Руководство по LABPP_Automat для ARCHICAD

Copyright © 2009-2021 by Jurii Tsepov LABPP. All Rights Reserved.

Table of contents

Введение 14
Особенности языка 14
Моя первая программа 15
Кодировка у файлов программных модулей 16
Структура каталогов у LabPP_Automat 16
Конфигурационный файл config.cpp у конфигурации 18
create_button 18
create_iconbutton 19
set_palette_size_and_message_place 19
Списки для выборки элементов из проекта 20
Семантика языка C++ для ARCHICAD 20
Общая структура программы 20
Функция main() 20
Идентификаторы 21
Типы переменных 21
Пакетное объявление и обращение к переменным через идентификаторы, задаваемые текстом - var_by_txt() 21
init 21
is_exist 24
set 24
get 24
Комментарии 24
Организация циклов 24
Цикл for 24
Цикл do-while 25
Цикл while 25
Логические операции 25
Условные переходы 25
Оператор if 25
Конструкция if-else 25
Конструкция if-else if-else 25
Оператор switch 26
Создание собственных функций (подпрограмм) 26
Директива #include 27
Директива #pragma region 27
Поиск по шаблону (коды REGEX) 27
Функции для работы с оболочкой shell_func 28
get_path 28
set_cur_dir 28
shellexecute 29
ExecCmd 29
file 30
copy 30
Буффер обмена Windows 31
clipboard_get 31
clipboard_set 31
Функции автоматизации – Automate 31

CloseCurWindow 31
Zoom 31
GoToView 31
ChangeWindow 31
ShowSelectionIn3D 32
ShowAllIn3D 32
ZoomToElements 32
Redraw 32
Rebuild 32
Функции окружения - Environment 32
AddLibraries 32
Story_GoTo 32
Вывод в окно сообщений 33
cout 33
coutvar 33
setcfg("COUT",...) 33
Файловые операции 33
open 33
write 33
read 34
close 34
Строковые функции 34
strcmp 34
tolower 34
toupper 34
alltrim 35
strcontains 35
strreplace 35
strlen 35
strposcharset 35
strposcharget 35
strleft 35
strright 35
str_regex_match 36
straddslashtoregexcodes 36
str2threekeys 36
strreplace_escape_u 36
strisnumber 36
smallnums 36
Объект ac_element_guid (ts_guid) 37
Объект ac_element 37
ac_request 37
get_floor_name_by_floor_index 37
get_floor_index_by_floor_name 37
store_cur_element_to_descr 37
set_current_element_from_descr 37
set_element_by_guidstr_as_current 37
get_guid_from_element 37
load_element_from_guid 37
load_elements_list 38

set_current_element_from_list 38
add_elements_list 39
load_elements_list_from_selection и add_elements_list_from_selection 39
load_elements_list_curdb 39
clear_list 39
store_current_element_to_list 39
get_loaded_elements_list_count 39
select_elements_from_list 39
Слой - layer 39
  create 39
  get_index 39
  set_layer_visible 39
get_element_overall_dimensions 40
get_quantity_value 40
get_layer_by_substring 45
elem_user_property 46
  get 46
  set 47
  create 47
  Property_Import 47
get_object_property_value 48
set_object_property_value 48
set_object_property_value_curdb 49
get_element_value 49
set_element_value 50
assign_element_values 50
load_element_default_values 50
set_element_infoidtext 50
create_element_on_project 50
get_element_infoidtext 50
project_property 51
autotext 51
  get 51
  set 51
  create 51
  get_as_table 52
interface_input2point 52
interface_input2dline 52
interface_input3dline 53
interface_input2dpoly 53
get_levelfromprojectnull_by_floorindexandlevel 53
get_floorindexandlevel_by_levelfromprojectnull 55
elem_classification_get_item_value 55
elem_classification_get_parent_item_value 55
hotlinks 56
  GetHotlinkNodes 56
geometry_calc_2d 58
  is_point_on_element_polygon 58
  is_curelem_inside_element_polygon 59
  is_point_on_element_refside 59

where_C_leftorright 59
getDistPointFrom2dLine 59
geometry_calc_3d 59
RotatePoint3dAroundVect3d 59
classification 59
find_item_by_name 60
check_class_is_inbranch 60
CreateClassificationSystem 60
CreateClassificationItem 60
set_item_to_curelem 61
GetClassificationSystems 61
Classification_Import 61
do_move_rot_and_scale_elemlist 62
ac_request_special 63
get_pie 63
load_elements_list 64
add_elements_list 65
load_elements_list_from_selection 65
add_elements_list_from_selection 65
load_elements_list_curdb 65
copy_elements_to_list_from_list_by_filter 65
check_cur_element_by_filter 65
GetMaterialSurfaceSquare 65
GetMaterialSurfaceSquareTable 66
get_element_value 67
set_element_value 67
set_element_value_curdb 67
RunGDLParScript 67
get_building_material 67
get_build_material_value 67
get_material 68
load_zonerelatedwallstolist 68
load_zonerelatedwallparttotable 69
Attribute_GetNum 69
clipboard_set 69
clipboard_get 69
element_user_data 70
read 70
write 70
write_to_file 70
clear 71
delete_variable 71
get_variable_value 71
set_variable_value 72
linkingElems 72
uplinkBiWardByFlags 72
getLinkedElemsByFlags 72
set_TSMAX_ELEMLIST_NUMBER 73
get_TSMAX_ELEMLIST_NUMBER 73
ac_typeidfromstring 73

ac_getresvaluetype 73
ac_getstrvalue 73
ac_getnumvalue 74
Гравитация на поверхность 74
do_elements_landing - приземление элементов 74
do_surface_landing - приземление по точкам 74
do_point_landing - приземление точки X,Y 74
Диалоги 75
Сообщения или выбор варианта - tsalert 75
Поиск и выбор файла - dialog_get_filename 75
editdoubledialog - ввод числа 76
editstringdialog - ввод строки 76
Работа с Excel 76
excel_attach 76
excel_detach 77
excel_putnumvalue 77
excel_putstrvalue 77
excel_select_range 77
excel_visible 77
excel_speedup 77
excel_getnumvalue 77
excel_getstrvalue 78
excel_request 78
set_column_width 78
get_column_width 78
set_row_height 78
get_row_height 78
set_borders 78
put_selection_values 79
put_selection_fontvalues 79
get_selection_area 79
merged_cell_info 80
is_merge_cells 80
set_backcolor 80
get_backcolor 80
set_interior 80
get_interior 80
selection_varvalues 80
selection_font_varvalues 80
sheet_select 80
range_copy 80
booknamedcell 80
работа с примечаниями 80
getcomment 80
clearcomments 80
setcomment 80
Загрузка и выгрузка табличных данных 81
excel_chart - диаграммы в Excel 81
AddChart 81
SetChartType 81

SetChartTitle 83
SetSourceDataRange 83
Пример 83
Работа с Word 83
word_attach 83
word_detach 83
word_visible 83
word_request 84
NCчитывание и запись полей переменных - docfield 84
get 84
set 84
update_all_docfields 84
Работа с AutoCAD 84
autocad_request 84
attach 84
detach 85
specify_color_RGB 85
draw_text 85
draw_polyline 85
draw_line 85
draw_arc 86
Пример - нарисовать контуры зон из ARCHICAD в AutoCAD полилиниями 86
Функция object 88
create 88
delete 88
Объект ts_table 89
Функция ts_table 89
init (clear) 89
clear_rows 89
add_column 89
set_first_key 89
add_row 89
add_row_sum 90
sort 90
sort_back 91
search 91
search_regex 91
search_extended 91
search_extended_table 91
search_bounds 92
select_row 92
add_row_from_eq_table 92
add_rows_from_eq_table_by_filter 93
add_rows_from_eq_table_by_regex_filter 93
add_rows_from_eq_table_by_extended_filter 93
add_rows_from_eq_table_by_extended_filter_table 94
get_value_of 94
get_rows_count 94
get_columns_count 94
set_columnstypes 95

get_heading_of 95
print_to_str 95
set_columns_to_add_sum 95
Функции для выгрузки и загрузки таблиц 96
export_to_excel 96
import_from_excel 97
import_columns_from_excel 97
import_columns_from_table 97
import_columns_from_dialogcontrol 97
import_from_dialogcontrol 97
export_to_dialogcontrol 97
export_to_dialogcontrol_by_filter 98
export_to_dialogcontrol_by_regex_filter 98
export_to_dialogcontrol_by_extended_filter 98
export_to_dialogcontrol_by_extended_filter_table 99
resetofffromexport 99
export_to_LABPP_Table 99
set_columns_width_when_export 101
mapping_columns_to_export 101
export_to_csv 102
import_from_csv 102
Функции стандартной библиотеки 103
Преобразование 103
atoi 103
itoa 103
atof 103
ctos 103
grad_to_radian 103
radian_to_grad 103
sprintf 103
ecvt_french 104
ecvt 104
Математические функции 104
abs 104
max 104
min 104
rand 105
ln 105
log 105
sqrt 105
sqr 105
pow 105
percent 105
tsround 105
tsround_best 105
floor 105
ceil 105
Тригонометрические функции 106
cos 106
sin 106

arcsin 106
arccos 106
tg 106
arctg 106
ctg 106
arcctg 106
Функции геометрического преобразования 106
is_curelem_inside_element_polygon 106
is_point_on_element_polygon 106
rotate_point_and_move 107
get_cross_point_of_2lines 108
get_rot_and_move_point 108
get_length_2point 108
is_point_on_line 108
get_line_angle_relative_to_center 108
Специальные функции 108
codemeter - измерение времени выполнения фрагмента кода 108
ac_save_messages_to_file - сохранить текст из окна сообщений в файл 109
Связь с квартирографией 109
solaris_test 109
get_flat_rooms 109
runtimecontrol 109
workline 109
Команды для управления в приложении LabPP_Calc 110
interface 110
calc_field 110
Внешние переменные 110
var_extern_set 110
var_extern_get 110
run_cpp - запуск другой программы 111
run_from_file 111
run_from_variable 111
Получение аргументов внутри программы 111
Обработка ошибок 112
Коды ошибок 112
Функции интерфейса LabPP_Automat 112
create_iconbutton 112
create_button 112
set_palette_size_and_message_place 113
Диалоги на основе окон 113
Класс ts_dialog 113
init_dialog 113
set_as_main_panel 113
SetClientWH 113
SetTitle 113
SendCloseRequest 114
PostCloseRequest 114
eventreaction 114
Invoke 114
set_sDescr 114

Класс ts_dialogcontrol 114
Button 115
SetText 115
GetText 115
SetFontSizecodeStyle 115
IconButton 115
TextEdit 115
SetText 115
GetText 116
SetFontSizecodeStyle 116
RealEdit 116
SetFontSizecodeStyle 116
CheckBox 116
SetText 117
GetText 117
SetCheck 117
GetCheck 117
SetFontSizecodeStyle 117
LeftText, CenterText, RightText 117
SetText 117
GetText 118
SetFontSizecodeStyle 118
PopUp 118
SelectItem 118
DisableItem 118
DeleteItem 118
EnableItem 118
InsertItem 119
AppendItem 119
SetFontSizecodeStyle 119
RadioButton 119
IsSelected 119
Select 119
SetText 119
GetText 119
SetFontSizecodeStyle 120
IconRadioButton 120
IsSelected 120
Select 120
SingleSelListBox 120
RepaintBackgroundItemsByColumnValue 120
SetItemBackgroundColor 121
SetItemColor 121
SelectItem 121
GetTabItemText 121
GetMouseClickedPosXY 122
GetTabFieldPosition 122
SetTabItemIcon 122
SetTabItemText 122
GetTabItemIconId 122

