is_merge_cells) [137](#_topic_is_merge_cells)

[set\_backcolor](#_topic_set_backcolor) [138](#_topic_set_backcolor)

[get\_backcolor](#_topic_get_backcolor) [138](#_topic_get_backcolor)

[set\_interior](#_topic_set_interior) [138](#_topic_set_interior)

[get\_interior](#_topic_get_interior) [138](#_topic_get_interior)

[selection\_varvalues](#_topic_selection_varvalues) [138](#_topic_selection_varvalues)

[selection\_font\_varvalues](#_topic_selection_font_varvalues) [138](#_topic_selection_font_varvalues)

[sheet\_select](#_topic_sheet_select) [138](#_topic_sheet_select)

[range\_copy](#_topic_range_copy) [138](#_topic_range_copy)

[booknamedcell](#_topic_booknamedcell) [138](#_topic_booknamedcell)

[save\_active\_sheet\_as\_csv](#_topic_save_active_sheet_as_csv) [138](#_topic_save_active_sheet_as_csv)

[save\_copy\_as](#_topic_save_copy_as) [139](#_topic_save_copy_as)

[workbook\_open](#_topic_workbook_open) [139](#_topic_workbook_open)

[workbook\_close](#_topic_workbook_close) [140](#_topic_workbook_close)

[работа с примечаниями](#_topic_Newtopic30) [140](#_topic_Newtopic30)

[getcomment](#_topic_getcomment) [140](#_topic_getcomment)

[clearcomments](#_topic_clearcomments) [140](#_topic_clearcomments)

[setcomment](#_topic_setcomment) [140](#_topic_setcomment)

[Загрузка и выгрузка табличных данных](#_topic_Newtopic17) [140](#_topic_Newtopic17)

[excel\_chart - диаграммы в Excel](#_topic_excel_chart_Excel) [140](#_topic_excel_chart_Excel)

[AddChart](#_topic_AddChart) [140](#_topic_AddChart)

[SetChartType](#_topic_SetChartType) [140](#_topic_SetChartType)

[SetChartTitle](#_topic_SetChartTitle) [142](#_topic_SetChartTitle)

[SetSourceDataRange](#_topic_SetSourceDataRange) [142](#_topic_SetSourceDataRange)

[Пример](#_topic_Newtopic31) [143](#_topic_Newtopic31)

[Работа с Word](#_topic_Word) [143](#_topic_Word)

[word\_attach](#_topic_word_attach) [143](#_topic_word_attach)

[word\_detach](#_topic_word_detach) [143](#_topic_word_detach)

[word\_visible](#_topic_word_visible) [143](#_topic_word_visible)

[word\_request](#_topic_word_request) [143](#_topic_word_request)

[NСчитывание и запись полей переменных - docfield](#_topic_N_docfield) [144](#_topic_N_docfield)

[get](#_topic_get2) [144](#_topic_get2)

[set](#_topic_set2) [144](#_topic_set2)

[update\_all\_docfields](#_topic_update_all_docfields) [144](#_topic_update_all_docfields)

[Работа через ts\_idispatch](#_topic_ts_idispatch) [144](#_topic_ts_idispatch)

[Работа с AutoCAD](#_topic_AutoCAD) [145](#_topic_AutoCAD)

[autocad\_request](#_topic_autocad_request) [145](#_topic_autocad_request)

[attach](#_topic_attach) [145](#_topic_attach)

[detach](#_topic_detach) [146](#_topic_detach)

[specify\_color\_RGB](#_topic_specify_color_RGB) [146](#_topic_specify_color_RGB)

[draw\_text](#_topic_draw_text) [146](#_topic_draw_text)

[draw\_polyline](#_topic_draw_polyline) [146](#_topic_draw_polyline)

[draw\_line](#_topic_draw_line) [147](#_topic_draw_line)

[draw\_arc](#_topic_draw_arc) [147](#_topic_draw_arc)

[Пример - нарисовать контуры зон из ARCHICAD в AutoCAD полилиниями](#_topic__ARCHICADAutoCAD) [147](#_topic__ARCHICADAutoCAD)

[Функция object](#_topic_object) [149](#_topic_object)

[create](#_topic_create3) [149](#_topic_create3)

[delete](#_topic_delete) [149](#_topic_delete)

[Объект ts\_table](#_topic_ts_table) [150](#_topic_ts_table)

[Функция ts\_table](#_topic_ts_table1) [150](#_topic_ts_table1)

[init (clear)](#_topic_initclear1) [150](#_topic_initclear1)

[clear\_rows](#_topic_clear_rows) [150](#_topic_clear_rows)

[add\_column](#_topic_add_column) [150](#_topic_add_column)

[is\_column\_exist](#_topic_is_column_exist) [151](#_topic_is_column_exist)

[set\_first\_key](#_topic_set_first_key) [151](#_topic_set_first_key)

[add\_row](#_topic_add_row) [151](#_topic_add_row)

[add\_row\_sum](#_topic_add_row_sum) [152](#_topic_add_row_sum)

[sort](#_topic_sort) [152](#_topic_sort)

[sort\_back](#_topic_sort_back) [153](#_topic_sort_back)

[set\_columns\_to\_sort\_as\_LikeABC10\_a](#_topic_set_columns_to_sort_as_LikeABC10) [153](#_topic_set_columns_to_sort_as_LikeABC10)

[search](#_topic_search) [154](#_topic_search)

[search\_regex](#_topic_search_regex) [154](#_topic_search_regex)

[search\_extended](#_topic_search_extended) [154](#_topic_search_extended)

[search\_extended\_table](#_topic_search_extended_table) [154](#_topic_search_extended_table)

[search\_bounds](#_topic_search_bounds) [155](#_topic_search_bounds)

[select\_row](#_topic_select_row) [155](#_topic_select_row)

[seek\_from\_top, seek\_from\_bottom, seek\_from\_current](#_topic_seek_from_topseek_from_bottomsee) [155](#_topic_seek_from_topseek_from_bottomsee)

[add\_row\_from\_eq\_table](#_topic_add_row_from_eq_table) [155](#_topic_add_row_from_eq_table)

[add\_rows\_from\_eq\_table\_by\_filter](#_topic_add_rows_from_eq_table_by_filter) [156](#_topic_add_rows_from_eq_table_by_filter)

[add\_rows\_from\_eq\_table\_by\_regex\_filter](#_topic_add_rows_from_eq_table_by_regex_) [156](#_topic_add_rows_from_eq_table_by_regex_)

[add\_rows\_from\_eq\_table\_by\_extended\_filter](#_topic_add_rows_from_eq_table_by_extend) [156](#_topic_add_rows_from_eq_table_by_extend)

[add\_rows\_from\_eq\_table\_by\_extended\_filter\_table](#_topic_add_rows_from_eq_table_by_exten1) [157](#_topic_add_rows_from_eq_table_by_exten1)

[get\_value\_of](#_topic_get_value_of) [157](#_topic_get_value_of)

[get\_rows\_count](#_topic_get_rows_count) [157](#_topic_get_rows_count)

[get\_columns\_count](#_topic_get_columns_count) [157](#_topic_get_columns_count)

[set\_columnstypes](#_topic_set_columnstypes) [158](#_topic_set_columnstypes)

[get\_heading\_of](#_topic_get_heading_of) [158](#_topic_get_heading_of)

[print\_to\_str](#_topic_print_to_str) [158](#_topic_print_to_str)

[set\_columns\_to\_add\_sum](#_topic_set_columns_to_add_sum) [158](#_topic_set_columns_to_add_sum)

[Функции для выгрузки и загрузки таблиц](#_topic_Newtopic19) [159](#_topic_Newtopic19)

[export\_to\_excel](#_topic_export_to_excel) [159](#_topic_export_to_excel)

[import\_from\_excel](#_topic_import_from_excel) [160](#_topic_import_from_excel)

[import\_columns\_from\_excel](#_topic_import_columns_from_excel) [160](#_topic_import_columns_from_excel)

[import\_columns\_from\_table](#_topic_import_columns_from_table) [160](#_topic_import_columns_from_table)

[import\_columns\_from\_dialogcontrol](#_topic_import_columns_from_dialogcontro) [160](#_topic_import_columns_from_dialogcontro)

[import\_from\_dialogcontrol](#_topic_import_from_dialogcontrol) [160](#_topic_import_from_dialogcontrol)

[export\_to\_dialogcontrol](#_topic_export_to_dialogcontrol) [161](#_topic_export_to_dialogcontrol)

[export\_to\_dialogcontrol\_by\_filter](#_topic_export_to_dialogcontrol_by_filte) [161](#_topic_export_to_dialogcontrol_by_filte)

[export\_to\_dialogcontrol\_by\_regex\_filter](#_topic_export_to_dialogcontrol_by_regex) [161](#_topic_export_to_dialogcontrol_by_regex)

[export\_to\_dialogcontrol\_by\_extended\_filter](#_topic_export_to_dialogcontrol_by_exten) [161](#_topic_export_to_dialogcontrol_by_exten)

[export\_to\_dialogcontrol\_by\_extended\_filter\_table](#_topic_export_to_dialogcontrol_by_exte1) [162](#_topic_export_to_dialogcontrol_by_exte1)

[resetofffromexport](#_topic_resetofffromexport) [162](#_topic_resetofffromexport)

[export\_to\_LABPP\_Table](#_topic_export_to_LABPP_Table) [163](#_topic_export_to_LABPP_Table)

[set\_columns\_width\_when\_export](#_topic_set_columns_width_when_export) [164](#_topic_set_columns_width_when_export)

[mapping\_columns\_to\_export](#_topic_mapping_columns_to_export) [164](#_topic_mapping_columns_to_export)

[get\_base\_colindex\_by\_mapped\_column](#_topic_get_base_colindex_by_mapped_colu) [165](#_topic_get_base_colindex_by_mapped_colu)

[get\_count\_of\_mapped\_columns](#_topic_get_count_of_mapped_columns) [166](#_topic_get_count_of_mapped_columns)

[export\_to\_csv](#_topic_export_to_csv) [166](#_topic_export_to_csv)

[import\_from\_csv](#_topic_import_from_csv) [167](#_topic_import_from_csv)

[export\_to\_ac\_object\_parameter](#_topic_export_to_ac_object_parameter) [167](#_topic_export_to_ac_object_parameter)

[set\_column\_user\_value](#_topic_set_column_user_value) [168](#_topic_set_column_user_value)

[get\_column\_user\_value](#_topic_get_column_user_value) [168](#_topic_get_column_user_value)

[set\_column\_user\_genesis](#_topic_set_column_user_genesis) [168](#_topic_set_column_user_genesis)

[get\_column\_user\_genesis](#_topic_get_column_user_genesis) [168](#_topic_get_column_user_genesis)

[delete\_rows](#_topic_delete_rows) [169](#_topic_delete_rows)

[set\_value\_of\_toallrows](#_topic_set_value_of_toallrows) [169](#_topic_set_value_of_toallrows)

[Функции стандартной библиотеки](#_topic_Newtopic18) [169](#_topic_Newtopic18)

[Преобразование](#_topic_Newtopic21) [169](#_topic_Newtopic21)

[atoi](#_topic_atoi) [169](#_topic_atoi)

[itoa](#_topic_itoa) [169](#_topic_itoa)

[atof](#_topic_atof) [169](#_topic_atof)

[ctos](#_topic_ctos) [169](#_topic_ctos)

[grad\_to\_radian](#_topic_grad_to_radian) [170](#_topic_grad_to_radian)

[radian\_to\_grad](#_topic_radian_to_grad) [170](#_topic_radian_to_grad)

[sprintf](#_topic_sprintf) [170](#_topic_sprintf)

[ecvt\_french](#_topic_ecvt_french) [170](#_topic_ecvt_french)

[ecvt](#_topic_ecvt) [171](#_topic_ecvt)

[Математические функции](#_topic_Newtopic22) [171](#_topic_Newtopic22)

[abs](#_topic_abs) [171](#_topic_abs)

[max](#_topic_max) [171](#_topic_max)

[min](#_topic_min) [171](#_topic_min)

[rand](#_topic_rand) [171](#_topic_rand)

[ln](#_topic_ln) [171](#_topic_ln)

[log](#_topic_log) [171](#_topic_log)

[sqrt](#_topic_sqrt) [172](#_topic_sqrt)

[sqr](#_topic_sqr) [172](#_topic_sqr)

[pow](#_topic_pow) [172](#_topic_pow)

[percent](#_topic_percent) [172](#_topic_percent)

[tsround](#_topic_tsround) [172](#_topic_tsround)

[tsround\_best](#_topic_tsround_best) [172](#_topic_tsround_best)

[floor](#_topic_floor) [172](#_topic_floor)

[ceil](#_topic_ceil) [172](#_topic_ceil)

[math\_const](#_topic_math_const) [172](#_topic_math_const)

[inrange](#_topic_inrange) [173](#_topic_inrange)

[Тригонометрические функции](#_topic_Newtopic24) [173](#_topic_Newtopic24)

[cos](#_topic_cos) [173](#_topic_cos)

[sin](#_topic_sin) [173](#_topic_sin)

[arcsin](#_topic_arcsin) [173](#_topic_arcsin)

[arccos](#_topic_arccos) [173](#_topic_arccos)

[tg](#_topic_tg) [173](#_topic_tg)

[arctg](#_topic_arctg) [173](#_topic_arctg)

[ctg](#_topic_ctg) [173](#_topic_ctg)

[arcctg](#_topic_arcctg) [173](#_topic_arcctg)

[Функции геометрического преобразования](#_topic_Newtopic25)  [174](#_topic_Newtopic25)

[is\_curelem\_inside\_element\_polygon](#_topic_is_curelem_inside_element_polygo) [174](#_topic_is_curelem_inside_element_polygo)

[is\_point\_on\_element\_polygon](#_topic_is_point_on_element_polygon) [174](#_topic_is_point_on_element_polygon)

[rotate\_point\_and\_move](#_topic_rotate_point_and_move) [175](#_topic_rotate_point_and_move)

[get\_cross\_point\_of\_2lines](#_topic_get_cross_point_of_2lines) [175](#_topic_get_cross_point_of_2lines)

[get\_rot\_and\_move\_point](#_topic_get_rot_and_move_point) [175](#_topic_get_rot_and_move_point)

[get\_length\_2point](#_topic_get_length_2point) [175](#_topic_get_length_2point)

[is\_point\_on\_line](#_topic_is_point_on_line) [176](#_topic_is_point_on_line)

[get\_line\_angle](#_topic_get_line_angle) [176](#_topic_get_line_angle)

[Специальные функции](#_topic_Newtopic20) [176](#_topic_Newtopic20)

[codemeter - измерение времени выполнения фрагмента кода](#_topic_codemeter_)  [176](#_topic_codemeter_)

[ac\_save\_messages\_to\_file - сохранить текст из окна сообщений в файл](#_topic_ac_save_messages_to_file_)  [176](#_topic_ac_save_messages_to_file_)

[Связь с квартирографией](#_topic_Newtopic23) [177](#_topic_Newtopic23)

[solaris\_test](#_topic_solaris_test) [177](#_topic_solaris_test)

[get\_flat\_rooms](#_topic_get_flat_rooms) [177](#_topic_get_flat_rooms)

[runtimecontrol](#_topic_runtimecontrol) [177](#_topic_runtimecontrol)

[workline](#_topic_workline) [177](#_topic_workline)

[Команды для управления в приложении LabPP\_Calc](#_topic_LabPP_Calc) [179](#_topic_LabPP_Calc)

[interface](#_topic_interface) [179](#_topic_interface)

[calc\_field](#_topic_calc_field) [179](#_topic_calc_field)

[get\_active](#_topic_get_active) [179](#_topic_get_active)

[Внешние переменные](#_topic_Newtopic26) [180](#_topic_Newtopic26)

[var\_extern\_set](#_topic_var_extern_set) [180](#_topic_var_extern_set)

[var\_extern\_get](#_topic_var_extern_get) [180](#_topic_var_extern_get)

[var\_extern\_delete](#_topic_var_extern_delete) [180](#_topic_var_extern_delete)

[Объект ts\_idispatcher](#_topic_ts_idispatcher) [181](#_topic_ts_idispatcher)

[CreateInstance](#_topic_CreateInstance) [183](#_topic_CreateInstance)

[attach](#_topic_attach1) [183](#_topic_attach1)

[detach](#_topic_detach1) [183](#_topic_detach1)

[AutoWrap](#_topic_AutoWrap) [184](#_topic_AutoWrap)

[Управление выполнением и поведением программы](#_topic_Newtopic40) [184](#_topic_Newtopic40)

[Прерывание выполнения длительных процессов ac\_process\_window](#_topic_ac_process_window) [184](#_topic_ac_process_window)

[Init](#_topic_Init1) [185](#_topic_Init1)

[SetPhase](#_topic_SetPhase) [185](#_topic_SetPhase)

[IsCanceled](#_topic_IsCanceled) [185](#_topic_IsCanceled)

[Close](#_topic_Close1) [185](#_topic_Close1)

[throw\_process](#_topic_throw_process) [186](#_topic_throw_process)

[Поведение программы при ошибках считывания/записи переменных элементов](#_topic_Newtopic38) [187](#_topic_Newtopic38)

[Остановка процесса помещением файла в каталоге](#_topic_Newtopic41) [188](#_topic_Newtopic41)

[Объект ts\_variant](#_topic_ts_variant) [189](#_topic_ts_variant)

[get\_pdispVal](#_topic_get_pdispVal) [189](#_topic_get_pdispVal)

[get\_value\_simple](#_topic_get_value_simple) [189](#_topic_get_value_simple)

[set\_value\_simple](#_topic_set_value_simple) [189](#_topic_set_value_simple)

[run\_cpp - запуск другой программы](#_topic_run_cpp_) [190](#_topic_run_cpp_)

[run\_from\_file](#_topic_run_from_file) [190](#_topic_run_from_file)

[run\_from\_variable](#_topic_run_from_variable) [190](#_topic_run_from_variable)

[Получение аргументов внутри программы](#_topic_Newtopic27) [190](#_topic_Newtopic27)

[Обработка ошибок](#_topic_Newtopic16) [191](#_topic_Newtopic16)

[Коды ошибок](#_topic_Newtopic28) [191](#_topic_Newtopic28)

[Функции интерфейса LabPP\_Automat](#_topic_LabPP_Automat1) [191](#_topic_LabPP_Automat1)

[create\_iconbutton](#_topic_create_iconbutton) [191](#_topic_create_iconbutton)

[create\_button](#_topic_create_button) [192](#_topic_create_button)

[set\_palette\_size\_and\_message\_place](#_topic_set_palette_size_and_message_pla) [192](#_topic_set_palette_size_and_message_pla)

[Диалоги на основе окон](#_topic_Newtopic29) [192](#_topic_Newtopic29)

[Класс ts\_dialog](#_topic_ts_dialog) [192](#_topic_ts_dialog)

[init\_dialog](#_topic_init_dialog) [192](#_topic_init_dialog)

[set\_as\_main\_panel](#_topic_set_as_main_panel) [192](#_topic_set_as_main_panel)

[SetClientWH](#_topic_SetClientWH) [193](#_topic_SetClientWH)

[SetTitle](#_topic_SetTitle) [193](#_topic_SetTitle)

[SendCloseRequest](#_topic_SendCloseRequest) [193](#_topic_SendCloseRequest)

[PostCloseRequest](#_topic_PostCloseRequest) [193](#_topic_PostCloseRequest)

[eventreaction](#_topic_eventreaction) [193](#_topic_eventreaction)

[Invoke](#_topic_Invoke) [193](#_topic_Invoke)

[set\_sDescr](#_topic_set_sDescr) [194](#_topic_set_sDescr)

[Класс ts\_dialogcontrol](#_topic_ts_dialogcontrol) [194](#_topic_ts_dialogcontrol)

[Button](#_topic_Button) [194](#_topic_Button)

[SetText](#_topic_SetText) [194](#_topic_SetText)

[GetText](#_topic_GetText) [194](#_topic_GetText)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle) [194](#_topic_SetFontSizecodeStyle)

[IconButton](#_topic_IconButton) [195](#_topic_IconButton)

[TextEdit](#_topic_TextEdit) [195](#_topic_TextEdit)

[SetText](#_topic_SetText1) [195](#_topic_SetText1)

[GetText](#_topic_GetText1) [195](#_topic_GetText1)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle1) [195](#_topic_SetFontSizecodeStyle1)

[RealEdit](#_topic_RealEdit) [195](#_topic_RealEdit)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle2) [195](#_topic_SetFontSizecodeStyle2)

[CheckBox](#_topic_CheckBox) [196](#_topic_CheckBox)

[SetText](#_topic_SetText2) [196](#_topic_SetText2)

[GetText](#_topic_GetText2) [196](#_topic_GetText2)

[SetCheck](#_topic_SetCheck) [196](#_topic_SetCheck)

[GetCheck](#_topic_GetCheck) [196](#_topic_GetCheck)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle5) [196](#_topic_SetFontSizecodeStyle5)

[LeftText, CenterText, RightText](#_topic_LeftTextCenterTextRightText) [197](#_topic_LeftTextCenterTextRightText)

[SetText](#_topic_SetText3) [197](#_topic_SetText3)

[GetText](#_topic_GetText3) [197](#_topic_GetText3)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle3) [197](#_topic_SetFontSizecodeStyle3)

[PopUp](#_topic_PopUp) [197](#_topic_PopUp)

[SelectItem](#_topic_SelectItem) [198](#_topic_SelectItem)

[DisableItem](#_topic_DisableItem) [198](#_topic_DisableItem)

[DeleteItem](#_topic_DeleteItem) [198](#_topic_DeleteItem)

[EnableItem](#_topic_EnableItem) [198](#_topic_EnableItem)

[InsertItem](#_topic_InsertItem) [198](#_topic_InsertItem)

[AppendItem](#_topic_AppendItem) [198](#_topic_AppendItem)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle6) [198](#_topic_SetFontSizecodeStyle6)

[GetItemIndexByText](#_topic_GetItemIndexByText) [199](#_topic_GetItemIndexByText)

[GetItemIndexByText](#_topic_GetItemIndexByText1) [199](#_topic_GetItemIndexByText1)

[RadioButton](#_topic_RadioButton) [199](#_topic_RadioButton)

[IsSelected](#_topic_IsSelected) [199](#_topic_IsSelected)

[Select](#_topic_Select) [199](#_topic_Select)

[SetText](#_topic_SetText4) [199](#_topic_SetText4)

[GetText](#_topic_GetText4) [199](#_topic_GetText4)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle4) [200](#_topic_SetFontSizecodeStyle4)

[IconRadioButton](#_topic_IconRadioButton) [200](#_topic_IconRadioButton)

[IsSelected](#_topic_IsSelected1) [200](#_topic_IsSelected1)

[Select](#_topic_Select1) [200](#_topic_Select1)

[SingleSelListBox](#_topic_SingleSelListBox) [200](#_topic_SingleSelListBox)

[RepaintBackgroundItemsByColumnValue](#_topic_RepaintBackgroundItemsByColumnVa) [200](#_topic_RepaintBackgroundItemsByColumnVa)

[SetItemBackgroundColor](#_topic_SetItemBackgroundColor) [201](#_topic_SetItemBackgroundColor)

[SetItemColor](#_topic_SetItemColor) [201](#_topic_SetItemColor)

[SelectItem](#_topic_SelectItem1) [201](#_topic_SelectItem1)

[GetTabItemText](#_topic_GetTabItemText) [201](#_topic_GetTabItemText)

[GetMouseClickedPosXY](#_topic_GetMouseClickedPosXY) [202](#_topic_GetMouseClickedPosXY)

[GetTabFieldPosition](#_topic_GetTabFieldPosition) [202](#_topic_GetTabFieldPosition)

[SetTabItemIcon](#_topic_SetTabItemIcon) [202](#_topic_SetTabItemIcon)

[SetTabItemText](#_topic_SetTabItemText) [202](#_topic_SetTabItemText)

[GetTabItemIconId](#_topic_GetTabItemIconId) [202](#_topic_GetTabItemIconId)

[SetTabItemIconId](#_topic_SetTabItemIconId) [202](#_topic_SetTabItemIconId)

[SetOnTabItem](#_topic_SetOnTabItem) [203](#_topic_SetOnTabItem)

[SetTabFieldCount](#_topic_SetTabFieldCount) [203](#_topic_SetTabFieldCount)

[SetHeaderItemSize](#_topic_SetHeaderItemSize) [203](#_topic_SetHeaderItemSize)

[SetTabFieldProperties](#_topic_SetTabFieldProperties) [203](#_topic_SetTabFieldProperties)

[SetItemHeight](#_topic_SetItemHeight) [203](#_topic_SetItemHeight)

[SetHeaderItemSizeableFlag](#_topic_SetHeaderItemSizeableFlag) [203](#_topic_SetHeaderItemSizeableFlag)

[SetHeadersSizeableFlag](#_topic_SetHeadersSizeableFlag) [203](#_topic_SetHeadersSizeableFlag)

[SetHeaderItemText](#_topic_SetHeaderItemText) [204](#_topic_SetHeaderItemText)

[SetHeaderItemSize](#_topic_SetHeaderItemSize1) [204](#_topic_SetHeaderItemSize1)

[InsertItem](#_topic_InsertItem1) [204](#_topic_InsertItem1)

[AppendItem](#_topic_AppendItem1) [204](#_topic_AppendItem1)

[DisableItem](#_topic_DisableItem1) [204](#_topic_DisableItem1)

[EnableItem](#_topic_EnableItem1) [204](#_topic_EnableItem1)

[DeleteItem](#_topic_DeleteItem1) [204](#_topic_DeleteItem1)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle8) [204](#_topic_SetFontSizecodeStyle8)

[MultiSellListBox](#_topic_MultiSellListBox) [205](#_topic_MultiSellListBox)

[SelectItem](#_topic_SelectItem2) [205](#_topic_SelectItem2)

[GetTabItemText](#_topic_GetTabItemText1) [205](#_topic_GetTabItemText1)

[GetMouseClickedPosXY](#_topic_GetMouseClickedPosXY1) [205](#_topic_GetMouseClickedPosXY1)

[GetTabFieldPosition](#_topic_GetTabFieldPosition1) [205](#_topic_GetTabFieldPosition1)

[SetTabItemIcon](#_topic_SetTabItemIcon1) [205](#_topic_SetTabItemIcon1)

[SetTabItemText](#_topic_SetTabItemText1) [205](#_topic_SetTabItemText1)

[GetTabItemIconId](#_topic_GetTabItemIconId1) [206](#_topic_GetTabItemIconId1)

[SetTabItemIconId](#_topic_SetTabItemIconId1) [206](#_topic_SetTabItemIconId1)

[SetOnTabItem](#_topic_SetOnTabItem1) [206](#_topic_SetOnTabItem1)

[SetTabFieldCount](#_topic_SetTabFieldCount1) [206](#_topic_SetTabFieldCount1)

[SetTabFieldCount](#_topic_SetTabFieldCount2) [206](#_topic_SetTabFieldCount2)

[SetHeaderItemSize](#_topic_SetHeaderItemSize2) [206](#_topic_SetHeaderItemSize2)

[SetTabFieldProperties](#_topic_SetTabFieldProperties1) [206](#_topic_SetTabFieldProperties1)

[SetItemHeight](#_topic_SetItemHeight1) [207](#_topic_SetItemHeight1)

[SetHeaderItemSizeableFlag](#_topic_SetHeaderItemSizeableFlag1) [207](#_topic_SetHeaderItemSizeableFlag1)

[SetHeadersSizeableFlag](#_topic_SetHeadersSizeableFlag1) [207](#_topic_SetHeadersSizeableFlag1)

[SetHeaderItemText](#_topic_SetHeaderItemText1) [207](#_topic_SetHeaderItemText1)

[SetHeaderItemSize](#_topic_SetHeaderItemSize3) [207](#_topic_SetHeaderItemSize3)

[InsertItem](#_topic_InsertItem2) [207](#_topic_InsertItem2)

[AppendItem](#_topic_AppendItem2) [208](#_topic_AppendItem2)

[DisableItem](#_topic_DisableItem2) [208](#_topic_DisableItem2)

[EnableItem](#_topic_EnableItem2) [208](#_topic_EnableItem2)

[DeleteItem](#_topic_DeleteItem2) [208](#_topic_DeleteItem2)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle9) [208](#_topic_SetFontSizecodeStyle9)

[MultiLineEdit](#_topic_MultiLineEdit) [208](#_topic_MultiLineEdit)

[SetText](#_topic_SetText5) [209](#_topic_SetText5)

[GetText](#_topic_GetText5) [209](#_topic_GetText5)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle16) [209](#_topic_SetFontSizecodeStyle16)

[IconItem](#_topic_IconItem) [209](#_topic_IconItem)

[IconCheckBox](#_topic_IconCheckBox) [209](#_topic_IconCheckBox)

[SetCheck](#_topic_SetCheck1) [209](#_topic_SetCheck1)

[GetCheck](#_topic_GetCheck1) [209](#_topic_GetCheck1)

[PushCheck](#_topic_PushCheck) [210](#_topic_PushCheck)

[SetText](#_topic_SetText6) [210](#_topic_SetText6)

[GetText](#_topic_GetText6) [210](#_topic_GetText6)

[SetCheck](#_topic_SetCheck3) [210](#_topic_SetCheck3)

[GetCheck](#_topic_GetCheck2) [210](#_topic_GetCheck2)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle10) [210](#_topic_SetFontSizecodeStyle10)

[IconPushCheck](#_topic_IconPushCheck) [210](#_topic_IconPushCheck)

[SetCheck](#_topic_SetCheck2) [210](#_topic_SetCheck2)

[GetCheck](#_topic_GetCheck3) [211](#_topic_GetCheck3)

[IconPushRadio](#_topic_IconPushRadio) [211](#_topic_IconPushRadio)

[SingleSelTreeView](#_topic_SingleSelTreeView) [211](#_topic_SingleSelTreeView)

[SelectItem](#_topic_SelectItem3) [211](#_topic_SelectItem3)

[TreeViewInsertItem](#_topic_TreeViewInsertItem) [211](#_topic_TreeViewInsertItem)

[DeleteItem](#_topic_DeleteItem3) [211](#_topic_DeleteItem3)

[InsertItem](#_topic_InsertItem3) [212](#_topic_InsertItem3)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle13) [212](#_topic_SetFontSizecodeStyle13)

[GetTVItemIconId](#_topic_GetTVItemIconId1) [212](#_topic_GetTVItemIconId1)

[SetTVItemIconId](#_topic_SetTVItemIconId) [212](#_topic_SetTVItemIconId)

[GetItemCount](#_topic_GetItemCount) [212](#_topic_GetItemCount)

[import\_classificator](#_topic_import_classificator) [213](#_topic_import_classificator)

[MultiSelTreeView](#_topic_MultiSelTreeView) [213](#_topic_MultiSelTreeView)

[SelectItem](#_topic_SelectItem4) [213](#_topic_SelectItem4)

[TreeViewInsertItem](#_topic_TreeViewInsertItem1) [214](#_topic_TreeViewInsertItem1)

[DeleteItem](#_topic_DeleteItem4) [214](#_topic_DeleteItem4)

[InsertItem](#_topic_InsertItem4) [214](#_topic_InsertItem4)

[AppendItem](#_topic_AppendItem3) [214](#_topic_AppendItem3)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle14) [214](#_topic_SetFontSizecodeStyle14)

[GetTVItemIconId](#_topic_GetTVItemIconId) [214](#_topic_GetTVItemIconId)

[SetTVItemIconId](#_topic_SetTVItemIconId1) [215](#_topic_SetTVItemIconId1)

[GetItemCount](#_topic_GetItemCount1) [215](#_topic_GetItemCount1)

[SingleSelListView](#_topic_SingleSelListView) [215](#_topic_SingleSelListView)

[DeleteItem](#_topic_DeleteItem5) [215](#_topic_DeleteItem5)

[InsertItem](#_topic_InsertItem5) [215](#_topic_InsertItem5)

[AppendItem](#_topic_AppendItem4) [215](#_topic_AppendItem4)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle11) [215](#_topic_SetFontSizecodeStyle11)

[GetItemCount](#_topic_GetItemCount2) [216](#_topic_GetItemCount2)

[MultiSelListView](#_topic_MultiSelListView) [216](#_topic_MultiSelListView)

[DeleteItem](#_topic_DeleteItem6) [216](#_topic_DeleteItem6)

[InsertItem](#_topic_InsertItem6) [216](#_topic_InsertItem6)

[AppendItem](#_topic_AppendItem5) [216](#_topic_AppendItem5)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle12) [216](#_topic_SetFontSizecodeStyle12)

[GetItemCount](#_topic_GetItemCount3) [216](#_topic_GetItemCount3)

[Splitter](#_topic_Splitter) [216](#_topic_Splitter)

[Attach\_controlled\_element](#_topic_Attach_controlled_element) [216](#_topic_Attach_controlled_element)

[ProgressBar](#_topic_ProgressBar) [217](#_topic_ProgressBar)

[SetMin](#_topic_SetMin) [217](#_topic_SetMin)

[SetMax](#_topic_SetMax) [217](#_topic_SetMax)

[SetValue](#_topic_SetValue) [217](#_topic_SetValue)

[GetMin](#_topic_GetMin) [217](#_topic_GetMin)

[GetMax](#_topic_GetMax) [217](#_topic_GetMax)

[GetValue](#_topic_GetValue) [217](#_topic_GetValue)

[NormalTab](#_topic_NormalTab) [217](#_topic_NormalTab)

[SelectItem](#_topic_SelectItem5) [217](#_topic_SelectItem5)

[DisableItem](#_topic_DisableItem3) [217](#_topic_DisableItem3)

[EnableItem](#_topic_EnableItem3) [218](#_topic_EnableItem3)

[DeleteItem](#_topic_DeleteItem7) [218](#_topic_DeleteItem7)

[InsertItem](#_topic_InsertItem7) [218](#_topic_InsertItem7)

[AppendItem](#_topic_AppendItem6) [218](#_topic_AppendItem6)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle7) [218](#_topic_SetFontSizecodeStyle7)

[TabPage](#_topic_TabPage) [218](#_topic_TabPage)

[SetToolTip](#_topic_SetToolTip) [218](#_topic_SetToolTip)

[GroupBox](#_topic_GroupBox) [219](#_topic_GroupBox)

[SetFontSizecodeStyle](#_topic_SetFontSizecodeStyle15) [219](#_topic_SetFontSizecodeStyle15)

[PushMenuCheck](#_topic_PushMenuCheck) [219](#_topic_PushMenuCheck)

[SetItemIcon](#_topic_SetItemIcon) [222](#_topic_SetItemIcon)

[GetCheck](#_topic_GetCheck4) [222](#_topic_GetCheck4)

[SetCheck](#_topic_SetCheck4) [222](#_topic_SetCheck4)

[SetItemCount](#_topic_SetItemCount) [222](#_topic_SetItemCount)

[SelectItem](#_topic_SelectItem6) [222](#_topic_SelectItem6)

[GetSelectedItem](#_topic_GetSelectedItem) [222](#_topic_GetSelectedItem)

[SetToolTip](#_topic_SetToolTip1) [222](#_topic_SetToolTip1)

[SetToolTipItem](#_topic_SetToolTipItem) [223](#_topic_SetToolTipItem)

[WasStateChange](#_topic_WasStateChange) [223](#_topic_WasStateChange)

[PreviousMenuItem](#_topic_PreviousMenuItem) [223](#_topic_PreviousMenuItem)

[PushMenuRadio](#_topic_PushMenuRadio) [223](#_topic_PushMenuRadio)

[SetItemIcon](#_topic_SetItemIcon1) [223](#_topic_SetItemIcon1)

[Select](#_topic_Select2) [224](#_topic_Select2)

[IsSelected](#_topic_IsSelected2) [224](#_topic_IsSelected2)

[SetItemCount](#_topic_SetItemCount1) [224](#_topic_SetItemCount1)

[SelectItem](#_topic_SelectItem7) [224](#_topic_SelectItem7)

[GetSelectedItem](#_topic_GetSelectedItem1) [224](#_topic_GetSelectedItem1)

[SetToolTip](#_topic_SetToolTip2) [224](#_topic_SetToolTip2)

[SetToolTipItem](#_topic_SetToolTipItem1) [224](#_topic_SetToolTipItem1)

[PreviousMenuItem](#_topic_PreviousMenuItem1) [224](#_topic_PreviousMenuItem1)

[WasSelectionChange](#_topic_WasSelectionChange) [224](#_topic_WasSelectionChange)

[Eventreaction](#_topic_Eventreaction1) [224](#_topic_Eventreaction1)

[SetAnchorToPanelResize](#_topic_SetAnchorToPanelResize) [227](#_topic_SetAnchorToPanelResize)

[Hide](#_topic_Hide) [227](#_topic_Hide)

[Show](#_topic_Show) [227](#_topic_Show)

[SetPosition](#_topic_SetPosition) [227](#_topic_SetPosition)

[Работа со структурами данных в формате JSON](#_topic_JSON) [227](#_topic_JSON)

[Класс ts\_json](#_topic_ts_json) [227](#_topic_ts_json)

[init (clear)](#_topic_initclear) [227](#_topic_initclear)

[DecodeFromURL](#_topic_DecodeFromURL) [228](#_topic_DecodeFromURL)

[DecodeFromString](#_topic_DecodeFromString) [228](#_topic_DecodeFromString)

[EncodeToStr](#_topic_EncodeToStr) [229](#_topic_EncodeToStr)

[Класс ts\_array](#_topic_ts_array) [229](#_topic_ts_array)

[Класс ts\_class](#_topic_ts_class) [229](#_topic_ts_class)

[Класс ts\_num](#_topic_ts_num) [229](#_topic_ts_num)

[Класс ts\_bool](#_topic_ts_bool) [229](#_topic_ts_bool)

[Класс ts\_string](#_topic_ts_string) [229](#_topic_ts_string)

[ReadFromURL](#_topic_ReadFromURL) [229](#_topic_ReadFromURL)

[TSVariator](#_topic_TSVariator) [229](#_topic_TSVariator)

[Редактор таблиц LABPP\_Table для ARCHICAD](#_topic_LABPP_TableARCHICAD) [231](#_topic_LABPP_TableARCHICAD)

[Структура данных объекта LABPP\_Table (для программистов)](#_topic_LABPP_Table) [235](#_topic_LABPP_Table)

[Непереносимые пробелы и дефисы](#_topic_Newtopic35) [236](#_topic_Newtopic35)

[Работа с СУБД MySQL](#_topic_MySQL) [237](#_topic_MySQL)

[connection\_init](#_topic_connection_init) [237](#_topic_connection_init)

[connection\_open](#_topic_connection_open) [238](#_topic_connection_open)

[real\_query](#_topic_real_query) [238](#_topic_real_query)

[store\_result](#_topic_store_result) [239](#_topic_store_result)

[get\_num\_fields](#_topic_get_num_fields) [239](#_topic_get_num_fields)

[get\_row\_count](#_topic_get_row_count) [239](#_topic_get_row_count)

[export\_to\_table](#_topic_export_to_table) [239](#_topic_export_to_table)

[free\_result](#_topic_free_result) [240](#_topic_free_result)

[connection\_close](#_topic_connection_close) [240](#_topic_connection_close)

[Пример - получение таблицы c сайта из базы данных MySQL в диалог ARCHICAD](#_topic__cMySQLARCHICAD) [240](#_topic__cMySQLARCHICAD)

[Работа с СУБД PostgreSQL](#_topic_PostgreSQL) [243](#_topic_PostgreSQL)

[init](#_topic_init2) [243](#_topic_init2)

[pqconnectdb](#_topic_pqconnectdb) [244](#_topic_pqconnectdb)

[pqexec](#_topic_pqexec) [244](#_topic_pqexec)

[pqnfields](#_topic_pqnfields) [244](#_topic_pqnfields)

[pqntuples - кол-во строк](#_topic_pqntuples__) [245](#_topic_pqntuples__)

[export\_to\_table](#_topic_export_to_table1) [245](#_topic_export_to_table1)

[pqclear](#_topic_pqclear) [245](#_topic_pqclear)

[pqfinish](#_topic_pqfinish) [246](#_topic_pqfinish)

[Пример - получение таблицы c сайта из базы данных PostgreSQL в диалог ARCHICAD](#_topic__cPostgreSQLARCHICAD) [246](#_topic__cPostgreSQLARCHICAD)

[Работа с СУБД SQLite](#_topic_SQLite) [249](#_topic_SQLite)

[init](#_topic_init3) [249](#_topic_init3)

[open\_db](#_topic_open_db) [249](#_topic_open_db)

[prepare\_v2\_query - запрос к базе данных](#_topic_prepare_v2_query_) [249](#_topic_prepare_v2_query_)

[column\_count](#_topic_column_count) [250](#_topic_column_count)

[row\_count](#_topic_row_count) [250](#_topic_row_count)

[export\_to\_table](#_topic_export_to_table2) [250](#_topic_export_to_table2)

[finalize\_stmt](#_topic_finalize_stmt) [251](#_topic_finalize_stmt)

[close\_db](#_topic_close_db) [251](#_topic_close_db)

[Пример - получение таблицы c сайта из базы данных PostgreSQL в диалог ARCHICAD](#_topic__cPostgreSQLARCHICAD1) [251](#_topic__cPostgreSQLARCHICAD1)

[Искусственный интеллект](#_topic_Newtopic43) [254](#_topic_Newtopic43)

[Нейросеть на основе перцептрона Розенблатта - ts\_aipercrozen](#_topic__ts_aipercrozen) [254](#_topic__ts_aipercrozen)

[init](#_topic_init4) [254](#_topic_init4)

[task\_add\_to\_list](#_topic_task_add_to_list) [255](#_topic_task_add_to_list)

[learn\_task\_list](#_topic_learn_task_list) [256](#_topic_learn_task_list)

[run\_task\_list](#_topic_run_task_list) [256](#_topic_run_task_list)

[get\_task\_result\_to\_table](#_topic_get_task_result_to_table) [256](#_topic_get_task_result_to_table)

[task\_list\_clear](#_topic_task_list_clear) [256](#_topic_task_list_clear)

[store\_matrices и restore\_matrices](#_topic_store_matricesrestore_matrices) [257](#_topic_store_matricesrestore_matrices)

[clear](#_topic_clear1) [257](#_topic_clear1)

[ts\_aipercrozen\_matrices](#_topic_ts_aipercrozen_matrices) [257](#_topic_ts_aipercrozen_matrices)

[init](#_topic_init5) [257](#_topic_init5)

[get\_from\_percrozen](#_topic_get_from_percrozen) [257](#_topic_get_from_percrozen)

[set\_to\_percrozen](#_topic_set_to_percrozen) [257](#_topic_set_to_percrozen)

[export\_to\_table](#_topic_export_to_table3) [258](#_topic_export_to_table3)

[import\_from\_table](#_topic_import_from_table) [258](#_topic_import_from_table)

[clear](#_topic_clear2) [259](#_topic_clear2)

[Пример](#_topic_Newtopic44) [259](#_topic_Newtopic44)

[BIM Master](#_topic_BIMMaster) [260](#_topic_BIMMaster)

[breakpoint](#_topic_breakpoint) [262](#_topic_breakpoint)

[programcodegen](#_topic_programcodegen) [262](#_topic_programcodegen)

[get\_bimmaster\_cur\_selected\_parameter\_info](#_topic_get_bimmaster_cur_selected_param) [262](#_topic_get_bimmaster_cur_selected_param)

[get\_bimmaster\_cur\_element\_guidastext](#_topic_get_bimmaster_cur_element_guidas) [263](#_topic_get_bimmaster_cur_element_guidas)

[show\_bimmaster\_as\_codegen](#_topic_show_bimmaster_as_codegen) [263](#_topic_show_bimmaster_as_codegen)

[show\_bimmaster](#_topic_show_bimmaster) [263](#_topic_show_bimmaster)

[Уроки](#_topic_Newtopic32) [264](#_topic_Newtopic32)

[Вычислить середину высоты элемента Морф относительно 0 проекта и записать в поле LabPP\_Calc](#_topic_0LabPP_Calc) [264](#_topic_0LabPP_Calc)

[Заполнить таблицу с расчетом плитки по элементам Штриховка и записать общую площадь в поле LabPP\_Calc](#_topic_LabPP_Calc1) [266](#_topic_LabPP_Calc1)

[FAQ (часто задаваемые вопросы)](#_topic_FAQ) [268](#_topic_FAQ)

[Как посмотреть значения параметров, заданных формулами](#_topic_Newtopic39) [268](#_topic_Newtopic39)

**Введение**



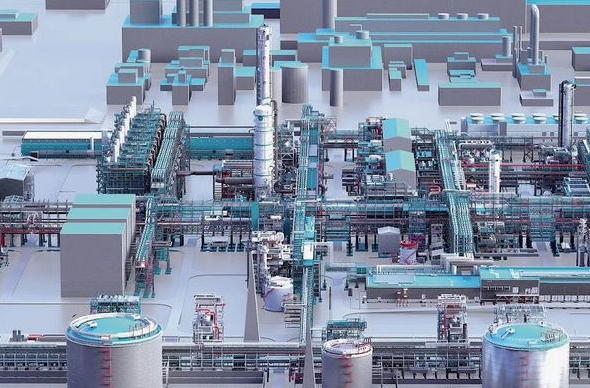
Вы можете легко создавать собственные текстовые программы для автоматизации различных проектных работ в ARCHICAD – расчеты, геометрия, интерфейс, отчеты, взаимодействие с электронными таблицами, манипуляция параметрами и свойствами элементов и проекта, данными квартирографии и многое другое.

Высокая скорость выполнения и читабельность удобны новичкам и профессионалам. Несколькими инструкциями можно выполнять как простые прямые действия, так и создавать крупные гибкие приложения с развернутыми диалогами на основе множества плавающих окон со списками, кнопками и т.п. создавая формы и даже текст самих программ прямо по ходу выполнения.

Вы сможете задействовать гораздо более мощные скрытые возможности, недоступные обычным пользователям ARCHICAD.

Реализация проекта началась в 2007 году.

Сегодня эффективность LABPP подтверждена на крупнейших стройках страны - от небоскребов Москва-Сити АБ Скуратова, до промышленных гигантов СИБУР.

Это руководство в других форматах:

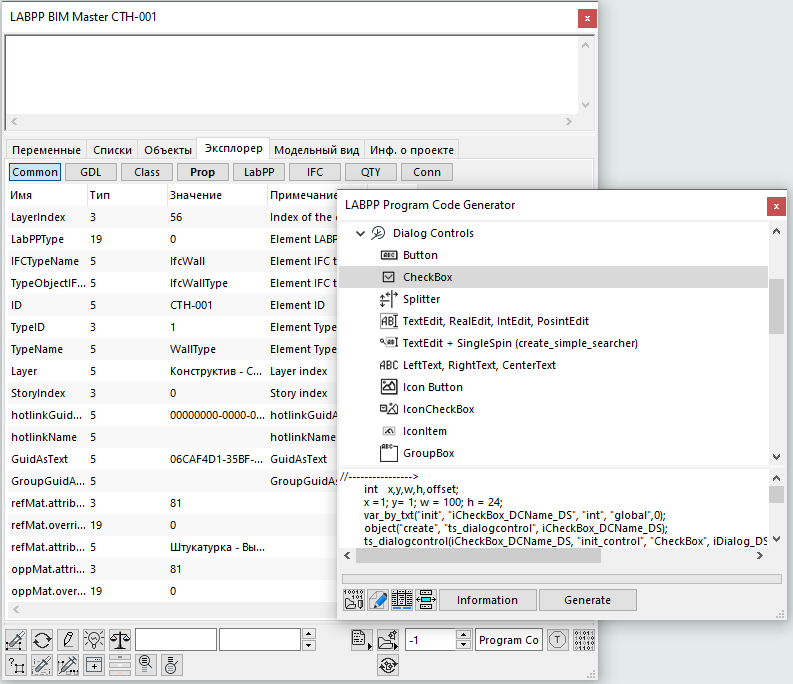
* [Adobe Acrobar PDF](https://www.labpp.net/TSDownload/LABPP_Automat%20ARCHICAD.pdf)
* [Windows стандартный help-файл CHM](https://www.labpp.net/TSDownload/LabPP_Automat.chm)
* [Word DOC](https://www.labpp.net/TSDownload/LABPP_Automat%20ARCHICAD.docx)
* [Электронная книга в формате EPUB](https://www.labpp.net/TSDownload/LABPP_Automat%20ARCHICAD.epub)
* [Для интернет браузеров HTML](https://www.labpp.net/TSHelp/LabPP_Automat/RUS/LabPP_Automat.html)

Подключение ARCHICAD к различным внешним SQL базам данных выполняется на "раз-два".Просто используйте в тексте программ соответствующие команды, описанные в этом руководстве.

Для разработки и отладки программ имеется встроенный многооконный инструмент BIM Master.

Встроенный интеллектуальный генератор кода позволяет быстро компоновать крупные фрагменты программ на основе свойств модели.

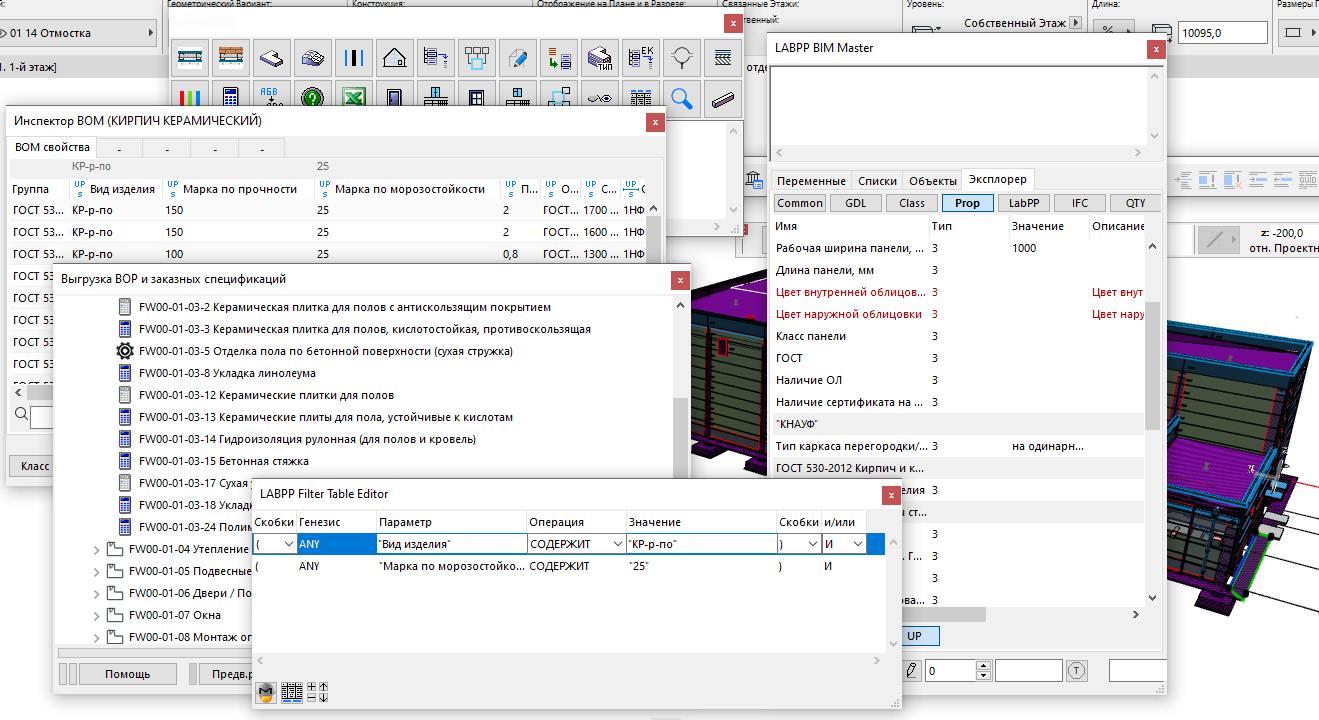


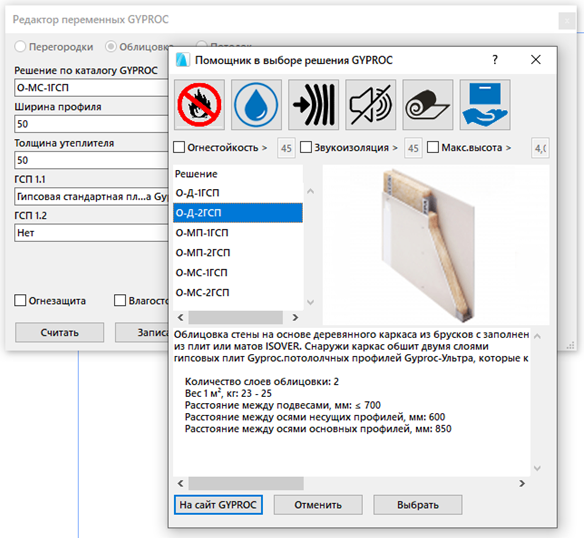


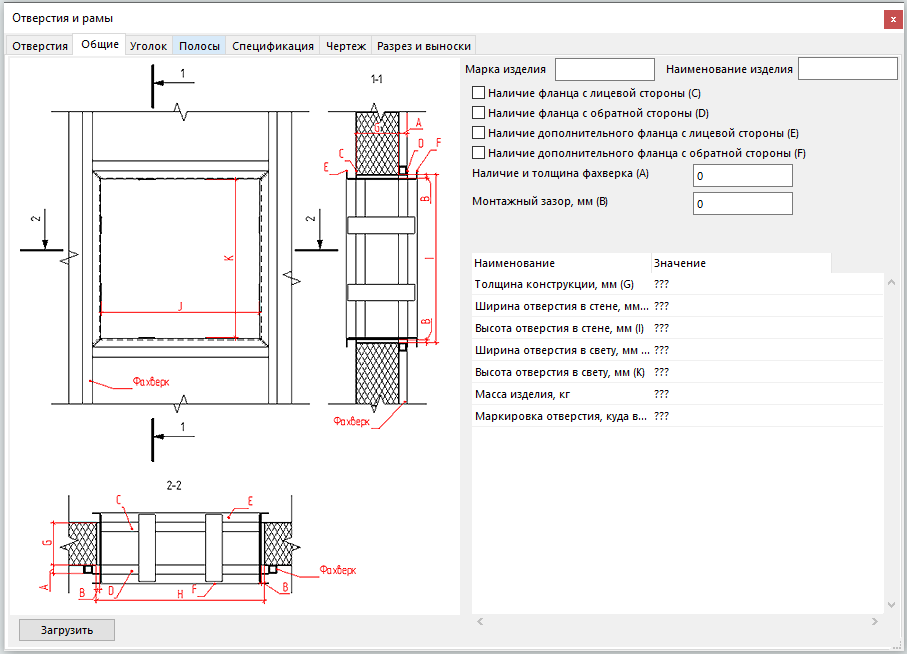
Прямо в язык программирования встроены элементы искусственного интеллекта (ИИ).

Например вы можете создавать, обучать и сохранять результаты обучения (весовые коэффициенты) на базе многослойного перцептрона Розенблатта с обратным распределением ошибки - наиболее удачная реализация нейроной сети.

Примеры диалогов в некоторых приложениях, созданных на базе LabPP\_Automat:







**Особенности языка**

В качестве основы языка программирования в LabPP\_Automat используется широко известный, простой и гибкий язык Си++.  
Достоинствами этого языка являются:

* высокая скорость выполнения;
* хорошая читаемость, лаконичность и понятность кода;
* простота создания и обращения к функциям;
* большая библиотека разнообразных процедур;
* его освоение намного легче и он более удобный для работы.

Для большего удобства использования конструкций языка в LabPP\_Automat добавлен тип данных string.  
Если Вы задали функцию MyStringFunc, которая возвращает строковое значение, то можно писать так:

string MyString = MyStringFunc();

Здесь мы сразу объявляем переменную типа "строка" MyString и сразу же записываем в нее значение из функции MyStringFunc().

Имеются ограничения при помощи которых создание программ значительно упрощено:  
  
1. можно использовать только встроенные классы через дескрипторы.  
2. отсутствуют указатели.  
3. отсутствует тип данных "ссылка".   
  
Программные модули создаются в виде отдельных текстовых файлов.

**Моя первая программа**

Создайте файл и вставьте в него следующее.

int main()

{

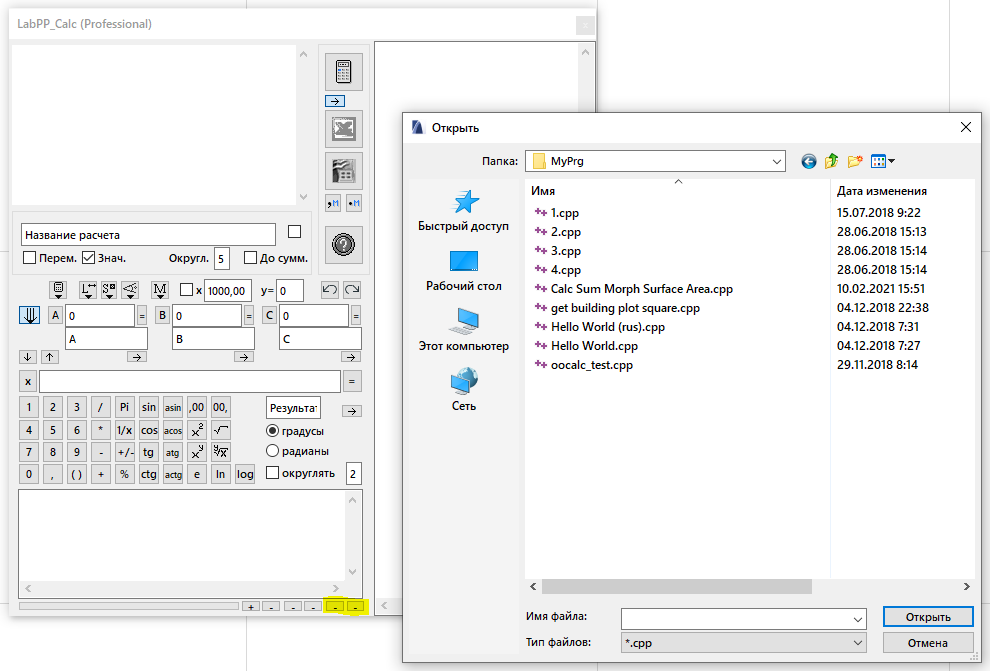
cout << "Здравствуй Мир!)))";

}

В программе labpp выберите и запустите этот файл на выполнение.

Приветствие появится в окне сообщений.

На картинке ниже - способ запуска программных модулей из приложения [LabPP\_Calc](http://www.labpp.ru/labpp-calcru)



В левом нижнем улгу окна приложения - кнопки быстрого запуска программных модулей.

Они имеются в любом приложении LabPP.

Самая правая кнопка открывает окно сообщений. Чуть левее - позволяет выбрать программный файл для выполнения.

Еще 4 кнопки слева - для быстрого повторного запуска уже выполнявшейся программы и для заранее запрограммированного выполнения программы с определенным именем файла (задается в конфигурационном файле config.cpp).

**Кодировка у файлов программных модулей**

Обычно нет необходимости разбираться с кодировками файлов.

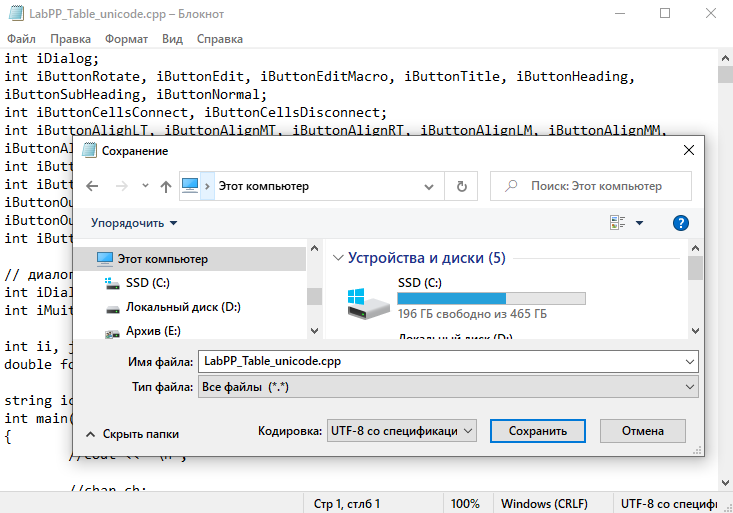
Но понимать это просто.

Файлы содержат коды символов. Одни и те же символы могут иметь разные коды в файлах с разными кодировками.

Если Вы не используете специальные символы, например, обозначение степени пишете как "М2" то кодировка программного файла может быть ANSI.

Если же необходимо писать что-то вроде "Площадь, м²", т.е., степень как в Excel в верхнем регистре, то необходимо использовать формат файлов в кодировке UTF8 со спецификацией.

Ниже на картинке показано как сохранять файлы для выполнения в LabPP\_Automat в кодировке UTF-8 со спецификацией.



**Структура каталогов у LabPP\_Automat**

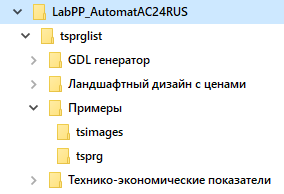
Тексты программных модулей могут использоваться самостоятельно или в составе конфигураций для LabPP\_Automat.

Каждая конфигурация для LabPP\_Automat представляет собой каталог с определенной структурой.

Именем конфигурации служит название этого каталога.

Каталоги конфигураций располагаются в каталоге tsprglist.

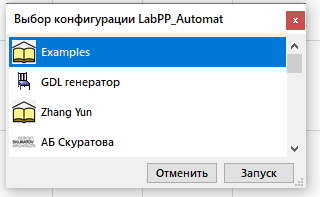
Ниже показан пример содержимого каталога tspgrlist.



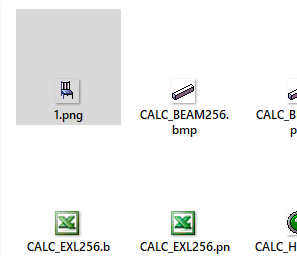
"GDL генератор", "Ландшафтный дизайн с ценами", "Примеры" и "Технико-экономические показатели" - это конфигурации.

При запуске панели LabPP\_Automat предлагает список этих конфигураций для выбора и загрузки для работы в данный момент.

Внутри каждой конфигурации имеется обязательные каталоги.

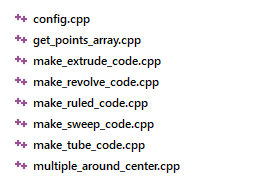


tsimages - каталог с картинками и иконками. Файл с именем 1.png будет использован в качестве иконки данной конфигурации в стартовом списке.



В каталоге tsprg располагается конфигурационный файл config.cpp и программные модули для запуска в данной конфигурации.

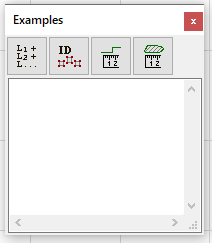
Например:



Файл config.cpp отвечает за вид рабочей панели конфигурации.

В нем задаются размеры и положение окна конфигурации и описание кнопок с присвоением им имен запускаемых программных модулей (файлы \*.cpp).

Ниже на рисунке - вид окна одной из конфигураций.



В каталоге LabPP\_AutomatAC... располагается программный модуль LabPP\_Automat.apx и ключевой файл LabPP\_Automat.key.

Ключевой файл при покупке программы для коммерческого использования заменяется.

Чтобы создать собственную конфигурацию можно просто скопировать имеющийся каталог какой-либо конфигурации и модифицировать по своему усмотрению.

**Конфигурационный файл config.cpp у конфигурации**

Файл config.cpp является обычным программным модулем.

Он так же может содержать разнообразные инструкции, сообщения и запросы.

В простейшем случае - это просто последовательный набор инструкций, задающих размеры главного окна и положение в нем окна сообщений и описывающих кнопки.

create\_button

Создать кнопку с текстовой надписью на панели диалога конфигурации.

Вызов:

ac\_request("create\_button", string button\_caption, int start\_x, int start\_y, int width, int height, string tooltip, string name\_cpp);

или

ac\_request("create\_button", string button\_caption, int start\_x, int start\_y, int width, int height, string tooltip, string name\_cpp, int iarg, double darg, string sarg);

Здесь:

button\_caption - надпись на кнопке;

start\_x, start\_y - координаты левого верхнего угла кнопки на панели диалога;

width, height - ширина и высота кнопки;

tooltip - всплывающая подсказка, которая будет появляться при наведении мыши на кнопку;

name\_cpp - имя файла с программой (из каталога tsprg).

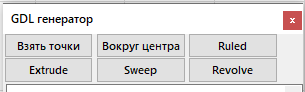
iarg, darg и sarg - целое число, число с плавающей точкой и строка - параметры, которые будут переданы в программный модуль при запуске по этой кнопке. Получить параметры в вызываемом модуле можно командой run\_cpp("get\_args"... :

int iArg1;

double dArg2;

string sArg3;

run\_cpp("get\_args", iArg1, dArg2, sArg3);



create\_iconbutton

Создать кнопку с иконкой на панели диалога конфигурации.

Вызов:

ac\_request("create\_iconbutton", string icon\_filename, int start\_x, int start\_y, int width, int height, string tooltip, string name\_cpp);

или

ac\_request("create\_iconbutton", string icon\_filename, int start\_x, int start\_y, int width, int height, string tooltip, string name\_cpp, int iarg, double darg, string sarg);

Здесь:

icon\_filename - имя файла иконки, которая должна быть отрисована на кнопке (из каталога tsimages);

start\_x, start\_y - координаты левого верхнего угла кнопки на панели диалога;

width, height - ширина и высота кнопки;

tooltip - всплывающая подсказка, которая будет появляться при наведении мыши на кнопку;

name\_cpp - имя файла с программой (из каталога tsprg).

