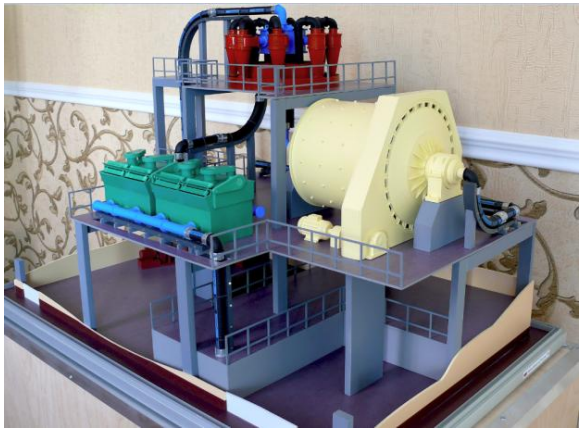
ARCHICAD
язык скриптов C++
справочное руководство

Добро пожаловать на сайт нашей макетной мастерской
www.МакетыРоссии.РФ



Вы увидите и можете заказать возможно самые красивые макеты.

Introduction

This manual is a complete reference to the LabPP's C++ scripting language for GRAPHISOFT® ARCHICAD®. The manual is recommended for those users who wish to expand on the possibilities presented by the GRAPHISOFT® ARCHICAD® - great BIM CAD software for architects. It gives a detailed description of C++ scripting for ARCHICAD, including syntax definitions, commands, variables, etc.
Applying scripts on this language is available in add-ons LabPP_Automat and any add-ons, that includes LabPP C++ interpreter for ARCHICAD.

Оглавление

Общий обзор

В качестве основы языка программирования в LabPP_Automat используется широко известный, простой и гибкий язык Си++.
Достоинствами этого языка являются:
- высокая скорость выполнения;
- орошая 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.

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

```
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";
```

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

В зависимости от значения числа в скобках выполняется соответствующих фрагмент программы.

```
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.
Обращение

#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.

Функции для работы с оболочкой

shell_func

get_path

Установить текущий каталог.

Обращение:

```
int res =  shell_func("get_path", string what, string result);
```

Переменная what задает тип возвращаемого пути:

"rootconfig" - получить путь к корневому каталогу текущей конфигурации

"tsimages" - получить путь к каталогу картинок текущей конфигурации

"tsprg" - получить путь к каталогу программ текущей конфигурации

"rootaddons" - получить путь к каталогу, где находится сам LabPP_Automat

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 или директива ("rootconfig", tsimages, tsprg или rootaddons).

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.
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","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.

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;
```

Файловые операции

Файловые операции выполняются при помощи функции ts_file. Для обращения требуется дескриптор объекта типа "ts_file".

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();

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

Работа с ARCHICAD

Объект ac_element_guid

Объект ac_element

ac_request()

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(....

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

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.

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,

set_current_element_from_list

Установить текущим элемент с указанным индексом из указанного списка.

Обращение:

ac_request("set_current_element_from_list", int iListNum, int index);

Здесь - iListNum - номер списка,

index - интекс элемента в списке.

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

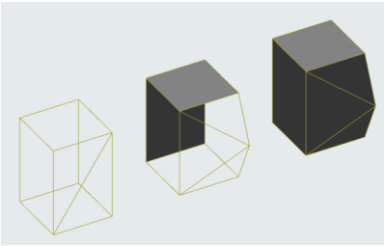
Оболочка - ShellType:

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";
  }
}
```

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";
}
```

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";
  }
}
```

get_gdlelem_property_value

Получить значение параметра текущего GDL элемента. Применяется для элементов типа паспорт зоны.

Обращение:

int iret = ac_request("get_gdlelem_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 текущей зоны значение параметра "ROOM_AREA" (измеренное значение площади);

```
int iret = ac_request("get_gdlelem_property_value", "ROOM_AREA");
double darea;
if(iret == 0)
{
  darea = ac_getnumvalue();
  cout << darea;   // написать площадь в окне сообщений
}
```

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

set_element_infoidtext

Задать текущему элементу новое значение ID

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";
}
```

interface_input2dline

Ввод пользователем 2d линии.

