Лабораторная работа 1 (Пример)

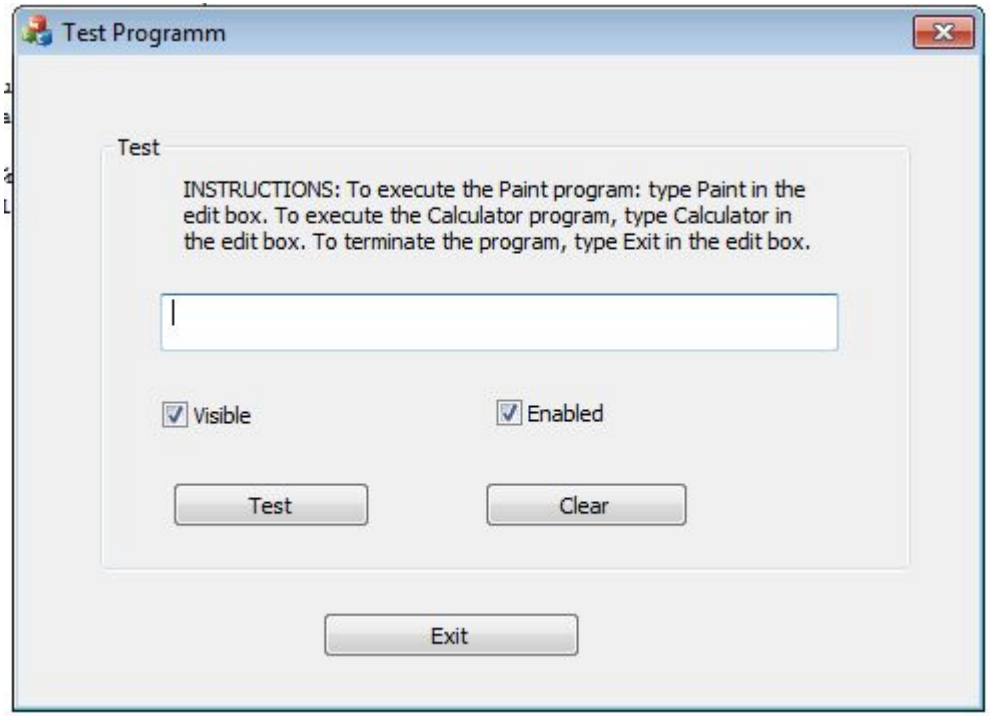
- Создается класс Student. Формируется массив объектов. При тестировании выводится: сформированный список студентов, список студентов заданного факультета, список студентов для заданных факультета и курса.

```cpp
#include <conio.h>
#include <string.h>
#include <iostream>

using namespace std;
struct date // дата рождения
{char daymon[6];
int year; };
//======= class Student ================
class Student{
char name[30]; //private
date t;
char adr[30], fac[20];
int kurs;
public:
Student();
char *getfac();
int getkurs();
void show();
};
Student::Student()
{cout<<"Input name:"; cin>>name;
cout<<"Input date of born\n";
cout<<"Day.mon:"; cin>>t.daymon;
cout<<"Year:"; cin>>t.year;
cout<<"Input adr:"; cin>>adr;
cout<<"Input fac:"; cin>>fac;
cout<<"Input kurs:"; cin>>kurs;
}
void Student::show()
{
cout<<"Name :"<<name<<endl;
cout<<"Was born :"<<t.daymon<<'.'<<t.year<<endl;
cout<<"Address :"<<adr<<endl;
cout<<"Fac :"<<fac<<endl;
cout<<"Kurs :"<<kurs<<endl;
}
```

```
38  char *Student::getfac() { return fac; }
39  int Student::getkurs() { return kurs; }
40  void spisfac(Student spis[],int n)//список студентов заданного факультетата
41  {char fac[20];
42  cout<<"Input faculty:"; cin>>fac;
43  for(int i=0;i<n;i++)
44  if(strcmp(spis[i].getfac(),fac)==0)spis[i].show();
45  }
46  void spisfackurs(Student spis[],int n)
47  //список студентов заданных факультета и курса
48  {int i,k;
49  char fac[20];
50  cout<<"Input faculty:"; cin>>fac;
51  cout<<"Input the course:"; cin>>k;
52  for(i=0;i<n;i++)
53  if ((strcmp(spis[i].getfac(),fac)==0)&&(spis[i].getkurs()==k))
54  spis[i].show();
55  }
56  //========= main ================
57  int main()
58  { Student *spis;
59  int n;
60  cout<<"Input a number of students: "; cin>>n;
61  spis=new Student [n];
62  for(int i=0;i<n;i++) {
63  cout<<"================================"<<endl;
64  spis[i].show();
65  }
66  spisfac(spis,n);
67  spisfackurs(spis,n);
68  delete [] spis;
69  cout<<"press any key!";
70  while(!kbhit());
71  }
```

