C++. Базовый уровень

Создание оконных приложений

КОД БУДУЩЕГО

Минцифры России

инжиниум МГТУ им. Н.Э. Баумана

НАЦИОНАЛЬНЫЕ ПРОЕКТЫ РОССИИ ЦИФРОВАЯ ЭКОНОМИКА

20.35 УНИВЕРСИТЕТ

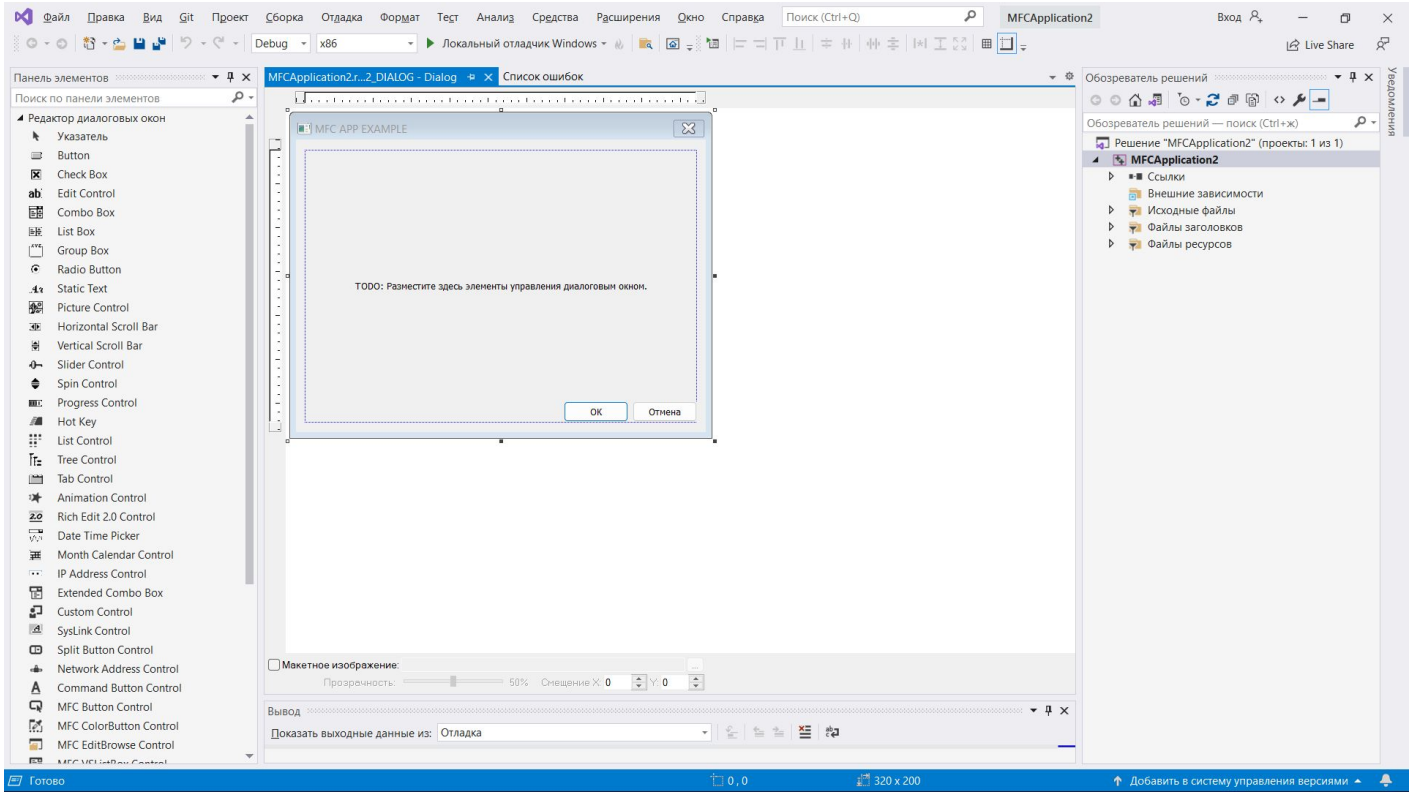
2023

Создание приложений

Прежде чем начать, убедитесь, что у вас установлена среда разработки Visual Studio с поддержкой MFC. Если у вас еще нет Visual Studio, вы можете скачать ее с официального сайта Microsoft.