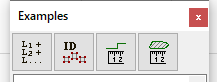
iarg, darg и sarg - целое число, число с плавающей точкой и строка - параметры, которые будут переданы в программный модуль при запуске по этой кнопке. Получить параметры в вызываемом модуле можно командой run\_cpp("get\_args"... :

int iArg1;

double dArg2;

string sArg3;

run\_cpp("get\_args", iArg1, dArg2, sArg3);



Руководство по LABPP\_Automat для ARCHICAD

Задать размеры главной панели конфигурации и положение в ней окна сообщений.

ac\_request("set\_palette\_size\_and\_message\_place",int start\_x, int start\_y, int width, int height, int messageplace\_start\_x, int messageplace\_y, int messageplace\_width, int messageplace\_height);

Здесь:

start\_x, int start\_y - левый верхний угол панели на экране;

width, height - ширина и высота панели при старте;

messageplace\_start\_x, int messageplace\_y - положение левого верхнего угла окна сообщений в панели;

messageplace\_width, messageplace\_height - ширина и высота окна сообщений.

**Списки для выборки элементов из проекта**

Для работы с элементами проекта удобно делать выборки.

Выборки делаются при помощи команд группы "load\_elements\_list".

Выборки можно делать с учетом типа элементов, слоям, этажам, классам и веткам классов, значениям параметров и т.п.

Затем, обычно, запрашиваем количество элементов в выборке и организуем цикл для обработки элементов.

При обращении к этим командам надо использовать номер списка элементов для выборки.

По умолчанию списков 10, поэтому этот номер может быть от 0 до 9.

Десяти списков, по опыту, более чем достаточно, чтобы обеспечить нужды алгоритма в программном модуле.

Но если этого недостаточно, то командой "set\_TSMAX\_ELEMLIST\_NUMBER" можно установить любое количество этих списков.

Считать текущее заданное количество списков можно командой "get\_TSMAX\_ELEMLIST\_NUMBER"

.

**Генезис свойств**

Генезис - признак параметра, отвечающий за его происхождение.

|  |  |  |
| --- | --- | --- |
| Текстовое значение | Числовое значение | Описание |
| "ANY" | 0 | Любой (для таблиц и т.п.) |
| "GDL" | 1 | Параметр GDL элемента |
| "EP" | 2 | Параметр элемента |
| "UP" | 3 | Пользовательский параметр |
| "UPb" | 4 | Базовый параметр |
| "UPub" | 5 | Пользовательский базовый параметр |
| "UPa" | 6 | Параметр |
| "Cls" | 7 | Класс |
| "IFC" | 8 | Параметр IFC |
| "UTS" | 9 | Параметр LABPP прикрепленный к элементу |
| "Q" | 10 | Quantity - количественный параметр |
| "AP" | 11 | Свойство атрибута |
| "MV" | 12 | Настройки модельного вида |
| "MV\_GDL" | 13 | Настройки GDL из модельного вида |
| "PInfo" | 14 | Информация о проекте |

**Окна LABPP**

Для приложений на LABPP для ARCHICAD можно создавать многооконный интерфейс.

Окна могут быть модальными и немодальными.

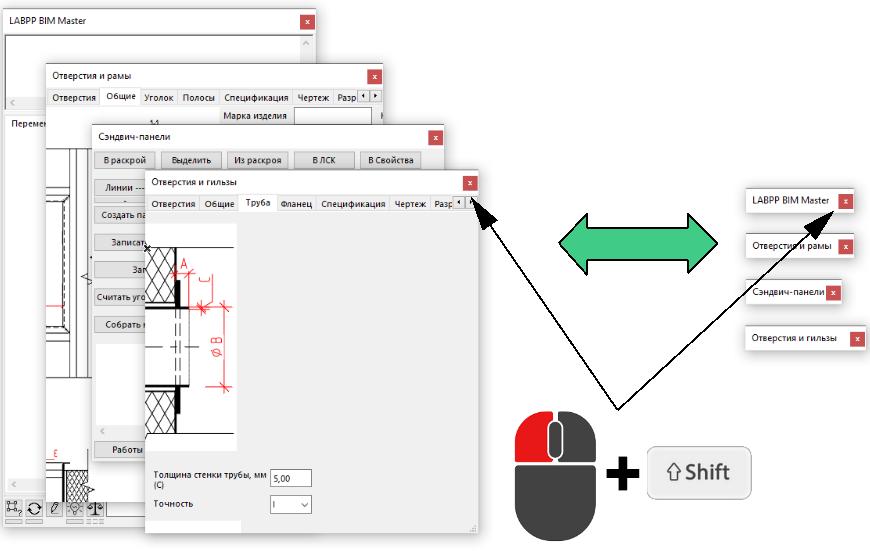
Диалоги, построенные на модальных окнах фокусируют работу программы только на текущем окне.

При этом все остальные действия блокируются до закрытия модального диалога.

А немодальные окна - позволяют работать с проектом и с другими окнами.  
Таких окон может быть открыто любое количество.

Их можно перетаскивать за пределы окна ARCHICAD и на другой монитор.

Если удерживать нажатой клавишу SHIFT и кликать мышкой по кнопке закрытия окна, то они будут сворачиваться и разворачиваться.



**Семантика языка C++ для ARCHICAD**

**Общая структура программы**

Программа должна содержать функцию main(), с которой начинается ее выполнение.

Можно создавать подпрограммы. Объявлять переменные.

Переменные могут быть глобальными и локальными.

Общая структура программы показана ниже

int my\_global\_var1 = 1; // объявление переменной типа целое число и присвоение ей первоначального значения.

double my\_global\_var2; // объявление переменной вещественного типа

Пример программы.

int main()

{

int my\_local\_var = 2; // объявление переменной типа целое число

my\_local\_var = my\_func(); // вызов функции my\_func() с присвоением результата переменной

}

// объявление собственной функции, возвращающей целое число

int my\_func()

{

int my\_local\_var = 3; // объявление переменной типа целое число внутри функции

return 10; // значение, которое вернет функция

}

**Функция main()**

Каждый программный файл должен содержать базовую функцию main().

Эта функция вызывается при старте программы. Как и все функции C++ ее содержимое заключается в фигурные скобки.

Общая структура

int main()

{

// тело функции заключено в фигурные скобки

return 0;

}

Чтобы была возможность получить данные извне программы имеются два варианта - получение аргументов, с которыми запущена программа (run\_cpp("get\_args",...)) и получение значений внешних переменных, инициированных другими программами (var\_extern\_get).

**Идентификаторы**

Идентификаторы - это названия переменных и функций. Названия могут быть только на английском языке. Прописные и строчные буквы не различаются. Так, что переменная int iValue и int IVALUE это одно и то же.

**Типы переменных, видимость и виды**

Имеется несколько способов объявления и использования переменных.

|  |  |  |
| --- | --- | --- |
| Вид переменных | Пример | Видимость |
| Локальные переменные | int function()  {  int i;  double d;  string s;  } | В рамках текущей функции |
| Глобальные переменные | int i;  double d;  string s;  int function()  {  } | В рамках текущего исполнительного файла cpp |
| Номерованные внутренние списки элементов (по умолчанию - от № 0 до № 9) | Собрать элементы зон во внутренний список элементов labpp № 7 и в цикле перебрать их для получения номера помещения и отображения их в окне сообщений.  ac\_request("load\_elements\_list\_from\_selection", 7,  "ZoneType","MainFilter",2+1024 ,"Classif",  "Классификация ARCHICAD, "ClassifVal", "Помещение");  ac\_request("get\_loaded\_elements\_list\_count", 7);  int icount = ac\_getnumvalue();  ac\_request("get\_loaded\_elements\_list\_count", 7);  icount = ac\_getnumvalue();  cout << "icount=" << icount << "\n";  if (icount == 0)  {  cout << "Нет зон в проекте\n";  return -1;  }  int i;  int ires;  string szonename;  for (i = 0; i < icount; i++)  {  ires = ac\_request("set\_current\_element\_from\_list",  7, i);  ires = ac\_request("get\_element\_value", "ZoneName");  szonename = ac\_getstrvalue();  coutvar << szonename;  } | В рамках текущего исполнительного файла cpp |
| Объекты | int iTable;  object("create","ts\_table", iTable);  object("KeepInMemory", iTable); | В рамках текущего сеанса приложения.  Можно использовать для передачи данных из одного файла cpp к другому.  Значение дескриптора (iTable) можно запомнить в одном файле и считать в другом через внешнюю переменную. |
| Внешние переменные | var\_extern\_set("sWorkBookUprFile", "c:\\filename.txt");  string sWorkBookUprFile;  int ires = var\_extern\_get("sWorkBookUprFile", sWorkBookUprFile, 0);  if (ires != 0)  {  cout << "Не задана внешняя переменная sWorkBookUprFile";  return -1;  } | В рамках текущего сеанса приложения.  Можно использовать для передачи данных из одного файла cpp к другому. |

В программе можно объявлять и использовать переменные со следующим типом данных:

bool - логическая (значения 0-ложь/1-истина).

int - целое число.

double - вечественное число (с плавающей точкой).

string - строковая переменная.

Пример.

Объявить переменную типа строка и записать в нее несколько слов.

string mystring;

mystring = "fragment1" + "fragment2";

mystring += "fragment3";

**Пакетное объявление и обращение к переменным через идентификаторы, задаваемые текстом - var\_by\_txt()**

Вы можете создавать большое количество переменных с именами, которые формируются как текстовые строки.

Например в цикле создать переменные от "var\_1\_1\_1" до "var\_100\_100\_100".

Удобно для формирования n-мерных массивов и т.п.

Далее можно обращаться к переменным напрямую:

s = var\_67\_45\_34;

Или присваивать

var\_34\_23\_23 = "значение для 34\_23\_23";

А так же можно присваивать значения переменной, имя которой формируется на лету в строке:

for (i = 1; i <= 100; i++)

{

for (j = 1; j <= 100; j++)

{

for (k = 1; k <= 100; k++)

{

var\_by\_txt("set", "var\_"+itoa(i)+"\_"+itoa(j)+"\_"+ itoa(k),

"значение для переменной " + itoa(i) + "\_" + itoa(j) + "\_" + itoa(k));

}

}

}

**init**

Объявление переменных с идентификаторами, формируемыми в тексте программы.

Вызов:

var\_by\_txt("init", string string\_with\_varnames, string vartype, string local\_or\_global, void preset\_value);

Здесь:

string\_with\_varnames - строка, содержащая один или список идентификаторов переменных через запятую ("var1" или "var1,var2,var3").

vartype - тип объявляемой переменной - "string", "int", "double", "bool".

local\_or\_global - в каком уровне видимости создавать переменные - "global" или "local".

preset\_value - значение, присваиваемое переменным по умолчанию.

Глобальные переменные можно создавать в цикле.

Локальные переменные в цикле или в других случаях с применением фигурных скобок, создавать нельзя, т.к. после выхода из понижающей области {} они удаляются.   
Это как бы мы написали:

if(...)

{

// объявляем локальную переменную k в пониженной области видимости

int k = 1;

}

// здесь, за пределами фигурных скобок уже локальная переменная k не существует.

Чтобы выполнить массовое объявление локальных переменных - сначала сформируйте строку с идентификаторами через запятую, а затем выполните функцию var\_by\_txt("init"...).

Повторное объявление глобальных переменных может привести к ошибке.

Чтобы избежать этого используйте var\_by\_txt("is\_exist",...).

Пример.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Создание переменных с идентификаторами из текстовой строки

// функция var\_by\_txt()

// LABPP 2021

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main()

{

test\_global\_variables();

test\_local\_variables();

}

// Сформировать имя для переменной

string make\_varname(string base, int ii, int jj)

{

return base + "\_" + itoa(ii) + "\_" + itoa(jj)

}

int test\_global\_variables()

{

int i,j;

string basename = "glob\_var";

cout << "---------------------------\n";

cout << "test\_global\_variables\n";

// если ранее не создана переменная с именем "glob\_var\_0\_0", то создать все переменные от "glob\_var\_0\_0" до "glob\_var\_9\_9"

if (var\_by\_txt("is\_exist",make\_varname(basename,0,0))==false)

{

for (i = 0; i < 10; i++)

{

for (j = 0; j < 10; j++)

{

cout << "объявляем переменную " << make\_varname(basename,i,j) << "\n";

var\_by\_txt("init", make\_varname(basename, i, j), "string", "global", "glob value " + itoa(i) + "\_" + itoa(j));

}

}

}

string s = glob\_var\_7\_7;

coutvar << s;

glob\_var\_7\_7 = "new value for 7\_7";

cout << "Все значения созданных глобальных переменных:\n";

string varname;

for (i = 0; i < 10; i++)

{

for (j = 0; j < 10; j++)

{

varname = make\_varname(basename, i, j);

cout << varname + "="<< var\_by\_txt("get", varname) << "\n";

}

}

return 0;

}

// тест для локальных переменных

// в отличии от глобальных - локальные переменные должны создаваться в теле функции не глубже других фигурных скобок

// { }

// иначе это понижает область видимости.

// Поэтому для локальных переменных сначала формируем строку с именами переменных через запятую.

// Такой прием возможен и для глобальных переменных.

int test\_local\_variables()

{

int i, j;

string basename = "local\_var";

cout << "---------------------------\n";

cout << "test\_local\_variables\n";

string string\_with\_varnames;

cout << "формируем список имен переменных в строку через запятую\n";

for (i = 0; i < 10; i++)

{

for (j = 0; j < 10; j++)

{

string\_with\_varnames += make\_varname(basename, i, j) + ",";

}

}

coutvar << string\_with\_varnames;

// создаем переменные от "local\_var\_0\_0" до "local\_var\_9\_9", инициируем пустой строкой (можно другим значением)

var\_by\_txt("init", string\_with\_varnames, "string", "local", "");

// присвоение значений

for (i = 0; i < 10; i++)

{

for (j = 0; j < 10; j++)

{

var\_by\_txt("set", make\_varname(basename, i, j), "local value " + itoa(i) + "\_" + itoa(j));

}

}

// прямое обращение к переменной

string s = local\_var\_7\_7;

coutvar << s;

local\_var\_7\_7 = "new value for 7\_7";

cout << "Все значения созданных локальных переменных:\n";

string varname;

for (i = 0; i < 10; i++)

{

for (j = 0; j < 10; j++)

{

varname = make\_varname(basename, i, j);

cout << varname + "=" << var\_by\_txt("get", varname) << "\n";

}

}

return 0;

}

**is\_exist**

Проверить существует ли переменная с указанным идентификатором.

bool var\_by\_txt("is\_exist", string varname);

Здесь varname - имя переменной, существование которой проверяем.

**set**

Задать значение переменной с идентификатором, формируемым при помощи текстовой строки.

Вызов:

var\_by\_txt("set", string varname, void varvalue);

Здесь:

varname - имя переменной (например "var\_0\_10\_343\_34").

varvalue - значение для присвоения переменной.

**get**

Получить значение переменной по идентификатору, формируемому в текстовой строке.

Вызов:

string s = var\_by\_txt("get", string varname);

double d = var\_by\_txt("get", string varname);

int i = var\_by\_txt("get", string varname);

bool b = var\_by\_txt("get", string varname);

Здесь varname - идентификатор переменной, заданный текстом ("var\_320\_ini" и т.п.).

К переменным, созданным массовым тиражом можно обращаться и по идентификатору:

s = var\_320\_ini;

**Комментарии**

Комментарий до конца строки - // комментарий

После // до конца строки выражение считается комментарием.

Пример:

int i = 0; // здесь мы объявляем переменную - целое число и присваиваем ему значение 0

Комментарий на нескольких строках /\* комментарий \*/

Весть текст между обозначениями /\* и \*/ независимо от количества строк считается комментарием.

Пример:

/\*Если нужно написать длинное пояснение на нескольких строках в тексте программы

или закомментировать фрагмент программы,

то удобно сделать так\*/

**Организация циклов**

**Цикл for**

Классическая форма организации цикла для C++.

Можно прервать выполнение цикла в любом месте директивой break;

**Примечание**. Фигурные скобки обязательны. Объявление переменной цикла внутри конструкции for(... ) не допускается.

Пример.

Написать в окне сообщений фразу "Здравствуй Мир!" 10 раз.

int i;

for (i = 0; i < 10; i++)

{

cout << "Здравствуй Мир!";

}

**Цикл do-while**

int i;

i = 0;

do {

cout << i;        // Вывести в окно сообщений LabPP\_Automat значение i

i++;                // Увеличить i на 1

} while (i < 10)   // Выполнять цикл покуда i меньше 10

**Цикл while**

int i;

i = 0;

while (i < 10) {

cout << i;

i++;

}

**Логические операции**

==   Равенство     Истинно, если число или строка слева равно числу или строке справа  
!=    Неравенство     Истинно, если число или строка слева НЕ равно числу или строке справа   
<     Меньше     Истинно, если число слева меньше числа справа  
<=   Меньше или равно     Истинно, если число слева меньше или равно числу справа  
>     Больше     Истинно, если число слева больше числа справа  
>=   Больше или равно     Истинно, если число слева больше или равно числу справа  
&&  Логическое И. Если выражение слева и справа истинны, то результат - истинно. Если хотя бы одно из выражений слева или справа ложно, то результат - ложно  
||     Логическое ИЛИ. Если хотя бы одно выражение слева и справа истинны, то результат - истина. Если оба выражения ложны, то результат - ложно.

**Условные переходы**

**Оператор if**

if (i < 0)

{

cout << "да";

}

**Конструкция if-else**

if (i < 0) // Если выражение в скобках истинно, то выполняется первый фрагмент

{

cout << "да";

}

else       // Если выражение в скобках ложно, то выполняется второй фрагмент

{

cout << "нет";

}

**Конструкция if-else if-else**

if (i < 0) // Если выражение в скобках истинно, то выполняется первый фрагмент

{

cout << "i < 0";

}

else if (i == 0) // Если выражение в скобках истинно, то выполняется второй фрагмент

{

cout << "i = 0";

}

else  // Если ни одно из выражений не истинно (пунктов else if может быть много)

{

cout << "i > 0";

}

**Оператор switch**

switch (i)

{

case 1:

cout << "i=1";

break;

case 2:

cout << "i=2";

break;

case 10:

cout << "i=10";

break;

default:

// операторы, выполняемые при любом другом значении i

}

**Создание собственных функций (подпрограмм)**

Чтобы легче было понимать и корректировать текст программ можно ее фрагменты выделять в отдельные подпрограммы.

Так же поступают и с повторяющимися фрагментами, содержащими ощутимое количество строк текста программы.

Такие фрагменты записывают ниже функции main() после ее последней фигурной скобки.

Им дают название, оснащают входящими аргументами и возвращаемым значением.

Пример.

Вывести в окно сообщений текст, в зависимости от кода, передаваемого в подпрограмму.

int main()

{

cout << "Строка из функции с арг. 1= " << get\_string(1) << ", то же но с арг. 2 = " << get\_string(2);

}

// объявление функции, принимает аргумент arg - целое число.

// возвращает строковое значение.

string get\_string(int arg)

{

string sresult;

if (arg == 1) // если arg равно 1

{

sresult = "строка 1"; // то присвоить переменной sresult это значение.

}

else if (arg == 2) // если же arg равно 2

{

sresult = "строка 2";

}

else // если же ни 1 ни 2, то при любом другом значении

{

sresult = "строка N"; // присвоить переменной sresult это значение.

}

return sresult; // вернуть в качестве значения функции содержимое переменной sresult.

}

В подпрограмму могут передаваться сразу несколько переменных (аргументов). Тогда они пишутся через запятую.

Пример.

Объявить функцию, которая получает в качестве аргументов два целых и одно число с плавающей точкой, а возвращает значение с плавающей точкой.

double my\_func(int a, int b, double c)

{

return 10;

}

Благодаря тому, что текст С++ имеет высокую скорость выполнения, можно создавать любое количество собственных функций.

**Директива #include**

Чтобы использовать один и тот же текст или подпрограммы в разных программных файлах удобно выносить текст программ в заголовочные файлы. Обычно на языке C++ таким файлам дают расширение .h или .hpp.

Пример.

#include "my\_heading.hpp"

В этом месте программы будет вставлен код, находящийся в файле my\_heading.hpp.

В отличии от обычного программного файла, код в подключаемом файле не должен содержать функцию main()

Если по каким-то причинам файл не будет найден во время выполнения - то будет сообщение об ошибке.

Чтобы подключить файл из каталога, который лежит выше от того, в котором находится текущий исполняемый файл cpp можно указывать относительный путь:

#include "..\\my\_heading.hpp"

**Особая конструкция cpp для ARCHICAD - управляемый #include**

Чтобы иметь возможность управлять содержимым программы в зависимости от параметров проекта можно вставлять особую конструкцию #include с автотекстом внутри.

#include <?PInfo Имя пользовательской переменной в информации о проекте?>  
или  
#include <?PInfo Имя пользовательской переменной в информации о проекте|альтернативное имя файла.ext?>

Здесь:

PInfo - генезис переменной, откуда следует взять имя файла для #include (PInfo - информация о проекте).

Имя пользовательской переменной в информации о проекте = буквально название переменной из информации о проекте, которая содержит имя файла, который следует подключить директивой #include.

альтернативное имя файла.ext = имя файла, которое препроцессору следует использовать если переменная в проекте отсутствует.

**Директива #pragma region**

#pragma region Создание диалога

// здесь код создания диалога, который может сворачиваться редактором текста

#pragma endregion

**Поиск по шаблону (коды REGEX)**

Вы можете организовать поиск в таблицах или выбор элементов из модели по числовым или текстовым данным.

Текст можно сравнивать по точному совпадению, частично или по встречающимся фрагментам, удовлетворяющим условию.

Удобно, например, когда надо выбрать строки со словом, например, "стена" в любом месте текста.

Тогда для выборки мы зададим строку ".\*стена.\*".

Здесь .\* означают клюбой символ повторенный сколько угодно раз. Т.е. сколько угодно до фрагмента "стена"и сколько угодно после.

Но вероятно где-то есть слово с большой буквы.

Тогда мы пишем ".\*[Сс]тена.\*". Или даже кто-то написал букву С из английской раскладки.

Можем это учесть - ".\*[CcСс]тена.\*".

В коде можно указать символ ^ - начало текста и $ - конец текста.

Кодировка REGEX дает широчайшие возможности для гибкой выборки в различных случаях обработки текстовой информации при выборках и т.п.

Например это можно использовать в выборке элементов по названию класса, слоя, ID и т.п.

Пример

Если пустая строка или есть русские буквы или есть символ # - то истина

"(^(?![\\w\\W]))|(?:.\*[а-я].\*)|(?:.\*[#].\*)"

Пример

Проверить на наличие текста "подстрока1" или "подстрока2" в любом месте текста:

"(?:.\*подстрока1.\*)|(?:.\*подстрока2.\*)"

Пример2

Выбрать внутренние стены в список №1, у которых есть строительный материал не тоньше 250мм, с именем, которое содержит:

...Кирпич ... керамич...

...кирпич ... керамич...

или

...керамич...Кирпич...

...керамич...кирпич...

ac\_request\_special("load\_elements\_list", 1, "WallType", 2,

"", "Cls", "Классификация АРХИКАД", "ВВЕТВИ", "Стена из кирпича", "", "И",

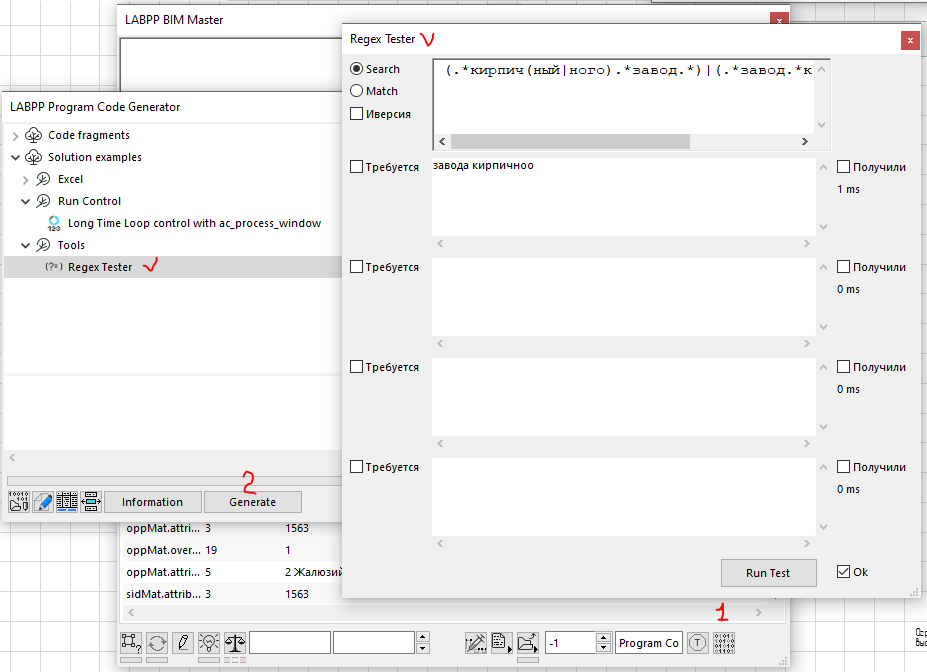
"", "UPub", "Расположение", "=", "Внутри", "", "И",

"", "EP", "buildingMaterialName\_thickness(\"(?:.\*[Кк]ирпич.\*керамич.\*)|(?:.\*керамич.\*[Кк]ирпич.\*)\")", ">=", 0.250, "", "");

Для тестирования выражений используйте специальную панель, которая идет в комплекте с программами LABPP.

Она запускается через BIM Master.

Панель BIM Master вызывается через (Ctrl+Shift+F9).



**Логическое И и логическое ИЛИ в регулярных выражениях**

Символ | между скобками или текстом означает ИЛИ.

Отсутствие этого символа означает И.

Например если надо отобрать текст со словосочетаниями

*кирпичный завод*

*кирпичного завода*

*завода кирпичного*

*завод кирпичный*

*около завода кирпичного стоит другой кирпичный завод*

Подойдет такой шаблон:

(?=.\*кирпич(ный|ного).\*завод.\*)|(?=.\*завод.\*кирпич(ный|ного).\*)

Между скобками стоит знак | - это означает что подойдет, если сработает ИЛИ то ИЛИ это.

Но если знак | убрать - это будет означать, что мы хотим чтобы было И то И это.

Т.е. выражение:

(?=.\*кирпич(ный|ного).\*завод.\*)(?=.\*завод.\*кирпич(ный|ного).\*)

отберет нам только текст

*около завода кирпичного стоит другой кирпичный завод*

Это потому, что в тексте одновременно встречаются оба фрагмента - И тот И этот.

Логическое отрицание - знак !

Если написать так:

(?=.\*кирпич(ный|ного).\*завод.\*)(?=.\*завод.\*кирпич(ный|ного).\*)(?!.\*силикат.\*)

То будет отброшен текст

*около завода кирпичного стоит другой кирпичный завод силикатного кирпича*

**Функции для работы с оболочкой shell\_func**

**get\_path**

Получить каталог или путь к файлу.

Вызов:

int res = shell\_func("get\_path", string what, string result);

Переменная what задает тип возвращаемого пути:

"rootconfig" - каталог, где находится список конфигураций (tsprglist) LabPP\_Automat;

"tsimages" - каталог, где находятся иконки конфигурации LabPP\_Automat;

"tsprg" - каталог, где находятся программы текущей конфигурации LabPP\_Automat;

"lib" - каталог, где находятся библиотечные элементы текущей конфигурации LabPP\_Automat;

"rootaddons" - каталог, где находится сам аддонс LabPP\_Automat или другое приложение, которое запустило текущий скрипт на выполнение;

"tmp" - каталог для временных файлов, по умолчанию, или заданный командой SETCFG("TMPFolder", string sTMPFolder);

"currentcpp" - каталог, где находится текущий скрипт;

"currentcppfiles" - подкаталог "files".

"currentcppfilename" - имя файла текущего скрипта (если он запущен из файла а не создан на лету из строки);

"curdir" - текущий каталог;

Результат записывается в переменную result

**set\_cur\_dir**

Вызов:

int res = shell\_func("set\_cur\_dir", string path\_or\_what);

path\_or\_what - может быть путь ("c:\\my\_dir\\") или директива:

"rootconfig" - каталог, где находится список конфигураций (tsprglist) LabPP\_Automat;

"tsimages" - каталог, где находятся иконки конфигурации LabPP\_Automat;

"tsprg" - каталог, где находятся программы текущей конфигурации LabPP\_Automat;

"lib" - каталог, где находятся библиотечные элементы текущей конфигурации LabPP\_Automat;

"tmp" - каталог для временных файлов, по умолчанию, или заданный командой SETCFG("TMPFolder", string sTMPFolder);

"rootaddons" - каталог, где находится сам аддонс LabPP\_Automat или другое приложение, которое запустило текущий скрипт на выполнение;

"currentcpp" - каталог, где находится текущий скрипт;

"currentcppfiles" - подкаталог "files".

Пример. Установить текущий каталог "C:\MyDir":

string mydir = "C:\\MyDir";

int res = shell\_func("set\_cur\_dir", mydir);

if (res == 0)

{

cout << "Рабочий каталог успешно изменен";

}

else

{

cout << "Установить каталог на " << mydir << " не удалось";

}

**shellexecute**

Выполнить операцию программной оболочки.

Обращение.

int res = shell\_func("shellexecute", "open", string spath);

Здесь:

spath - путь к программе или файлу, который надо открыть.

Можно открывать файлы, отправлять их на печать и т.д. при помощи соответствующей программы, заданной на уровне операционной системы.  
Пример. Открыть файл "Пример.xls", находящийся в каталоге текущей конфигурации LabPP\_Automat.

int res = shell\_func("set\_cur\_dir", "rootconfig");

if (res != 0)

{

return -1;

}

res = shell\_func("shellexecute", "open", "Пример.xls");

Будет открыт файл Пример.xls при помощи той программы, которая используется по умолчанию для файлов с расширением .xls на компьютере пользователя. Обычно это EXCEL.  
Так можно открывать файлы EXCEL из ARCHICAD.

**ExecCmd**

Запустить сторонний исполняемый модуль с аргументами и получить из него результаты в текстовую строку.

Вызов:

int ierr = shell\_func("ExecCmd", string exenamewithargs, string sresult);

Здесь:

exenamewithargs - имя стороннего программного модуля с аргументами.

Если в пути или имени или аргументах есть пробелы - то пишется в кавычках.

sresult - содержимое консольного вывода этого модуля.

Возвращаемое значение:

-1 - ошибка.

0 - успешное выполнение модуля;

Пример.

Выполнить сторонний программный модуль "TSChild Process.exe" с аргументами "Аргумент 1" и "Аргумент 2" и вывести в окно сообщений.

int main()

{

string sresult;

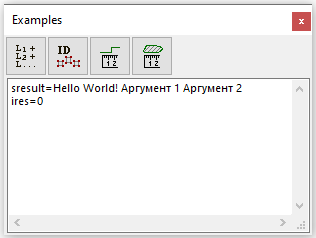
int ires = shell\_func("ExecCmd", "\"C:\\source\\TSChild Process.exe\" \"Аргумент 1\" \"Аргумент 2\"", sresult);

coutvar << sresult;

coutvar << ires;

}

Результат выполнения скрипта в окне сообщений:



Текст исполняемого модуля "TSChild Process.exe" на языке Си++

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Этот exe модуль запускается и пишет на консоли

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include <iostream>

using namespace std;

int main(int argc, char\* argv[])

{

if (argc == 2)

{

cout << "Hello World! Привет! " << argv[1] << "\n";

}

else if (argc == 3)

{

cout << "Hello World! Привет! " << argv[1] << " " << argv[2] << "\n";

}

else

{

cout << "Мало аргументов\n";

}

return 1;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**file**

**copy**

Копирует файлы.

Вызов:

int err = shell\_func("file", "copy", string spath\_from, string spath\_to);

Здесь:

spath\_from - путь к файлу, из которого копируем,

spath\_to - путь к файлу, в который копируем.

Возвращаемое значение:

0 - копирование прошло успешно;

-2222 - не найден исходный файл;

-2223 - файл уже существует;

-2224 - устройство переполнено;

-2225 - недостаточно прав пользователя.

-1 - общая ошибка.

**is\_exist**

Проверить наличие файла с заданным именем в заданном месте.

Вызов:  
int ires = shell\_func("file", "is\_exist", string spath, bool bexist);

Здесь:

spath - полный путь к файлу, который надо проверить на существование.

bexist - возвращаемое значение (1/0 - файл существует/файл не существует).

**delete**

Удалить заданный файл.

Вызов:

int ires = shell\_func("file", "delete", string spath);

Здесь:

spath - полный путь к файлу, который надо удалить.

Пример:

Удалить файл, если он существует.

string filename = "h:\\1.tmp";

ires = shell\_func("file", "is\_exist", filename, bexist);

if (bexist)

{

cout << "Удаляем файл " << filename << "\n";

ires = shell\_func("file", "delete", filename);

}

**get\_SLANG\_CODE**

Получить код языка ("RUS", "INT" и т.п.) на котором работает скрипт в данный момент .

Вызов:

shell\_func("get\_SLANG\_CODE", sLANG);

|  |  |
| --- | --- |
| Код языка | Значение |
| "CHN" | Китайский упрощенный |
| "DEU" | Немецкий |
| "FRA" | Французский |
| "INT" | Аглийский |
| "ITA" | Итальянский |
| "JPN" | Японский |
| "KOR" | Корейский |
| "POL" | Польский |
| "PRT" | Португальский |
| "RUS" | Русский |

**Управление окнами labpp**

Эти инструкции позволяют управлять немодальными (плавающими) окнами.

Вызов:

int ErrCode = labpp\_windows\_list\_dialog(string sCommand);

Здесь:

sCommand - текстовая команда.

"HideAllWindows" - скрыть все окна приложения LabPP;

"ShowAllWindows" - показать все окна приложения LabPP, которые были открыты, включая окна BIM Master'а и окна диалога формирования фильтра;

"SwitchWindowsVisible" - переключить видимость окон (если отображались, то спрятать и наоборот).

**Буффер обмена Windows**

**clipboard\_get**

Получить скопированный фрагмент текста из буффера Windows в строковую переменную.

Вызов:

ac\_request\_special("clipboard\_get", string str);

**clipboard\_set**

ac\_request\_special("clipboard\_set", str);

**Функции автоматизации – Automate**

**CloseCurWindow**

Закрыть текущее окно.

Обращение.

int iret = ac\_request("Automate",CloseCurWindow");

Возвращает 0 если окно успешно закрыто.

**Zoom**

Показать в окне область заданного размера.

Обращение.

ac\_request("Automate","Zoom",double minx, double miny, double maxx, double maxy);

Здесь:

minx, miny - координаты левого нижнего угла окна;

maxx, maxy - координаты правого верхнего угла окна.

**GoToView**

Переключает ARCHICAD на окно с указанным видом.

Обращение.

int ires = ac\_request("Automate","GoToView",int iViewLinkGuid);

Здесь:

iViewLinkGuid - дескриптор объекта ts\_guid в котором записан guid вида, в который нужно переключить окно.

Возвращает 0 при успешном переключении.

**ChangeWindow**

Изменить текущее окно.

Обращение.

int ires = ac\_request("automate","ChangeWindow", int iDatabaseDescr);

Здесь: iDatabaseDescr - дескриптор объекта ac\_database

Возвращает 0 при успешном переключении

**ShowSelectionIn3D**

Показать только выбранные элементы в окне 3D.

Если в 2d окне выбраны элементы, то по этой команде открывается окно 3d и в нем показываются только они.

Аналогично нажатию клавиши F5.

Обращение.

ac\_request("Automate","ShowSelectionIn3D");

**ShowAllIn3D**

Показать все элементы в окне 3D.

Аналогично нажатию клавиши F5 при отсутствии выбора.

Обращение.

ac\_request("Automate","ShowAllIn3D");

**ZoomToElements**

Показать отобранные в список элементы в текущем окне (2d или 3d).

Обращение.

ac\_request("Automate","ZoomToElements", int iListNumber);

или

ac\_request("Automate","ZoomToElements", int iListNumber, double doffset);

Здесь:

iListNumber - номер внутреннего списка элементов.

doffset - расстояние, на которое увеличить или уменьшить базовый фокус в окне.

**Redraw**

Перерисовать содержимое окна.

Обращение.

int res = ac\_request("Automate","Redraw");

**Rebuild**

Перестроить содержимое окна с или без перестроения всей модели.

Обращение.

int res = ac\_request("Automate","Rebuild", bool doregenerate);

Здесь:

doregenarate - 0/1 - без перестроения или с перестроением модели.

**Функции окружения - Environment**

**AddLibraries**

формат запроса:

int ires = ac\_request("Environment","AddLibraries",string LibPath);

**Story\_GoTo**

Переход на этаж по индексу

Вызов:

int ires = ac\_request("Environment","Story\_GoTo", int storyindex);

**Функции для работы с базами проекта - database**

**GetCurrent**

Получить текущую базу в объект базы данных, заданный дескриптором

Вызов:  
int err = ac\_request("database", "GetCurrent", int iDatabaseDescr);

Здесь:

err - отрицательное число (код ошибки) или 0 при успешном завершении операции;

iDatabaseDescr - дескриптор объекта базы данных

Пример.

int iDatabaseDescr;  
**object**("create", "ac\_database", iDatabaseDescr);  
int err = ac\_request("database", "GetCurrent", iDatabaseDescr);  
courvar << err;

**Select**

Показать в ARCHICAD окно базы данных, заданное дескриптором

Вызов:  
int err = ac\_request("database", "Select", int iDatabaseDescr);

Здесь:

err - отрицательное число (код ошибки) или 0 при успешном завершении операции;

iDatabaseDescr - дескриптор объекта базы данных

Пример.

Переключиться в окно 2D.

int iDatabase2d;  
**object**("create", "ac\_database", iDatabase2d);  
ac\_request("database", "SetType",iDatabase2d, 1);  
ac\_request("database", "Select", iDatabase2d);

**SetTypeID**

Задать тип базе данных, на которую указывает дескриптор.

Вызов:

ac\_request("database", "SetTypeID", int iDatabase, int itype);

Здесь:

iDatabase - дескриптор базы данных;

itype - код типа бд (окно 2d, окно 3d или пр.);

|  |  |  |
| --- | --- | --- |
| Числовое значение | Текстовое значение | Расшифровка |
| 0 | ZombieWindow |  |
| 1 | FloorPlan | 2D окно |
| 2 | Section | Сечение |
| 3 | Detail | Окно детали |
| 4 | 3DModel | 3D окно |
| 5 | Wind\_Layout | layout window |
| 6 | Drawing | Чертеж |
| 7 | MyText | custom text |
| 8 | MyDraw | custom draw |
| 9 | MasterLayout | master layout |
| 10 | Elevation | elevation window |
| 11 | InteriorElevation | interior elevation |
| 12 | Worksheet | worksheet |
| 13 | Report | report window |
| 14 | DocumentFrom3D | 3D Document window |
| 15 | External3D | External 3D |
| 16 | Movie3D | movie 3D |
| 17 | MovieRendering | movie rendering |
| 18 | Rendering | rendering window |
| 4096 | IESCommonDrawing | Interactive Schedule window |

**GetTypeID**

Получить тип базы данных, на которую указывает дескриптор.

Вызов:

ac\_request("database", "GetTypeID", int iDatabase);

Здесь:

iDatabase - дескриптор базы данных;

itype - код типа бд (окно 2d, окно 3d или пр.);

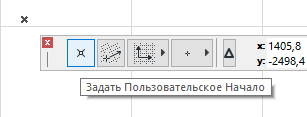
Результат считывается следующим оператором ac\_getnumvalue();

Пример.

int iDatabase;  
**object**("create", "ac\_database", iDatabase);  
ac\_request("database", "GetCurrent", iDatabase);  
ac\_request("database", "GetType", iDatabase);  
int itype = ac\_getnumvalue();  
coutvar << itype;

**GetLocOrigo**

Считать положение точки локального начала координат.



Вызов:

ac\_request("database", "GetLocOrigo", int iDatabase, double& x\_res, double& y\_res, double& z\_res);

Здесь:

iDatabase - дескриптор базы данных;

x\_res, y\_res, z\_res - полученные координаты точки локального начала координат;

Пример.

int iDatabase;  
**object**("create", "ac\_database", iDatabase);  
ac\_request("database", "GetCurrent", iDatabase);

double x\_res, y\_res, z\_res;  
ac\_request("database", "GetLocOrigo", iDatabase, x\_res, y\_res, z\_res);  
coutvar << x\_res;

coutvar << y\_res;

coutvar << z\_res;

**get\_section\_infoidtext**

Получить название сечения по его дескриптору.

Вызов:

int ires = ac\_request("database", "get\_section\_infoidtext", int iDatabase, string & sectioninfoidtextres);

Здесь:

ires - код ошибки или 0;

iDatabase - дескриптор базы данных окна;

sectioninfoidtextres - полученное название сечения;

Пример:

int iDatabase;  
**object**("create", "ac\_database", iDatabase);  
ac\_request("database", "GetCurrent", iDatabase);

string sectioninfoidtextres;  
ac\_request("database", "get\_section\_infoidtext", iDatabase, sectioninfoidtextres);  
coutvar << sectioninfoidtextres;

**get\_sectionmarkers2dpositions**

Получить координаты двух крайних точек обозначения сечения и угол линии между этими точками по дескриптору сечения.

Вызов:

int ires = ac\_request("database", "get\_sectionmarkers2dpositions", iDatabase, double& sx, double& sy, double& ex, double& ey, double& alph);

Здесь:

iDatabase - дескриптор базы данных окна сечения;

sx,sy,ex,ey - координаты крайних точек обозначения сечения в 2d окне;

alph - угол линии между крайними точками обозначения сечения в 2d окне.

Пример.

double alph;  
int ires = ac\_request("database", "get\_sectionmarkers2dpositions", idatabase, sx, sy, ex, ey, alph);

**GetRVMChanges**

Получить список всех изменений, связанных с базой данных окна.

Вызов:

int ires = ac\_request("database","GetRVMChanges", int iDatabaseDescr, int iArrayDescr);

Здесь:

ires - код ошибки или 0;

iDatabaseDescr - дескриптор базы данных;

iArrayDescr - список изменений.

Пример.

Получить общее количество изменений, связанных с текущим окном.

int iArrayDescr;  
int iDatabaseDescr;  
  
**object**("create","ts\_array",iArrayDescr);  
**object**("create","ac\_database", iDatabaseDescr);  
ac\_request("database","GetCurrent",iDatabaseDescr);  
  
ts\_array(iArrayDescr,"init","ac\_RVMChange");  
ac\_request("database","GetRVMChanges",iDatabaseDescr,iArrayDescr);  
   
int iSize;  
ts\_array(iArrayDescr,"GetSize",iSize);  
cout << "Вообще с листом связано изменений=" << iSize << "\n";  
  
**object**("delete",iArrayDescr);  
**object**("delete",iDatabaseDescr);

**GetRVMLayoutCurrentRevisionChanges**

Получить список изменений, связанных с базой данных окна, видимых на поле листа.

Вызов:

int ires = ac\_request("database","GetRVMLayoutCurrentRevisionChanges", int iDatabaseDescr, int iArrayDescr);

Здесь:

ires - код ошибки или 0;

iDatabaseDescr - дескриптор базы данных;

iArrayDescr - список изменений.

Пример.

Получить общее количество изменений, связанных с текущим окном.

int iArrayDescr;  
int iDatabaseDescr;  
  
**object**("create","ts\_array",iArrayDescr);  
**object**("create","ac\_database", iDatabaseDescr);  
ac\_request("database","GetCurrent",iDatabaseDescr);  
  
ts\_array(iArrayDescr,"init","ac\_RVMChange");  
ac\_request("database","GetRVMLayoutCurrentRevisionChanges",iDatabaseDescr,iArrayDescr);  
   
int iSize;  
ts\_array(iArrayDescr,"GetSize",iSize);  
cout << "Число изменений, видимых на листе=" << iSize << "\n";  
  
**object**("delete",iArrayDescr);  
**object**("delete",iDatabaseDescr);

**GetTypeName**

Получить название типа окна.

Вызов:

int ires = ac\_request("database","GetTypeName", int iDatabaseDescr);

Здесь:

ires - код ошибки или 0;

iDatabaseDescr - дескриптор базы данных;

Название типа окна возвращается при помощи ac\_getstrvalue();

Пример.

string stypename;  
int ires = ac\_request("database","GetTypeName", iDatabaseDescr);  
**if**(ires == 0){  
 stypename = ac\_getstrvalue();  
}

Возвращаемые значения:

"ZombieWindow"

"FloorPlan";

"Wind\_Section";

"Wind\_Detail";

"Wind\_3DModel";

"Wind\_Layout";

"Wind\_Drawing";

"Wind\_MyText";

"Wind\_MyDraw";

"Wind\_MasterLayout";

"Wind\_Elevation";

"Wind\_InteriorElevation";

"Wind\_Worksheet";

"Wind\_DocumentFrom3D";

"Wind\_IESCommonDrawing";

**GetSizeX, GetSizeY**

Получить размеры листа.

Вызов:

int ires = ac\_request("database","GetSizeX", int iDatabaseDescr);

int ires = ac\_request("database","GetSizeY", int iDatabaseDescr);

Здесь:

ires - код ошибки или 0;

iDatabaseDescr - дескриптор базы данных;

Размер возвращается при помощи ac\_getnumvalue();

Пример.

int iDatabase;  
**object**("create", "ac\_database", iDatabase);  
ac\_request("database", "GetCurrent", iDatabase);

ac\_request("database","GetSizeX", iDatabaseDescr);

double sizeX = ac\_getnumvalue();

ac\_request("database","GetSizeY", iDatabaseDescr);

double sizeY = ac\_getnumvalue();

**object**("delete", iDatabase);

**leftMargin,topMargin,rightMargin,bottomMargin**

Получить отступы на листе.

Вызов:

int ires = ac\_request("database","leftMargin", int iDatabaseDescr);

int ires = ac\_request("database","topMargin", int iDatabaseDescr);

int ires = ac\_request("database","rightMargin", int iDatabaseDescr);

int ires = ac\_request("database","bottomMargin", int iDatabaseDescr);

Здесь:

ires - код ошибки или 0;

iDatabaseDescr - дескриптор базы данных;

Размер возвращается при помощи ac\_getnumvalue();

Пример.

int iDatabase;  
**object**("create", "ac\_database", iDatabase);  
ac\_request("database", "GetCurrent", iDatabase);  
ac\_request("database","leftMargin", iDatabaseDescr);  
double leftmargin = ac\_getnumvalue();  
coutvar << leftmargin;  
**object**("delete", iDatabase);

**layoutPageNumber**

Получить число страниц на листе.

Вызов:

int ires = ac\_request("database","layoutPageNumber", int iDatabaseDescr);

Здесь:

ires - код ошибки или 0;

iDatabaseDescr - дескриптор базы данных;

Количество страниц возвращается при помощи ac\_getnumvalue();

Пример.

int iDatabase;  
**object**("create", "ac\_database", iDatabase);  
ac\_request("database", "GetCurrent", iDatabase);  
ac\_request("database","layoutPageNumber", iDatabaseDescr);  
double layoutPageNumber = ac\_getnumvalue();  
coutvar << layoutPageNumber;  
**object**("delete", iDatabase);

**layoutName**

Получить название листа.

Вызов:

int ires = ac\_request("database","layoutName", int iDatabaseDescr);

Здесь:

ires - код ошибки или 0;

iDatabaseDescr - дескриптор базы данных;

Название листа возвращается при помощи ac\_getstrvalue();

Пример.

int iDatabase;  
**object**("create", "ac\_database", iDatabase);  
ac\_request("database", "GetCurrent", iDatabase);  
ac\_request("database","layoutName", iDatabaseDescr);  
string layoutName = ac\_getstrvalue();  
coutvar << layoutName;  
**object**("delete", iDatabase);

**Вывод в окно сообщений**

**cout**

Команда cout позволяет выводить информацию в окно сообщений. Числовые данные преобразуются автоматически.

Формат команды:

cout << arg1 << arg2 << argN;

Здесь:

arg1, arg2, argN - любое количество переменных любого типа.

Пример.

Вывести в окно сообщений результат вычисления суммы площадей всех квартир из переменной dsum

cout << "Площадь всех квартир = " << dsum;

**coutvar**

Вывести информацию о значении переменной вместе с ее именем.

Удобно во время отладки программ.

Вызов:

coutvar << my\_variable;

**setcfg("COUT",...)**

Включить или выключить вывод сообщений.

Вызов:

setcfg("COUT", int onoff);

здесь onoff - 0/1 запретить/разрешить вывод сообщений.

Если задать 0, то все команды cout и coutvar будут игнорироваться до следующей команды с ключем 1.

**Файловые операции**

**open**

Открыть файл

Int ires = ts\_file(int iFileDescr, "open", string filepath, string what, string mode);

Здесь: iFileDescr - дескриптор объекта файла, полученный при создании объекта командой object("create"..., filepath - путь к файлу в файловой системе, what - что делать если файл не найден ("create" - создать новый, "fail" - выдать ошибку и остановиться, "ignore" - игнорировать и продолжить).

mode - режим открытия файла:

"r" - только для чтения

"w" - только для записи

"we" - для записи очистить файл

"rw" - для чтения и записи

"a" - добавления в конец файла

ires = ts\_file(iFileDescr, "open",filepath,"create","we");

**write**

Записать в файл

int ires = ts\_file(int iFileDescr, "write",string stowrite);

Здесь: iFileDescr - дескриптор объекта файла, stowrite - строка для записи в файл. Возвращает 0, если запись прошла успешно. Количество записанных символов можно считать при помощи функции ac\_getnumvalue();

**read**

Считать из файла

int ires = ts\_file(int iFileDescr, "read",string &storead);

Здесь: iFileDescr - дескриптор объекта файла, storead - текстовая переменная для считывания из файла. Возвращает 0, если запись прошла успешно. Количество считанных символов можно считать при помощи функции ac\_getnumvalue().

**close**

Закрыть файл

int ires = ts\_file(Int iFileDescr,"close");

Здесь: iFileDescr - дескриптор объекта файла. Возвращает 0 при успешном закрытии файла.

Пример.

Создать файл "my\_file.txt" на диске "C:" в корневом каталоге и записать в него несколько строк.

string filepath = "c:\\my\_file.txt";

int iFileDescr;

object("create", "ts\_file", iFileDescr); // создать объект типа файл в памяти

// открыть для записи чистый файл, если его нет, то создать

int ires = ts\_file(iFileDescr, "open", filepath, "create", "we");

if (ires != 0)

{

cout << "Файл не удалось открыть:" << filepath; // выдать в окно сообщений

return;

}

ires = ts\_file(iFileDescr, "write", "Первая строка\nВторая строка\n"); // записать в файл две строки

if (ires != 0)

{

cout << "Не удалось записать в файл";

return;

}

ires = ts\_file(iFileDescr, "write", "Третья строка"); // записать третью строку

ires = ts\_file(iFileDescr, "close"); // закрыть файл

object("delete", iFileDescr); // удалить объект файла из памяти

cout << "Завершение программы \n";

**SetIsUNICODE**

Вызов:

ts\_file(iFileDescr, "SetIsUNICODE");

Команда указывает что файл содержит данные в кодировке UTF-8.

**SetIsUTF16**

Вызов:

ts\_file(iFileDescr, "SetIsUTF16");

Команда указывает что файл содержит данные в кодировке UTF-16.

Пример.

Считать таблицу из файла, созданного Excel командой "Сохранить текст в Unicode" (сохраняется таблица с разделителем - табуляция, в кодировке UTF-16).

int main()

{

int iTable;

object("create", "ts\_table", iTable);

string sFullPath = "c:\\1.txt";

string csv\_source = "";

int iFileDescr;

object("create", "ts\_file", iFileDescr);

ts\_file(iFileDescr, "SetIsUTF16");

int ires = ts\_file(iFileDescr, "open", sFullPath, "ignore", "r");

if (ires == 0)

{

ts\_file(iFileDescr, "read", csv\_source);

ts\_file(iFileDescr, "close");

coutvar << csv\_source;

ts\_table(iTable, "import\_from\_csv", csv\_source, "\t");

string str;

ts\_table(iTable, "print\_to\_str", str);

coutvar << str;

}

else

{

tsalert(-1, "Error", "Can't open file " + sFullPath, "Check availability of the file", "Ok");

return -1;

}

object("delete", iTable);

object("delete", iFileDescr);

}

**Строковые функции**

**strcmp**

Сравнение двух строк.

Вызов:

bool result = strcmp(string s1, string s2);

Здесь:

string1 и string2 - строковые переменные для сравнения

Функция возвращает 1 если строки полностью совпадают

**tolower**

Перевести все символы строки в нижний регистр

Вызов:

string sresult = tolower(string svalue);

**toupper**

Перевести все символы строки в верхний регистр

Вызов:

string sresult = toupper(string svalue);

**alltrim**

Удалить с начала и с конца строки все пробелы

Вызов:

string sretult = alltrim(string svalue);

**strcontains**

Проверяет содержит ли строка указанный фрагмент.

Обращение

int ires = strcontains(string sstring, string sfragment);

Здесь:

sstring - строка, в которой нужно узнать есть ли фрагмент sfragment. Если да, то ires получит 1.

**strreplace**

Заменить в строке все фрагменты на другие фрагменты.

Вызов:

string sresult = strreplace(string s, string fragmentold, string fragmentnew, int how, int pos, int count);

Здесь:

s - строка, в которой произвести замену фрагмента fragmentold на fragmentnew,

how - откуда отступить начало - 0/1/2 - все/от начала/от конца

pos - сколько отступить

count - сколько заменять (-1 - полностью).

Пример.

Перевести число с плавающей точкой в строку и заменить точку, разделяющую целую и дробную часть на запятую.

double d\_value = 126.983;  
string s\_value = sprintf("%10.3f", d\_value);  
s\_value = strreplace(s\_value, ".", ",", 0, 0, -1);  
Результат - 126,983

**strlen**

Получить длину строки.

Вызов:

int ilen = strlen(string s);

Здесь:

s - строка

**substr**

Отрезать часть строки.

Вызов:

string substr(string source, int startpos, int count);

Здесь:

source - исходная строка, от которой требуется отрезать часть,

startpos - стартовая позиция в строке,

count - количество символов (если -1, то отрезать все до конца).

**strposcharset**

Установить символ в строке в заданной позиции. Если строка короче - вернуть -1  
Отсчет начинается от 0, т.е. первый символ имеет позицию 0.

Вызов:

int strposcharset(string sourcestr, int ipos, char chartoplace, string &strresult);

Здесь:

sourcestr - исходная строка;

ipos - позиция, в которой произвести замену;

chartoplace - символ, на который заменить.

strresult - строка результат.

Пример:

Считать из позиции 2 символ в строке "ABCDE" и записать символ '\_' в позицию 4 этой строки.

char ch;

string st, strresult;

st = "ABCDEF";

int iret = strposcharget(st, 2, ch);

coutvar << ch ;

ch = 'W';

iret = strposcharset(st, 1, ch, strresult);

coutvar << strresult;

В итоге в окно сообщений будет выведено:

B

ABCD\_F

**strposcharget**

Получить символ в строке. если строка короче - вернуть -1

Вызов:

int strposcharget(string sourcestr, int ipos, char &charres);

**strleft**

Получить левую часть строки разделенной фрагментом (сам фрагмент в результат не включается).

Вызов:

string strleft(string str, string substr, bool isfromlast=**false**);

Здесь:

str - исходная строка,

substr - фрагмент-разделитель.

Если указано isfromlast и оно true - то при наличии нескольких фрагментов считать разделителем последний фрагмент.

**strright**

Получить правую часть строки разделенной фрагментом.

Вызов:

string strright(string str, string substr, bool isfromlast=**false**);

Здесь:

str - исходная строка,

substr - фрагмент-разделитель.

Если указано isfromlast и оно true - то при наличии нескольких фрагментов считать разделителем последний фрагмент.

Сам разделяющий фрагмент в результат не записывается

**str\_regex\_search**

Проверить строку на наличие фрагмента шаблона регулярного выражения.

В отличии от команды str\_regex\_match(), шаблон может содержать конструкции типа (?=text) и (?!text). Это позволяет делать запросы с тестированием на то, что строка содержит какие-то фрагменты, а другие фрагменты не содержит.

Например:

(?=Утеплитель)(?=.\*минерал.\*)(?=.\*ват.\*)(?!.\*рулон.\*)

Это означает, что мы хотим выбрать строки, подержащие такой текст:

*Минераловатный утеплитель*

*Утеплитель из минеральной ваты*

*Утеплитель из минерало-ватных плит*

Но в выборку не попадут

*Утеплитель рулонный из минеральной ваты*

Это удобно, если нужно одним оператором проверить наличие в тексте, например, фрагмента типа "В22", когда есть вероятность, что это написано большой 'В', или маленькой 'в', на английском или русском алфавите [(см regex шаблоны)](#regex).

Вызов:

bool str\_regex\_search(string source, string code);

Здесь:

source - строка, в которой производится поиск,

code - регулярное выражение.

**str\_regex\_match**

Проверить строку на соответствие шаблону регулярного выражения.

Например, мы можем протестировать соответствует ли текст по форме записи адреса email или форме записи даты или номера телефона и т.п.

Это удобно, если нужно одним оператором проверить наличие в тексте, например, фрагмента типа "В22", когда есть вероятность, что это написано большой 'В', или маленькой 'в', на английском или русском алфавите [(см regex шаблоны)](#regex).

Вызов:

bool str\_regex\_match(string source, string code);

Здесь:

source - строка, в которой производится тестирование на соответствие шаблону,

code - регулярное выражение.

Пример.

Проверить соответствует ли текст форме записи адреса email в домене ru.

string code = ".\*@.\*\.ru";

// шаблон читается как - впереди идет что угодно, затем символ @ затем что угодно, затем точка ру.

**str\_regex\_replace**

Заменить фрагмент текста по шаблону regex (регулярные выражения).

Вызов:

string sResult = str\_regex\_replace(string sCodedValue, string sOriginalValue, string sReplaceFragment);

Здесь:

sCodedValue - шаблон для поиска в формате регулярных выражений (regex, типа "abc[12]"),

sOriginalValue - исходный текст, где требуется заменить фрагменты, удовлетворяющие шаблону,

sReplaceFragment - фрагмент, на который надо заменить все эти фрагменты.

В sResult - результат операции.

Если фрагменты, удовлетворяющие шаблону не найдены, то возвращается строка sOriginalValue без изменения.

Пример.

Заменить в тексте "123 abc 456" часть 12 и 6 на символ #.

string sres = str\_regex\_replace("(?:12)|(?:[6])", "123 abc 456", string "#");  
coutvar << sres;

Результат:

sres = "#3 abc 45#"

Пример.

Выделить из значения заданного пользовательского свойства элемента число слева в конструкции "10001m" или "2200k" при условии, что буква может быть любая.

ac\_request\_special("get\_element\_value", "UP", "Технические переменные для отверстий для LabPP\\Маркировка отверстия");

svalue = ac\_getstrvalue();

svalue = str\_regex\_replace("[a-z]", svalue, "#");

svalue = strleft(svalue, "#");

coutvar << svalue;

Результат:

10001 2200 и т.п. в зависимости от значения свойства.

**str\_regex\_substr**

Получить из текста подстроку, соответствующую шаблону.

Вызов:

bool str\_regex\_substr(string source, string code, string usSubstrRes, string usPrefix, string usSuffix);

Здесь:

source - строковая переменная с текстом, из которого требуется получить текст, соответствующий шаблону;

code - шаблон;

usSubstrRes - реальный фрагмент из текста, соответствующая шаблону;

usPrefix - строка, предшествующая фрагменту текста;

usSuffix - строка за фрагментом до конца текста;

Возвращает true - если текст, соответствующий шаблону найден.

Пример: получить из текста ширину и высоту дверного блока

string s = "Дверной блок из алюминиевых профилей (BxH): 1400x2100 мм"; // исходный текст  
string usSubstrRes, usPrefix, usSuffix;  
string code = "\\(BxH\\)\\: \\d{1,}x.\*\\d{1,}"; // шаблон означает: текст, у которого (BxH): "одна или несколько цифр" x "одна или несколько цифр"  
bool b = str\_regex\_substr(s, code, usSubstrRes, usPrefix, usSuffix);  
**if** (b)  
{  
 coutvar << usSubstrRes; // в окно сообщений будет выведено "(BxH): 1400x2100 мм"  
 code = "\\d{1,}"; // шаблон означает: текст, у которого "одна или несколько цифр"  
 b = str\_regex\_substr(usSubstrRes, code, usSubstrRes, usPrefix, usSuffix);  
 coutvar << usSubstrRes; // в окно сообщений будет выведено "1400"   
 **if** (b)  
 {  
 code = "\\d{1,}"; // шаблон означает: текст, у которого "одна или несколько цифр"  
 // для поиска второго числа используем суффикс от прошлого поиска  
 b = str\_regex\_substr(usSuffix, code, usSubstrRes, usPrefix, usSuffix);  
 coutvar << usSubstrRes; // будет выведено "2100"  
 }  
}

**straddslashtoregexcodes**

заменить в строке '.' на '\.' , '[' на '\[', '(' на '\(' и т.д. для точного сравнения.

Вызов:

string straddslashtoregexcodes(string source);

**strremovebaddelimiters**

Убрать из текста определенные засоряющие разделители.

Убираются разделители перед и в конце строки. Вместо них ставиться "".

А так же убираются серии разделителей, идущие подряд через пробел вместе с пробелами перед такими сериями и после них.

Вместо серий из разделителей ставится один " ".

Вызов:

string strremovebaddelimiters(string originalstr, string delimiter);

Здесь:

originalstr - оригинальная строка, в которой требуется убрать мусор,

delimiter - разделитель. Он может состоять из нескольких символов, когда у вас текст разделен на части не одним знаком, а целой последовательностью (например "???!!!").

Пример.

Убрать засоряющие "," из текста

int main()  
{  
 string s = ", text,, еще текст ,, , и еще вот текст, ";  
 string s2 = strremovebaddelimiters(s,",");  
 coutvar << s2;  
 // результат: "text еще текст и еще вот текст"  
}

**str2threekeys**

Выделить из строки три ключа для сортировки в таблицах (для 115.2 или 14а.1 и т.п.)

Вызов:

int str2threekeys(string sSource, double& k1,double &k2, double &k3);

Здесь:

sSource - исходная строка.

Функция возвращает кол-во ключей (числовых полей).

Она выявляет три числовых поля. Буквы приводит к числам 10000+код буквы,

в k1,k2 и k3 возвращаются числовые значения этих трех полей или ноль, если поле не найдено.

**strCompareLikeABC10\_a**

Сравнить строки по принципу номеров помещений.

Возвращает -1/0/1 если строка s1 < s2 / s1=s2/ s1 > s2.

Сравнение идет в нижнем регистре.

Вызов:  
int strCompareLikeABC10\_a(string s1, string s2);

Здесь:

s1 и s2 - сравниваемые строки.

Пример:

Сравнить выше или ниже по списку должно размещаться помещение с именем "ПОМЕЩЕНИЕ 102.2b" относительно помещения с именем "Помещение 102.3.b"

string s1 = "ПОМЕЩЕНИЕ 102.2b";  
string s2 = "Помещение 102.3.b";  
  
int ires = strCompareLikeABC10\_a(s1, s2);

coutvar << ires;

Результат - в поле сообщений будет выведено значение -1. Это означает, что помещение 102.2b в списке должно располагаться раньше чем помещение 102.3.b.

**Руководство по LABPP\_Automat для ARCHICAD**

Заменить escape-последовательности типа "\u0420\u0435" на символы.

Вызов:

string strreplace\_escape\_u(string sSource);

возвращает строку с замененными на символы escape-последовательностями.

**strisnumber**

Является ли строка числом.  
Если в строковой переменной записано число, то возвращает 1, в проитвном случае возвращает 0.

Обращение

bool result = strisnumber(string str);

Здесь:

str - строковая переменная для определения текст это или число.

**smallnums**

Перевести цифры в тексте в цифры в верхний регистр (unicode).

Вызов:

string sresult = smallnums(string str);

Здесь:

str - строковая переменная с числом.

Пример.

Записать в строковую переменную значение "м²".

string str = "Объем, м" + smallnums("2");

**ts\_text\_mc**

Получить текст стандартного сообщения на языке программы.

Вызов:

string sresult = ts\_text\_mc(int itextcode);

Здесь:

itextcode - индекс сообщения по таблице значений.

|  |  |
| --- | --- |
| Индекс | Текст |
| 1 | Запуск |
| 2 | Старт |
| 3 | Отменить |
| 4 | Продолжить |
| 5 | Ошибка |
| 6 | Информация |
| 7 | Предупреждение |
| 8 | Проверка |
| 9 | Выполнить все |
| 10 | Ok |
| 11 | Создать |
| 12 | Открыть |
| 13 | Закрыть |
| 14 | Считать |
| 15 | Записать |
| 16 | Да |
| 17 | Нет |
| 18 | Помощь |

Пример.

Получить текст слова "Запуск" на том языке, на котором работает программа, запустившая скрипт на выполнение.

string str = ts\_text\_mc(1);

**Объект ac\_element\_guid (ts\_guid)**

**Объект ac\_element**

**ac\_request**

**get\_floor\_name\_by\_floor\_index**

Получить индекс этажа по его имени.

Вызов:

int ires = ac\_request("get\_floor\_index\_by\_floor\_name", string sStoryName);

Индекс этажа считывается вызовом ac\_getnumvalue();

Возвращает в ires -1 если этажа с таким именем нет или возникла ошибка.

При успешном завершении возвращает 0.

Пример:

Получить индекс этажа с именем "2-й этаж" и вывести его в окно сообщений.

int ifloorindex;  
int ires = ac\_request("get\_floor\_index\_by\_floor\_name", "2-й этаж");  
**if**(ires == 0)  
{  
 ifloorindex = ac\_getnumvalue();  
 coutvar << ifloorindex;  
}

**get\_floor\_index\_by\_floor\_name**

Получить имя этажа по его индексу.

Вызов:

int ires = ac\_request("get\_floor\_name\_by\_floor\_index", int iStoryIndex);

Имя этажа считывается вызовом ac\_getstrvalue();

Возвращает в ires -1 если этажа с таким индексом нет или возникла ошибка.

