**Лекция 1.**

**Windows API . Библиотека Microsoft Foundation Classes.**

Все взаимодействие между любым приложением Windows и самой системой Windows использует программный интерфейс Windows, известный также под назва­нием Windows API (Application Programming Interfase). Он состоит из многих сотен стандартных функций, с помощью которых приложение взаимодействует с операционной системой Windows и наобо­рот. , с помощью которых реализуются все необходимые системные действия, такие как выделение памяти, вывод на экран, создание окон и т.п..

Windows API был разработан в те времена, когда главным используемым языком был С, задолго до появления С++, и по этой причине в нем часто используются структуры вместо классов для передачи некоторого рода данных между Windows и вашим приложением.

Windows API покрывает все аспекты взаимодействия между Windows и вашим приложением. Поскольку количество функций, содержащихся в этом API, очень велико, непосредственное использование их может оказаться достаточно сложным , иногда непросто даже понять, что делает каждая из них. И здесь на помощь разработчику приложений приходит Microsof Visual С++ . Он упаковывает Windows API таким образом, что его функции структурируются в объектно-ориентированной манере, и предлагает облегченный способ применения интерфейса в С++ со значительной долей функциональности по умолчанию. Это принимает форму библиотеки Microsoft Foundation Classes, или MFC.

По скольку библиотека MFC объединяет (путем инкапсуляции) функции API в логически организованное множество классов, интерфейсом становится значительно легче управлять.

Средства автоматизированного создания приложений включены в компилятор Microsoft Visual C++ и называются **MFC AppWizard**. Заполнив несколько диалоговых панелей, можно указать характеристики приложения и получить его тексты, снабженные обширными комментариями. MFC AppWizard позволяет создавать однооконные и многооконные приложения, а также приложения, не имеющие главного окна. Вместо него используется диалоговая панель. Можно также включить поддержку технологии OLE, баз данных, справочной системы.

Конечно, MFC AppWizard не всесилен. Прикладную часть приложения программисту придется разрабатывать самостоятельно. Исходный текст приложения, созданный MFC AppWizard, станет только основой, к которой нужно подключить остальное. Но работающий шаблон приложения - это уже половина всей работы.

MFC AppWizard создает тексты приложений только с использованием библиотеки классов MFC (Microsoft Foundation Class library). Поэтому только изучив язык C++ и библиотеку MFC, можно пользоваться средствами автоматизированной разработки и создавать свои приложения.

MFC – это базовый набор (библиотека) классов, написанных на языке С++ и предназначенных для упрощения и ускорения процесса программирования для Windows. Библиотека содержит многоуровневую иерархию классов, насчитывающую около 200 членов. Они дают возможность создавать Windows-приложения на базе объектно-ориентированного подхода. С точки зрения программиста, MFC представляет собой каркас, на основе которого можно писать программы для Windows.

Библиотека MFC разрабатывалась для упрощения задач, стоящих перед программистом. Как известно, традиционный метод программирования под Windows требует написания достаточно длинных и сложных программ, имеющих ряд специфических особенностей. В частности, для создания только каркаса программы таким методом понадобится около 75 строк кода. По мере же увеличения сложности программы ее код может достигать поистине невероятных размеров. Однако та же самая программа, написанная с использованием MFC, будет примерно в три раза меньше, поскольку большинство частных деталей скрыто от программиста.

Поскольку MFC представляет собой набор классов, написанных на языке С++, поэтому программы, написанные с использованием MFC, должна быть в то же время программами на С++.

Процесс написания Windows-программы включает создание и использование объектов MFC либо объектов классов, унаследованных от MFC.

Объекты этих основанных на MFC типов классов включают методы для взаимодействия с Windows, обработки сообщений Windows, а также отправки сообщений друг другу. Эти производные классы, конечно же, наследуют все члены своих базовых классов. Унаследованные функции выполняют практически всю черновую работу, обеспечивающую работу Windows-приложения. Все, что должны сделать - добавить поля и методы для настройки поведения классов и обеспечения специфичной для приложения функциональности, которая требуется от программы.

**Нотация MFC**

Все классы MFC имеют имена, начинающиеся с С, вроде CDocument или CView. Если вы используете те же соглашения, определяя собственные классы или наследуя их от классов из библиотеки MFC, то ваши программы будут легче восприниматься. Поля классов MFC снабжены префиксом m\_. Желательно следовать этому соглашению в примерах, использующих MFC.

Вы обнаружите, что MFC применяет венгерскую запись для многих имен переменных - в частности тех, что происходят от Windows API. Как вы помните, это предусматривает префикс р для указателей, n - для int, 1 - для long, h - для дескрипторов и так далее. Например, имя *m\_lpCmdLine* ссылается на поле класса (поскольку включает префикс m\_) типа "указатель на long".

Такая практика явного указания типа переменной в ее имени была важна в среде С из-за недостаточного контроля типов; благодаря тому, что можно было определить тип по имени, снижалась вероятность неправильной интерпретации значений. Отрицательной стороной таких имен переменных была некоторая громоздкость, из-за которой код выглядел сложнее, чем он был на самом деле. Поскольку С++ обладает строгим контролем типов, исключающим случаи неверного использования переменных, которые были возможны в С, подобная нотация уже не столь важна, поэтому возможно не применять ее в примерах . Однако рекомендуется оставить префикс р для указателей и некоторые другие простые обозначения типов, поскольку это помогает повысить читабельность кода.