1. Откройте Visual Studio и выберите "Создать проект".
2. В окне "Создание проекта" выберите "MFC Application" и нажмите "Далее".
3. Введите имя проекта и выберите место сохранения. Нажмите "Далее".
4. В окне "Выбор типа приложения" выберите “На основе диалоговых окон" и нажмите "Далее".
5. В окне "Выбор опций MFC" убедитесь, что выбрана опция "Использовать MFC в виде разделяемой DLL" и нажмите "Далее".
6. В окне “Функции пользовательского интерфейса" укажите заголовок диалогового окна и нажмите "Далее".
7. Нажмите "Готово" дл

Создание приложений



Создание приложений

Как работает MFC-приложение?

MFC-приложение основано на архитектуре **документ-представление** (document-view), которая разделяет **данные** (документ) и их **отображение** (представление) в разные классы.

Это позволяет иметь несколько представлений одного документа и обновлять их автоматически при изменении данных. Кроме того, MFC-приложение использует **модель сообщений** (message-driven) для взаимодействия с пользователем и операционной системой.

Каждое действие пользователя или системы генерирует сообщение, которое посылается в очередь сообщений приложения. Затем сообщение обрабатывается соответствующим объектом-окном, который вызывает функцию-обработчик для выполнения нужных действий.

MFC предоставляет механизм **карт сообщений** (message maps), который связывает сообщения с функциями-обработчиками в классах MFC

Создание приложений

При создании проекта в MFC все исходные файлы играют важную роль в определении структуры и функциональности проекта.

Для приложений, основанных на документах, структура более сложная, однако для понимания архитектуры кратко её рассмотрим:

1. ClassView: Это представление классов в среде разработки Visual Studio
2. FileView: Это представление файлов в среде разработки Visual Studio
3. MainFrm: Этот файл содержит реализацию класса CMainFrame, который представляет главное окно приложения. Он определяет внешний вид и поведение главного окна, включая размещение элементов управления и обработку событий
4. ***<имя проекта>***: Этот файл представляет ***главный класс приложения***. Он содержит точку входа в приложение и определяет основные настройки и поведение приложения

Создание приложений

5. ***<имя проекта>*** Doc: Этот файл представляет класс документа приложения. Он содержит данные, связанные с документом, и определяет методы для работы с этими данными.
6. ***<имя проекта>*** View: Этот файл представляет класс представления приложения. Он определяет визуальное представление документа и обрабатывает события, связанные с пользовательским интерфейсом.
7. D: Этот файл содержит реализацию класса COutputWnd, который представляет окно вывода в среде разработки Visual Studio. Оно используется для отображения сообщений, предупреждений и ошибок, связанных с процессом разработки.
8. pch: Этот файл содержит предкомпилированный заголовочный файл. Он используется для ускорения процесса компиляции, предварительно компилируя общие заголовки и включения.
9. PropertiesWnd: Этот файл содержит реализацию класса CPropertiesWnd, который представляет окно свойств в среде разработки Visual Studio. Оно используется для отображения и редактирования свойств объектов в проекте.
10. ViewTree: Этот файл содержит реализацию класса CViewTree, который представляет дерево представлений в среде разработки Visual Studio. Оно используется для отображения и организации представлений проекта.

Создание приложений

К примеру, проект называется **MyMFCApp.**

Самый важный файл - это **MyMFCApp.cpp**, в котором определена функция **WinMain**, точка входа в ваше приложение. В этом файле также создается объект класса **CMyMFCApp**, который является производным от класса **CWinApp**.

Класс **CWinApp** представляет ваше приложение в MFC и содержит важные методы, такие как **InitInstance** и **ExitInstance**, которые вызываются при запуске и завершении приложения соответственно.

В методе **InitInstance** вы можете инициализировать различные ресурсы, такие как кисти, шрифты, палитры и т.д., а также создать главное окно вашего приложения.

В методе **ExitInstance** вы должны освободить все выделенные ресурсы и выполнить любую необходимую очистку.

Создание приложений

Главное окно вашего приложения представлено классом **CMyMFCAppDlg**, который наследуется от класса **CDialogEx**.

Класс **CDialogEx** является расширенной версией класса **CDialog**, который представляет диалоговое окно в MFC.

Диалоговое окно - это специальный вид окна, который обычно содержит различные элементы управления, такие как кнопки, поля ввода, списки, флажки и т.д., и используется для взаимодействия с пользователем.

Создание приложений

Вы можете настроить внешний вид и поведение вашего диалогового окна с помощью редактора ресурсов Visual Studio, который позволяет визуально добавлять, удалять и изменять элементы управления на вашем диалоговом окне.

Каждый элемент управления имеет уникальный идентификатор, который используется для связывания его с переменной или функцией-обработчиком в вашем коде.