При успешном завершении возвращает 0.

Пример:

Получить имя этажа с индексом 1 и вывести его в окно сообщений.

string sStoryName;  
int ires = ac\_request("get\_floor\_name\_by\_floor\_index", "2-й этаж");  
**if**(ires == 0)  
{  
 sStoryName = ac\_getstrvalue();  
 coutvar << sStoryName;  
}

**store\_cur\_element\_to\_descr**

Сохранить текущий элемент в объект ac\_element\_guid.

Чтобы работать с элементом через функции ac\_request(... мы делаем элемент текущим. Например это делается когда мы загрузили элементы в один из внутренних списков и выбрали текущую позицию.  
Но может потребоваться сохранить его уникальный номер для обработки другими функциями. Тогда создаем объект типа ac\_element\_guid. При создании объекта получаем его дескриптор - внутренний номер объекта. И с этим дескриптором обращаемся сюда. В результате - в объект ac\_element\_guid будет записан уникальный номер элемента.

Формат запроса:

ac\_request("store\_cur\_element\_to\_descr", int &iGuidDescr);

Здесь iGuidDescr - дескриптор объекта типа ac\_element\_guid, в который будет сохранен уникальный номер текущего элемента.

**set\_current\_element\_from\_descr**

Сделать текущим для работы с командами ac\_reauest(... элемент из объекта ac\_element\_guid. Это по сути обратная операция указанной выше.

Формат запроса:

ac\_request("set\_current\_element\_from\_descr", int &iGuidDescr);

Здесь iGuidDescr - дескриптор объекта типа ac\_element\_guid, который будет установлен в качестве текущего для работы с функциями ac\_request(....

**set\_element\_by\_guidstr\_as\_current**

Сделать текущим для работы с командами ac\_reauest(... элемент с GUID заданным текстовым аргументом.

Вызов:

ac\_request("set\_element\_by\_guidstr\_as\_current", string sGUID);

Здесь sGUID - текстовая переменная, в которой задан guid элемента текстом.

**get\_guid\_from\_element**

Получить уникальный номер элемента ARCHICAD из объекта типа ac\_element в объект типа ac\_element\_guid.

Такое преобразование может потребоваться для технических целей.

Формат запроса:

ac\_request("get\_guid\_from\_element",int iElementDescr, int iGuidDescr);

Здесь:

iElementDescr - дескриптор объекта дипа ac\_element, из которого будет записан уникальный код в объект типа ac\_element\_guid, на который указывает iGuidDescr.

**load\_element\_from\_guid**

Загрузить в объект типа ac\_element данные элемента ARCHICAD, с уникальным номером (guid), который указан в объекте типа ac\_element\_guid.

Это тоже техническая функция, для создания возможности работать с элементом через механизм специального объекта ac\_element.

Вызов:

ac\_request("load\_element\_from\_guid",int iElementDescr, int iGuidDescr);

Здесь:

iElementDescr - дескриптор объекта дипа ac\_element, в который загружаем уникальный код элемента, записанный в объекте типа ac\_element\_guid, на который указывает iGuidDescr.

**load\_elements\_list**

Загрузить заданный список элементами ARCHICAD по заданным условиям

Вызов:

ac\_request("load\_elements\_list",int iListNum,string sElemTypeName,"MainFilter",int iMainFilterValue,string filterparametrname, string/double filterparametervalue,...);

Здесь:

iListNum - номер внутреннего списка элементов (от 0...9).

sElemTypeName - название типа элемента. Если нужно выбрать любые элементы - "ZombieElemType", если только колонны, то "ColumnType", стены - "WallType" и т.д.

"MainFilter" - сообщает, что мы задаем параметры для отфильтровывания элементов по типу доступности для редактирования, видимости и т.п. iMainFilterValue - обычно значение для видимых и редактируемых элементов - 3.

Далее идут названия параметров и их значения для дополнительного отбора.

Например "ID","Значение ID" - означает, что нужно выбрать элементы с ID="Значение ID" и т.д.

Фильтр "MainFilter" можно и не указывать. Это равносильно тому, что написать "MainFilter",0.

Т.е. при формировании списка в него попадут все подходящие элементы в проекте без учета основного критерия.

Значение фильтра собирается по формуле.

MainFilterValue=j1+2\*j2+4\*j3+8\*j4+16\*J5+32\*j6+64\*j7+128\*j8+256\*j9+512\*j10+1024\*j11+2048\*j12+4096\*j13+268435456\*j14,

где каждое j может быть 0 или 1.

j1: только редактируемые.

j2: на видимом слое.

j3: на текущем этаже

j4: имеет представление в 3d окне

j5: в моем рабочем пространстве

j6: не подчиненные а только независимые элементы

j7: на активном чертеже

j8: отображается внутри обрезанной части базы данных чертежа

j9: указывает, передан ли данному элементу идентификатор изменения в параметре variationID

j10: имеются права доступа к элементу

j11: элемент виден в реновации

j12: дополнительный флаг; проверяет, переопределены ли атрибуты элемента текущим фильтром обновления

j13: дополнительный флаг; проверяет видимость элемента с учетом текущей настройки отображения структуры

j14: только из 2d окна

**set\_current\_element\_from\_list**

Установить текущим для обработки элемент из заданного списка.

Вызов:

int ires = ac\_request("set\_current\_element\_from\_list", int listindex, int elemindexinlist);

Здесь:

listindex - номер списка элементов, из которого надо сделать элемент текущим для выполнения операций над ним.

elemindexinlist - интекс элемента в этом списке.

ires - при успешном выполнении равно 0.

**add\_elements\_list**

То же что и load\_element\_list, с той лишь разницей, что список не обнуляется. Так мы можем дописать в список элементы, отобранные по любым параметрам и любых типов.

**load\_elements\_list\_from\_selection и add\_elements\_list\_from\_selection**

Тоже, что и load\_elements\_list, только элементы берутся из числа выбранных элементов в текущем окне ARCHICAD.

**load\_elements\_list\_curdb**

То же, что и load\_elements\_list, но выбор элементов производится без перехода в другое окно и сохраняется текущее выделение элементов.

**clear\_list**

Очистить указанный список элементов.

Вызов:

ac\_request("clear\_list",int iListNum);

Здесь:

iListNum - номер внутреннего списка элементов, который нужно очистить.

**store\_current\_element\_to\_list**

Сохранить текущий элемент в другой список.

Вызов:

ac\_request("store\_current\_element\_to\_list",int listnum, int index);

Здесь:

listnum - номер списка, в который добавить текущий элемент.

Если индекс index = -1 то добавить в конец списка.

Иначе элемент записать в позицию индекса. В списках индекс начинается с 0.

Список не раздвигается.

**get\_loaded\_elements\_list\_count**

Определить количество элементов в указанном списке.

Вызов:

ac\_request("get\_loaded\_elements\_list\_count", int iListNum);

Здесь:

iListNum - номер списка, у которого запрашивается количество элементов.

Результат операции считывается командой ac\_getnumvalue();

**select\_elements\_from\_list**

Выделить в текущем окне ARCHICAD элементы, содержащиеся во внутреннем списке с указанным номером.

Вызов:

ac\_request("select\_elements\_from\_list", int iListNum, bool bClear);

Здесь - iListNum - номер списка,

bClear - снять выделение с элементов перед выделением новых.

**Слой - layer**

**create**

Создать слой ARCHICAD.

Вызов:

int ires = ac\_request("layer","create", string slayername);

Здесь:

slayername - имя слоя, которй необходимо создать.

Если слой успешно создан в ires возвращается 0.

Если произошла ошибка, то возвращается отрицательное значение.

**get\_index**

Получить индекс слоя в проекте ARCHICAD.

Вызов:

int ires = ac\_request("layer","get\_index", string slayername);

Здесь:

slayername - имя слоя, у которого требуется получить индекс.

в ires возвращается 0 если обращение произошло без ошибок,

или отрицательное значение если произошла ошибка или такого слоя в проекте нет.

индекс слоя получается функцией ac\_getnumvalue();

Пример.

Получить индекс слоя "01 10 Несущие стены", если такого слоя нет - то создать его и вывести в окно сообщений.

string slayername = "01 10 Несущие стены";  
int ires = ac\_request("layer","get\_index",slayername);  
**if**(ires < 0)  
{  
 *// layer not exist - create it*   
 ires = ac\_request("layer","create",slayername);  
 **if**(ires == 0)  
 {  
 ires = ac\_request("layer","get\_index",slayername);  
 }  
 **else**  
 {  
 cout << "cann't create layer "<<slayername;  
 **return** -1;  
 }  
}  
  
int iLayerIndex = ac\_getnumvalue();  
cout << "Index of the layer=" << iLayerIndex << "\n";

**set\_layer\_visible**

Управление видимостью слоя.

Оращение:

ac\_request("layer", string whatdo);

Здесь whatdo = "ON"/"OFF"/"SWITCH" включить, выключить или переключить состояние (был выключен - включить и наоборот).

**get\_element\_overall\_dimensions**

Получить общие габариты элемента.

Это можно использовать для ускорения принятия решения когда требуется определить взаиморасположение элементов и т.п. задач.

Вызов:

ac\_request("get\_element\_overall\_dimensions", double &lx, double &ly, double &lz);

Здесь: lx,ly,sz - возвращаемые значения габаритных размеров текущего элемента.

**get\_quantity\_value**

Получить количественные данные из текущего элемента

Вызов:

int iret = ac\_request("get\_quantity\_value", string svaluename);

Здесь:

svaluename - имя расчетного параметра текущего элемента,

iret - 0, если произошло успешное получение данных.

Результат запроса считывается функцией ac\_getnumvalue();

Пример.

Выбрать в список №1 элементы с ID="s участка".

Суммировать площади элементов и выдать результат в окно сообщений.

ac\_request("load\_elements\_list",1,"ZombieElemType","ID","s участка","MainFilter",3);  
ac\_request("get\_loaded\_elements\_list\_count", 1);  
int icount = ac\_getnumvalue();  
cout << "Число выбранных элементов=" << icount;  
int i; *// объявляем переменную цикла.*  
double value; *// для текущих значений*  
double summa=0; *// для суммарной площади*  
**for**(i=0;i<icount;i++)  
{  
 ac\_request("set\_current\_element\_from\_list", 1, i);  
 ac\_request("get\_quantity\_value", "Surface");  
 value = ac\_getnumvalue();  
 cout <<"элемент № "<< i << ", площадь="<< value << "\n";  
 summa += value;  
}  
cout << "суммарная площадь=" << summa;

Подробности.

Для разных типов элементов доступны различные количественные показатели.

#### Стены - WallType:

volume - объем

length - средняя длина стены:

volumeсond - условный объем;

volumeaskin - объем покрытия стены со стороны опорной линии;

volumebskin - объем покрытия стены с обратной стороны опорной линии;

volumeaskincond - условный объем покрытия стены со стороны опорной линии;

volumebskincond - условный объем покрытия стены с обратной стороны опорной линии;

surfacereflineside - площадь поверхности со стороны опорной линии;

surfacereflineoppside - площадь поверхности с обратной стороны опорной линии;

surfaceofedge - площадь поверхности торца стены;

surfacereflinesidecond - условная площадь поверхности со стороны опорной линии;

surfacereflineoppsidecond - условная площадь поверхности с обратной стороны опорной линии;

surfacewindows - площадь оконных проемов;

surfacedoors - площадь дверных проемов;

surfacetmptyholes - площадь незаполненных проемов;

columnsvolume - объем колонн в стене;

columnsnumber - количество колонн в стене;

widthofwindows - суммарная ширина всех окон;

widthofdoors - суммарная ширина всех дверей;

minheight - минимальная высота стены;

maxheight - максимальная высота стены;

minheightaskin - минимальная высота покрытия стены со стороны опорной линии;

maxheightaskin - максимальная высота покрытия стены со стороны опорной линии;

minheightbskin - минимальная высота покрытия стены с обратной стороны опорной линии;

maxheightbskin - максимальная высота покрытия стены с обратной стороны опорной линии;

centerlength - длина стены по центральной линии;

area - площадь опоры стены;

petimeter - периметр опоры стены;

grossvolume - общий объем стены;

grosssurfacereflineside - общая площадь поверхности стены по стороне опорной линии;

grosssurfacereflineoppside - общая площадь поверхности стены с обратной стороны опорной линии;

emptyholesvolume - аналитический объем открытий в стене;

emptyholessurfreflineside - аналитическая площадь открытия в стене со стороны опорной линии;

emptyholessurfreflineoppside - аналитическая площадь открытий в стене с обратной стороны опорной линии;

lengthonreflineside - длина стены по со стороны опорной линии;

lengthonreflineoppside - длина стены с обратной стороны опорной линии;

lengthonreflinesidecond - условная длина стены со стороны опорной линии;

lengthonreflineoppsidecond - условная длина стены с обратной стороны опорной линии:

insulationskinthickness - толщина изоляции стены;

wallairskinthickness - толщина воздушной прослойки в стене;

skinreflinethickness - толщина изоляции по стороне опорной линии;

skinreflineoppthickness - толщина изоляции с обратной стороны опорной линии;

reflinelength - длина стены по опорной линии.

#### Колонна - ColumnType:

surface - площадь;

coresurface - площадь базы колонны;

venesurface - площадь отделки;

volume - объем;

veneervolume - объем отделки;

minheight - минимальная высота;

maxheight - максимальная высота;

perimeter - периметр;

area - площадь;

grosssurfaceofcore - валовая площадь поверхности основы;

grosssurfaceofveneer - валовая площадь поверхности отделки;

coregrossvolume - валовый объем основы колонны;

veneergrossvolume - валовый объем основы колонны;

coretopsurface - площадь поверхности верхней части основы колонны;

corebottomsurface - площадь поверхности опоры основы колонны;

veneertopsurface - поверхность отделки сверху;

veneergrosssurface - валовая поверхность отделки снизу;

coregrosstopandbotsurface - валовая поверхность основы сверху и снизу;

veneergrosstopandbotsurface - валовая поверхность отделки сверху и снизу.

#### Перемычка - BeamType:

rightlength - длина перемычки с правой стороны опорной линии;

leftlength - длина перемычки с левой стороны опорной линии;

length - средняя длина;

bottomsurface - площадь нижней поверхности;

topsurface - площадь верхней поверхности;

edgesurfaceleft - площадь поверхности слева от опорной линии;

edgesurfaceright - площадь поверхности справа от опорной линии;

edgesurface площадь поверхностей обоих концов перемычки;

holessurface - поверхность вырезов;

holesedgesurface - площадь вырезов на боковых гранях перемычки;

holesnumber - количество вырезов;

volume - объем;

condvolume - условный объем;

holesvolume - объем вырезов;

#### Окно - WindowType:

openwidthrevside - ширина открытия на стороне раскрытия;

openwidthrevsideopp - ширина открытия на стороне, противоположной стороне раскрытия;

openheightrevside - высота открытия на стороне раскрытия;

openheightrevsideopp - высота отверстия на стороне, противоположной стороне раскрытия;

opensurfacerevside - поверхность открытия на стороне раскрытия;

opensurfacerevsideopp - поверхность открытия на стороне, противоположной стороне раскрытия;

nominalopenwidthrevside - номинальная ширина открытия на стороне раскрытия;

nominalopenwidthrevsideopp - номинальная ширина открытия на стороне, противоположной стороне раскрытия;

nominalopenheightrevside - - номинальная высота открытия на стороне раскрытия;

nominalopenheightrevsideopp - номинальная высота открытия на стороне, противоположной стороне раскрытия;

nominalopensurfacerevside - номинальная поверхность открытия со стороны стороны раскрытия;

nominalopensurfacerevsideopp - номинальная поверхность открытия на стороне, противоположной стороне раскрытия;

volume - объем;

nominalopensurgace - номинальная площадь открытия;

nominalopenvolume - номинальный объем открытия;

surface - площадь;

nominalsillheight - номинальная высота подоконника;

nominalsillheightrevside - высота подоконника на стороне раскрытия;

nominalsillheightrevsideopp - высота подоконника на стороне, противоположной стороне раскрытия;

nominalheadheight - номинальная высота оконной головки;

nominalheadheightrevside высота оконной головки на стороне раскрытия

nominalheadheightrevsideopp - высота оконной головки на стороне, противоположной стороне раскрытия;

sillheightaccvertanchor - высота подоконника в соответствии с вертикальным анкером;

headheightaccvertanchor - высота оконной головки в соответствии с вертикальным анкером.

#### Двери - DoorType:

openwidthrevside - - ширина открытия на стороне раскрытия;

openwidthrevsideopp - ширина открытия на стороне, противоположной стороне раскрытия;

openheightrevside - высота открытия на стороне раскрытия;

openheightrevsideopp - высота отверстия на стороне, противоположной стороне раскрытия;

opensurfacerevside - поверхность открытия на стороне раскрытия;

opensurfacerevsideopp - поверхность открытия на стороне, противоположной стороне раскрытия;

nominalopenwidthrevside - номинальная ширина открытия на стороне раскрытия;

nominalopenwidthrevsideopp - номинальная ширина открытия на стороне, противоположной стороне раскрытия;

nominalopenheightrevside - номинальная высота открытия на стороне раскрытия;

nominalopenheightrevsideopp - номинальная высота открытия на стороне, противоположной стороне раскрытия;

nominalopensurfacerevside - номинальная поверхность открытия со стороны стороны раскрытия;

nominalopensurfacerevsideopp - - номинальная поверхность открытия на стороне, противоположной стороне раскрытия;"

volume - объем;

nominalopensurgace - номинальная площадь открытия;

nominalopenvolume - номинальный объем открытия;

surface - площадь;

nominalsillheight - номинальная высота подоконника;

nominalsillheightrevside - высота порога двери на стороне раскрытия;

nominalsillheightrevsideopp - высота порога двери на стороне, противоположной стороне раскрытия;

nominalheadheight - номинальная высота дверной головки;

nominalheadheightrevside - номинальная высота дверной головки на стороне раскрытия;

nominalheadheightrevsideopp - высота дверной головки на стороне, противоположной стороне раскрытия;

sillheightaccvertanchor - высота порога в соответствии с вертикальным анкером;

headheightaccvertanchor - высота дверной головки в соответствии с вертикальным анкером.

#### Объект - ObjectType:

volume - объем;

surface - площадь символа.

#### Элемент освещения - LampType:

volume - объем;

surface - площадь.

#### Плита - SlabType:

bottomsurface - площадь опорной поверхности;

surface - площадь верхней поверхности;

edgesurface - площадь боковых граней;

condbottomsurface - условная площадь опорной поверхности;

condtopsurface - условная площадь верхней поверхности;

volume - объем;

condvolume - условный объем;

perimeter - периметр;

holessurface - площадь вырезов;

holesperimeter - периметр вырезов;

grossbottomsurface - общая площадь опорной поверхности;

grosstopsurface - общая площадь верхней поверхности;

grossedgessurface - общая площадь поверхности граней;

grossvolume - общий объем;

grossbottomsurfacewithholes - общая площадь опорной поверхности вместе с вырезами;

grosstopsurfacewithholes - общая площадь верхней поверхности вместе с вырезами;

grossedgessurfacewithholes - общая площадь граней вместе с вырезами;

grossvolumewithholes - общий объем вместе с вырезами.

#### Крыша - RoofType:

bottomsurface - площадь внутренней поверхности;

topsurface - площадь верхней поверхности;

edgesurface - площадь боковых граней;

condbottomsurface - условная площадь внутренней поверхности;

condtopsurface - условная площадь верхней поверхности;

volume - объем;

сondvolume - условный объем;

perimeter - периметр;

holessurface - площадь вырезов;

holesperimeter - площадь периметра;

grossbottomsurface - общая внутренняя площадь;

grosstopsurface - общая площадь верхней поверхности;

grossedgesurface - общая площадь граней;

areacontourpoligon - площадь крыши по полигонам контура;

grossvolume - общий объем;

insskinthickness - толщина слоя изоляции;

lengthofridgeedgesdiv2 - длина граней реберного типа, деленная на 2;

lengthofvalleyedgesdiv2 - длина граней желобного типа, деленная на 2;

lengthofgableedges - длина фронтонов;

lengthofhipedgesdiv2 - длина ребер типа бедра, деленная на 2;

lengthofeaveedges - длина ребер типа краев;

lengthofpeakedges - длина краев пикового типа;

lengthofsidewalledges - длина ребер типа боковых стенок;

lengthofendwalledges - длина ребер типа окончания стен;

lengthofrtdomeedgesdiv2 - длина ребер типа rtdom, разделенная на 2;

lengthofrthollowedgesdiv2 - длина краев типа rthollow, деленная на 2;

sumofopeningssurfaces - суммарная площадь открытий в крыше;

numofholes - количество вырезов;

numofskylight - количество просветов (skylight).

#### 3d сетка - MeshType:

bottomsurface - площадь опорной поверхности;

topsurface - площадь верхней поверхности;

edgesurface - площадь боковых граней;

volume - объем;

perimeter - периметр;

holessurface площадь вырезов;

holesperimeter - периметр вырезов;

projectedarea - площадь проекции;

#### Зона - ZoneType:

area - площадь зоны;

perimeter - периметр;

holesperimeter - периметр вырезов;

wallsperimeter - периметр стен;

numberofcornersprojectedarea - количество углов зоны;

numberofconcavecorners - количество вогнутых углов зоны;

surfaceareaofperimeterwall - площадь поверхности стен по периметру зоны;

widthofdoors - ширина всех дверей в стенах по периметру зоны;

surfaceofdoors - площадь всех дверей в стенах по периметру зоны;

widthofwindows - ширина всех окон в стенах по периметру зоны;

surfaceofwindows -площадь поверхности всех окон в стенах по периметру зоны;

floorlevel - уровень пола в зоне;

subfloorthickness - толщина под полом в зоне;

height - высота;

netarea - чистая площадь зоны;

netperimeter - чистый периметр;

volume - объем зоны;

areareducement - величина уменьшения площади;

calcarea - расчитанная площадь;

totalextractedarea - вычтенная площадь у зоны;

reducedextractedarea - уменьшенная площадь зоны;

lowareaofzone - вычтенная нижняя часть зоны;

extractedwallarea - вычтенная площадь за счет стен;

extractedcolumnarea - площадь вычтенная за счет колонн;

extractedfillarea - площадь, вычтенная за счет штриховок;

wallincettopsurface - верхняя поверхность настенной вставки;

wallincetbacksidesurface - задняя сторона стены;

wallincetsidesurface - бортовая поверхность настенной вставки;

tsfloorplintuslength - длина плинтуса с учетом дверей (если под дверью можно проложить плинтус, то считаем);

tsseilingplintuslength - длина потолочного плинтуса;

#### Штриховка - HatchType:

surface - площадь поверхности;

perimeter - периметр;

holesperimeter - периметр вырезов;

holessurface - площадь вырезов.

#### Линия - LineType:

length - длина линии.

#### Полилиния - PolyLineType:

length - длина полилинии.

#### Дуга - ArcType:

length - длина дуги.

#### Окружность - CircleType:

length - длина окружности.

#### Сплайн - \_SplineType:

length - длина сплайна

#### Оболочка - ShellType:

bottom\_surface - площадь поверхности снизу;")

top\_surface - площадь поверхности сверху;

volume - объем;

surface - площадь поверхности по опорной линии;

holes\_surface - площадь вырезов.

#### Морф - MorphType:

surface - площадь поверхности;

volume - объем;

floorplanprojarea - площадь опоры;

elevation - возвышение;

baseheight - базовая высота - диапазон высоты от самой нижней точки наклонных внутрь граней до самой высокой точки в этаже высоты относительно этажа;

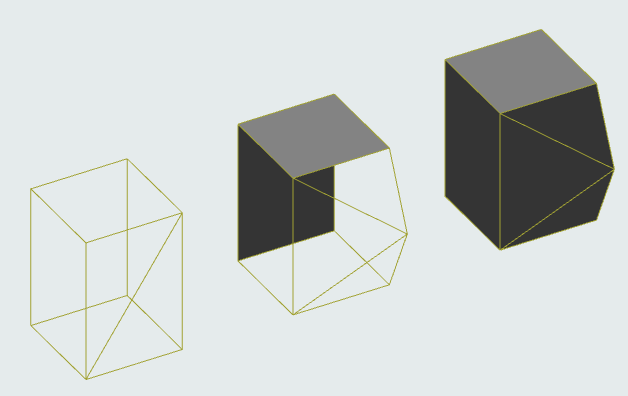
height - высота тела;

floorplanprojperimeter - периметр проекции в плане;

numberofnodes - количество вершин;

numberofedges - количество ребер;

numberofhiddenedges - количество невидимых ребер;



numberofsoftedges - количество "мягких" ребер;

numberofvisiblenodes - количество видимых вершин;

numberoffacenodes - количество лицевых сторон в тела.

length\_edges - длина всех ребер (видимых и невидимых).

length\_edge\_max - длина самого длинного ребра

length\_edges\_pgons - длина всех ребер (видимых и невидимых) где есть полигоны граней.

length\_edge\_pgons\_max - длина самого длинного ребра, где есть полигоны граней.

**get\_layer\_by\_substring**

Получить полное имя слоя по частичному фрагменту.

Требуется, когда пользователи договариваются именовать слои в ARCHICAD с числовыми или текстовыми префиксами.

Например слой "Квартиры" можно написать как "01 Квартиры" или "20 Квартиры". Такой подход встречается когда пользователь хочет установить удобный ему порядок следования слоев в проекте.  
Однако с точки зрения программы - это совершенно разные слои.

Чтобы избежать ошибки, удобно пользоваться данной функцией.

Вызов:

int ires = ac\_request("get\_layer\_by\_substring",string slayernamefragment, int from);  
int ires = ac\_request("get\_layer\_by\_substring",string slayernamefragment, int from, int range);

Здесь:

slayernamefragment - фрагмент в имени, который точно определяет слой (например "Квартиры").

from - порядковый номер символа от начала полного имени слоя в проекте, с которого начать сравнение на соответствие.

range - количество символов, которые нужно проверить.

ires - 0, если слой найден.

Полное наименование найденного слоя считывается командой ac\_getstrvalue();

Пример.

Получить полное имя слоя "Квартиры" в проекте, если известно, что в организации принято для сортировки слоев пользоваться префиксом типа "01 ".

string smalllayername="Квартиры";  
string longlayername;  
ires = ac\_request("get\_layer\_by\_substring",smalllayername,3); *// начать сравнение с 3-й позиции*  
**if**(ires != 0) {  
 tsalert(-1,"Ошибка во время выполнения","Не обнаружен слой",smalllayername);  
 cout << "Ошибка: Не обнаружен слой - "<< smalllayername << "\n";  
}  
longlayername = ac\_getstrvalue(); *// считать полное имя слоя*

**elem\_user\_property**

Пользовательские параметры элемента.

В ARCHICAD имеется возможность назначать собственные свойства любому элементу. Эти свойства могут быть числовыми, текстовыми, логическими и т.п. По значению этих свойств элемент может затем быть отобран в интерактивных каталогах, можно сделать ему автозамену при показе в окне и т.п. Следующие команды позволяют манипулировать пользовательскими свойствами элементов в скриптах C++.

**get**

Считать значение пользовательского параметра для текущего элемента.

Вызов:

int ires = ac\_request("elem\_user\_property","get", string sparname);

Здесь:

svarname - имя параметра, из которого необходимо получить значение.

ires - 0, если считалось успешно.

Результат получается обращением к функции ac\_getnumvalue() или ac\_getstrvalue();

Пример.

Считать значение логического параметра "Полезная площадь здания" из текущего элемента.

string sUP = "Полезная площадь здания";  
int ires = ac\_request("elem\_user\_property","get",sUP);  
**if**(ires==0)  
{  
 istrue = ac\_getnumvalue();  
 **if**(istrue==1)  
 {  
 cout << "Эта зона относится к полезной площади здания";  
 }  
}

Если в проекте есть пользовательские переменные с одинаковыми именами в разных группах, например переменные "Ширина проема" созданы в группе "Переменные окон" и в группе "Переменные дверей", то чтобы обратиться к переменной "Ширина проема" в группе "Переменные окон" надо указать группу в имени переменной через двойной слэш:  
sUP = "Переменные окон\\Ширина проема";

**set**

Записать в пользовательский параметр для текущего элемента новое значение.

Вызов:

int ires = ac\_request("elem\_user\_property","set", string sparname, bool/int/double/string value);

Здесь:

svarname - имя параметра.

value - значение параметра по умолчанию. Тип параметра определяется по типу передаваемого здесь значения переменной.

srazdel - раздел, в котором будет создана пользовательская переменная.

ires - 0- если переменная создана успешно.

Пример.

Установить новое значение логического параметра "Полезная площадь здания" для текущего элемента, в положение "истина".

string sUP = "Полезная площадь здания";  
int istrue = 1; *// Новое значение переменной - 1, значит "истина"*  
int ires = ac\_request("elem\_user\_property","set",sUP,istrue);  
**if**(ires==0)  
{  
 cout << "Новое значение успешно установлено";  
}

Если в проекте есть пользовательские переменные с одинаковыми именами в разных группах, например переменные "Ширина проема" созданы в группе "Переменные окон" и в группе "Переменные дверей", то чтобы обратиться к переменной "Ширина проема" в группе "Переменные окон" надо указать группу в имени переменной через двойной слэш:  
sUP = "Переменные окон\\Ширина проема";

**create**

Создать пользовательский параметр для текущего элемента.

Вызов:

int ires = ac\_request("elem\_user\_property","create", string sparname, string svartype, bool/int/double/string value, string svartype, string spargroupname);

Здесь:

svarname - имя создаваемого параметра.

value - значение параметра по умолчанию.

svartype - Тип переменной - "String"/"Real"/"Integer"/"Boolean" - текстовое, вещественное, целое или логическое.

spargroupname - раздел, в котором будет создана пользовательская переменная.

ires - 0- если переменная создана успешно.

Пример.

Создать у текущего элемента пользовательский параметр "Полезная площадь здания" в разделе "Раздел ТЭП". Тип параметра - логический. Значение по умолчанию - "ложь".

string sUP = "Полезная площадь здания";  
string sUPRazdelName = "Раздел ТЭП";  
int ires = ac\_request("elem\_user\_property","create",sUP,0,"Boolean", sUPRazdelName);   
**if**(ires==0)  
{  
 int istrue = ac\_getnumvalue();  
 **if**(istrue==1)  
 {  
 cout << "Эта зона относится к полезной площади здания";  
 }  
}

**Property\_Import**

Импортировать в проект пользовательские свойства, заданные в тексте XML.

Вызов:

int iret = ac\_request("elem\_user\_property", "Property\_Import", string sXMLsource, int conflictResolutionPolicy);

Здесь:

sXMLsource - текст в формате XML с описанием импортируемых пользовательских свойств,

conflictResolutionPolicy - директива как действовать в случае если встречаются свойства с одинаковыми именами:

0 - добавить новое свойство,

1 - заменить старое свойство на новое,

2 - пропустить добавление конфликтующего свойства.

Пример:

Скачать с сайта XML описание свойств и загрузить их в проект:

int do\_iButtonloadProperties()

{

cout << "load properties\n";

int iStringDescr;

object("create", "ts\_string", iStringDescr);

// загрузка свойств

string spropfname = "Свойства КСИ";

int iret = ts\_string(iStringDescr, "ReadFromURL", "http://www.labpp.ru/TSDownload/KSI/" + spropfname + ".txt");

object("read\_simple\_value", iStringDescr);

string sXMLsource = ac\_getstrvalue();

ts\_dialogcontrol(iProgressBar, "SetValue", 2);

if (iret == 0)

{

coutvar << sXMLsource;

iret = ac\_request("elem\_user\_property", "Property\_Import", sXMLsource, 1);

if (iret == 0)

{

tsalert(-3, "Сообщение", "Свойства успешно загружены", spropfname);

}

}

object("delete", iStringDescr);

}

**get\_object\_property\_value**

Получить значение параметра текущего GDL элемента. Применяется для объектов.

Вызов:

int iret = ac\_request("get\_object\_property\_value",string parametername);

Результат обращения считывается следующей функцией ac\_getnumvalue() или ac\_getstrvalue() - для получения числового или текстового значения соответственно.

Возвращает в iret 0- если запрос прошел успешно.

Пример.

Получить из текущего объекта значение текстового параметра "SNAME".

int iret = ac\_request("get\_object\_property\_value","SNAME");  
string svalue;  
**if**(iret == 0)  
{  
 svalue = ac\_getstrvalue();  
 cout << svalue; *// написать полученное значение в окне сообщений*  
}

**set\_object\_property\_value**

Записать новое значение в параметр текущего GDL элемента. Применяется для объектов.

Вызов:

int iret = ac\_request("set\_object\_property\_value",string parametername, string/double valuetowrite);

Здесь:

parametername - имя параметра объекта, в который необходимо записать значение;

valuetowrite - логическое, целое, вещественное число или текст значения для этого параметра.

Возвращает в iret 0- если запись прошла успешно.

Пример.

Записать в текущий объект значение текстового параметра "SNAME" - "my new name".

int iret = ac\_request("set\_object\_property\_value", "SNAME", "my new name");  
**if**(iret == 0)  
{  
 cout << "Запись прошла успешно";  
}

**set\_object\_property\_value\_curdb**

То же что и set\_object\_property\_value, но без смены текущего окна ARCHICAD.

**get\_element\_value**

Получить значение переменной текущего элемента.

Вызов:

int ires = ac\_request("get\_element\_value", string svaluename);

Здесь:

svaluename - имя переменной элемента.

Возвращаемое значение - 0 - если запрос прошел успешно.

Значение переменной считывается следующим обращением ac\_getnumvalue() или ac\_getstrvalue();

Пример.

Считать номер этажа у текущего элемента.

int floornum;  
int ires = ac\_request("get\_element\_value","StoryIndex");  
**if**(ires == 0)  
{  
 floornum = ac\_getnumvalue();  
 cout << "Номер этажа=" << floornum;  
}

Значения svaluename:

#### TypeID

Получить тип элемента в виде числа.

#### TypeName

Получить тип элемента в виде текста.

#### Layer

Получить имя слоя, на котором находится элемент.

#### ID

Получить ID элемента.

#### StoryIndex

Получить индекс этажа.

#### GuidAsText

Получить уникальный идентификатор (guid) элемента в текстовом виде.

#### Для элементов типа Объект

##### ObjectName

Получить имя библиотечного элемента.

##### level

Возвышение по Z

##### pos.x

Позиция X.

##### pos.y

Позиция Y.

##### offset.x

Смещение внутренней точки начала координат от положения pos.x.

##### offset.y

Смещение внутренней точки начала координат от положения pos.y.

##### другое имя

Если указано другое имя, оно рассматривается как имя параметра, которое надо считать из объекта.

**set\_element\_value**

**assign\_element\_values**

Присвоить значение переменной элемента. Изменяет данные только в объекте ac\_element без воздействия в проекте.

Формат обращения:

ac\_request("assign\_element\_values",int iElemDescr, string paramname, string/double/int paramvalue);

**load\_element\_default\_values**

Загрузить в объект элемента значения по умолчанию.

**set\_element\_infoidtext**

Задать ID для текущего элемента.

Вызов:

ac\_request("set\_element\_infoidtext", string svalue);

Здесь: svalue - новое значение ID для текущего элемента.

**create\_element\_on\_project**

Создать элемент в проекте на базе текущих значений из объекта типа ac\_element.

Вызов:

int ires = acrequest("create\_element\_on\_project",int iElemDescr);

Здесь :

iElemDescr - дескриптор объекта, в котором содержатся данные для создаваемого в проекте элемента.

ires - при успешном создании возвращается 0.

**get\_element\_infoidtext**

Считать ID текущего элемента.

Вызов:

int ires = ac\_request("get\_element\_infoidtext");

Данные получаются следующей функцией ac\_getstrvalue();

ires - 0 если считывание прошло успешно.

**project\_property**

**Руководство по LABPP\_Automat для ARCHICAD**

CalcUnitsAreaDecimals

Получить заданное в проекте число точек после запятой.

Вызов:

ac\_request("project\_property","get","CalcUnitsAreaDecimals");

Результат считывается вызовом функции ac\_getnumvalue().

**autotext**

Считать, записать или создать автотекст ARCHICAD.

В информации о проекте имеется специальный набор переменных. Можно создавать переменные самостоятельно. Значения этих переменных можно подставлять в текст в различных местах проекта. Значение текста в этих местах всегда будет соответствовать актуальному значению переменной проекта.

**get**

Считать значение переменной из информации о проекте.

Вызов:

int iret = ac\_request("autotext","get", string svarname);

Здесь: svarname - имя переменной в информации о проекте, которое нужно считать.

iret - 0, если считалось без ошибок.

Результат получается вызовом функции ac\_getstrvalue();

Пример.

Считать значение переменной "Адрес объекта" из информации о проекте.

string svaluename = "Адрес объекта";  
int iret = ac\_request("autotext","get", svarname);  
**if**( iret == 0)  
{  
 string saddress = ac\_getstrvalue();  
 cout << "Адрес = " << saddress;  
}  
**else**  
{  
 cout << "Считать адрес не удалось. Возможно переменная " << svaluename << " в проекте не создана";  
}

**set**

Задать значение переменой в информации о проекте.

Вызов:

int iret = ac\_request("autotext","set", string svarname, string svarvalue);

Пример.

Записать новое значение в переменную "Адрес объекта" из информации о проекте.

string svaluename = "Адрес объекта";  
string snewvalue;  
snewvalue = "г. Москва, Шаболовка, 37";  
int iret = ac\_request("autotext","set", svarname, snewvalue);  
**if**( iret == 0)  
{  
 cout << "Адрес успешно изменен";  
}  
**else**  
{  
 cout << "Записать адрес не удалось. Возможно переменная " << svaluename << " в проекте не создана";  
}

**create**

Создать в информации о проекте переменную с указанным именем и присвоить ей значение.

int iret = ac\_request("autotext","create", string svarname, string svalue);

Здесь:

svarname - имя новой переменной,

svalue - значение для этой переменной,

iret - результат выполнения функции. Если 0 - то переменная создана.

Пример.

Создать новую переменную "Адрес объекта" в информации о проекте и записать первоначальное значение.

string svaluename = "Адрес объекта";  
string snewvalue;  
snewvalue = "г. Москва, Шаболовка, 37";  
int iret = ac\_request("autotext","create", svarname, snewvalue);  
**if**( iret == 0)  
{  
 cout << "Переменная успешно создана";  
}  
**else**  
{  
 cout << "Создать переменную " << svaluename << " не удалось";  
}

**get\_as\_table**

Получить полную таблицу переменных из информации о проекте.

ac\_request("autotext", "get\_as\_table", int iTable, int type);

Здесь:

iTable - дескриптор таблицы для заполнения данными из проекта;

формат колонок таблицы на выходе:

"text" (string) - имена переменных;

"key" (string) - внутренние идентификаторы переменных;

"value" (string) - значения.

type - битовый флаг для фильтрации типов переменных из проекта - 1-фиксированные (fixed)/2-собственные(custom)/4 - прочие (other).

Можно суммировать, например 1+2 означает выбрать фиксированные и собственные переменные (см.пример).

Пример.

Выбрать в iTable все переменные из информации о проекте.

int iTable;

object("create", "ts\_table", iTable, "Autotext variables list");

int type=1+2;

ac\_request("autotext", "get\_as\_table", iTable, type);

**interface\_input2point**

Ввод пользователем точки.

Вызов:

int iret ac\_request("interface\_input2dpoint",string firstMessage, double &x, double &y);

Здесь:

sFirstMessage - подсказка пользователю при вводе точки,

x, y - координаты введенных точек,

Возвращает 0, если ввод был успешным.

**interface\_input2dline**

Ввод пользователем 2d линии.

Вызов:

int iret ac\_request("interface\_input2dline", string sFirstMessage, string sSecondMessage, double x1, double y1, double x2, double y2, double angleInRad, double length);

Здесь:

sFirstMessage, sSecondMessage - подсказка пользователю при вводе первой и второй точки,

x1, y1, x2, y2 - координаты введенных точек,

angleInRad - угол в радианах от положительного направления оси X,

length - длина отрезка.

Возвращает 0, если ввод был успешным.

**interface\_input3dline**

Ввод пользователем 3d линии.

Вызов:

int iret ac\_request("interface\_input3dline", string sFirstMessage, string sSecondMessage, double x1, double y1, double z1, double x2, double y2, double z2, double angleInRad, double length);

Здесь:

sFirstMessage, sSecondMessage - подсказка пользователю при вводе первой и второй точки,

x1, y1, z1, x2, y2, z2 - координаты введенных точек,

angleInRad - угол в радианах от положительного направления оси X,

length - длина отрезка

Возвращает 0, если ввод был успешным.

**interface\_input2dpoly**

Ввод пользователем 2d полигона. Возвращает площадь и периметр.

Формат запроса:

int ires = ac\_request("interface\_input2dpoly",string sMessage, double &square, double &perimeter);

Здесь:

sMessage - текстовое сообщение пользователю,

square, perimeter в эти переменные возвращается значение площади и периметра фигуры, которую задал пользователь кликами мыши

**get\_levelfromprojectnull\_by\_floorindexandlevel**

Зная индекс этажа и высоту от уровня этого этажа (может быть с минусом) получить высоту от ноля проекта.

Вызов:

ac\_request("get\_levelfromprojectnull\_by\_floorindexandlevel", int floorInd, double bottomOffset, double &levelfromprojectnull);

Здесь:

floorind - индекс этажа,

bottomOffset - смещение относительно уровня этого этажа,

levelfromprojectnull - результат - смещение по вертикали относительно ноля проекта.

Пример:

Выбрать элементы из проекта.

У элементов типа Морф определить расстояние середины элемента от уровня 0 проекта.

Вариант 1. Сначала выборка всех элементов, затем по ходу опроса элементов определяем нужные по типу.

// загрузить из проекта в список 1 все элементы, у которых присвоено любое значение классификатора ARCHICAD (не обязательно)

ac\_request\_special("add\_elements\_list", 1, "ZombieType", 2,

"", "Cls", "Классификация ARCHICAD", "ASSIGNED", "", "");

// запросить количество собранных элементов

ac\_request("get\_loaded\_elements\_list\_count", 1);

int icount = ac\_getnumvalue(); // получить количество в переменную

coutvar << icount; // вывести в окно сообщений

if (icount == 0)

{

cout << "В списке нет элементов";

return -1;

}

// объявляем необходимые переменные для работы

double dLevel, dHeight, elemlevel, elemprojectlevel;

int ielemstoryindex;

string sElemTypeName, sID;

int i;

int ires;

// выполнить цикл icount раз

for (i = 0; i < icount; i++)

{

ires = ac\_request("set\_current\_element\_from\_list", 1, i); // сделать текущим i-вый элемент из списка 1

ac\_request("get\_element\_value", "TypeName"); // запросить название типа этого элемента

sElemTypeName = ac\_getstrvalue();

coutvar << sElemTypeName; // вывести имя типа элемента в окно сообщений

// если это элемент типа Морф:

if (sElemTypeName == "MorphType")

{

ires = ac\_request("get\_element\_value", "ID"); // запросить ID элемента

sID = ac\_getstrvalue(); // получить его в переменную

coutvar << sID; // вывести ID в окно сообщений

ires = ac\_request("get\_element\_value", "Level"); // запрашиваем свойство Level (для Морф - высота над уровнем его этажа)

dLevel = ac\_getnumvalue(); // получаем его в переменную

coutvar << dLevel; // выводим в окно сообщений

ires = ac\_request("get\_quantity\_value", "max\_height"); // запрашиваем максимальный размер Морф по вертикали

dHeight = ac\_getnumvalue(); // получаем его в переменную

coutvar << dHeight; // выводим в окно сообщений

elemlevel = dLevel + dHeight / 2; // рассчитываем уровень центра Морф по высоте относительно этажа

ac\_request("get\_element\_value", "StoryIndex"); // запрашиваем индекс этажа

ielemstoryindex = ac\_getnumvalue(); // получаем индекс этажа в переменную

// получаем высоту центра Морф относительно 0 проекта в переменную elemprojectlevel

ac\_request("get\_levelfromprojectnull\_by\_floorindexandlevel", ielemstoryindex, elemlevel, elemprojectlevel);

coutvar << elemprojectlevel; // выводим в окно сообщений

}

}

Вариант 2. Сразу выбираем только элементы типа Морф, которые классифицированы как "Лестница"

// загрузить из проекта в список 1 элементы типа Морф, у которых присвоено значение классификатора ARCHICAD "Лестница"

ac\_request\_special("add\_elements\_list", 1, "MorphType", 2,

"", "Cls", "Классификация ARCHICAD", "=", "Лестница", "");

// запросить количество собранных элементов

ac\_request("get\_loaded\_elements\_list\_count", 1);

int icount = ac\_getnumvalue(); // получить количество в переменную

coutvar << icount; // вывести в окно сообщений

if (icount == 0)

{

cout << "В списке нет элементов типа Морф, классифицированных как \"Лестница\"";

return -1;

}

// объявляем необходимые переменные для работы

double dLevel, dHeight, elemlevel, elemprojectlevel;

string sID;

int ielemstoryindex;

int i;

int ires;

// выполнить цикл icount раз

for (i = 0; i < icount; i++)

{

ires = ac\_request("set\_current\_element\_from\_list", 1, i); // сделать текущим i-вый элемент из списка 1

ires = ac\_request("get\_element\_value", "ID"); // запросить ID элемента

sID = ac\_getstrvalue(); // получить его в переменную

coutvar << sID; // вывести ID в окно сообщений

ires = ac\_request("get\_element\_value", "Level"); // запрашиваем свойство Level (для Морф - высота над уровнем его этажа)

dLevel = ac\_getnumvalue(); // получаем его в переменную

coutvar << dLevel; // выводим в окно сообщений

ires = ac\_request("get\_quantity\_value", "max\_height"); // запрашиваем максимальный размер Морф по вертикали

dHeight = ac\_getnumvalue(); // получаем его в переменную

coutvar << dHeight; // выводим в окно сообщений

elemlevel = dLevel + dHeight / 2; // рассчитываем уровень центра Морф по высоте относительно этажа

ac\_request("get\_element\_value", "StoryIndex"); // запрашиваем индекс этажа

ielemstoryindex = ac\_getnumvalue(); // получаем индекс этажа в переменную

// получаем высоту центра Морф относительно 0 проекта в переменную elemprojectlevel

ac\_request("get\_levelfromprojectnull\_by\_floorindexandlevel", ielemstoryindex, elemlevel, elemprojectlevel);

coutvar << elemprojectlevel; // выводим в окно сообщений

}

**get\_floorindexandlevel\_by\_levelfromprojectnull**

Зная высоту от ноля проекта получить индекс этажа и смещение от этого этажа.

Вызов:

ac\_request("get\_floorindexandlevel\_by\_levelfromprojectnull", double levelfromprojectnull, int &floorInd, double &bottomOffset);

Здесь:

levelfromprojectnull - смещение по вертикали относительно ноля проекта.

floorind - результат - индекс этажа,

bottomOffset - результат - смещение относительно уровня этого этажа,

**get\_storyinfo\_as\_table**

Получить список этажей в таблицу.

Вызов:

int ierr = ac\_request("get\_story\_info\_as\_table", int iTableStories);

Здесь:

iTableStories - дескриптор таблицы, в которую будет записан список этажей.

ierr - результат операции (если 0 - успешно).

|  |  |  |
| --- | --- | --- |
| Имя колонки | Тип данных | Значение |
| index | int | индекс этажа |
| floorId | int | уникальный Id этажа |
| notMine | int | этаж принадлежит другому клиенту (для командных проектов) |
| dispOnSections | int | показывается на разрезах и видах |
| level | double | уровень этажа |
| name | string | имя |
| realfloornumber | int | реальный номер этажа от уровня земли |

Пример.

Считать список этажей в таблицу и показать в окне сообщений.

int iTableStories;

object("create", "ts\_table", iTableStories);

ac\_request("get\_story\_info\_as\_table", iTableStories);

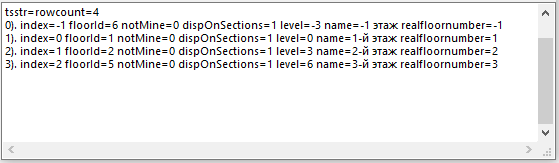
string tsstr;

ts\_table(iTableStories, "print\_to\_str", tsstr);

coutvar << tsstr;

object("delete", iTableStories);

Результат:



**elem\_classification\_get\_item\_value**

Получить имя класса элемента в заданной классификации.

Вызов:

int err = ac\_request("elem\_classification\_get\_item\_value",string sclassifname);

Здесь:

err = 0 если считано без ошибок,

sclassifname - имя классификатора,

Результат считывается командой ac\_getstrvalue();

**elem\_classification\_get\_parent\_item\_value**

Получить имя родительского класса элемента в заданной классификации.

Вызов:

int err = ac\_request("elem\_classification\_get\_item\_value",string sclassifname);

Здесь:

err = 0 если считано без ошибок,

sclassifname - имя классификатора,

Результат считывается командой ac\_getstrvalue();

**hotlinks**

Работа с модулями, вставленными в проект

**GetHotlinkNodes**

Получить список модулей, вставленных в проект.

Вызов:

int err = ac\_request("hotlinks","GetHotlinkNodes",int iTable);

Здесь:

err - возвращает отрицательное число или 0 если нет ошибки,

iTable - дескриптор таблицы, куда записать результат (колонки таблицы сформируются автоматически).

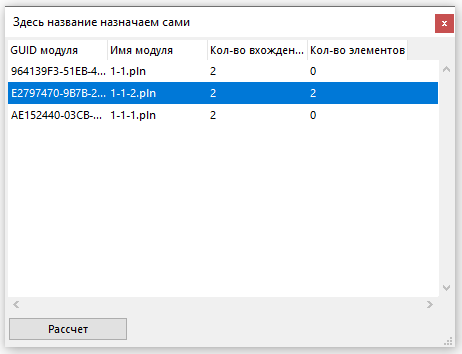
Формат таблицы:

|  |  |  |
| --- | --- | --- |
| Имя колонки | Тип данных | Значение |
| guid | string | guid базового модуля |
| name | stting | имя модуля |
| type | int | тип 0/1/2 - неизвестный/модуль типа проекта ARCHICAD/dwg или dxf и др. |
| refFloorName | string | Имя этажа, если привязан к одному этажу |
| refFloorInd | int | Индекс этажа, если привязан к одному этажу |
| instancesCount | int | Количество применений в проекте |

Пример.

Получить список ставленных в проект модулей и вывести таблицу в окно сообщений, а затем только колонки guid, name и instancesCount с другими заголовками вывести в элемент диалога iListBox.

По кнопке дать возможность посчитать количество элементов балок в модуле из выбранной строки.



//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Create dialog and listbox

// Get from project list of the modules

// Print list to message window

// Place 3 columns of the list to listbox

// Calc beams in modules by pressing button

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int iDialogDescr; // Дескриптор диалога

int iListBoxNodes, iTableNodes; // Листбокс элементов hotlink nodes

int iTable;

int iButtonCalcElementsCountInModule;

int main()

{

#pragma region Создаем диалог

int x, y, w, h;

object("create", "ts\_dialog", iDialogDescr);

ts\_dialog(iDialogDescr, "init\_dialog", "palette", 0, 0, 450, 310); // Создаем окно диалога как палитку, т.е. немодальное

ts\_dialog(iDialogDescr, "set\_as\_main\_panel"); // Если так сделать, то все немодальные окна этого сеанса будут закрываться вместе с этим окном

ts\_dialog(iDialogDescr, "SetTitle", "Здесь название назначаем сами");

// листбокс

object("create", "ts\_dialogcontrol", iListBoxNodes, "iListBoxNodes");

ts\_dialogcontrol(iListBoxNodes, "init\_control", "singlesellistbox", iDialogDescr, 3, 3, 450-3, 300-5-22, 48, 20);

ts\_dialogcontrol(iListBoxNodes, "SetAnchorToPanelResize", 0, 0, 1, 1);

// кнопка расчета для позиции диалога

int delta = 3;

int yy = 282;

x = 3; y = yy; w = 120; h = 20;

object("create", "ts\_dialogcontrol", iButtonCalcElementsCountInModule, "iButtonCalcElementsCountInModule");

ts\_dialogcontrol(iButtonCalcElementsCountInModule, "init\_control", "button", iDialogDescr, x, y, w, h);

ts\_dialogcontrol(iButtonCalcElementsCountInModule, "eventreaction", "Event\_ButtonClicked");

ts\_dialogcontrol(iButtonCalcElementsCountInModule, "settext", "Рассчет");

ts\_dialogcontrol(iButtonCalcElementsCountInModule, "SetAnchorToPanelResize", 0, 1, 0, 0);

ts\_dialogcontrol(iButtonCalcElementsCountInModule, "SetToolTip", "Рассчитать количество элеменов балок в модуле");

#pragma endregion

object("create", "ts\_table", iTable);

// получить список hotlinks - колонки таблицы сформируются автоматически

ac\_request("hotlinks", "GetHotlinkNodes", iTable);

// добавить в конец колонку "количество элементов"

ts\_table(iTable, "add\_column", -1, "int", "Кол-во элементов");

// вывести список hotlink nodes в окно сообщений

string str;

ts\_table(iTable, "print\_to\_str", str);

coutvar << str;

// задать имена колонок перед экспортом в листбокс

ts\_table(iTable, "mapping\_columns\_to\_export",

"guid", "GUID модуля", "",

"name", "Имя модуля", "",

"instancesCount", "Кол-во вхождений в проект", "",

"Кол-во элементов","","" // имя колонки оставляем как есть

);

// задать ширину колонок при экспорте

ts\_table(iTable, "set\_columns\_width\_when\_export", 100);

// экспортировать таблицу в листбокс

ts\_table(iTable, "export\_to\_dialogcontrol", iListBoxNodes, -1, -1);

//object("delete", iTable); если объект таблицы больше не потребуется, то можно ее удалить

//но если потом будем к ней обращаться из обработчиков событий (кнопки и т.п.) то оставляем в памяти

bool bres;

// запускаем диалог

ts\_dialog(iDialogDescr, "invoke", bres);

}

// обработчик событий кнопок на нажатие

int Event\_ButtonClicked(int iDescr, string sDescr)

{

if (sDescr == "iButtonCalcElementsCountInModule") {

cout << sDescr << "\n";

do\_iButtonCalcElementsCountInModule();

}

}

int do\_iButtonCalcElementsCountInModule()

{

// получить имя модуля, выбранного в листбоксе

int item;

string smodulename;

ts\_dialogcontrol(iListBoxNodes, "GetSelectedItem", item);

if (item == 0) {

return -1;

}

ts\_table(iTable, "select\_row", item - 1);

ts\_table(iTable, "get\_value\_of", "name", smodulename);

// выбрать элементы, у которых совпадает имя модуля с выбранным в листбоксе

ac\_request\_special("load\_elements\_list", 2, "BeamType", 2,

"", "EP", "hotlinkName", "=", smodulename, "");

ac\_request("get\_loaded\_elements\_list\_count", 2);

int icount = ac\_getnumvalue();

// записать результат в текущую строку таблицы

ts\_table(iTable, "set\_value\_of", "Кол-во элементов", icount);

coutvar << icount;

// экспортировать таблицу в листбокс (вид колонок уже был задан при первом экспорте в main())

ts\_table(iTable, "export\_to\_dialogcontrol", iListBoxNodes, -1, -1);

ts\_dialogcontrol(iListBoxNodes, "SelectItem", item);

}

**geometry\_calc\_2d**

**is\_point\_on\_element\_polygon**

Определить находится ли точка X,Y внутри полигона заданного элемента

int err = ac\_request("geometry\_calc\_2d","is\_point\_on\_element\_polygon",double X, double Y, int iElemDescr);  
или  
int err = ac\_request("geometry\_calc\_2d","is\_point\_on\_element\_polygon",double X, double Y, int iElemDescr, bool mainContOnly, double grow\_contour, double grow\_holes);

Здесь:

X,Y - координаты исследуемой точки;

iElemDescr - дескриптор элемента (не "ts\_guid" а именно "ac\_element");

mainContOnly - учитывать только внешний контур (без учета вырезов);

grow\_contour - на сколько расширить или сузить (если с минусом) внешний контур перед выполнением расчета;

grow\_holes - на сколько расширить или сузить (если с минусом) вырезы перед выполнением расчета;

Возвращает 0 при отсутствии ошибки.

Результат считывается следующей функцией ac\_getnumvalue() - если 1 - внутри полигона, 0 - снаружи (или в вырезе, если mainContOnly - true)

**is\_curelem\_inside\_element\_polygon**

Определить находится ли текущий элемент внутри контура, обозначенного другим полигональным элементом, при необходимости расширив его границы на заданное расстояние с целью исключения погрешностей моделирования.

Вызов:

int ires = ac\_request("geometry\_calc\_2d","is\_curelem\_inside\_element\_polygon", int iElemGuidDescr);   
или  
int ires = ac\_request("geometry\_calc\_2d","is\_curelem\_inside\_element\_polygon", int iElemGuidDescr, bool mainContOnly, double grow\_contour, double grow\_holes, int mode);

Здесь:

iElemmDescr - дескриптор объекта guid элемента ("ts\_guid").

mainContOnly - 0/1 - если 0, то анализ идет без учета внутренних вырезов, если 1 - то если элемент iElemDescr находится внутри выреа текущего элемента, то счтается, что он не находится внутри этого элемента.

grow\_contour - расстояние в мм, на которое необходимо расширить внешний контур текущего полигонального элемента перед тестированием.

grow\_holes - расстояние в мм, на которое необходимо сузить вырезы текущего полигонального элемента перед тестированием.

mode - режим тестирования - у элемента iElemDescr должны попасть внутрь контура: -1 - все точки контура, 0 - точка позиционирования (pos.x pos.y), 1 - хотя бы одна точка контура, 2 - хотя бы две точки контура.

**is\_point\_on\_element\_refside**

С какой стороны элемента находится заданная точка.

Вызов:

ac\_request("is\_point\_on\_element\_refside", int px, int py, int elemguid, bool &bIsOnRefSide);

Здесь:

px, py - анализируемая точка;

bIsOnRefSide - рзультат. Внешняя сторона стены или элемента, oppSide - внутренняя сторона стены или элемента.

**where\_C\_leftorright**

Определить с какой стороны вектора находится точка - справа или слева.

Выхов:

ac\_request("geometry\_calc\_2d","where\_C\_leftorright",double X1, double Y1, double X2, double Y2, double pX, double pY, double &dist);

Здесь:

X1, Y1,X2, Y2 - точки начала и конца вектора,

pX, pY - коодинаты тестируемой точки

dist - результат - расстояние точки от вектора. Меньше 0 - точка слева, 0 - точка на векторе, > 0 - точка справа.

Пример - если вектор снизу вверх, то если точка слева - то возвращает значение < 0, если точка на векторе - то 0, если точка справа - то >0.

**getDistPointFrom2dLine**

Получить расстояние от точки до прямой, заданной двумя точками.

ac\_request("geometry\_calc\_2d","getDistPointFrom2dLine",double X1, double Y1,double X2, double Y2, double pX, double pY, double &dist);

Здесь:

X1, Y1, X2, Y2 - координаты прямой;

pX, pY - координаты тестируемой точка;

dist - результат, расстояние точки от прямой;

**geometry\_calc\_3d**

**RotatePoint3dAroundVect3d**

Повернуть точку вокруг вектора на заданный угол.

Начало вектора - в точке 0,0,0.

ac\_request("geometry\_calc\_3d","RotatePoint3dAroundVect3d",double rx, double ry, double rz, double alpha, double x, double y, double z, double& resx, double& resy, double& resz);

Здесь:

rx,ry,rz - координаты вектора, вокруг которого осуществить поворот;

alpha - угол поворота;

x,y,z - координаты поворачиваемой точки;

resx, resy, resz - результат.

**GetBounds**

Получить размеры габаритного контейнера для текущего элемента.

Вызов:

err = ac\_request("geometry\_calc\_3d","GetBounds",double xMin, double yMin, double zMin, double xMax, double yMax, double zMax);

**classification**

Получить имя родительского класса элемента в заданной классификации.

Вызов:

int err = ac\_request("elem\_classification\_get\_item\_value",string sclassifname);

Здесь:

err = 0 если считано без ошибок,

sclassifname - имя классификатора,

Результат считывается командой ac\_getstrvalue();

**find\_item\_by\_name**

Получить имя класса, заданного текущему элементу в заданном классификаторе.

Вызов:

int err = ac\_request("classification","find\_item\_by\_name", string classifname, string itemname, int iGuidItem);

Здесь:

err = 0 если найдено и без ошибок,

sclassifname - имя классификатора,

itemname - имя класса,

iGuidItem - дескриптор объекта ts\_guid, в который запишется guid класса чтобы его можно было присваивать элементам модели ARCHICAD.

**check\_class\_is\_inbranch**

Проверить класс на его нахождение в ветке другого класса в заданном классификаторе.

Вызов:

bool bres = check\_class\_is\_inbranch(UniString classification, UniString classname1, UniString classnametest)

Здесь:

bres - true если класс classnametest находится в какой либо из веток класса clsaaname1.

classnametest - имя класса, который надо протестировать,

classname1 - имя класса, к которому проверяем что он является прародителем класса classnametest.

classification - имя классификатора, в котором это проверяется.

**CreateClassificationSystem**

Создать новый классификатор.

Вызов:

int err = ac\_request("classification","CreateClassificationSystem", int iguiddescr, string name, string description, string source, string editionVersion, int year, int month, int day);

Здесь:

err = 0 если создано без ошибок,

iguiddescr - дескриптор вновьсозданного классификатора,

name - имя создаваемого классификатора,

description - произвольное описание создаваемого классфикатора,

source - произвольный текст, описывающий источник данных,

editionVersion - произвольный текст, описывающий версию,

year - номер года,

month - номер месяца,

day - номер дня, которые требуется указать при создании классификатора.

**CreateClassificationItem**

Создать класс в классификаторе.

Вызов:

int err = ac\_request("classification","CreateClassificationItem", int iguiddescritem/string sguidresultitem, string id, string name, string description, int isystemguid/string ssystemguid, int iparentitemguid, int inextitemguid)

Здесь:

err = 0 если создано без ошибок,

iguiddescritem - дескриптор объекта ts\_guid куда будет записан guid создаваемого класса,

или

sguidresultitem - текстовая переменная, куда запишется guid создаваемого класса в текстовом виде,

id - id создаваемого класса,

name - имя создаваемого класса,

description - произвольное описание создаваемого класса,

isystemguid - дескриптор объекта ts\_guid с guid'ом классификатора, в котором создаем класс,

или

ssystemguid - guid классификатора, в котором создаем класс - в текстовом виде,

iparentitemguid - дескриптор guid'а родительского класса, к которому хотим создать подкласс (может быть -1, если создавать нужно в корневом разделе классификатора),

или

sparentitemguid - то же но guid в текстовом виде,

inextitemguid - дескриптор элемента, за которым поставить класс (может быть -1)

или

snextitemguid - то же, но в текстовом виде.

**DeleteClassificationItem**

Удалить класс по его guid.

Вызов:

int err = ac\_request("classification","DeleteClassificationItem", int iguiddescritem/string sguiditem)

Здесь:

err = 0 если удаление произошло без ошибок,

iguiddescritem - дескриптор объекта ts\_guid удаляемого класса,

или

sguiditem - guid класса в текстовом виде.

**set\_item\_to\_curelem**

Присвоить класс текущему элементу.

Вызов:

int err = ac\_request("classification","set\_item\_to\_curelem", iGuidItem);

Здесь:

err = 0 если присвоено без ошибок,

iGuidItem - дескриптор guid значения класса.

Перед обращением к этой функции надо создать объект типа ts\_guid. При его создании нам выдается дескриптор.

Затем находим в нужном классификаторе по имени нужный класс через "find\_item\_by\_name" и получаем guid класса.

**GetClassificationSystems**

Получить список классификаторов, установленных в проекте.

Вызов:

int iret = ac\_request("classification", "GetClassificationSystems", int iTableClassificatorsInProject);

Здесь:

iTableClassificatorsInProject - дескриптор таблицы, в которую будут записаны данные об установленных классификаторах

В iret - код ошибки или 0.

В iTableClassificatorsInProjectLoc формируется таблица с колонками:

"guid" - строка;

"name" - строка;

"description"- строка;

"source" - строка;

"editionVersion"- строка;

"editionDate" - дата.

Пример:

int iTableClassificatorsInProjectLoc;

object("create", "ts\_table", iTableClassificatorsInProjectLoc);

iret = ac\_request("classification", "GetClassificationSystems", iTableClassificatorsInProjectLoc);

**GetClassificationSystem\_guid\_by\_name**

Получить guid классификатора по его имени и версии.

Вызов:

int ires = ac\_request("classification", "GetClassificationSystem\_guid\_by\_name", string classifsysname, string classifsysver, int isystemguid/string ssystemguid);

Здесь:

classifsysname - имя классификатора,

classifsysver - версия классификатора,

isystemguid - дескриптор объекта ts\_guid в который вернется guid классификатора,

или

ssystemguid - текстовая переменная, куда вернется guid классификатора в текстовом виде.

Пример:

Создать класс в классификаторе "Структура ЦИМ".

string classifsysname = "Структура ЦИМ", classifsysver = "1.0";  
string systemguid;  
int ires = ac\_request("classification", "GetClassificationSystem\_guid\_by\_name", classifsysname, classifsysver, systemguid);  
**if** (ires == 0)  
{  
 string id, name, description;  
 id = "new id"; name = "new name"; description = "new descr";  
 string snewguid;  
 ac\_request("classification", "CreateClassificationItem", snewguid, id, name, description, systemguid, -1, -1);  
}

**Classification\_Import**

Импортировать в проект классификатор, представленный в виде xml текста.

iret = ac\_request("classification", "Classification\_Import", strinf sXMLsource, int iclassifpolicy, int ipropertypolicy);

Здесь:

sXMLsource - исходный текст классификатора в xml кодировке,

iclassifpolicy - как поступать при конфликтах во время загрузки классификатора:

0 - объединить с существующей классификацией,

1 - заменить,

2 - пропустить.

ipropertypolicy - как поступать при конфликтах во время загрузки параметров у классификации:

0 - объединить,

1 - пропустить.