**Понятие диалогов. Элементы управления**

Большинство вещей, которые вы видите на экране под Windows, являются окнами. Типичное диалоговое окно показано на рис.1.



рис. 1

Аннотации показывают разнообразные элементы управления, из которых собирается интуитивно понятный интерфейс для выбора файла, который требуется открыть. Это делает диалог легким в использовании, даже , несмотря на очень широкий диапазон его возможностей.

Для создания и отображения диалогового окна в программе MFC требуются две вещи: физическое появление диалогового окна, которое определено в ресурсном файле, и объект класса диалога, используемый для управления операциями диалога и его элементов управления. MFC предоставляет вам класс по имени CDialog для ис­пользования после того, как вы определяете ваш ресурс диалога.

**Что такое элементы управления?**

В Windows вам доступно множество различных элементов управления, и в боль­шинстве случаев система обеспечивает высокую степень гибкости в отношении их внешнего вида и поведения.

Большинство из этих элементов попадает в одну из шести категорий:

Статические элементы управления . Эти элементы используются для отображения заголовков или описательной информации.

Кнопочные элементы управления . Кнопки представляют механизм ввода "одним щелчком". Существуют три разновидности кнопочных элементов управления: простые нажимаемые кнопки, кнопки-переключатели, из группы которых в каждый отдельный момент времени может быть выбрана только одна, и кнопки-флажки, которые могут пребывать в выбранном состоянии, независимо от соседних флажков в той же группе.

Линейки прокрутки обычно используются для прокрутки текста или графических изображений, горизонтально или вертикально, внутри другого элемента управления. Предоставляют списки выбора, в которых одновременно можно выделить один или более вариантов.

Элементы редактирования, принимающие текстовый ввод или позволяющие редактировать отображаемый текст.

Комбинированные списки предоставляют списки выбора, из которых можно выбирать один вариант, комбинированные с возможностью самостоятельного ввода текста.

Элемент управления может быть или не быть ассоциирован с объектом класса. Статические элементы управления ничего не делают непосредственно, поэтому в ассоциированном классе могут показаться излишними; однако в MFC предусмотрен класс CStatic, который представляет функции, позволяющие вам изменить внешний вид статических элементов управления. Кнопочные элементы управления также во многих случаях могут управляться объектами диалога, но опять-таки MFC предоставляет класс СButton для использования в ситуациях, когда нужен объект класса для управления элементом управления. MFC также предлагает полный набор классов для поддержки всех прочих элементов управления. Поскольку элементы управления явля­ются окнами, все они унаследованы от CWnd.

**Общие элементы управления**

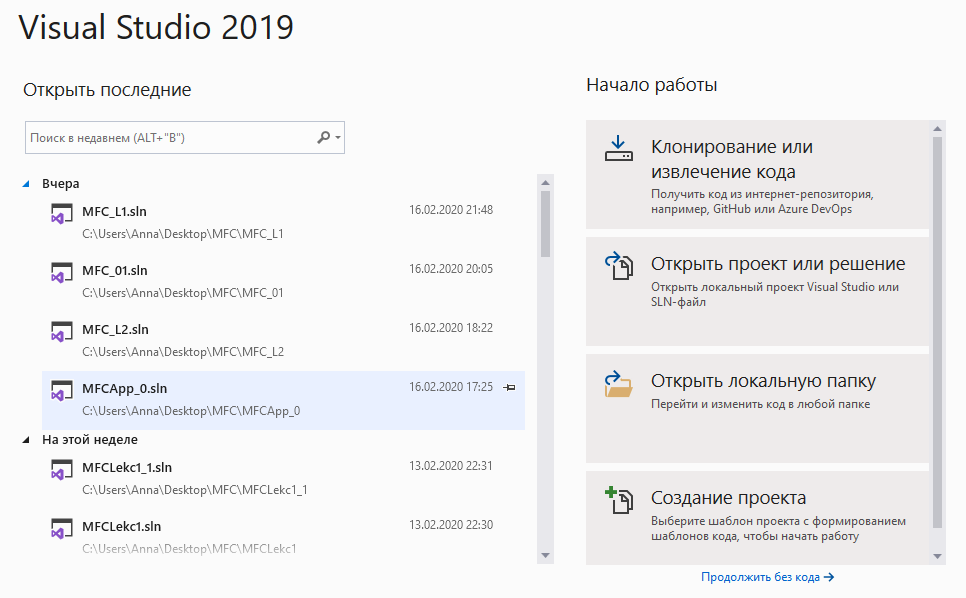
Набор стандартных элементов управления, поддерживаемый MFC и редактором ресурсов, называется общими элементами управления. Общие элементы управления включают все, что вы уже видели, а также ряд других более сложных элементов, таких, например, как анимированные элементы управления, которые обладают способностью воспроизводить файлы AVI (Audio Video Interleaved), и древовидные элементы, которые могут отображать древовидную иерархию элементов.