Вы можете использовать класс **CWnd** и его производные классы, такие как **CButton**, **CEdit**, **CListBox** и т.д., для представления и управления элементами управления в MFC.

Создание приложений

Ваш класс **CMyMFCAppDlg** также содержит несколько методов, которые отвечают за инициализацию, отображение и закрытие вашего диалогового окна, а также обработку сообщений от элементов управления.

Сообщения - это способ коммуникации между окнами и приложениями в Windows. Каждое сообщение имеет код, который определяет его тип, и параметры, которые передают дополнительную информацию.

Например, сообщение **WM_PAINT** отправляется окну, когда оно должно перерисовать свое содержимое, а сообщение **WM_COMMAND** отправляется диалоговому окну, когда пользователь нажимает кнопку или выбирает пункт меню.

Вы можете обрабатывать сообщения в своем классе с помощью метода **OnMessage**, где **Message** - это имя сообщения, например, **OnPaint** или **OnCommand**. Вы также можете использовать макросы **BEGIN_MESSAGE_MAP** и **END_MESSAGE_MAP**, чтобы объявить таблицу соответствия между сообщениями и методами-обработчиками в вашем классе.

Создание приложений

Основная структура класса CMyMFCAppDlg:

```
// MyMFCAppDlg.h - заголовочный файл класса CMyMFCAppDlg
#pragma once

// CMyMFCAppDlg - класс диалогового окна
class CMyMFCAppDlg : public CDialogEx
{
public:
// Конструктор по умолчанию
CMyMFCAppDlg();

// Деструктор
virtual ~CMyMFCAppDlg();

// Идентификатор диалогового окна
enum { IDD = IDD_MYMFCAPP_DIALOG };

protected:
// Метод для инициализации данных диалогового окна
virtual BOOL OnInitDialog();

// Метод для отображения диалогового окна
virtual void DoDataExchange(CDataExchange* pDX);

// Метод для закрытия диалогового окна
virtual void OnOK();

// Метод для обработки сообщения WM_PAINT
afx_msg void OnPaint();

// Метод для обработки сообщения WM_COMMAND от кнопки "Hello"
afx_msg void OnBnClickedHello();

// Объявление таблицы соответствия сообщений и методов-обработчиков
DECLARE_MESSAGE_MAP()
};
```

Создание приложений

Для приложений, на основе диалоговых окон, пример которого мы создали в начале, эта логика «спрятана».

Как же добавлять элементы в окно? Как обрабатывать сообщения?

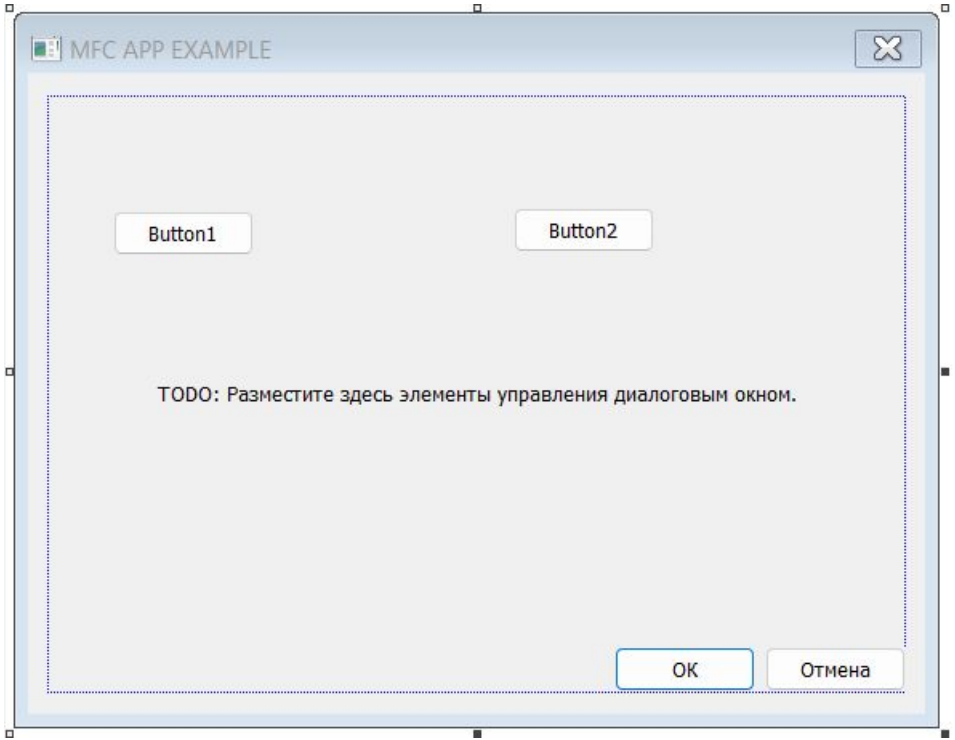
Рассмотрим на примере кнопки.