SetTabItemIconId 122
SetOnTabItem 123
SetTabFieldCount 123
SetHeaderItemSize 123
SetTabFieldProperties 123
SetItemHeight 123
SetHeaderItemSizeableFlag 123
SetHeadersSizeableFlag 123
SetHeaderItemText 124
SetHeaderItemSize 124
InsertItem 124
AppendItem 124
DisableItem 124
EnableItem 124
DeleteItem 124
SetFontSizecodeStyle 124
MultiSellListBox 125
SelectItem 125
GetTabItemText 125
GetMouseClickedPosXY 125
GetTabFieldPosition 125
SetTabItemIcon 125
SetTabItemText 125
GetTabItemIconId 126
SetTabItemIconId 126
SetOnTabItem 126
SetTabFieldCount 126
SetTabFieldCount 126
SetHeaderItemSize 126
SetTabFieldProperties 126
SetItemHeight 127
SetHeaderItemSizeableFlag 127
SetHeadersSizeableFlag 127
SetHeaderItemText 127
SetHeaderItemSize 127
InsertItem 127
AppendItem 128
DisableItem 128
EnableItem 128
DeleteItem 128
SetFontSizecodeStyle 128
MultiLineEdit 128
SetText 129
GetText 129
SetFontSizecodeStyle 129
IconItem 129
IconCheckBox 129
SetCheck 129
GetCheck 129
PushCheck 129

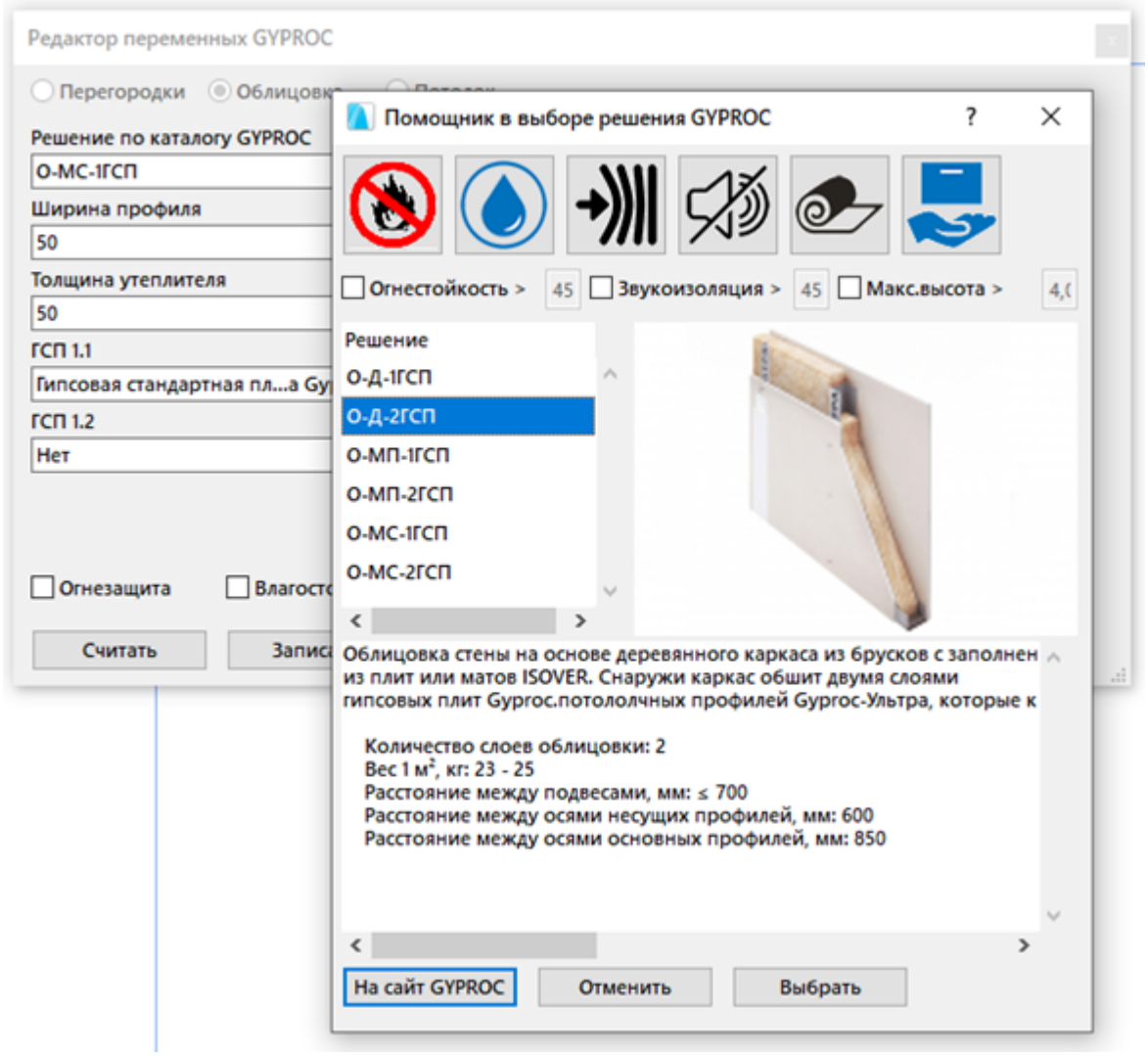
SetText 129
GetText 130
SetCkeck 130
GetCheck 130
SetFontSizecodeStyle 130
IconPushCheck 130
SetCheck 130
GetCheck 130
IconPushRadio 130
SingleSelTreeView 131
SelectItem 131
TreeViewInsertItem 131
DeleteItem 131
InsertItem 131
SetFontSizecodeStyle 131
MultiSelTreeView 132
SelectItem 132
TreeViewInsertItem 132
DeleteItem 132
InsertItem 132
AppendItem 132
SetFontSizecodeStyle 132
SingleSelListView 133
DeleteItem 133
InsertItem 133
AppendItem 133
SetFontSizecodeStyle 133
MultiSelListView 133
DeleteItem 133
InsertItem 134
AppendItem 134
SetFontSizecodeStyle 134
Splitter 134
Attach_controlled_element 134
ProgressBar 134
SetMin 134
SetMax 135
SetValue 135
GetMin 135
GetMax 135
GetValue 135
NormalTab 135
SelectItem 135
DisableItem 135
EnableItem 135
DeleteItem 135
InsertItem 136
AppendItem 136
SetFontSizecodeStyle 136
TabPage 136

SetToolTip 136
GroupBox 136
SetFontSizecodeStyle 136
Eventreaction 137
SetAnchorToPanelResize 139
Hide 139
Show 139
SetPosition 139
Работа со структурами данных в формате JSON 139
Класс ts_json 139
init (clear) 139
DecodeFromURL 140
DecodeFromString 140
EncodeToStr 140
Класс ts_array 141
Класс ts_class 141
Класс ts_num 141
Класс ts_bool 141
Класс ts_string 141
ReadFromURL 141
TSVariator 141
Редактор таблиц LABPP_Table для ARCHICAD 143
Структура данных объекта LABPP_Table (для программистов) 147
Непереносимые пробелы и дефисы 148
Отладчик с браузером данных элементов 149
Уроки 151
Вычислить середину высоты элемента Морф относительно 0 проекта и записать в поле LabPP_Calc 151
Заполнить таблицу с расчетом плитки по элементам Штриховка и записать общую площадь в поле LabPP_Calc 152
FAQ (часто задаваемые вопросы) 154
Как посмотреть значения параметров, заданных формулами 154

Введение

Вы можете легко создавать собственные текстовые программы для автоматизации различных проектных работ в ARCHICAD – расчеты, геометрия, интерфейс, отчеты, взаимодействие с электронными таблицами, манипуляция с 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.
Эффективность применения проверена на многих проектных организациях с 2012-го года.

Вид диалога в одном из приложений, созданном 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.

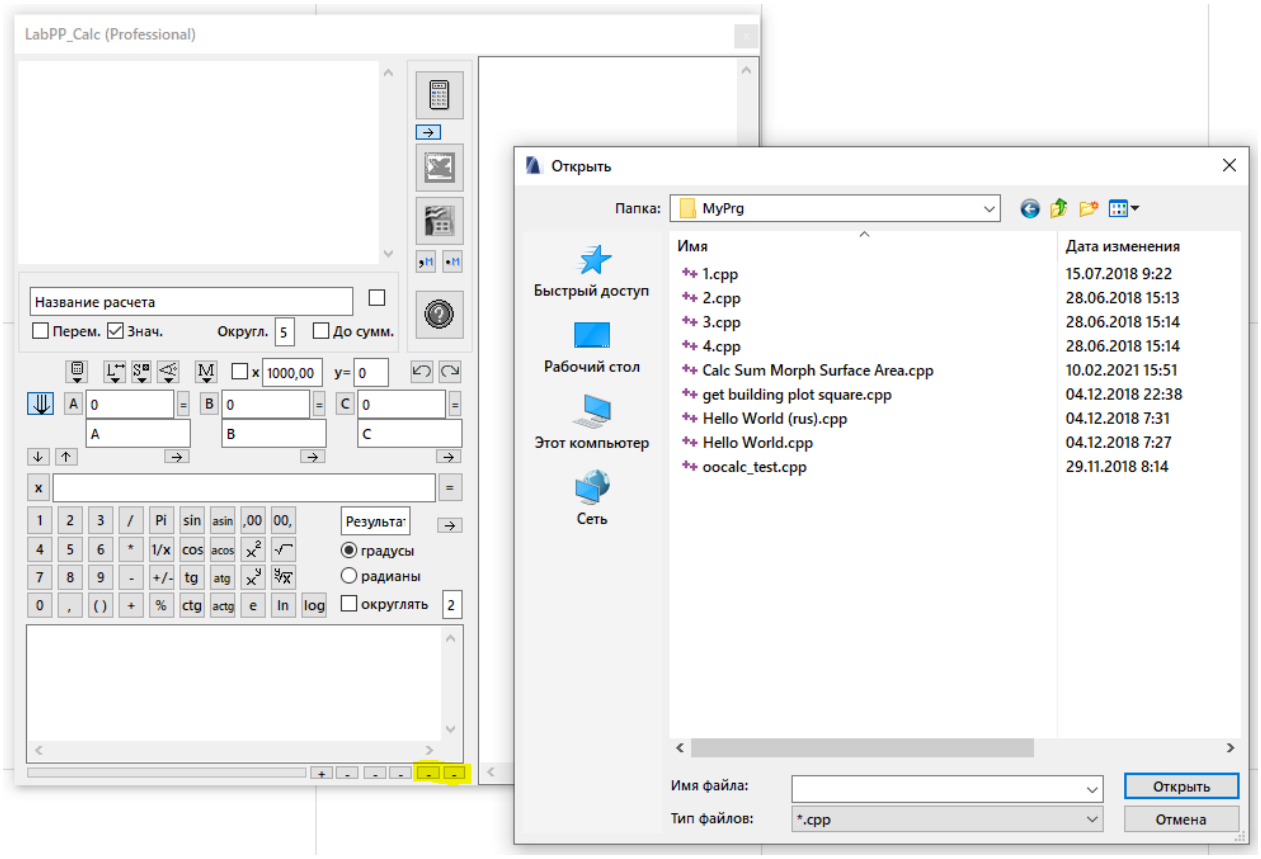
Моя первая программа

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.