**Лекция 2.**

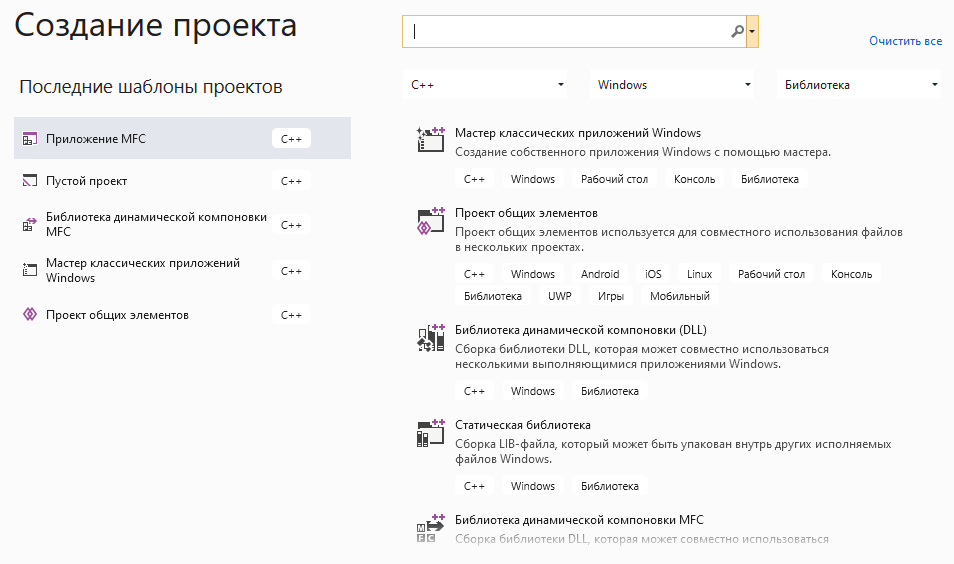
**Создание диалогового приложения**

Приложение, основанное на диалоге – это Dialog Based Application, этапы создания которого следующие.

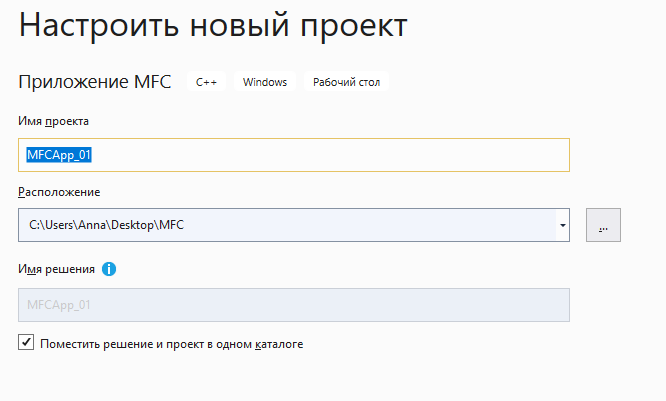
1. Запускаем Visual Studio. Выбираем пункт Создание проекта.



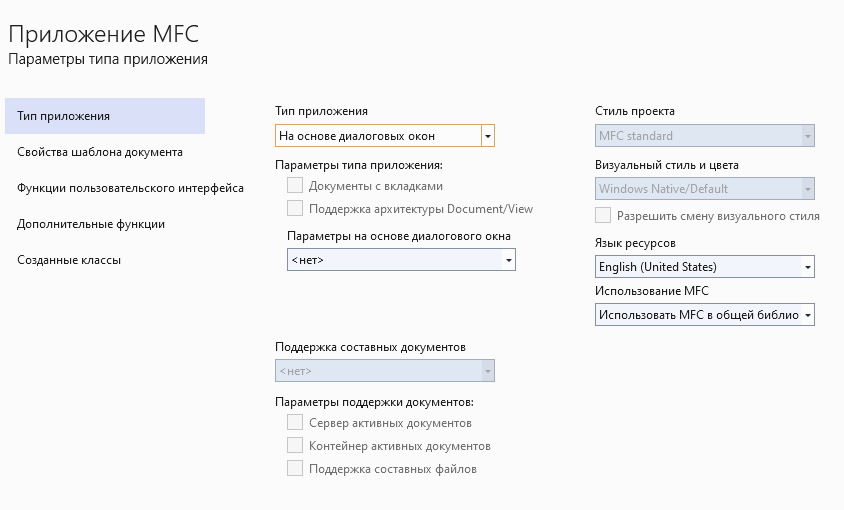
1. Выбираем пункт Приложение MFC.



1. Задаем имя проекта и его расположение.

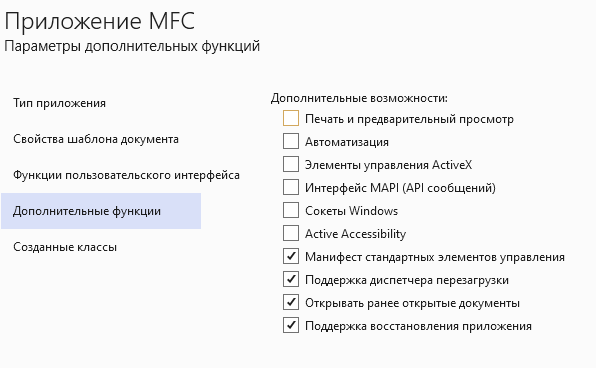


1. Выбираем тип приложения. В нашем случае это На основе диалоговых окон.