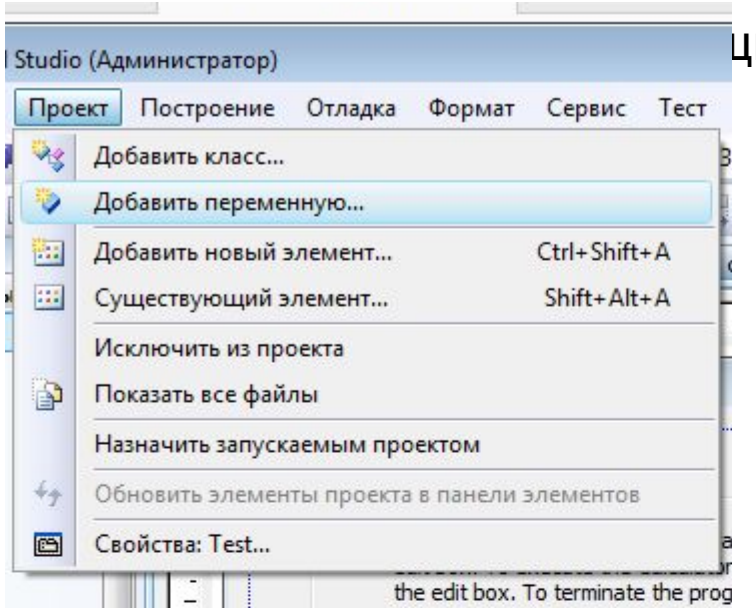
Элементы управления

- Создадим программу, выполняющую следующие действия: При вводе в строке редактирования слов "Paint" и "Calculator", запускаются соответственно "Windows Paint" и "Windows Calculator»
- При снятии флажка Visible, строка редактирования исчезает, а при снятии флажка Enabled, закрывается доступ к окну редактирования.
- При отмечании флажков, все происходит наоборот.