Пример:

Получить с сайта текст классификатора из файла и загрузить его в проект.

int do\_iButtonloadClassificator()

{

string sclassifname = "Классификация КСИ- Oбъекты капитального строительства";

int iStringDescr;

object("create", "ts\_string", iStringDescr);

int iret = ts\_string(iStringDescr, "ReadFromURL", "http://www.labpp.ru/TSDownload/KSI/" + sclassifname + ".txt");

object("read\_simple\_value", iStringDescr);

string sXMLsource = ac\_getstrvalue();

if (iret == 0)

{

coutvar << sXMLsource;

iret = ac\_request("classification", "Classification\_Import", sXMLsource, 1, 0);

if (iret == 0)

{

tsalert(-3, "Сообщение", "Классификатор успешно загружен", sclassifname);

}

}

object("delete", iStringDescr);

refresh\_classificators\_list\_for\_loaded();

}

**GetClassifSystem\_By\_ClassTtemGuid**

Получить данные о классификаторе по guid позиции класса.

Вызов:

int err = ac\_request("classification", "GetClassifSystem\_By\_ClassTtemGuid", int/string guid, int/string classifsystemguid, string name, string description, string source, string editionVersion, string editionDate);

Здесь:

guid - GUID позиции класса, для которого надо узнать данные классификатора. Задается как дескриптор или текст;

classifsystemguid - результат - guid классификатора (текст или дескриптор);

name, description, source, editionVersion, editionDate - результат - данные классификатора.

**do\_move\_rot\_and\_scale\_elemlist**

Переместить или скопировать элементы, собранные в заданном списке на заданное расстояние, с учетом группировки, с поворотом на угол с масштабированием и записать вновь созданные элементы в заданный список.

ac\_request("do\_move\_rot\_and\_scale\_elemlist", int iElemList, double coordtoX, double coordtoY, double coordRotCenterX, double coordRotCenterY, double angle, double scale, bool bDoRotate, bool bWithDelete, int bpt, int mmode, int iElemList2);

Здесь:

iElemList - номер списка элементов, которые копировать или перемещать;

coordtoX, coordtoY - координаты куда переместить;

coordRotCenterX, сoordRotCenterY - координаты центра поворота;

angle - угол в радианах;

scale - масштаб (доли 1);

bDoRotate - поворачивать или нет;

bWithDelete - удалять оригиналы или нет;

bpt - что взять за базовую точку

0 - TopLeft,

1 - TopMiddle,

2 - TopRight,

3 - MiddleLeft,

4 - MiddleMiddle,

5 - MiddleRight,

6 - BottomLeft,

7 - BottomMiddle,

8 - BottomRight,

9 - Default,

10 - ByParentCoordCenter.

mmode - режим учета групп

каждый элемент по отдельности BySingle=0,

по последней группировке ByGroupsBelow=1,

по первой группировке ByGroupsTop=2

все как единое целое ByAll=3

iElemList2 - номер списка элементов, куда поместить вновь созданные элементы.

Пример

Скопировать элемент кровли на расстояние 1 м вправо, находящийся в списке 2 и записано созданный элемент в список 4,

и задать номер многослойной конструкции 111.

int iElemList = 2;

int iElemList2 = 4;

double coordtoX = 1;

double coordtoY = 0;

double coordRotCenterX = 0;

double coordRotCenterY = 0;

double angle = 0;

double scale = 1;

bool bDoRotate = false;

bool bWithDelete = false;

int bpt = 0;

int mmode = 0;

coordtoX = coordtoX + offset;

ac\_request("do\_move\_rot\_and\_scale\_elemlist", iElemList, coordtoX, coordtoY, coordRotCenterX, coordRotCenterY, angle, scale, bDoRotate, bWithDelete, bpt, mmode, iElemList2);

ac\_request("set\_current\_element\_from\_list", 4, 0);

int compositindex = 111;

ac\_request\_special("set\_element\_value", "EP","composite", compositindex);

**clearmessagewindow**

Очистить окно сообщений.

Вызов:

ac\_request("clearmessagewindow");

**get\_labpp\_product\_name**

Получить имя add-ons LABPP. Например, если скрипт запущен из программы LabPP\_Calc, то будет выдано "LabPP\_Calc".

Вызов:

ac\_request("get\_labpp\_product\_name");

Возвращает значение через вызов ac\_getstrvalue();

Пример:

ac\_request("get\_labpp\_product\_name");  
string sproductname = ac\_getstrvalue();  
**if** (sproductname != "LabPP\_Calc")  
{  
 ts\_dialogcontrol(iButton\_ToCalc\_DS, "disable");  
}

**ac\_request\_special**

**get\_pie**

Считать атрибуты пирога слоев материалов в таблицу 1, и слои в таблицу 2.

Вызов:

ac\_request2("get\_pie", int pieindex, int iTable1, int iTable2);

Здесь:

pieindex - индекс многослойной конструкции (пирога). Если задать -1, тогда грузим вообще все пироги, которые имеются в проекте;

iTable1 - дескриптор объекта ts\_table для заполнения данными пирога слоев (или пирогов слоев);

На выходе она содержит записи со следующими колонками:

"composite\_index" тип int - индекс пирога слоев в проекте;

"piename" тип string - название пирога;

"totalThick" тип double - общая толщина пирога;

"nComps" тип int - чсло слоев в пироге.

iTable2 - дескриптор объекта ts\_table для заполнения данными о составах пирогов слоев.

На выходе она содержит записи со следующими колонками:

"composite\_index" тип int - индекс пирога слоев, к которому относится слой;

"buildingmaterial" тип int - индекс строительного материала в проекте;

"pielayername" тип string - название строительного материала;

"ID" тип string - id стройматериала;

"thickness" тип double - толщина слоя;

"flagbits" тип int - набор битовых характеристик;

"leftright" тип int - описатель положения слоя по отношению к ядру пирога многослойки (слева отделка - 0, ядро - 1, справа отделка 2);

"cutMaterial" тип int - индекс покрытия;

"cutFill" тип int - индекс штриховки;

**load\_elements\_list**

Загрузить заданный список элементами ARCHICAD по заданным условиям

Вызов:

Первый вариант:

ac\_request("load\_elements\_list",int iListNum,string sElemTypeName,"MainFilter",int iMainFilterValue,string filterparametrname, string/double filterparametervalue,...);

Здесь:

iListNum - номер внутреннего списка элементов (от 0...9).

sElemTypeName - название типа элемента. Если нужно выбрать любые элементы - "ZombieElemType", если только колонны, то "ColumnType", стены - "WallType" и т.д.

"MainFilter" - сообщает, что мы задаем параметры для отфильтровывания элементов по типу доступности для редактирования, видимости и т.п. iMainFilterValue - обычно значение для видимых и редактируемых элементов - 3.

Далее идут названия параметров и их значения для дополнительного отбора.

Например "ID","Значение ID" - означает, что нужно выбрать элементы с ID="Значение ID" и т.д.

Фильтр "MainFilter" можно и не указывать. Это равносильно тому, что написать "MainFilter",0.

Т.е. при формировании списка в него попадут все подходящие элементы в проекте без учета основного критерия.

Значение фильтра собирается по формуле.

MainFilterValue=j1+2\*j2+4\*j3+8\*j4+16\*J5+32\*j6+64\*j7+128\*j8+256\*j9+512\*j10+1024\*j11+2048\*j12+4096\*j13+268435456\*j14,

где каждое j может быть 0 или 1.

j1: только редактируемые.

j2: на видимом слое.

j3: на текущем этаже

j4: имеет представление в 3d окне

j5: в моем рабочем пространстве

j6: не подчиненные а только независимые элементы

j7: на активном чертеже

j8: отображается внутри обрезанной части базы данных чертежа

j9: указывает, передан ли данному элементу идентификатор изменения в параметре variationID

j10: имеются права доступа к элементу

j11: элемент виден в реновации

j12: дополнительный флаг; проверяет, переопределены ли атрибуты элемента текущим фильтром обновления

j13: дополнительный флаг; проверяет видимость элемента с учетом текущей настройки отображения структуры

j14: только из 2d окна

Второй вариант:

ac\_request\_special("load\_elements\_list", int iListNum, string ElemTypeName, int iMainFilterValue,  
 string left\_braces1, string genesis1, string parname1, string operation1, string value1, string right\_braces1, string operation1and2,  
 string left\_braces2, string genesis2, string parname2, string operation2, string value2, string right\_braces2, string operation2and3,  
 ...  
 string left\_bracesN, string genesisN, string parnameN, string operationN, string parvalueN, string right\_bracesN);

Здесь:

left\_braces1 - открывающие скобки выражения, например "((".

right\_braces - закрывающие скобки выражения, например ")))".

Количество скобок должно быть четное и осмысленное.

genesis1 - происхождение сравниваемого параметра,

Возможны значения:

"ANY" - любое,

"GDL" - параметр GDL элемента,

"EP" - параметр элемента проекта,

"UP" - пользовательское свойство,

"UPb" - пользовательское свойство встроенное,

"UPub" - пользовательское свойство встроенное, созданное пользователем проекта,

"UPa" - пользовательское свойство встроенное,

"Cls" - параметр классификатора,

"IFC" - свойство UFC,

"UTS" - собственное свойство, назначенное элементу,

"Q" - количественное свойство

value1 - значение, с которым сравнивается значение параметра.

operation1and2 - операция с нижеследующими выражениями.

Может быть "И" (AND) или "ИЛИ" (OR).

Третий вариант:

ac\_request\_special("load\_elements\_list", int iListNum, string ElemTypeName, int iMainFilterValue, int iTableFilter);

Здесь:

iListNum - номер списка, в который нужно загрузить элементы,

ElemTypeName - низвание типа выбираемых элементов,

iMainFilterValue - значение главного фильтра элементов по видимости, доступности и т.п (см.выше).

iTableFilter - дескриптор таблицы, в которой заданы параметры фильтрации.

Этот вариант удобен, если нужно формировать сложные запросы фильтрации по ходу выполнения программы.

Пример:

Загрузить маркеры квартир в список №1 с этажей, у которых индекс 1 и 2.

int iTableFilter;

object("create", "ts\_table", iTableFilter);

ts\_table(iTableFilter, "create\_structure\_for\_filter\_table");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "EP", "ObjectName", "CONTAINS", "labelLabPP\_Flat", "", "AND");

// дополнить таблицу фильтра элементов индексами этажей

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "EP", "StoryIndex", "!=", 1, "", "AND");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "EP", "StoryIndex", "!=", 2, "", "");

ac\_request\_special("load\_elements\_list", 1, "ObjectType", 2, iTableFilter);

object("delete", iTableFilter);

**add\_elements\_list**

То же что "load\_elements\_list", но перед заполнением список не очищается а дополняется новыми элементами, удовлетворяющими условиям выборки.

**load\_elements\_list\_from\_selection**

То же что "load\_elements\_list", но выборка происходит из выбранных элементов.

**add\_elements\_list\_from\_selection**

То же что "load\_elements\_list", но выборка происходит из выбранных элементов и перед заполнением список не очищается а дополняется новыми элементами, удовлетворяющими условиям выборки.

**load\_elements\_list\_curdb**

То же, что и "load\_elements\_list" но при отборе не происходит переключение в другую базу данных.

**copy\_elements\_to\_list\_from\_list\_by\_filter**

То же, что и "load\_elements\_list", но отбор делается не из проекта, а из другого списка элементов.

**check\_cur\_element\_by\_filter**

То же что и "load\_elements\_list", но выборка делается не из проекта, а проверяется текущий элемент на соответствие условиям фильтра.

**GetMaterialSurfaceSquare**

Вычислить площадь поверхности элемента, покрытую заданным материалом.

Например, у элементов типа Morph можно назначать различным граням собственное поерытие (удерживайте CTRL+SHIFT и кликайте на нужные грани - они будут выбираться.

Затем в редакторе элемента задайте название материала).

Может быть сколько угодно типов покрытий у элемента Morph. Все их площади можно получить этой командой задавая название нужного материала.

Вызов:

int ires = ac\_request\_special("GetMaterialSurfaceSquare", string materialname);

или

int ires = ac\_request\_special("GetMaterialSurfaceSquare", int materialindex);

Здесь:

materialname - имя материала (не стройматериала, а материала, т.к. покрытия).

materialindex - индекс материала.

Возвращает -1 если имя материала не найдено.

Пример. Получить площадь поверхности у элемента Morph, у которой задано покрытие "Лестница - ступень".

ires = ac\_request\_special("GetMaterialSurfaceSquare","Лестница - ступень");

square = ac\_getnumvalue();

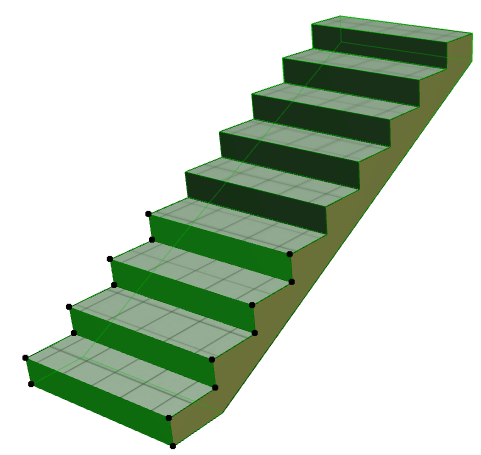
cout << "Площадь ступеней=" << square << "\n";

**GetMaterialSurfaceSquareTable**

Суммировать все площади всех граней элемента раздельно по назначенным материалам.

Например, у элементов типа Morph можно назначать различным граням собственное поерытие (удерживайте CTRL+SHIFT и кликайте на нужные грани - они будут выбираться.

Затем в редакторе элемента задайте название материала).



Все площади покрытий можно получить одной командой.

Результат выводится в таблицу.

Вызов:

int ires = ac\_request\_special("GetMaterialSurfaceSquareTable", int iTableDescr);

Здесь:

iTableDescr - таблица с результатом.

Перед вызовом функции надо создать объект таблицы.

Функция создаст колонки:

"materialindex" - тип целое число;

"materialname" - тип строка;

"square" - тип число с плавающей точкой.

Пример. Получить площади всех поверхностей у выбранного элемента Morph раздельно по материалам и вывести в окно сообщений.

int main()

{

ac\_request\_special("load\_elements\_list\_from\_selection", 1, "MorphType", 0);

ac\_request("get\_loaded\_elements\_list\_count",1);

int iсount = ac\_getnumvalue();

if (icount == 0)

{

cout << "Сначала выберите элемента типа Morph";

return;

}

ac\_request("set\_current\_element\_from\_list", 1, 0);

int iTableMat;

object("create", "ts\_table", iTableMat);

ac\_request\_special("GetMaterialSurfaceSquareTable", iTableMat);

string str;

ts\_table(iTableMat, "print\_to\_str", strrr);

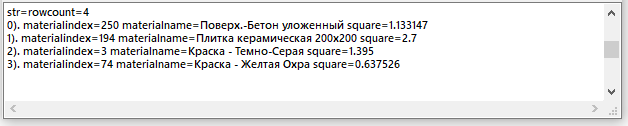
coutvar << str;

object("delete", iTableMat);

cout << "Программа успешно завершена";

}

Результат:



**get\_element\_value**

Считать значение свойства или параметра у текущего элемента.

**buildingMaterialName\_thickness**

Для текущего элемента рассчитать общую толщину стройматериалов многослойки или одного материала, которые использованы в назначении элементу при условии, что имя материала отвечает шаблону.

int err = ac\_request\_special("get\_element\_value", "EP", "buildingMaterialName\_thickness( \"(?=.\*Минерал.\*)(?=.\*ват.\*)(?=.\*плит.\*)\")",int iTableResult);

или

int err = ac\_request\_special("get\_element\_value", "EP", "buildingMaterialName\_thickness( \"(?=.\*Минерал.\*)(?=.\*ват.\*)(?=.\*плит.\*)\")");

Результат получаем через ac\_getnumvalue();

Если указан дескриптор таблицы, в нее записывается характеристики материалов, имена которых отвечают шаблону, а толщины одинаковых материалов складываются.

Т.е. если в многослойке есть один слой утеплителя одного типа и два слоя утеплителя другого типа - то в результате будет посчитана общая толщина утеплителя, а в таблицу выпадут две строки - утеплитель первого типа с толщиной, и утеплитель второго типа с суммарной толщиной его двух слоев.

**buildingMaterialName\_used**

Для текущего элемента рассчитать общую толщину стройматериалов многослойки или одного материала, которые использованы в назначении элементу при условии, что имя материала отвечает шаблону.

int err = ac\_request\_special("get\_element\_value", "EP", "buildingMaterialName\_used( \"(?=.\*Минерал.\*)(?=.\*ват.\*)(?=.\*плит.\*)\")",int iTableResult);

или

int err = ac\_request\_special("get\_element\_value", "EP", "buildingMaterialName\_used( \"(?=.\*Минерал.\*)(?=.\*ват.\*)(?=.\*плит.\*)\")");

Результат получаем через ac\_getnumvalue();

Если указан дескриптор таблицы, в нее записывается характеристики материалов, имена которых отвечают шаблону, а толщины одинаковых материалов складываются.

Т.е. если в многослойке есть один слой утеплителя одного типа и два слоя утеплителя другого типа - то в результате будет посчитана общая толщина утеплителя, а в таблицу выпадут две строки - утеплитель первого типа с толщиной, и утеплитель второго типа с суммарной толщиной его двух слоев.

Свойство может указываться в фильтрах для отбора строк по табличному фильтру.

|  |  |  |
| --- | --- | --- |
| Имя колонки | Тип колонки | Назначение |
| building\_material\_index | int | Индекс стройматериала в ARCHICAD |
| name | string | Имя стройматериала в ARCHICAD |
| id | string | ID стройматериала в ARCHICAD |
| thickness | double | Суммарная толщина стройматериала в пироге |

**set\_element\_value**

Задать значение свойства или параметра текущему элементу.

**set\_element\_value\_curdb**

То же, что и "set\_element\_value" но операция выполняется без переключения в другую базу данных.

**RunGDLParScript**

Выполнить скрипт у текущего GDL объекта.

Вызов:

int ires = ac\_request\_special("RunGDLParScript");

**get\_building\_material**

Получить информацию о стройматериале по его индексу в проекте, или все стройматериалы проекта.

Вызов:

int ires = ac\_request\_special("get\_material", int bmatindex, int iTableBMat);

Здесь:

iTableBMat - таблица, описывающая стройматериал.

Если указать в matindex конкретный индекс стройматериала, то будет только одна строка.

Если указать -1 - то будет записан весь список стройматериалов.

Структура таблицы на выходе:

"building\_material\_index" тип int - индекс стройматериала,

"name" тип string - название стройматериала,

"id" тип string - ID стройматериала,

"cutMaterial" тип string - материал для сечения стройматериала,

"strMaterialGuid" - guid стройматериала в текстовом виде.

**get\_build\_material\_value**

Получить значения свойст, назначенных строительному материалу.

Вызов:

int ires = ac\_request\_special("get\_build\_material\_value", int ibuildmatindex, string genesis, void &variable);

или

int ires = ac\_request\_special("get\_build\_material\_value", string sbuildmatname, string genesis, void &variable);

Здесь:

ibuildmatindex - индекс строительного материала,

sbuildmatname - название строительного материала.

genesis - имя происхождения свойства, например "UP" - свойство является пользовательским свойством.

variable - переменная, в которую вернется значение запрашиваемого параметра строительного материала.

Для получения значения может быть представлен дескриптор объекта типа ts\_table.

Пример:

Считать строительный материал текущего элемента, считать у этого материала значение параметра "Имя стройматериала на английском" и записать все это в пользовательские свойства текущего элемента.

int bmatindex;

string bmatname, bmatid, bmatnameenglish;

// считать у текущего элемента назначенный стройматериал

ac\_request("get\_element\_value", "buildingMaterial");

bmatindex = ac\_getnumvalue();

// получить имя этого стройматериала

err = ac\_request("get\_bmnamebyindex", bmatindex, bmatname, bmatid);

// получить имя этого стройматериала на английском из пользовательского свойства стройматериала

ac\_request\_special("get\_build\_material\_value", bmatindex, "UP", "Имя стройматериала на английском");

bmatnameenglish = ac\_getstrvalue();

// записать все эти свойства стройматериала в пользовательские свойства текущего элемента

ac\_request("elem\_user\_property", "set", "Имя материала у элемента", bmatname);

ac\_request("elem\_user\_property", "set", "Имя материала у элемента на английском", bmatnameenglish);

**get\_material**

Получить информацию о материале (покрытии) по его индексу в проекте, или все материалы (покрытия) проекта.

Вызов:

int ires = ac\_request\_special("get\_material", int matindex, int iTableMat);

Здесь:

iTableMat - таблица, описывающая материал (покрытие).

Если указать в matindex конкретный индекс материала, то будет только одна строка.

Если указать -1 - то будет записан весь список материалов (покрытий).

Структура таблицы на выходе:

"material\_index" тип int - индекс материала (покрытия),

"name" тип string - название материала (покрытия),

"texturepath" тип string - текстура,

"surfaceRGB" тип string - код RGB покрытия,

"surfaceRGB255" - тип string - RGB в последовательности через запятую (например "255,100,20").

**load\_zonerelatedwallstolist**

Загрузить во внутренний список стены, ассоциированные с заданной зоной.

Вызов:

int ires = ac\_request\_special("load\_zonerelatedwallstolist", int iZoneGuid, int ilistindex);

Здесь:

iZoneGuid - дескриптор объекта типа ts\_guid, содержащий guid зоны, у которой нужно получить список стен.

ilistindex - индекс внутреннего списка элементов, в который надо собрать эти стены.

**load\_zonerelatedwallparttotable**

Загрузить в таблицу список фрагментов стен, ассоциированные с заданной зоной.

Вызов:

int ires = ac\_request\_special("load\_zonerelatedwallstolist", int iZoneGuid, int iTableDescr);

Здесь:

iZoneGuid - дескриптор объекта типа ts\_guid, содержащий guid зоны, у которой нужно получить список фрагментов стен.

iTableDescr - дескриптор объекта типа ts\_table для получения фрагментов стен.

Функция создаст колонки:

"GUID" - тип строка - guid стены, к которой относится фрагмент;

"roomEdge" - тип целое - номер вершины у полигона зоны, к которому привязан фрагмент стены;

"tBeg" - тип число с плавающей точкой - стартовая позиция фрагмента стены, отсчитывая от точки начала стены.

"tEnd" - тип число с плавающей точкой - финишная позиция фрагмента стены, отсчитывая от точки начала стены.

**Attribute\_GetNum**

Получить количество атрибутов проекта заданного типа.

Вызов:

ac\_request2("Attribute\_GetNum", int iattrtype, int &icount);

Здесь:

iattrtype - тип атрибута.

Перья - 1,

Слои - 2,

Типы линий - 3,

Типы штриховок - 4,

Пироги материалов - 5,

Материалы - 6,

Города - 7,

Комбинации слоев - 8,

Категории зон - 9,

Шрифты - 10,

Профили - 11,

Таблицы перьев - 12,

Стандарты размеров - 13,

Список модельных видов - 14,

MEP системы - 15,

Энергетические назначения - 16,

Строительные материалы - 17,

Стили маркировки - 18

icount - возвращается количество атрибутов заданного типа.

**clipboard\_set**

Занести значение в буффер Windows. Записанное значение можно вставлять куда угодно штатными средствами Windows (Ctrl+V или Shift+Ins).

Вызов:

ac\_request\_special("clipboard\_set", string str);

Здесь:

str - переменная, содержащая текст, который необходимо занести в буффер Windows.

**clipboard\_get**

Считать строку из буффера Windows.

Вызов:

ac\_request\_special("clipboard\_get",string &str);

Здесь:

str - имя переменной, куда хотим записать содержимое из буффера Windows.

**element\_user\_data**

Помимо штатных свойств и параметров имеется возможность создавать любое количество собственных переменных и присваивать им значения.

В отличии от штатных свойств, эти переменные могут быть назначены не только элементам модели, но и штриховкам, линиям и т.п. элементам.

Порядок использования механизма для элемента простой.

Сделать элемент текущим.

Командой "read" считать содержимое уже имеющихся присвоенных ранее переменных этого элемента (обязательно, даже если раньше еще не создавались эти переменные).

Произвести манипуляции с переменными - создать переменную (присвоить значение), считать, удалить и т.п.

Затем, если необходимо зафиксировать изменения - выполнить команду "write".

В текстовце переменные можно записывать таблицы. Преобразование таблиц в текст - см. класс ts\_table.

**read**

Считать все пользовательские переменные текущего элемента (зоны, штриховки, линии и т.п.) для дальнейших манипуляций.

Вызов:

int ires = ac\_request\_special(""element\_user\_data","read");

**write**

Записать в текущий элемент все выполненные изменения с переменными.

Вызов:

int ires = ac\_request\_special("element\_user\_data","write");

**write\_to\_file**

Записать блок данных пользовательских данных в файл.

int ires = ac\_request\_special("element\_user\_data", "read\_from\_file", string fullpath);

Возвращает в ires 0 в отсутствии ошибок.

Если возникла ошибка - возвращяется -1.

fullpath - путь к файлу, откуда считать блок данных.

Пример.

Упаковать рабочие таблицы в блок пользовательских данных и записать блок в файл и обратная операция - прочитать из файла и распаковать в таблицы.

int WriteTablesToFile()

{

string str, tmppath, filename;

shell\_func("get\_path", "tmp", tmppath);

filename = "LabPP\_SAINT-GOBAIN.ini";

tmppath += "\\" + filename;

coutvar << tmppath;

ac\_request\_special("element\_user\_data", "clear");

ts\_table(iTableMaterials, "print\_to\_str\_as\_json", str);

coutvar << str;

ac\_request\_special("element\_user\_data", "set\_variable\_value", "iTableMaterials", str);

ts\_table(iTableSolutionsMats, "print\_to\_str\_as\_json", str);

ac\_request\_special("element\_user\_data", "set\_variable\_value", "iTableSolutionsMats", str);

ts\_table(iTableSolutions, "print\_to\_str\_as\_json", str);

ac\_request\_special("element\_user\_data", "set\_variable\_value", "iTableSolutions", str);

ts\_table(iTableSysProps, "print\_to\_str\_as\_json", str);

ac\_request\_special("element\_user\_data", "set\_variable\_value", "iTableSysProps", str);

int ires = ac\_request\_special("element\_user\_data", "write\_to\_file", tmppath);

if (ires < 0)

{

tsalert(-2, "Предупреждение", "Не удалось записать данные в файл " + tmppath, "Этот файл предназначен для быстрой загрузки данных GYPROC. Возможно этот путь не доступен для записи. Настройте config.cpp", "Ok");

return -1;

}

}

int ReadTablesFromFile()

{

string str, tmppath, filename;

filename = "LabPP\_SAINT-GOBAIN.ini";

shell\_func("get\_path", "tmp", tmppath);

tmppath += "\\"+filename;

int ires = ac\_request\_special("element\_user\_data", "read\_from\_file", tmppath);

if (ires < 0)

{

tsalert(-2, "Предупреждение", "Не удалось считать данные из файла " + tmppath, "Этот файл содержит информацию GYPROC из файла Excel для быстрой загрузки", "Ok");

return -1;

}

ac\_request\_special("element\_user\_data", "get\_variable\_value", "iTableMaterials", str);

cout << "-------------------------------------\n";

coutvar << str;

ts\_table(iTableMaterials, "load\_from\_json\_str", str);

ac\_request\_special("element\_user\_data", "get\_variable\_value", "iTableSolutionsMats", str);

ts\_table(iTableSolutionsMats, "load\_from\_json\_str", str);

ac\_request\_special("element\_user\_data", "get\_variable\_value", "iTableSolutions", str);

ts\_table(iTableSolutions, "load\_from\_json\_str", str);

ac\_request\_special("element\_user\_data", "get\_variable\_value", "iTableSysProps", str);

ts\_table(iTableSysProps, "load\_from\_json\_str", str);

}

**clear**

Очистить информацию обо всех переменных.

Вызов:

int ires = ac\_request\_special("element\_user\_data","clear");

**delete\_variable**

Удалить собственную переменную у текущего элемента.

Вызов:

int ires = ac\_request\_special("element\_user\_data","delete\_variable",string varname);

Здесь:

varname - имя удаляемой переменной.

**get\_variable\_value**

Считать значение собственной переменной текущего элемента.

Вызов:

int err = ac\_request\_special("element\_user\_data","get\_variable\_value",string varname, void &varnametovalue);

Здесь:

varname - имя переменной;

varnametovalue - переменная соответствующего типа, куда считать значение.

err - при успешном считывании значения - 0.

**set\_variable\_value**

Задать значение переменной у текущего элемента. Если переменная еще не создана, то в памяти создается переменная с указанным именем и типом, соответствующим типу переменной, примененной задающей значение.

Вызов:

ac\_request\_special("element\_user\_data","set\_variable\_value",string varname, void &varvalue);

Здесь:

varname - имя переменной, которой необходимо задать значение;

varvalue - переменная любого типа, чье значение будет присвоено задаваемой переменной элемента.

**linkingElems**

Удобно связывать элементы друг с другом для выполнения разнообразных операций.

Эти связи хранятся в проекте. Так что связанные элементы можно потом найти, связать, отвязать и т.д.

Связь односторонняя. Т.е. один элемент может быть связан с другим, а этот другой с первым - нет.

Связь характеризуется битами. Например у числа 1 - установлен последний бит. А у числа 2 - установлен предпоследний бит.

Можно устанавливать или разрывать связи по этим битам одновременно, если указать число 3 (оба бита установлены в 1).

Или по раздельности.

По ходу выполнения программ можно получать связанные элементы, связывать их двусторонней связью (от элемента другому - 1 а от того к первому - 2).

Для этих манипуляций имеются удобные функции, описанные ниже.

**uplinkBiWardByFlags**

Присоединить/отсоединить элемент к списку элементов по заданному биту и одновременно присоединить/отсоединить элементы списка к элементу по другому заданному биту.

Вызов:

int err = ac\_request\_special("linkingElems", "uplinkBiWardByFlags", int iGuid, int bitvalue\_from\_to, bool bFromToOn, int bitvalue\_to\_from, bToFromOn, int iTableGuids);

Здесь:

iGuid - дескриптор объекта типа ts\_guid, содержащего guid элемента, который надо связать/отвязать с/от списка элементов.

bitvalue\_from\_to - целое число, которое задает в двоичном виде какие биты связи от элемента к элементам списка, заданного в таблице iTableGuids надо установить (при bToFromOn==true) или снять (при bFromToOn==false).

bitvalue\_to\_from - целое число, которое задает в двоичном виде какие биты связи от элементов списка, заданного в таблице iTableGuids к элементу iGuid надо установить (при bFromToOn==true) или снять (при bToFromOn==false);

iTableGuids - дескриптор объекта типа ts\_table, в котором содержатся текстовые guid элементов для привязки/отвязки.

**getLinkedElemsByFlags**

Получить список текстовых guid элементов, связанных с заданным элементом по определенному флагу.

Вызов:

ac\_request\_special("linkingElems", "getLinkedElemsByFlags", iGuid, int bitvalue\_from\_to, int iTable);

Здесь:

iGuid - дескриптор объекта типа ts\_guid, который указывает на элемент.

bitvalue\_from\_to - число, задающее биты, по которым установлена связь этого элемента с другими элементами.

iTable - дескриптор объекта типа ts\_table, в который возвращается список текстовых guid связанных элементов.

**set\_TSMAX\_ELEMLIST\_NUMBER**

Увеличить количество списков для выборок элементов.

Если для реализации алгоритма недостаточно 10-ти выделенных списков для выборки элементов, то можно увеличить их число до необходимого.

Вызов:

ac\_request\_special("set\_TSMAX\_ELEMLIST\_NUMBER", int ineeded\_count);

Здесь:

ideeded\_count - желаемое количество внутренних списков элементов.

Пример.

Задать количество списков для выборки = 100, если это уже не сделано ранее.

int iicount;

ac\_request\_special("get\_TSMAX\_ELEMLIST\_NUMBER", iicount);

if (iicount < 100)

{

ac\_request\_special("set\_TSMAX\_ELEMLIST\_NUMBER", 100);

}

coutvar << iicount;

**get\_TSMAX\_ELEMLIST\_NUMBER**

Считать количество списков для выборок элементов.

Если для реализации алгоритма недостаточно 10-ти выделенных списков для выборки элементов, то можно увеличить их число до необходимого.

Чтобы узнать достаточно ли выделено уже списков, используем эту функцию.

Вызов:

ac\_request\_special("get\_TSMAX\_ELEMLIST\_NUMBER", int &icurrent\_lists\_count);

Здесь:

icurrent\_lists\_count - текущее количество внутренних списков элементов.

**set\_elemlistname**

Задать списку элементов имя. Если имя уже задействовано для другого номера списка, и если для заданного номера уже задано имя - старые назначения удаляются, а назначается новое.

Назначение имени позволяет обращаться в любых командах вместо номера списка - по его имени.

Вызов:

ac\_request\_special("set\_elemlistname", int ilistnum, string slistnewname);

Здесь:

ilistnum - номер списка, которому задается имя.

slistnewname - новое имя списка.

**Get\_Elements\_Collisions**

Проверка списков элементов на коллизии.

Вызов:

int ires = ac\_request\_special("Get\_Elements\_Collisions",int iList1, int iList2, int iTable\_res,  
 double volumeTolerance, bool performSurfaceCheck, double surfaceTolerance);

Здесь:

iList1, iList2 - номера списков элементов, в которых собраны элементы, проверяемые на коллизии;

iTable\_res - дескриптор таблицы, в которую будут записываться результаты проверки.

|  |  |  |
| --- | --- | --- |
| Имя колонки | Тип | Назначение |
| GUID1 | строка | guid элемента 1 |
| TypeName1 | строка | тип элемента 1 |
| ID1 | строка | ID элемента 1 |
| hasBodyCollision1 | bool | есть коллизия по объему |
| hasClearenceCollision1 | bool | есть коллизия по зазору |
| GUID2 | строка | guid элемента 2 |
| TypeName2 | строка | тип элемента 2 |
| ID2 | строка | ID элемента 2 |
| hasBodyCollision2 | bool | есть коллизия по объему |
| hasClearenceCollision2 | bool | есть коллизия по зазору |

double volumeTolerance - точность по объему,

bool performSurfaceCheck - делать или нет поверхностное тестирование,

double surfaceTolerance - точность по поверхностям;

**delete\_elements\_from\_list**

Удалить заданное количество позиций элементов в заданном списке начиная с заданного индекса.

Вызов:

ac\_request\_special("delete\_elements\_from\_list", int/string listindex/listname, int start\_index, int count);

Здесь:

listindex или listname - индекс или имя внутреннего списка элементов, в котором надо удалить записи.

start\_index - индекс, начиная с которого надо удалить записи элементов.

count - количество удаляемых записей элементов.

**delete\_elements\_from\_project**

Удалить элементы, указанные в заданном списке из проекта и очистить список.

Вызов:

int err = ac\_request\_special("delete\_elements\_from\_project", int/string listindex/listname);

Здесь:

listindex или listname - индекс или имя внутреннего списка элементов, в котором надо удалить записи.

start\_index - индекс, начиная с которого надо удалить записи элементов.

count - количество удаляемых записей элементов.

Возвращает 0 при успешном выполнении удаления элементов из проекта.

**ac\_request\_attributes**

**Таблица типов атрибутов**

|  |  |  |
| --- | --- | --- |
| Текстовое значение | Числовое значение | Назначение |
| "Pen" | 1 | Перо |
| "Layer" | 2 | Слой |
| "LineType" | 3 | Тип линии |
| "FillType" | 4 | Тип штриховки |
| "CompWall" | 5 | Многослойка |
| "Material" | 6 | Покрытие |
| "City" | 7 | Город |
| "LayerComb" | 8 | Комбинация слоев |
| "ZoneCat" | 9 | Категория зон |
| "Font" | 10 | Шрифт |
| "Profile" | 11 | Профиль |
| "PenTable" | 12 | Таблица перьев |
| "DimStand" | 13 | Стандарт размеров |
| "ModelViewOptions" | 14 | Опции модельного вида |
| "MepSystem" | 15 | MEP система |
| "OperationProfile" | 16 | Операционный профиль |
| "BuildingMaterial" | 17 | Стройматериал |
| "MarkUpStyle" | 18 | Стиль отметки |

**create\_attr**

Создать атрибут (перо, покрытие, стройматериал и т.п.) заданного типа с заданным именем на основании заданного другого атрибута.

Вызов:

int err = ac\_request\_attributes("create\_attr", int/string attrtypeid, string name, string id, int/strnig baseattr);

Здесь:

attrtypeid - [тип атрибута;](#_topic_Newtopic36)

name - название атрибута;

id - id атрибута

baseattr - название атрибута, взятого за основу.

**get\_num**

**delete\_attr**

Удалить заданный атрибут заданного типа.

Вызов:

int ires = ac\_request\_attributes("delete\_attr", int/string attrtypeid, int attr\_index);

Здесь:

attrtypeid - [тип атрибута;](#_topic_Newtopic36)

attr\_index - индекс атрибута

Возвращает 0 при успешном выполнении операции.

**set\_attr\_value**

Задать значение свойству атрибута.

Вызов:

int ires = ac\_request\_attributes("set\_attr\_value", int/string attrtype, int attrindex/string attrname, int/string genesis, string/bool/int/double value);

Здесь:

attrtypeid - [тип атрибута;](#_topic_Newtopic36)

attrindex или attrname - индекс или имя типа атрибута;

genesis - [генезис свойства](#_topic_Newtopic37);

value - присваиваемое значение.

Возвращает 0 при успешной записи.

**get\_attr\_value**

Получить значение свойства атрибута.

Вызов:

int ires = ac\_request\_attributes("get\_attr\_value", int/string attrtype, int attrindex/string attrname, string/int genesis, string/bool/int/double value);

Здесь:

attrtypeid - [тип атрибута;](#_topic_Newtopic36)

attrindex или attrname - индекс или имя типа атрибута;

genesis - [происхождение атрибута (генезис)](#_topic_Newtopic37);

Считывание результата производится фунцией ac\_getnumvalue() или ac\_getstrvalue().

Возвращает 0 при успешной записи.

Пример.

Считать значение свойства description у стройматериала с заданым индексом

int building\_material\_index = 12;  
ac\_request\_attributes("get\_attr\_value", "BuildingMaterial", building\_material\_index, "AP", "description");  
string description = ac\_getstrvalue();

coutvar << description;

**ac\_typeidfromstring**

Получить код типа элемента по его названию.

Вызов:

int itype = ac\_typeidfromstring(string selemtypename);

Здесь:

selemtypename - текстовое название типа элемента ("ZoneType", "MeshType", "ObjectType" и т.д.)

На выходе выдается числовое значение, соответствующее этому названию.

**ac\_getresvaluetype**

Определяет тип данных результата, полученного при выполнении предыдущей операции.

Вызов:

string stype = ac\_getresvaluetype();

В stype будет содержать "String"/"Real"/"Integer"/"Boolean" если результат предыдущей операции - строка, вещественное число, целое число или логический 1/0.

**ac\_getstrvalue**

Возвращает текстовое значение предыдущей операции.

Вызов:

string sresult = ac\_getstrvalue();

**ac\_getnumvalue**

Возвращает числовое значение предыдущей операции.

Вызов:

double dresult = ac\_getnumvalue();

**ac\_getupvaluestatus**

Определить после операции считывания пользовательского свойства - статус его значения.

Статус может быть Null (пусто), Undefined (не присвоено) или Normal.

В любых обстоятельствах при выполнении операций считывания, даже при ошибках, возвращается хотя бы 0 или пустая строка.

О произошедших ошибках, например отсутствии такого свойства у элемента, операции считывания сообщают возвращаемым кодом операции - 0 - успешно, а отрицательное значение - ошибка.

Чтобы понять что свойство является Null или Undefined - требуется выполнить дополнительно эту команду.

Null и Undefined, т.е. 0 и 1 являются близкими и по сути отвечают смыслу Null.

Лучше ориентироваться на значение 2 - т.е. если параметр имеет нормально присвоенное значение - то свойство не Null.

Вызов:

int ires = ac\_getupvaluestatus();

Здесь:

ires - Возвращаемое значение:

0 - было считано пустое значение (Null);

1 - значение свойству не было присвоено пользователем (Undefined);

2 - свойство имеет обычное нормальное значение (Normal);

Пример.

Считать свойство и понять оно имеет значение или оно Null.

ac\_request\_special("get\_element\_value", "UP", "Переменные водосточной системы\\Координаты элементов водосточной системы");  
string svalue = ac\_getstrvalue();  
int ivalue = ac\_getupvaluestatus();  
**if**(ivalue == 2) {  
 cout << "Свойство имеет нормальное значение";  
}  
**else**  
{  
 cout << "Свойство не имеет присвоенного значения или является Null";  
}

**Гравитация на поверхность**

**do\_elements\_landing - приземление элементов**

Можно выбрать элементы и "приземлить" их на поверхность других элементов.

Обращение

ac\_request("do\_elements\_landing", int iLandList, int iLandingElemsList, double doffset, int mmode);

Здесь:

iLandList - номер внутреннего списка элементов, где выбраны элементы, задающие поверхность приземления.

iLandingElemsList - номер внутреннего списка элементов, где выбраны элементы, которые нужно приземлить.

doffset - высота над поверхностью, на которой остановить приземление.

mmode - 0 - приземлять каждый элемент по отдельности, 1 - приземлять с учетом группирования по младшей группе, 2 - с учетом группирования по старшей группе, 3 - группа элементов как единое целое.

**do\_surface\_landing - приземление по точкам**

Можно выбрать элементы 3d сетки и "приземлить" точки их поверхностей на поверхность других элементов.  
Этой же командой приземляются элементы типа "Балка" обеими точками с наклоном по криволинейной поверхности.

Вызов:

ac\_request("do\_surface\_landing", int iLandList, int iLandingElemsList, double doffset);

Здесь:

iLandList - номер внутреннего списка элементов, где выбраны элементы, задающие поверхность приземления.

iLandingElemsList - номер внутреннего списка элементов, где выбраны элементы, которые нужно приземлить.

doffset - высота над поверхностью, на которой остановить приземление.

**do\_point\_landing - приземление точки X,Y**

Можно задать координаты X и Y и получить координату Z на поверхности, составленной элементами, собранными в списке iLandList. Если точка лежит не над поверхностью "земли" или находится над отверстием, то приземление не выполняется.

Вызов:

ac\_request("do\_point\_landing", int iLandList, double doffset, double x1,double y1, double &z1);

Здесь:

x1,y1 - исходные координаты точки.

doffset - остаточное смещение над поверхностью.

z1 - результат приземления.

iLandList - номер внутреннего списка с элементами - земля.

Пример.

Выбрать в список №1 любвн элементы со слоя "My land" и приземлить точку с координатами x=1, y=1 на высоту 0, т.е. прямо на поверхность.

ac\_request("load\_elements\_list,1,"ZombieElemType","Layer","My land","MainFilter",3);  
  
double doffset=0;  
double x=1, y=1, z=0;  
ac\_request("do\_point\_landing", 1, doffset, x,y,z);  
cout << "Результат - координата z на поверхности = "<< z;

**3d сетки**

**get\_lands\_combined\_contour\_coords**

Сформировать таблицу 2d линий объединенного контура множества геометрических элементов типа 3d сетка, плита, штриховка, крыша, зона, полилиния (т.е. контуры которых ARCHICAD хранит полилинией).

Вызов:

ac\_request("get\_lands\_combined\_contour\_coords", int iElementsList, int iTablePointsXY\_S);

Здесь:

iElementsList - номер списка элементов, для которых требуется собрать контур;

iTablePointsXY\_S - дескриптор таблицы, в которую будет записан список точек контура.

Структура таблицы будет сгенерирована автоматически.

Колонки: "x","y" - числовые 2d координаты точек объединенного контура;

"s" - числовой флаг для отслеживания вложенных контуров вырезов (зарезервировано).

Пример.

Получить общий контур из выделенных элементов типа 3d сетка и показать точки в виде таблицы в окне сообщений.

int iTable\_Coords2DTableContour;

object("create", "ts\_table", iTable\_Coords2DTableContour);

ac\_request\_special("load\_elements\_list\_from\_selection", 1, "MeshType", 2);

ac\_request("get\_loaded\_elements\_list\_count", 1);

int icount = ac\_getnumvalue();

cout << "Отобрано элементов для анализа = " << icount << "\n";

ac\_request("get\_lands\_combined\_contour\_coords", 1, iTable\_Coords2DTableContour);

string str;

ts\_table(iTable\_Coords2DTableContour, "print\_to\_str", str);

cout << "Контур\n";

coutvar << str;

**do\_create\_contour\_around\_points**

Создать контур вокруг набора точек.

Вызов:

ac\_request("do\_create\_contour\_around\_points", int iTablePointsXYZ, int iTablePointsXY\_S, bool bJustSquare, double& dMinZ);

Здесь:

iTablePointsXYZ - дескриптор таблицы точек, вокруг которых сделать контур;

iTablePointsXY\_S - дескриптор таблицы, куда записать результат;

bJustSquare - если true, то контур - габаритный прямоугольник, если false - контур - наименьший выпуклый многоугольник.

dMinZ - возвращается минимальная высота среди предоставленных точек;

**get\_lands\_topcoords**

Получить в таблицу список точек геометрии верхней поверхности множества элементов, собранного в списке.

Затем таблица используется, например, для построения 3d сеток в качестве списка точек или списка линий.

Вызов:

ac\_request("get\_lands\_topcoords", int iElementsList, int iTablePointsXYZ\_s\_polynum, int iMode);

Здесь:

iElementsList - номер списка элементов, из которого надо выбрать точки поверхностей;

iTablePointsXYZ\_s\_polynum - дескриптор таблицы, в которую записать результат;

Структура генерируется автоматически.

Колонки:

"x","y","z" - координаты 3d точек поверхности;

"s" - числовой флаг для отслеживания вложенных контуров вырезов (зарезервировано);

"polynum" - уникальный номер полигона или линии в таблице.

iMode - вид, в котором получаем данные о точках - 0/1/2/3.

0 - в таблицу заносятся все точки контуров верхних граней поверхностей элементов. В поле "polynum" записывается порядковый номер грани в рамках списка. Используется для наиболее точного воссоздания поверхностей за счет абсолютно точного воспроизведения всех граней.

1 - в таблицу заносятся только уникальные точки, т.е. для смежных граней точки смежных ребер будут указаны только один раз.

2 - заносятся пары точек для каждой линии, задающей ребро, номера ребер в рамках списка уникальные.

3 - заносятся пары точек для двух линий, задающих ребра каждой грани - номера ребер в рамках списка уникальные.

При значении iMode 0,2 или 3 таблица может использоваться как источник данных для построения 3d сеток по линиям.

Пример.

Записать выделенные элементы 3d сеток в список №1 и

получить список уникальных точек верхних поверхностей в таблицу.

Вывести эту таблицу в окно сообщений.

ac\_request\_special("load\_elements\_list\_from\_selection", 1, "MeshType", 2);

int iTable\_Coords3DTable;

object("create", "ts\_table", iTable\_Coords3DTable);

ac\_request("get\_loaded\_elements\_list\_count", 1);

int icount = ac\_getnumvalue();

cout << "Отобрано элементов для анализа = " << icount << "\n";

string str;

ac\_request("get\_lands\_topcoords", 1, iTable\_Coords3DTable, 1);

ts\_table(iTable\_Coords3DTable, "print\_to\_str", str);

cout << "\nКоординаты\n";

coutvar << str;

**do\_create\_mesh\_by\_points\_and\_lines**

Создать элемент 3d mesh (3d сетка) из таблицы точек и/или таблицы линий уровня по таблице контура.

Вызов:

ac\_request("do\_create\_mesh\_by\_points\_and\_lines", int iTable\_Coords3DPointsTable, int iTable\_Coords2DTableContour, int iTable\_Coords3DPointsTable);

Здесь:

iTable\_Coords3DPointsTable - дескриптор таблицы, описывающей 3d точки уровня поверхности (может быть -1, если уровни заданы только линиями);

iTable\_Coords2DTableContour - дескриптор таблицы, описывающей 2d точки контура. Если указать -1 - то контуром будут точки будут рассчитаные автоматически по таблицам точек уровня и линий;

iTable\_Coords3DPointsTable - дескриптор таблицы, описывающей 3d линии уровней (может быть -1, если уровни задаются только точками).

Пример.

Построить общую поверхность для набора выделенных элементов в виде 3d сетки, у которой координаты уровней заданы точками.

int iTable\_Coords3DTable;

object("create", "ts\_table", iTable\_Coords3DTable);

int iTable\_Coords2DTableContour;

object("create", "ts\_table", iTable\_Coords2DTableContour);

ac\_request\_special("load\_elements\_list\_from\_selection", 1, "MeshType", 2);

ac\_request("get\_loaded\_elements\_list\_count", 1);

int icount = ac\_getnumvalue();

cout << "Отобрано элементов для анализа = " << icount << "\n";

ac\_request("get\_lands\_combined\_contour\_coords", 1, iTable\_Coords2DTableContour);

string str;

ts\_table(iTable\_Coords2DTableContour, "print\_to\_str", str);

cout << "Контур\n";

coutvar << str;

ac\_request("get\_lands\_topcoords", 1, iTable\_Coords3DTable, 1);

ts\_table(iTable\_Coords3DTable, "print\_to\_str", str);

cout << "\nКоординаты\n";

coutvar << str;

ac\_request("do\_create\_mesh\_by\_points\_and\_lines", iTable\_Coords3DTable, iTable\_Coords2DTableContour,-1);

object("delete", iTable\_Coords2DTableContour);

object("delete", iTable\_Coords3DTable);

**Диалоги**

**Сообщения или выбор варианта - tsalert**

Функция вызывает диалог выбора или просто сообщение. Можно создать до трех кнопок включительно.

Формат обращения:

int res = tsalert(int messagecode,string stitle,string smessagebig, string ssmall,string button1,string button2,string button3);

Здесь:

messagecode - числовой код вида окна:

-1 - сообщение об ошибке

-2 - предупреждение

-3 - информационное сообщение.

stitle - заголовок окна, smessagebig - сообщение крупными буквами, ssmall - сообщение ниже мелким текстом, button1 - button3 - тексты для соответствующих кнопок.

В res возвращается результат выбора пользователя.

Если нажата кнопка button1 - результат 1, button2 - 2, button3 - 3. Если пользователь отказался от выбора - выдается 0.

Пример.

Получить от пользователя решение сколько выводить элементов (10, 1 или все). Пояснить что можно попробовать. если пользователь выберет "Все" - сообщить в окно сообщений. Если откажется - тоже сообщить в окно сообщений.

int res = tsalert(-3, "Задайте значение", "Сколько выводить элементов?", "Для пробной выгрузки удобно вывести сначала не все маркеры", "10", "1", "Все");  
**if** (res == 0)  
{  
 cout << "Пользователь отказался";  
 **return**;  
}

**Поиск и выбор файла - dialog\_get\_filename**

Функция ac\_request с директивой dialog\_get\_filename вызывает диалог выбора файла.

Формат обращения:

int iret = ac\_request("dialog\_get\_filename",string Title, string Filter, string sStartFolder, string for\_what, string &FileNameAndPath);

Здесь: Title - заголовок окна, Filter - строка фильтра для выбора файла, sStartFolder - начальный путь где выбирать файл.

Полный путь к выбранному файлу возвращается в переменную FileNameAndPath.

for\_what - текстовая переменная, задающая характер обращения к диалогу "OpenFile" или "Save".

Если iret будет -1 - это означает, что пользователь отказался от выбора.

Пример.

Выбрать программный файл с расширением .cpp. Искать сначала в корневом каталоге диска C:. Дополнительно - запустить этот выбранный программный файл на выполнение.

string sFileNameAndPath;  
string sStartFolder = "c:\\";  
int iret = ac\_request("dialog\_get\_filename", "Выберите файл для запуска", "cpp", sStartFolder, "OpenFile", sFileNameAndPath);  
**if** (iret == -1)  
{  
 cout << "Пользователь отказался от выбора файла\n";  
 **return** -1;  
}  
run\_cpp("run\_from\_file", sFileNameAndPath);  
cout << "Файл выбран и выполнен";

Можно составлять сложные фильтры и группы фильтров по расширениям файлов.

Пример.

Предоставить пользователю выбрать файл из трех вариантов выборки.

вариант1 - Excel с расширениями xls,xslx, xslm;

вариант2 - Archicad с расширениями pln, pla, tpl;

вариант3 - Text с расширениями txt, sx.

Стартовый каталог - текущий;

string FileNameAndPath;  
ac\_request("dialog\_get\_filename", "Выберите файл", "Excel,xls,xlsx,xlsm|Archicad,pln,pla,tpl|Text,txt,sx", "", "OpenFile", FileNameAndPath);   
coutvar << FileNameAndPath;

**editdoubledialog - ввод числа**

функция ac\_request() с директивой "editdoubledialog" вызывает диалог для ввода числа с плавающей точкой.

Формат обращения:

int res = ac\_request("editdoubledialog",string smessage, string sstartvalue);

или

int res = ac\_request("editdoubledialog",string smessage, double dstartvalue);

Здесь:

smessage - сообщение в заголовке панели диалога, sstartvalue или dstartvalue - число, которое подставлено по умолчанию в строку редактирования диалога.

Возвращает 0 если пользователь отказался от ввода (нажал кнопку "Отменить", нажал кнопку с крестиком на рамке диалога или клавишу "Esc" на клавиатуре).

Если была нажата клавиша "Ok" то возвращается 1.

Пример.

Получить от пользователя дистанцию между элементами. По умолчанию предложить значение 20.

int res = ac\_request("editdoubledialog","Введите дистанцию между элементами (м)","20");

**editintdialog - ввод целого числа**

функция ac\_request() с директивой "editdoubledialog" вызывает диалог для ввода числа с плавающей точкой.

Формат обращения:

int res = ac\_request("editintdialog",string smessage, string sstartvalue);

или

int res = ac\_request("editintdialog",string smessage, double dstartvalue);

Здесь:

smessage - сообщение в заголовке панели диалога, sstartvalue или dstartvalue - число, которое подставлено по умолчанию в строку редактирования диалога.

Возвращает 0 если пользователь отказался от ввода (нажал кнопку "Отменить", нажал кнопку с крестиком на рамке диалога или клавишу "Esc" на клавиатуре).

Если была нажата клавиша "Ok" то возвращается 1.

Пример.

Получить от пользователя количество элементов. По умолчанию предложить значение 20.

Вариант 1:

int res = ac\_request("editintdialog","Введите количество элементов (шт)","20");

if(res == 1) {

ivalue = ac\_getnumvalue();

}

Вариант 2:

int res = ac\_request("editintdialog","Введите количество элементов (шт)", 20);

if(res == 1) {

ivalue = ac\_getnumvalue();

}

Вариант 3:

int ivalue = 20;

int res = ac\_request("editintdialog","Введите количество элементов (шт)",ivalue);

if(res == 1) {

ivalue = ac\_getnumvalue();

}

**editstringdialog - ввод строки**

функция ac\_request() с директивой "editstringdialog" вызывает диалог для строки.

Формат обращения:

int res = ac\_request("editstringdialog",string smessage, string sdefaultstr);

Здесь:

smessage - сообщение в заголовке панели диалога,

sdefaultstr - строка, которая будет подставлена по умолчанию в строку редактирования диалога.

Возвращает 0 если пользователь отказался от ввода (нажал кнопку "Отменить", нажал кнопку с крестиком на рамке диалога или клавишу "Esc" на клавиатуре).

Если была нажата клавиша "Ok" то возвращается 1.

Пример.

Получить от пользователя ФИО архитектора. По умолчанию предложить значение "Иванов В.В.".

string svalue = "Иванов В.В."  
int res = ac\_request("editstringdialog","Введите ФИО архитектора", svalue);  
**if**(res == 1)  
{  
 svalue = ac\_getstrvalue();  
}  
coutvar << svalue;

**editstringlistdialog - выбор строки из списка**

функция ac\_request() с директивой "editstringlistdialog" вызывает диалог для выбора строки из списка popup.

Формат обращения:

int res = ac\_request("editstringlistdialog",string smessage, int iTable, string sstr);

или

int res = ac\_request("editstringlistdialog",string smessage, int iTable, int index);

или

int res = ac\_request("editstringlistdialog",string smessage, string savailablelist, string sselected, string sdelimiter);

Здесь:

smessage - сообщение в заголовке панели диалога,

iTable - дескриптор таблицы, в которой в первой колонке заданы возможные значения для выбора в диалоге.

str - строка, которая будет выбрана в popup элементе диалога;

index - индекс текста строки из таблицы, которая будет выбрана в качестве предустановки в popup элементе диалога.

Обратите внимание, что в ts\_table индекспервой строки - 0 а в элементе popup - индекс первой строки - 1. Поэтому 0-я строка из таблицы будет 1-й в элементе popup.В index задается и получается индекс выбора по элементу popup.

savailablelist - список элементов для выбора через разделитель;

sselected - предварительно выбранный элемент;

sdelimiter - разделитель.

Возвращает 0 если пользователь отказался от ввода (нажал кнопку "Отменить", нажал кнопку с крестиком на рамке диалога или клавишу "Esc" на клавиатуре).

Если была нажата клавиша "Ok" то возвращается 1.

Результат получается функцией ac\_getstrvalue() - если нужно получить строку или ac\_getnumvalue() если нужно получить индекс.

Пример 1.

Получить от пользователя выбранную строку из таблицы.Предложить значение "Иванов В.В."

int iTable;  
**object**("create","ts\_table",iTable);  
ts\_table(iTable,"add\_column",-1,"string","h1");  
ts\_table(iTable,"add\_row",0,"Иванов В.В.");  
ts\_table(iTable,"add\_row",0,"Петров В.В.");  
ts\_table(iTable,"add\_row",0,"Сидоров В.В.");  
string svalue = "Петров В.В.";  
int index;  
int res = ac\_request("editstringlistdialog","Введите ФИО архитектора", iTable, svalue);  
**if**(res == 1)  
{  
 svalue = ac\_getstrvalue();  
 index = ac\_getnumvalue();  
}  
coutvar << svalue;  
coutvar << index;  
**object**("delete",iTable);

Пример 2

string s = "one;two;three";  
string ss = "two";  
int iires = ac\_request("EditStringListDialog", "Тип линии", s, ss,";");  
coutvar<<iires;  
coutvar<< ss;  
**return**;

**editcheckstringsdialog - выбор нескольких строк из списка**

функция ac\_request() с директивой "EditCheckStringsDialog" вызывает диалог для выбора нескольких строк.

Формат обращения:

int res = ac\_request("EditCheckStringsDialog",string smessage, int iTable);

или

int res = ac\_request("EditCheckStringsDialog", string smessage, string savailablelist, string sselected, string sdelimiter);

Здесь:

smessage - сообщение в заголовке панели диалога,

iTable - дескриптор таблицы из двух колонок - bool с отметкой да/нет и string - текст для выбора.

savailablelist - список возможных вариантов через разделитель;

sselected - список помеченых вариантов через разделитель и возвращаемый набор значений, так же через разделитель;

sdelimiter - разделитель.

Возвращает 0 если пользователь отказался от ввода (нажал кнопку "Отменить", нажал кнопку с крестиком на рамке диалога или клавишу "Esc" на клавиатуре).

Если была нажата клавиша "Ok" то возвращается 1.

Пример.

Получить от пользователя список ФИО архитекторов с отметкой да/нет.

object("create", "ts\_table", iTable);

ts\_table(iTable, "add\_column", -1, "bool", "h1");

ts\_table(iTable, "add\_column", -1, "string", "h2");

ts\_table(iTable, "add\_row", 0, false, 1, "Иванов В.В.");

ts\_table(iTable, "add\_row", 0, true , 1, "Петров В.В.");

ts\_table(iTable, "add\_row", 0, false, 1, "Сидоров В.В.");

int res = ac\_request("EditCheckStringsDialog", "Введите ФИО архитекторов", iTable);

if (res == 1)

{

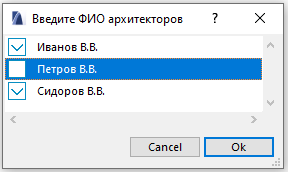
string str;

ts\_table(iTable, "print\_to\_str", str);

coutvar << str;

}

object("delete", iTable);



str=rowcount=3

0). h1=0 h2=Иванов В.В.

1). h1=1 h2=Петров В.В.

2). h1=0 h2=Сидоров В.В.

Пример 2.

string savailable = "1 значение;2 значение;3 значение";  
string sselected = "2 значение;3 значение";  
int res = ac\_request("EditCheckStringsDialog", "Тип линии", savailable, sselected,";");  
**if** (res == 1)  
{  
 coutvar << sselected;  
}

**editbooldialog - выбор Да/Нет**

функция ac\_request() с директивой "editbooldialog" вызывает диалог для выбора значения логической переменной - Истина/Ложь (Да/Нет).

Формат обращения:

int res = ac\_request("editbooldialog", string smessage, bool bdefaultvalue);

Здесь:

smessage - сообщение в заголовке панели диалога,

bdefaultvalue - значение по умолчанию.

Возвращает 0 если пользователь отказался от ввода (нажал кнопку "Отменить", нажал кнопку с крестиком на рамке диалога или клавишу "Esc" на клавиатуре).

Если была нажата клавиша "Ok" то возвращается 1.

Пример.

Получить от пользователя ФИО архитектора. По умолчанию предложить значение "Иванов В.В.".

bool bRes = false;  
int res = ac\_request("editbooldialog","Включать заголовки при экспорте данных?", bRes);  
**if**(res == 1)  
{  
 bRes = ac\_getnumvalue();  
}  
coutvar << bRes;

**editstringlistandcustomdialog - ввод строки или выбор из списка**

функция ac\_request() с директивой "EditStringListAndCustomDialog" вызывает диалог для выбора нескольких строк.

Формат обращения:

int res = ac\_request("EditStringListAndCustomDialog",string smessage, int iTable, string str);

или

int res = ac\_request("EditStringListAndCustomDialog",string smessage, int iTable, int index);

или

int res = ac\_request("EditStringListAndCustomDialog",string smessage, string savailablelist, string sselected, string sdelimiter);

Здесь:

smessage - сообщение в заголовке панели диалога,

iTable - дескриптор таблицы с одной колонкой string - текст для выбора.

Возвращает 0 если пользователь отказался от ввода (нажал кнопку "Отменить", нажал кнопку с крестиком на рамке диалога или клавишу "Esc" на клавиатуре).

Если была нажата клавиша "Ok" то возвращается 1.

str - строка - предустановленное значение.

savailablelist - список элементов для выбора через разделитель;

sselected - предварительно выбранный элемент и возвращаемое значение;

sdelimiter - разделитель.

index - индекс предустановленной строки;

Результат получается функцией ac\_getstrvalue() - если нужно получить строку или ac\_getnumvalue() если нужно получить индекс. Обратите внимание, что если ac\_getnumvalue() вернет 0 - то значит пользователь ввел собственный текст, которого точно нет в списке iTable или savailablelist;

Пример.

Получить от пользователя строку из предложенных вариантов или любую другую строку.

int iTable;

object("create", "ts\_table", iTable);

ts\_table(iTable, "add\_column", -1, "string", "col");

ts\_table(iTable, "add\_row", 0, "Row 1");

ts\_table(iTable, "add\_row", 0, "Row 2");

ts\_table(iTable, "add\_row", 0, "Row 3");

bool bres = ac\_request("EditStringListAndCustomDialog","Enter value", iTable, "Row 0");

if(bres)

{

string sres = ac\_getstrvalue();

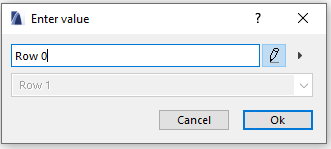
int ires = ac\_getnumvalue();

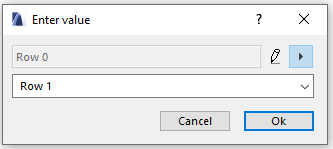
coutvar << sres;

coutvar << ires;

object("delete", iTable);

}





Пример 2

string s = "one;two;three";  
string ss = "two";  
int iires = ac\_request("EditStringListAndCustomDialog", "Тип линии", s, ss,";");  
coutvar<<iires;  
coutvar<< ss;  
**return**;

**Работа с Excel**

**excel\_attach**

Подключить Excel для работы.

В момент запуска программа Excel должна быть открыта.

По умолчанию активной становится текущая страница Excel.

Вызов:

int ires = excel\_attach();

Если возвращается 0, то подключение успешное.

Пример.

Подключить текущую таблицу Excel и вывести в текущее положение маркера число 100.1.

int my\_variable;  
coutvar << my\_variable;  
int ires = excel\_attach();  
**if** (res != 0)  
{  
 cout << "Нет связи с Excel";  
 **return** -1;  
}  
  
double dvalue = 100.1;  
excel\_putnumvalue(dvalue);  
excel\_detach();

**excel\_detach**

Отключить Excel.

Обращение

excel\_detach();

**excel\_putnumvalue**

Записать в текущую позицию фокуса Excel числовое значение.

Вызов:

excel\_putnumvalue(double dvalue);

**excel\_putstrvalue**

Записать в текущую позицию фокуса Excel текстовое значение.

Обращение

excel\_putstrvalue(string svalue);

**excel\_select\_range**

Выделить фокусом Excel указанные ячейки.

Формат обращения:

excel\_select\_range( string srange);

Здесь: srange - текстовый адрес ячейки Excel.

Пример.

Выделить в текущей таблице Excel ячейки в диапазоне "A2:C4".

excel\_select\_range( "A2:C4");

**excel\_visible**

Сделать видимым и вынести окно Excel на передний план.

Обращение

excel\_visible();

**excel\_speedup**

Выполняет действия по ускорению передачи данных в/из Excel. Перед выводом значительных объемов данных в таблицу рекомендуется применять ускорение. По окончании - выключить.