Переключатель Use of MFC (Использование MFC) по умолчанию установлен в положение Use MFC in a shared DLL (Использование динамически подключаемых библиотек). В этом случае размер выполняемого файла будет небольшим, но программа не сможет работать на компьютерах, где не установлен пакет Microsoft Visual Studio из-за отсутствия динамически подключаемых библиотек. Если выбрать опцию Use MFC in a static library (Использование статических библиотек), то программа будет легко переносима на другие компьютеры, но размер выполняемого файла будет намного больше, т. к. в него будут включены все необходимые библиотеки. В любом случае, выбор типа библиотек всегда можно изменить в свойствах уже построенного проекта.

1. Выбираем параметры дополнительных функций.



Advanced Features (Дополнительные возможности).

ActiveX controls (Элементы управления ActiveX) — выбор этого флажка дает возможность использовать элементы управления ActiveX (внедренные элементы управления других приложений, например, таблицы Excel);

Windows sockets (Оконные сокеты) — при установке этого флажка приложение может иметь непосредственный доступ к Internet, используя протоколы FTP и HTTP. По умолчанию этот флажок снят;

Active Accessibility (Активная доступность) — установка этого флажка делает доступным работу с объектами COM (Component Object Model, Модель компонентных объектов);

1. Можем изменить имена созанных класов.

На этом работа мастера MFC AppWizard завершается, каркас приложения создан.

**Этапы работы MFC приложения.**

В результате работы MFC приложения, были сгенерированы классы:

- класс приложения;

- класс диалового окна.

Класс приложения отвечает за управление программы в целом.

Каждая программа , созданная с помощью приложения MFC, должна содержать функцию WinMain(), которая вызывается ОС в момент запуска программы. Эта функция невидима для программиста и находится внутри библиотеки MFC и одинакова для всех программ.

В момент старта программы, роисходит создание глобальных объектов, в частности theApp, (т.е. до вызова функции Win Main). Вызывается конструктор, в котором происходит начальная инициализация объектов. Это и обеспечивает отличие одной прогаммы от другой, несмотря на то, что WinMain одинакова для всех программ.

**Работа функции WinMain**.

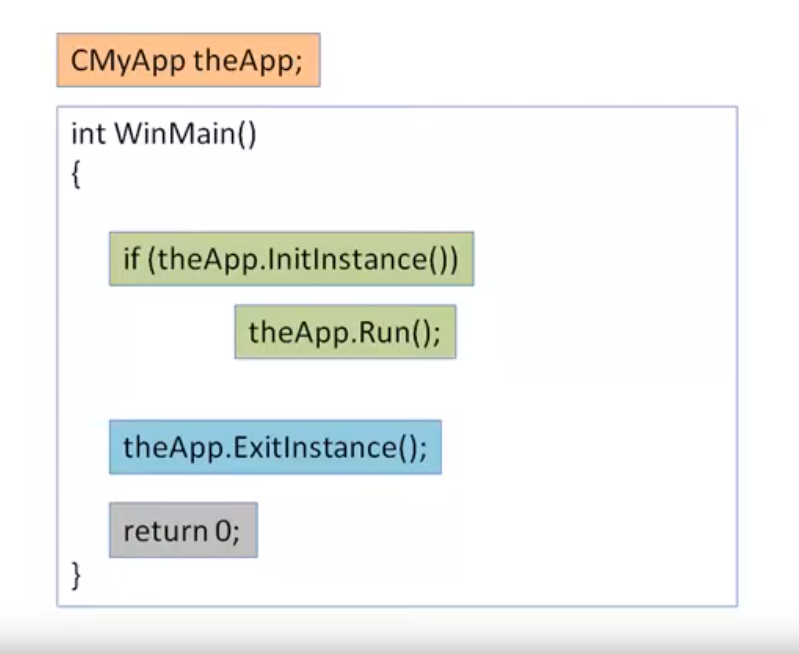
В первую очероед функция производит инициализацию библиотек MFC.

Затем работу функции WinMain можно разделить на 3 этапа:

1. Вызов функции InitInstance() для класса нашего приложения.

Функция InitInstance() возвращает значение типа BOOL (TRUE или FALSE).

1. Если возвращаемое значение равно TRUE, то вызывается функция RUN(), где происходит обработка событий программы.
2. Вызов функции ExitInstance(), происходит деинициализация, т.е. освобождение ресурсов .



**Функция InitInstance()** производит инициализацию программы с учетом специфики ее работы.

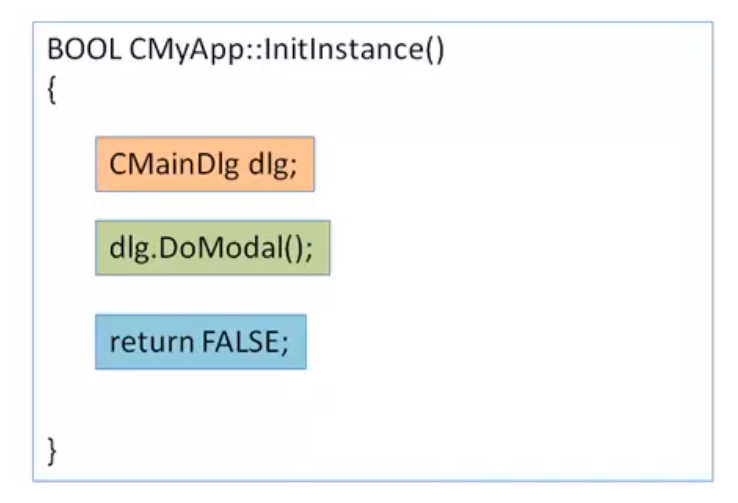
С каждой программой под Windows должно быть связано хотя бы одно окно.