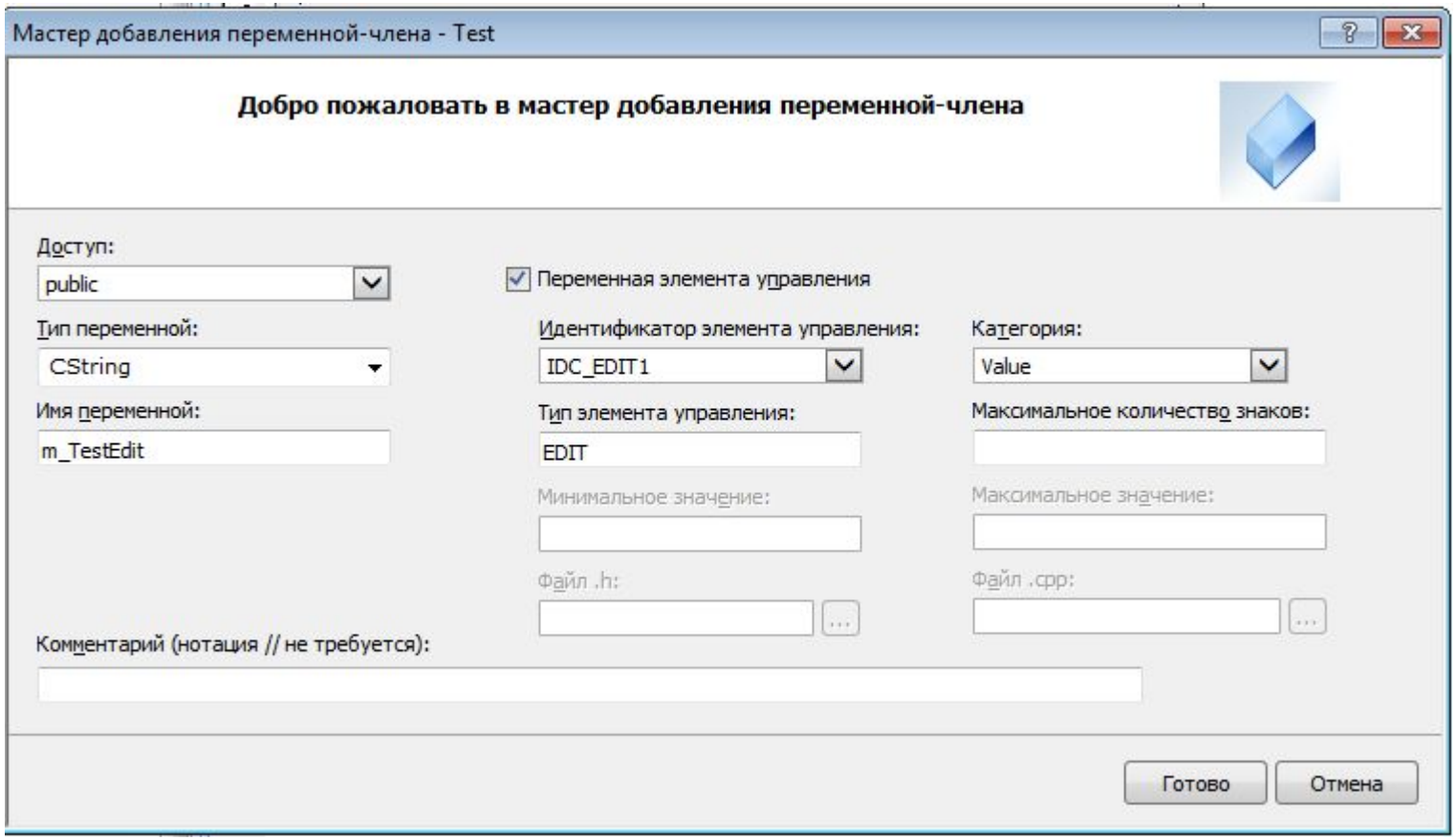


Для окна редактирования нужна переменная типа CString, используя эту переменную, мы сможем извлекать текст из окна редактирования, а также изменять его содержимое.