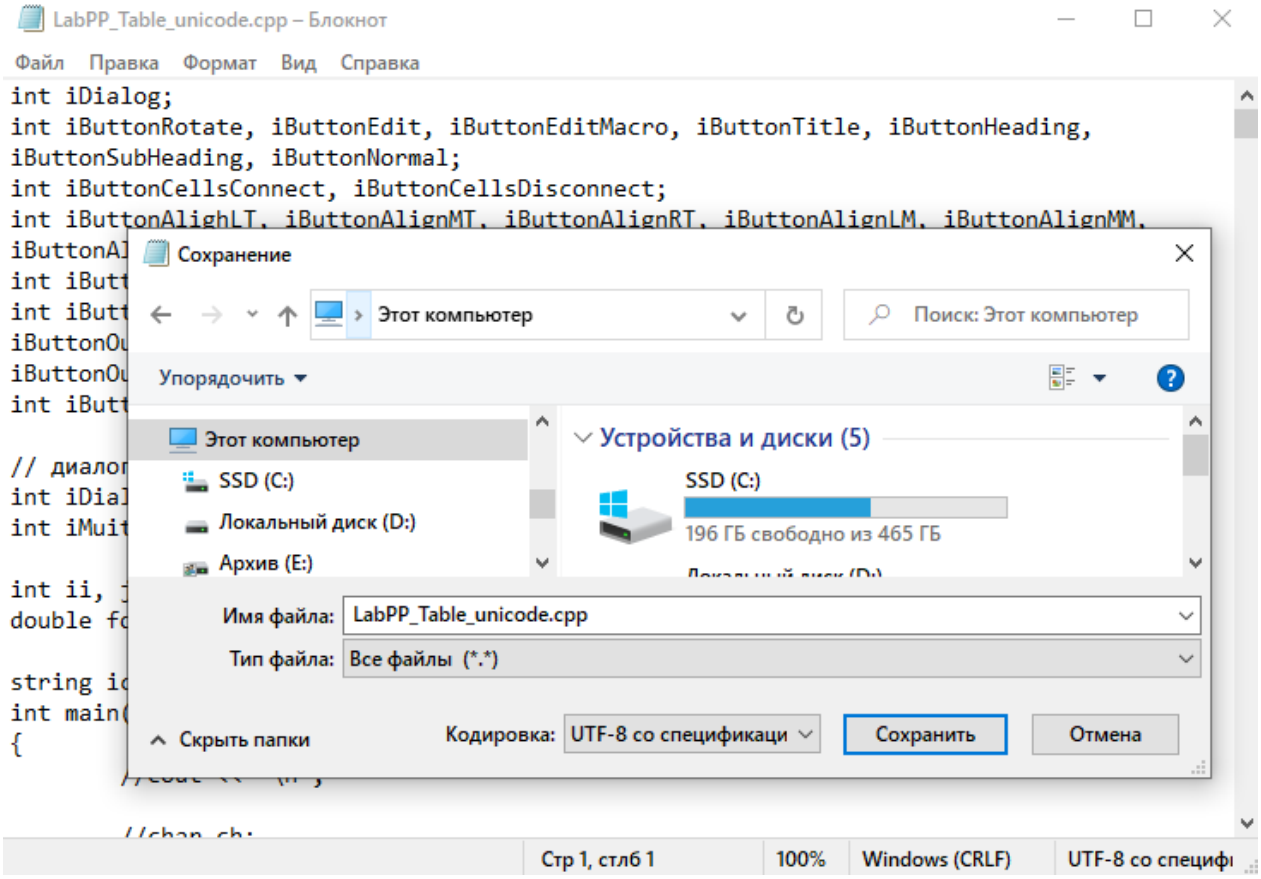
На картинке ниже - способ запуска программных модулей из приложения LabPP_Calc



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).

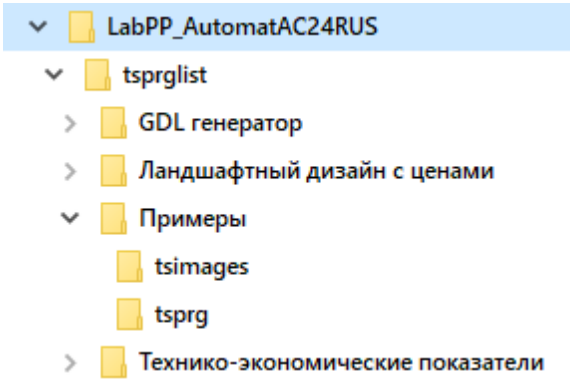
Кодировка у файлов программных модулей

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 "M2" 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.

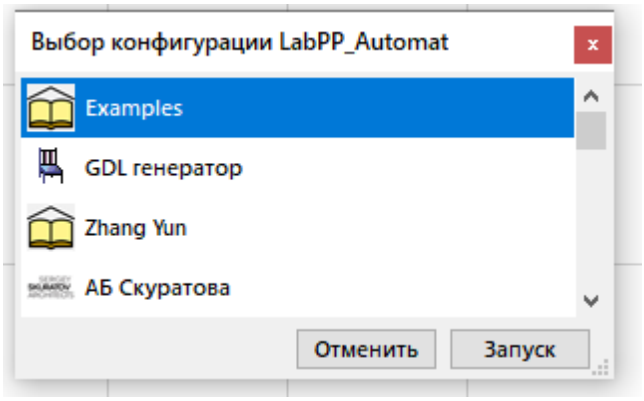


Структура каталогов у LabPP_Automat

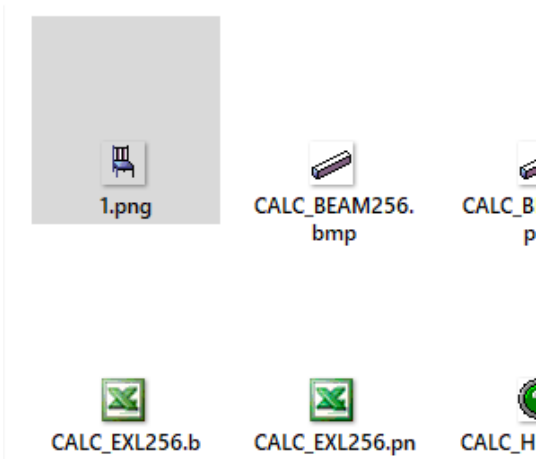
Тексты программных модулей могут использоваться самостоятельно или в составе конфигураций для LabPP_Automat.
Каждая конфигурация для LabPP_Automat представляет собой каталог с определенной структурой.
Именем конфигурации служит название этого каталога.
Каталоги конфигураций располагаются в каталоге tsprglist.
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.

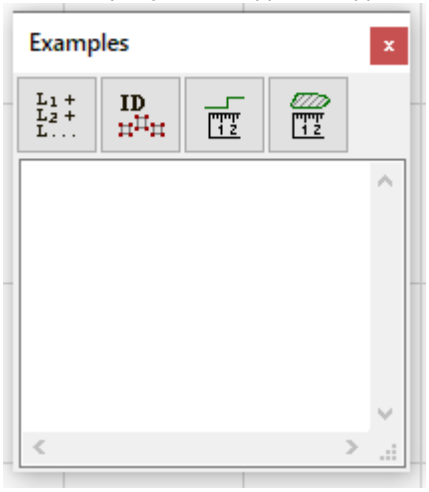


В каталоге tsprg располагается конфигурационный файл config.cpp и программные модули для запуска в данной конфигурации.
Например:

- config.cpp
- get_points_array.cpp
- make_extrude_code.cpp
- make_revolve_code.cpp
- make_ruled_code.cpp
- make_sweep_code.cpp
- make_tube_code.cpp
- multiple_around_center.cpp

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.
Обращение:

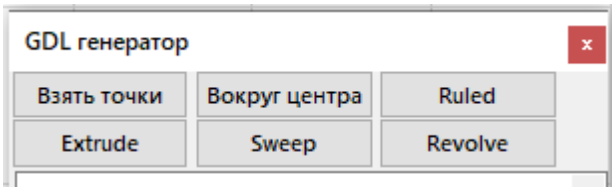
```
ac_request("create_button", string button_caption, int start_x, int start_y, int width,
int height, string tooltip, string name_cpp);

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"... :`

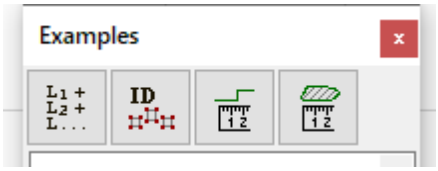
```
int iArg1;
double dArg2;
string sArg3;
run_cpp("get_args", iArg1, dArg2, sArg3);
```



create_iconbutton

Создать кнопку с иконкой на панели диалога конфигурации.
Обращение:
`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"`
.

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

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.
Обращение:

```
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.
Обращение:
`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.
Обращение:
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 C++ 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.

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 - ".*[СсСс]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
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, "", "");
```

Функции для работы с оболочкой shell_func

get_path

Получить каталог или путь к файлу.
Обращение:
`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

Обращение:
`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.

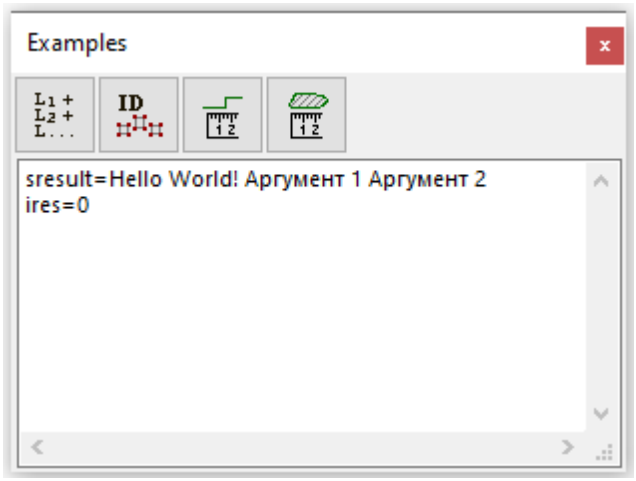
ExecCmd

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.
Обращение:
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.

Буффер обмена Windows

clipboard_get

Получить скопированный фрагмент текста из буффера Windows в строковую переменную.
Обращение:

```
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
Обращение:
int ires = ac_request("Environment","Story_GoTo", int storyindex);

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.
Обращение:
```
coutvar << my_variable;
```

setcfg("COUT",...)

Включить или выключить вывод сообщений.
Обращение:
```
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";
```

Строковые функции

strcmp

Сравнение двух строк
Обращение
`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
Обращение
`string sresult = tolower(string svalue);`

toupper

Перевести все символы строки в верхний регистр
Обращение
`string sresult = toupper(string svalue);`

alltrim

Удалить с начала и с конца строки все пробелы
Обращение
```
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.
Обращение
```
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.
Обращение
```
Int ilen = strlen(string s);
```
Здесь:
s - строка

strposcharset

strposcharget

Получить символ в строке. если строка короче - вернуть -1
```
int strposcharget(string sourcestr, int ipos, char &charres);
```

strleft

Получить левую часть строки разделенной фрагментом
```
string strleft(string str, string substr, bool isfromlast=false);
```
Если указано isfromlast и оно true - то при наличии нескольких фрагментов считать разделителем последний фрагмент.

strright

Получить правую часть строки разделенной фрагментом
```
string strright(string str, string substr, bool isfromlast=false);
```
Если указано isfromlast и оно true - то при наличии нескольких фрагментов считать разделителем последний фрагмент.
Сам разделяющий фрагмент в результат не записывается

str_regex_match

```
bool str_regex_match(string source, string code);
```
Здесь:
source - строка, в которой производится поиск,
code - регулярное выражение.

(см regex шаблоны)

straddslashtoregexcodes

заменить в строке . на \. , [на \[, (на \(и т.д. для точного сравнения;
```
string straddslashtoregexcodes(string source);
```

str2threekeys

Выделить из строки три ключа для сортировки в таблицах (для 115.2 или 14a.1 и т.п.)
Обращение:
```
int str2threekeys(string sSource, double& k1,double &k2, double &k3);
```
возвращает кол-во ключей
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.

Руководство по LABPP_Automat для ARCHICAD

Заменить escape-последовательности типа "\u0420\u0435" на символы.
Обращение:
```
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).

Если в строковой переменной записано число, то возвращает 1, в проитвном случае возвращает 0.
Обращение
```
string sresult = smallnums(string str);
```
Здесь:
str - строковая переменная с числом.

Пример.
Записать в строковую переменную значение "м²".