Этим окном в нашем случае является главный диалог .

- В функции **InitInstance()** создается объект этого класса. Вызавается конструктор этого объекта. Но диалог еще не отбражается.

- Вызов функции DoModal() осуществляет отбражение диалога. Завершается после закрытия диалового окна.

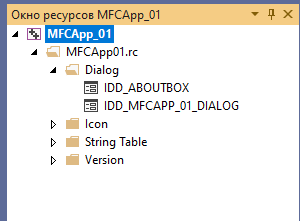
- возврат в функцию WinMain значения FALSE означает, что функцию RUN() вызывать не нужно. Можно завершать пограмму.



**Лекция 3.**

**Проектирование диалоговой панели**

Спроектируем диалоговую панель. Для этого вначале выбирете закладку "Resourse View" (Окно ресурсов) .



Здесь вы видите две записи:

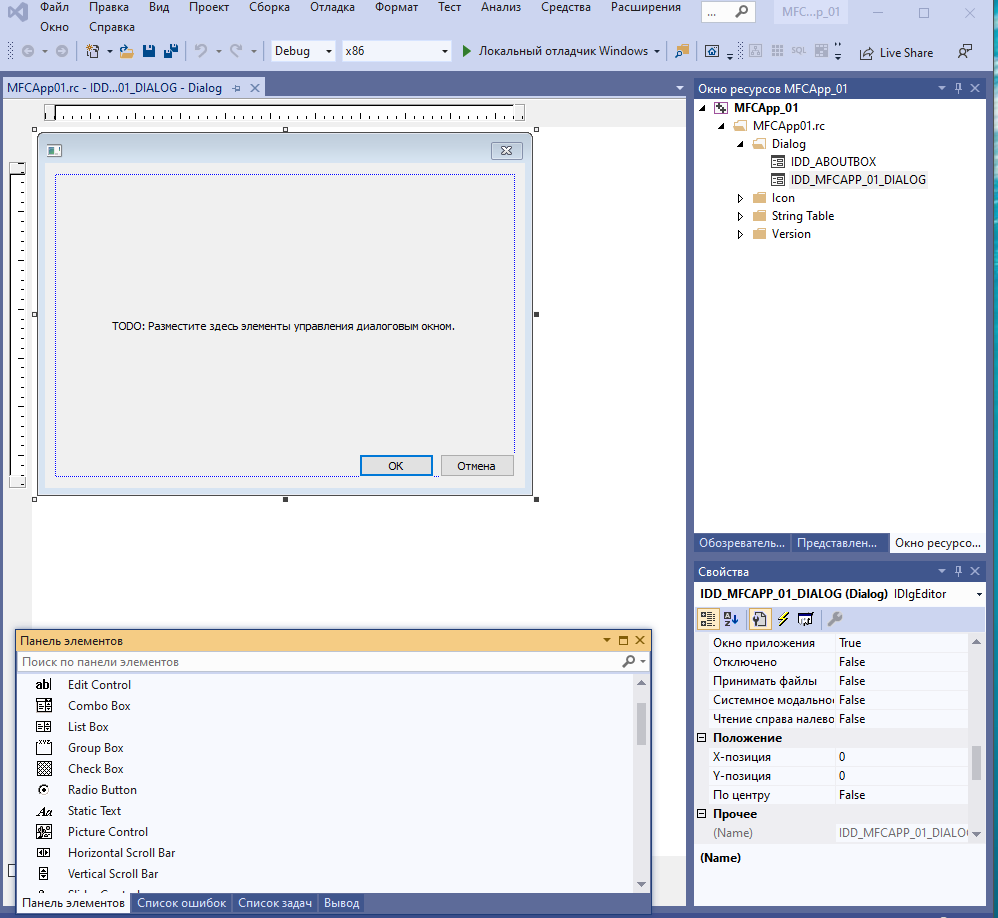
1) IDD\_ABOUTBOX

2) IDD\_MFCAPP\_01\_DIALOG

1) - это название диалоговой панели ABOUT, а

2)- название гланой диалоговой панели. Выполните щелчок по второй строчке(2) и справа появится, диалоговая панель. В этом режиме вы можете ее редактировать .

Стандартная диалоговая панель (рис. 2), создаваемая средой по умолчанию, включает в себя строку заголовка с кнопкой закрытия окна, статический текст «TODO: place dialog controls here» и две кнопки – OK и Cancel; обе кнопки выполняют по умолчанию одно и то же действие – завершение программы.



Удалите из диалоговой панели текст: "TODO:.." и две кнопки, для этого щелкните на эти элементы и нажмите DEL. Теперь у вас чистая панель и вы можете спроектировать ее на свой вкус.

**Использование в приложении элементов управления**

Элементы управления используются в приложении для создания графического интерфейса пользователя. К элементам управления относятся статические тексты, кнопки, радиокнопки, флажки, окна редактирования, списки, изображения, а также более сложные элементы управления, например, календарь. Перечень элементов управления постоянно пополняется новыми элементами.