Также необходимо связать переменные с флажками, чтобы

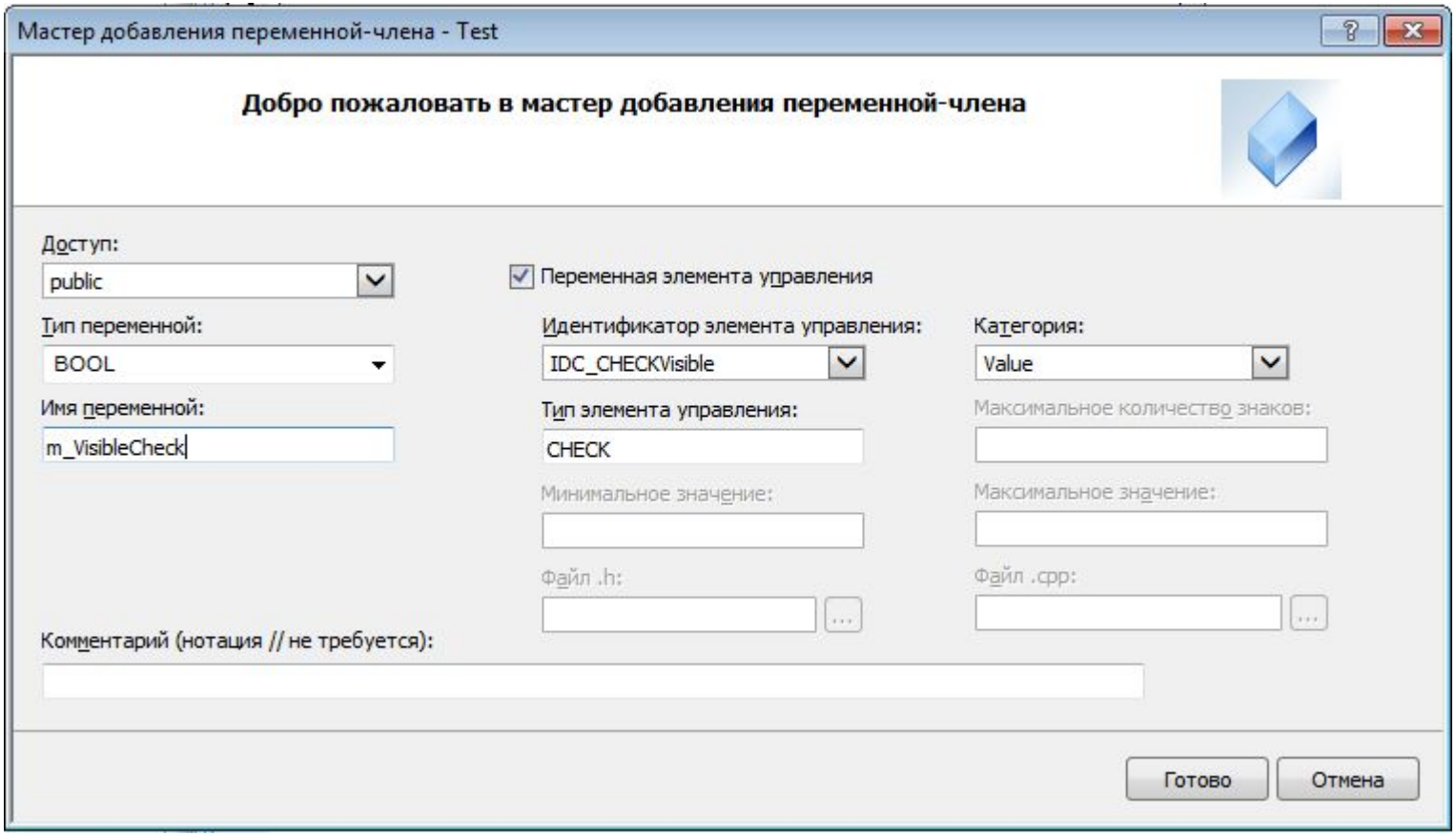


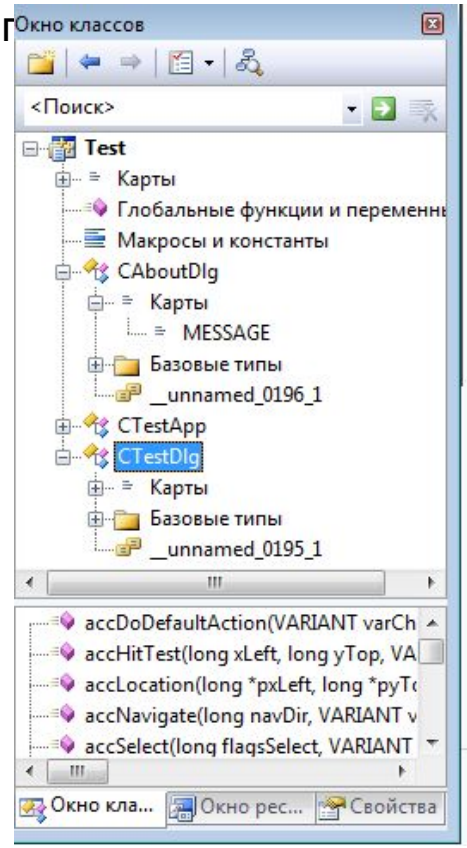
ции состояния: включено,

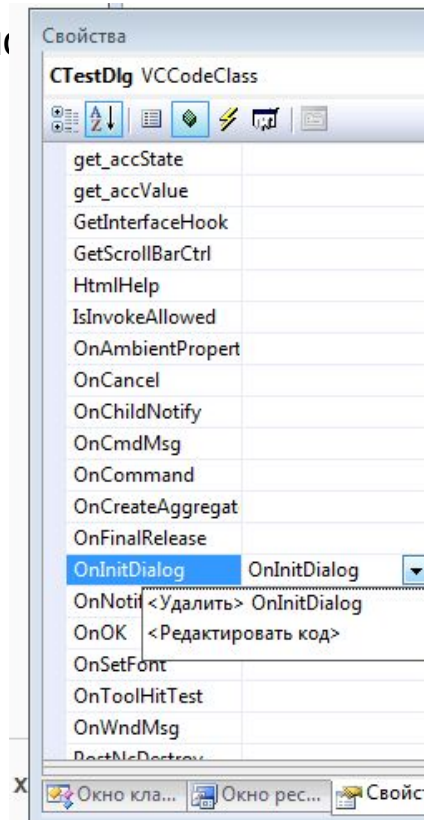


Категория - здесь выбирается категория переменной. К примеру, Value - это категория переменной, предназначенная для передачи значения переменной, Control - категория предназначена для управления элементами. К примеру, с помощью переменной такой категории можно менять названия кнопок, добавлять строки в ListBox и многое др.

Varible Type- тип переменной.