Вызов:

excel\_speedup(int what);

Здесь what - 0/1 - отключить/включить ускоренную обработку данных программой Excel.

**excel\_getnumvalue**

Получить из текущей позиции фокуса Excel числовое значение.

Обращение

double dvalue = excel\_getnumvalue();

**excel\_getstrvalue**

Получить из текущей позиции фокуса Excel текстовое значение.

Обращение

string svalue = excel\_getstrvalue();

**excel\_request**

**set\_column\_width**

Задает ширину столбца или диапазона столбцов.

Вызов:

excel\_request("set\_column\_width",string sdiapazon, double width);

Здесь: sdiapazon - текстовое значение, описывающее диапазон столбцов. Например: "B:B" - означает столбец B. Если написать "B:D" то ширина будет задана для колонок B,C и D.  
width - ширина колонки.

Пример.

Установить ширину колонки B в 18.86:

excel\_request("set\_column\_width","B:B",18.86);

**get\_column\_width**

Получить ширину колонки или диапазона колонок

Вызов:

excel\_request("get\_column\_width", string sdiapazon, double width);

Здесь: sdiapazon - текстовое значение, описывающее диапазон столбцов. Например: "B:B" - означает столбец B. Если написать "B:D" то ширина будет задана для колонок B,C и D.  
в width - будет записана ширина колонки

**set\_row\_height**

Задать высоту указанной строки или диапазона строк

Вызов:

excel\_request("set\_row\_height", string srowdiapazon, double height);

Здесь: srowdiapazon - строковое значение, определяющее диапазон строк, которым нужно установить высоту. Например "1:1" - первая строка, "2:4" - строки со второй по четвертую.

height - высота строки.

Пример.

Задать высоту для строк со 2-й по 8-ю в 15.75.

excel\_request("set\_row\_height","2:8",15.75);

**get\_row\_height**

Получить высоту строки или диапазона строк.

Вызов:

excel\_request("get\_row\_height", string srowdiapazon, double height);

Здесь: srowdiapazon - строковое значение, определяющее диапазон строк, которым нужно установить высоту. Например "1:1" - первая строка, "2:4" - строки со второй по четвертую.

в height - будет записана высота строки.

**set\_borders**

Установить бордюры вокруг текущей выделенной области ячеек.

Вызов:

excel\_request("set\_borders", int border\_left,int border\_top,int border\_right,int border\_bottom);

Здесь: border\_left, border\_top, border\_right, border\_bottom - переменные, которые задают будет или нет бордюр слева,сверху,справа и снизу, соответственно. 1-да, 0-нет.  
Вызов без этих аргументов означает установку всех бордюров:

excel\_request("set\_borders");

**put\_selection\_values**

Присвоить значение ширины, высоты и объединения у текущей выбранной ячейки или группы выбранных ячеек.

Вызов:

excel\_request("put\_selection\_values", string varname1, string/int/double/bool varvalue1,..., string varnamen, string/int/double/bool varvaluen);

Здесь:

varname1 - varnamen - имена переменных, которые доступны для ячейки или группы выделенных ячеек в Excel по аналогии с Range в Excel VBA.

varvalue1-varvaluen - задаваемые значения для этих переменных.

Пример 1:

Задать ширину и высоту текущей выбранной ячейки Excel.

excel\_request("put\_selection\_values","Width",10,"Height",20);

Пример 2:

Объединить выбранные ячейки Excel.

excel\_request("put\_selection\_values","MergeCells",1);

**put\_selection\_fontvalues**

Задать значения для шрифта у текущего выбора Excel (ячейки или группы ячеек).

Вызов:

excel\_request("put\_selection\_fontvalues", string fontparname1, string/double/int/bool fontparvalue1, ... , string fontparnamen, string/double/int/bool/ fontparvaluen);

Здесь:

fontparname1-fontparnamen - имена переменных, которые можно задавать в ячейке Excel (по аналогии с Excel VBA).

fornparvalue1-fontparvaluen - значения для этих переменных.

Пример:

Задать в текущей ячейке Excel размер шрифта 10.32, жирный наклонный.

excel\_request("put\_selection\_fontvalues","Size",10.32,"Bold",1,"Italic",1);

**get\_selection\_area**

Получить номер строки и колонки левого верхнего и правого нижнего углов текущего выбора в таблице Excel.

Вызов:

excel\_request("get\_selection\_area", int start\_row, int start\_column, int end\_row, int end\_column);

Здесь:

start\_row, start\_column - номер строки и колонки левого верхнего угла текущего выбора Excel,

end\_row, end\_column - номер строки и колонки правого нижнего угла текущего выбора Excel.

Пример.

В текущей таблице Excel узнать какие ячейки выбраны.

int start\_row,start\_column,end\_row,end\_column,rows,columns;  
excel\_request("get\_selection\_area",start\_row,start\_column,end\_row,end\_column);

**merged\_cell\_info**

Определить номер колонки и строки верхнего левого и нижнего правого угла объединенной области для текущей ячейки.

Вызов:

excel\_request("merged\_cell\_info", int &merged\_start\_row, int &merged\_start\_col, int &merged\_rows, int &merged\_cols);

Здесь:

merged\_start\_row, merged\_start\_col - номер строки и колонки левого верхнего угла объединенной области в которую входит текущая ячейка,

merged\_rows - число строк, которые объединены в область,

merged\_cols - число колонок, которые объединены в область.

**is\_merge\_cells**

Определить объединены или нет ячейки текущей выбранной области в таблице Excel.

Вызов:

excel\_request("is\_merge\_cells", int & ismerged);

Здесь:

ismerged - если ячейки в текущей выбранной области таблицы Excel объединены, то в ismerged запишется 1.

**set\_backcolor**

**get\_backcolor**

**set\_interior**

**get\_interior**

**selection\_varvalues**

**selection\_font\_varvalues**

**sheet\_select**

**range\_copy**

**booknamedcell**

**save\_active\_sheet\_as\_csv**

Сохранить текущую активную таблицу в файл CSV без вопросов.

Вызов:

int ires = excel\_request("save\_active\_sheet\_as\_csv", string filename);

Здесь:

filename - полный путь и имя файла куда сохранить текущую страницу Excel в виде файла CSV.

Пример.

Выбрать текущим файл "Управляющий файл.xlsm" (должен быть уже открытым в Excel).

Удалить, если уже существуют временные файлы чтобы программа при сохранении не запрашивала подтверждение.

Сохранить копию в файл "h:\1.xlsx".

Открыть его в Excel.

Сохранить его страницы в файлах "1\_Имя таблицы.csv".

Закрыть файл "h:\1.xlsx",

int main()  
{  
 excel\_attach();  
 bool bexist;  
 int ires;  
  
 string sWorkBookUprFile = "Управляющий файл.xlsm";  
 ires = excel\_request("workbook\_select", sWorkBookUprFile);  
 **if** (ires != 0)  
 {  
 tsalert(-1, "Ошибка во время выполнения", "Не получается переключиться в файл excel", sWorkBookUprFile);  
 excel\_detach();  
 **return** -1;  
 }  
  
 string filename = "h:\\1.tmp";  
 ires = shell\_func("file", "is\_exist", filename, bexist);  
 **if** (bexist)  
 {  
 cout << "Удаляем файл " << filename << "\n";  
 ires = shell\_func("file", "delete", filename);  
 }  
  
 excel\_request("save\_copy\_as", filename);  
 excel\_request("workbook\_open", filename);  
 ires = save\_sheet\_as\_csv\_file("Пироги стен");  
 **if** (ires != 0)  
 {  
 **return**;  
 }  
 ires = save\_sheet\_as\_csv\_file("Пироги колонн");  
 ires = save\_sheet\_as\_csv\_file("Пироги балок");  
 ires = save\_sheet\_as\_csv\_file("Пироги потолков");  
 ires = save\_sheet\_as\_csv\_file("Пироги низа и торца лестниц");  
 ires = save\_sheet\_as\_csv\_file("Пироги полов");  
 ires = save\_sheet\_as\_csv\_file("Плинтуса и торцы фундам. оборуд");  
 ires = save\_sheet\_as\_csv\_file("Узлы");  
 ires = save\_sheet\_as\_csv\_file("Слои\_пирогов\_отделки");  
 ires = save\_sheet\_as\_csv\_file("Слои\_пирогов\_полов");  
 excel\_request("workbook\_close");  
  
 excel\_detach();  
}  
  
  
int save\_sheet\_as\_csv\_file(string sSheetName)  
{  
 bool bexist;  
  
 int ires = excel\_request("sheet\_select", sSheetName);  
 **if** (ires != 0) {  
 tsalert(-1, "Ошибка во время выполнения", "Не получается переключиться на страницу", sSheetName);  
 excel\_detach();  
 **return** -1;  
 }  
  
 string filename;  
  
 filename = "h:\\1\_" + sSheetName + ".csv";  
 ires = shell\_func("file", "is\_exist", filename, bexist);  
 **if** (bexist)  
 {  
 cout << "Удаляем файл " << filename << "\n";  
 ires = shell\_func("file", "delete", filename);  
 }  
 ires = excel\_request("save\_active\_sheet\_as\_csv", filename);  
 **return** ires;  
}

**save\_copy\_as**

Сохранить копию текущего файла Excel без его открытия в Excel.

Вызов:

int ires = excel\_request("save\_copy\_as", string filename);

Здесь:

filename - имя и полный путь к файлу, куда надо сохранить копию текущего файла Excel.

**workbook\_open**

Открыть заданный файл в Excel.

Вызов:

int ires = excel\_request("workbook\_open", string filename);

Здесь:

filename - имя и полный путь к файлу, который надо открыть в Excel.

**workbook\_close**

Закрыть текущий файл Excel.

Вызов:

int ires = excel\_request("workbook\_close");

**работа с примечаниями**

getcomment

Получить примечания к текущей ячейке.

Вызов:

excel\_request("comment", "get", string &scomment);

Здесь: scomment - строковое значение комментария к ячейке.

Пример.

string scomment;  
excel\_request("comment","get", scomment);  
cout << scomment;

clearcomments

Очистить примечания к текущей ячейке.

Вызов:

excel\_request("comment", "clear");

setcomment

Задать примечания к текущей ячейке.

Вызов:

excel\_request("comment", "set", string scomment);

Здесь: scomment - строковое значение комментария к ячейке.

Пример.

string scomment = "Это комментарий к ячейке\n в две строки";  
excel\_request("comment","set", scomment);

**Загрузка и выгрузка табличных данных**

**excel\_chart - диаграммы в Excel**

**AddChart**

Создать диаграмму в текущей таблице Excel.

Обращение.

excel\_chart(int iChartDescr,"AddChart");

Здесь:

iChartDescr - дескриптор объекта excel\_chart. Сюда возвращается элемент управления созданной диаграммой.

**SetChartType**

Задать тип диаграммы Excel.

Обращение.

excel\_chart(int iChartDescr,"SetChartType", int itype);

Здесь:

itype - числовой код типа диаграммы Excel.

|  |  |  |
| --- | --- | --- |
| Имя типа диаграммы | Код типа | Описание |
| xl3DArea | -4098 | Трехмерная область. |
| xl3DAreaStacked | 78 | Трехмерная область с накоплением. |
| xl3DAreaStacked100 | 79 | 100% с областями с накоплением. |
| xl3DBarClustered | 60 | Трехмерная кластерная панель. |
| xl3DBarStacked | 61 | Трехмерная линейчатая диаграмма с накоплением. |
| xl3DBarStacked100 | 62 | Трехмерная линейчатая диаграмма с накоплением 100. |
| xl3DColumn | -4100 | Трехмерный столбец. |
| xl3DColumnClustered | 54 | Трехмерный кластеризованный столбец. |
| xl3DColumnStacked | 55 | Объемный столбец с накоплением. |
| xl3DColumnStacked100 | 56 | Трехмерная гистограмма с накоплением 100%. |
| xl3DLine | -4101 | Трехмерная линия. |
| xl3DPie | -4102 | Трехмерная круговая диаграмма. |
| xl3DPieExploded | 70 | Развернутая трехмерная круговая диаграмма. |
| xlArea | 1 | Область |
| xlAreaStacked | 76 | Область с накоплением. |
| xlAreaStacked100 | 77 | 100% с областями с накоплением. |
| xlBarClustered | 57 | Панель кластера. |
| xlBarOfPie | 71 | Вторичная гистограмма. |
| xlBarStacked | 58 | Линейчатая диаграмма с накоплением. |
| xlBarStacked100 | 59 | 100% линейчатой диаграммы с накоплением. |
| xlBubble | 15 | Плыв. |
| xlBubble3DEffect | 87 | Пузырьковая с трехмерными эффектами. |
| xlColumnClustered | 51 | Кластеризованный столбец. |
| xlColumnStacked | 52 | Гистограмма с накоплением. |
| xlColumnStacked100 | 53 | 100% гистограммы с накоплением. |
| xlConeBarClustered | 102 | Кластеризованная коническая с полосой. |
| xlConeBarStacked | 103 | Коническая с накоплением. |
| xlConeBarStacked100 | 104 | 100% нормированной конусной полосы. |
| xlConeCol | 105 | Столбец объемных конусов. |
| xlConeColClustered | 99 | Кластеризованный конусный столбец. |
| xlConeColStacked | 100 | Конусный столбец с накоплением. |
| xlConeColStacked100 | 101 | 100% конусный столбец с накоплением. |
| xlCylinderBarClustered | 95 | Кластеризованная цилиндрическая панель. |
| xlCylinderBarStacked | 96 | Вертикальная полоса цилиндра. |
| xlCylinderBarStacked100 | 97 | 100% нормированной цилиндрической полосы. |
| xlCylinderCol | 98 | Столбец 3D-цилиндра. |
| xlCylinderColClustered | 92 | Кластеризованный конусный столбец. |
| xlCylinderColStacked | 93 | Конусный столбец с накоплением. |
| xlCylinderColStacked100 | 94 | 100% нормированный столбец цилиндра. |
| xlDoughnut | -4120 | Сектор. |
| xlDoughnutExploded | 80 | Разрезанная кольцевая диаграмма. |
| xlLine | 4 | Междустрочный. |
| xlLineMarkers | 65 | Строка с маркерами. |
| xlLineMarkersStacked | 66 | Линия с накоплением с маркерами. |
| xlLineMarkersStacked100 | 67 | 100% графика с накоплением с маркерами. |
| xlLineStacked | 63 | График с накоплением. |
| xlLineStacked100 | 64 | 100% графика с накоплением. |
| xlPie | 5 | Кругов. |
| xlPieExploded | 69 | Разрезанная круговая диаграмма. |
| xlPieOfPie | 68 | Вторичная круговая диаграмма. |
| xlPyramidBarClustered | 109 | Кластеризованная пирамида. |
| xlPyramidBarStacked | 110 | Пирамидальная диаграмма с накоплением. |
| xlPyramidBarStacked100 | 111 | 100% пирамидальной диаграммы с накоплением. |
| xlPyramidCol | 112 | Столбец 3D пирамиды. |
| xlPyramidColClustered | 106 | Кластеризованный столбец пирамиды. |
| xlPyramidColStacked | 107 | Столбец пирамиды с накоплением. |
| xlPyramidColStacked100 | 108 | 100% столбца пирамиды с накоплением. |
| xlRadar | -4151 | Лепестковые. |
| xlRadarFilled | 82 | Заполненная лепестковая диаграмма. |
| xlRadarMarkers | 81 | Лепестковая диаграмма с маркерами данных. |
| xlRegionMap | 140 | Диаграмма карты. |
| xlStockHLC | 88 | Высокая — минимум — закрытие. |
| xlStockOHLC | 89 | Открытие — высокая — минимум — закрытие. |
| xlStockVHLC | 90 | Громкость — высокая — минимальная — закрытие. |
| xlStockVOHLC | 91 | Volume — Open — High – минимум — Close. |
| xlSurface | 83 | Трехмерная поверхность. |
| xlSurfaceTopView | 85 | Поверхность (вид сверху). |
| xlSurfaceTopViewWireframe | 86 | Поверхность (каркас верхнего уровня представления). |
| xlSurfaceWireframe | 84 | Трехмерная поверхность (каркасная схема). |
| xlXYScatter | -4169 | Рабочие. |
| xlXYScatterLines | 74 | Точечная диаграмма со строками. |
| xlXYScatterLinesNoMarkers | 75 | Точечная диаграмма со строками без маркеров данных. |
| xlXYScatterSmooth | 72 | Точечная диаграмма с сглаженными линиями. |
| xlXYScatterSmoothNoMarkers | 73 | Точечная диаграмма со сглаженными линиями без маркеров данных. |

**SetChartTitle**

Задать заголовок диаграмме.

Обращение.

excel\_chart(int iChartDescr,"SetChartTitle", string sTitle);

Здесь:

iChartDescr - дескриптор объекта excel\_chart,

sTitle - новый заголовок

**SetSourceDataRange**

задать текущую выделенную область в таблице Excel в качестве источник данных диаграммы

Вызов:

excel\_chart(int iChartDescr, "SetSourceDataRange");

Здесь:

iChartDescr - дескриптор диаграммы Excel

**Пример**

//------------------------------------------------------

// LABPP

// Создать диаграмму в Excel по данным в ячейках C7:F8

//------------------------------------------------------

int main()

{

excel\_attach(); // подключить Excel

int iChartDescr;

object("create","excel\_chart", iChartDescr); // создать объект

excel\_chart(iChartDescr,"AddChart"); // создать диаграмму на текущей странице таблицы Excel

excel\_chart(iChartDescr,"SetChartType",5); // задать тип диаграммы

excel\_chart(iChartDescr,"SetChartTitle","Квартирография"); // задать заголовок

excel\_select\_range("C7:F8"); // выбрать ячейки в таблице Excel

excel\_chart(iChartDescr,"SetSourceDataRange"); // задать эти ячейки как источник данных диаграммы

object("delete",iChartDescr); // удалить объект

excel\_detach(); // отключить Excel

}

**Работа с Word**

**word\_attach**

Подключение программы Word для обмена данными и управления.

Вызов:

int ires = word\_attach();

Возвращает 0, если подключение произошло успешно.

**word\_detach**

Отключение программы Word.

Вызов:

word\_detach();

**word\_visible**

Делает видимым и выносит на передний план окно программы Word. При спрятанном окне Word обрабатывает обращения значительно быстрее, т.к. не занимается переформатированием отображения. Так что разумно для ускорения вывода перед записью спрятать окно, а после - показать.

Вызов:

word\_visible(int ivisible);

Здесь: ivisible - 0/1 - спрятать/показать окно Word.

**word\_request**

В документах Word имеются возможность использования специальных переменных. По типу автотекста в ARCHICAD, переменные в Word могут быть вставлены в разные места форматированного текста. Так что нам достаточно присвоить им значение и документ оставаясь правильно отформатированным отразит новые данные.

**NСчитывание и запись полей переменных - docfield**

get

Считать значение переменной документа Word

int ires = word\_request("docfield","get", string wordfielsname, string/double/int/bool value);

Здесь:

wordfielsname - имя переменной Word,

string/double/int/bool value - значение любого из перечисленных типов для присвоения переменной Word.

Возвращает 0 при успешном выполнении функции.

set

Задать значение переменной документа Word.

Вызов:

int ires = word\_request("docfield","set",string wordfielsname, string/double/int/bool value);

Здесь:

wordfielsname - имя переменной Word,

string/double/int/bool value - значение любого из перечисленных типов для присвоения переменной Word.

Возвращает 0 при успешном выполнении функции.

update\_all\_docfields

После изменения значений переменных документа Word, чтобы измененные данные отразились в тексте необходимо обновить поля в тексте документа.

Вызов:

word\_request("update\_all\_docfields");

**Работа через ts\_idispatch**

Пример:

Подключиться к открытому документу Word и написать слово, изменить шрифт и написать фразу.

//-----------------------------------------------

// LABPP Word пишем текст в Word и меняем шрифт

// LABPP 2023

//-----------------------------------------------

int main()

{

int iIDispatch\_Word;

object("create", "ts\_idispatch", iIDispatch\_Word);

int ires = ts\_idispatch(iIDispatch\_Word, "attach", "Word.Application"); // create dispatcher object

if (ires != 0) {

cout << "\nError whan attach to word";

return -1;

}

int iVariant\_Selection;

object("create", "ts\_variant", iVariant\_Selection);

ires = ts\_idispatch(iIDispatch\_Word, "AutoWrap", iVariant\_Selection, "DISPATCH\_PROPERTYGET", "Selection", 0);

int iIDispatch\_Selection;

object("create", "ts\_idispatch", iIDispatch\_Selection);

ts\_variant(iVariant\_Selection, "get\_pdispVal", iIDispatch\_Selection);

// Для информации об объекте-диспетчере, какие у него есть переменные и функции (не обязательно) ---->

// получить список методов через перенос строки (\n)

string smethodslist;

ires = ts\_idispatch(iIDispatch\_Selection, "GetIDispatchMethods", smethodslist);

coutvar << smethodslist;

//<----

int iVariant\_Null;

object("create", "ts\_variant", iVariant\_Null);

int iVariant\_Text;

object("create", "ts\_variant", iVariant\_Text);

ts\_variant(iVariant\_Text, "set\_value\_simple", "Привет");

ires = ts\_idispatch(iIDispatch\_Selection, "AutoWrap", iVariant\_Null, "DISPATCH\_METHOD", "TypeText", 1, iVariant\_Text);

int iVariant\_Font;

object("create", "ts\_variant", iVariant\_Font);

ires = ts\_idispatch(iIDispatch\_Selection, "AutoWrap", iVariant\_Font, "DISPATCH\_PROPERTYGET", "Font", 0);

int iIDispatch\_Font;

object("create", "ts\_idispatch", iIDispatch\_Font);

ts\_variant(iVariant\_Font, "get\_pdispVal", iIDispatch\_Font);

ts\_variant(iVariant\_Text, "set\_value\_simple", "GOST 2.304 type A");

ires = ts\_idispatch(iIDispatch\_Font, "AutoWrap", iVariant\_Null, "DISPATCH\_PROPERTYPUT", "Name", 1, iVariant\_Text);

ts\_variant(iVariant\_Text, "set\_value\_simple", "Привет другим шрифтом");

ires = ts\_idispatch(iIDispatch\_Selection, "AutoWrap", iVariant\_Null, "DISPATCH\_METHOD", "TypeText", 1, iVariant\_Text);

ts\_idispatch(iIDispatch\_Word, "detach");

object("delete", iIDispatch\_Word);

object("delete", iIDispatch\_Selection);

object("delete", iVariant\_Selection);

object("delete", iVariant\_Text);

object("delete", iVariant\_Null);

object("delete", iVariant\_Font);

object("delete", iIDispatch\_Font);

}

**Работа с AutoCAD**

**autocad\_request**

**attach**

Подключить программу AutoCAD.

Вызов:

int ires = autocad\_request("attach");

Возвращает 0 при успешном подключении.

Пример:

int ires;  
ires = autocad\_request("attach");  
**if**(ires != 0)  
{  
 cout << "Can't attach AutoCAD\n";  
 **return** -1;  
}

**detach**

Отключить программу AutoCAD.

Вызов:

autocad\_request("detach");

**specify\_color\_RGB**

Задать цвет для вывода в чертеж.

autocad\_request("specify\_color\_RGB", int red, int green, int blue);

Здесь red, green и blue - коэффициенты интенсивности составляющих RGB от 0 до 255.

Пример.

Задать белый цвет для вывода.

autocad\_request("specify\_color\_RGB", 255, 255, 255);

**draw\_text**

Написать текст на чертеже.

autocad\_request("draw\_text", string stext, int posx, int posy, int textsize);

Здесь stext - текст для вывода,

posx и posy - позиция текста,

textsize - размер текста.

Пример.

Вывести текст "123".

string stext = "123";  
autocad\_request("draw\_text", stext, 1.2, -2.3, 0.2);

**draw\_polyline**

Нарисовать полилинию на чертеже.

autocad\_request("draw\_polyline", int iCoordTableDesct);

Здесь iCoordTableDescr - номер идентификатора таблицы с координатами (дескриптор).

Формат таблицы координат:

double x, double y, double b

Здесь x,y - координаты точки, b - выпуклость (для линий - 0).

Пример.

Нарисовать полилинией квадрат со стороной 1 м - (0;0),(1;0),(1;1),(0,1),(0;0).

int iTable;  
**object**("create","ts\_table",iTable);  
ts\_table(iTable,"add\_column",0,"double","x",0);  
ts\_table(iTable,"add\_column",1,"double","y",0);  
ts\_table(iTable,"add\_column",2,"double","b",0);  
  
ts\_table(iTable,"add\_row","x",0,"y",0,"b",0);  
ts\_table(iTable,"add\_row","x",1,"y",0,"b",0);  
ts\_table(iTable,"add\_row","x",1",y",1,"b",0);  
ts\_table(iTable,"add\_row","x",0,"y",1,"b",0);  
ts\_table(iTable,"add\_row","x",0,"y",0,"b",0);  
  
autocad\_request("draw\_polyline", iTable);  
**object**("delete",iTable);

**draw\_line**

Нарисовать линию в AutoCAD.

Вызов:

ires = autocad\_drawline( double x1, double y1, double x2, double y2);

Здесь:

x1, y1, x2, y2 - позиции двух точек линии.

**draw\_arc**

Нарисовать дугу в AutoCAD.

Вызов:

int ires = autocad\_drawarc( double centerX, double centerY, double radius, double startAngle, double endAngle);

Здесь:

centerX, centerY - позиция центра дуги,

radius - радиус;

startAngle - начальный угол:

endAngle - конечный угол.

**Пример - нарисовать контуры зон из ARCHICAD в AutoCAD полилиниями**

//---------------------------------------------

// LABPP

// Нарисовать зоны полилиниями в AutoCAD

// Draws Zone elements in AutoCAD by PolyLines

//---------------------------------------------

int main()

{

int ires;

ires = autocad\_request("attach");

if(ires != 0)

{

cout << "Can't attach AutoCAD\n";

return -1;

}

int iTable;

object("create","ts\_table",iTable);

ts\_table(iTable,"add\_column",0,"double","x",0);

ts\_table(iTable,"add\_column",1,"double","y",0);

ts\_table(iTable,"add\_column",2,"double","b",0);

ac\_request("load\_elements\_list\_from\_selection",0,"ZoneType");

ires = ac\_request("get\_loaded\_elements\_list\_count",0);

int icount = ac\_getnumvalue();

if(icount == 0)

{

cout << "You have not selected a Zone element.";

return -1;

}

int rowcount;

int iCoordTable;

object("create","ts\_table",iCoordTable);

int i,j;

double xvalue,yvalue,si;

double pi = 3.1415926535;

int itablerowcount;

runtimecontrol("workline", "setmax", icount);

for(i=0;i<icount; i++)

{

runtimecontrol("workline", "setpos", i+1);

ac\_request("set\_current\_element\_from\_list",0,i);

ac\_request("get\_gdlelem\_property\_value","ZoneName");

string szonename = ac\_getstrvalue();

cout << "zonename=" << szonename << "\n";

ac\_request("get\_gdlelem\_property\_value","ROOM\_AREA");

double ra = tsround(ac\_getnumvalue(),2);

cout << "room\_area=" << ra << "\n";

ac\_request("get\_gdlelem\_property\_value","pos.x");

double posx = ac\_getnumvalue();

ac\_request("get\_gdlelem\_property\_value","pos.y");

double posy = ac\_getnumvalue();

string stext = szonename;

autocad\_request("specify\_color\_RGB",0,254,0);

autocad\_request("draw\_text",stext,posx-0.5,posy,0.2);

stext = "(" + sprintf("%g",ra) + ")";

autocad\_request("draw\_text",stext,posx-0.5,posy-0.3,0.2);

autocad\_request("specify\_color\_RGB",254,254,0);

ts\_table(iTable, "clear\_rows");

ac\_request("get\_element\_value","CoordTable",iCoordTable);

ts\_table(iCoordTable, "get\_rows\_count", rowcount);

double lastxvalue, lastyvalue, lastsi;

double radius, firstangle;

double startangle, endangle;

double centerx,centery;

double xfornext, yfornext;

double tg\_of\_quatro\_angle;

for(j=0;j<rowcount;j++)

{

ts\_table(iCoordTable,"select\_row", j);

ts\_table(iCoordTable,"get\_value\_of","x",xvalue);

ts\_table(iCoordTable,"get\_value\_of","y",yvalue);

ts\_table(iCoordTable,"get\_value\_of","si",si);

//cout << "x="<<xvalue<<", y="<< yvalue << ", si="<< si << "\n";

if(j!=0)

{

if((si!=900)&&(si!=4000))

{

if(si==-1)

{

ts\_table(iTable,"add\_row",0,xvalue,1,yvalue,2,0);

autocad\_request("draw\_polyline",iTable);

ts\_table(iTable, "clear\_rows");

}

else {

ts\_table(iTable,"add\_row",0,xvalue,1,yvalue,2,0);}

}

else if(si==900)

{

ac\_request("geometry\_calc\_2d","get\_length\_2point",lastxvalue, lastyvalue, xvalue, yvalue, radius);

ac\_request("geometry\_calc\_2d","get\_line\_angle",lastxvalue, lastyvalue, xvalue, yvalue, firstangle);

centerx = xvalue;

centery = yvalue;

//cout << "radius=" << radius << " , firstangle="<< firstangle << "\n";

}

else if(si==4000)

{

//cout << "yvalue=" << yvalue << "\n";

if(yvalue > 0)

{

startangle = PI - (xvalue\*pi/180.0 - firstangle);

endangle = PI - (-yvalue\*pi/180.0 - firstangle);

}

else

{

endangle = PI - (xvalue\*pi/180.0 - firstangle);

startangle = PI - (-yvalue\*pi/180.0 - firstangle);

}

ac\_request("geometry\_calc\_2d","rotate\_point\_and\_move",centerx, centery, lastxvalue,lastyvalue, endangle, centerx, centery, 1, xfornext, yfornext);

tg\_of\_quatro\_angle = tg((yvalue\*pi/180.0)/4);

ts\_table(iTable,"get\_rows\_count", itablerowcount);

ts\_table(iTable,"select\_row", itablerowcount-1);

ts\_table(iTable,"set\_value\_of",2,tg\_of\_quatro\_angle);

//cout << "tg=" << tg\_of\_quatro\_angle << "\n";

xvalue = xfornext;

yvalue = yfornext;

}

}

else {

ts\_table(iTable,"add\_row",0,xvalue,1,yvalue,2,0); }

lastxvalue = xvalue;

lastyvalue = yvalue;

lastsi = si;

}

ts\_table(iTable,"get\_rows\_count", itablerowcount);

if(itablerowcount>0) {

autocad\_request("draw\_polyline",iTable);}

}

object("delete",iCoordTable);

object("delete",iTable);

autocad\_request("detach");

return 0;

}

**Функция object**

Обеспечивает возможность базовых операций над объектами - создание, удаление и т.п.

**create**

Формат:

int object("create", string objclass, int descr);

Создает объект типа objclass и возвращает его дескриптор в переменную descriptor.

Возвращает 0 при успешном создании объекта.

**delete**

int object("delete", int descr);

Удаляет из памяти объект с дескриптором descr.

**Пример.**

Cоздать объект ts\_table (динамическая таблица) и удалить его.

int TableDescr1;

object("create","ts\_table",TableDescr1);

object("delete",TableDescr1);

**Объект ts\_table**

**Функция ts\_table**

Функция ts\_table позволяет выполнять операции над объектом динамических таблиц.

**init (clear)**

Полностью очистить информацию в таблице включая данные о столбцах.

Вызов:

ts\_table(int iTable,"init");

или

ts\_table(int iTable,"clear");

Здесь:

iTable - дескриптор объекта типа ts\_table, у которого необходимо очистить всю информацию полностью.

**Руководство по LABPP\_Automat для ARCHICAD**

Очистить все строки таблицы.

При этом информация о колонках остается.

Вызов:

ts\_table(int iTable, "clear\_rows");

Здесь:

iTable - дескриптор стираемой таблицы.

**add\_column**

Добавить колонку в таблицу. Колонки могут быть строковыми или числовыми.  
Вызов:

ts\_table(int descriptor, "add\_column", int columnnumber, string columntype, string columnname);

или

ts\_table(int descriptor, "add\_column", int columnnumber, string columntype, string columnname, bool/int/double/string defalutvalue);

Здесь:  
descriptor - дескриптор экземпляра объекта таблицы, у которой добавляется колонка.  
columnnumber - номер колонки (можно поставить -1 чтобы программа сама создала номер).

columntype - тип значений в колонке ("string", "int", "double", "bool").  
columnname - название (заголовок) колонки.

Пример 1.

Добавить в таблицу TableDescr1 текстовую колонку №0 с заголовком "код материала + ед.изм.":

ts\_table(TableDescr1, "add\_column",0,"string","код материала + ед.изм.");

Пример 2.

Добавить числовую колонку в конец таблицы (номер колонки определить автоматически) со значением поумолчанию -10.1 .

ts\_table(TableDescr1, "add\_column", -1,"double",szonename, -10.1);

**is\_column\_exist**

Определить наличие колонки с указанным именем в таблице и если такая колонка существует вернуть ее номер. Если нет - то вернуть -1.  
Вызов:

int iret = ts\_table(int descriptor, "is\_column\_exist", string columnname);

Здесь:  
descriptor - дескриптор экземпляра объекта таблицы, у которой запрашивается наличие колонки.  
columnname - имя колонки колонки

Пример 1.

Добавить в таблицу iTableDescr1 логическую колонку "Есть в проекте" если она не существует.

int ires = ts\_table(iTableDescr1, "is\_column\_exist", "Есть в проекте");  
**if**(ires < 0) {  
 ts\_table(iTableDescr1, "add\_column",-1,"bool","Есть в проекте");  
}

**set\_first\_key**

Когда нужно избежать дублирования записей в таблице можно использовать удобный механизм.    
Одну или несколько колонок можно задать в качестве так называемого "первичного ключа".

Если заданы сразу несколько колонок, то если в строках эти две колонки имеют одинаковое значение, то запись не добавляется.

Например если задан ключ по колонке 0 и 1, то новая строка "Значение колонки 0" , "Значение колонки 1", "Что-то неважно 1" не будет добавлена если в таблице уже есть "Значение колонки 0" , "Значение колонки 1", "Что-то другое неважно 2".  
После этого все добавляемые строки будут анализироваться и если в таблице уже есть строка с таким же значением в колонке(ах), то новая строка создаваться не будет.  
А если добавление строки производится не простой директивой "add\_row" а "add\_row\_sum" то в обнаруженной строке будут суммированы значения числовых колонок (см. add\_row\_sum).

Пример, чтобы сделать колонку №1 ключевой:

ts\_table(TableDescr1,"set\_first\_key",1);

Пример, чтобы сделать колонку с названием "Наименование объекта" ключевой:

ts\_table(TableDescr1,"set\_first\_key","Наименование объекта");

**add\_row**

Добавление строки в таблицу.

ts\_table(int TableDescr,"add\_row", int column1index/string column1name, bool/int/double/string column1value, ... , int columnNindex/string columnNname, bool/int/double/string columnNvalue);

Здесь:

column1index или column1name - индекс (от 0) или название колонки, в которую нужно записать значение column1value;

Если колонка №0 задана как "первичный ключ" (указана в директиве set\_first\_key), то можно записывать значения колонок за несколько команд:

ts\_table(TableDescr1,"add\_row", 0, objectname, 1, value\_to\_column\_1);  
ts\_table(TableDescr1,"add\_row", 0, objectname, 2, value\_to\_column\_2);

Данные допишутся в ту же строку со значением в ключевой колонке №0.

**add\_row\_sum**

Добавление строки с суммированием в числовых колонках со строкой, у которой совпадает значение ключевой колонки.  
Проще всего объяснить на актуальном примере.  
Имеются объекты "Доска". В этих объектах есть поле "типоразмер" ("100х40","50х40" и т.п.), "единица измерения" ("пог.м","кв.м","куб.м") и "количество".  
Мы хотим получить суммарную таблицу, где бы все объекты типа "Доска", в проекте просуммировались следующим образом:

|  |  |  |
| --- | --- | --- |
| Материал | Ед.изм | Кол-во |
| Доска 100х50 | пог.м | 1000 |
| Доска 50х40 | куб.м. | 2000 |
| Доска 50х20 | кв.м. | 1900 |

Для этого создаем таблицу с одной ключевой колонкой и колонками для обычных данных

ts\_table(TableDescr1,"add\_column",0,"string","имя объекта + типоразмер + ед.изм");  
ts\_table("TableDescr, "set\_first\_key",0);  
ts\_table(TableDescr1,"add\_column", 1,"string","objectname");  
ts\_table(TableDescr1,"add\_column", 2,"string","tiporazmer");  
ts\_table(TableDescr1,"add\_column", 3,"string","edizm");  
ts\_table(TableDescr1,"add\_column", 4,"double","kolvo");  
ts\_table(TableDescr1,"set\_columns\_to\_add\_sum", "kolvo");

Для добавления строки в таблицу просто используем следующую запись:

ts\_table(TableDescr1,"add\_row\_sum",0,objectname+tiporazmer+edizm, 1,objectname, 2, tiporazmer, 3, edizm, 4, kolvo);

В результате в таблице все объекты с одинаковыми значением имя+типоразмер+ед.изм. сведутся в единые записи, а в колонке "количество" будет стоять сумма количеств.

**sort**

Сортировка строк таблицы по заданной колонке или по ключевой колонке (если без аргументов). Можно указывать до 7-ми колонок.

Колонки могут задаваться как числовым индексом, так и названием. Можно одни числом, другие названием одновременно.

Вызов:

ts\_table(int iTableDescr, "sort", int icolnum1, int icolnum2, ... ,int icolnum7);

или

ts\_table(int iTableDescr, "sort", string scolname1, string scolname2, ... ,string scolname7);

Здесь:

icolumn1... icolumn7 - числовые индексы колонок, по которым производить сортировку.

scolname1... icolname7 - текстовые названия колонок.

Пример 1. Сортировать таблицу по колонке №0

ts\_table(TableDescr1,"sort",0);

Пример 2. Сортитовать таблицу по ключевым колонкам, указанным ранее директивой "set\_first\_key".

ts\_table(TableDescr1,"sort");

Пример 3. Сортитовать таблицу по колонкам сначала по колонке с индексом 0, затем по колонке с именем "Класс" и потом "Подкласс".

ts\_table(TableDescr1,"sort", 0, "Класс","Подкласс");

**sort\_back**

Обратная сортировка строк таблицы по заданной колонке или по ключевой колонке (если без аргументов). Можно указывать до 7-ми колонок.

Колонки могут задаваться как числовым индексом, так и названием. Можно одни числом, другие названием одновременно.

Вызов:

ts\_table(int iTableDescr, "sort", int icolnum1, int icolnum2, ... ,int icolnum7);

или

ts\_table(int iTableDescr, "sort", string scolname1, string scolname2, ... ,string scolname7);

Здесь:

icolumn1... icolumn7 - числовые индексы колонок, по которым производить сортировку.

scolname1... icolname7 - текстовые названия колонок.

Пример 1. Сортировать таблицу по колонке №0

ts\_table(TableDescr1,"sort",0);

Пример 2. Сортитовать таблицу по ключевым колонкам, указанным ранее директивой "set\_first\_key".

ts\_table(TableDescr1,"sort");

Пример 3. Сортитовать таблицу по колонкам сначала по колонке с индексом 0, затем по колонке с именем "Класс" и потом "Подкласс".

ts\_table(TableDescr1,"sort", 0, "Класс","Подкласс");

**set\_columns\_to\_sort\_as\_LikeABC10\_a**

Задать колонки, в которых сортировку проводить по принципу номеров помещений.

Вызов:

ts\_table(iTable, "set\_columns\_to\_sort\_as\_LikeABC10\_a");

или

ts\_table(iTable,"set\_columns\_to\_sort\_as\_LikeABC10\_a",

int colnum1/string colname1,

int colnum2/string colname2,

...

int colnumN/string colnameN

);

Здесь:

colnumN - номер колонки, в которой будет сортироваться значение как ABC10\_a;

colname1 - то же можно задать названием колонки.

Предыдущее назначение колонок с сортировкой ABC10\_a отменяется.

По умолчанию все колонки с нормальной сортировкой.

Если задать команду без списка колонок, то все колонки будут нормальные.

Пример:

Отсортировать таблицу по колонке с номерами помещений ("Помещение 1001.b1", "ПОМЕЩЕНИЕ 1001.a" и т.п.).

ts\_table(iTable, "set\_columns\_to\_sort\_as\_LikeABC10\_a", "Номер помещения");

задать суммирование по колонкам:

ts\_table(iTable,"sort","Номер помещения");

**search**

Быстрый поиск в таблице первого совпадающего значения колонки.  
Возвращает номер строки или -1, если такая строка отсутствует в таблице.

Пример. Найти строку со значением в колонке №1 "Доска"

int irow = ts\_table(TableDescr1,"search",1,"Доска");

**search\_regex**

Быстрый поиск строки в таблице по содержимому ячеек, причем для текстовыйх колонок применяется принцип ставнения по шаблону по системе REGEX кодов.  
Возвращает номер строки или -1, если такая строка отсутствует в таблице.

Пример.

Найти строку со значением в колонке №1 "Доска"

int irow = ts\_table(TableDescr1,"search\_regex",1,"Доска", 2, "\*");  
(см. regex шаблоны)

**search\_extended**

Найти в таблице строку, используя расширенный фильтр.

На примере показано как найти в таблице iTable строку, у которой в колонке с названием "Колонка1" значение "СТЕНА", значение в колонке "Колонка2" - содержит "3", в колонке "Колонка3" содержится в любом месте "В22" или "В22" (В - или на русском или на английском) и в колонке "Колонка 4" число больше или равное 2.

bool stoponfirstnoteq = **false**;  
bool add\_sum = **false**;  
  
int ires = ts\_table(iTable, "search\_extended",   
 "(", "ANY", "Колонка1", "=", "СТЕНА", "", "AND",   
 "" , "ANY", "Колонка2", "CONTAINS", "3", "", "AND",   
 "" , "ANY", "Колонка3", "REGEXMATCH", ".\*[BВ]22.\*", "", "AND",   
 "", "ANY", "Колонка4", ">=", 2, ")" );

Здесь

iTable - дескриптор таблицы, в которой надо найти строку;

Возвращает -1 если строка не найдена, или индекс найденой строки (>=0).

**search\_extended\_table**

Найти в таблице строку, используя расширенный фильтр.

Фильтр берется из таблицы фильтра.

На примере показано как найти в таблице iTable строку, у которой в колонке с названием "Колонка1" значение "СТЕНА", значение в колонке "Колонка2" - содержит "3", в колонке "Колонка3" содержится в любом месте "В22" или "В22" (В - или на русском или на английском) и в колонке "Колонка 4" число больше или равное 2.

bool stoponfirstnoteq = false;

bool add\_sum = false;

int iTableFilter;

object("create", "ts\_table", iTableFilter);

ts\_table(iTableFilter, "create\_structure\_for\_filter\_table");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "(", "ANY", "Колонка1", "=", "СТЕНА", "", "AND");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "ANY", "Колонка2", "CONTAINS", "3", "", "AND");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "ANY", "Колонка3", "REGEXMACH", ".\*[BВ]22.\*", "", "AND");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "ANY", "Колонка4", ">=", 2, ")", "");

int ires = ts\_table(iTable, "search\_extended", iTableFilter);

Здесь

iTable - дескриптор таблицы, в которой надо найти строку;

iTableFilter - дескриптор таблицы, в которой описаны требования фильтра;

Возвращает -1 если строка не найдена, или индекс найденой строки (>=0).

**search\_bounds**

Для таблиц, где в строках записаны границы прямоугольников определить строку, куда попадает указанная точка.

Вызов:

int ires = ts\_table(1,"search\_bounds", int colnumSX, int columnSY, int columnEX, int columnEY, double posX, double posY, int startrow, int endrow);

ires - результат поиска -1 если не нашлось, или номер строки.

columnSX, columnSY, columnEX, columnEY - числовые индексы или текстовые названия колонок, в которых находятся соответствующие значения, описывающие границы прямоугольников;

posX, posY - координаты точки, которую проверяем в какие границы она попала (ищем строку с прямоугольником, у которого эта точка внутри);

startrow - индекс начальной строки поиска;

endrow - индекс конечной строки поиска. Если -1 - значит искать до конца.

Пример можно посмотреть в файле LABPP\_Table\_unicode.cpp - быстрое определение ячейки в таблице по координатам щелчка мыши.

**select\_row**

Сделать текущей указанную строку в таблице.

Пример.

Сделать текущей первую строку в таблице (индекс строки - от 0 до n-1):

int i=0;  
ts\_table(TableDescr1,"select\_row",i);

**seek\_from\_top, seek\_from\_bottom, seek\_from\_current**

Задание нового положения текущей строки таблицы в виде смещения относительно начала, конца или текущей строки таблицы.

Вызов:

int ires = ts\_table(int iTable, "seek\_from\_top", int ioffset); // смещение от начала  
int ires = ts\_table(int iTable, "seek\_from\_bottom", int ioffset); // смещение от конца таблицы  
int ires = ts\_table(int iTable, "seek\_from\_current", int ioffset); //смещение от текущей позиции

Здесь:

iTable - дескриптор таблицы;

ioffset - число, задающее смещение.

ires - результат операции - отрицательный или 0 при успешном выполнении.

**add\_row\_from\_eq\_table**

Добавить строку в таблицу из аналогичной таблицы.

Вызов:

ts\_table(int iTableDescr, "add\_row\_from\_eq\_table", **in** tiTableFrom, int indexrowfrom, bool bsum);

Здесь:

iTableDescr - дескриптор таблицы, к которой добавить строки;

iTableFrom - дескриптор таблицы, из которой взять эти строки;

indexrowfrom - индекс начальной строки в копируемой таблице с которой начать добавление;

bsum - если true, то добавлять строки как add\_row\_sum (т.е. если есть строка совпадающая по ключевой колонке, то просто суммировать значения в числовых колонках);

**add\_rows\_from\_eq\_table\_by\_filter**

Добавить в таблицу из аналогичной таблицы строки, у которых значения в колонках соответствуют заданным.

Вызов:

ts\_table(int iTableDescr, "add\_rows\_from\_eq\_table\_by\_filter", int iTableFrom, bool bstoponfirstnoteq, bool bsum, int icolumnindex1, string value1, ... string scolumnnameN, double valueN);

Здесь:

iTableDescr - дескриптор таблицы, к которой добавить строки;

iTableFrom - дескриптор таблицы, из которой взять эти строки;

bstoponfirstnoteq - остановить выборку как только после найденых строк появилась первая несовпадающая (т.е. считать что все нужные строки локализованы в одном месте таблицы);

bsum - если true, то добавлять строки как add\_row\_sum (т.е. если есть строка совпадающая по ключевой колонке, то просто суммировать значения в числовых колонках);

icolumnindex1...N - номера колонок, в которых искать соответствие;

scolumnname1...N - колонки могут задаваться их названиями;

svalue1... N - текстовые значение для отбора;

dvalue2... N - числовые или другие значения для отбора.

Если не указаны колонки и их значения у фильтра - то добавляется все содержимое полностью.

**add\_rows\_from\_eq\_table\_by\_regex\_filter**

Добавить в таблицу из аналогичной таблицы строки, у которых значения в колонках соответствуют заданным шаблонным значениям в кодировке REGEX.

Вызов:

ts\_table(int iTableDescr, "add\_rows\_from\_eq\_table\_by\_regex\_filter", int iTableFrom, bool bstoponfirstnoteq, bool bsum, int icolumnindex1, string value1, ... string scolumnnameN, double valueN);

Здесь:

iTableDescr - дескриптор таблицы, к которой добавить строки;

iTableFrom - дескриптор таблицы, из которой взять эти строки;

bstoponfirstnoteq - остановить выборку как только после найденых строк появилась первая несовпадающая (т.е. считать что все нужные строки локализованы в одном месте таблицы);

bsum - если true, то добавлять строки как add\_row\_sum (т.е. если есть строка совпадающая по ключевой колонке, то просто суммировать значения в числовых колонках);

icolumnindex1...N - номера колонок, в которых искать соответствие;

scolumnname1...N - колонки могут задаваться их названиями;

svalue1... N - текстовые значение для отбора;

dvalue2... N - числовые или другие значения для отбора.

Если не указаны колонки и их значения у фильтра - то добавляется все содержимое полностью.

**add\_rows\_from\_eq\_table\_by\_extended\_filter**

Скопировать строки из одной таблицы в другую, используя расширенный фильтр.

На примере показано как скопировать из таблицы iTable в iTableFiltered все записи без суммирования числовых переменных одинаковых строк у которых в колонке с названием "Колонка1" значение "СТЕНА", значение в колонке "Колонка2" - содержит "3", в колонке "Колонка3" содержится в любом месте "В22" или "В22" (В - или на русском или на английском) и в колонке "Колонка 4" число больше или равное 2.

bool stoponfirstnoteq = false;

bool add\_sum = false;

ts\_table(iTableFiltered, "add\_rows\_from\_eq\_table\_by\_extended\_filter", iTable, bool stoponfirstnoteq, bool add\_sum,

"(", "ANY", "Колонка1", "=", "СТЕНА", "", "AND",

"" , "ANY", "Колонка2", "CONTAINS", "3", "", "AND",

"" , "ANY", "Колонка3", "REGEXMATCH", ".\*[BВ]22.\*", "", "AND",

"", "ANY", "Колонка4", ">=", 2, ")" );

Здесь

iTableFiltered - дескриптор таблицы, в которую будут копироваться строки из таблицы iTable;

stoponfirstnoteq - остановить копирование при первом столкновении с неподходящей строкой;

add\_sum - при собпадении данных у строк по ключевой колонке - суммировать числовые значения;

**Руководство по LABPP\_Automat для ARCHICAD**

Скопировать строки из одной таблицы в другую, используя расширенный фильтр, передаваемый в виде таблицы.

На примере показано как скопировать из таблицы iTable в iTableFiltered все записи без суммирования числовых переменных одинаковых строк у которых в колонке с названием "Колонка1" значение "СТЕНА", значение в колонке "Колонка2" - содержит "3", в колонке "Колонка3" содержится в любом месте "В22" или "В22" (В - или на русском или на английском) и в колонке "Колонка 4" число больше или равное 2.

bool stoponfirstnoteq = false;

bool add\_sum = false;

int iTableFilter;

object("create", "ts\_table", iTableFilter);

ts\_table(iTableFilter, "create\_structure\_for\_filter\_table");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "(", "ANY", "Колонка1", "=", "СТЕНА", "", "AND");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "ANY", "Колонка2", "CONTAINS", "3", "", "AND");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "ANY", "Колонка3", "REGEXMACH", ".\*[BВ]22.\*", "", "AND");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "ANY", "Колонка4", ">=", 2, ")", "");

ts\_table(iTableFiltered, "add\_rows\_from\_eq\_table\_by\_extended\_filter", iTable, bool stoponfirstnoteq, bool add\_sum);

Здесь

iTableFiltered - дескриптор таблицы, в которую будут копироваться строки из таблицы iTable;

stoponfirstnoteq - остановить копирование при первом столкновении с неподходящей строкой;

add\_sum - при собпадении данных у строк по ключевой колонке - суммировать числовые значения;

**get\_value\_of**

Получить значение из заданной колонки текущей строки таблицы

Пример. Получить значение из колонки №0 текущей строки таблицы в переменную objectname:

string objectname;  
ts\_table(TableDescr1,"get\_value\_of",0,objectname);

**get\_rows\_count**

Получить количество строк в таблице в заданную переменную

int rowcount;  
ts\_table(TableDescr1,"get\_rows\_count", rowcount);

**get\_columns\_count**

Получить количество колонок в таблице в заданную переменную

int colcount;  
ts\_table(TableDescr1,"get\_columns\_count", colcount);

**set\_columnstypes**

Задать тип определенным колонкам.

Вызов:

ts\_table(iTableDescr,"set\_columnstypes",string col1name, string stype1, string col2name, string stype2,...);

или

ts\_table(iTableDescr,"set\_columnstypes",int col1, string stype1, int col2, string stype2,...);

Здесь:

col1name, col2name - колонки по именам, можно непоследовательно,

stype1, stype2 - названия типов ("string", "int", "double", "bool");

col1, col2 - номера колонок.

**get\_heading\_of**

Получить заголовок определенной колонки таблицы.

Вызов:

ts\_table(iTableDescr,"get\_heading\_of", int col, string &colheading);

Здесь:

col - номер колонки (индекс от 0).

**print\_to\_str**

Экспорт всего содержимого таблицы в строковую переменную. Удобно при технологической работе для анализа содержимого таблицы в ходе отладки программ. Результат можно выводить через cout << text; в окно сообщений.

Вызов:

ts\_table(iTableDescr,"print\_to\_str", string &text);

Здесь:

text - строка с результатом - содержимое таблицы построчно с кол-вом строк, и перечислением значений с именами колонок.

**set\_columns\_to\_add\_sum**

Задать колонки для суммирования при добавлении новой записи, если такая уже есть в таблице (функцией "add\_row\_sum").

Предыдущее назначение суммируемых колонок отменяется.

По умолчанию все колонки суммируемые.

Если задать команду без списка колонок, то все колонки будут не суммируемые.

Вызов:

отменить суммирование всех колонок для команды "add\_row\_sum":

ts\_table(iTable,"set\_columns\_to\_add\_sum");

задать суммирование по колонкам:

ts\_table(iTable,"set\_columns\_to\_add\_sum",

int colnum1/string colname1,

int colnum2/string colname2,

...

int colnumN/string colnameN

);

Здесь:

colnumN - номер колонки, в которой будет суммироваться значение, если значение первичного ключа (набор колонок, заданных в команде "set\_first\_key") совпадает с новой записью;

colname1 - то же можно задать названием колонки.

Пример.

Суммировать количество из таблицы iTableMatRes по GUID'у в таблицу iTableMatResTmp и выгрузить ее в Excel, изменив названия и последовательность колонок.

int ires = excel\_attach();

if (ires != 0)

{

tsalert(-1, "Ошибка во время выполнения", "Не получается подключиться к файлу excel", sWorkBook);

return -1;

}

ires = excel\_request("workbook\_select", sWorkBook);

if (ires != 0)

{

tsalert(-1, "Ошибка во время выполнения", "Не получается переключиться в файл excel", sWorkBook);

excel\_detach();

return -1;

}

string sSheetName;

sSheetName = "Результат материалы";

ires = excel\_request("sheet\_select", sSheetName);

if (ires != 0) {

tsalert(-1, "Ошибка во время выполнения", "Не получается переключиться на страницу", sSeetName);

excel\_detach();

return -1;

}

int iTableMatResTmp;

object("create", "ts\_table", iTableMatResTmp);

ts\_table(iTableMatResTmp, "import\_columns\_from\_table", iTableMatRes);

ts\_table(iTableMatResTmp, "set\_first\_key", "GUID материала");

ts\_table(iTableMatResTmp, "set\_columns\_to\_add\_sum", "Кол-во материала");

ts\_table(iTableMatResTmp, "add\_rows\_from\_eq\_table\_by\_filter", iTableMatRes, false, true);

ts\_table(iTableMatResTmp, "mapping\_columns\_to\_export",

-1, "Поз.", "",

"Наименование материала (Rus)", "Наименование и техническая характеристика", "",

-1, "Код продукции","",

-1, "Поставщик", "",

"Ед.изм. материала", "Ед.изм.", "",

"Кол-во материала", "Кол-во", "",

-1, "Масса 1 ед.,кг","",

-1, "Примечание", ""

);

ts\_table(iTableMatResTmp, "export\_to\_excel", 1, "A", 1, 0, -1);

breakpoint(101);

object("delete",iTableMatResTmp);

excel\_detach();

**Функции для выгрузки и загрузки таблиц**

Для выгрузки в электронные таблицы и загрузки данных из них, для быстрого заполнения данных в элементы диалогов и получения таблиц из них созданы ряд функций, оптимизирующих эту работу. Можно выводить или получать целиком, построчно, только заголовки или структуру, разрешать или запрещать вывод определенных колонок.

export\_to\_excel

Выгрузка таблицы в Excel. Можно выводить все содержимое целиком или можно сначала вывести только заголовок, потом по одной строке в цикле с анализом или вставками между строк.

Вызов:

ts\_table(iTableDescr, "export\_to\_excel", int withheader, string startcolletter, int startexcelrow, int starttablerow, int maxrows);

Здесь:

withheader = 0/1 - с заголовками или без

startcollerrer - "A" или строка с другой буквой - начальная колонка электронной таблицы для экспорта (если "", то начало - текущая колонка где стоит курсор в Excel)

starttablerow - с какой строки начинать вывод из таблицы (0 - с первой)

maxrows - сколько строк выводить. (-1 - все)

import\_from\_excel

Импортировать таблицу из Excel.

ts\_table(iTableDescr, "import\_from\_excel", string startcolletter, int startrow, int maxrows, bool add\_sum, bool deleteoldrows);

Здесь:

startcolletter - буква колонки (если "" - то с текущей ячейки),

startrow - номер строки (индекс от 1),

maxrows - максимальное количество строк (если -1 - то до первой пустой ячейки в первой колонке),

add\_sum - использовать при добавлении строк механизм "add\_row\_sum" (см. выше) - 0/1,

deleteoldrows - очистить или нет содержимое таблицы перед загрузкой.

Загрузка таблицы останавливается на строке, где в левой колонке ячейка пустая.

import\_columns\_from\_excel

Импорт колонок из таблицы Excel. Задается начальная ячейка или с текущей, загрузка идет слева направо по заданному количеству колонок или до первой пустой ячейки. Все колонки по умолчанию текстовые. Чтобы задать типы колонок используйте директиву "set\_columntypes".

Вызов:

ts\_table(iTableDescr, "import\_columns\_from\_excel", string startcolletter, int startrow, int maxcols);

Здесь:

startcolletter - буква начальной колонки в Excel, или "" если с текущей,

startrow - номер строки в Excel (-1 - означает с текущей),

maxcols - количество колонок (-1 - означает до первой пустой ячейки);

Загрузка колонок останавливается на первой пустой ячейке.

import\_columns\_from\_table

Импортировать заголовки и типы колонок из другой таблицы.

Вызов:

ts\_table(iTableDescr, "import\_columns\_from\_table", int iTableDescrFrom);

Здесь:

iTableDescrFrom - дескриптор таблицы, откуда импортировать колонки.

import\_columns\_from\_dialogcontrol

Импортировать названия колонок из табличного элемента диалога (ListBox и т.п.)

Вызов:

ts\_table(iTableDescr, "init\_columns\_from\_dialogcontrol", int iDailogControlDescr);

Здесь:

iDailogControlDescr - дескриптор элемента диалога, из которого брать колонки.

import\_from\_dialogcontrol

Импортировать содержимое строк из табличного элемента диалога.

Вызов:

ts\_table(iTableDescr, "import\_from\_dialogcontrol", int iDailogControlDescr);

Здесь:

iDailogControlDescr - дескриптор элемента диалога, из которого брать строки.

export\_to\_dialogcontrol

Экспортировать строки таблицы в табличный элемент диалога.

ts\_table(iTableDescr, "export\_to\_dialogcontrol", int iDailogControlDescr, int startrow, int maxrows);

Здесь:

iDailogControlDescr - дескриптор элемента диалога, куда записывать строки,

startrow - начальная строка таблицы (индекс от 0),

maxrows - максимальное количество строк (-1 означает все строки).

export\_to\_dialogcontrol\_by\_filter

Экспортировать строки таблицы в табличный элемент диалога с использованием фильтра по колонкам таблицы.

В каждую строку таблицы подставляется номер строки таблицы, так, что если какие-то строки отфильтрованы при экспорте, то сопоставление строки в списке элемента диалога и в таблице производится получением пользовательского свойства у текущей выбранной строки элемента списка диалога (getuservalue).

ts\_table(iTableDescr, "export\_to\_dialogcontrol\_by\_filter", int iDailogControlDescr, int startrow, int maxrows, bool bstoponfirstnoteq, string scolname1, **void** colvalue1, ...);

Здесь:

iDailogControlDescr - дескриптор элемента диалога, куда записывать строки,

startrow - начальная строка таблицы (индекс от 0),

maxrows - максимальное количество строк (-1 означает все строки).

scolname1... - заголовок колонки в таблице, по которой отфильтровывать строки при экспорте;

colvalue1... - эталонное значение для сравнения.

export\_to\_dialogcontrol\_by\_regex\_filter

Экспортировать строки таблицы в табличный элемент диалога с использованием фильтра по колонкам таблицы.

Фильтрация производится с использованием шаблонов REGEX.

В каждую строку таблицы подставляется номер строки таблицы, так, что если какие-то строки отфильтрованы при экспорте, то сопоставление строки в списке элемента диалога и в таблице производится получением пользовательского свойства у текущей выбранной строки элемента списка диалога (getuservalue).

ts\_table(iTableDescr, "export\_to\_dialogcontrol\_by\_regex\_filter", int iDailogControlDescr, int startrow, int maxrows, bool bstoponfirstnoteq, string scolname1, **void** colvalue1, ...);

Здесь:

iDailogControlDescr - дескриптор элемента диалога, куда записывать строки,

startrow - начальная строка таблицы (индекс от 0),

maxrows - максимальное количество строк (-1 означает все строки).

scolname1... - заголовок колонки в таблице, по которой отфильтровывать строки при экспорте (можно использовать номер колонки - индексация от 0);

colvalue1... - эталонное значение для сравнения.

[(см. regex шаблоны).](#_topic_REGEX)

export\_to\_dialogcontrol\_by\_extended\_filter

Записать содержимое таблицы в табличный элемент диалога с использованием расширенного фильтра.

На примере показано как скопировать из таблицы iTable в табличный элемент диалога (iListBox) все записи от начала до конца таблицы с учетом фильтра - в колонке с названием "Колонка1" значение "СТЕНА", значение в колонке "Колонка2" - содержит "3", в колонке "Колонка3" содержится в любом месте "В22" или "В22" (В - или на русском или на английском) и в колонке "Колонка 4" число больше или равное 2.

int startrow = 0;

bool stoponfirstnoteq = false;

int maxrows = -1;

ts\_table(iTable, "add\_rows\_from\_eq\_table\_by\_extended\_filter", iListBox, startrow, stoponfirstnoteq, maxrows,

"(", "ANY", "Колонка1", "=", "СТЕНА", "", "AND",

"" , "ANY", "Колонка2", "CONTAINS", "3", "", "AND",

"" , "ANY", "Колонка3", "REGEXMATCH", ".\*[BВ]22.\*", "", "AND",

"", "ANY", "Колонка4", ">=", 2, ")" );

Здесь

iTable - дескриптор таблицы, из которой будут копироваться строки в табличный элемент диалога;

int startrow - индекс строки, с которой начинать копирование;

int maxrows - количество строк (-1 - все строки до конца таблицы);

bool bstoponfirstnoteq - остановиться при первом несоответствии строки условию фильтра.

export\_to\_dialogcontrol\_by\_extended\_filter\_table

Записать содержимое таблицы в табличный элемент диалога с использованием расширенного фильтра, переданного в виде таблицы.

Удобно когда необходимо задавать условия фильтров по ходу выполнения программы.

На примере показано как скопировать из таблицы iTable в табличный элемент диалога (iListBox) все записи от начала до конца таблицы с учетом фильтра - в колонке с названием "Колонка1" значение "СТЕНА", значение в колонке "Колонка2" - содержит "3", в колонке "Колонка3" содержится в любом месте "В22" или "В22" (В - или на русском или на английском) и в колонке "Колонка 4" число больше или равное 2.

int startrow = 0;

bool stoponfirstnoteq = false;

int maxrows = -1;

int iTableFilter;

object("create", "ts\_table", iTableFilter);

ts\_table(iTableFilter, "create\_structure\_for\_filter\_table");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "(", "ANY", "Колонка1", "=", "СТЕНА", "", "AND");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "ANY", "Колонка2", "CONTAINS", "3", "", "AND");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "ANY", "Колонка3", "REGEXMACH", ".\*[BВ]22.\*", "", "AND");

ts\_table(iTableFilter, "add\_row\_to\_filter\_table", "", "ANY", "Колонка4", ">=", 2, ")", "");

ts\_table(iTable, "add\_rows\_from\_eq\_table\_by\_extended\_filter", iListBox, startrow, stoponfirstnoteq, maxrows, iTableFilter);

Здесь

iTable - дескриптор таблицы, из которой будут копироваться строки в табличный элемент диалога;

int startrow - индекс строки, с которой начинать копирование;

int maxrows - количество строк (-1 - все строки до конца таблицы);

bool bstoponfirstnoteq - остановиться при первом несоответствии строки условию фильтра.

resetofffromexport

Запретить экспорт определенных колонок таблицы. Запрет, выполненный предыдущим вызовом этой функции отменяется. Можно перечислять сколько угодно колонок, можно не подряд.

ts\_table(iTableDescr,"resetofffromexport",int col1, int col2);  
ts\_table(iTableDescr,"resetofffromexport",string col1name, string col2name,...);

Здесь:

col1,col2 - номера колонок (индекс от 0).

col1name, col2name - имена колонок.

export\_to\_LABPP\_Table

Экспортировать строки таблицы в табличный элемент LABPP\_Table.gsm.

ts\_table(iTable, "export\_to\_LABPP\_Table", bool withheader, int startcolindex, int startObjectrow, int starttablerow, int maxrows, string digit\_delimiter);

или

ts\_table(iTable, "export\_to\_LABPP\_Table", bool withheader, int startcolindex, int startObjectrow, int starttablerow, int maxrows, string digit\_delimiter, int iLABPP\_TableGuidDescr);

Здесь:

withheader = 0/1 - с заголовками или без

startcolindex - индекс колонки в элементе таблицы, с которой надо выводить в таблицу ( начиная с 1);

startObjectrow - индекс строки в в элементе таблицы, с которой надо выводить в таблицу (начиная с 1);

starttablerow - с какой строки начинать вывод из таблицы (начиная с 0-й строки)

digit\_delimiter - разделитель целой и дробной части заменить на этот символ ("." или ",")

maxrows - сколько строк выводить. (-1 - все).