```
string str = "Объем, м" + smallnums("2");
```

Объект ac_element_guid (ts_guid)

Объект ac_element

ac_request

get_floor_name_by_floor_index

get_floor_index_by_floor_name

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.
Формат запроса:
```
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.
Обращение:
```
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
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
ac_request("store_current_element_to_list",int listnum, int index);
```
Если индекс = -1 то добавить в конец списка

get_loaded_elements_list_count

Определить количество элементов в указанном списке.
Обращение:
```
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.
Обращение:
```
ac_request("select_elements_from_list", int iListNum);
```
Здесь - iListNum - номер списка,

Слой - layer

create

Создать слой ARCHICAD.

get_index

Получить индекс слоя в проекте ARCHICAD.

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.
Обращение:
`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
Обращение:
`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:
volumecond - 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;
condvolume - 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:

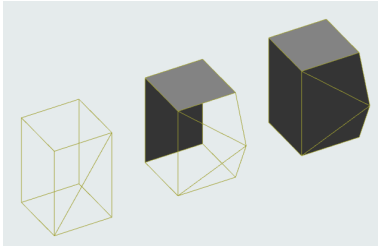
length - длина сплайна

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.

Обращение:
```
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.
Обращение:
```
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.
Обращение:
`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.
Обращение:
`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.
Обращение:
```
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.
Обращение:
```
int iret = ac_request("set_object_property_value",string parametername, string/double
valuetowrite);
```

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.
Обращение:
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","StoreIndex");
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.

StoreIndex

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.
Обращение:

```
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.
Обращение:

```
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.
Обращение:

```
int ires = ac_request("get_element_infoidtext");
```

Данные получаются следующей функцией ac_getstrvalue();
ires - 0 если считывание прошло успешно.

project_property

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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:

```
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.
Обращение:

```
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.
Обращение:
```
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.
Обращение:
```
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 ielemstoreindex;
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", "StoreIndex"); // запрашиваем индекс этажа
		ielemstoreindex = ac_getnumvalue(); // получаем индекс этажа в переменную
		// получаем высоту центра Морф относительно 0 проекта в переменную
elemprojectlevel
		ac_request("get_levelfromprojectnull_by_floorindexandlevel",
ielemstoreindex, 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 ielemstoreindex;
```

```
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", "StoreIndex"); // запрашиваем индекс этажа
	ielemstoreindex = ac_getnumvalue(); // получаем индекс этажа в переменную
	// получаем высоту центра Морф относительно 0 проекта в переменную
elemprojectlevel
	ac_request("get_levelfromprojectnull_by_floorindexandlevel", ielemstoreindex,
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.
Обращение:
Здесь:
`ac_request("get_floorindexandlevel_by_levelfromprojectnull", double levelfromprojectnull, int &floorInd, double &bottomOffset);`
levelfromprojectnull - смещение по вертикали относительно ноля проекта.
floorind - результат - индекс этажа,
bottomOffset - результат - смещение относительно уровня этого этажа,

elem_classification_get_item_value

Получить имя класса элемента в заданной классификации.
Обращение:
`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.
Обращение:
`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:

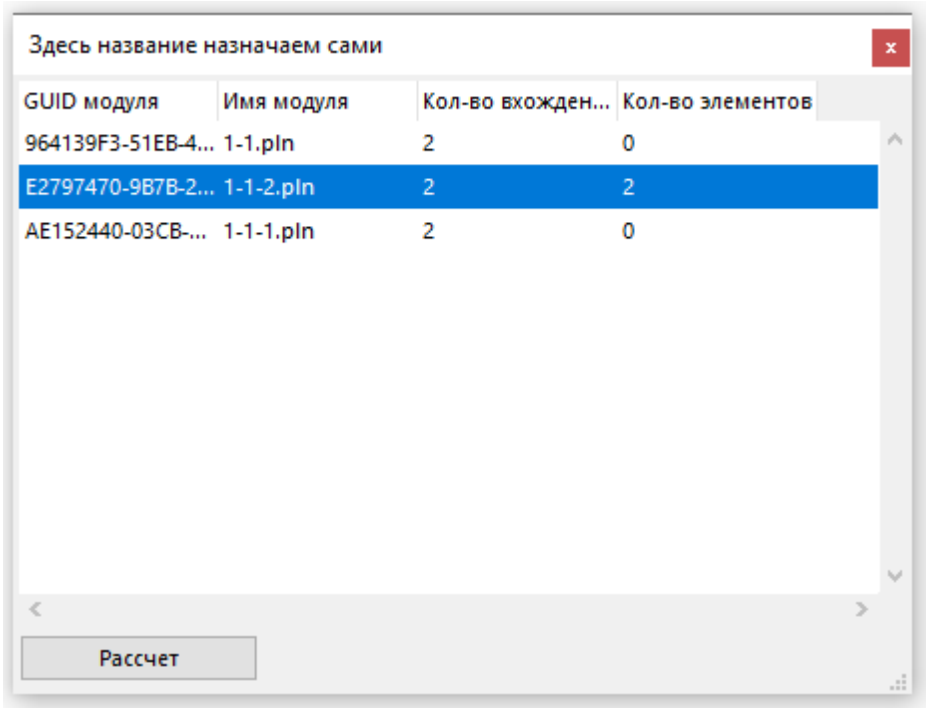
```
int err = ac_request("hotlinks","GetHotlinkNodes",int iTable);
```

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

```
err = ac_request("geometry_calc_2d","is_point_on_element_polygon",double X, double Y, int
iElemDescr);
```

или

```
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

```
ac_request("geometry_calc_2d","is_curelem_inside_element_polygon", int iElemDescr[, bool mainContOnly, double grow_contour, double grow_holes, int mode]); // дескриптор элемента (! не guid)
```

Руководство по LABPP_Automat для ARCHICAD

С какой стороны элемента находится заданная точка.
```
refSide - внешняя сторона стены или элемента, oppSide - внутренняя сторона стены или элемента
ac_request("is_point_on_element_refside",px,py,int elemguid, bool &bIsOnRefSide)
```

where_C_leftorright

Определить с какой стороны вектора находится точка - справа или слева.
Пример - если вектор снизу вверх, то если точка слева - то возвращает значение < 0, если точка на векторе - то 0, если точка справа - то >0.
```
ac_request("geometry_calc_2d","where_C_leftorright",double X1, double Y1,double X2, double Y2, double pX, double pY, double dist);
```

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);
```

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.

classification

Получить имя родительского класса элемента в заданной классификации.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
int err = ac_request("classification","CreateClassificationItem", int iguiddescritem, string id, string name, string description, int isystemguid, int iparentitemguid, int inextitemguid)
```

Здесь:
err = 0 если создано без ошибок,
iguiddescritem - дескриптор объекта ts_guid куда будет записан guid создаваемого класса,
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,

iparentitemguid - дескриптор guid'a родительского класса, к которому хотим создать подкласс (может быть пустым),
inextitemguid - дескриптор элемента, за которым поставить класс (может быть пустым).

set_item_to_curelem

Присвоить класс текущему элементу.
Обращение:
```
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.
Обращение:
```
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);
```

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- Объекты 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();
}
```

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, coordRotCenterY - 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);
```

ac_request_special

get_pie

Считать атрибуты пирога слоев материалов в таблицу 1, и слои в таблицу 2.
Обращение:
`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
Обращение:
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).

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.
Обращение:

```
int ires = ac_request_special("GetMaterialSurfaceSquare", string materialname);
```

или

```
int ires = ac_request_special("GetMaterialSurfaceSquare", int materialindex);
```

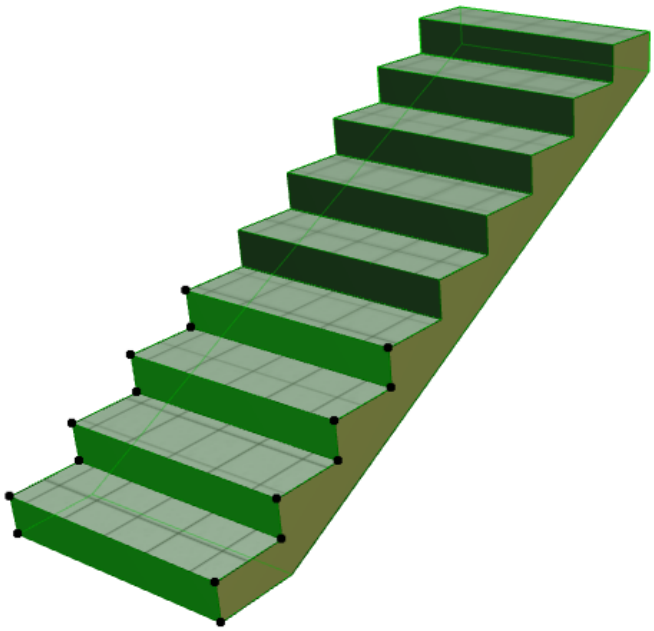
Здесь:

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.
Обращение:
`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.

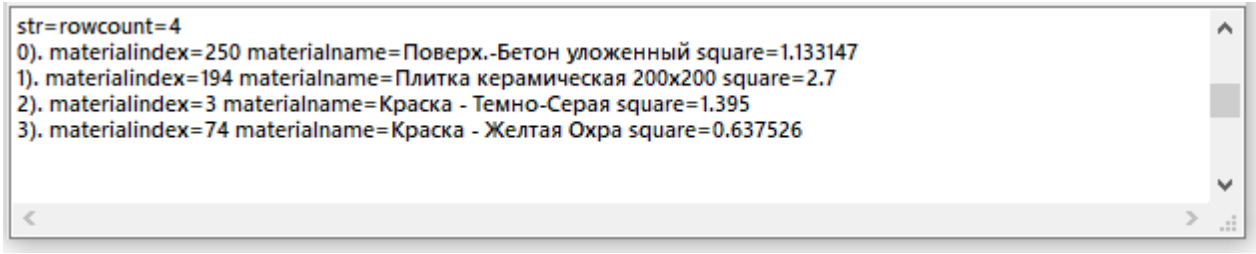
```
int main()
{
   ac_request_special("load_elements_list_from_selection", 1, "MorphType", 0);
   ac_request("get_loaded_elements_list_count",1);
   int icount = ac_getnumvalue();
   if (icount == 0)
   {
      cout << "Сначала выберите элемента типа Morph";
```

```
        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.

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.
Обращение:
`int ires = ac_request_special("RunGDLParScript");`

get_building_material

Получить информацию о стройматериале по его индексу в проекте, или все стройматериалы проекта.
Обращение:
`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.
Обращение:

```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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).
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
int ires = ac_request_special(""element_user_data","read");
```

write

Записать в текущий элемент все выполненные изменения с переменными.
Обращение:
```
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 - возвращается -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.
Обращение:
`int ires = ac_request_special("element_user_data","clear");`

delete_variable

Удалить собственную переменную у текущего элемента.
Обращение:
`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.
Обращение:

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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
`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.
Обращение:
`ac_request_special("get_TSMAX_ELEMLIST_NUMBER", int &icurrent_lists_count);`
Здесь:
icurrent_lists_count - текущее количество внутренних списков элементов.

ac_typeidfromstring

Получить код типа элемента по его названию.
Обращение.
`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.
Обращение
`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.
Обращение
`string sresult = ac_getstrvalue();`

ac_getnumvalue

Возвращает числовое значение предыдущей операции.
Обращение
```
double dresult = ac_getnumvalue();
```

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.
Обращение
```
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.
Обращение:
```
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;

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 &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.
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, 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", "", 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).
Пример.
Получить от пользователя дистанцию между элементами. По умолчанию предложить значение 20.

```
int res = ac_request("editdoubledialog","Введите дистанцию между элементами (м)","20");
```

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).
Пример.
Получить от пользователя ФИО архитектора. По умолчанию предложить значение "Иванов В.В.".

```
int res = ac_request("editstringdialog","Введите ФИО архитектора", "Иванов В.В.");
string svalue = ac_getstrvalue();
```

Работа с Excel

excel_attach

Подключить Excel для работы.
В момент запуска программа Excel должна быть открыта.
По умолчанию активной становится текущая страница Excel.
Обращение

```
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.
Обращение
`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.
Обращение:
`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.
Обращение:
```
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
Обращение:

```
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
Обращение:
```
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.
Обращение:
```
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.
Обращение:

```
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

is_merge_cells

set_backcolor

get_backcolor

set_interior

get_interior

selection_varvalues

selection_font_varvalues

sheet_select

range_copy

booknamedcell

работа с примечаниями

getcomment

Получить примечания к текущей ячейке.
Обращение:
`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.
Обращение:
`excel_request("comment", "clear");`

setcomment

Задать примечания к текущей ячейке.
Обращение:
`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
Обращение:
`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.
Обращение:
`int ires = word_attach();`
Возвращает 0, если подключение произошло успешно.

word_detach

Отключение программы Word.
Обращение:
`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.
Обращение:
`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.