Обращение:

int iret ac_request("interface_input3dline", 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

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();

описание раздела в разработке

Для конфигурации приложений

setcfg()

Конфигурация квартирографии

Число знаков после запятой - ROUNDINGPRECISION

Формат запроса:
SETCFG("SETAUTOMATBUTTON",int buttonnum,string smessage, string prgfilename);

Здесь: buttonnum - номер программируемой кнопки (обычно от 1 до 4), smessage - сообщение, которое будет выводиться когда пользователь наведет мышь на кнопку, prgfilename - имя файла с программой для выполнения по этой кнопке.
Пример.
Закрепить за первой программируемой кнопкой на панели приложения - выполнение программы из файла 1.cpp. При указании мышкой на эту кнопку выдать сообщение что это за первая кнопка.

```
SETCFG("SETAUTOMATBUTTON",1,"Первая программируемая кнопка","1.cpp");
```

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

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";
```

Ввод числа или строки

editdoubledialog

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

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, ssdefaultstr - 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.");
```

Гравитация на поверхность (LABPP)

Приземление элементов - do_elements_landing

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.

Приземление по точкам - do_surface_landing

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.

Приземление точки X,Y - do_point_landing

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;
```

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 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

Большая функция для работы с Excel при помощи директив.

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");

get_borders

Получить расстановку бордюров вокруг текущего выделенного фрагмента ячеек.
Обращение:

excel_request("get_borders",int border_left,int border_top,int border_right,int border_bottom);

Здесь: border_left, border_top, border_right, border_bottom - переменные, куда записываются значения 0 или 1 в зависимости установлен или нет бордюр слева,сверху,справа и снизу, соответственно.

put_selection_values

put_selection_fontvalues

get_selection_area

merged_cell_info

is_merge_cells

set_backcolor

get_backcolor

set_interior

get_interior

selection_varvalues

selection_font_varvalues

sheet_select

range_copy

booknamedcell

описание раздела в разработке

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

Считывание и запись полей переменных - docfield

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.

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.
Обновление полей переменных в тексте документа Word -

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");

описание раздела в разработке

Внутренние объекты

Множество механизмов реализовано в виде внутренних объектов.
Объекты можно создавать, манипулировать ими и удалять.
Работа с объектами производится через дескриптор объекта.
Это целое число, которое присваивается при создании объекта.

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);
```

Объекты для обработки табличных данных

Функция ts_table

Функция ts_table позволяет выполнять операции над объектом динамических таблиц.

add_column

Добавить колонку в таблицу. Колонки могут быть строковыми или числовыми.

Формат команды:

ts_table(int descriptor, "add_column", int columnnumber, string columnname);

Здесь:

descriptor - дескриптор экземпляра объекта таблицы, у которой добавляется колонка.

columnnumber - номер колонки (можно поставить -1 чтобы программа сама создала номер).

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);
```

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).
Пример. Сортировка таблицы по колонке №0

```
ts_table(TableDescr1,"sort",0);
```

Пример. Сортитовка таблицы по ключевой колонке, указанной ранее.

```
ts_table(TableDescr1,"sort");
```

После добавления новой строки требуется повторная сотрировка.

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");
```

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);
```

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);
```

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"

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

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

Аргкосинус

tg

`Тангенс угла в радианах`

arctg

Арктангенс

ctg

Котангенс

arcctg

Арккотангенс

Функции для перевода угловых величин

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.

putchar

sprintf

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"

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);

Функции геометрического преобразования - geometry_calc_2d

Обращение к функции ac_request() с директивой geometry_calc_2d предоставляет широкий набор операций для модификации координат и определения взаиморасположения различных элементов.

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);
Здесь:
x1 и y1 - координаты исследуемой точки, iElemDescr - дескриптор объекта, чей полигон проверяется на содержание точки.
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

Определить угол наклона отрезка относительно центра координат.

Обращение

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

Измерение времени выполнения фрагмента кода - codemeter

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.
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";
```

Сохранить текст из окна сообщений в файл - ac_save_messages_to_file

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

Тест связи с квартирографией - ac_request("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;
}
```

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");

Обмен данными между программами - 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.

Запуск другой программы - run_cpp

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.

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.

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.

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", 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.

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.
Ораще́ние:
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.

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 (кнопка)

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.

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.

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.

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.

LEFTTEXT, CENTERTEXT, RIGHTTEXT

Создают статический текст.
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.

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

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.

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");

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.

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

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.