Создание приложений

Для добавления кнопки в MFC-приложение вам нужно выполнить следующие шаги:

1. Откройте редактор ресурсов в Visual Studio и выберите диалоговое окно или окно представления, в котором вы хотите разместить кнопку.
2. В палитре элементов управления выберите инструмент **Button** и перетащите его на форму. Вы можете изменить размер и положение кнопки с помощью мыши или свойств.
3. В свойствах кнопки укажите ее идентификатор (например, IDC_BUTTON1) и текст (например, “Click Me”).
4. Для создания функции-обработчика нажатия на кнопку вам нужно дважды щелкнуть по кнопке в редакторе ресурсов или выбрать ее и нажать **Add Event Handler** в панели свойств. В диалоговом окне **Add Event Handler** выберите класс, в котором вы хотите создать обработчик (например, CMyDialog или CMyView), и тип события (например, BN_CLICKED). Затем нажмите **Add and Edit**. Visual Studio сгенерирует прототип и реализацию функции-обработчика в заголовочном и исходном файле класса и откроет исходный файл для редактирования.
5. В теле функции-обработчика вы можете написать код, который будет выполняться при нажатии на кнопку. Например, вы можете вывести сообщение, изменить цвет фона, открыть файл и т.д.

Создание приложений



Создание приложений

```
//пример кода функции-обработчика нажатия на кнопку,
//которая выводит сообщение “Hello, world!” в статусную
строку:
void CMFCApplication2Dlg::OnBnClickedButton1()
{
/// TODO: Add your control notification handler code here
AfxGetMainWnd()->SetWindowText(_T("Hello, world!"));
}
```

```
//пример кода функции-обработчика нажатия на кнопку,
//которая меняет цвет фона окна на случайный
void CMFCApplication2Dlg::OnBnClickedButton2()
{
// TODO: Add your control notification handler code here
COLORREF color = RGB(rand() % 256, rand() % 256, rand() % 256);
CBrush brush(color);
CClientDC dc(this);
CRect rect;
GetClientRect(&rect);
dc.FillRect(&rect, &brush);
}
```

Создание приложений

Создайте диалоговое окно с кнопкой "Нажми меня".

Добавьте обработчик события нажатия на кнопку, который выводит сообщение "Кнопка нажата!".

При запуске диалогового окна, при нажатии на кнопку должно появиться сообщение.

Создание приложений

```
void CMFCApplication2Dlg::OnBnClickedButton1()
{
AfxMessageBox(_T("Кнопка нажата!"));
}
```

Создание приложений

Создайте диалоговое окно с текстовым полем и кнопкой "Отправить".

При нажатии на кнопку "Отправить" получите текст из текстового поля и выведите его в диалоговом окне или в консоли.

Убедитесь, что текстовое поле принимает ввод и передает его обработчику события нажатия кнопки.

Создание приложений

```
void CMFCApplication2Dlg::OnBnClickedButton2()
{
CString inputText;
GetDlgItemText(IDC_EDIT1, inputText);
AfxMessageBox(_T("Введенный текст: ") + inputText);
}
```

Создание приложений

Создайте диалоговое окно с выпадающим списком выбора и кнопкой "Выбрать". Заполните список выбора несколькими вариантами.

При нажатии на кнопку "Выбрать" получите выбранный элемент из списка и выведите его в диалоговом окне или в консоли.

Убедитесь, что список выбора работает и передает выбранный элемент обработчику события нажатия кнопки.

Создание приложений

В этой функции OnBnClickedButton3() мы используем указатель на CComboBox для получения доступа к выпадающему списку с идентификатором IDC_COMBO1.
Используем функцию GetCurSel() для получения индекса выбранного элемента и функцию GetLBText() для получения текста выбранного элемента.
Затем используем функцию AfxMessageBox() для вывода сообщения "Выбранный элемент: " вместе с выбранным текстом в диалоговом окне.

```
// Обработчик для кнопки "Выбрать"
void CMFCApplication2Dlg::OnBnClickedButton3()
{
CComboBox* pComboBox = (CComboBox*)GetDlgItem(IDC_COMBO1);
int selectedIndex = pComboBox->GetCurSel();
CString selectedOption;
pComboBox->GetLBText(selectedIndex, selectedOption);
AfxMessageBox(_T("Выбранный элемент: ") + selectedOption);
}
```