Пример.

Создать таблицу и записать ее содержимое в выбранный элемент LABPP\_Table в проекте.

int main()

{

int iTable;

object("create", "ts\_table", iTable); // создать таблицу

// Создаем колонки таблицы

ts\_table(iTable, "add\_column", 0, "string", "Моя первая колонка");

ts\_table(iTable, "add\_column", 1, "double", "Моя вторая колонка с числом");

ts\_table(iTable, "add\_column", 2, "string", "Примечания");

// Добавляем строки

ts\_table(iTable, "add\_row", 0, "Строка 1", 1, 10000.1111, 2, "Примечание 1");

ts\_table(iTable, "add\_row", 0, "Строка 2", 1, 20000.2222);

ts\_table(iTable, "add\_row", 0, "Строка 3", 1, 30000.335, 2, "Примечание 3");

// Загрузить из выбранных элементов объекты LABPP\_Table

ac\_request("load\_elements\_list\_from\_selection", 1, "ObjectType", "ObjectName", "LABPP\_Table", "MainFilter", 2);

ac\_request("get\_loaded\_elements\_list\_count", 1);

int icount = ac\_getnumvalue();

int i;

bool withheader = true;

int startcolindex = 1;

int startObjectrow = 1;

int starttablerow = 0;

int maxrows = -1;

int maxcolumns;

// получить число колонок и строк

ts\_table(iTable, "get\_columns\_count", maxcolumns);

ts\_table(iTable, "get\_rows\_count", maxrows);

// Перебрать все отобранные объекты LABPP\_Table. Если нужен только единственный - можно цикл не делать.

for (i = 0; i < icount; i++)

{

// Установить текущим элементом объект из списка 1 № i

ac\_request("set\_current\_element\_from\_list", 1, i);

// Записать в него содержимое таблицы

ts\_table(iTable, "export\_to\_LABPP\_Table", withheader, startcolindex, startObjectrow, starttablerow, maxrows,",");

// Задать в объекте количество использованных строк

ac\_request("set\_object\_property\_value", "TS\_RowsUsedCount", maxrows+1);

// Задать в объекте количество использованных колонок

ac\_request("set\_object\_property\_value", "TS\_ColumnsUsedCount", maxcolumns);

// Дать команду объекту выполнить внутренний перерасчет

ac\_request("set\_object\_property\_value", "TS\_DoRecalcAll", 1);

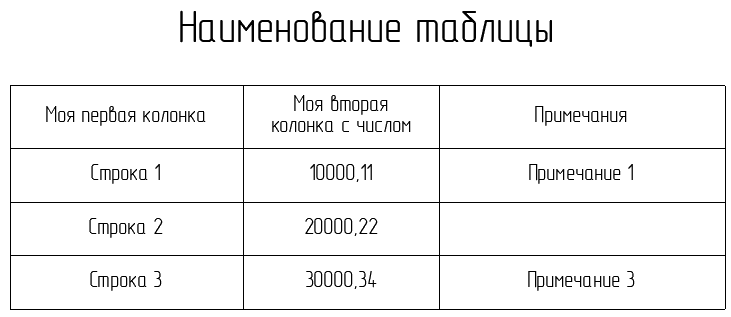
}

// Удалить из памяти таблицу

object("delete", iTable);

}

В результате выполнения программы в таблицу будут занесены данные.



set\_columns\_width\_when\_export

Задать размер колонок при экспорте в табличный элемент диалога.

Вызов:

ts\_table(iTable,"set\_columns\_width\_when\_export",40);

Здесь 40 - ширина колонок в пискелях.

mapping\_columns\_to\_export

Задать новую последовательность колонок при экспорте.

Вызов:

отменить маппирование и при следующем экспорте:

ts\_table(iTable,"mapping\_columns\_to\_export");

задать маппирование:

ts\_table(iTable,"mapping\_columns\_to\_export",

int basecolnum1/string basecolname1, string altcolname1, bool/int/double/string default1,

int basecolnum2/string basecolname2, string altcolname2, bool/int/double/string default2,

...

int basecolnumN/string basecolnameN, string altcolnameN, bool/int/double/string defaultN

);

Здесь:

basecolnumN - номер базовой колонки, из которой брать данные. Если -1, то будет пустая колонка, заполненная значением defaultN;

basecolnameN - здесь может быть указано имя базовой колонки;

altcolnameN - альтернативное название колонки, если значение "", то берется название базовой колонки;

defaultN - значение, которое будет выводиться в случае, если задана пустая колонка (-1). Если задать "", то заполнения не будет.

Пример.

ts\_table(iTableMatRes, "mapping\_columns\_to\_export",

-1, "Поз.", "",

"Наименование материала (Rus)", "Наименование и техническая характеристика", "",

-1, "Тип, марка, обозначение документа, опросного листа", "",

-1, "Поставщик", "",

"Ед.изм. материала", "Ед.измерения", "",

"Кол-во материала", "Кол.", "",

-1, "Масса 1 ед.,кг","",

-1, "Примечание", ""

);

get\_base\_colindex\_by\_mapped\_column

Получить номер базовой колонки таблицы по ранее заданной последовательности колонок для экспорта.

Вызов:

int irealycolnum = ts\_table(int iTable, "get\_base\_colindex\_by\_mapped\_column", string sMappingColName);  
int irealycolnum = ts\_table(int iTable, "get\_base\_colindex\_by\_mapped\_column", int iMappingColIndex);

Здесь:

irealycolnum - номер базовой колонки, на которую указывает псевдоним или порядковый номер псевдонима.

sMappingColName - псевдоним или индекс (от 0 до N) псевдонима колонки.

Возвращает индекс базовой колонки (область определения от 0 до N) или -1.

Пример.

Получить номера базовых колонок для псевдонима "Ед.изм. материала" и псевдонима с индексом 3 ("Поставщик")

ts\_table(iTableMatRes, "mapping\_columns\_to\_export",  
 -1, "Поз.", "",  
 "Наименование материала (Rus)", "Наименование и техническая характеристика", "",  
 -1, "Тип, марка, обозначение документа, опросного листа", "",  
 -1, "Поставщик", "",  
 "Ед.изм. материала", "Ед.измерения", "",  
 "Кол-во материала", "Кол.", "",  
 -1, "Масса 1 ед.,кг","",  
 -1, "Примечание", ""  
 );  
  
int irealycolnum;  
irealycolnum = ts\_table(iTableMatRes, "get\_base\_colindex\_by\_mapped\_column", "Ед.изм. материала");  
// результат - в irealycolnum теперь номер базовой колонки ("Ед.измерения") в таблице iTableMatRes.  
  
irealycolnum = ts\_table(iTableMatRes, "get\_base\_colindex\_by\_mapped\_column", 3);  
// результат - в irealycolnum теперь -1, т.к. в таблице iTableMatRes при маппировании базовая колонка не указана (-1).

get\_count\_of\_mapped\_columns

Узнать сколько колонок таблицы смаппировано для экспорта.

Вызов:

int mappedcolumnscount = ts\_table(int iTable,"get\_count\_of\_mapped\_columns");

Здесь:

iTable - дескриптор таблицы,

mappedcolumnscount = количество колонок, которые смаппированы для экспорта.

export\_to\_csv

Выгрузить содержимое таблицы в текстовую переменную.

ts\_table(iTable, "export\_to\_csv", bool bWithHeadings, string column\_separator, string digit\_separator, int start\_row, int numrows, string csv\_result);

Здесь:

iTable - дескриптор таблицы,

bWithHeadings - выводить заголовки таблицы или нет,

column\_separator - символ разделителя колонок - обычно ";",

digit\_separator - разделитель целой и дробной частей чисел - обычно ",",

start\_row - с какой строки выводить таблицу,

num\_rows - сколько строк выводить (если -1 - то выводить все);

Пример

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Импорт ts\_table из текста в формате CSV

// и экспорт в этот формат

// LABPP 2021

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int main()

{

int iTable;

object("create", "ts\_table", iTable);

string csvstr;

csvstr = "\"Заголовок 1\";\"Заголовок 2\";\"Заголовок 3\"\n\

\"Значение 11 две кавычки \"\" текст\";123.3;\"Значение 31\"\n\

\"Значение 12\";321.21;\"Значение 32\"";

ts\_table(iTable, "import\_from\_csv", csvstr, ";");

string str;

ts\_table(iTable, "print\_to\_str", str);

coutvar << str;

string csv\_result;

ts\_table(iTable, "export\_to\_csv", true, ";", "," , 0, -1, csv\_result);

coutvar << csv\_result;

string FileNameAndPath = "c:\\LabPP\\test\_07.csv";

int iret = 0;

int iFileDescr;

object("create", "ts\_file", iFileDescr);

iret = ts\_file(iFileDescr, "open", FileNameAndPath, "create", "we");

if (iret == 0)

{

ts\_file(iFileDescr, "write", csv\_result);

ts\_file(iFileDescr, "close");

}

else

{

tsalert(-3, "Ошибка записи", "Проверьте, пожалуйста доступность файла для записи", FileNameAndPath);

}

object("delete", iFileDescr);

object("delete", iTable);

}

import\_from\_csv

Загрузить таблицу из текстовой переменной в формате CSV.

ts\_table(iTable, "import\_from\_csv", string csv\_source, string column\_separator);

Здесь:

iTable - дескриптор таблицы,

csv\_source - переменная с текстом в формате csv,

column\_separator - разделитель колонок - обычно ";".

export\_to\_ac\_object\_parameter

Выгрузить таблицу в параметр GDL типа массив.

Размерность массива изменяется у конкретного экземпляра библиотечного элемента, в который происходит выгрузка.

Вызов:

int ires = ts\_table(int iTable, "export\_to\_ac\_object\_parameter", string sParameterName);

Здесь:

iTable - дескриптор таблицы,

sParameterName - имя GDL параметра (массив).

Пример.

У отобранных в список №1 GDL элементов имеется текстовый табличный параметр sMyParameters.

|  |  |  |
| --- | --- | --- |
| Имя | Значение | Тип |
| Мой параметр 1 | 8 | число |
| Мой параметр 2 | значение параметра 2 | текст |

Требуется у элемента с индексом 0 в списке №1 :

1. считать значение Мой параметр 1

2. если надо, то создать и записать новый параметр с именем "Мой новый текстовый параметр" и записать туда значение "значение 3"

int iTable;  
**object**("create", "ts\_table", iTable);  
int ires = ac\_request("set\_current\_element\_from\_list", 1, 0);  
ac\_request\_special("get\_element\_value", "GDL","sMyParameters", iTable); // считать содержимое и структуру GDL параметра в таблицу iTable  
  
string str;  
ts\_table(iTable, "print\_to\_str", str);  
coutvar << str;  
  
//считаем значение Мой параметр 1 в svalue;  
ires = ts\_table(iTable,"search",0,"Мой параметр 1");  
string svalue;  
**if**(ires >=0) {  
 ts\_table(iTable, "select\_row", ires);  
 ts\_table(iTable, "get\_value\_of", 0, svalue);  
}  
  
//добавим, если такого еще нет нужно, считаем значение Мой параметр 1 в svalue;  
ires = ts\_table(iTable,"search",0,"Мой новый текстовый параметр");  
**if**(ires < 0) {  
 ts\_table(Table,"add\_row", 0, "Мой новый текстовый параметр", 1, "значение 3", 2, "строка");  
}  
**else**  
{  
 ts\_table(iTable, "select\_row", ires);  
 ts\_table(iTable, "set\_value\_of", 1, "значение 3");  
}

// Записать таблицу назад в параметр текущего элемента. Размер таблицы-параметра GDL - изменить с учетом размера таблицы iTable.

ires = ts\_table(iTable, "export\_to\_ac\_object\_parameter", "sMyParameters");

**set\_column\_user\_value**

Задать пользовательское значение для определенной колонки таблицы.

Например для того, чтобы различать колонки между собой для своих целей в программе.

Вызов:

int ires = ts\_table(int iTable, "set\_column\_user\_value", int icol, string/double/int/bool valuetoset);

Здесь:

iTable - дескриптор таблицы;

icol - номер колонки таблицы;

vsluetoset - значение, которое надо присвоить колонке.

**get\_column\_user\_value**

Считать пользовательское значение у определенной колонки таблицы.

Вызов:

int ires = ts\_table(int iTable, "get\_column\_user\_value", int icol, string/double/int/bool vartogetvalue);

Здесь:

iTable - дескриптор таблицы;

icol - номер колонки таблицы;

vartogetvalue - имя переменной, куда считать значение.

**set\_column\_user\_genesis**

Задать пользовательское значение генезиса для определенной колонки таблицы.

Например для того, чтобы различать колонки между собой для своих целей в программе.

Вызов:

int ires = ts\_table(int iTable, "set\_column\_user\_genesis", int icol, string sgenesis);

Здесь:

iTable - дескриптор таблицы;

icol - номер колонки таблицы;

sgenesis - значение генезиса текстом.

**get\_column\_user\_genesis**

Считать пользовательское значение генезиса из определенной колонки таблицы.

Например для того, чтобы различать колонки между собой для своих целей в программе.

Значение должно быть раньше записано при помощи команды "get\_column\_user\_genesis".

Вызов:

int ires = ts\_table(int iTable, "get\_column\_user\_genesis", int icol, string sgenesis);

Здесь:

iTable - дескриптор таблицы;

icol - номер колонки таблицы;

sgenesis - значение генезиса текстом.

**delete\_rows**

Удалить заданное количество строк из таблицы начиная с заданной позиции.

Вызов:

ts\_table(int iTable, "delete\_rows", int index, int count);

Здесь:

iTable - дескриптор таблицы;

index - индекс строки (в таблице строки нумеруются от 0 до n-1);

count - количество удаляемых строк.

Чтобы удалить все строки используйте "clear\_rows".

**set\_value\_of\_toallrows**

Задать значение в ячейка для всех строк таблицы начиная с заданной строки по заданную конечную строку.

Вызов:

ts\_table(int iTable, "set\_value\_of\_toallrows", index/string column\_index\_or\_name, bool/int/double/string newvalue, int irowfrom, int irowto);

Здесь:

iTable - дескриптор таблицы;

column\_index\_or\_name - индекс или имя колонки;

newvalue - значение, которое требуется установить ячейкам;

irowfrom - индекс строки, с которой необходимо задать значение (от 0 до n-1);

irowto - индекс последней строки для изменения (если -1 - то до конца таблицы).

**Функции стандартной библиотеки**

**Преобразование**

**atoi**

Перевод строковой переменной в целое число

Вызов:

string ivalue = atoi(string svalue);

**itoa**

Перевод целого числа в строку.

string svalue = itoa(int ivalue);

**atof**

Перевод строковой переменной в число с плавающей точкой

double dresult = atof(string svalue);

Здесь: svalue - число, записанное текстом, возвращаемое значение - число с плавающей точкой.

**ctos**

Перевод кода символа в строку. Функция полезна при динамическом формировании имени колонки в Excel и т.п.

Формат обращения:

string ssymbol = ctos(int isymbolcode);

Здесь:

icymbolcode - код символа.

ssymbol - строковая переменная с символом, соответствующим коду.

Пример:

Сформировать текстовый адрес ячейки Excel во второй колонке во второй строке.

int chcolumn = 'A'+1;  
string ssymbol = ctos(chcolumn);  
string address = ssymbol+itoa(2)+":"+ssymbol+itoa(2);  
cout << address;

В окно сообщений будет выведено "B2:B2"

**grad\_to\_radian**

Перевести градусы в радианы.

Вызов:

double grad\_to\_radian(double grad);

Здесь : grad - угол в градусах. Возвращаемое значение - число в радианах.

Перевести градусы в радианы.  
Пример. Конвертировать в радианы 180 градусов:

double result = grad\_to\_radian(180);

В результате переменная result будет содержать число Пи (3.14....).

**radian\_to\_grad**

Перевести угол в радианах в градусы.

Вызов:

double radian\_to\_grad(double rad);

Пример. Перевести угол 3.14 в градусы.

double result = radian\_to\_grad(3.14);

В результате переменная result будет содержать число 180.

**sprintf**

Сформировать строку по принципу printf.

Вызов:

string sprintf( string format, value);

Пример. Вывести чило с двумя знаками после запятой.

string sresult = sprintf("%.2f", 2.024);

В результате переменная sresult будет содержать строку "2.02".

Еще примеры.

|  |  |  |  |
| --- | --- | --- | --- |
| Требуемый результат | Формат | Тип параметра | Пример конструкции |
| 001 | "%03i" | int | int m = 10;  string s = sprintf("%03i",m); |

**ecvt\_french**

Переводит число в строку с разделителем - запятой и пробелами, отделяющими тысячные разряды.

Вызов:

string svalue = ecvt\_french(double dvalue);

Пример.

Перевести в строку число 111222333.11.

double dvalue = 111222333.11;  
string sresvalue = ecvt\_french(svalue);  
cout << sresvalue;

Результат - в окне сообщений появится строка "111 222 333,11"

**ecvt**

Переводит число в строку без пробелов, отделяющих тысячные разряды. Разделитель целой и дробной части - точка.

Вызов:

string svalue = ecvt(double dvalue);

Пример.

Перевести в строку число 111222333.11.

double dvalue = 111222333.11;  
string sresvalue = ecvt(svalue);  
cout << sresvalue;

Результат - в окне сообщений появится строка "111222333.11"

**Математические функции**

**abs**

Вернуть абсолютное значение.

double dres = abs(double dvalue);

В dres - число dvalue всегда положительное.

**max**

Вернуть максимальное из чисел

Вызов:

double dres = max(double dvalue1, double dvalue2);

В dres - максимальное из dvalue1 и dvalue2

**min**

Вернуть минимальное из чисел

Формат команды:

double dres = min(double dvalue1, double dvalue2);

В dres - минимальное из чисел dvalue1 и dvalue2.

**rand**

Генератор случайных чисел

Вызов:

double dres = rand();

Возвращает случайное число от 0 до 1;

**ln**

Натуральный логарифм.

**log**

Логарифм.

**sqrt**

Корень квадратный.

**sqr**

Корень заданной степени.

**pow**

Возведение в степень.

**percent**

Вычисление процентов числа.

**tsround**

Правильное математическое округление.

Вызов:

double tsround(double dvalue, int numpos);

Здесь:

dvalue - значение, которое нужно округлить.

numpos - количество знаков после запятой.

**tsround\_best**

Округление в большую сторону. Один из вариантов алгоритма округления.

Округляет каждый знак после запятой начиная с самого младшего разряда. Так, что если он больше 5, то в больший разряд переходит 1.

Вызов:

double tsround\_best(dvalue, int numpos);

Здесь:

dvalue - значение, которое нужно округлить.

numpos - количество знаков после запятой.

**floor**

Функция округляет аргумент до наибольшего целого числа, которое меньше или равно аргументу.

Вызов:

double dresult = floor(double dvalue);

**ceil**

Функция ceil выполняет округление и возвращает ближайшее целое значение к dvalue, но это значение будет не  меньше самого dvalue.

Вызов:

double dresult = ceil(double dvalue);

**math\_const**

Получить значение математической константы (Пи и т.п.)

Вызов:

double math\_const(string sConstName);

Здесь:

sConstName - имя константы.

|  |  |
| --- | --- |
| Константа | Значетие/описание |
| Pi | 3.14159265359 |
| MinDouble | Минимальное значение double |
| MaxDouble | Максимальное значение double |
| e | 2.71828182846 |

**inrange**

Проверить находится ли значение в рамках заданного диапазона.

Вызов:

bool inrange(double value, string left\_parenthesis, double left\_margin, double right\_margin, string right\_parenthesis);

Здесь:

value - тестируемо значение;

left\_parenthesis = левая скобка ("]","(" или "[" - "от минус бесконечности", от "left\_margin не включительно", от "left\_margin включая left\_margin");

left\_margin - левая граница диапазона;

right\_parenthesis = правая скобка ("[",")" или "]" - "до плюс бесконечности", до "right\_margin не включительно", до "right\_margin включая right\_margin");

right\_margin - правая граница диапазона;

**Тригонометрические функции**

**cos**

Косинус угла. Угол задается в радианах.

Вызов:

double dresult = cos(double cornerinrad);

**sin**

Синус угла. Угол задается в радианах.

Вызов:

double dresult = sin(double cornerinrad);

**arcsin**

Аргсинус.

**arccos**

Арккосинус.

**tg**

Тангенс

**arctg**

Арктангенс.

**ctg**

Котангенс.

**arcctg**

Арккотангенс.

**Функции геометрического преобразования**

Обращение к функции ac\_request() с директивой geometry\_calc\_2d предоставляет широкий набор операций для модификации координат и определения взаиморасположения различных элементов.

**is\_curelem\_inside\_element\_polygon**

Определить лежит ли текущий элемент внутри полигона элемента, имеющего площадь опоры (штриховка, плита, 3d-сетка и т.п.). Возможно предварительное увеличение или уменьшение контура или отверстий полигона второй фигуры.

Вызов:

bool res = ac\_request("geometry\_calc\_2d"," is\_curelem\_inside\_element\_polygon",double x1,double y1, int iElemDescr, bool mainContOnly, double grow\_contour, double grow\_holes, int mode);

**is\_point\_on\_element\_polygon**

Определить лежит ли точка внутри полигона элемента, имеющего площадь опоры (штриховка, плита, 3d-сетка и т.п.)

Обращение

bool res = ac\_request("geometry\_calc\_2d","is\_point\_on\_element\_polygon",double x1,double y1, int iElemDedcr, bool mainContOnly, double grow\_contour, double, grow\_holes);

Здесь:

x1 и y1 - координаты исследуемой точки, iElemDescr - дескриптор объекта, чей полигон проверяется на содержание точки.

mainContOnly – если да, то учитывать только внешний контур второго полигона,

grow\_contour и grow\_holes – на сколько расширить или сузить контур и отверстия второго полигона.

Пример.

Считать элементы 3d-сеток с ID="Рельеф" в список элементов, взять первый, создать на его базе объект ac\_eleemnt\_guid, из него получить объект ac\_element. Получить рельеф в таблицу координат. И проверить лежит ли точка внутри полигона.

// считать таблицу координат элемент рельефа 3d mesh с ID="Рельеф"  
int iMeshCoordTable; // дескриптор таблицы координат полигона  
// считать элементы типа 3d-сетка с ID="Рельеф" в список № 2  
ac\_request("load\_elements\_list",2,"MeshType","ID","Рельеф","MainFilter",2);  
// определить сколько элементов содержит список № 2  
ac\_request("get\_loaded\_elements\_list\_count",2);  
int iicount = ac\_getnumvalue();  
**if**(iicount == 0)  
{  
 cout << "В проекте отсутствует 3d сетка с ID=Рельеф.\nПрограмма остановлена";  
 **return** -1;  
}  
// создать объект динамической таблицы для координат полигона  
**object**("create","ts\_table",MeshCoordTable);  
// установить фокус на первом элементе (индекс 0) списка № 2  
ac\_request("set\_current\_element\_from\_list",2, 0);  
// создать объект для guid элемента  
int iMeshGuidDescr; // дескриптор объекта guid  
**object**("create","ac\_element\_guid",iMeshGuidDescr);  
// записать из текущего элемента в списке № 2 guid в объект iMeshGuidDescr   
ac\_request("store\_cur\_element\_to\_descr", iMeshGuidDescr);  
int iMeshElemDedcr; // дескриптор объекта элемента  
// создать объект для работы с элементом  
**object**("create","ac\_element",iMeshElemDedcr);   
// загрузить объект элемент из guid  
ac\_request("load\_element\_from\_guid",iMeshElemDedcr,iMeshGuidDescr);  
// считать таблицу координат полигона   
ac\_request("get\_element\_value","SimpleCoordTable",iMeshCoordTable);  
// проверить сколько точек содержит полученная таблица координат полигона  
int irowcount;  
ts\_table(iMeshCoordTable, "get\_rows\_count", irowcount);  
cout << "Количество точек в контуре рельефа=" << irowcount << "\n";  
**object**("delete",MeshCoordTable);  
double x1=1.1, y1=2.2; // координаты исследуемой точки  
// проверить лежит ли точка с координатами x1, y1   
int res = ac\_request("geometry\_calc\_2d","is\_point\_on\_element\_polygon",x1,y1, iMeshElemDedcr);  
**if**(res == 1)   
{  
 cout << "Точка находится внутри контура полигона элемента";  
}

**rotate\_point\_and\_move**

Точку повернуть относительно заданного центра на заданный угол, перенести центр поворота на заданное расстояние и разместить точку на расстоянии с учетом масштабирования.

Вызов:

ac\_request("geometry\_calc\_2d","rotate\_point\_and\_move",double centerXfrom, double centerYfrom,double pXfrom,double pYfrom, double alpha\_rad, double centerXto, double centerYto,double scale,double &pXres, double& pYres);

centerXfrom, centerYfrom - исходные координаты центра поворота,

pXfrom, pYfrom - исходные координаты точки,

alpha\_rad - угол поворота точки вокруг центра,

centerXto, centerYto - новые координаты центра поворота,

scale - масштаб в диапазоне ]0,1],

Результат - pXres, PYres - новые координаты точки.

**get\_cross\_point\_of\_2lines**

Найти точку пересечения двух прямых, заданных точками.

Обращение

ac\_request("geometry\_calc\_2d","get\_cross\_point\_of\_2lines",double l1sX, double l1sY,double l1eX,double l1eY, double l2sX, double l2sY, double l2eX,double l2eY,double& pXres, double& pYres);

Здесь:

l1sX,l1sY,l1eX, l1eY - координаты начальной и конечной точек первой прямой,

l2sX,l2sY,l2eX, l2eY - координаты начальной и конечной точек второй прямой.  
Координаты точки пересечения прямых выводятся в pXres и pYres.

**get\_rot\_and\_move\_point**

Переместить точку в заданном направлении на заданное расстояние.

Обращение

ac\_request("geometry\_calc\_2d","get\_rot\_and\_move\_point",double P1X, double P1Y, double dL, double dAngleRad, double & pXres, double& pYres);

Здесь:

P1X, P1Y - исходные координаты точки, dL - расстояние, на которое нужно переместить точку, dAngleRad - угол против часовой стрелки от положительного направления оси X.

pXres, pYres - координаты перемещенной точки.

**get\_length\_2point**

Вычислить расстояние между двумя точками.

Вызов:

ac\_request("geometry\_calc\_2d","get\_length\_2point",double l1sX, double l1sY,double l1eX,double l1eY, double &resLength);

Здесь:

l1sX, l1sY, l1eX, lieY - координаты точек.

Расстояние между точками возвращается в resLength.

**is\_point\_on\_line**

Определить находится ли точка на линии, заданной двумя точками.

Вызов:

bool bres = ac\_request("geometry\_calc\_2d","is\_point\_on\_line",double P1X, double P1Y, double P2X, double P2Y, double PointX, double PointY);

P1X,P1Y,P2X,P2Y - начальная и конечная точки линии.

PointX, PointY - координаты исследуемой точки.

Результат bres - 1 или 0 в зависимости от того, лежит или нет точка на линии.

**get\_line\_angle**

ac\_request("geometry\_calc\_2d","get\_line\_angle",double l1sX, double l1sY,double l1eX,double l1eY, double &resAngleInRad);

Здесь:

l1sX,l1sY,l1eX, l1eY - координаты начальной и конечной точек отрезка.

Угол наклона в радианах возвращается в resAngleInRad.

**Специальные функции**

**codemeter - измерение времени выполнения фрагмента кода**

Формат обращения:

double dvalue = codemeter(int what);

Здесь what - 0 или 1.

Если задан 0, то таймер обнуляется.

Если задана 1 - то функция возвращает время в секундах с момента обнуления счетчика.

Если задана 2 - то функция возвращает время в милисекундах.

Пример.

Определить время выполнения цикла в 10000 проходов.

codemeter(0);  
int i;  
**for**(i=0;i<10000;i++)  
{  
 cout << i << "\n";  
}  
cout<< "Цикл в 10 000 оборотов выполнен за " << codemeter(1) << " секунд";

**ac\_save\_messages\_to\_file - сохранить текст из окна сообщений в файл**

Формат обращения:

ac\_save\_messages\_to\_file(string filepath);

Записывает содержимое окна сообщений в файл filepath

Пример.

Записать содержимое окна сообщений в файл c:\my\_file.txt:

ac\_save\_messages\_to\_file("c:\\my\_file.txt");

Обратите внимание на двойной обратный слэш.

В строках C++ одинарный обратный слэш используется в служебных целях. Так что в строках нужно писать двойной слэш, а программа при выполнении автоматически один уберет.

**Связь с квартирографией**

**solaris\_test**

Чтобы использовать функционал квартирографии ARCHICAD необходимо чтобы модуль квартирографии был установлен и доступен для обращения.

Тест выполняется командой ac\_request с директивой solaris\_test.

Форма обращения

int ires = ac\_request("solaris test");

При успешном тестировании возвращается 0.

int res;  
res = ac\_request("solaris\_test");  
**if**(res != 0) {  
 cout << "Нет связи с LabPP\_Solaris\n";  
 **return** -1;  
}

**get\_flat\_rooms**

Получить список помещений квартиры - ac\_request("get\_flat\_rooms"...)

Заполняет заданный внутренний список элементами зон, закрепленных за указанным маркером квартиры.

Обращение

int ires = ac\_request("get\_flat\_rooms",int iFlatGuidObjDescr, int iListNum);

Здесь:

iFlatGuidObjDescr - дескриптор объекта типа guid маркера квартиры/офиса.

iListNum - номер внутреннего списка элементов, в который записать зоны этой квартиры/офиса.

На выходе - 0 - отсутствие ошибки.

**runtimecontrol**

**workline**

Управление дорожкой процентов выполнения программы в главном окне.

Формат обращения:

runtimecontrol("workline", string sdirective, int ivalue);

или назначить текущее значение дорожки процентов и одновременно написать это значение в окно сообщений через каждые igap итераций

runtimecontrol("workline", "setpos\_and\_print\_through\_gap", int ivalue, int igap);

Здесь:

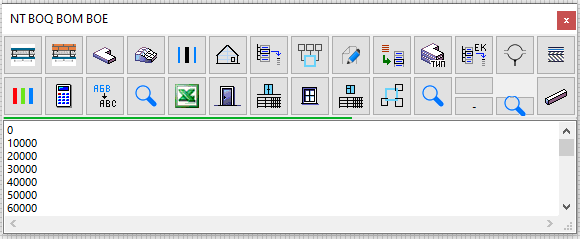
directive - директива, задающая что сделать"setmin"/"setmax" /"setpos" - установить минимальное, максимальное или текущее значение дорожки процентов,

если задана директива "setpos\_and\_print\_through\_gap" - то одновременно с назначением позиции

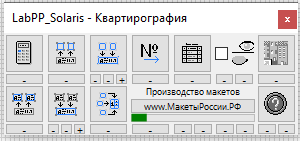
ivalue - число, в зависимости от директивы.

igap - промежуток, через который выводить счетчик в окно сообщений (например выводить через каждые 100 или 1000).

В окне LabPP\_Automat это выглядит так:



В окнах LabPP\_Solaris и других программах эта дорожка процентов располагается тоже на главной панели программы.

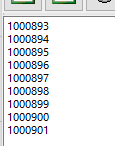


Пример 1.

Выполнить в цикле показ дорожки процентов и вывести в окно сообщений значения индекса цикла.

int main()  
{  
 int imax = 1000901;  
 runtimecontrol("workline", "setmax",imax);  
 int i;  
 **for** (i = 0; i < imax; i++)  
 {  
 cout << i << "\n";  
 runtimecontrol("workline", "setpos", i);  
 }  
 runtimecontrol("workline", "setpos", 0);  
}

Результат:

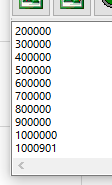


Пример 2.

Выполнить в цикле показ дорожки процентов и вывести в окно сообщений значения индекса цикла каждые 100 000 итераций и последнее значение 1000901.

int main()  
{  
 int imax = 1000901;  
 runtimecontrol("workline", "setmax",imax);  
 int i;  
 **for** (i = 0; i < imax; i++)  
 {  
 cout << i << "\n";  
 runtimecontrol("workline", "setpos\_and\_print\_through\_gap", i+1, 100000);  
 }  
 runtimecontrol("workline", "setpos", 0);  
}

Результат:



**Команды для управления в приложении LabPP\_Calc**

**interface**

**calc\_field**

Считать или записать содержимое оперативных полей и полей-примечаний калькулятора для ARCHICAD. В приложении имеется 4 поля и под ними для удобства пользователя - поля комментарии.

Вызов:

ac\_request("interface", "calc\_field", string what, string fieldname, int dochangedot, string svalue,string scomment);

Здесь:

what - выражение "set" или "get" - записать или получить,

fieldname - имя поля "a"/"b"/"c"/"main" - к какому из полей будет применена команда.

dochangedot - заменять запятую на точку при считывании оперативного поля и наоборот при записи или не заменять (1/0).

svalue - текстовое значение оперативного поля,

scomment - текстовое значение для поля-комментария к оперативному полю.

При выполнении команды "set" выполняется сохранение информации для выполнения отката или повторения операций.

Пример.

Записать в операционное поле "a" значение 111.11 автоматически заменяя точку на запятую. В комментарии к этому полю записать, что это площадь квартиры.

ac\_request("interface","calc\_field","set","a",1,"111.11","s квартиры");

get\_active

Получить имя поля калькулятора, у которого нажата кнопка "A","B","C" или "D".

Возвращает значение через ac\_getstrvalue() - "a","b","c" или "main".

Пример:

ac\_request("interface", "calc\_field", "get\_active");

string sletter = ac\_getstrvalue();

**Внешние переменные**

После выполнения программы все ее переменные уничтожаются из памяти.

Исключение составляют внешние переменные.

Их можно создавать по ходу выполнения одной программы и считывать в другой без ограничений. Это делается вызовом функций var\_extern\_get и var\_extern\_set.

Внешние переменные имеют идентификатор и значение.

**var\_extern\_set**

Задать внешнюю переменную и присвоить ей значение. Значение присваивается в текстовом виде.

Вызов:

var\_extern\_get(string varname, string varvalue);

или

var\_extern\_get(string varname, double varvalue);

Здесь:

varname - имя переменной, varvalue значение переменной текст или число.

**var\_extern\_get**

Получение значения внешней переменной из памяти по ее идентификатору

Вызов:

var\_extern\_set(string varname, string varnameforvalue);

или

var\_extern\_set(string varname, double varnameforvalue);

Здесь:

varname - имя переменной, varnameforvalue - имя переменной в которую запишется значение внешней переменной.

**var\_extern\_delete**

Удалить из памяти внешнюю переменную.

Вызов:

var\_extern\_delete( string sVarName);

Здесь:

sVarName - имя переменной, которую необходимо удалить из памяти.

Пример.

Сделать так, чтобы при закрытии окна немодального диалога удалялся из памяти объект таблицы и удалялась переменная.

int main()  
{  
#pragma region Создание диалога  
 **object**("create","ts\_dialog",iDialogDescr);  
 ts\_dialog(iDialogDescr, "init\_dialog","palette",0,0,450,400);  
 ts\_dialog(iDialogDescr, "set\_as\_main\_panel");  
 ts\_dialog(iDialogDescr, "eventreaction", "Event\_PanelCloseRequested"); // задание реакции на событие  
 ts\_dialog(iDialogDescr, "SetTitle","Выгрузка BOQ и BOM");

...

#pragma endregion   
  
 ts\_dialog(iDialogDescr, "invoke", bres);  
 }  
  
// Обработчик событий  
int Event\_PanelCloseRequested(int iDescr, string sDescr)  
{  
 int toret = 0;  
  
 int iTableReestrEdinichnyhRascenok; // реестр единичных расценок  
 int ires = 0;  
 ires = var\_extern\_get("iTableReestrEdinichnyhRascenok", iTableReestrEdinichnyhRascenok, 0);  
 **if** (ires >= 0)  
 {  
 **object**("delete", iTableReestrEdinichnyhRascenok);  
 var\_extern\_delete("iTableReestrEdinichnyhRascenok");  
 }  
 **return** toret;  
}  
  
// Создать таблицу, если она еще не создана  
int CreateTableIfNeeded()  
{  
 int ires = 0;  
 ires = var\_extern\_get("iTableReestrEdinichnyhRascenok", iTableReestrEdinichnyhRascenok, 0);  
 **if** (ires != 0)  
 {  
 //   
 **object**("create", "ts\_table", iTableReestrEdinichnyhRascenok);  
 **object**("KeepInMemory", iTableReestrEdinichnyhRascenok);  
 ...  
 }  
 **else**  
 {  
 ts\_table(iTableReestrEdinichnyhRascenok, "reset\_cpp\_to\_current");  
 }  
 **return** 0;  
}

**Объект ts\_idispatcher**

Для использования механизма взаимодействия с другими программами через механизм OLE (Windows) удобно пользоваться объектом ts\_idispatcher и одноименной функцией.

Это позволяет создавать собственные функции и манипулировать различными программами Windows. Например, если не хватает имеющихся функций в LABPP\_Automat для работы с Excel можно создавать свои собственные процедуры.

Пример 1.

Подключиться к текущей активной таблице Excel чтобы считать значение из текущей ячейки.

Полученное значение вывести в окно сообщений программы.

//------------------------------------------------------

// Example

// Read number current cell in Excell table

// LABPP 2022

//------------------------------------------------------

int main()

{

int iIDispatchExcel;

object("create","ts\_idispatch",iIDispatchExcel);

ts\_idispatch(iIDispatchExcel,"attach","Excel.Application"); // create dispatcher object

int iVariantRange;

object("create","ts\_variant",iVariantRange); // create variant object to obtain dispatcher object of the current active cell

int ires = ts\_idispatch(iIDispatchExcel, "AutoWrap", iVariantRange, "DISPATCH\_PROPERTYGET", "ActiveCell", 0);

if(ires != 0) {

return -1;

}

int iIDispatchActiveCell;

object("create","ts\_idispatch",iIDispatchActiveCell);

ts\_variant(iVariantRange,"get\_pdispVal", iIDispatchActiveCell);

int iVariantParm;

object("create","ts\_variant",iVariantParm);

// ts\_variant(iVariantParm,"set\_dblVal",111.10);

// ts\_variant(iVariantParm,"set\_bstrVal","Stop War");

ires = ts\_idispatch(iIDispatchActiveCell,"AutoWrap", iVariantParm,"DISPATCH\_PROPERTYGET", "FormulaR1C1", 0);

// ires = ts\_idispatch(iIDispatchActiveCell,"AutoWrap", iVariantParm,"DISPATCH\_PROPERTYGET", "Value", 0);

if (ires == 0)

{

string s;

ts\_variant(iVariantParm, "get\_value\_simple",s);

coutvar << s;

int i;

ts\_variant(iVariantParm, "get\_value\_simple",i);

coutvar << i;

double d;

ts\_variant(iVariantParm, "get\_value\_simple",d);

coutvar << d;

}

object("delete",iVariantParm);

object("delete",iVariantRange);

object("delete",iIDispatchActiveCell);

ts\_idispatch(iIDispatchExcel,"detach");

object("delete",iIDispatchExcel);

}

Пример 2.

Подключиться к текущей активной таблице Excel чтобы записать значение в текущую ячейку.

//------------------------------------------------------

// Example

// Write number 111.10 to current cell in Excell table

// LABPP 2022

//------------------------------------------------------

int main()

{

int iIDispatchExcel;

object("create","ts\_idispatch",iIDispatchExcel);

ts\_idispatch(iIDispatchExcel,"attach","Excel.Application"); // create dispatcher object

int iVariantRange;

object("create","ts\_variant",iVariantRange); // create variant object to obtain dispatcher object of the current active cell

int ires = ts\_idispatch(iIDispatchExcel, "AutoWrap", iVariantRange, "DISPATCH\_PROPERTYGET", "ActiveCell", 0);

if(ires != 0) {

return -1;

}

int iIDispatchActiveCell;

object("create","ts\_idispatch",iIDispatchActiveCell);

ts\_variant(iVariantRange,"get\_pdispVal", iIDispatchActiveCell);

int iVariantParm;

object("create","ts\_variant",iVariantParm);

// ts\_variant(iVariantParm,"set\_dblVal",111.10);

ts\_variant(iVariantParm,"set\_value\_simple","Stop War");

int iVariantNull;

object("create","ts\_variant",iVariantNull);

ires = ts\_idispatch(iIDispatchActiveCell,"AutoWrap", iVariantNull,"DISPATCH\_PROPERTYPUT", "FormulaR1C1", 1, iVariantParm);

object("delete",iVariantParm);

object("delete",iVariantRange);

object("delete",iIDispatchActiveCell);

object("delete",iVariantNull);

ts\_idispatch(iIDispatchExcel,"detach");

object("delete",iIDispatchExcel);

}

**CreateInstance**

Подключение к сторонней программе Windows через механизм OLE функцией CreateInstance.

Вызов:

int iret = ts\_idispatch(int iIDispatchApp,"CreateInstance",string appclassname, string runcontext);

Здесь:

iIDispatchApp - дескриптор объекта ts\_idispatch куда будет помещен создаваемый объект программы;

appclassname - имя класса объекта программы ("Excel.Application" - для Excel);

runcontext - контекст размещения объекта.

|  |  |
| --- | --- |
| Значение | Описание |
| "REMOTE\_SERVER" | Программа выполняется на сервере |
| "LOCAL\_SERVER" | Программа выполняется локально |
| "ALL" | Любое размещение |

**attach**

Подключение к сторонней программе Windows через механизм OLE функцией GetActiveObject.

В этом случае мы можем подключиться к уже открытой программе (например к текущей таблице Excel).

Вызов:

int iret = ts\_idispatch(int iIDispatchApp,"attach",string appclassname);

Здесь:

iIDispatchApp - дескриптор объекта ts\_idispatch куда будет помещен создаваемый объект программы;

appclassname - имя класса объекта программы ("Excel.Application" - для Excel);

**detach**

Отключить объект от программы.

Вызов:

ts\_idispatch(int iIDispatch,"detach");

Здесь:

iIDispatch - дескриптор объекта диспетчера программы, который нужно отключить.

**AutoWrap**

Выполнить действия с элементом программы Windows через его диспетчер.

Вызов:

int ires = ts\_idispatch(iIDispatch,"AutoWrap", int iVariantRes, string commandtypecall, string commandname, 0);  
или   
int ires = ts\_idispatch(iIDispatch,"AutoWrap", int iVariantRes, string commandtypecall, string commandname, int par\_count, int iVariantArg1, ...);

Здесь:

ires - 0 если нет ошибки,

iIDispatch - объект типа ts\_idispatch,

iVariantRes - объект типа ts\_variant в который записывается результат выполнения обращения.

commandtypecall - форма обращения

|  |  |
| --- | --- |
| Значение | Назначение |
| "DISPATCH\_PROPERTYGET" | Получить |
| "DISPATCH\_PROPERTYPUT" | Присвоить |
| "DISPATCH\_METHOD" | Выполнить |
| "DISPATCH\_PROPERTYPUTREF" | Прочее |

commandname - имя команды или переменной программы ( например "ActiveCell" в Excel и т.п.).

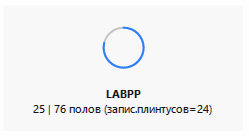
**Управление выполнением и поведением программы**

**Прерывание выполнения длительных процессов ac\_process\_window**

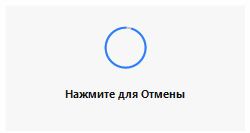
Для информирования пользователя о ходе выполняемой длительной операции и возможности ее прервать - используется специальное окно.

На следующем примере показано как вызвать такое окно, чтобы показать сколько обработано элементов полов (25), сколько осталось обработать (76), и скольким элементам плинтусов назначена тип пола (24).

Так выглядит окно процесса с информацией по его состоянию.



При наведении курсора мыши окно сообщает что если на него кликнуть - процесс будет остановлен.



bool bStop = **false**;  
ac\_process\_window("Init", "LABPP"); // иницииновать окно процесса  
string sPhase = "полов (запис.плинтусов=";  
int iAssigned = 0;  
  
**for** (i = 0; i < icount; i++)  
{  
 bStop = ac\_process\_window("SetPhase", sPhase+itoa(iAssigned)+")", icount-i);  
 **if** (bStop == **true**)  
 {  
 cout << "\nПроцесс остановлен пользователем\n";  
 ac\_process\_window("Close");  
 **return** -1;  
 }  
 ac\_request("set\_current\_element\_from\_list", 1, i);  
 ac\_request("elem\_user\_property", "set", "Тип отделки пола", "");  
}  
ac\_process\_window("Close");

**Init**

Инициировать окно процесса.

Вызов:

ac\_process\_window("Init", string sTitle);

Здесь:

sTitle - название процесса.

**SetPhase**

Поддержать показ окна процесса, обновить в нем информацию и получить обратную связь от пользователя, если процесс был им остановлен нажатием мыши на окно.

Каждый вызов этой функции увеличивает внутренний счетчик.

Вызов:

bool bStop = ac\_process\_window("SetPhase", string sPhase, int iCounter2);

Здесь:

sPhase - текст, который будет располагаться в окне под названием процесса.

iCounter2 - дополнительный параметр, который будет отображаться между внутренним счетчиком и sPhase.

Если iCounter2 = -1, то он не отображается.

Возвращает true если пользователь кликал мышкой на окне процесса с целью его остановить.

**IsCanceled**

Проверить не поступал ли запрос от пользователя остановить процесс.

Если нажималась кнопка мыши на окне процесса, то вернется true.

Вызов:

bool bStop = ac\_process\_window("IsCanceled");

**Close**

Закрыть окно процесса.

Вызов:

ac\_process\_window("Close");

**throw\_process**

Принудительная остановка процесса с сообщением и возможностью вызова панели BIM Master для анализа ситуации.

Вызов:

**void** throw\_process(string smessage);

Здесь:

smessage - текст сообщения, которое получит пользователь в момент выполнения программы.

Пример.

Остановить программу если переменная i меньше или равно 0.

int main()

{

int i=0;

if (i <= 0)

{

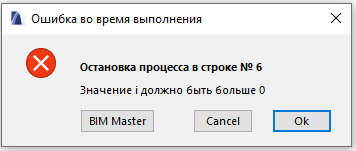
throw\_process("Значение i должно быть больше 0");

}

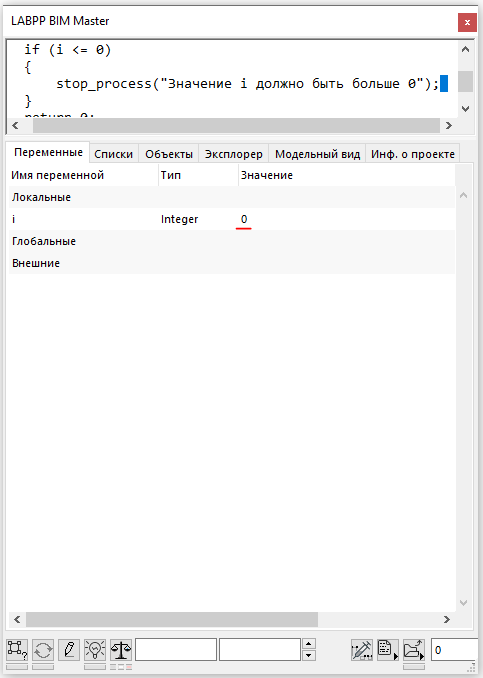
return 0;

}

В ходе выполнения программы пользователь получит такое сообщение:



Он сможет проанализировать, при необходимости, ситуацию при помощи панели BIM Master.



**Поведение программы при ошибках считывания/записи переменных элементов**

Функции чтения/записи данных в переменные элементов проекта возвращается код ошибки.

Если 0 - ошибки нет. Если < 0 , то в ходе выполнения операции возникла ошибка.

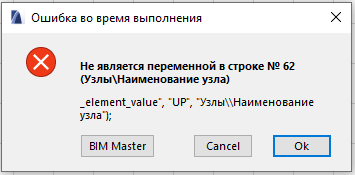
В случае, если ошибки считывания/записи являются критическими, т.е. если нельзя допустить чтобы у каких-то элементов выборки отсутствовали переменные или записались с ошибкой - можно сократить код используя соответствующие конструкции.

SETCFG("stop\_on\_err\_elem\_value\_get", bool bStopOnError);

SETCFG("stop\_on\_err\_elem\_value\_set", bool bStopOnError);

Данные конструкции влияют на текущий сеанс интерпретатора (запуск программы);

Если случится ситуация, когда переменная элемента (например пользовательское свойство или GDL параметр) будет отсутствовать или записано с ошибкой, то пользователь увидит информацию об ошибке в окне и программа предложит вызвать BIM Master для выяснений обстоятельств и устранения проблемы на месте.



**Остановка процесса помещением файла в каталоге**

Для остановки длительного сильно нагруженного процесса можно использовать простую технику контроля наличия файла с определенным именем в определенном месте диска.

Ниже приведен пример программы, которая в цикле выполняет 100 тыс. операций.

Имеется подпрограмма, которая мониторит наличие файла c:\labpp\stop.txt.

Если такой файл появился - программа его удаляет и запрашивает пользователя следует ли ей остановить процесс выполнения.

Файл может создаваться .bat файлом с размещением его, например, на рабочем столе.

В отличии от окна управления процессом, такой способ наиболее четко гарантирует остановку программы из любого положения.

int main()

{

int i = 0, j=0;

int ires;

for (i = 0; i < 100000; i++, j++)

{

if (j >= 10000)

{

j = 0;

сoutvar << i;

}

if (check\_user\_break()) {

break;

}

}

}

bool check\_user\_break()

{

string stopfilename = "c:\\labpp\\stop.txt";

bool bexist = false;

shell\_func("file", "is\_exist", stopfilename, bexist);

if (bexist)

{

shell\_func("file", "delete", stopfilename);

if (tsalert(-3, "Запрос", "Остановить выполнение?", "", "Да", "Нет") == 1)

{

return true;

}

}

return false;

}

**Объект ts\_variant**

Объекты ts\_variant предназначены для хранения данных любого типа в среде Windows.

Этот объект может содержать логическое, целое или вещественное число, строку и даже объект Windows IDispatch для общения и управления другими программами.

**get\_pdispVal**

Получить из объекта ts\_variant объект ts\_idispatch для управления программами или элементами программ через OLE Windows.

Вызов:

**int** iret = ts\_variant(**int** iVariantRange,"get\_pdispVal", **int** iIDispatchActiveCell);

Здесь:

iVariantRange - дескриптор объекта типа ts\_variant, который содержит объект управления IDispatch.

iIDispatchActiveCell - дескриптор объекта типа ts\_idispatch, в который надо записать этот элемент управления IDispatch.

Пример.

Получить внутренний элемент управления текущей ячейкой таблицы Excel.

int iIDispatchExcel;

object("create","ts\_idispatch",iIDispatchExcel);

ts\_idispatch(iIDispatchExcel,"attach","Excel.Application"); // create dispatcher object

int iVariantRange;

object("create","ts\_variant",iVariantRange); // create variant object to obtain dispatcher object of the current active cell

int ires = ts\_idispatch(iIDispatchExcel, "AutoWrap", iVariantRange, "DISPATCH\_PROPERTYGET", "ActiveCell", 0);

**get\_value\_simple**

Получить простое значение из объекта ts\_variant в переменную.

Вызов:

ts\_variant(int iVariant, "get\_value\_simple", bool/int/double/string value\_ret);

Здесь:

iVariant - дескриптор объекта типа ts\_variant;

valut\_ret - имя переменной, в которую надо переписать значение из объекта iVariant.

Пример.

Считать текстовое значение из объекта, заданного дескриптором iVariantParm и вывести в окно сообщений.

string s;  
ts\_variant(iVariantParm, "get\_value\_simple",s);  
coutvar << s;

**set\_value\_simple**

Задать простое значение в объект ts\_variant.

Вызов:

ts\_variant(int iVariant, "set\_value\_simple", bool/int/double/string value);

Здесь:

iVariant - дескриптор объекта типа ts\_variant;

value - значение, которое надо переписать в объект iVariant.

**run\_cpp - запуск другой программы**

Функция run\_cpp позволяет выполнять другие программы прямо из текста текущей программы.

Возможен запуск из файла или прямо из текстовой строки, которую можно формировать во время выполнения текущей программы.

**run\_from\_file**

Выполнение программы из заданного файла.

Форма обращения

double dret = run\_cpp("run\_from\_file",string sFileNameAndPath,int arg1,double arg2,string arg3);

Здесь:

sFileNameAndPath - имя и полный путь к программному файлу, который нужно выполнить.

arg1, arg2 и arg3 - аргументы, с которыми будет выполнена программа (в ней можно их считать функцией get\_args)

Результат возвращенный в dret соответствует числу, которое указано в команде return в программном файле, который здесь запускается.

Пример.

Выполнить программу из файла "c:\my\_program.cpp" и передать в нее целое число 100.

int myintvar = 100;

double dret = run\_cpp("run\_from\_file","c:\my\_program.cpp",myintvar,0,"");

**run\_from\_variable**

Выполнение программы из строковой переменной.

Это удобно в случае, если требуется сформировать алгоритм действий программы по ходу выполнения.

Форма обращения:

double iret = run\_cpp("run\_from\_variable",strint programtext,int arg1,double arg2,string arg3);

Здесь:

programtext - текст программы для выполнения.

arg1, arg2 и arg3 - аргументы, с которыми будет выполнена программа (в ней можно их считать функцией get\_args)

Пример.

Сформировать текст программы и выполнить ее с параметрами 100, 111.1 и "аргумент текстовый". В окне сообщений показать результат выполнения программы.

string programtext = "int main(){ cout << \"Моя программа \"<<\"\n\"; int iarg1; double darg2; string sarg3; run\_cpp(\"get\_args\",iarg1, darg2, sarg3);";

programtext += "cout << iarg1 << \" ,\" << darg2 << \" ,\" << sarg3; return -1;}";

int iret = run\_cpp("run\_from\_variable", programtext, 100, 111.1, "аргумент текстовый");

cout << "iret = " << iret << "\n";

Мы сгенерировали программу так, что по завершению она выполняет функцию return -1.

Значит iret будет содержать -1.

**Получение аргументов внутри программы**

Если программный файл запущен на выполнение с аргументами, то их можно получить функцией run\_cpp с директивой get\_args.

Обращение

run\_cpp("get\_args",int iarg1, double darg2, string sarg3);

Здесь:

iarg1, darg2, sarg3 - это переменные типа целое, вещественное и строка, куда будут записаны значения аргументов, которые были заданы при запуске программы.

**Обработка ошибок**

Во время выполнения скриптов при обнаружении ошибки выдается сообщение в диалоговом окне и работа программы останавливается. В сообщении указывается место и суть ошибки. Поэтому ее легко понять, найти и устранить.

**Коды ошибок**

В обычном случае при возникновении какой-либо ошибки при выполнении функции возвращается значение меньше 0, а если ошибки нет - то 0.

Таблица кодов ошибок

|  |  |  |
| --- | --- | --- |
| Код ошибки | Значение | Идентификатор |
|  | **Общий интерпретатор C++** |  |
| 0 | Успешное завершение операции |  |
| -1 | Общая ошибка | TSErrCommonError |
| -1100 | Логическая ошибка | TSErrLogicError |
| -1222 | Переменная отсутствует или недоступна | TSErrVarNotExist |
| -1223 | Неверный индекс | TSErrBadIndex |
| -1224 | Значение параметра недоступно | TSErrVarValueNotAvailable |
|  | **Файловые операции** |  |
| -2222 | Не найден исходный файл | TSErrIOSourceNotFound |
| -2223 | Файл уже существует | TSErrIOTagetExists |
| -2224 | Устройство переполнено | TSErrIODeviceIsFull |
| -2225 | Недостаточно прав пользователя | TSErrIOAccessDenied |
|  | **JSON** |  |
| 0 | Успешно | tsjsonerr\_no\_err |
| 2000 | Нет данных для загрузки | tojsonerr\_empty\_json |
| 2001 | Ожидалась строка | tsjsonerr\_not\_a\_string |
| 2002 | Ожидалось двоеточик | tsjsonerr\_not\_a\_colon |
| 2003 | Несоответствие типов данных | tsjsonerr\_type\_mismatch |
| 2004 | Переменная не существует | tsjsonerr\_not\_exist |

**Функции интерфейса LabPP\_Automat**

При загрузке LabPP\_Automat создает список имеющихся в его рабочем каталоге конфигураций и показывает пользователю на выбор в своем стартовом окне. Названием конфигурации является название ее каталога. В каждом каталоге конфигурации есть каталог tsimages ( для хранения картинок) и tsprg (там хранятся программные файлы).

Когда пользователь сделал выбор, то LabPP\_Automat загружает выбранную конфигурацию и выполняет программу config.cpp из нее.

В этот момент командами формирования интерфейса LabPP\_Automat можно создавать элементы интерфейса (кнопки и т.п.).

**create\_iconbutton**

Создать кнопку с картинкой.

Вызов:

ac\_request("create\_iconbutton",string sPictureName,int sx,int sy, int ex, int ey, string sToolTip, string sPrgCppFileName);

Здесь:

sPictureName - имя файла с картинкой для кнопки (из каталога tsimages).

sx,sy,ex,ey - координаты кнопки на рабочей панели LabPP\_Automat,

sToolTip - текстовая подсказка для показа в момент, когда пользователь наводит мышь на кнопку,

sPrgCppFileName - имя файла программы (из каталога tsprg), которая будет запускаться по этой кнопке.

**create\_button**

Создать кнопку с текстовой надписью.

Вызов:

ac\_request("create\_button",string sButtonText, int sx, int sy, int ex, int ey, string sToolTip, string sPrgCppFileName);

Здесь:

sButtonText - текст в кнопке.

sx,sy,ex,ey - координаты кнопки на рабочей панели LabPP\_Automat,

sToolTip - текстовая подсказка для показа в момент, когда пользователь наводит мышь на кнопку,

sPrgCppFileName - имя файла программы (из каталога tsprg), которая будет запускаться по этой кнопке.

**set\_palette\_size\_and\_message\_place**

Установить размеры диалога для рабочей панели LabPP\_Automat и размеры окна сообщений в ней.

ac\_request("set\_palette\_size\_and\_message\_place", int x, int y, int ex, int ey, int xx, int yy, int exx, int eyy);

Здесь:

x, y, ex, ey - координаты рабочей панели на экране,

xx, yy, exx, eyy - координаты окна сообщений в рабочей панели.

**Диалоги на основе окон**

Имеется возможность создания полноценных оконных диалогов.

Диалоги могут быть модальные (когда работать с проектом можно только закрыв диалог) или немодальные (при открытом окне диалога можно переключиться и делать что угодно с проектом).

**Класс ts\_dialog**

Для работы с диалогами имеется внутренний класс ts\_dialog. Чтобы создать диалог мы создаем объект сласса ts\_dialog инструкцией object с директивой "create".

object("create", "ts\_dialog", iDialogDescr);

Удаление диалога из памяти производится командой

object("delete", iDialogDescr);

Немодальные диалоги удалять ненужно, т.к. они работают после завершения функции main() на экране. Они удаляются автоматически с закрытием главного окна. Если вы создаете и запускаете несколько немодальных диалогов одновременно, то один из них надо сделать главным командой "set\_as\_main\_panel".

**init\_dialog**

Подготовить объект диалога к работе.

Вызов:

int ts\_dialog(int iDialogDescr,"init\_dialog", string modaltype, int x, int y, int width, int height);

Здесь:

smodaltype - модальный или немодальный диалог. Возможны значения "modaldialog" - модальный, "palette" - немодальный;

x, y, width, height - координаты левого верхнего угла, ширина и высота окна диалога.

**set\_as\_main\_panel**

Задает диалог как главную панель.

Если при выполнении программы создаются несколько немодальных диалогов, то все они остаются на экране до того, как будут закрыты пользователем.

И если пользователь закрывает диалог, отмеченный как главная панель - то все остальные окна закроются вместе с ним.

Вызов:

ts\_dialog(int iDialogDescr, "set\_as\_main\_panel");

**SetClientWH**

Задает ширину и высоту окна диалога.

Вызов:

ts\_dialog(int iDialogDescr, "SetClientWH", int width, int height);

Здесь:

width, height - ширина и высота окна диалога.

**SetTitle**

Задает заголовок для окна диалога.

Вызов:

**ts\_dialog(int iDialogDescr,"SetTitle", string sTitle);**

Здесь:

sTitle - заголовок для диалога.

**SendCloseRequest**

Закрыть диалог на базе палитры, т.е. немодальный диалог.

Вызов:

**ts\_dialog("iDialogDescr, "SendCloseRequest");**

**PostCloseRequest**

Закрыть модальный диалог.

Вызов:

ts\_dialog(iDialogDescr,"PostCloseRequest",string okorcancel);

Здесь:

okorcancel - строковое значение "ok" или "cancel" - означает с каким результатом закрыть диалог.

По "ok" функция invoke вернет 1, иначе 0.

**eventreaction**

Задает реакцию диалога на события, такие как изменение размера и т.п.

В тексте программы можно создавать функции с определенным названием, в котором фигурирует название события.

И связывать элементы с этим событием. Так, что будет выполняться эта функция.

Для диалогов доступны события:

"Event\_PanelCloseRequested" - вызывается перед закрытием окна диалога.

Связывание окна диалога с функцией - рекацией на событие производится командой "eventreaction".  
Вызов:

ts\_dialog(iDialogDescr, "eventreaction", string event\_name);

или

ts\_dialog(iDialogDescr, "eventreaction", string event\_name, string eventfunctionname );

Здесь:

event\_name - имя события. Если не задано имя функции - реакции на событие, то за имя функции принимается имя события.

eventfunctionname - имя функции - реакции на событие.

Если функция возвращает -1 - то панель не закрывается. Если любое другое значение - то панель закрывается.

**Invoke**

Запускает диалог на экране.

Оращение:

**int ts\_dialog(iDialogDescr,"invoke");**

Возвращает 1 или 0 если диалог был закрыт с "ok" или "cancel" соответственно.

Это работает только для модальных диалогов. При запуске немодальной панели программа выполняется дальше не ожидая закрытия окна диалога.

**Руководство по LABPP\_Automat для ARCHICAD**

Изменить имя панели диалога чтобы различать при выполнении событий панелей диалогов кто из них вызвал событие.

ts\_dialog( int iDialogDescr, string snew\_sDescr);

Здесь:

iDialogDescr - дескриптор диалога, у которого изменяем;

snew\_sDescr - новое имя для распознавания диалога (или состояния диалога) в программе.

**Класс ts\_dialogcontrol**

При помощи класса ts\_dialogcontrol выполняется работа с элементами диалогов, такими как листбоксы, поля для редактирования и т.п.

Вы можете задать реакцию на события этих элементов диалогов. Например на изменение выбора в списке, изменение текста в поле редактирования, нажатие кнопки, изменение размера и т.п.

Сначала создается панель диалога.

Затем - элементы управления. При создании элементов управления указывается панель, где они будут размещаться.

Элемент управления может находиться не непосредственно на панели диалога, а на элементе "tabpage" элемента "normaltab".

**Button**

Создает текстовую кнопку.

Вызов:

**ts\_dialogcontrol(int iButtonDescr, "init\_control", "button",int x, int y, int w, int h);**

Здесь:

x,y,w,h - позиция левого верхнего угла кнопки, ширина и высота в пикселях.

SetText

Присвоить текстовое значение элементу управления.

Вызов:

**ts\_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);**

Здесь:

text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.

Вызов:

**ts\_diaolgcontrol(iDialogcontrolDescr, "GetText", string text);**

Здесь:

text - текстовое значение, считанное из элемента управления.

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**IconButton**

Создает кнопку с картинкой.

Вызов:

**ts\_dialogcontrol(int iButtonDescr, "init\_control", "iconbutton",int x, int y, int w, int h);**

Здесь:

x,y,w,h - позиция левого верхнего угла кнопки, ширина и высота в пикселях.

**TextEdit**

Создает элемент панель редактирования текстовой строки

Вызов:

**ts\_dialogcontrol(int iButtonDescr, "init\_control", "iconbutton",int x, int y, int w, int h);**

Здесь:

x,y,w,h - позиция левого верхнего угла кнопки, ширина и высота в пикселях.

SetText

Присвоить текстовое значение элементу управления.

Вызов:

**ts\_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);**

Здесь:

text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.

Вызов:

**ts\_diaolgcontrol(iDialogcontrolDescr, "GetText", string text);**

Здесь:

text - текстовое значение, считанное из элемента управления.

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**RealEdit**

Создает элемент панель редактирования чисел

Вызов:

**ts\_dialogcontrol(int iButtonDescr, "init\_control", "realedit",int x, int y, int w, int h);**

Здесь:

x,y,w,h - позиция левого верхнего угла кнопки, ширина и высота в пикселях.

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**CheckBox**

Создает чекбокс.

Вызов:

**ts\_dialogcontrol(int iButtonDescr, "init\_control", "iconbutton",int x, int y, int w, int h);**

Здесь:

x,y,w,h - позиция левого верхнего угла кнопки, ширина и высота в пикселях.

SetText

Присвоить текстовое значение элементу управления.

Вызов:

**ts\_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);**

Здесь:

text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.

Вызов:

**ts\_diaolgcontrol(iDialogcontrolDescr, "GetText", string text);**

Здесь:

text - текстовое значение, считанное из элемента управления.

SetCheck

Задать значение элементу управления.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetCheck", int onoff);

Здесь:

iDialogcontrolDescr - дескриптор панели диалога.

onoff - 0/1.

GetCheck

Получить значение из элемента управления.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "GetCheck", int &onoff);

Здесь:

iDialogcontrolDescr - дескриптор элемента CheckBox.

onoff - возвращаемое значение - 0/1.

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**LeftText, CenterText, RightText**

Создают статический текст.

Вызов:

ts\_dialogcontrol(int iTextDescr, "init\_control", "lefttext",int x, int y, int w, int h);

ts\_dialogcontrol(int iTextDescr, "init\_control", "centertext",int x, int y, int w, int h);

ts\_dialogcontrol(int iTextDescr, "init\_control", "righttext",int x, int y, int w, int h);

Здесь:

x,y,w,h - позиция левого верхнего угла кнопки, ширина и высота в пикселях.