Наиболее часто используемые элементы управления собраны в панели Controls в виде кнопок и добавляются в проект визуально с помощью мыши:

* рисунок (Picture),
* статический текст (Static Text),
* окно редактирования (Edit Box),
* группа (Group Box),
* кнопка (Button),
* флажок (Check Box),
* радиокнопка (Radio Button),
* комбинированный список (Combo Box),
* окно со списком (List Box),
* линейный регулятор (Slider).
* линейный индикатор (Progress bar).

Сначала на панели Controls мышью выбирается элемент, затем щелчком левой кнопкой мыши выбранный элемент помещается в указанном месте в пределах панели диалога. Элементы, помещенные в диалоговую панель, могут быть выделены по одному или группой, выделенные элементы можно удалить с панели диалога или переместить в ее пределах.

Контекстное меню элемента управления содержит команду Properties (Свойства), с помощью которой можно настроить внешний вид элемента. Окно свойств имеет три вкладки: General – здесь собраны свойства, отвечающие за функционирование элемента; Styles – набор оригинальных свойств, характерных для данного элемента; Extended Styles – универсальный набор стандартных свойств.

Рассмотрим наиболее общие свойства элементов управления.

В разделе General для элементов определяются значения следующих свойств:

* ID – идентификатор элемента управления, используется для обращения к элементу в программе, объявление идентификатора помещается в файл resource.h с помощью директивы define;
* Caption – строка текста, определяющая надпись на элементе;
* Visible – определяет, будет ли элемент отображаться на экране в начале работы программы, тип свойства – логический, по умолчанию имеет значение TRUE;
* Disabled – определяет возможность использования элемента. Если свойство имеет значение TRUE, элемент отображается на экране, но не может быть активным; по умолчанию свойство имеет значение FALSE;
* Group – определяет первый элемент в группе, имеет логический тип; при формировании группы это свойство может быть установлено только у одного элемента из группы (см. радиокнопки);
* Tab Stop – определяет возможность обхода элемента при помощи клавиши Tab, тип логический; если свойство имеет значение TRUE, данный элемент
* может быть активизирован нажатием клавиши Tab, иначе при повторных нажатиях Tab элемент будет обходиться стороной.

Наборы свойств, собранных в разделе Styles, сильно отличаются для различных типов элементов, поэтому этот раздел будет рассмотрен позже для каждого типа элемента отдельно. В этот раздел обычно включаются свойства, определяющие выравнивание текста в элементе (alignment), размещение текста в нескольких строках (multiline).

В разделе Extended Styles находятся следующие свойства:

Client edge – устанавливает утопленную рамку вокруг элемента управления;

Static edge – устанавливает обычную рамку вокруг элемента управления;

Modal frame – устанавливает объемную рамку вокруг элемента управления;

Transparent – устанавливает прозрачность элемента.

По назначению элементы управления делятся на:

* элементы, используемые для отображения (вывода) информации, например, статический текст;
* элементы, используемые для сбора (ввода) информации, например, окно редактирования;
* элементы, используемые для управления информацией, например, кнопки, флажки.

Все элементы управления представлены в программе объектами соответствующих классов. Так, для создания статической строки используется класс CStatic, окна редактирования – класс CEdit, кнопок – CButton. Все классы, представляющие собой элементы управления, являются потомками класса окна CWnd, тогда все элементы управления являются оконными объектами. Как и для всех оконных объектов, процесс создания включает в себя два этапа:

создание элемента управления (например, размещение его в шаблоне диалога);

связывание его с объектом или переменной.

Если элемент управления не требуется изменять в процессе выполнения приложения, то для него можно не выполнять связывание с объектом или с переменной.

Элементы управления могут хранить некоторые значения. С такими элементами управления в программе связываются специальные переменные, которые представляют свой элемент управления в программе. Эти переменные имеют тот же тип, что и значения, присвоенные элементам управления. Например, окно редактирования хранит строку текста типа CString, в программе представителем этого элемента управления будет переменная того же типа. Связывание элементов управления и переменных в программе осуществляется с помощью мастера MFC ClassWizard, вкладка Member Variables, при этом могут быть созданы переменные двух категорий: значение (value) или элемент управления (control).

Разница между категориями переменных в том, что для категории переменной control происходит автоматическое обновление значения переменной, если изменилось значение в элементе управления, и, наоборот, если во время выполнения программы изменилось значение переменной, связанной с элементом управления, автоматически обновляется значение, хранимое элементом управления. Элемент управления в таком случае связывается с объектом класса, соответствующего данному элементу управления. Для категории переменных типа value обновление значений переменных и элементов управления производится с помощью функции UpdateData(), и связывание выполнение с переменной базового типа, либо объектом класса CString. С другой стороны, такие элементы, как кнопки (button), не хранят никаких значений, используются лишь для вызова какой-либо функции и не имеют связанных переменных.

**Мастер классов MFC ClassWizard**

Мастер классов MFC ClassWizard используется для формирования служебной части кода класса и для редактирования класса. Простейший способ вызывать Мастер классов – это воспользоваться контекстным меню какого-либо элемента управления.

Окно MFC ClassWizard имеет несколько вкладок Message Maps, Member Variables, Class Info и др.