NCчитывание 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.
Обращение:
`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.
Обращение:
`word_request("update_all_docfields");`

Работа с AutoCAD

autocad_request

attach

Подключить программу AutoCAD.
Обращение:
`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.
Обращение:
`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.
Обращение:
`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.
Обращение:
`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.
Создать 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.
Обращение:
```
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.
Обращение:
```
ts_table(int iTable, "clear_rows");
```
Здесь:
iTable - дескриптор стираемой таблицы.

add_column

Добавить колонку в таблицу. Колонки могут быть строковыми или числовыми.
Формат команды:
```
ts_table(int descriptor, "add_column", int columnnumber, string columntype, string columnname);
```
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 чтобы добавить текстовую колонку №0:
ts_table(TableDescr1, "add_column",0,"string","код материала + ед.изм.");
Пример чтобы добавить числовую колонку с вычисляемым номером по значению переменной № = i + 3
ts_table(TableDescr1, "add_column",i+3,"double",szonename);

set_first_key

Когда нужно избежать дублирования записей в таблице можно использовать удобный механизм.
Одну из колонок можно задать в качестве так называемого "первичного ключа".
После этого все добавляемые строки будут анализироваться и если в таблице уже есть строка с таким же значением в колонке, 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.
```
ts_table(TableDescr1,"add_row", 0, objectname, 1, value_to_column_1, 2, value_to_column_2);
```

Здесь в колонку №0 записываем значение из переменной objectname, в колонку №1 записываем значение value_to_column_1, и в колонку №2 записываем значение переменной value_to_column_2. 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" ("100x40","50x40" 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 100x50	пог.м	1000
Доска 50x40	куб.м.	2000
Доска 50x20	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.
Обращение:
```
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. Сортировать таблицу по колонке №0
ts_table(TableDescr1,"sort",0);
Пример. Сортитовка таблицы по ключевой колонке, указанной ранее.
ts_table(TableDescr1,"sort");

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.
Обращение:
```
ts_table(int iTableDescr, "sort_back", int icolnum1, int icolnum2, ... ,int icolnum7);
```
или
```
ts_table(int iTableDescr, "sort_back", string scolname1, string scolname2, ... ,string scolname7);
```
Здесь:
icolumn1... icolumn7 - числовые индексы колонок, по которым производить сортировку.
scolname1... icolname7 - текстовые названия колонок.

Пример. Сортировать таблицу по колонке №0
ts_table(TableDescr1,"sort_back",0);
Пример. Сортитовка таблицы по ключевой колонке, указанной ранее.
ts_table(TableDescr1,"sort_back");

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 "B22" или "B22" (B - 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", ".*[BB]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 "B22" 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",
".*[BB]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.
Обращение:
```
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);
```

add_row_from_eq_table

Добавить строку в таблицу из аналогичной таблицы.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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 "B22" или "B22" (B - 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", ".*[BB]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 "B22" или "B22" (B - 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",
".*[BB]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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.

Обращение:
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.
Обращение:
`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".
Обращение:
```
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.
Обращение:
```
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.)
Обращение:
```
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.
Обращение:
```
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).

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 "B22" или "B22" (B - 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", ".*[BB]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 "B22" или "B22" (B - 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",
".*[BB]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.

Наименование таблицы

Моя первая колонка	Моя вторая колонка с числом	Примечания
Строка 1	10000,11	Примечание 1
Строка 2	20000,22	
Строка 3	30000,34	Примечание 3

set_columns_width_when_export

Задать размер колонок при экспорте в табличный элемент диалога.
Обращение:

```
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.

Обращение:
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", ""
```

```
	);
```

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 str2;
	ts_table(iTable, "export_to_csv", true, ";", "," , 0, -1, str2);
	coutvar << str2;

	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 ";".

Функции стандартной библиотеки

Преобразование

atoi

Перевод строковой переменной в целое число
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.

Обращение:
`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.
Обращение:
`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.
Обращение:
`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
Формат команды
`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
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
double dresult = ceil(double dvalue);
```

Тригонометрические функции

cos

Косинус угла. Угол задается в радианах.
Обращение:
```
double dresult = cos(double cornerinrad);
```

sin

Синус угла. Угол задается в радианах.
Обращение:
```
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.
Обращение:
```
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.
Обращение:

```
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.
Обращение:
```
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.
Обращение:
```
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_relative_to_center

```
ac_request("geometry_calc_2d","get_line_angle_relative_to_center",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.
Формат обращения:

```
runtimecontrol("workline", string sdirective, double dvalue);
```
Здесь:
directive - директива, задающая что сделать"setmin"/"setmax" /"setpos" - установить минимальное, максимальное или текущее значение дорожки процентов,
dvalue - число, в зависимости от директивы.

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.
Обращение:
```
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");

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.
Обращение
```
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
Обращение
```
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.

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
0	Успешное завершение операции
-1	Общая ошибка
-1222	Переменная отсутствует или недоступна
-1223	Неверный индекс
-2222	Не найден исходный файл
-2223	Файл уже существует
-2224	Устройство переполнено
-2225	Недостаточно прав пользователя

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.
Обращение:

```
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.
Обращение:

```
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 класса 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.
Обращение:
```
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.
Обращение:
```
ts_dialog(int iDialogDescr, "set_as_main_panel");
```

SetClientWH

Задает ширину и высоту окна диалога.
Обращение:
```
ts_dialog(int iDialogDescr, "SetClientWH", int width, int height);
```
Здесь:
width, height - ширина и высота окна диалога.

SetTitle

Задает заголовок для окна диалога.
Обращение:
ts_dialog(int iDialogDescr,"SetTitle", string sTitle);

Здесь:
sTitle - заголовок для диалога.

SendCloseRequest

Закрыть диалог на базе палитры, т.е. немодальный диалог.
Обращение:
ts_dialog("iDialogDescr, "SendCloseRequest");

PostCloseRequest

Закрыть модальный диалог.
Обращение:
```
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".
Обращение:
ts_dialog(iDialogDescr, "eventreaction", string eventfunctionname);
Здесь:
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.
Обращение:
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.
Обращение:
ts_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);
Здесь:
text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.
Обращение:
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.
Обращение:
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
```
Обращение:
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.
Обращение:
ts_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);

Здесь:
text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.
Обращение:
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
```
Обращение:
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.
```
Обращение:
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.
Обращение:
ts_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);
Здесь:
text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.
Обращение:
ts_diaolgcontrol(iDialogcontrolDescr, "GetText", string text);
Здесь:
text - текстовое значение, считанное из элемента управления.

SetCheck

Задать значение элементу управления.
Обращение:
ts_diaolgcontrol(iDialogcontrolDescr, "SetCheck", int onoff);
Здесь:
onoff - 0/1.

GetCheck

Получить значение из элемента управления.
Обращение:
ts_diaolgcontrol(iDialogcontrolDescr, "SetCheck", int &onoff);
Здесь:
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.
Обращение:
```
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.

Обращение:
ts_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);
Здесь:
text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.
Обращение:
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.

SelectItem

Выбрать значение из списка в элементе управления как текущее.
Обращение:
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.
Обращение:
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.
Обращение:
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.
Обращение:
ts_dialogcontrol(iDialogControlDescr, "EnableItem", int itemindex);

Здесь:
itemindex - позиция, которую сделать доступной (индекс от 1).

InsertItem

Вставить новый элемент в указанную позицию списка.
Обращение:
ts_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);
Здесь:
itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.
Обращение:
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.

RadioButton

Элемент управления - радиокнопки.

IsSelected

Проверить выбран ли этот элемент радио-кнопки или нет.
Обращение:
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.
Обращение:
ts_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);
Здесь:
text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.
Обращение:
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

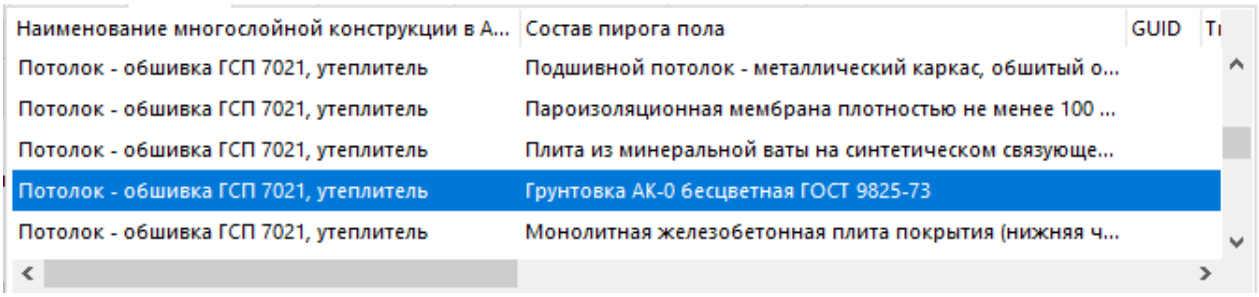
IsSelected

Проверить выбран ли этот элемент радио-кнопки или нет.
Обращение:
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");