SetText

Присвоить текстовое значение элементу управления.

Вызов:

**ts\_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);**

Здесь:

text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.

Вызов:

**ts\_diaolgcontrol(iDialogcontrolDescr, "GetText", string text);**

Здесь:

text - текстовое значение, считанное из элемента управления.

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**PopUp**

Создает элемент для выбора из списка.

[](https://youtu.be/WssMMcTlpoo)

SelectItem

Выбрать значение из списка в элементе управления как текущее.

Вызов:

**ts\_dialogcontrol(iDialogcontrolDescr,"SelectItem", int itemindex);**

Здесь:

Itemindex = индекс выбранного значения в списке элемента управления.

DisableItem

Сделать недоступной для редактирования и выбора позицию элемента управления.

Вызов:

**ts\_dialogcontrol(iDialogControlDescr, "DisableItem", int itemindex);**

Здесь:

itemindex - позиция, которую сделать недоступной (индекс от 1).

DeleteItem

Сделать недоступной для редактирования и выбора позицию элемента управления.

Вызов:

**ts\_dialogcontrol(iDialogControlDescr, "DisableItem", int itemindex);**

Здесь:

itemindex - позиция, которую сделать недоступной (индекс от 1).

EnableItem

Сделать доступной для редактирования и выбора позицию элемента управления.

Вызов:

**ts\_dialogcontrol(iDialogControlDescr, "EnableItem", int itemindex);**

Здесь:

itemindex - позиция, которую сделать доступной (индекс от 1).

InsertItem

Вставить новый элемент в указанную позицию списка.

Вызов:

**ts\_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);**

Здесь:

itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.

Вызов:

**ts\_dialogcontrol(iDialogControlDescr, "AppendItem");**

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

GetItemIndexByText

Получить индекс по его тексту. Возвращает индекс от 1-n или 0, если такой текст в списке отсутствует.

Вызов:

int ires = ts\_dialogcontrol(int iPopUp,"GetItemIndexByText", string stext, int index);

Здесь:

iPopUp - дескриптор элемента управления;

ires - -1 если текст не найден, или 0;

index - индекс текста в списке элемента управления, или 0, если текст не найден;

stext - искомый текст в списке элемента управления, индекс которого надо получить.

GetItemIndexByText

Для popup задаем текст и получаем индекс 1-базед (если не нашли - 0)

**RadioButton**

Элемент управления - радиокнопки.

IsSelected

Проверить выбран ли этот элемент радио-кнопки или нет.

Вызов:

**ts\_dialogcontrol(iDialogcontrolDescr,"IsSelected", int ret);**

Здесь:

ret - результат запроса 0/1 - не выбран/выбран.

Select

Выбрать элемент управления типа радио-кнопки.

**ts\_dialogcontrol(iDialogcontrolDescr,"Select");**

SetText

Присвоить текстовое значение элементу управления.

Вызов:

**ts\_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);**

Здесь:

text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.

Вызов:

**ts\_diaolgcontrol(iDialogcontrolDescr, "GetText", string text);**

Здесь:

text - текстовое значение, считанное из элемента управления.

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**IconRadioButton**

IsSelected

Проверить выбран ли этот элемент радио-кнопки или нет.

Вызов:

**ts\_dialogcontrol(iDialogcontrolDescr,"IsSelected", int ret);**

Здесь:

ret - результат запроса 0/1 - не выбран/выбран.

Select

Выбрать элемент управления типа радио-кнопки.

**ts\_dialogcontrol(iDialogcontrolDescr,"Select");**

**SingleSelListBox**

### 

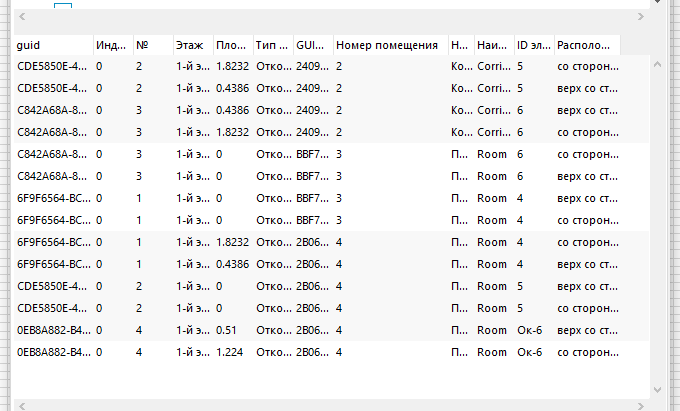
RepaintBackgroundItemsByColumnValue

Перекрасить подложки у строк в табличном диалоговом элементе в соответствии со значением колонки.

Пример.

В элементе листбокс, заданном дескриптором iListBoxElements выкрасить подложки строк полосами по ходу значений в колонке 8. Цвета - белый/светло-серый.

ts\_dialogcontrol(iListBoxElements, "RepaintBackgroundItemsByColumnValue", 8, 255, 255, 255, 248, 248, 248);



В результате помещения визуально выделены полосками.

SetItemBackgroundColor

Установить цвет подложки текста для заданной строки в табличном элементе диалога.

ts\_dialogcontrol(int iListBox, "SetItemBackgroundColor", int row, int red, int green, int blue);

Здесь:

iListBox - дескриптор табличного элемента диалога,

row - номер строки,

red, green, blue - значения от 0 до 255 для составляющий цветов красного, зеленого и синего.

SetItemColor

Установить цвет текста для заданной строки в табличном элементе диалога.

ts\_dialogcontrol(int iListBox, "SetItemColor", int row, int red, int green, int blue);

Здесь:

iListBox - дескриптор табличного элемента диалога,

row - номер строки,

red, green, blue - значения от 0 до 255 для составляющий цветов красного, зеленого и синего.

SelectItem

Выбрать значение из списка в элементе управления как текущее.

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr,"SelectItem", int itemindex);

Здесь:

Itemindex = индекс выбранного значения в списке элемента управления.

GetTabItemText

Получить значение ячейки из таблицы ListBox.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "GetTabItemText", int item, int tabpos, string restext);

Здесь:

item - строка таблицы (индекс от 1), tabpos - колонка таблицы (индекс от 1), restext - возвращаемое значение текста из этой ячейки.

GetMouseClickedPosXY

Получить координаты позиции мыши в окне ListBox в момент отработки события Event\_ListBoxClicked.

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr,"GetMouseClickedPosXY", int &x, int &y);

Здесь:

x,y - координаты мыши, когда был сделан щелчок, вызвавший событие.

GetTabFieldPosition

Получить позицию колонки от левого края окна элемента управления.

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr," GetTabFieldPosition ", int tabpos, int &begx, int &endx);

Здесь:

tabpos - номер колонки (индекс от 1),

begx, endx - начало и конец в пикселях от левого края окна элемента управления (не путать с левым краем самого диалога).

SetTabItemIcon

Задать картинку для ячейки в таблице элемента управления.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "SetTabItemIcon", int item, int tabpos, string path);

Здесь:

item - строка таблицы (индекс от 1), tabpos - колонка таблицы (индекс от 1), path - полный путь к файлу картинки

SetTabItemText

Задать текст в ячейку элемента управления.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "SetTabItemText", int item, int tabpos, string text);

Здесь:

item - строка таблицы (индекс от 1), tabpos - колонка таблицы (индекс от 1), text - новое значение текста из этой ячейки.

GetTabItemIconId

Получить номер картинки из ячейки таблицы элемента управления.

В программе есть набор картинок, которые можно показывать задавая число.

Это удобно для создания на месте элементов типа CheckBox.

Командой SetTableItemIconId можно задавать такую картинку. А в функции - обработчике события по клику мыши - изменять картинку с пометкой на пустой квадрат или наоборот.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "GetTabItemIconId", int item, int tabpos, int iconid);

Здесь:

item - строка таблицы (индекс от 1), tabpos - колонка таблицы (индекс от 1), iconid - номер картинки.

SetTabItemIconId

Задать изображение в ячейке таблицы элемента управления по коду.

Вызов:

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "GetTabItemIconId", int item, int tabpos, int iconid);

Здесь:

item - строка таблицы (индекс от 1), tabpos - колонка таблицы (индекс от 1), iconid - номер картинки.

SetOnTabItem

Выбрать элемент диалога, который будет отображаться в заданном поле таблицы в списке. Используется, когда необходимо задать редактирование данных непосредственно в ячейке.

Вызов:

ts\_dialogcontrol(iListBoxDescr, "SetOnTabItem", int iConrtolDescr);

Здесь:

iControlDescr - дескриптор элемента управления, который будет использоваться для показа и обработки данных в ячейке.

SetTabFieldCount

Задать количество колонок в таблице в элементе управления.

Вызов:

ts\_dialogcontrol(iListBoxDescr, "SetOnTabItem", int count);

Здесь:

count - количество колонок.

SetHeaderItemSize

Задать высоту строки заголовков в таблице элемента управления.

Вызов:

ts\_dialogcontrol(iListBoxDescr, "SetHeaderItemSize", int height);

Здесь:

height - высота строки заголовков в точках.

SetTabFieldProperties

Задать параметры колонки для строк данных в таблице элемента управления.

Вызов:

ts\_dialogcontrol(iListBox, "SetTabFieldProperties", int col, int startpix, int endpix, int justification, int truncation);

Здесь:

col - колонка таблицы (индекс от 1),

startpix, endpix - начало и конец колонки в точках,

justification - выравнивание текста (влево 0, центр - 256, вправо - 512),

truncation - обрезка текста, если он не поместился на экран (обычно 4096).

SetItemHeight

Задать высоту строки в таблице значений.

Вызов:

ts\_dialogcontrol(iListBox, "SetItemHeight", int itemheight);

Здесь:

itemheight - высота строки.

SetHeaderItemSizeableFlag

Задать параметры изменения размера и управляемости мышью для колонки таблицы

Вызов:

ts\_dialogcontrol(iListBox, "SetHeaderItemSizeableFlag", int headerindex, int sizable, int minwidth);

Здесь:

headerindex - колонка (индекс от 1),

sizable - изменяемость 0/1 мышью,

minwidth - минимальная ширина колонки

SetHeadersSizeableFlag

Задать параметры изменения размера и управляемости мышью для всех колонок таблицы

Вызов:

ts\_dialogcontrol(iListBox, "SetHeaderItemSizeableFlag", int sizable, int minwidth);

Здесь:

sizable - изменяемость 0/1 мышью,

minwidth - минимальная ширина колонки

SetHeaderItemText

Задать текст заголовка для колонки таблицы.

Вызов:

ts\_dialogcontrol(iListBox, "SetHeaderItemText", int headerindex, string text);

Здесь:

headerindex - колонка (индекс от 1),

text - название колонки.

SetHeaderItemSize

Задать ширину заголовка для таблицы элемента управления.

Вызов:

ts\_dialogcontrol(iListBox, "SetHeaderItemSize", int headerindex, int width);

Здесь:

headerindex - колонка (индекс от 1),

width - ширина заголовка колонки.

InsertItem

Вставить новый элемент в указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);

Здесь:

itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "AppendItem");

DisableItem

Сделать недоступной для редактирования и выбора позицию элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "DisableItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать недоступной (индекс от 1)

EnableItem

Сделать доступной для редактирования и выбора позицию элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "EnableItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать доступной (индекс от 1)

DeleteItem

Удалить указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "DeleteItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать доступной (индекс от 1). Если задан 0, то удаляются все позиции.

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**MultiSellListBox**

SelectItem

Выбрать значение из списка в элементе управления как текущее.

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr,"SelectItem", int itemindex);

Здесь:

Itemindex = индекс выбранного значения в списке элемента управления.

GetTabItemText

Получить значение ячейки из таблицы ListBox.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "GetTabItemText", int item, int tabpos, string restext);

Здесь:

item - строка таблицы (индекс от 1), tabpos - колонка таблицы (индекс от 1), restext - возвращаемое значение текста из этой ячейки.

GetMouseClickedPosXY

Получить координаты позиции мыши в окне ListBox в момент отработки события Event\_ListBoxClicked.

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr,"GetMouseClickedPosXY", int &x, int &y);

Здесь:

x,y - координаты мыши, когда был сделан щелчок, вызвавший событие.

GetTabFieldPosition

Получить позицию колонки от левого края окна элемента управления.

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr," GetTabFieldPosition ", int tabpos, int &begx, int &endx);

Здесь:

tabpos - номер колонки (индекс от 1),

begx, endx - начало и конец в пикселях от левого края окна элемента управления (не путать с левым краем самого диалога).

SetTabItemIcon

Задать картинку для ячейки в таблице элемента управления.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "SetTabItemIcon", int item, int tabpos, string path);

Здесь:

item - строка таблицы (индекс от 1), tabpos - колонка таблицы (индекс от 1), path - полный путь к файлу картинки

SetTabItemText

Задать текст в ячейку элемента управления.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "SetTabItemText", int item, int tabpos, string text);

Здесь:

item - строка таблицы (индекс от 1), tabpos - колонка таблицы (индекс от 1), text - новое значение текста из этой ячейки.

GetTabItemIconId

Получить номер картинки из ячейки таблицы элемента управления.

В программе есть набор картинок, которые можно показывать задавая число.

Это удобно для создания на месте элементов типа CheckBox.

Командой SetTableItemIconId можно задавать такую картинку. А в функции - обработчике события по клику мыши - изменять картинку с пометкой на пустой квадрат или наоборот.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "GetTabItemIconId", int item, int tabpos, int iconid);

item - строка таблицы (индекс от 1), tabpos - колонка таблицы (индекс от 1), iconid - номер картинки.

SetTabItemIconId

Задать изображение в ячейке таблицы элемента управления по коду.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "GetTabItemIconId", int item, int tabpos, int iconid);

Здесь:

item - строка таблицы (индекс от 1), tabpos - колонка таблицы (индекс от 1), iconid - номер картинки.

SetOnTabItem

Выбрать элемент диалога, который будет отображаться в заданном поле таблицы в списке. Используется, когда необходимо задать редактирование данных непосредственно в ячейке.

Вызов:

ts\_dialogcontrol(iListBoxDescr, "SetOnTabItem", int iConrtolDescr);

Здесь:

iControlDescr - дескриптор элемента управления, который будет использоваться для показа и обработки данных в ячейке.

SetTabFieldCount

Задать количество колонок в таблице в элементе управления.

Вызов:

ts\_dialogcontrol(iListBoxDescr, "SetOnTabItem", int count);

Здесь:

count - количество колонок.

SetTabFieldCount

Задать количество колонок в таблице в элементе управления.

Вызов:

ts\_dialogcontrol(iListBoxDescr, "SetOnTabItem", int count);

Здесь:

count - количество колонок.

SetHeaderItemSize

Задать высоту строки заголовков в таблице элемента управления.

Вызов:

ts\_dialogcontrol(iListBoxDescr, "SetHeaderItemSize", int height);

Здесь:

height - высота строки заголовков в точках.

SetTabFieldProperties

Задать параметры колонки для строк данных в таблице элемента управления.

Вызов:

ts\_dialogcontrol(iListBox, "SetTabFieldProperties", int col, int startpix, int endpix, int justification, int truncation);

Здесь:

col - колонка таблицы (индекс от 1),

startpix, endpix - начало и конец колонки в точках,

justification - выравнивание текста (влево 0, центр - 256, вправо - 512),

truncation - обрезка текста, если он не поместился на экран (обычно 4096).

SetItemHeight

Задать высоту строки в таблице значений.

Вызов:

ts\_dialogcontrol(iListBox, "SetItemHeight", int itemheight);

Здесь:

itemheight - высота строки.

SetHeaderItemSizeableFlag

Задать параметры изменения размера и управляемости мышью для колонки таблицы

Вызов:

ts\_dialogcontrol(iListBox, "SetHeaderItemSizeableFlag", int headerindex, int sizable, int minwidth);

Здесь:

headerindex - колонка (индекс от 1),

sizable - изменяемость 0/1 мышью,

minwidth - минимальная ширина колонки

SetHeadersSizeableFlag

Задать параметры изменения размера и управляемости мышью для всех колонок таблицы

Вызов:

ts\_dialogcontrol(iListBox, "SetHeaderItemSizeableFlag", int sizable, int minwidth);

Здесь:

sizable - изменяемость 0/1 мышью,

minwidth - минимальная ширина колонки

SetHeaderItemText

Задать текст заголовка для колонки таблицы.

Вызов:

ts\_dialogcontrol(iListBox, "SetHeaderItemText", int headerindex, string text);

Здесь:

headerindex - колонка (индекс от 1),

text - название колонки.

SetHeaderItemSize

Задать ширину заголовка для таблицы элемента управления.

Вызов:

ts\_dialogcontrol(iListBox, "SetHeaderItemSize", int headerindex, int width);

Здесь:

headerindex - колонка (индекс от 1),

width - ширина заголовка колонки.

InsertItem

Вставить новый элемент в указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);

Здесь:

itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "AppendItem");

DisableItem

Сделать недоступной для редактирования и выбора позицию элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "DisableItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать недоступной (индекс от 1)

EnableItem

Сделать доступной для редактирования и выбора позицию элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "EnableItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать доступной (индекс от 1)

DeleteItem

Удалить указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "DeleteItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать доступной (индекс от 1). Если задан 0, то удаляются все позиции.

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**MultiLineEdit**



SetText

Присвоить текстовое значение элементу управления.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);

Здесь:

text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "GetText", string text);

Здесь:

text - текстовое значение, считанное из элемента управления.

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**IconItem**

**IconCheckBox**

SetCheck

Задать значение элементу управления.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetCheck", int onoff);

Здесь:

iDialogcontrolDescr - дескриптор панели диалога.

onoff - 0/1.

GetCheck

Получить значение из элемента управления.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "GetCheck", int &onoff);

Здесь:

iDialogcontrolDescr - дескриптор элемента CheckBox.

onoff - возвращаемое значение - 0/1.

**PushCheck**

SetText

Присвоить текстовое значение элементу управления.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);

Здесь:

text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.

Вызов:

ts\_diaolgcontrol(iDialogcontrolDescr, "GetText", string text);

Здесь:

text - текстовое значение, считанное из элемента управления.

SetCheck

Задать значение элементу управления.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetCheck", int onoff);

Здесь:

iDialogcontrolDescr - дескриптор панели диалога.

onoff - 0/1.

GetCheck

Получить значение из элемента управления.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "GetCheck", int &onoff);

Здесь:

iDialogcontrolDescr - дескриптор элемента CheckBox.

onoff - возвращаемое значение - 0/1.

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**IconPushCheck**

SetCheck

Задать значение элементу управления.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetCheck", int onoff);

Здесь:

iDialogcontrolDescr - дескриптор панели диалога.

onoff - 0/1.

GetCheck

Получить значение из элемента управления.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "GetCheck", int &onoff);

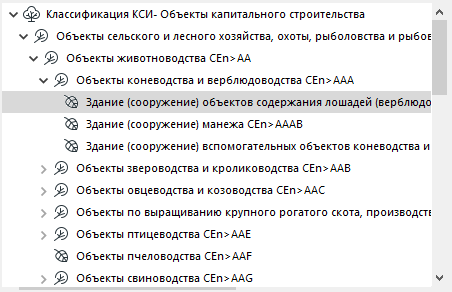
Здесь:

iDialogcontrolDescr - дескриптор элемента CheckBox.

onoff - возвращаемое значение - 0/1.

**IconPushRadio**

**SingleSelTreeView**



SelectItem

Выбрать значение из списка в элементе управления как текущее.

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr,"SelectItem", int itemindex);

Здесь:

Itemindex = индекс выбранного значения в списке элемента управления.

TreeViewInsertItem

Добавить пункт в элемент управления TreeView

Вызов:

ts\_dialogcontrol(int iDescr,"TreeViewInsertItem",int parent, int tvitem, int &resitem);

Здесь:

parent - номер родительского пункта,

tvitem - тип вставки в список RootItem (0), BotItem (-65534), TopItem (-65535).

resitem - возвращенный номер вставленного пункта.

DeleteItem

Удалить указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "DeleteItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать доступной (индекс от 1). Если задан 0, то удаляются все позиции.

InsertItem

Добавить позицию в список элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "InsertItem");

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

GetTVItemIconId

Получить идентификатор иконки у позиции элемента TreeView.

Вызов:

ts\_dialogcontrol(int iTreeViewElement, "GetTVItemIconId", int tvindex, int iwhat, int iconid);

Здесь:

iTreeViewElement - дескриптор элемента TreeView;

tvindex - индекс позиции элемента TreeView;

iwhat - 0/1 - иконка статуса/иконка обычная;

iconid - идентификатор иконки в системе.

SetTVItemIconId

Задать идентификатор иконки у позиции элемента TreeView.

Вызов:

ts\_dialogcontrol(int iTreeViewElement, "SetTVItemIconId", int tvindex, int iwhat, int iconid);

Здесь:

iTreeViewElement - дескриптор элемента TreeView;

tvindex - индекс позиции элемента TreeView;

iwhat - 0/1 - иконка статуса/иконка обычная;

iconid - идентификатор иконки в системе.

GetItemCount

Получить количество элементов.

Вызов:

ts\_dialogcontrol(int iTreeView, "GetItemCount", int icount);

Здесь:

iTreeView - дескриптор элемента управления TreeView;

icount - результат - количество элементов в TreeView.

Пример.

Задать иконку с ID 32000 (квадратик без "галочки") для всех элементов в элементе TreeView.

int i,icount;

ts\_dialogcontrol(iTreeView\_DCClassSel, "GetItemCount", icount);

coutvar << icount;

for (i = 1; i <= icount; i++)

{

ts\_dialogcontrol(iTreeView\_DCClassSel, "SetTVItemIconId", i, 1, 32000);

}

import\_classificator

Импортировать классификатор в элемент TreeView.

Вызов:

int err = ts\_dialogcontrol(int iTreeView, "import\_classificator", string classificator\_name, int idoobnul, int iTableClassGuids);

Здесь:

iTreeView - дескриптор элемента TreeView, куда грузим классификатор;

classificator\_name - имя загружаемого классификатора;

idoobnul - 0/1 не обнулять/обнулять таблицу и элемент TreeView перед загрузкой;

iTableClassGuids - таблица guid-ов загруженных классификаторов;

Позиция таблицы записывается в поле Value позиции в элементе TreeView.

Ее можно получить из TreeView через функцию "getitemvalue".

Структура таблицы формируется автоматически.

|  |  |  |
| --- | --- | --- |
| Наименование колонки | Тип | Назначение |
| itemguid | string | GUID класса.  У позиции классификатора это поле пустое. |
| level | int | Уровень позиции в TreeView (0-корневая позиция, соответствует имени классификатора) |

В TreeView одновременно может быть загружено сколько угодно классифиакторов.

Пример.

Загрузить несколько классификаторов в элемент TreeView с дескриптором iTreeView\_DCClassSel.

ts\_dialogcontrol(iTreeView\_DCClassSel, "import\_classificator", "Классификация ЛАБПП", 0, iTableClassifGuids);

ts\_dialogcontrol(iTreeView\_DCClassSel, "import\_classificator", "ЛАБПП-ДВЕРИ", 0, iTableClassifGuids);

ts\_dialogcontrol(iTreeView\_DCClassSel, "import\_classificator", "ЛАБПП-ОКНА", 0, iTableClassifGuids);

**MultiSelTreeView**

SelectItem

Выбрать значение из списка в элементе управления как текущее.

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr,"SelectItem", int itemindex);

Здесь:

Itemindex = индекс выбранного значения в списке элемента управления.

TreeViewInsertItem

Добавить пункт в элемент управления TreeView

Вызов:

ts\_dialogcontrol(int iDescr,"TreeViewInsertItem",int parent, int tvitem, int &resitem);

Здесь:

parent - номер родительского пункта,

tvitem - тип вставки в список RootItem (0), BotItem (-65534), TopItem (-65535).

resitem - возвращенный номер вставленного пункта.

DeleteItem

Удалить указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "DeleteItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать доступной (индекс от 1). Если задан 0, то удаляются все позиции.

InsertItem

Вставить новый элемент в указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);

Здесь:

itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "AppendItem");

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

GetTVItemIconId

Получить идентификатор иконки у позиции элемента TreeView.

Вызов:

ts\_dialogcontrol(int iTreeViewElement, "GetTVItemIconId", int tvindex, int iwhat, int iconid);

Здесь:

iTreeViewElement - дескриптор элемента TreeView;

tvindex - индекс позиции элемента TreeView;

iwhat - 0/1 - иконка статуса/иконка обычная;

iconid - идентификатор иконки в системе.

SetTVItemIconId

Задать идентификатор иконки у позиции элемента TreeView.

Вызов:

ts\_dialogcontrol(int iTreeViewElement, "SetTVItemIconId", int tvindex, int iwhat, int iconid);

Здесь:

iTreeViewElement - дескриптор элемента TreeView;

tvindex - индекс позиции элемента TreeView;

iwhat - 0/1 - иконка статуса/иконка обычная;

iconid - идентификатор иконки в системе.

GetItemCount

**SingleSelListView**

DeleteItem

Удалить указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "DeleteItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать доступной (индекс от 1). Если задан 0, то удаляются все позиции.

InsertItem

Вставить новый элемент в указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);

Здесь:

itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "AppendItem");

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

GetItemCount

**MultiSelListView**

DeleteItem

Удалить указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "DeleteItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать доступной (индекс от 1). Если задан 0, то удаляются все позиции.

InsertItem

Вставить новый элемент в указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);

Здесь:

itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "AppendItem");

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

GetItemCount

**Splitter**

Attach\_controlled\_element

Подключить к сплиттеру контролируемый элемент (который будет изменять свою форму или положение в зависимости от перемещения сплиттера).

Вызов:

ts\_dialogcontrol(int iSplitter, "attach\_element\_to\_resize", int iControlDescr, int leftanchor, int topanchor, int rightanchor, int bottomanchor);

Здесь:

iControlDescr - дестриптор подключаемого элемента управления,

leftanchor, topanchor, rightanchor, bottomanchor - -1/0/1 - привязки к элементу управления. 1 - будет изменяться в плюс со смещением сплиттера вправо/вниз, 0 - не будет перемещаться никуда, -1 при смещении вправо/вниз - будет изменяться в обратную сторону.

**ProgressBar**

Элемент - дорожка процентов выполнения.



SetMin

Задать минимальное значение.

Вызов:

ts\_dialogcontrol(iProgressBar, "SetMin", int minvalue);

Здесь:

minvalue - минимальное значение, от которого считать процент выполнения.

SetMax

Задать максимальное значение.

Вызов:

ts\_dialogcontrol(iProgressBar, "SetMax", int maxvalue);

Здесь:

maxvalue - максимальное значение, до которого считать процент выполнения.

SetValue

GetMin

Считать текущeе значение параметра элемента.

Вызов:

ts\_dialogcontrol(iProgressBar, "GetMin", int &value);

Здесь:

value - считанное значение.

GetMax

Считать текущeе значение параметра элемента.

Вызов:

ts\_dialogcontrol(iProgressBar, "GetMax", int &value);

Здесь:

value - считанное значение.

GetValue

Считать текущeе значение параметра элемента.

Вызов:

ts\_dialogcontrol(iProgressBar, "GetValue", int &value);

Здесь:

value - считанное значение.

**NormalTab**

SelectItem

Выбрать значение из списка в элементе управления как текущее.

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr,"SelectItem", int itemindex);

Здесь:

Itemindex = индекс выбранного значения в списке элемента управления.

DisableItem

Сделать недоступной для редактирования и выбора позицию элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "DisableItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать недоступной (индекс от 1)

EnableItem

Сделать доступной для редактирования и выбора позицию элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "EnableItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать доступной (индекс от 1).

DeleteItem

Удалить указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "DeleteItem", int itemindex);

Здесь:

itemindex - позиция, которую сделать доступной (индекс от 1). Если задан 0, то удаляются все позиции.

InsertItem

Вставить новый элемент в указанную позицию списка.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);

Здесь:

itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.

Вызов:

ts\_dialogcontrol(iDialogControlDescr, "AppendItem");

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**TabPage**

SetToolTip

Задать текст, всплывающий текст, который будет показываться при помещении указателя мышки над элементом управления.

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr, "SetToolTip", string tooltiptext);

Здесь:

tooltiptext - строка всплывающего текста с пояснением.

**GroupBox**

SetFontSizecodeStyle

Задать размер и стиль шрифта.

Размер шрифта задается номером - не изменять/большой/маленький/самый маленький -1/0/4/8.

Стиль шрифта тоже задается номером - не изменять/обычный/болд/италик/подчеркнутый -1/0/16/32/48.

Коды стиля могут суммироваться. Италик с подчеркиванием = 32+48.

Вызов:

ts\_diaolgcontrol(int iDialogcontrolDescr, "SetFontSizecodeStyle", int iFontSizeCode, int iFontStyleCode);

Здесь:

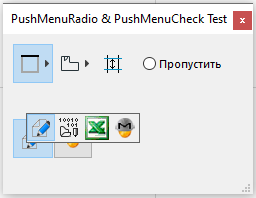
iDialogcontrolDescr - дескриптор элемента управления у которого надо поменять размер и/или стиль шрифта.

iFontSizeCode - код размера шрифта,

iFontStyleCode - код стиля шрифта.

**PushMenuCheck**





[Видеоролик](https://youtube.com/shorts/AMGVvfdBkqI)

[Файл](https://www.labpp.net/TSDownload/LabPP_Automat/LABPP%20Example%20-%20PushMenuRadio%20And%20PushMenuCheck.cpp)

//---------------------------------------------------------

// PushMenuRadio & PushMenuCheck example

// LABPP 2023

//---------------------------------------------------------

int iMenu1, iMenu11, iIconPushRadio, iMenu2, iIconPushCheck;

int main()

{

int ires;

ac\_request("clearmessagewindow");

// Create Dialog

var\_by\_txt("init", "iDialog\_DS", "int", "global", 0);

int x, y, w, h, offset;

object("create", "ts\_dialog", iDialog\_DS);

x = 0; y = 0; w = 250; h = 160;

ts\_dialog(iDialog\_DS, "init\_dialog", "palette", x, y, w, h);

ts\_dialog(iDialog\_DS, "eventreaction", "Event\_PanelCloseRequested", "Event\_PanelCloseRequested\_DS");

ts\_dialog(iDialog\_DS, "set\_as\_main\_panel");

ts\_dialog(iDialog\_DS, "SetGrowType", 3);

ts\_dialog(iDialog\_DS, "SetTitle", "PushMenuRadio & PushMenuCheck Test");

x = 10; w = 40; y = 10; h = 40;

object("create", "ts\_dialogcontrol", iMenu1);

ts\_dialogcontrol(iMenu1, "init\_control", "PushMenuRadio", iDialog\_DS, x, y, w, h, 1);

ts\_dialogcontrol(iMenu1, "SetItemCount", 3);

string iconpath = "c:\\";

ts\_dialogcontrol(iMenu1, "SetItemIcon", 1, iconpath + "files\\outlineonleft.png");

ts\_dialogcontrol(iMenu1, "settooltip", "Задание линии ячейки:");

ts\_dialogcontrol(iMenu1, "SetToolTipItem", 1, "Левый край");

ts\_dialogcontrol(iMenu1, "SetItemIcon", 2, iconpath + "files\\outlineontop.png");

ts\_dialogcontrol(iMenu1, "SetToolTipItem", 2, "Верхний край");

ts\_dialogcontrol(iMenu1, "SetItemIcon", 3, iconpath + "files\\outlineonright.png");

ts\_dialogcontrol(iMenu1, "SetToolTipItem", 3, "Правый край");

ts\_dialogcontrol(iMenu1, "eventreaction", "Event\_PushMenuChanged","Event\_PushMenuChanged\_DCName\_DS");

ts\_dialogcontrol(iMenu1, "SetAnchorToPanelResize", 0, 1, 0, 0);

x = 50; w = 40; y = 10; h = 40;

object("create", "ts\_dialogcontrol", iMenu11);

ts\_dialogcontrol(iMenu11, "init\_control", "PushMenuRadio", iDialog\_DS, x, y, w, h, 1);

ts\_dialogcontrol(iMenu11, "SetItemCount", 3);

ts\_dialogcontrol(iMenu11, "settooltip", "Выборка по цвету пола:");

ts\_dialogcontrol(iMenu11, "SetItemIcon", 1, iconpath + "files\\floor.png");

ts\_dialogcontrol(iMenu11, "SetToolTipItem", 1, "Светлый пол");

ts\_dialogcontrol(iMenu11, "SetItemIcon", 2, iconpath + "files\\floors.png");

ts\_dialogcontrol(iMenu11, "SetToolTipItem", 2, "Средние тона");

ts\_dialogcontrol(iMenu11, "SetItemIcon", 3, iconpath + "files\\floors\_group.png");

ts\_dialogcontrol(iMenu11, "SetToolTipItem", 3, "Темный пол");

ts\_dialogcontrol(iMenu11, "SetAnchorToPanelResize", 0, 1, 0, 0);

x = 90; w = 40; y = 10; h = 40;

object("create", "ts\_dialogcontrol", iIconPushRadio);

ts\_dialogcontrol(iIconPushRadio, "init\_control", "IconPushRadio", iDialog\_DS, x, y, w, h, 1);

ts\_dialogcontrol(iIconPushRadio, "settooltip", "Задать высоту ячейки");

ts\_dialogcontrol(iIconPushRadio, "SetIcon", iconpath + "files\\cell\_height.png");

ts\_dialogcontrol(iIconPushRadio, "SetAnchorToPanelResize", 0, 1, 0, 0);

x = 140; y = 18; w = 100; h = 24;

var\_by\_txt("init", "iRadioButton\_DCName\_DS", "int", "global", 0);

object("create", "ts\_dialogcontrol", iRadioButton\_DCName\_DS);

int igroup = 1;

ts\_dialogcontrol(iRadioButton\_DCName\_DS, "init\_control", "RadioButton", iDialog\_DS, x, y, w, h, 1);

ts\_dialogcontrol(iRadioButton\_DCName\_DS, "settooltip", "Пропустить выполнение команды");

ts\_dialogcontrol(iRadioButton\_DCName\_DS, "SetText", "Пропустить");

ts\_dialogcontrol(iRadioButton\_DCName\_DS, "Select");

x = 10; w = 40; y = 85; h = 40;

object("create", "ts\_dialogcontrol", iMenu2);

ts\_dialogcontrol(iMenu2, "init\_control", "PushMenuCheck", iDialog\_DS, x, y, w, h);

ts\_dialogcontrol(iMenu2, "SetItemCount", 4);

ts\_dialogcontrol(iMenu2, "settooltip", "Показать отчет в:");

ts\_dialogcontrol(iMenu2, "SetItemIcon", 1, iconpath + "files\\edit.png");

ts\_dialogcontrol(iMenu2, "SetToolTipItem", 1, "Редактор текста");

ts\_dialogcontrol(iMenu2, "SetItemIcon", 2, iconpath + "files\\EditPRGitem.png");

ts\_dialogcontrol(iMenu2, "SetToolTipItem", 2, "Редактор программ");

ts\_dialogcontrol(iMenu2, "SetItemIcon", 3, iconpath + "files\\пути\_excel.png");

ts\_dialogcontrol(iMenu2, "SetToolTipItem", 3, "Excel");

ts\_dialogcontrol(iMenu2, "SetItemIcon", 4, iconpath + "files\\LABPP\_BIM\_Master.png");

ts\_dialogcontrol(iMenu2, "SetToolTipItem", 4, "BIM Master");

ts\_dialogcontrol(iMenu2, "SetAnchorToPanelResize", 0, 1, 0, 0);

ts\_dialogcontrol(iMenu2, "eventreaction", "Event\_PushMenuChanged", "Event\_PushMenuChanged\_DCName\_DS");

x = 50; w = 40; y = 84; h = 41;

object("create", "ts\_dialogcontrol", iIconPushCheck);

ts\_dialogcontrol(iIconPushCheck, "init\_control", "IconPushCheck", iDialog\_DS, x, y, w, h);

ts\_dialogcontrol(iIconPushCheck, "settooltip", "Показать отчет в BIM Master");

ts\_dialogcontrol(iIconPushCheck, "SetIcon", iconpath + "files\\LABPP\_BIM\_Master.png");

ts\_dialogcontrol(iIconPushCheck, "SetAnchorToPanelResize", 0, 1, 0, 0);

bool bres;

bres = ts\_dialog(iDialog\_DS, "invoke", bres);

return 0;

}

int Event\_PanelCloseRequested\_DS(int iDescr, string sDescr)

{

int ires = tsalert(-3, "Message", "Are you sure you want to close the panel?", "", "Ok", "Cancel");

if (ires == 1)

{

return 0; // Allow panel to close;

}

else

{

return -1; // No close panel

}

}

// Events

int Event\_PushMenuChanged\_DCName\_DS(int iDescr, string sDescr)

{

if (sDescr == "iMenu1"){

bool bSelected;

ts\_dialogcontrol(iMenu1, "IsSelected", bSelected);

coutvar << bSelected;

bool bSelectionChanged;

ts\_dialogcontrol(iMenu1, "GetNamedEventValue","WasSelectionChange", bSelectionChanged);

coutvar << bSelectionChanged;

int iPreviousMenuItem;

ts\_dialogcontrol(iMenu1, "GetNamedEventValue","PreviousMenuItem", iPreviousMenuItem);

coutvar << iPreviousMenuItem;

}

if (sDescr == "iMenu2") {

bool bChecked;

ts\_dialogcontrol(iMenu2, "GetCheck", bChecked);

coutvar << bChecked;

bool bStateChanged;

ts\_dialogcontrol(iMenu2, "GetNamedEventValue", "WasStateChange", bStateChanged);

coutvar << bStateChanged;

int iPreviousMenuItem;

ts\_dialogcontrol(iMenu2, "GetNamedEventValue", "PreviousMenuItem", iPreviousMenuItem);

coutvar << iPreviousMenuItem;

}

}

SetItemIcon

Задать картинку для заданного пункта меню

Вызов:

ts\_dialogcontrol(int iPushMenu, "SetItemIcon", int index, string sIconFilePath);

GetCheck

Проверить нажата или отжата кнопка элемента PushMenu

Вызов:

ts\_dialogcontrol(int iPushMenuCheck, "GetCheck", bool bChecked);

SetCheck

Задать состояние "нажат/отжат" элементу PushMenuCheck.

Вызов:

ts\_dialogcontrol(int iPushMenu, "SetCheck", bool bChecked);

SetItemCount

Задать количество пунктов меню.

Вызов:

ts\_dialogcontrol(int iPushMenu, "SetItemCount", int icount);

SelectItem

Выбрать пункт меню по индексу (индекс - от 1 до n).

Вызов:

ts\_dialogcontrol(int iPusMenuCheck, "SelectItem", int index);

GetSelectedItem

Получить индекс текущего выбранного пункта меню.

Вызов:

ts\_dialogcontrol(int iPushMenu, "GetSelectedItem", int iSelectedItem);

SetToolTip

Задать общую всплывающую подсказку для элемента PushMenu.

Вызов:

ts\_dialogcontrol(int iMenu1, "settooltip", string sText);

SetToolTipItem

Задать дополнение к всплывающей подсказке когда выбран пункт меню.

Вызов:

ts\_dialogcontrol( int iPushMenu, "SetToolTipItem", int index, string sText);

Здесь:

iPushMenu - дескриптор элемента PushMenu,

index - индекс пункта меню, для которого задается дополнительный всплывающий текст (индекс от 1 до n).

sText - дополнительный текст к всплывающему сообщению.

WasStateChange

Узнать было ли изменение состояния меню во время вызова события.

Вызов:

ts\_dialogcontrol( int iPushMenu, "GetNamedEventValue", "WasStateChange", bool bStateChanged);

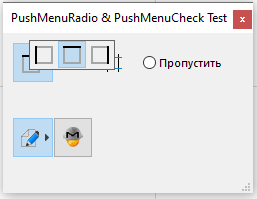
PreviousMenuItem

Какой индекс меню был выбранным до того, как выбран новый пункт меню.

ts\_dialogcontrol( int iPushMenu, "GetNamedEventValue", "PreviousMenuItem", int iPreviousMenuItem);

**PushMenuRadio**





[См. пример здесь](#_topic_PushMenuCheck)

SetItemIcon

Задать картинку для заданного пункта меню

Вызов:

ts\_dialogcontrol(int iPushMenu, "SetItemIcon", int index, string sIconFilePath);

Select

IsSelected

SetItemCount

Задать количество пунктов меню.

Вызов:

ts\_dialogcontrol(int iPushMenu, "SetItemCount", int icount);

SelectItem

Выбрать пункт меню по индексу (индекс - от 1 до n).

Вызов:

ts\_dialogcontrol(int iPusMenu, "SelectItem", int index);

GetSelectedItem

Получить индекс текущего выбранного пункта меню.

Вызов:

ts\_dialogcontrol(int iPushMenu, "GetSelectedItem", int iSelectedItem);

SetToolTip

Задать общую всплывающую подсказку для элемента PushMenu.

Вызов:

ts\_dialogcontrol(int iMenu1, "settooltip", string sText);

SetToolTipItem

Задать дополнение к всплывающей подсказке когда выбран пункт меню.

Вызов:

ts\_dialogcontrol( int iPushMenu, "SetToolTipItem", int index, string sText);

Здесь:

iPushMenu - дескриптор элемента PushMenu,

index - индекс пункта меню, для которого задается дополнительный всплывающий текст (индекс от 1 до n).

sText - дополнительный текст к всплывающему сообщению.

PreviousMenuItem

Какой индекс меню был выбранным до того, как выбран новый пункт меню.

ts\_dialogcontrol( int iPushMenu, "GetNamedEventValue", "PreviousMenuItem", int iPreviousMenuItem);

WasSelectionChange

Узнать было ли изменение состояния меню во время вызова события.

Вызов:

ts\_dialogcontrol( int iPushMenu, "GetNamedEventValue", "WasStateChange", bool bStateChanged);

**Eventreaction**

Привязать к элементу управления функцию - обработчик события или задать имя конечной функции для обработки события.

ts\_dialogcontrol(int iButtonDescr, "eventreaction", string eventfunctionname [, string eventfunctionname\_more]);

Здесь:

eventfunctionname - имя функции, которая обрабатывает событие.

eventfunctionname\_more - имя конечной функции, которая будет вызываться при событии eventfunctionname (не обязательно, просто бывает удобно объединить реакцию на различные события в различных элементах в одну функцию).

Пример. Чтобы обработать реакцию на нажатие кнопки создайте функцию:

int iDialogDescr, iButtonZoom;  
int main()  
{  
...// создать окно диалога  
 int x, y, w, h;  
 x=1,y=1, w=200; h = 200;  
 **object**("create","ts\_dialog",iDialogDescr);  
 ts\_dialog(iDialogDescr, "init\_dialog","palette",x,y,w,h);  
 ts\_dialog(iDialogDescr, "eventreaction", "Event\_PanelCloseRequested");  
 ts\_dialog(iDialogDescr, "SetTitle","Расчет ведомости объемов работ");  
 bool bres;  
 // создать элемент управления  
 x=1; y=1; w=50; h=20;  
 **object**("create","ts\_dialogcontrol",iButtonZoom,"iButtonZoom");  
 ts\_dialogcontrol(iButtonZoom, "init\_control", "button",iDialogDescr, x, y, w, h);  
 ts\_dialogcontrol(iButtonZoom, "eventreaction", "Event\_ButtonClicked");  
 ts\_dialogcontrol(iButtonZoom, "settext", "Показать");  
...  
 ts\_dialog(iDialogDescr, "invoke",bres);  
 cout << bres;  
}  
  
int Event\_ButtonClicked(int iDescr, string sDescr)  
{  
 **if**(sDescr == "ButtonCancel")  
 {  
 ts\_dialog(iDialogDescr,"PostCloseRequest","cancel"); // закрыть диалог с результатом отмены  
 }  
 **else** **if**(sDescr == "ButtonOK")  
 {  
 ts\_dialog(iDialogDescr,"PostCloseRequest","ok"); // закрыть диалог с результатом "Ok"  
 }  
 **else** **if**(sDescr == "ButtonCalc")  
 {  
 Calc();  
 }  
 **else** **if**(sDescr == "ButtonZoom")  
 {  
 ZoomElementInProject();  
 }  
}

Теперь по нажатию кнопки будет выполняться эта функция.

В момент ее выполнения iDescr - числовое значение дескриптора кнопки, которая вызвала событие, а sDescr - ее имя объекта (не путать с текстом кнопки). Можно различать какая кнопка нажата по конструкции

**if**(iDescr == iDescrZoom)   
{  
 //  
}  
Либо   
**if**(sDescr == "iButtonZoom")  
{  
 //  
}

Бывает удобно и по числу и по имени. Например можно задать одинаковые имена группе кнопок, выполнять общие действия, а затем различать их по числовому дескриптору.

Ниже дана таблица имен функций - обработчиков событий для каждого типа элементов управления диалогов.

|  |  |  |
| --- | --- | --- |
| Имя функции - обработчика события | Элементы, к которым можно применять | Примечание |
| Event\_TreeViewContextMenuRequested | MultiselTreeView, SingleSelTreeView | Запрос вызова контекстного меню |
| Event\_TreeViewItemCollapsed | Свертывание дерева |
| Event\_TreeViewItemDoubleClicked | Двойной щелчок |
| Event\_TreeViewItemExpanded | Раскрытие элемента |
| Event\_TreeViewLabelEditFinished | Завершение редактирования текста элемента |
| Event\_TreeViewLabelEditStarted | Начало редактирования текста элемента |
| Event\_TreeViewSelectionChanged | Смена выбранного элемента дерева |
| Event\_TreeViewStateIconClicked | Щелчок на иконке состояния |
|  |  |  |
| Event\_ListViewContextMenuRequested | MultiselListView, SingleselListView | Запрос контекстного меню |
| Event\_ListViewDoubleClicked | Двойной щелчок |
| Event\_ListViewItemUpdate | Обновление позиции |
| Event\_ListViewSelectionChanged | Изменение текущей позиции |
|  |  |  |
| Event\_RealEditChanged | RealEdit | Изменение числа в поле редактирования |
|  |  |  |
| Event\_TextEditChanged | TextEdit | Изменение текста в поле редактирования |
|  |  |  |
| Event\_ItemFocusGained | Все фокусируемые элементы | Элемент получил фокус |
| Event\_ItemFocusLost | Все фокусируемые элементы | Элемент потерял фокус |
|  |  |  |
| Event\_ImageClicked | IconItem | Щелчок на картинке |
|  |  |  |
| Event\_ListBoxClicked | MultiselListBox, SingleselListBox | Щелчок мыши |
| Event\_ListBoxDoubleClicked | Двойной щелчок |
| Event\_ListBoxSelectionChanged | Изменение текущего выбора |
|  |  |  |
| Event\_PopUpChanged | PopUp | Изменение значения в элементе PopUp |
|  |  |  |
| Event\_StaticTextClicked | LeftText, CenterText, RightText | Щелчок на тексте |
| Event\_StaticTextDoubleClicked | Двойной щелчок |
|  |  |  |
| Event\_CheckItemChanged | CheckBox, IconCheckBox, IconPushCheck, PushCheck | Смена выбранного элемента |
| Event\_CheckItemDoubleClicked | Двойной щелчок |
|  |  |  |
| Event\_ButtonClicked | Button, IconButton | Нажатие кнопки |
|  |  |  |
| Event\_NormalTabChanged | NormalTab | Изменение текущего элемента в NormalTab (переход на другую панель) |
| Event\_NormalTabClicked | NormalTab | Щелчек мыши на поле элемента NormalTab |
| Event\_PushMenuChanged | PushMenuCheck  PushMenuRadio | Изменение состояния |
| Event\_PushMenuDoubleClicked | Двойной щелчек мыши |
|  |  |  |
|  |  |  |

**SetAnchorToPanelResize**

Установить зависимость размеров и положения элемента от размеров панели, на которой он установлен.

Если пользователь будет изменять размеры окна диалога, то элемент может оставаться на месте, сдвигаться влево-вправо, вверх-вниз, расширяться и т.д.

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr, "SetAnchorToPanelResize", int leftanchor, int topanchor, int rightanchor, int bottomanchor);

Здесь:

leftanchor, topanchor, rightanchor, bottomanchor - числовые значения 0/1 якорей левого, верхнего, правого и нижнего края, соответственно.

При значении 0 - край остается на месте. При значении 1 - край сдвигается в соответствии с изменением размеров окна.

**Hide**

Скрыть элемент управления

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr, "Hide");

**Show**

Показать скрытый элемент управления

Вызов:

ts\_dialogcontrol(iDialogcontrolDescr, "Show");

**SetPosition**

Переместить элемент управления в новую позицию.

ts\_dialogcontrol(iDialogcontrolDescr, "SetPosition", int x, int y);

Здесь:

x,y - новые координаты левого верхнего угла перемещаемого элемента управленияю.

**Работа со структурами данных в формате JSON**

**Класс ts\_json**

**init (clear)**

Очистить содержимое объекта ts\_json.

Вызов:

ts\_json(int iJSONObjectDescr,"clear");

Здесь:

iJSONObjectDescr - дескриптор объекта JSON, который нужно очистить.

**DecodeFromURL**

Загрузить JSON код из файла, расположенного на web сайте.

Вызов:

int ires = ts\_json(int iJSONObjectDescr, "DecodeFromURL", string sURL);

Здесь:

iJSONObjectDescr - дескриптор объекта типа JSON,

sURL - адрес файла в Интернет.

Возвращает 0 при успешной загрузке.

Пример.

Загрузить JSON код из файла config.json с сайта www.appcarrier.com

int iJSONConfig; // дескриптор объекта

object("create", "ts\_json", iJSONConfig); // создать объект в памяти

int ires = ts\_json(iJSONConfig, "DecodeFromURL", "http://www.appcarrier.com/config.json");

if (ires != 0)

{

cout << "JSON Decode from URL error";

}

else

{

cout << "File JSON decoded successfully\n";

}

object("delete", iJSONConfig); // удалить объект

**DecodeFromString**

Загрузить JSON код из строковой переменной.

Вызов:

int ires = ts\_json(int iJSONObjectDescr, "DecodeFromString", string sJSON);

Здесь:

iJSONObjectDescr - дескриптор объекта типа JSON, в который надо загрузить код,

sJSON - строковая переменная, в которой находится текст JSON.

Возвращает 0 при успешной загрузке.

Пример.

Загрузить JSON код из строки.

int iJSONConfig; // дескриптор для объекта ts\_json

object("create", "ts\_json", iJSONConfig); // создать объект в памяти

string sJSON = "{ \"My class in JSON code\":{\

\"StringVariable\":\"my string\",\

\"DoubleVariable\" : 1.23\

}}";

int ires = ts\_json(iJSONConfig, "DecodeFromString", sJSON);

if (ires != 0)

{

cout << "JSON Decode from URL error";

}

else

{

cout << "File JSON decoded successfully\n";

}

object("delete", iJSONConfig); // удалить объект

**EncodeToStr**

Выгрузить JSON код в строковую переменную.

Вызов:

int ires = ts\_json(int iJSONObjectDescr, "EncodeToStr", string &sJSON);

Здесь:

iJSONObjectDescr - дескриптор объекта типа JSON, из которого надо вызагрузить код,

sJSON - строковая переменная, в которую надо выгрузить JSON объект.

Возвращает 0 при успешной выгрузки.

Пример.

Выгрузить JSON код в строку.

string sJSON;

int ires = ts\_json(iJSONConfig, "EncodeToStr", sJSON);

coutvar << sJSON; // вывести содержимое переменной sJSON в окно сообщений

**Класс ts\_array**

**Класс ts\_class**

**Класс ts\_num**

**Класс ts\_bool**

**Класс ts\_string**

**Руководство по LABPP\_Automat для ARCHICAD**

int iStringDescr;

object("create", "ts\_string", iStringDescr);

int iret = ts\_string(iStringDescr, "ReadFromURL", "http://www.labpp.ru/TSDownload/KSI/" + sclassifname + ".txt");

object("read\_simple\_value", iStringDescr);

string sXMLsource = ac\_getstrvalue();

**TSVariator**

Механизм чтобы хранить и получать значения различных свойств из одного текста.

Например в электронной таблице чтобы не плодить множество колонок под каждое свойство можно записать все их в ячейку одной колонки.

При этом количество свойст может быть любое.

Обращение для решения вариатора.

int ires = ts\_json(int iJSON, "tsvariator", "apply", int what\_answere\_if\_notexist);

Здесь:

ires - результат команды, если 0 - значит без ошибок и значение искомого свойства задано.

В ином случае - либо ошибка, либо искомого свойства нет в тексте.

what\_answere\_if\_notexist - что вернуть при отсутствии запрашиваемого свойства. Если 0 - то вернуть содержимое переменной запроса. Если 1 - то вернуть значение из свойства "По умолчанию".

Пример записи текста в формате TSVariator:

{ "TSVariator":

[

{"Result":"Нет",

"SourceVarValue" : "По умолчанию",

"Comment" : "По умолчанию, если нет параметра"

},

{

"SourceVarValue": "Ширина профиля",

"Result" : "50",

"Comment" : ""

},

{

"SourceVarValue": "Толщина утеплителя",

"Result" : "50",

"Comment" : ""

},

{

"SourceVarValue": "Огнестойкость",

"Result" : "Нет",

"Comment" : "для отметки огнестойкости пп у элемента"

},

{

"SourceVarValue": "Влагостойкость",

"Result" : "Нет",

"Comment" : "для отметки влагостойкости пп у элемента"

}],

"SourceVarName":"Свойство"

}

В квадратных скобках заключен массив классов переменных.

Каждая переменная имеет имя ("SourceVarValue"), примечание ("Comment") и значение ("Result").

"SourceVarName" - имя внешней переменной в коде скрипта, которая используется для получения данных.

В примере имя этой переменной - "Свойство";

Весь текст можно поместить в одну ячейку электронной таблицы и получать из нее значения различных свойств по их названиям ("Ширина профиля", "Толщина профиля" и т.д.). Если запрашиваемого свойства в тексте нет, то выдается значение из первой записи (в примере это значение - "Нет").

Пример.

Получить значение свойства "Огнестойкость" из текста в примере выше.

string sexample; // переменная в которой находится текст из примера выше

var\_extern\_set("Свойство", "Огнестойкость"); // записать во внешнюю переменную с именем "Свойство" имя свойства, которое хотим считать из текста.

int iJSON;

object("create", "ts\_json", iJSON); // создадим объект типа ts\_gson

string sresult;

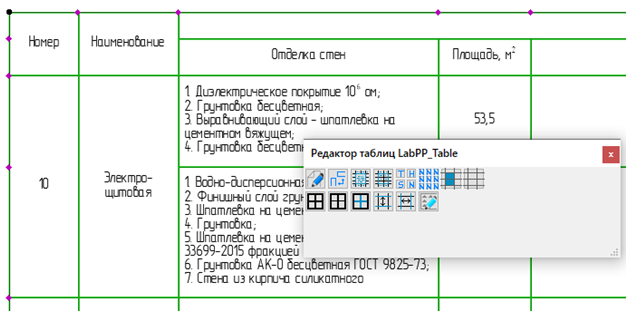
ts\_json(iJSON, "DecodeFromString", sexample);

ires = ts\_json(iJSON, "tsvariator", "apply", 1); // если не найдена переменная - вернуть значение по умолчанию sresult = ac\_getstrvalue();

сout << "Значение огнестойкости = " << sretuls << "\n";

object("delete",iJSON); // удалить объект

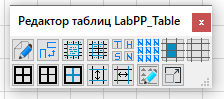
**Редактор таблиц LABPP\_Table для ARCHICAD**



Для создания в проекте таблиц можно использовать удобный GDL элемент [LABPP\_Table.gsm](http://www.labpp.ru/TSDownload/LABPP_Table.gsm).

Управление этим элементом удобно осуществлять через редактор LABPP\_Table\_unicode.cpp ([LABPP\_Table\_unicode.zip](http://www.labpp.ru/TSDownload/LabPP_Table_unicode.zip)).

Этот редактор можно модифицировать самостоятельно. Добавлять собственные кнопки и процедуры и т.п.



Основные возможности.

Максимальное число колонок - 100;

Максимальное число строк - 500;

Объединение ячеек - да;

Типов линий для отрисовки ячеек - 2 (перо для тонких линий, перо для толстых линий, без линии);

Регулировка размеров ячеек - да;

Групповое изменение высоты - да;

Групповое изменение ширины - да;

Число стилей в ячейках - 4:

Параметры для настройки стиля - шрифт, размер шрифта, отступ от границ ячейки, наклонный, жирный, коэффициент межстрочного интервала;

Перо для текста - общее для всех стилей;

Использование знаков UNICODE - да (не м2 а м²);

Выравнивание в ячейках - по горизонтали и вертикали (9 вариантов);

Расположение текста по вертикали - да;

Размещение в ячейках эскизов из других элементов - да (настройка масштаба, положения);

Описание работы редактора.

Запустить программу LabPP\_Table\_unicode.cpp.

Появится плавающее окно с кнопками управления редактором таблиц.

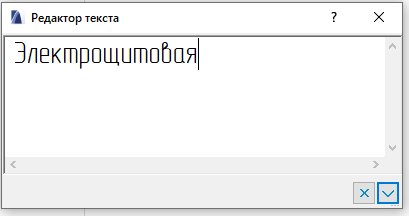
Все действия производятся при выделенном элементе таблицы.

Кнопки слева направо верхний ряд.

**Редактирование текста в ячейке**.

Щелкнуть на ячейке.

В появившемся окне отредактировать текст. Нажать кнопку записи.

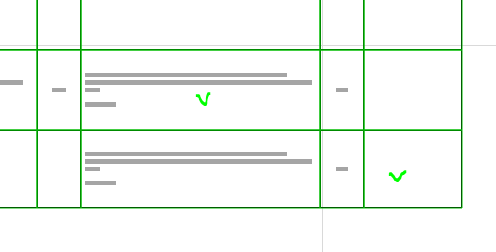


**Поворот текста в ячейке.**

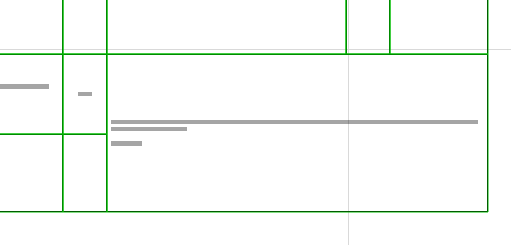
Щелкнуть на ячейке - текст в ней повернется в другое положение.

**Объединение ячеек**.

После нажатия кнопки щелкните на левый верхний угол воображаемого прямоугольника будущего объединения ячеек, затем - на правый нижний угол этого воображаемого прямоугольника.



Ячейки объединятся.



**Разъединение ячеек.**

Щелкните на объединенную ячейку.

Ячейки, ее составляющие, разъединятся.

Задать стиль в ячейке.

Нажать одну из кнопок - T, H, S, N - название, заголовок, подзаголовок и обычный текст.

Соответствующий стиль задается в настройках элемента таблицы.

**Задать отрисовку линий пером для толстых линий.**

**Задать отрисовку линий пером для тонких линий.**

**Выключить отрисовку линий.**

Выполняются одинаково - щелкнуть на воображаемый верхний левый угол и на воображаемый нижний правый угол.

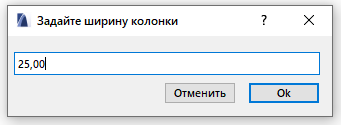
В рамках воображаемого прямоугольника линии будут заданы в соответствии с командой.

Задать высоту строки.

Задать ширину колонки.

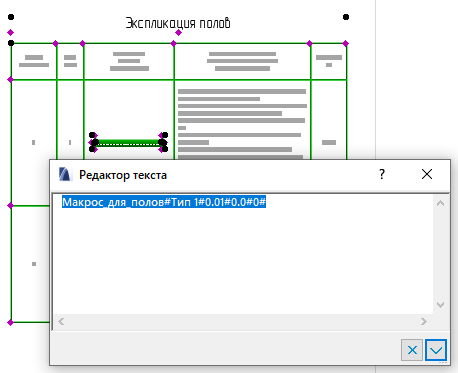
Выполняются одинаково - щелкнуть на воображаемый верхний левый угол и на воображаемый нижний правый угол.

В окне задайте соответствующее желаемое значение (можно предварительно измерить в проекте).



В рамках воображаемого прямоугольника колонки или строки будут заданы в соответствии с командой.

**Задать в ячейке отображение эскиза из другого элемента**.



В появившемся окне задать параметры отображаемого фрагмента.

Формат:

Имя макроса#Название фрагмента в макросе#Масштаб#Смещение от центра ячейки по горизонтали#Смещение от центра ячейки по вертикали#Текстовое поле для передачи дополнительных параметров

Для использования макроса в тексте 2d скрипта GDL необходимо разместить фрагменты с указанием названия и ответа для указания исходных размеров.

Пример фрагмента для отображения двух штриховок.

При отображении в таблице штриховки так же масштабируются.

За масштаб отвечает переменная TSScaleK.

Таких фрагментов может быть сколько угодно.

!----------------------------------------------------------

IF LABPP\_PictureName = "Тип 1" THEN

IF LABPP\_Command = "GET SIZES" THEN

! x,y,w,h

EXIT 0.951, -16.940, 3.9937, 0.751

ENDIF

fillAttribute\_1 = 214

fillAttribute\_2 = 330

lineTypeAttribute\_1 = 1

lineTypeAttribute\_2 = 0

pen penAttribute\_1

fill fillAttribute\_1

poly2\_b{5} 5, 2, 1, 8, penAttribute\_1, penAttribute\_2,

2.947696588428, -16.52173742916, TSScaleK, 0, 0, TSScaleK, 0,

1.09620462548, -16.47923742916, 33,

1.09620462548, -16.56423742916, 33,

4.799188551377, -16.56423742916, 33,

4.799188551377, -16.47923742916, 33,

1.09620462548, -16.47923742916, 33

set line\_type lineTypeAttribute\_1

pen penAttribute\_4

poly2\_b{5} 7, 1, 0, 3, 1, 0,

0, 0, TSScaleK, 0, 0, TSScaleK, 0,

1.22479041471, -16.51429212963, 1,

1.202131508217, -16.51429212963, 900,

0, 180, 4001,

1.179472601724, -16.51429212963, 1,

1.202131508217, -16.51429212963, 900,

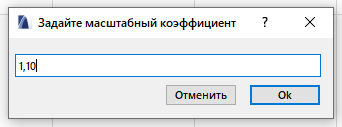
0, 180, 4001,

1.22479041471, -16.51429212963, 1

ENDIF

**Масштабирование таблицы**.

При нажатии кнопки в появившемся диалоге введите требуемый коэффициент увеличения/уменьшения ( меньше 1 - уменьшение, больше 1 - увеличение).



**Структура данных объекта LABPP\_Table (для программистов)**

Изменяя значения можно управлять таблицей и заполнять данными из собственных программных модулей

Как это делается можно посмотреть в файле редактора таблиц LABPP\_Table\_unicode.cpp.

**TS\_Cells\_poscode** - коды содержимого в ячейках таблицы.

Массив текстовый 500х100

Код показывает информацию о содержимом ячейки.

Первые 4 символа отвечают за наличие раздела ячейки с каждой из 4-х сторон.

LTRB - означает, что все границы есть.

Если стоит \_ на какой-то позиции, значит ячейка объединена по этой стороне.

У смежной ячейки должен такой же символ присутствовать по стороне сопряжений.

Например, если две ячейки с индексами [2][1] и [3][1] (в 1-й колонке в строке 2 и 3) объединены, то у верхней будет:

LTR\_ а у нижней L\_RB.

Позиция 5 отвечает за направление текста.

Пробел - горизонтально.

'a' - текст првернут на 90 градусов.

Позиция 6 отвечает за стиль текста в ячейке.

't', 'h', 's', 'n' - стиль названия таблицы, заголовка, подзаголовка или обычный текст.

7-я позиция - код формата 0-текст, 1 - целое, 2 - число с 1 знаком, 3 - число с 2 знаками.

8-я позиция - l/m/f/r выравнивание по горизонтали влево/по центру/вправо/по ширине

9-я позиция - t/m/b - выравнивание по вертикали вверх/посредине/вниз

Позиции 10,11,12,13 - тип линий в ячейке - лев/верх/прав/низ -> 0/1/2 - нет/жирная/тонкая

по умолчанию

LTRB n0mm1111

**TS\_Cells\_value** - значения в ячейках.

Массив текстовый 500х100

**TS\_Cells\_macro** - коды для отрисовки макросов в ячейках таблицы

Массив текстовый 500х100

**TS\_ColumnsUsedCount** - число колонок в таблице.

**TS\_RowsUsedCount** - число строк в таблице.

**TS\_RowsH** - массив высот колонок.

**TS\_ColumnsW** - массив ширин колонок.

**TS\_ShowSelection** - показывать или нет выбор ячеек.

После изменения данных необходимо присвоить переменной **TS\_DoRecalcAll** значение 1.

При этом элемент таблицы пересчитает служебные переменные.

Из следующих служебных переменных можно получить реальную картину элемента LABPP\_Table на текущий момент.

**TS\_RealCells** - реальная рассчитанная картина всех ячеек

Массив типа линейная длина N x 17.

Таблица плоская, значения для ячеек идут друг за другом слева направа сверху вниз.

Т.е. первые 17 элементов - первая строка таблицы, вторые 17 элементов - вторая строка таблицы и т.д. по числу задействованных ячеек.

Назначение позиций

1 - SX

2 - SY

3 - EX

4 - EY

5 - I - номер колонки

6 - J - номер строки

7 - стартовая позиция текста

8 - конечная позиция текста

9 - левая граница нет/да

10 - верхняя граница нет/да

11 - правая граница нет/да

12 - нижняя граница нет/да

13 - кол-во позиций вниз совмещено

14 - кол-во позиций вправо совмещено

15 - самый левый нет/да

16 - самый верхний нет/да

17 - selected

**TS\_RealCellsSelected** - массив выбранных ячеек.

Рассчитывается по 17-й позиции из TS\_RealCells

Чтобы задать выбранные ячейки - надо установить 17-ую позицию у соответствующих элементов ячеек в поле TS\_RealCells.