Message Maps – карта сообщений, связывает в программе сообщение, объект, реагирующий на это сообщение, и функцию, которая выполняется в ответ на это сообщение. При возникновении какого-либо сообщения, например, BN\_CLICKED, вследствие нажатия кнопки IDC\_BUTTON\_MSG вызывается метод OnButtonMsg(); если бы это же сообщение возникло при нажатии другой кнопки, был бы вызван другой метод. При помощи окна Message Maps можно добавить класс (кнопка Add Class…), функцию (Add Function…), отредактировать код функции (Edit Code…), удалить функцию (Delete Function…).

Member Variables – связывает элементы управления, хранящие какие-либо данные, и переменные, которые являются представителями своих элементов управления в программе. Добавление переменной для элемента управления выполняется в окне Member Variables с помощью кнопки Add Variable…; имя переменной вводится в специальное окно панели Add Member Variable, в начале имени должны быть символы "m\_", затем добавляется строка текста, поясняющая связь переменной с элементом управления и ее назначением, например, для окна редактирования IDC\_EDIT\_RESULT может быть создана переменная с именем m\_Result или m\_Edit\_Result. Для переменной также должен быть установлен тип значения и категория переменной. Категория переменной определяет характер связи между переменной и элементом управления и может принимать одно из двух значений – value или control. . Value - это категория переменной, предназначенная для передачи значения переменной. Control - а это категория предназначена для управления элементами. Если переменной присвоена категория control, происходит связывание элемента управления с объектом соответствующего класса. Если переменной присвоена категория value, автоматического обновления значений не происходит, обновление выполняется функцией UpdateData(). Каждый элемент управления имеет свой набор атрибутов, называемых также свойствами элемента управления

Следует правильно завершить добавление переменной – нажать кнопку OK для закрытия окна Add Member Variable и кнопку OK для закрытия окна ClassWizard. Игнорирование этих двух кнопок приведет к тому, что переменная не будет добавлена. Обнаружится такая ошибка при компиляции проекта – имя переменной будет неизвестным. Повторное добавление переменной через ClassWizard исправит ошибку.

Class Info – служит для просмотра списка классов, их размещения в файлах и добавления в проект новых классов.

**Установка начального значения элементам управления**

Так как набор свойств, устанавливаемых для элементов управления перед запуском программы через окно Properties, ограничен, можно устанавливать начальные значения свойств элементам стандартной функцией OnInitDialog(), которая выполняется один раз при создании диалоговой панели. Редактирование этой функции выполним через ClassWizard следующим образом.

* Вызовем ClassWizard, откроем вкладку Message Maps;
* В окне Project должно быть установлено имя проекта Lab1, в окне Class name – имя класса диалоговой панели CLab1Dlg;
* В окне Object IDs выберем объект – диалоговую панель CLab1Dlg; в окне Messages найдем сообщение WM\_INITDIALOG, выделим его, при этом автоматически будет выделена в окне Functions функция OnInitDialog(), так как она уже входит в состав класса диалоговой панели;
* Нажмем кнопку Edit Code. После нажатия кнопки в окне редактора откроется модуль, содержащий код этой функции:

*BOOL CLab1Dlg::OnInitDialog()*

*{*

*CDialog::OnInitDialog();*

*………………*

*//TODO: Add extra initialization here*

*return TRUE;*

*}*

Отредактируем код функции OnInitDialog, добавим следующие строки (они будут выделены жирным шрифтом), место вставки нового кода указывает текст //TODO: Add extra initialization here (добавьте дополнительную инициализацию здесь):

*BOOL CLab1Dlg::OnInitDialog()*

*{*

*CDialog::OnInitDialog();*

*………………*

*//TODO: Add extra initialization here*

*m\_Result=" initial text";* //Установка начального значения переменной

*UpdateData(FALSE);* //Обновление экрана

*return TRUE;*

}

После запуска такой программы окно редактирования будет содержать слова «*initial text*».

**Элемент управления рисунок (Picture)**

Изображение (Picture) – это группа элементов управления, предназначенных для представления графической информации в рамках приложения. С помощью этой группы элементов в диалоговую панель можно вставить: пиктограмму (Icon), растровый рисунок (Bitmap), метафайл Windows (Enhanced Metafile), прямоугольную рамку – кадр (Frame), заполненный прямоугольник (Rectangle).

Рассмотрим порядок размещения изображения на диалоговой панели, для определенности зададим, что наш рисунок представляет собой растровый файл my.bmp:

· скопируйте файл с рисунком в каталог проекта RES;

· откройте вкладку Resource View в окне Workspace, вызовите контекстное меню для папки Lab1 Resources и вызовите команду Import.

· в открывшемся окне найдите и выделите файл my.bmp, затем нажмите кнопку Import; на диаграмме ресурсов в Resource View появится новая папка Bitmap, а наш рисунок будет добавлен в проект под именем IDB\_BITMAP1;

· поместите на диалоговой панели элемент управления Picture;

· настройте свойства элемента управления – на вкладке General в окне Type укажите тип изображения – Bitmap, в окне Image выберите ресурс – IDB\_BITMAP1. Идентификатор элемента управления ID – IDC\_STATIC – можно оставить без изменения. Размещение рисунка в диалоговой панели закончено.

**Элемент управления окно редактирования (Edit Box)**

Класс CEdit обеспечивает функционирование элемента управления "окно редактирования", называемое также текстовым полем. Окно редактирования может быть как однострочным, так и многострочным. Иногда однострочное окно редактирования также называют полем ввода.