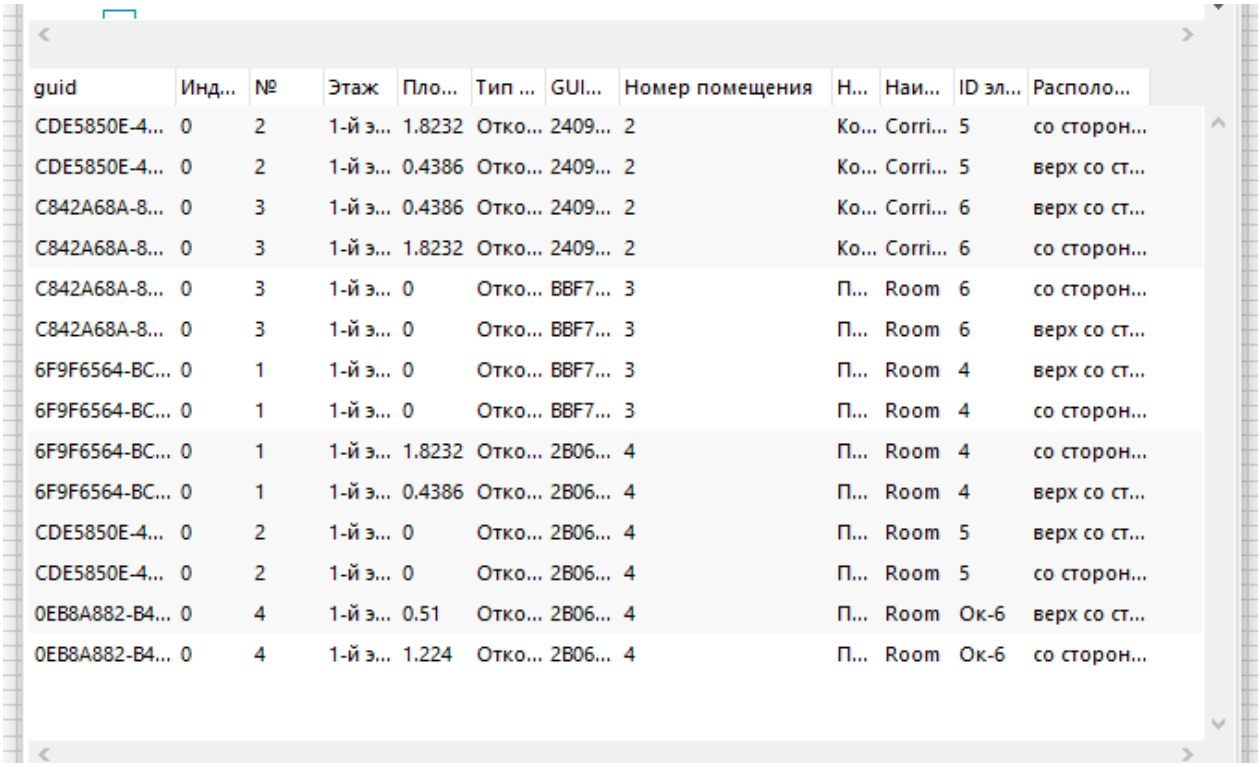
SingleSelListBox



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);
```



guid	Инд...	№	Этаж	Пло...	Тип ...	GUI...	Номер помещения	Н...	Наи...	ID эл...	Располо...
CDE5850E-4...	0	2	1-й э...	1.8232	Отко...	2409...	2	Ко...	Corri...	5	со сторон...
CDE5850E-4...	0	2	1-й э...	0.4386	Отко...	2409...	2	Ко...	Corri...	5	верх со ст...
C842A68A-8...	0	3	1-й э...	0.4386	Отко...	2409...	2	Ко...	Corri...	6	верх со ст...
C842A68A-8...	0	3	1-й э...	1.8232	Отко...	2409...	2	Ко...	Corri...	6	со сторон...
C842A68A-8...	0	3	1-й э...	0	Отко...	BBF7...	3	П...	Room	6	со сторон...
C842A68A-8...	0	3	1-й э...	0	Отко...	BBF7...	3	П...	Room	6	верх со ст...
6F9F6564-BC...	0	1	1-й э...	0	Отко...	BBF7...	3	П...	Room	4	верх со ст...
6F9F6564-BC...	0	1	1-й э...	0	Отко...	BBF7...	3	П...	Room	4	со сторон...
6F9F6564-BC...	0	1	1-й э...	1.8232	Отко...	2B06...	4	П...	Room	4	со сторон...
6F9F6564-BC...	0	1	1-й э...	0.4386	Отко...	2B06...	4	П...	Room	4	верх со ст...
CDE5850E-4...	0	2	1-й э...	0	Отко...	2B06...	4	П...	Room	5	верх со ст...
CDE5850E-4...	0	2	1-й э...	0	Отко...	2B06...	4	П...	Room	5	со сторон...
0EB8A882-B4...	0	4	1-й э...	0.51	Отко...	2B06...	4	П...	Room	Ок-6	верх со ст...
0EB8A882-B4...	0	4	1-й э...	1.224	Отко...	2B06...	4	П...	Room	Ок-6	со сторон...

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.
Обращение:
ts_dialogcontrol(iDialogcontrolDescr,"SelectItem", int itemindex);
Здесь:
Itemindex = индекс выбранного значения в списке элемента управления.

GetTabItemText

Получить значение ячейки из таблицы ListBox.
Обращение:
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iListBoxDescr, "SetOnTabItem", int count);
```
Здесь:
count - количество колонок.

SetHeaderItemSize

Задать высоту строки заголовков в таблице элемента управления.
Обращение:
```
ts_dialogcontrol(iListBoxDescr, "SetHeaderItemSize", int height);
```
Здесь:
height - высота строки заголовков в точках.

SetTabFieldProperties

Задать параметры колонки для строк данных в таблице элемента управления.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iListBox, "SetItemHeight", int itemheight);
```
Здесь:
itemheight - высота строки.

SetHeaderItemSizeableFlag

Задать параметры изменения размера и управляемости мышью для колонки таблицы
Обращение:
```
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
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);
```
Здесь:
itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.
Обращение:
```
ts_dialogcontrol(iDialogControlDescr, "AppendItem");
```

DisableItem

Сделать недоступной для редактирования и выбора позицию элемента управления.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iDialogControlDescr, "EnableItem", int itemindex);
```
Здесь:
itemindex - позиция, которую сделать доступной (индекс от 1)

DeleteItem

Удалить указанную позицию списка.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iDialogcontrolDescr,"SelectItem", int itemindex);
```
Здесь:
Itemindex = индекс выбранного значения в списке элемента управления.

GetTabItemText

Получить значение ячейки из таблицы ListBox.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.

Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iListBoxDescr, "SetOnTabItem", int count);
```
Здесь:
count - количество колонок.

SetTabFieldCount

Задать количество колонок в таблице в элементе управления.
Обращение:
```
ts_dialogcontrol(iListBoxDescr, "SetOnTabItem", int count);
```
Здесь:
count - количество колонок.

SetHeaderItemSize

Задать высоту строки заголовков в таблице элемента управления.
Обращение:
```
ts_dialogcontrol(iListBoxDescr, "SetHeaderItemSize", int height);
```
Здесь:
height - высота строки заголовков в точках.

SetTabFieldProperties

Задать параметры колонки для строк данных в таблице элемента управления.

Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iListBox, "SetItemHeight", int itemheight);
```
Здесь:
itemheight - высота строки.

SetHeaderItemSizeableFlag

Задать параметры изменения размера и управляемости мышью для колонки таблицы
Обращение:
```
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
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);
```
Здесь:
itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.
Обращение:
```
ts_dialogcontrol(iDialogControlDescr, "AppendItem");
```

DisableItem

Сделать недоступной для редактирования и выбора позицию элемента управления.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iDialogControlDescr, "EnableItem", int itemindex);
```
Здесь:
itemindex - позиция, которую сделать доступной (индекс от 1)

DeleteItem

Удалить указанную позицию списка.
Обращение:
```
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.
Обращение:
```
ts_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);
```
Здесь:
text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.
Обращение:
```
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.
Обращение:
```
ts_diaolgcontrol(iDialogcontrolDescr, "SetCheck", int onoff);
```
Здесь:
onoff - 0/1.

GetCheck

Получить значение из элемента управления.
Обращение:
```
ts_diaolgcontrol(iDialogcontrolDescr, "SetCheck", int &onoff);
```
Здесь:
onoff - возвращаемое значение - 0/1.

PushCheck

SetText

Присвоить текстовое значение элементу управления.
Обращение:
```
ts_diaolgcontrol(iDialogcontrolDescr, "SetText", string text);
```
Здесь:

text - текстовое значение.

GetText

Считать текстовое значение из элемента управления.
Обращение:
```
ts_diaolgcontrol(iDialogcontrolDescr, "GetText", string text);
```
Здесь:
text - текстовое значение, считанное из элемента управления.

SetCkeck

Задать значение элементу управления.
Обращение:
```
ts_diaolgcontrol(iDialogcontrolDescr, "SetCheck", int onoff);
```
Здесь:
onoff - 0/1.

GetCheck

Получить значение из элемента управления.
Обращение:
```
ts_diaolgcontrol(iDialogcontrolDescr, "SetCheck", int &onoff);
```
Здесь:
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

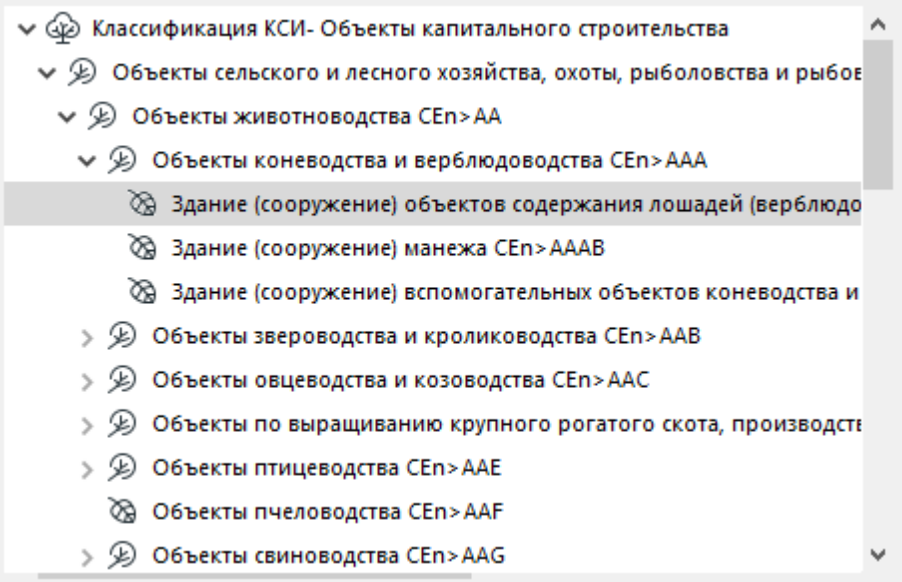
Задать значение элементу управления.
Обращение:
```
ts_diaolgcontrol(iDialogcontrolDescr, "SetCheck", int onoff);
```
Здесь:
onoff - 0/1.

GetCheck

Получить значение из элемента управления.
Обращение:
```
ts_diaolgcontrol(iDialogcontrolDescr, "SetCheck", int &onoff);
```
Здесь:
onoff - возвращаемое значение - 0/1.

IconPushRadio

SingleSelTreeView



SelectItem

Выбрать значение из списка в элементе управления как текущее.
Обращение:
```
ts_dialogcontrol(iDialogcontrolDescr,"SelectItem", int itemindex);
```
Здесь:
Itemindex = индекс выбранного значения в списке элемента управления.

TreeViewInsertItem

Добавить пункт в элемент управления TreeView
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
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.

MultiSelTreeView

SelectItem

Выбрать значение из списка в элементе управления как текущее.
Обращение:
```
ts_dialogcontrol(iDialogcontrolDescr,"SelectItem", int itemindex);
```
Здесь:
Itemindex = индекс выбранного значения в списке элемента управления.

TreeViewInsertItem

Добавить пункт в элемент управления TreeView
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);
```
Здесь:
itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.
Обращение:
```
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.

SingleSelListView

DeleteItem

Удалить указанную позицию списка.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);
```
Здесь:
itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.
Обращение:
```
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.

MultiSelListView

DeleteItem

Удалить указанную позицию списка.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);
```
Здесь:
itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.
Обращение:
```
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.

Splitter

Attach_controlled_element

Подключить к сплиттеру контролируемый элемент (который будет изменять свою форму или положение в зависимости от перемещения сплиттера).
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iProgressBar, "SetMin", int minvalue);
```
Здесь:
minvalue - минимальное значение, от которого считать процент выполнения.

SetMax

Задать максимальное значение.
Обращение:
```
ts_dialogcontrol(iProgressBar, "SetMax", int maxvalue);
```
Здесь:
maxvalue - максимальное значение, до которого считать процент выполнения.

SetValue

GetMin

Считать текущее значение параметра элемента.
Обращение:
```
ts_dialogcontrol(iProgressBar, "GetMin", int &value);
```
Здесь:
value - считанное значение.

GetMax

Считать текущее значение параметра элемента.
Обращение:
```
ts_dialogcontrol(iProgressBar, "GetMax", int &value);
```
Здесь:
value - считанное значение.

GetValue

Считать текущее значение параметра элемента.
Обращение:
```
ts_dialogcontrol(iProgressBar, "GetValue", int &value);
```
Здесь:
value - считанное значение.

NormalTab

SelectItem

Выбрать значение из списка в элементе управления как текущее.
Обращение:
```
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.
Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iDialogControlDescr, "EnableItem", int itemindex);
```
Здесь:
itemindex - позиция, которую сделать доступной (индекс от 1).

DeleteItem

Удалить указанную позицию списка.

Обращение:
```
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.
Обращение:
```
ts_dialogcontrol(iDialogControlDescr, "InsertItem", int itemindex);
```
Здесь:
itemindex - позиция, в которую вставить новый элемент.