Мастер добавления переменной-члена - Test
Добро пожаловать в мастер добавления переменной-члена
Доступ:
public
Переменная элемента управления
Тип переменной:
BOOL
Идентификатор элемента управления:
IDC_CHECKVisible
Категория:
Value
Имя переменной:
m_VisibleCheck
Тип элемента управления:
CHECK
Максимальное количество знаков:
Минимальное значение:
Максимальное значение:
Файл .h:
Файл .cpp:
Комментарий (нотация // не требуется):
Готово
Отмена

При запуске программы, основанной на диалоге, необходимо установить определенные значения некоторых элементов управления: необходимо, чтобы флажки Visible и Enable были отмечены, иначе окна редактирования не будет видно. Для этог 



// TODO: добавьте дополнительную инициализацию

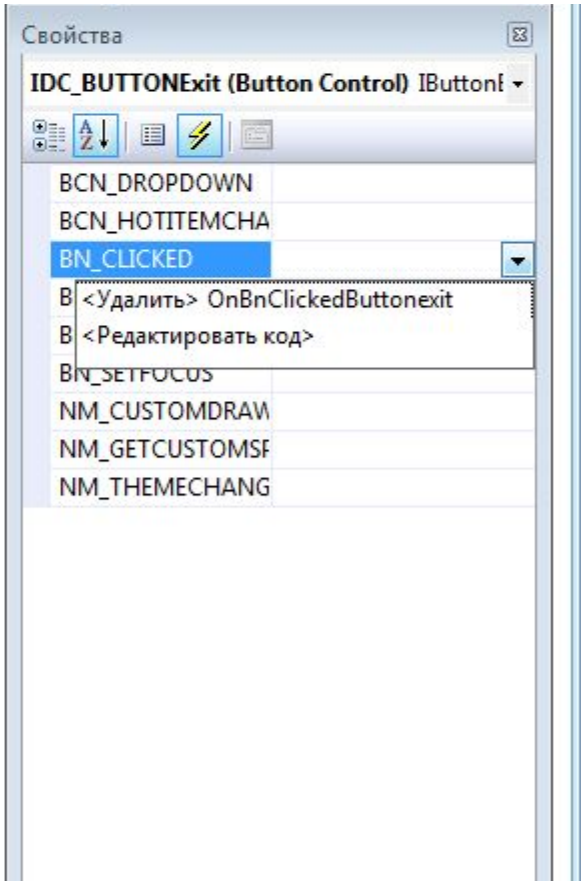
```
    // TODO: добавьте дополнительную инициализацию

//Мой код начинается здесь//////////
//Установить переменную флажка VisibleCheck и EnabledCheck в состояние TRUE

m_VisibleCheck=TRUE;

m_EnableCheck=TRUE;

//Обновить экран
UpdateData(FALSE);

////////Мой код заканчивается здесь//////////

  return TRUE;  // возврат значения TRUE, если фокус не передан элементу управления
}
```

Рассмотрим код:

Первый и второй операторы присваивают переменным m_VisibleCheck и m_EnableCheck значение TRUE. Это означает, что при запуске программы флажки будут отмечены.

Последний оператор UpdateData(FALSE) обновляет экран, т.е. он обновляет значения переменных элементов управления на текущие. В нашем случае, при выполнении этого оператора текущее содержимое переменных связанных с флажками будет передано к ним.

Связывание кода с событием BN_CLICKED кнопки Exit




```
//void CTestDlg::OnBnClickedButtonexit()
{
//  // TODO: добавьте свой код обработчика уведомлений
    OnOK();
}
```

Связывание кода с событием BN_CLICKED кнопки Test

При нажатии на кнопку Test программа Test.Exe в окне редактирования напишет текст: This is a Test. Чтобы связать код с событием BN_CLICKED кнопки Test, выполните следующие действия:

```
void CTestDlg::OnBnClickedButtontest()
{
    // TODO: добавьте свой код обработчика уведомлений
    //Присвоить переменной окна редактирования IDC_TEST_EDIT значение This is a Test.
    m_TestEdit="This is a Test";

// Обновить экран
    UpdateData(FALSE);
}
```

Связывание кода с событием BN_CLICKED кнопки

```
void CTestDlg::OnBnClickedButtonclear()
{
    // TODO: добавьте свой код обработчика уведомлений
    ///Присвоить переменной окна редактирования IDC_TEST_EDIT значение NULL

m_TestEdit=" ";

// Обновить экран
UpdateData(FALSE);

}
```