Картинка	Номер в программе
	32000
	32001
	32002
	32003
	32004

SetTabItemIconId

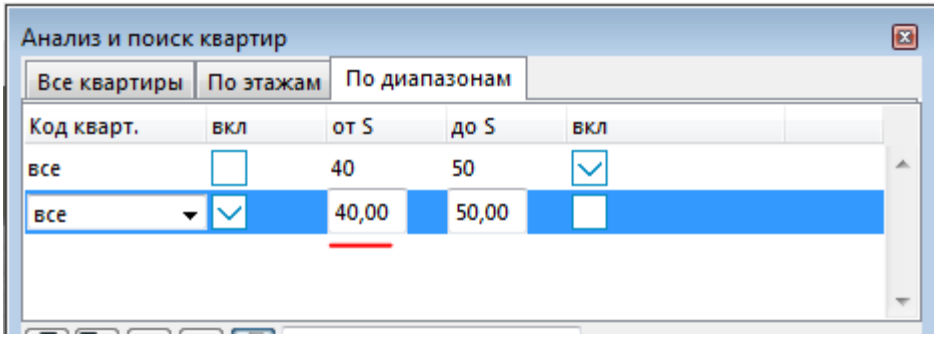
Задать изображение в ячейке таблицы элемента управления по коду.
Обращение:
Обращение:

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");

MULTISELLISTBOX

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.

Картинка	Номер в программе
	32000
	32001
	32002
	32003
	32004

SetTabItemIconId

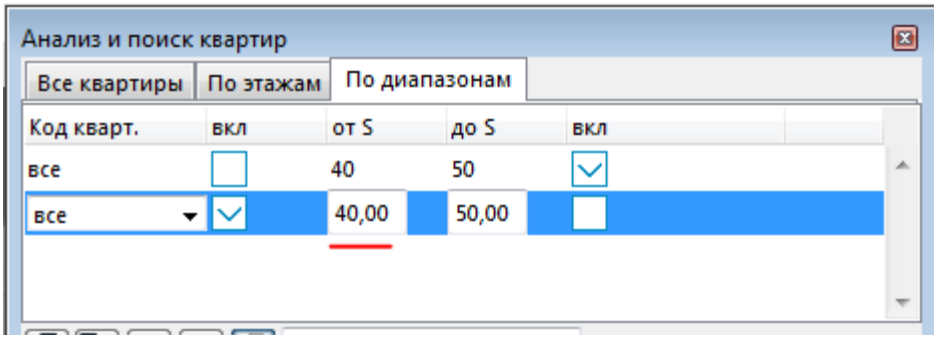
Задать изображение в ячейке таблицы элемента управления по коду.
Обращение:

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.

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.

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),

SetHeaderSynchronState

Установить чтобы заголовки таблицы синхронизировались по ширине и другим атрибутам с колонками.

Обращение:

ts_dialogcontrol(iDialogControlDescr, "SetHeaderSynchronState", int onoff);

Здесь:

onoff - 1/0,

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.

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.

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.

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.

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");

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");

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");

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");

SPLITTER

Attach_controlled_element

Подключить к сплиттеру контролируемый элемент (который будет изменять свою форму или положение в зависимости от перемещения сплиттера).
Обращение:
ts_dialogcontrol(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

Задать текущее значение процесса выполнения.
Обращение:
ts_dialogcontrol(iProgressBar, "SetValue", int value);
Здесь:
value - текущее значение, по которому считать процент выполнения.

GetValue, GetMin, GetMax

Считать текущие параметры элемента.
Обращение:
ts_dialogcontrol(iProgressBar, "GetMin", 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");

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.

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, RightText	Щелчок на тексте
Event_StaticTextDoubleClicked		Двойной щелчок
Event_CheckItemChanged	CheckBox, IconCheckBox, IconPushCheck, PushCheck	Смена выбранного элемента
Event_CheckItemDoubleClicked		Двойной щелчок
Event_ButtonClicked	Button, IconButton	Нажатие кнопки
Event_NormalTabChanged	NormalTab	Изменение текущего элемента в NormalTab (переход на другую панель)
Event_NormalTabClicked	NormalTab	Щелчек мыши на поле элемента NormalTab

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 любой сложности

Класс ts_json

Класс ts_array

Класс ts_class

Класс ts_num

Класс ts_bool

Класс ts_string