AppendItem

Добавить позицию в конец списка элемента управления.
Обращение:
```
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.
Обращение:
```
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.

Eventreaction

Привязать к элементу управления функцию - обработчик события.
`ts_dialogcontrol(int iButtonDescr, "eventreaction", string eventfunctionname);`
Здесь:
eventfunctionname - имя функции, которая обрабатывает событие.
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,	Щелчок на тексте

Event_StaticTextDoubleClicked	RightText	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

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.
Обращение:
```
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
Обращение:
```
ts_dialogcontrol(iDialogcontrolDescr, "Hide");
```

Show

Показать скрытый элемент управления
Обращение:
```
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 управлениюю.

Работа со структурами данных в формате JSON

Класс ts_json

init (clear)

Очистить содержимое объекта ts_json.
Обращение:
```
ts_json(int iJSONObjectDescr,"clear");
```
Здесь:

iJSONObjectDescr - дескриптор объекта JSON, который нужно очистить.

DecodeFromURL

Загрузить JSON код из файла, расположенного на web сайте.
Обращение.
```
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.
Обращение.
```
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.
Обращение.
```
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" : ""
```

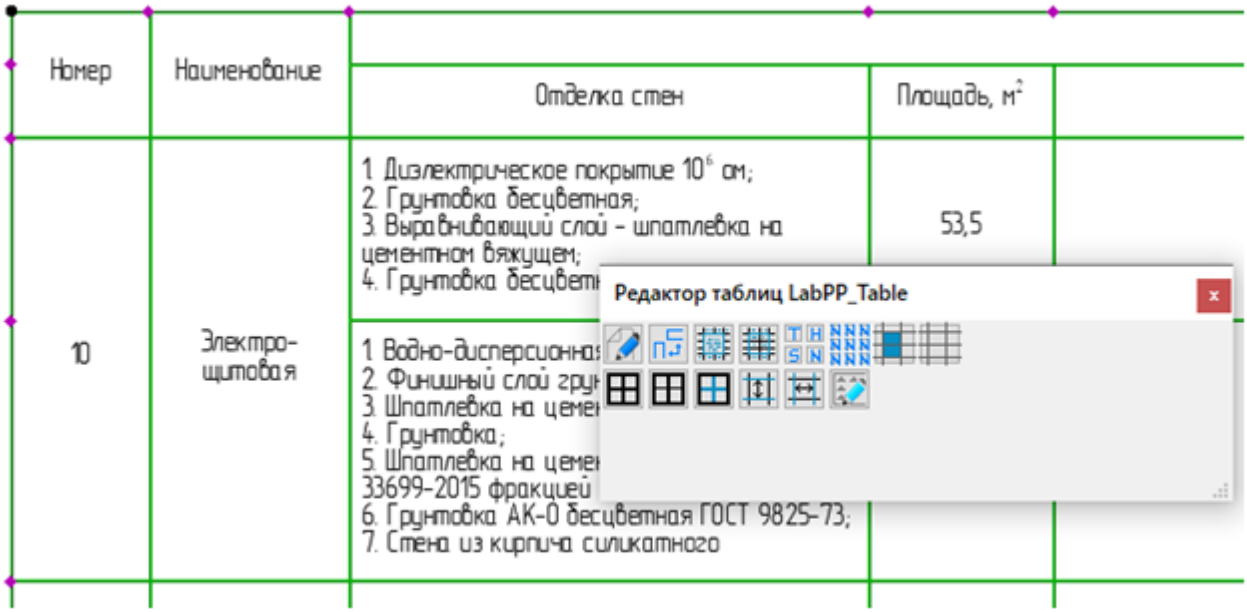
```
},
{
 "SourceVarValue": "Толщина утеплителя",
 "Result" : "50",
 "Comment" : ""
},
{
 "SourceVarValue": "Огнестойкость",
 "Result" : "Нет",
 "Comment" : "для отметки огнестойкости пп у элемента"
},
{
 "SourceVarValue": "Влагостойкость",
 "Result" : "Нет",
 "Comment" : "для отметки влагостойкости 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").

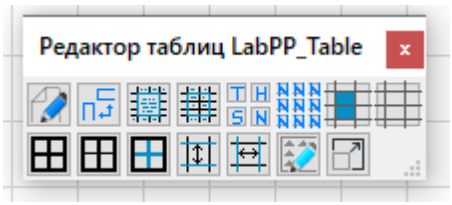
Пример.
Получить значение свойства "Огнестойкость" из текста в примере выше.

```
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();
cout << "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.
Управление этим элементом удобно осуществлять через редактор LABPP_Table_unicode.cpp (LABPP_Table_unicode.zip).
Этот редактор можно модифицировать самостоятельно. Добавлять собственные кнопки и процедуры и т.п.

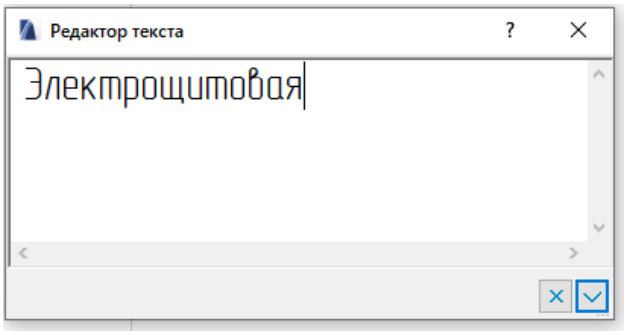


Основные возможности.
Максимальное число колонок - 100;
Максимальное число строк - 500;
Объединение ячеек - да;
Типов линий для отрисовки ячеек - 2 (перо для тонких линий, перо для толстых линий, без линии);
Регулировка размеров ячеек - да;
Групповое изменение высоты - да;
Групповое изменение ширины - 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.

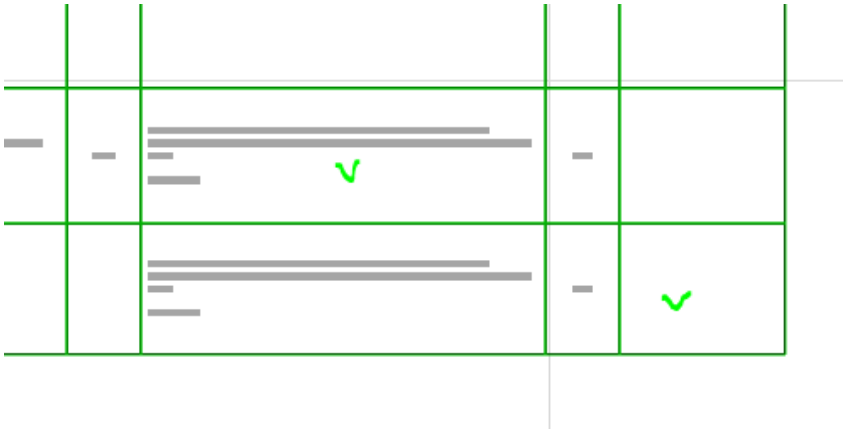
Кнопки слева направо верхний ряд.

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.

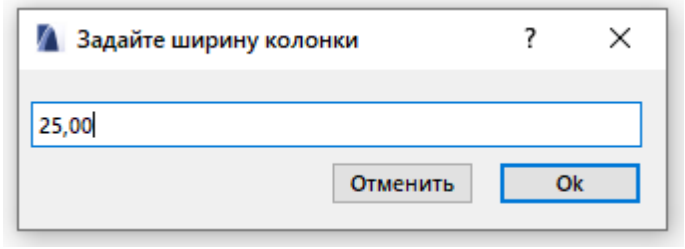


Разъединение ячеек.
Щелкните на объединенную ячейку.
Ячейки, ее составляющие, разъединятся.

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.

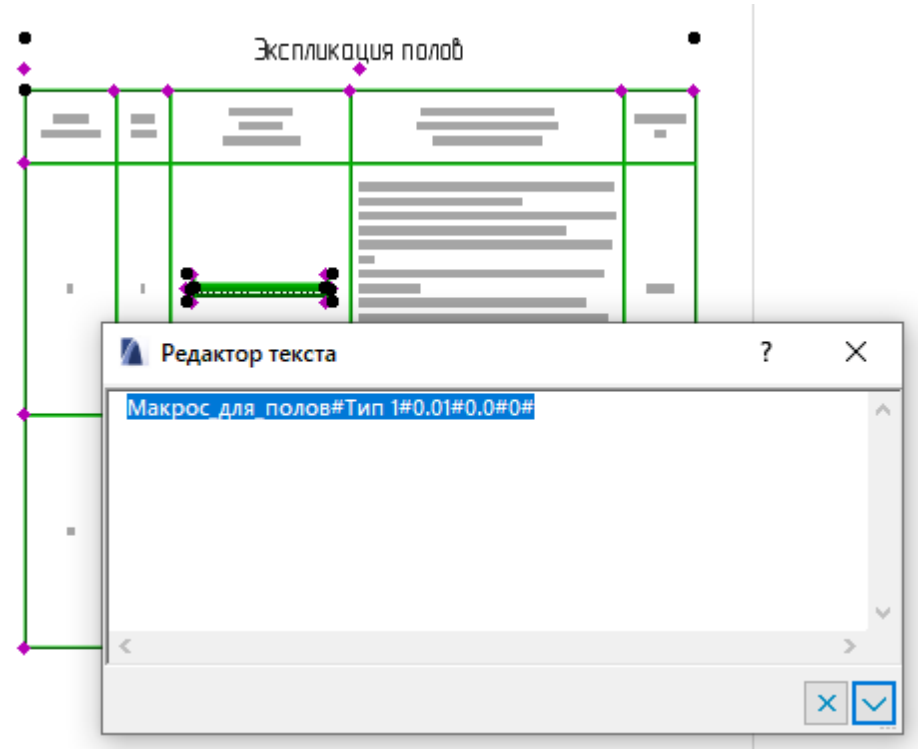
Задать отрисовку линий пером для толстых линий.
Задать отрисовку линий пером для тонких линий.
Выключить отрисовку линий.
Выполняются одинаково - щелкнуть на воображаемый верхний левый угол и на воображаемый нижний правый угол.
В рамках воображаемого прямоугольника линии будут заданы в соответствии с 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#
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

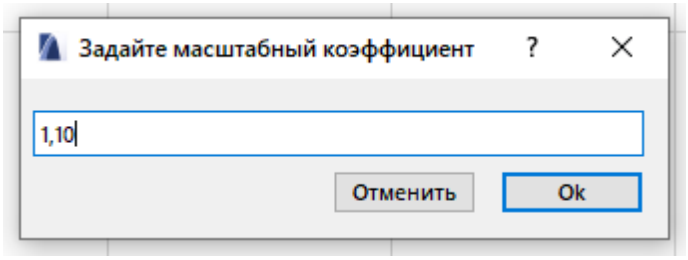
	fillAttribute_1 = 214
	fillAttribute_2 = 330

	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 500x100
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 500x100

TS_Cells_macro - коды для отрисовки макросов в ячейках таблицы
Массив текстовый 500x100