Связывание кода с событием BN_CLICKED флажка

Visible

При включении флажка Visible программа Test.Exe должна сделать окно редактирования невидимым, а при выключении - наоборот

```
void CTestDlg::OnBnClickedCheckvisible()
{
    // TODO: добавьте свой код обработчика уведомлений
//Обновить значения переменных элементов управления(содержимое эрана передается переменным элементов управления)

UpdateData(TRUE);

//Если отметка флажка Visible зделать окно редактирования видимым, а если нет - то невидимым
if (m_VisibleCheck==TRUE)

GetDlgItem(IDC_EDIT1)->ShowWindow(SW_SHOW);
else GetDlgItem(IDC_EDIT1)->ShowWindow(SW_HIDE);

}
```

Код, который вы ввели, содержит следующие операторы:

- UpdateData(TRUE); - этот оператор обновляет значения переменных элементов управления текущими значениями, которые содержаться на экране. Т.е. при нажатии на флажок переменная может принимать два значения TRUE или FALSE, TRUE - это когда флажок включен, а FALSE - наоборот. Значит, при выполнении этого оператора переменная флажка управления принимает текущее положение флажка и все остальные переменные обновляются значениями, которые отображаются на экране.

- Следующий оператор проверяет: включен или выключен флажок. Если он включен, то выполняется оператор GetDlgItem(IDC_TEST_EDIT)->ShowWindow(SW_SHOW); где GetDlgItem(IDC_TEST_EDIT) извлекает указатель на элемент управления, а функция ShowWindow(SW_SHOW); с параметром SW_SHOW, делает окно редактирования видимым. А если флажок не отмечен, то выполняется та же самая функция ShowWindow(); с параметром SW_HIDE (спрятать окно редактирования).

Связывание кода с событием BN_CLICKED флажка

Enable

При включении флажка Enable программа Test.exe должна сделать окно редактирования доступным, а при выключении - недоступным. Чтобы связать код с событием BN_CLICKED флажка Enable, выполните действия аналогичные

```
void CTestDlg::OnBnClickedCheckenabled()
{
    // TODO: добавьте свой код обработчика уведомлений
    //Обновить значения переменных элементов управления,
//(содержимое эрана передается переменным элементов управления) UpdateData(TRUE);
//Если отметка флажка Enable сделать окно редактирования видимым, а если нет - то невидимым

if(m_EnableCheck==TRUE)

GetDlgItem(IDC_EDIT1)->EnableWindow(SW_SHOW);
else GetDlgItem(IDC_EDIT1)->EnableWindow(SW_HIDE);

}
```

Связывание кода с событием EN_CHANGE окна редактирования

```
void CTestDlg::OnEnChangeEdit1()
{
    // TODO:  Если это элемент управления RICHEDIT, то элемент управления не будет
    // send this notification unless you override the CDialog::OnInitDialog()
    // function and call CRichEditCtrl().SetEventMask()
    // with the ENM_CHANGE flag ORed into the mask.

    // TODO:  Добавьте код элемента управления
    UpdateData(TRUE);

///Содать переменную типа CString, присвоить ей значение переменной m_TestEdit и
    //выполнить перевод символов в верхний регистр.

CString UpperValue;
UpperValue=m_TestEdit;
UpperValue.MakeUpper();

///Если в окне редактирования напечатано PAINT

//запускается редактор PAINT и окно редактирования становится пустым.
if(UpperValue=="PAINT")
{
system("mspaint.exe");
m_TestEdit=" ";
UpdateData(FALSE);
}

//Если в окне редактирования напечатано CALCULATOR запускается калькулятор
//и окно редактирования становится пустым.
if(UpperValue=="CALCULATOR")
{
system("calc.exe");
m_TestEdit=" ";
UpdateData(FALSE);
}
}
```

Код содержит следующие операторы:

- UpdateData(TRUE); - обновляет переменную m_TestEdit значением содержимого окна редактирования, при каждом его изменении, так как окно редактирования связано с событием EN_CHANGE.

- Следующий оператор CString UpperValue создает новую переменную типа CString.

- Затем переменная UpperValue приравнивается к переменной m_TestEdit, это можно сделать, так как они имеют одинаковый тип.

- Оператор UpperValue.MakeUpper(); переводит все символы переменной UpperValue в верхний регистр.

- Оператор if(UpperValue=="PAINT"); проверяет, если введено ли слово PAINT, то выполняются следующие три оператора: system("pbrush.exe"); - запускает графический редактор, так как не указан явный путь к файлу, то программа будет искать его в каталоге C:\WINDOWS; m_TestEdit=""; - присваивает переменной окна редактирования значение NULL; UpadteData(FALSE) - обновляет экран.