Имейте в виду, что после перерасчета таблицы информация о выбранных ячейках сбрасывается как в электронных таблицах.

**Непереносимые пробелы и дефисы**

Обычно требуется, чтобы шифр ГОСТ переносился на следующую строку не отдельными фрагментами, а целиком.

Для этого используют непереносимые пробелы и дефисы.

Это такие же знаки, только у них другие коды.

LABPP\_Table поддерживает эту технологию.

Поэтому там где надо чтобы фраза с пробелами и дефисами переносилась целиком, так же как в текстовых редакторах можно ставить:

На клавиатуре - удерживая Alt наберите 0160 для ввода непереносимого пробела.

Чтобы получить непереносимый дефис - Alt+8211.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Знак | ⁰ | ¹ | ² | ³ | ⁴ | ⁵ | ⁶ | ⁷ | ⁸ | ⁹ |
| Число для ввода Alt+ | 8304 | 0185 | 0178 | 0179 | 8308 | 8309 | 8310 | 8311 | 8312 | 8313 |
| Unicode | 2070 | 00B9 | 00B2 | 00B3 | 2074 | 2075 | 2076 | 2077 | 2078 | 2079 |



**Работа с СУБД MySQL**



Имеется возможность подключать базы данных MySQL непосредственно из текста программ LABPP.

Скачайте с сайта MySQL следующие файлы:

1. libmysql.dll
2. libcrypto-3-x64.dll
3. libssl-3-x64.dll

и поместите их в каталог ARCHICAD.

**connection\_init**

Инициализация объекта для присоединения к БД MySQL из ARCHICAD  
Вызов:  
int ires = ts\_dbmysql(int iMySQL, "connection\_init");  
Здесь:  
ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения инициализации.  
iMySQL - дескриптор объекта подключения.  
  
Пример:  
  
int iMySQL;  
**object**("create", "ts\_dbmysql", iMySQL);  
int ires;  
ires = ts\_dbmysql(iMySQL, "connection\_init");  
**if** (ires != 0) {  
 **return** -1;  
}  
cout << "init success\n";

**connection\_open**

Открыть коннект с базой данных (присоедититься к БД).

Вызов:

int ires = ts\_dbmysql(iMySQL, "connection\_open", string server, string user, string password, string basename, int port);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iMySQL - дескриптор объекта подключения;

server - имя или IP сервера, где находится подключаемая база данных;

user - имя пользователя

password - пароль

basename - имя базы данных (в MySQL это называется схемой)

port - порт для подключения (обычно 3306);

Пример:

string server = "31.41.196.220";  
string user = "u0490570";  
string password = "hP2uH9fdsaY2zpA5eA4";  
string basename = "u0490570";  
int port = 3306;  
ires = ts\_dbmysql(iMySQL, "connection\_open", server, user, password, basename, port);  
**if** (ires != 0) {  
 **return** -1;  
}  
cout << "connection success\n";

**real\_query**

Выполнить запрос к базе данных.

Запрос может выполняться с использованием символов unicode (например м²).

Результат запроса далее может быть получен, например, в виде таблицы ts\_table.

Вызов:

int ires = ts\_dbmysql(iMySQL, "real\_query", string query);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iMySQL - дескриптор объекта подключения;

query - текст запроса к базе данных на языке SQL (например SELECT \* FROM u0490570.my\_table);

Пример:

string query = "SELECT \* FROM u0490570.my\_table"; // запросить все записи из таблицы my\_table в базе данных u0490570  
int j = 0;  
ires = ts\_dbmysql(iMySQL, "real\_query", query);  
**if** (ires != 0) {  
 **return** -1;  
}  
cout << "query success\n";

**store\_result**

Сохранить результат запроса.

Результат запроса далее может быть получен, например, в виде таблицы ts\_table.

Вызов:

int ires = ts\_dbmysql(iMySQL, "store\_result");

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iMySQL - дескриптор объекта подключения;

Пример:

ires = ts\_dbmysql(iMySQL, " ires = ts\_dbmysql(iMySQL, "store\_result");

if (ires != 0) {

return -1;

}

cout << "store result success\n";");

**get\_num\_fields**

Получить количество колонок в результате запроса.

Вызов:

int ires = ts\_dbmysql(iMySQL, "get\_num\_fields");

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iMySQL - дескриптор объекта подключения;

Пример:

ires = ts\_dbmysql(iMySQL, "get\_num\_fields");  
cout << "num fields = " << ires << "\n";

**get\_row\_count**

Получить количество строк в результате запроса.

Вызов:

int ires = ts\_dbmysql(iMySQL, "get\_row\_count", count);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iMySQL - дескриптор объекта подключения;

count - возвращает количество строк в полученной таблице.

Пример:

int count;

ires = ts\_dbmysql(iMySQL, "get\_row\_count",count);

cout << "row count = " << count << "\n";

**export\_to\_table**

Выгрузить результат, полученный по запросу из базы данных, в таблицу ts\_table.

Вызов:

int ires = ts\_dbmysql(iMySQL, "export\_to\_table", int iTable);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iMySQL - дескриптор объекта подключения;

iTable - дескриптор объекта таблицы ts\_table, в которую требуется выгрузить результат.

Пример:

int iTable;

object("create", "ts\_table", iTable);

ts\_dbmysql(iMySQL, "export\_to\_table", iTable);

**free\_result**

Освободить память от полученного результата.

Когда результат запроса к БД больше не нужен или требуется выполнить новый запрос - освобождаем память.

Вызов:

int ires = ts\_dbmysql(iMySQL, "free\_result");

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iMySQL - дескриптор объекта подключения;

Пример:

// освободить память результата запроса в подключении у объекта iMySQL  
ts\_dbmysql(iMySQL, "free\_result");

**connection\_close**

Закрыть соединение в объекте подключения, заданном дескриптором.

После закрытия соединения объект можно использовать повторно или для подключения к другим БД.

Вызов:

int ires = ts\_dbmysql(iMySQL, "connection\_close");

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iMySQL - дескриптор объекта подключения;

Пример:

ts\_dbmysql(iMySQL, "connection\_close");

**Пример - получение таблицы c сайта из базы данных MySQL в диалог ARCHICAD**

Сформируем диалог, который покажет таблицу из большой таблицы из базы данных MySQL.

Параметры подключения:

Сервер IP - 31.41.196.220 (или можно указать имя сервера)

Имя пользователя - u0490570

Пароль пользователя - hP2uH9fdsaY2zpA5eA4

Имя базы данных (схемы) - u0490570

Номер порта для подключения (обычно) - 3306;



int iListBox\_MD, iButton\_Ok\_MD;

int iTable\_MD;

int iDialog\_MD;

int main()

{

int iMySQL;

object("create", "ts\_dbmysql", iMySQL);

int ires;

ires = ts\_dbmysql(iMySQL, "connection\_init");

if (ires != 0) {

return -1;

}

cout << "init success\n";

string server = "31.41.196.220";

string user = "u0490570";

string password = "hP2uH9fdsaY2zpA5eA4";

string basename = "u0490570"; // scheme

int port = 3306;

ires = ts\_dbmysql(iMySQL, "connection\_open", server, user, password, basename, port);

if (ires != 0) {

return -1;

}

cout << "connection success\n";

// Query

// string query = "SELECT \* FROM u0490570\_nsi.labpp\_mat\_table WHERE labpp\_mat\_short\_text = \'Материал м² 2\'";

// string query = "SELECT \* FROM u0490570\_nsi.labpp\_mat\_table WHERE labpp\_mat\_short\_text LIKE \'%м²%\'";

string query = "SELECT \* FROM u0490570.my\_table";

int j = 0;

ires = ts\_dbmysql(iMySQL, "real\_query", query);

if (ires != 0) {

return -1;

}

cout << "query success\n";

ires = ts\_dbmysql(iMySQL, "store\_result");

if (ires != 0) {

return -1;

}

cout << "store result success\n";

ires = ts\_dbmysql(iMySQL, "get\_num\_fields");

cout << "num fields = " << ires << "\n";

int count;

ires = ts\_dbmysql(iMySQL, "get\_row\_count",count);

cout << "row count = " << count << "\n";

int iTable;

object("create", "ts\_table", iTable);

ts\_dbmysql(iMySQL, "export\_to\_table", iTable);

ts\_dbmysql(iMySQL, "free\_result");

ts\_dbmysql(iMySQL, "connection\_close");

object("delete", iMySQL);

// Create Dialog

int x, y, w, h, offset;

object("create", "ts\_dialog", iDialog\_MD);

ts\_dialog(iDialog\_MD, "init\_dialog", "palette", 0, 0, 350, 400);

ts\_dialog(iDialog\_MD, "set\_as\_main\_panel");

ts\_dialog(iDialog\_MD, "SetGrowType", 3);

ts\_dialog(iDialog\_MD, "SetTitle", "Наменования материалов для заказчика");

object("create", "ts\_dialogcontrol", iListBox\_MD, "iListBox\_MD");

ts\_dialogcontrol(iListBox\_MD, "init\_control", "singlesellistbox", iDialog\_MD, 5, 5, 340, 350, 48, 20);

ts\_dialogcontrol(iListBox\_MD, "SetAnchorToPanelResize", 0, 0, 1, 1);

ts\_table(iTable, "set\_columns\_width\_when\_export", 100);

ts\_table(iTable, "mapping\_columns\_to\_export",

"id", "№", "",

"\_IDRRef","Идентификатор","",

"\_Description","Наименование для заказчика", "",

"\_ParentIDRRef","Идентификатор родителя", ""

);

ts\_table(iTable, "export\_to\_dialogcontrol", iListBox\_MD, -1, -1);

x = 10;

w = 98;

int delta = 10;

x = x + w + delta;

y = 370;

h = 20;

x = x + w + delta;

object("create", "ts\_dialogcontrol", iButton\_Ok\_MD, "iButton\_Ok\_MD");

ts\_dialogcontrol(iButton\_Ok\_MD, "init\_control", "button", iDialog\_MD, x, y, w, h);

ts\_dialogcontrol(iButton\_Ok\_MD, "eventreaction", "Event\_ButtonClicked", "Event\_ButtonClicked\_MD");

ts\_dialogcontrol(iButton\_Ok\_MD, "settext", "Ok");

ts\_dialogcontrol(iButton\_Ok\_MD, "SetAnchorToPanelResize", 1, 1, 0, 0);

bool bres;

ts\_dialog(iDialog\_MD, "invoke", bres);

//<-

object("delete", iTable);

}

int Event\_ButtonClicked\_MD(int iDescr, string sDescr)

{

if (sDescr == "iButton\_Ok\_MD") {

ts\_dialog(iDialog\_MD, "SendCloseRequest");

}

}

**Работа с СУБД PostgreSQL**



Имеется возможность подключать базы данных PostgreSQL непосредственно из текста программ LABPP.

Скачайте с сайта PostgreSQL файлы:

1. libiconv-2.dll
2. libintl-9.dll
3. libpq.dll
4. libwinpthread-1.dll

и поместите их в каталог ARCHICAD.

**init**

Инициализация объекта для присоединения к БД PostgreSQL из ARCHICAD  
Вызов:  
int ires = ts\_dbpostgresql(int iPostgreSQL, "init");  
Здесь:  
ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения инициализации.  
iPostgreSQL - дескриптор объекта подключения.  
  
Пример:  
  
int iPostgreSQL;  
**object**("create", "ts\_dbpostgresql", iPostgreSQL);  
int ires;  
ires = ts\_dbpostgresql(iPostgreSQL, "init");  
**if** (ires != 0) {  
 **return** -1;  
}  
cout << "init success\n";

**pqconnectdb**

Открыть коннект с базой данных (присоедититься к БД).

Вызов:

int ires = ts\_dbpostgresql(iPostgreSQL, "pqconnectdb", string connectstring);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iPostgreSQL - дескриптор объекта подключения;

connectstring - строка подключения к базе данных;

Пример:

ires = ts\_dbpostgresql(iPostgreSQL, "pqconnectdb", "user=postgres password=Ts0020010 host=localhost port=5432 dbname=postgres target\_session\_attrs=read-write");  
**if** (ires != 0) {  
 **return** -1;  
}  
cout << "connection success\n";

**pqexec**

Выполнить запрос к базе данных.

Запрос может выполняться с использованием символов unicode (например м²).

Результат запроса далее может быть получен, например, в виде таблицы ts\_table.

Вызов:

int ires = ts\_dbpostgresql(iPostgreSQL, "pqexec", string query);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iPostgreSQL - дескриптор объекта подключения;

query - текст запроса к базе данных на языке SQL;

Пример:

string query = "DECLARE myportal CURSOR FOR select \* from pg\_type"; // запросить все записи из таблицы pq\_type и создать курсор myportal  
int j = 0;  
ires = ts\_dbpostgresql(iPostgreSQL, "pqexec", query);  
**if** (ires != 0) {  
 **return** -1;  
}  
cout << "query success\n";

**pqnfields**

Получить количество колонок в результате запроса.

Вызов:

int ires = ts\_dbpostgressql(iPostgreSQL, "pqnfields");

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iPostgreSQL - дескриптор объекта подключения;

Пример:

ires = ts\_dbpostgresql(iPostgreSQL, "pqnfields");  
cout << "num fields = " << ires << "\n";

**pqntuples - кол-во строк**

Получить количество строк в результате запроса.

Вызов:

int ires = ts\_dbpostgresql(iPostgreSQL, "pqntuples", count);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iPostgreSQL - дескриптор объекта подключения;

count - возвращает количество строк в полученной таблице.

Пример:

int count;

ires = ts\_dbpostgresql(iPostgreSQL, "pqntuples",count);

cout << "row count = " << count << "\n";

**export\_to\_table**

Выгрузить результат, полученный по запросу из базы данных, в таблицу ts\_table.

Вызов:

Чтобы получить всю таблицу запроса и типы колонок в таблице расставить в соответствии с типами данных в запросе:

int ires = ts\_dbpostgresql(iPostgreSQL, "export\_to\_table", int iTable);

Чтобы получить только требуемые колонки из запроса и задать тип данных принудительно:

int ires = ts\_dbpostgresql(iPostgreSQL, "export\_to\_table", int iTable, int/string colname1, string labppcoltype1, int/string colname2, string labppcoltype2, ...);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iPostgreSQL - дескриптор объекта подключения;

iTable - дескриптор объекта таблицы ts\_table, в которую требуется выгрузить результат.

int/string colname - индекс или наименование колонки запроса

Пример:

int iTable;

object("create", "ts\_table", iTable);

// вариант 1

ts\_dbpostgresql(iPostgreSQL, "export\_to\_table", iTable);

// вариант 2

ts\_dbpostgresql(iPostgreSQL, "export\_to\_table", iTable, "id","string","count","int",5,"double");

**pqclear**

Освободить память от полученного результата.

Когда результат запроса к БД больше не нужен или требуется выполнить новый запрос - освобождаем память.

Вызов:

int ires = ts\_dbpostgresql(iPostgreSQL, "pqclear");

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения;  
iPostgreSQL - дескриптор объекта подключения;

Пример:

// освободить память результата запроса в подключении у объекта iMySQL  
ts\_dbpostgresql(iPostgreSQL, "clear");

**pqfinish**

Закрыть соединение в объекте подключения, заданном дескриптором.

После закрытия соединения объект можно использовать повторно или для подключения к другим БД.

Вызов:

int ires = ts\_dbpostgresql(iPostgreSQL, "pqfinish");

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iPostgreSQL - дескриптор объекта подключения;

Пример:

ts\_dbpostgresql(iPostgreSQL, "pqfinish");

**Пример - получение таблицы c сайта из базы данных PostgreSQL в диалог ARCHICAD**

Сформируем диалог, который покажет таблицу из большой таблицы из базы данных PostgreSQL.

Параметры подключения:

Сервер - localhost

Имя пользователя - labpp\_user\_ivanov

Пароль пользователя - pspsps

Имя базы данных (схемы) - LABPP\_ERM

Номер порта для подключения (обычно) - 5432;



int iListBox\_MD, iButton\_Ok\_MD;

int iTable\_MD;

int iDialog\_MD;

int main()  
{  
 int iPostgreSQL;  
 **object**("create", "ts\_dbpostgresql", iPostgreSQL);  
 int ires;  
 ires = ts\_dbpostgresql(iPostgreSQL, "init");  
 **if** (ires != 0) {  
 **return** -1;  
 }  
 cout << "init success\n";  
  
 ires = ts\_dbpostgresql(iPostgreSQL, "pqconnectdb", "user=labpp\_user\_ivanov password=pspsps host=localhost port=5432 dbname=LABPP\_ERM target\_session\_attrs=read-write");  
 **if** (ires != 0) {  
 **return** -1;  
 }  
 cout << "connection success\n";  
  
 ires = ts\_dbpostgresql(iPostgreSQL, "pqexec", "BEGIN");  
 **if** (ires != 0) {  
 **return** -1;  
 }  
 cout << "begin transaction success\n";  
  
 ts\_dbpostgresql(iPostgreSQL, "pqclear");  
  
 ires = ts\_dbpostgresql(iPostgreSQL, "pqexec", "DECLARE myportal CURSOR FOR select \* from pg\_type");  
 **if** (ires != 0) {  
 **return** -1;  
 }  
 cout << "creating cursor success\n";  
 ts\_dbpostgresql(iPostgreSQL, "pqclear");  
  
 ires = ts\_dbpostgresql(iPostgreSQL, "pqexec", "FETCH ALL in myportal");  
 **if** (ires != 0) {  
 **return** -1;  
 }  
 cout << "fitch all success\n";  
  
 int nfields = ts\_dbpostgresql(iPostgreSQL, "pqnfields");  
 coutvar << nfields;  
  
 int ntuples = ts\_dbpostgresql(iPostgreSQL, "pqntuples");  
 coutvar << ntuples;  
 // Считываем в таблицу  
 int iTable;  
 **object**("create", "ts\_table", iTable);  
 ts\_dbpostgresql(iPostgreSQL, "export\_to\_table", iTable);  
 string sstr;  
 ts\_table(iTable, "print\_to\_str", sstr);  
 coutvar << sstr;  
 //  
 ts\_dbpostgresql(iPostgreSQL, "pqclear");  
  
 ires = ts\_dbpostgresql(iPostgreSQL, "pqexec", "CLOSE myportal");  
 **if** (ires != 0) {  
 **return** -1;  
 }  
 cout << "close portal success\n";  
  
 ts\_dbpostgresql(iPostgreSQL, "pqclear");  
  
 ires = ts\_dbpostgresql(iPostgreSQL, "pqexec", "END");  
 **if** (ires != 0) {  
 **return** -1;  
 }  
 cout << "end transaction success\n";  
  
 ts\_dbpostgresql(iPostgreSQL, "pqclear");  
  
 // Create Dialog  
 int x, y, w, h, offset;  
  
 **object**("create", "ts\_dialog", iDialog\_MD);  
 ts\_dialog(iDialog\_MD, "init\_dialog", "palette", 0, 0, 350, 400);  
 ts\_dialog(iDialog\_MD, "set\_as\_main\_panel");  
 ts\_dialog(iDialog\_MD, "SetGrowType", 3);  
  
 ts\_dialog(iDialog\_MD, "SetTitle", "Наменования материалов для заказчика");  
  
 **object**("create", "ts\_dialogcontrol", iListBox\_MD, "iListBox\_MD");  
  
 ts\_dialogcontrol(iListBox\_MD, "init\_control", "singlesellistbox", iDialog\_MD, 5, 5, 340, 350, 48, 20);  
 ts\_dialogcontrol(iListBox\_MD, "SetAnchorToPanelResize", 0, 0, 1, 1);  
  
 ts\_table(iTable, "set\_columns\_width\_when\_export", 100);  
  
 ts\_table(iTable, "export\_to\_dialogcontrol", iListBox\_MD, -1, -1);  
  
 x = 10;  
 w = 98;  
 int delta = 10;  
 x = x + w + delta;  
 y = 370;  
 h = 20;  
  
 x = x + w + delta;  
 **object**("create", "ts\_dialogcontrol", iButton\_Ok\_MD, "iButton\_Ok\_MD");  
 ts\_dialogcontrol(iButton\_Ok\_MD, "init\_control", "button", iDialog\_MD, x, y, w, h);  
 ts\_dialogcontrol(iButton\_Ok\_MD, "eventreaction", "Event\_ButtonClicked", "Event\_ButtonClicked\_MD");  
 ts\_dialogcontrol(iButton\_Ok\_MD, "settext", "Ok");  
 ts\_dialogcontrol(iButton\_Ok\_MD, "SetAnchorToPanelResize", 1, 1, 0, 0);  
  
 bool bres;  
 ts\_dialog(iDialog\_MD, "invoke", bres);  
  
 //<-  
 **object**("delete", iTable);  
 **object**("delete", iPostgreSQL);  
}  
  
int Event\_ButtonClicked\_MD(int iDescr, string sDescr)  
{  
 **if** (sDescr == "iButton\_Ok\_MD") {  
 ts\_dialog(iDialog\_MD, "SendCloseRequest");  
 }  
}

**Работа с СУБД SQLite**



Имеется возможность подключать базы данных SQLite непосредственно из текста программ LABPP.

Скачайте с сайта SQLite файл sqlite3.dll и поместите его в каталог ARCHICAD.

**init**

Инициализация объекта для присоединения к БД SQLite из ARCHICAD  
Вызов:  
int ires = ts\_dbsqlite(int iSQLite, "init");  
Здесь:  
ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения инициализации.  
iSQLite - дескриптор объекта подключения.  
  
Пример:  
  
int iSQLite;  
**object**("create", "ts\_dbsqlite", iSQLite);  
int ires;  
ires = ts\_dbsqlite(iSQLite, "init");  
**if** (ires != 0) {  
 **return** -1;  
}  
cout << "init success\n";

**open\_db**

Открыть коннект с базой данных (присоедититься к БД).

Вызов:

int ires = ts\_dbsqlite(iSQLite, "open\_db", string connectstring);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iSQLite - дескриптор объекта подключения;

connectstring - строка подключения к базе данных;

Пример:

ires = ts\_dbsqlite(iSQLite, "open\_db", ""E:/Work/ЛПП2402-SQLite/q.db"");  
**if** (ires != 0) {  
 **return** -1;  
}  
cout << "connection success\n";

**prepare\_v2\_query - запрос к базе данных**

Выполнить запрос к базе данных.

Запрос может выполняться с использованием символов unicode (например м²).

Результат запроса далее может быть получен, например, в виде таблицы ts\_table.

Вызов:

int ires = ts\_dbqlite(iSQLite, "", string query);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iSQLite - дескриптор объекта подключения;

query - текст запроса к базе данных на языке SQL;

Пример:

string query = string query = "select \* from my\_first where (id <= 10000)OR(id >= 90000)";  
int j = 0;  
ires = ts\_dbsqlite(iSQLite, "prepare\_v2\_query", query);  
**if** (ires != 0) {  
 **return** -1;  
}  
cout << "query success\n";

**column\_count**

Получить количество колонок в результате запроса.

Вызов:

int ires = ts\_dbsqlite(iSQLite, "column\_count");

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iSQLite - дескриптор объекта подключения;

Пример:

ires = ts\_dbsqlite(iSQLite, "column\_count");  
cout << "num fields = " << ires << "\n";

**row\_count**

Получить количество строк в результате запроса.

Вызов:

int ires = ts\_dbsqlite(iSQLite, "row\_count", count);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iSQLite - дескриптор объекта подключения;

count - возвращает количество строк в полученной таблице.

Пример:

int count;  
ires = ts\_dbsqlite(iSQLite, "row\_count",count);  
cout << "row count = " << count << "\n";

**export\_to\_table**

Выгрузить результат, полученный по запросу из базы данных, в таблицу ts\_table.

Вызов:

Чтобы получить всю таблицу запроса и типы колонок в таблице расставить в соответствии с типами данных в запросе:

int ires = ts\_dbsqlite(iSQLite, "export\_to\_table", int iTable);

Чтобы получить только требуемые колонки из запроса и задать тип данных принудительно:

int ires = ts\_dbsqlite(iSQLite, "export\_to\_table", int iTable, int/string colname1, string labppcoltype1, int/string colname2, string labppcoltype2, ...);

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iSQLite - дескриптор объекта подключения;

iTable - дескриптор объекта таблицы ts\_table, в которую требуется выгрузить результат.

int/string colname - индекс или наименование колонки запроса

Пример:

int iTable;

object("create", "ts\_table", iTable);

// вариант 1

ts\_dbsqlite(iSQLite, "export\_to\_table", iTable);

// вариант 2

ts\_dbsqlite(iSQLite, "export\_to\_table", iTable, "id","string","count","int",5,"double");

**finalize\_stmt**

Освободить память от полученного результата.

Когда результат запроса к БД больше не нужен или требуется выполнить новый запрос - освобождаем память.

Вызов:

int ires = ts\_dbsqlite(iSQLite, "finalize\_stmt");

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения;  
iSQLite - дескриптор объекта подключения;

Пример:

// освободить память результата запроса в подключении у объекта iSQLite  
ts\_dbsqlite(iSQLite, "finalize\_stmt");

**close\_db**

Закрыть соединение в объекте подключения, заданном дескриптором.

После закрытия соединения объект можно использовать повторно или для подключения к другим БД.

Вызов:

int ires = ts\_dbsqlite(iSQLite, "close\_db");

Здесь:

ires = возвращаемое значение -1/0 - в случае ошибки/успешного выполнения операции;  
iSQLite - дескриптор объекта подключения;

Пример:

ts\_dbsqlite(iPostgreSQL, "close\_db");

**Пример - получение таблицы c сайта из базы данных PostgreSQL в диалог ARCHICAD**

Сформируем диалог, который покажет таблицу из большой таблицы из базы данных SQLite.

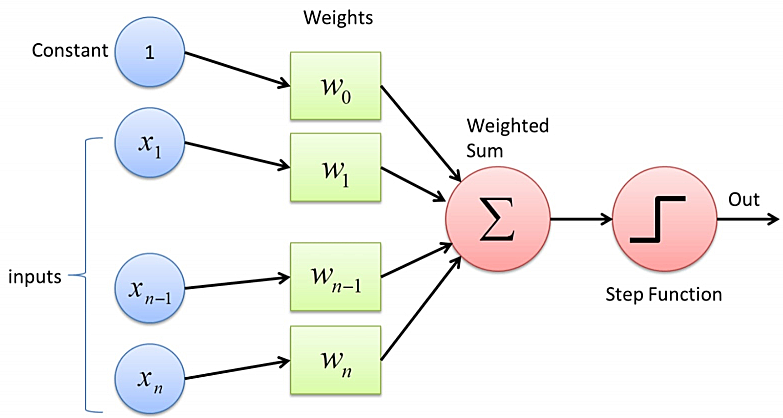
Если в качестве пути к базе данных указать любое имя, то база данных будет создана в памяти компьютера.



int iListBox\_MD, iButton\_Ok\_MD;  
int iTable\_MD;  
int iDialog\_MD;  
  
int main()  
{  
 int iSQLite;  
 **object**("create", "ts\_dbsqlite", iSQLite);  
 int ires;  
 ires = ts\_dbsqlite(iSQLite, "init");  
 **if** (ires != 0) {  
 **return** -1;  
 }  
 cout << "init success\n";  
 ires = ts\_dbsqlite(iSQLite, "open\_db", "E:/Work/ЛПП2402-SQLite/q.db");  
 **if** (ires != 0) {  
 **return** -1;  
 }  
 cout << "connection success\n";  
  
 string query = "select \* from my\_first where (id <= 10000)OR(id >= 90000)";  
   
 int j = 0;  
  
 ires = ts\_dbsqlite(iSQLite, "prepare\_v2\_query", query);  
 **if** (ires != 0) {  
 **return** -1;  
 }  
 cout << "query success\n";  
  
 ires = ts\_dbsqlite(iSQLite, "column\_count");  
 cout << "num fields = " << ires << "\n";  
  
 int iTable;  
 **object**("create", "ts\_table", iTable);  
  
 ts\_dbsqlite(iSQLite, "export\_to\_table", iTable, 0,"string",1,"string", 2, "string");  
 string sstr;  
 ts\_table(iTable, "print\_to\_str", sstr);  
 coutvar << sstr;  
  
 ts\_dbsqlite(iSQLite, "finalize\_stmt");  
 ts\_dbsqlite(iSQLite, "close\_db");  
  
 // Create Dialog  
 int x, y, w, h, offset;  
  
 **object**("create", "ts\_dialog", iDialog\_MD);  
 ts\_dialog(iDialog\_MD, "init\_dialog", "palette", 0, 0, 350, 400);  
 ts\_dialog(iDialog\_MD, "set\_as\_main\_panel");  
 ts\_dialog(iDialog\_MD, "SetGrowType", 3);  
  
 ts\_dialog(iDialog\_MD, "SetTitle", "Наменования материалов для заказчика");  
  
 **object**("create", "ts\_dialogcontrol", iListBox\_MD, "iListBox\_MD");  
  
 ts\_dialogcontrol(iListBox\_MD, "init\_control", "singlesellistbox", iDialog\_MD, 5, 5, 340, 350, 48, 20);  
 ts\_dialogcontrol(iListBox\_MD, "SetAnchorToPanelResize", 0, 0, 1, 1);  
  
 ts\_table(iTable, "set\_columns\_width\_when\_export", 100);  
  
 ts\_table(iTable, "export\_to\_dialogcontrol", iListBox\_MD, -1, -1);  
  
 x = 10;  
 w = 98;  
 int delta = 10;  
 x = x + w + delta;  
 y = 370;  
 h = 20;  
  
 x = x + w + delta;  
 **object**("create", "ts\_dialogcontrol", iButton\_Ok\_MD, "iButton\_Ok\_MD");  
 ts\_dialogcontrol(iButton\_Ok\_MD, "init\_control", "button", iDialog\_MD, x, y, w, h);  
 ts\_dialogcontrol(iButton\_Ok\_MD, "eventreaction", "Event\_ButtonClicked", "Event\_ButtonClicked\_MD");  
 ts\_dialogcontrol(iButton\_Ok\_MD, "settext", "Ok");  
 ts\_dialogcontrol(iButton\_Ok\_MD, "SetAnchorToPanelResize", 1, 1, 0, 0);  
  
 bool bres;  
 ts\_dialog(iDialog\_MD, "invoke", bres);  
  
 //<-  
 **object**("delete", iTable);  
 **object**("delete", iSQLite);  
}  
  
int Event\_ButtonClicked\_MD(int iDescr, string sDescr)  
{  
 **if** (sDescr == "iButton\_Ok\_MD") {  
 ts\_dialog(iDialog\_MD, "SendCloseRequest");  
 }  
}

**Искусственный интеллект**

**Нейросеть на основе перцептрона Розенблатта - ts\_aipercrozen**



В LABPP имеется встроенный механизм на основе многослойного перцептрона Розенблатта с обратным распределением ошибки.

Это один из самых удачных вариантов инструмента искусственного интеллекта.

Например, на вход могут подаваться сотни чисел, а на выходе, в результате получаются обобщающие сочетания нескольких чисел, которые заранее известны и в целом описывают ситуацию, на которую он обучен.

Может обучаться, загружать и выгружать результаты обучения, выполнять тестирование задач по одной или списком.

Высокоэффективная реализация на низкоуровневом машинном языке дает высокие показатели быстродействия.

**init**

Инициализация перцептрона.

Вызов:

int ires = ts\_aipercrozen(int iPercRozen, "init", int in1,int in2,int inn,int **out**);

Здесь:

ires = -1 или 0, если нет ошибки.

iPercRozen - дескриптор объекта перцептрона.

int in1 .... out - числовые характеристики слоев перцептрона.

Пример.

Инициировать перцептрон 500:2 со слоями нейронов:

|  |  |  |
| --- | --- | --- |
| Индекс слоя | Входные нейроны | Выходные нейроны |
| 0 | 500 | 200 |
| 1 | 200 | 50 |
| 2 | 50 | 5 |
| 3 | 5 | 2 |

int iPercRozen;  
**object**("create","ts\_percrozen",iPercRozen);  
int ires = ts\_aipercrozen(int iPercRozen, "init", 500, 200, 50, 5, 2);

**task\_add\_to\_list**

Добавить задачу для решения или обучения в список задач.

Объект перцептрона имеет внутренний список задач.

Задача состоит из входного набора чисел, количество которых соответствуе числу входных нейронов, второго набора чисел, соответствующих числу выходных нейронов (требуемые значения) и названия задачи.

Перед обучением мы создаем наборы входных значений с соответствующим набором желаемых значений на выходе.

Вызов:

int ires = ts\_aipercrozen(int iPercRozen, "task\_add\_to\_list", string taskname, int iTableIN, int iTableOUT);

Здесь:

ires - отрицательное значение или 0, в случае успешного выполнения;

taskname - имя задачи;

iTableIN - плоская таблица в одну колонку с набором числовых входных параметров типа double;

iTableOUT - плоская таблица в одну колонку с набором числовых желаемых выходных параметров для обучения.

Пример.

Задать две задачи для обучения перцептрона на 200 входных и 2 выходных нейрона. Провести обучение с обратным распределением ошибки и вывести результаты в окно сообщений.

int iTable11;

object("create", "ts\_table", iTable11);

ts\_table(iTable11, "add\_column", 0, "double", "Вход");

int i, icount;

icount = 200;

for (i = 0; i < icount; i++)

{

ts\_table(iTable11, "add\_row", 0, i+1);

}

int iTable12;

object("create", "ts\_table", iTable12);

ts\_table(iTable12, "add\_column", 0, "double", "Выход для обучения");

ts\_table(iTable12, "add\_column", 1, "double", "Выход реальный");

ts\_table(iTable12, "add\_row", 0, 0.77, 1,0);

ts\_table(iTable12, "add\_row", 0, 0.88, 1,0);

ts\_aipercrozen(iPercRozen, "task\_add\_to\_list", "Задача 1", iTable11, iTable12);

int iTable1;

object("create", "ts\_table", iTable21);

ts\_table(iTable21, "add\_column", 0, "double", "Вход");

icount = 200;

for (i = 0; i < icount; i++)

{

ts\_table(iTable21, "add\_row", 0, 100 - i);

}

int iTable22;

object("create", "ts\_table", iTable22);

ts\_table(iTable22, "add\_column", 0, "double", "Выход для обучения");

ts\_table(iTable22, "add\_column", 1, "double", "Выход реальный");

ts\_table(iTable22, "add\_row", 0, 0.55, 1, 0);

ts\_table(iTable22, "add\_row", 0, 0.77, 1, 0);

ts\_aipercrozen(iPercRozen, "task\_add\_to\_list", "Задача 2", iTable21, iTable22);

ts\_aipercrozen(iPercRozen, "learn\_task\_list", 200000);

ts\_aipercrozen(iPercRozen, "run\_task\_list");

ts\_aipercrozen(iPercRozen, "get\_task\_result\_to\_table", 0, iTable12,false,1);

string str;

ts\_table(iTable12, "print\_to\_str", str);

coutvar << str;

ts\_aipercrozen(iPercRozen, "get\_task\_result\_to\_table", 1, iTable22,false,1);

ts\_table(iTable22, "print\_to\_str", str);

coutvar << str;

**learn\_task\_list**

Выполнить обучение перцептрона на заранее заданном списке задач c эталонными желаемыми выходными значениями.

Вызов:

int ires = ts\_aipercrozen(int iPercRozen, "learn\_task\_list", int learncount);

Здесь:

ires - отрицательное значение или 0, при успешном выполнении обучения;

iPercRozen - дескриптор объекта перцептрона;

learncount - число итераций обучения с обратным распределением ошибки.

**run\_task\_list**

Получить результаты для заранее заданного списка задач.

Вызов:

int ires = ts\_aipercrozen(iPercRozen, "run\_task\_list");

Здесь:

ires - отрицательное значение или 0 при успешном выполении.

**get\_task\_result\_to\_table**

Вывести результат выполнения задачи в таблицу.

Вызов:

int ires = ts\_aipercrozen(int iPercRozen, "get\_task\_result\_to\_table", int iTaskIndex, int iTableOUT, bool bDoClearTable, int iColumnIndex);

Здесь:

ires - минус или 0 при успешном выполнении;

iPercRozen - дескриптор объекта перцептрона;

iTaskIndex - индекс задачи в списке заранее подготовленных задач;

iTableOUT - дескриптор выходной таблицы;

bDoClearTable - если истина, то таблица перед заполнением результатами полностью очищается;

В ином случае данные типа double дополняются в колонку с индексом iColumnIndex.

Число строк должно соответствовать числу выходных нейронов.

iColumnIndex - индекс колонки таблицы, куда выводить результат если не задана очистка таблицы.

**task\_list\_clear**

Очистить список задач у заданного объекта перцептрона.

Вызов:

ts\_aipercrozen(int iPercRozen, "task\_list\_clear");

Здесь:

iPercRozen - дескриптор объекта перцептрона.

**store\_matrices и restore\_matrices**

Запомнить матрицы весов во внутреннем буфере перцептрона. Восстановить матрицы весов из внутреннего буфера перцептрона.

При обучении перцептрон настраивает весовые коэффициенты. Так можно временно сохранить их не тронутыми при следующих итерациях обучения.

Вызов:

ts\_aipercrozen(iPercRozen, "store\_matrices");  
ts\_aipercrozen(iPercRozen, "restore\_matrices");

**clear**

Освободить память.

Операция выполняется автоматически перед новым вызовом init или при удалении объекта.

Вызов:

ts\_aipercrozen(iPercRozen, "clear");

**ts\_aipercrozen\_matrices**

Объект для хранения и операций с матрицами весовых коэффициентов перцептрона Розенблатта.

init

Инициализация объекта матриц перцептрона.

Если объект используется для вывода матриц из существующего перцептрона, то инициализация выполняется автоматически (т.е. инициализация не требуется).

Вызов:

int ires = ts\_aipercrozen\_matrices(int iPercRozenMatrices, "init", int in1,int in2,int inn,int **out**);

Здесь:

ires = -1 или 0, если нет ошибки.

iPercRozenMatrices - дескриптор объекта матриц перцептрона.

int in1 .... out - числовые характеристики слоев перцептрона.

get\_from\_percrozen

Получить матрицы из существующего перцептрона.

Вызов:

int ires = ts\_aipercrozen\_matrices(int iPercRozenMatrices, "get\_from\_percrozen", int iPercRozen);

Здесь:

ires = -1 или 0, если нет ошибки;

iPercRozenMatrices - дескриптор объекта матриц перцептрона;

iPercRozen - дескриптор существующего перцептрона.

Пример:

int iPercRozenMatrices;  
**object**("create", "ts\_aipercrozen\_matrices", iPercRozenMatrices);  
ts\_aipercrozen\_matrices(iPercRozenMatrices, "get\_from\_percrozen", iPercRozen);

set\_to\_percrozen

Записать матрицы весовых коэффициентов в существующий перцептрон.

Вызов:

int ires = ts\_aipercrozen\_matrices(int iPercRozenMatrices, "set\_to\_percrozen", int iPercRozen);

Здесь:

ires = -1 или 0, если нет ошибки;

iPercRozenMatrices - дескриптор объекта матриц перцептрона;

iPercRozen - дескриптор существующего перцептрона.

Пример:

Получить матрицы из базы данных и записать в перцептрон.

int iPercRozenMatrices;  
**object**("create", "ts\_aipercrozen\_matrices", iPercRozenMatrices);

ts\_aipercrozen\_matrices(iPercRozenMatrices, "load\_from\_PostgreSQL", iPercRozen, iDBPostgreSQL, sTableName, percGUID);  
ts\_aipercrozen\_matrices(iPercRozenMatrices, "set\_to\_percrozen", iPercRozen);

ts\_aipercrozen\_matrices(iPercRozenMatrices, "set\_to\_percrozen", iPercRozen);

export\_to\_table

Передать матрицы весовых коэффициентов в таблицу ts\_table.

Вызов:

int ires = ts\_aipercrozen\_matrices(int iPercRozenMatrices, "export\_to\_table", int iTable);

Здесь:

ires = -1 или 0, если нет ошибки;

iPercRozenMatrices - дескриптор объекта матриц перцептрона;

iTable - дескриптор таблицы.

Структура таблицы при выполнении операции формируется автоматически.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер слоя | Число входных нейронов | Число выходных нейронов | Весовой коэффициент  W1 | Веловой коэффициент W... | Весовой коэффициент Wn |
| 0 | 100 | 30 | 1.2 | ... | ... |
| 0 | 100 | 30 | ... | ... | ... |
| ... | ... | ... | ... | ... | ... |
| 1 | 30 | 10 | 0.000123 | ... | ... |

Пример:

int iTableMatrices;  
**object**("create", "ts\_table", iTableMatrices);  
ts\_aipercrozen\_matrices(iPercRozenMatrices, "export\_to\_table", iTableMatrices);

import\_from\_table

Получить матрицы весовых коэффициентов из таблицы ts\_table.

Обычно используется для загрузки/выгрузки опыта ИИ в базы данных, текстовые файлы, JSON конструкции и т.п.

Вызов:

int ires = ts\_aipercrozen\_matrices(int iPercRozenMatrices, "import\_from\_table", int iTable);

Здесь:

ires = -1 или 0, если нет ошибки;

iPercRozenMatrices - дескриптор объекта матриц перцептрона;

iTable - дескриптор таблицы.

На момент импорта таблица должна иметь структуру и состав, соответствующий настройкам объекта ts\_aipercrozen\_matrices, который в свою очередь соответствует ts\_aipercrozen объекту перцептрона (число слоев, их размерность и число строк данных).

При обнаружении несоответствия структуры данных программа выдает ошибку.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер слоя | Число входных нейронов | Число выходных нейронов | Весовой коэффициент  W1 | Веловой коэффициент W... | Весовой коэффициент Wn |
| 0 | 100 | 30 | 1.2 | ... | ... |
| 0 | 100 | 30 | ... | ... | ... |
| ... | ... | ... | ... | ... | ... |
| 1 | 30 | 10 | 0.000123 | ... | ... |

Пример:

ts\_aipercrozen\_matrices(iPercRozenMatrices, "import\_from\_table", iTableMatrices);

clear

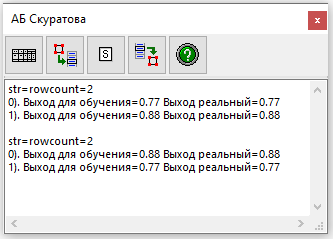
Освободить память.

Операция выполняется автоматически перед новым вызовом init или при удалении объекта.

Вызов:

ts\_aipercrozen\_matrices(iPercRozen, "clear");

**Пример**



int main()

{

int iPercRozen;

object("create", "ts\_aipercrozen", iPercRozen);

ts\_aipercrozen(iPercRozen, "init", 200,100,10,5,2);

int iTable11;

object("create", "ts\_table", iTable11);

ts\_table(iTable11, "add\_column", 0, "double", "Вход");

int i, icount;

icount = 200;

for (i = 0; i < icount; i++)

{

ts\_table(iTable11, "add\_row", 0, i+1);

}

int iTable12;

object("create", "ts\_table", iTable12);

ts\_table(iTable12, "add\_column", 0, "double", "Выход для обучения");

ts\_table(iTable12, "add\_column", 1, "double", "Выход реальный");

ts\_table(iTable12, "add\_row", 0, 0.77, 1,0);

ts\_table(iTable12, "add\_row", 0, 0.88, 1,0);

ts\_aipercrozen(iPercRozen, "task\_add\_to\_list", "Задача 1", iTable11, iTable12);

int iTable21;

object("create", "ts\_table", iTable21);

ts\_table(iTable21, "add\_column", 0, "double", "Вход");

icount = 200;

for (i = 0; i < icount; i++)

{

ts\_table(iTable21, "add\_row", 0, 200 - i);

}

int iTable22;

object("create", "ts\_table", iTable22);

ts\_table(iTable22, "add\_column", 0, "double", "Выход для обучения");

ts\_table(iTable22, "add\_column", 1, "double", "Выход реальный");

ts\_table(iTable22, "add\_row", 0, 0.88, 1, 0);

ts\_table(iTable22, "add\_row", 0, 0.77, 1, 0);

ts\_aipercrozen(iPercRozen, "task\_add\_to\_list", "Задача 2", iTable21, iTable22);

ts\_aipercrozen(iPercRozen, "learn\_task\_list", 200000);

ts\_aipercrozen(iPercRozen, "run\_task\_list");

ts\_aipercrozen(iPercRozen, "get\_task\_result\_to\_table", 0, iTable12,false,1);

string str;

ts\_table(iTable12, "print\_to\_str", str);

coutvar << str;

ts\_aipercrozen(iPercRozen, "get\_task\_result\_to\_table", 1, iTable22,false,1);

ts\_table(iTable22, "print\_to\_str", str);

coutvar << str;

object("delete", iTable11);

object("delete", iTable12);

object("delete", iTable21);

object("delete", iTable22);

object("delete", iPercRozen);

}

**BIM Master**

BIM Master - это удобный многооконный многофункциональный инструмент, который позволяет получать, анализировать и изменять данные в элементах.

Можно открывать несколько окон для сравнения информации.

Окно может хранить и отображать информацию из одного и того же или из разных элементов для разных значений параметров.

BIM Master используется в качестве отладчика в программах LABPP\_Automat.

В тексте программы можно указывать директиву breakpoint() для получения слепка данных на момент выполнения этой команды в программе.

Отладчик с фунциями браузера позволяет просматривать и изменять информацию у элементов.

Вы можете, например, увидеть значения параметров (в том числе заданных формулами), значения параметров GDL элементов, IFC свойства и т.д.

Отладчик с функциями браузера данных элементов вызывается сочетанием клавиш

Ctrl+Shift+F9

Для просмотра данных об элементе выберите его в окне и нажмите кнопку 

Окно считает данные из элемента.

Обратите внимание, чтобы был выделен только один элемент.

Если было выделено несколько элементов, то данные будут показаны из первого элемента в группе.

Окно запоминает элемент.

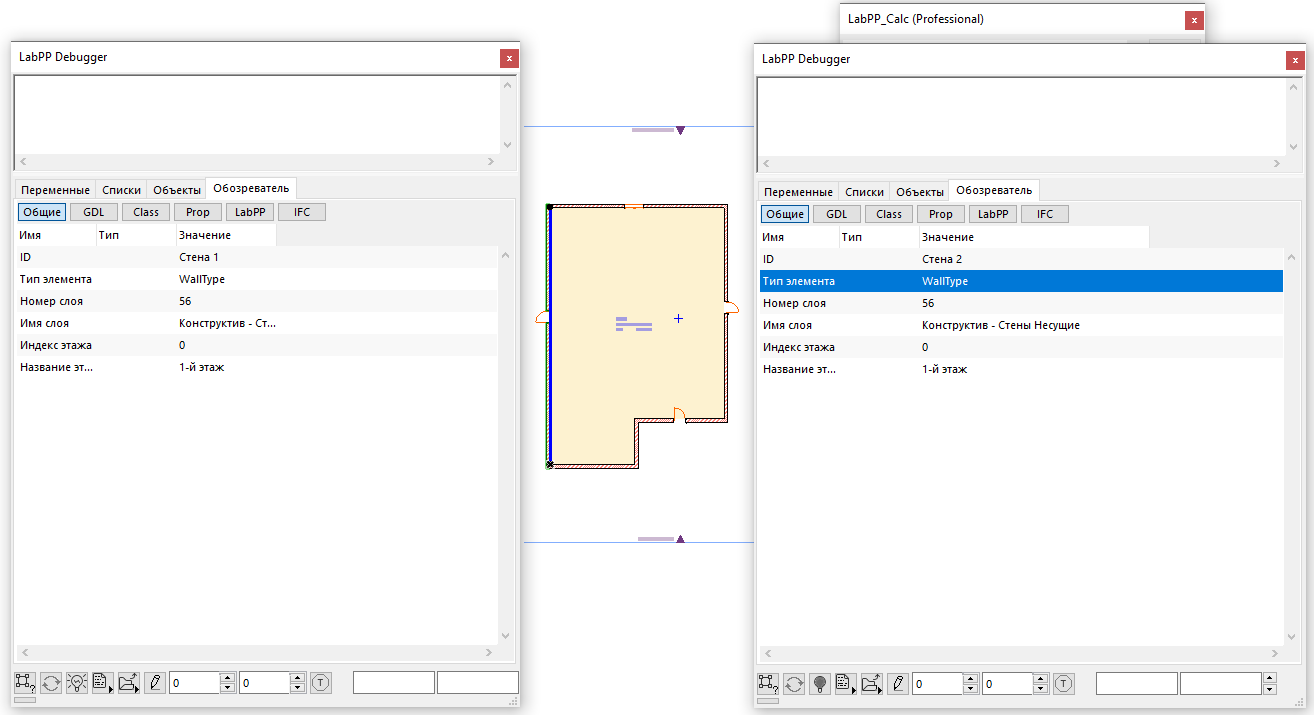
Подсветить считанный элемент можно узкой кнопкой в левом нижнем углу.

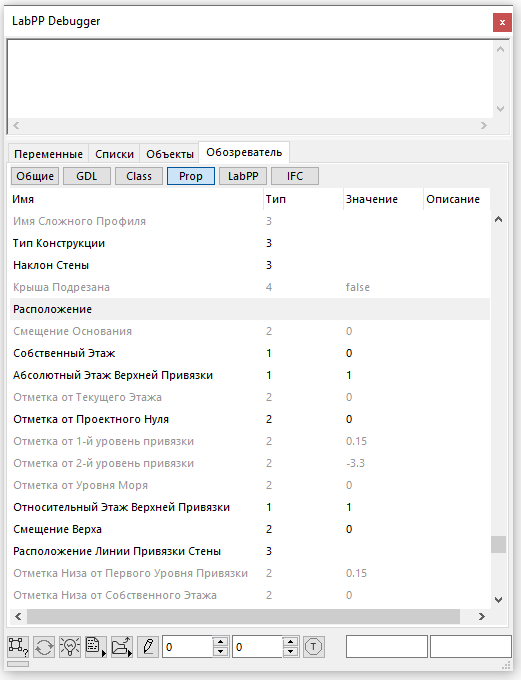
Второй слева кнопкой можно повторно считать данные из того же элемента.

Чтобы показать данные сразу нескольких элементов можно вызвать сколько угодно отладчика.

Второе и последующие окна открываются если в остальных окнах кнопка  находится в положении .

Тогда нажатие сочетания Ctrl+Shift+F9 вызовет еще одно окно и т.д.





**breakpoint**

Открыть BIM Master'а из программы в качестве отладчика.

Вызов:

breakpoint(int icondition);

или

breakpoint(int icondition, bool/int/double/string condition2);

Здесь:

icondition - любое целое число. Окно BIM Master будет вызываться если в нем указано это же число в поле

**programcodegen**

**get\_bimmaster\_cur\_selected\_parameter\_info**

Получить из панели BIM Master информацию о выбранном параметре.

Вызов:

programcodegen("get\_bimmaster\_cur\_selected\_parameter\_info", string genesis, string v\_type, string varname, string svalue, string sdescription, string sgroup);

Здесь:

genesis - генезис переменной (что это - пользовательский параметр, GDL параметр и т.п.)

v\_type - тип переменной

|  |  |
| --- | --- |
| "string" | Строка |
| "real" | Число |
| "integer" | Целое число |
| "boolean" | Логическая переменная |

varname - наименование переменной

svalue - значение переменной

sdescription - описание переменной

sgroup - группа переменной

**get\_bimmaster\_cur\_element\_guidastext**

Вызвать BIM Master для элемента с заданным текстовым GUID.

Вызов:

programcodegen("get\_bimmaster\_cur\_element\_guidastext", string sGUIDAsText);

Здесь:

sGUIDAsText - GUID, заданный текстом, элемента, данные которого необходимо отразить в панели BIM Master.

**show\_bimmaster\_as\_codegen**

Вызвать BIM Master в качестве генератора программного кода.

Вызов:

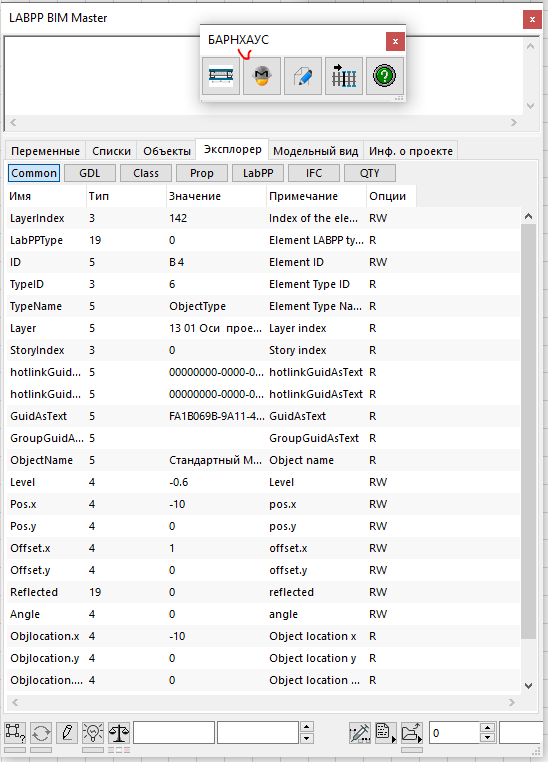
programcodegen("show\_bimmaster\_as\_codegen");

**show\_bimmaster**

Вызывает панель BIM Master в обычном режиме для инспектирования данных.

Вызов:

programcodegen("show\_bimmaster");



**Уроки**

**Вычислить середину высоты элемента Морф относительно 0 проекта и записать в поле LabPP\_Calc**

Скачать файл можно [здесь](http://www.labpp.ru/TSDownload/LabPP_Automat%20Lesson1.cpp).

// Урок 1

// Для начала - в проекте АРХИКАД создать элемент типа Морф.

// задать ему классификатор "Лестница" по классификации "Классификация ARCHICAD"

// выбрать его

// открыть LabPP\_Calc

// открыть в нем правой нижней маленькой кнопкой окно сообщений

// второй справа нижней маленькой кнопкой запустить этот программный модуль средствами LabPP\_Calc

// Результат - в расчетное поле "a" калькулятора запишется высота середины элемента Морф а в комментарий - его ID

// Задание 1

// Сделать так, чтобы значение записывалось во второе поле

// Задание 2

// Создать в классификаторе "Классификация ARCHICAD" класс "Отделка стен".

// Присвоить этот класс элементу Морф

// Сделать так, чтобы в поле LabPP\_Calc записывалась середина высоты этого элемента.

// Подсказка: при выборке элементов из проекта - слово "Лестница" надо на что-то заменить

// Задание 3

// Результат перед записью в поле увеличить на 10% (умножить на 1.1)

int main()

{

// загрузить из проекта в список 1 все элементы, у которых присвоено любое значение классификатора ARCHICAD

ac\_request\_special("add\_elements\_list\_from\_selection", 1, "MorphType", 2,

"", "Cls", "Классификация ARCHICAD", "=", "Лестница", "");

// запросить количество собранных элементов

ac\_request("get\_loaded\_elements\_list\_count", 1);

int icount = ac\_getnumvalue(); // получить количество в переменную

coutvar << icount; // вывести в окно сообщений

if (icount == 0)

{

cout << "В списке нет элементов";

return -1;

}

// объявляем необходимые переменные для работы

double dLevel, dHeight, elemlevel, elemprojectlevel;

string sElemTypeName, sID;

int ielemstoryindex;

int i;

int ires;

// выполнить цикл icount раз

// в нашем случае нам нужен один элемент, поэтому отработаем цикл только 1 раз на элементе с индексом 0

icount = 1;

for (i = 0; i < icount; i++)

{

ires = ac\_request("set\_current\_element\_from\_list", 1, i); // сделать текущим i-вый элемент из списка 1

ires = ac\_request("get\_element\_value", "Level"); // запрашиваем значение свойства Level (для Морф - высота над уровнем его этажа)

dLevel = ac\_getnumvalue(); // получаем его в переменную

coutvar << dLevel; // выводим в окно сообщений

ires = ac\_request("get\_quantity\_value", "max\_height"); // запрашиваем максимальный размер Морф по вертикали

dHeight = ac\_getnumvalue(); // получаем его в переменную

coutvar << dHeight; // выводим в окно сообщений

elemlevel = dLevel + dHeight / 2; // рассчитываем уровень центра Морф по высоте относительно этажа

ac\_request("get\_element\_value", "StoryIndex"); // запрашиваем индекс этажа

ielemstoryindex = ac\_getnumvalue(); // получаем индекс этажа в переменную

// получаем высоту центра Морф относительно 0 проекта в переменную elemprojectlevel

ac\_request("get\_levelfromprojectnull\_by\_floorindexandlevel", ielemstoryindex, elemlevel, elemprojectlevel);

coutvar << elemprojectlevel; // выводим в окно сообщений

ires = ac\_request("get\_element\_value", "ID"); // запросить ID элемента

sID = ac\_getstrvalue(); // получить его в переменную

coutvar << sID; // вывести ID в окно сообщений

}

string svalue = ecvt(elemprojectlevel);

string scomment = "Высота элемента с ID=" + sID + "от 0 проекта";

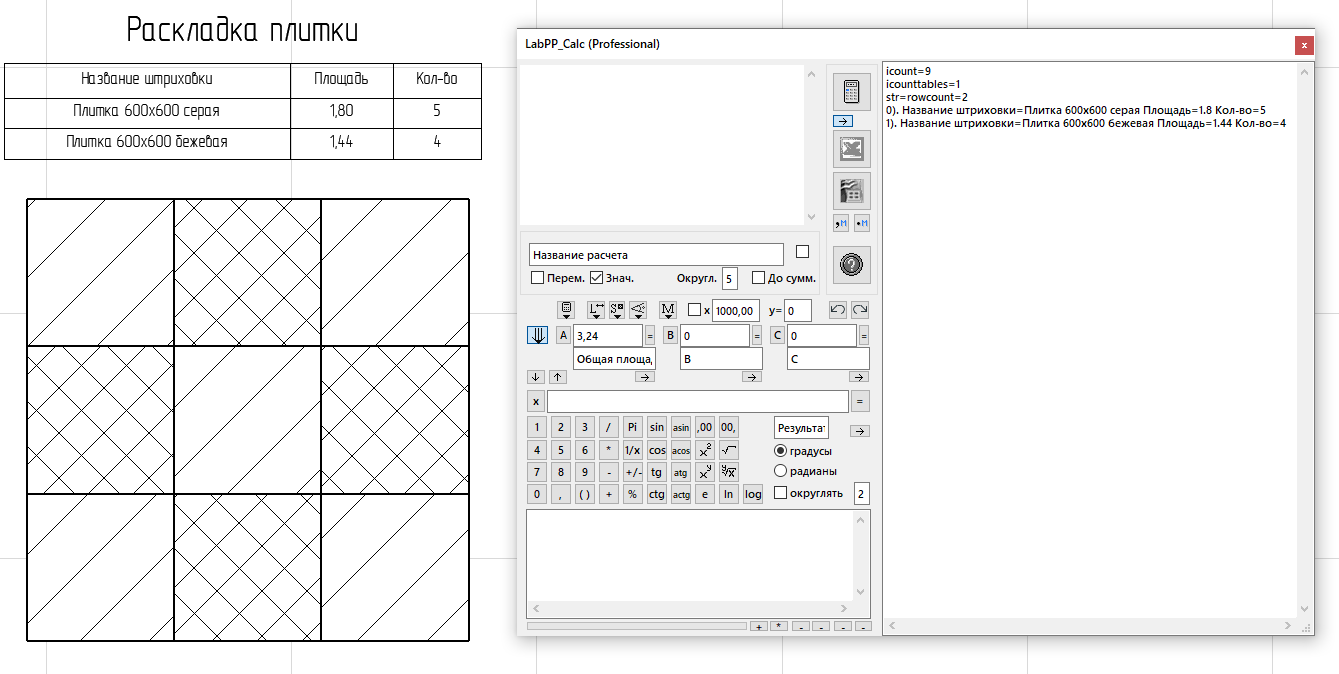
ac\_request("interface", "calc\_field", "set", "a", 1, svalue, scomment);

// "a" - здесь имя ячейки у LabPP\_Calc куда записать значение. Всего ячеек - a,b,c и main

}

**Заполнить таблицу с расчетом плитки по элементам Штриховка и записать общую площадь в поле LabPP\_Calc**

Скачать файл можно [здесь](http://www.labpp.ru/TSDownload/LabPP_Automat%20Lesson2.cpp).



// Урок 2

// Для начала - в проекте АРХИКАД в любом окне разложить штриховки 600х600. Задать им типы штриховок произвольно.

// Поставить там же GDL элемент LABPP\_Table.gsm.

// Выделить одновременно все штриховки и элемент таблицы.

// открыть LabPP\_Calc

// открыть в нем правой нижней маленькой кнопкой окно сообщений

// второй справа нижней маленькой кнопкой запустить этот программный модуль средствами LabPP\_Calc

// Результат - в таблицу запишется список штриховок с названием типа и количеством

// Задание 1

// Переименовать колонку "Название штриховки" в "Тип плитки"

// Задание 2

// Поменять местами колонки.

// Задание 3

// Упорядочить строки по колонке с названием штриховки (см. команду "Sort")

int main()

{

// загрузить из проекта в список 1 все выбранные элементы штриховок

ac\_request\_special("add\_elements\_list\_from\_selection", 1, "HatchType", 2);

// зарузить из проекта в список 2 все выбранные элементы таблиц

ac\_request\_special("add\_elements\_list\_from\_selection", 2, "ObjectType", 2,

"", "EP", "ObjectName", "=", "LABPP\_Table", "");

// запросить количество собранных элементов штриховок

ac\_request("get\_loaded\_elements\_list\_count", 1);

int icount = ac\_getnumvalue(); // получить количество в переменную

coutvar << icount; // вывести в окно сообщений

if (icount == 0)

{

cout << "В списке нет элементов штриховок";

return -1;

}

// запросить количество собранных элементов штриховок

ac\_request("get\_loaded\_elements\_list\_count", 2);

int icountTables = ac\_getnumvalue(); // получить количество в переменную

coutvar << icountTables; // вывести в окно сообщений

if (icountTables == 0)

{

cout << "В списке нет элемента таблицы";

return -1;

}

int iTable;

object("create", "ts\_table", iTable); // создать таблицу

// Создаем колонки таблицы

ts\_table(iTable, "add\_column", 0, "string", "Название штриховки");

ts\_table(iTable, "add\_column", 1, "double", "Площадь");

ts\_table(iTable, "add\_column", 2, "int", "Кол-во");

ts\_table(iTable, "set\_first\_key", 0); // по этой колонке будет отслеживаться одинаковость строк при суммировании показатей

// объявляем необходимые переменные для работы

double dSquare, dAllSquare=0;

string sFillName;

int i;

int ires;

// выполнить цикл icount раз чтобы заполнить таблицу.

// обратите внимание, что одинаковые строки в таблице суммируются автоматически ("add\_row\_sum")

for (i = 0; i < icount; i++)

{

ires = ac\_request("set\_current\_element\_from\_list", 1, i); // сделать текущим i-вый элемент из списка 1

ires = ac\_request("get\_element\_value", "fillName"); // запрашиваем название типа штриховки у элемента штриховки

sFillName = ac\_getstrvalue(); // получаем его в переменную

ires = ac\_request("get\_quantity\_value", "Surface"); // запрашиваем площадь элемента штриховки

dSquare = ac\_getnumvalue(); // получаем его в переменную

dAllSquare += dSquare; // суммируем общую площадь

// Добавляем строку в таблицу, если уже есть - то суммируем

ts\_table(iTable, "add\_row\_sum", 0, sFillName, 1, dSquare, 2, 1);

}

// вывести содержимое таблицы в окно сообщений

string str;

ts\_table(iTable, "print\_to\_str", str);

coutvar << str;

// вывести содержимое таблицы в элемент LABPP\_Table.gsm

ac\_request("set\_current\_element\_from\_list", 2, 0); // сделать текущим 0-ой элемент из списка 2

bool withheader = true; // выводить с заголовками колонок

int startcolindex = 1; // стартовая колонка в LABPP\_Talbe будет 1

int startObjectrow = 1; // стартовая строка в LABPP\_Table будет 1 (т.е. с левого верхнего угла)

int starttablerow = 0; // из таблицы iTable начинать с первой строки (индекс первой строки 0)

int maxcolumns; // сколько колонок задействовано

int maxrows; // сколько строк получилось задействовано

// получить число колонок и строк

ts\_table(iTable, "get\_columns\_count", maxcolumns);

ts\_table(iTable, "get\_rows\_count", maxrows);

ts\_table(iTable, "export\_to\_LABPP\_Table", withheader, startcolindex, startObjectrow, starttablerow, maxrows);

// Задать в объекте количество использованных строк

ac\_request("set\_object\_property\_value", "TS\_RowsUsedCount", maxrows + 1);

// Задать в объекте количество использованных колонок

ac\_request("set\_object\_property\_value", "TS\_ColumnsUsedCount", maxcolumns);

// Дать команду GDL объекту таблицы LABPP\_Table выполнить внутренний перерасчет

ac\_request("set\_object\_property\_value", "TS\_DoRecalcAll", 1);

// Удалить из памяти таблицу

object("delete", iTable);

// Записываем общую площадь в поле "a" калькулятора с комментарием

string svalue = ecvt(dAllSquare); // переводим общую площадь в текст

string scomment = "Общая площадь шриховок";

ac\_request("interface", "calc\_field", "set", "a", 1, svalue, scomment);

// "a" - здесь имя ячейки у LabPP\_Calc куда записать значение. Всего ячеек - a,b,c и main

}

**FAQ (часто задаваемые вопросы)**

**Руководство по LABPP\_Automat для ARCHICAD**

Для просмотра результатов пользовательских параметров элементов, заданных формулами используйте [LabPP Debugger](#_topic_BIMMaster).