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		1	2	3	4	5	6	7	
Число для ввода Alt+	8304	0185	0178	0179	8308	8309	8310	8311	
Unicode	2070	00B9	00B2	00B3	2074	2075	2076	2077	



Ведомость

Наименование	Площадь	Кол-во
Блоки оконные из алюминиевых сплавов ГОСТ 21519-2003	1,44	4
Плитка 600x600 бежевая	1,400	4
Железобетон	0,36	1

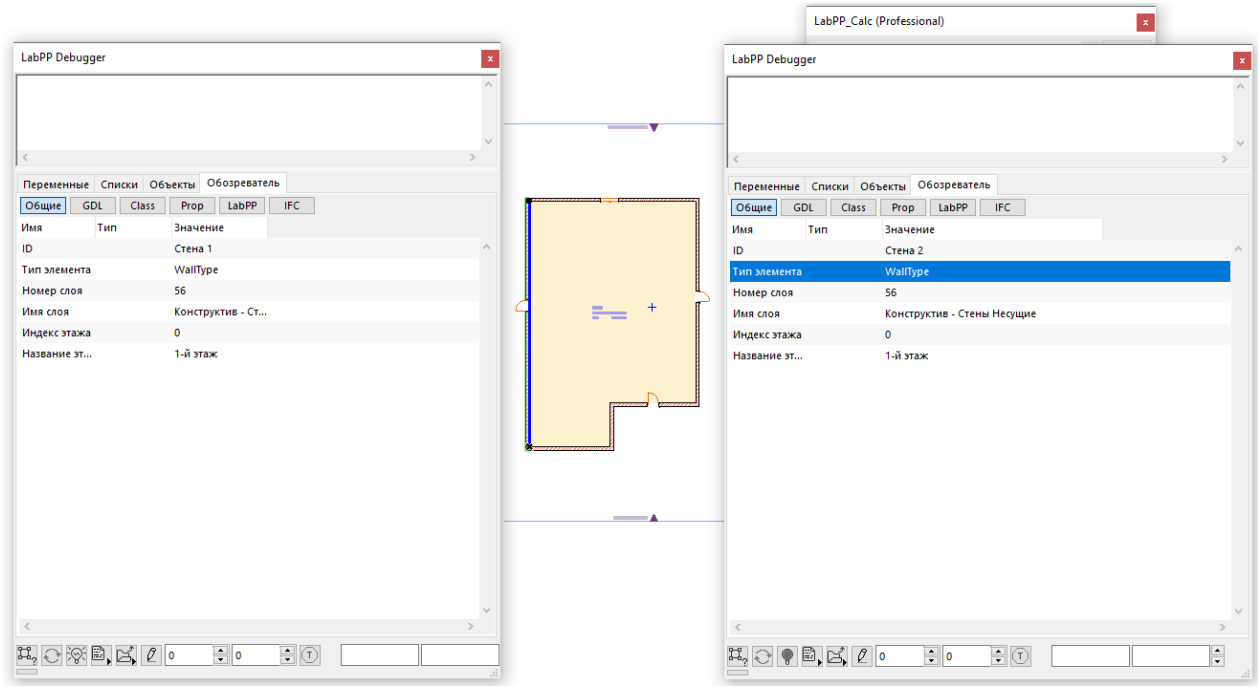
Отладчик с браузером данных элементов

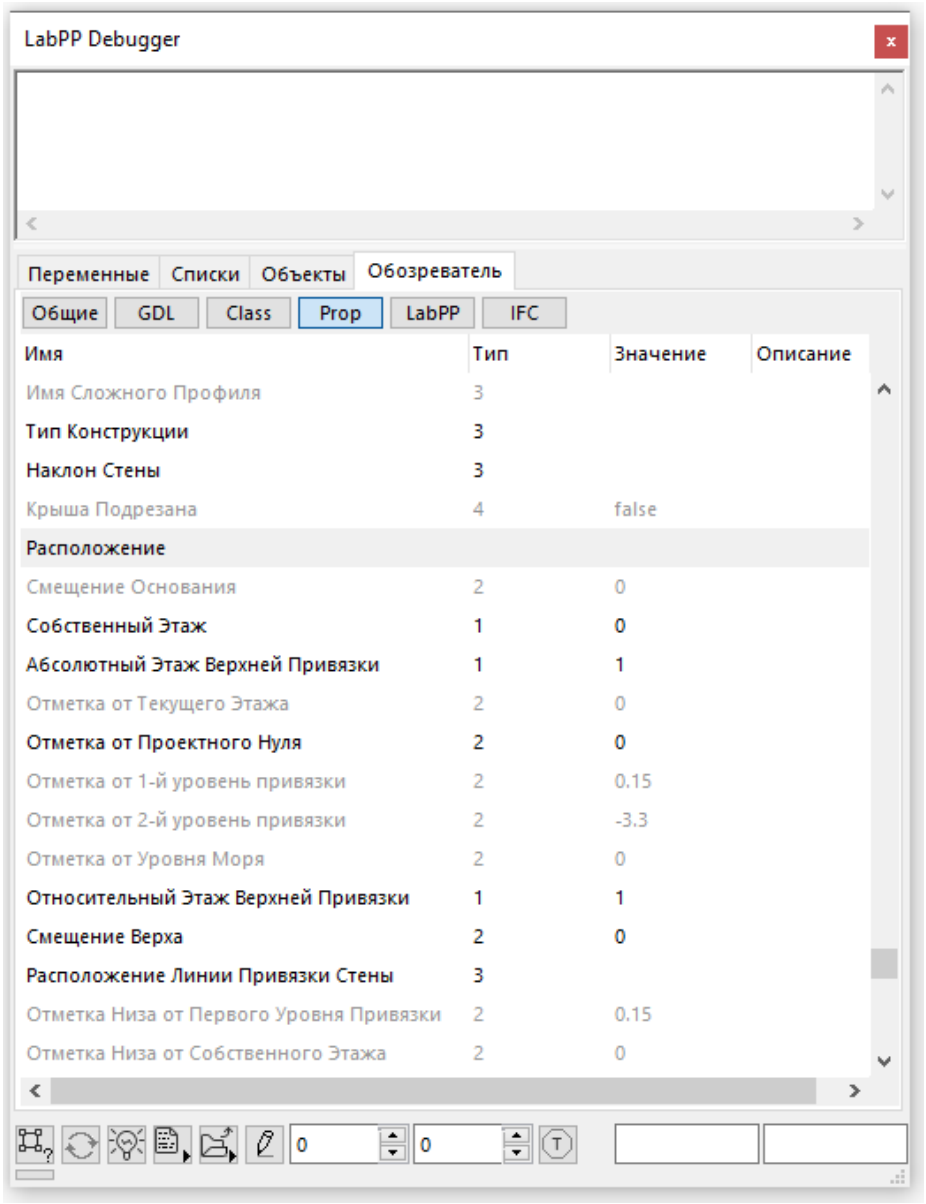
Отладчик с фунциями браузера позволяет просматривать и изменять информацию у элементов.
Вы можете, например, увидеть значения параметров (в том числе заданных формулами), значения параметров GDL элементов, IFC свойства и т.д.

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.


LabPP Debugger
Переменные
Списки
Объекты
Обозреватель
Общие
GDL
Class
Prop
LabPP
IFC
Имя
Тип
Значение
ID
Стена 1
Тип элемента
WallType
Номер слоя
56
Имя слоя
Конструктив - Ст...
Индекс этажа
0
Название эт...
1-й этаж
LabPP_Calc (Professional)
LabPP Debugger
Переменные
Списки
Объекты
Обозреватель
Общие
GDL
Class
Prop
LabPP
IFC
Имя
Тип
Значение
ID
Стена 2
Тип элемента
WallType
Номер слоя
56
Имя слоя
Конструктив - Стены Несущие
Индекс этажа
0
Название эт...
1-й этаж


LabPP Debugger
Переменные
Списки
Объекты
Обозреватель
Общие
GDL
Class
Prop
LabPP
IFC
Имя
Тип
Значение
Описание
Имя Сложного Профиля
3
Тип Конструкции
3
Наклон Стены
3
Крыша Подрезана
4
false
Расположение
Смещение Основания
2
0
Собственный Этаж
1
0
Абсолютный Этаж Верхней Привязки
1
1
Отметка от Текущего Этажа
2
0
Отметка от Проектного Нуля
2
0
Отметка от 1-й уровень привязки
2
0.15
Отметка от 2-й уровень привязки
2
-3.3
Отметка от Уровня Моря
2
0
Относительный Этаж Верхней Привязки
1
1
Смещение Верха
2
0
Расположение Линии Привязки Стены
3
Отметка Низа от Первого Уровня Привязки
2
0.15
Отметка Низа от Собственного Этажа
2
0

Уроки

Вычислить середину высоты элемента Морф относительно 0 проекта и записать в поле LabPP_Calc

Скачать файл можно здесь.

```
// 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 ielemstoreindex;
	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;

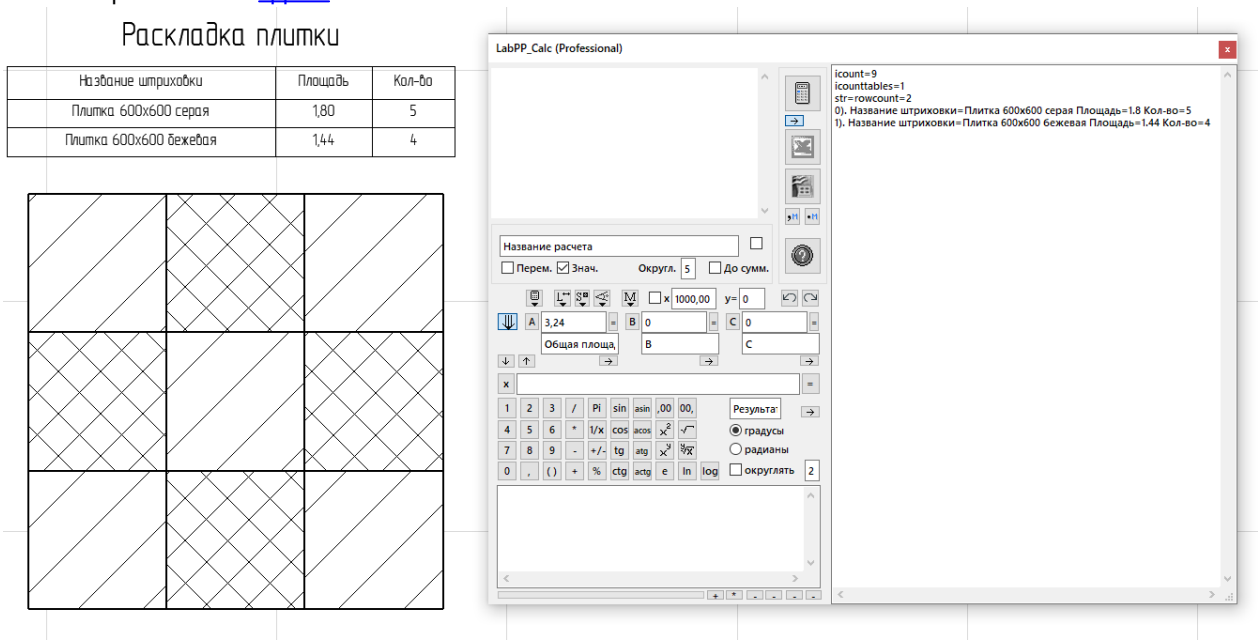
	for (i = 0; i < icount; i++)
	{
```

```
            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", "StoreIndex"); // запрашиваем индекс этажа
            ielemstoreindex = ac_getnumvalue(); // получаем индекс этажа в переменную
            // получаем высоту центра Морф относительно 0 проекта в переменную
elemprojectlevel
            ac_request("get_levelfromprojectnull_by_floorindexandlevel",
ielemstoreindex, 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.




```
// 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.