Для создания многострочного окна редактирования следует установить значение свойства Multilin равным True. Для установки разбиения на строки с использованием мягкого конца строки следует установить свойство Auto HScroll.

При мягком конце строки, в отличие от жесткого, окно редактирования отображает разделение на строки, не вставляя непосредственно в сам текст символы конца строки и перевода каретки.

При двойном щелчке мыши на элементе управления выполняется автоматическая вставка метода обработчика сообщения OnEnChangeEdit.

Класс CEdit предоставляет широкий набор методов для работы с окном редактирования, включая следующие:

CanUndo - метод возвращает ненулевое значение, если последнее изменение в окне редактирования можно отменить, и 0 - если изменение отменить нельзя.

CharFromPos- метод возвращает номер символа в строке и строки (начиная с 0) для символа, наиболее близко расположенного к указанной параметром точке.

Clear - метод удаляет в окне редактирования текущее выделение текста.

Copy - метод копирует текущее выделение текста из окна редактирования в буфер промежуточного хранения.

Cut - метод удаляет текущее выделение текста из окна редактирования и копирует его в буфер промежуточного хранения.

Paste - метод выполняет вставку данных из буфера промежуточного хранения в текущую позицию окна редактирования.

GetFirstVisibleLine - метод возвращает номер первой отображаемой строки в окне редактирования.

GetLine - метод копирует указанную строку текста (без нулевого символа) из окна редактирования в буфер и при успешном завершении возвращают количество скопированных байтов.

GetLineCount - метод возвращает количество строк текста в многострочном окне редактирования.

GetSel - метод определяет позицию первого и последнего символа выделенного фрагмента текста.

LimitText - метод устанавливает максимально допустимую длину (в байтах) вводимого пользователем текста.

LineIndex - метод возвращает номер первого символа в указанной строке, а при значении параметра, равном -1 - номер первого символа в текущей строке.

LineLength - метод возвращает длину строки.

ReplaceSel - метод выполняет замену выделенного фрагмента текста на указанную строку.

SetPasswordChar - метод устанавливает символ, отображаемый при вводе пароля, вместо любого вводимого пользователем символа.

SetSel - метод выполняет выделение в окне редактирования указанного фрагмента текста.

**Элемент управления статический текст (Static Text)**

Класс CStatic реализует работу со статическим текстом (меткой). Этот элемент управления предназначается для отображения различных надписей и не может непосредственно редактироваться пользователем. Содержание элемента управления Static Box определяется на этапе проектирования свойством Caption и в дальнейшем может изменяться только программным путем.

Выполнить связывание элемента управления "статический текст" можно с переменной типа CString или объектом класса CStatic.

Свойство Border определяет, что вокруг элемента управления будет отображаться рамка.

Свойство Transparent определяет, является ли фон элемента прозрачным.

**Элементы управления кнопка (Button),**

**флажок (Check Box), радиокнопка (Radio Button)**

Класс CButton применяется для работы со следующими элементами управления:

* командная кнопка;
* флажок;
* радиокнопка (переключатель).

Элемент управления "кнопка", называемый также командная кнопка, как правило, используется для обработки сообщения BN\_CLICKED.

Свойство Default Button позволяет указать командную кнопку, устанавливаемую как кнопка по умолчанию: нажатие пользователем клавиши Enter интерпретируется как щелчок на данной кнопке.

Свойство Icon позволяет указать, что вместо текста будет отображена пиктограмма (стиль BS\_ICON).

Свойство Bitmap позволяет указать, что вместо текста будет отображено изображение (стиль BS\_BITMAP).

Свойство Multiline используется в том случае, если текст слишком длинный, чтобы уместиться на кнопке в одну строку.

Для работы с элементом управления "кнопка" следует использовать класс CButton.

Элемент управления "флажок Check Box" может иметь два или три состояния: включенное, выключенное или неопределенное (необязательно).

Кроме возможности связывания флажка с объектом типа CButton, флажок можно связать с переменной типа BOOL.

Свойство Auto позволяет создавать элемент управления, переключение состояний (включен/выключен) которого происходит автоматически при щелчке мышью.

Свойство Tri-state используется для создания элемента управления "флажок", имеющего три состояния. Одновременно с состояниями "включен" или "выключен", используется состояние "неопределен", в котором флажок выглядит "посеревшим". Если значение свойства Push Like установлено равным True, то создается элемент управления "флажок", отображаемый как командная кнопка: при нажатии флажок имеет вид вдавленной кнопки, а при ненажатом состоянии - выпуклой.

Свойство Flat определяет, будет ли создаваемый элемент управления иметь вид плоской кнопки.

Элемент управления "радиокнопка" (Radio Button) аналогичен флажку, но не может иметь неопределенного состояния. Также при объединении несколько радиокнопок в группу только одна из них может иметь включенное состояние.

Кроме возможности связывания флажка с объектом типа CButton, радиокнопку можно связать с переменной типа BOOL.

При использовании класса CButton тип кнопки можно определить ее стилем, указываемым в методе Create при создании кнопки. Класс CButton наследуется классом CBitmapButton, реализующим кнопки с изображениями вместо текста.

Создать кнопку можно как с использованием редактора ресурсов, так и непосредственно программным путем.

Если объект CButton создается расположенным в диалоговом окне, то он автоматически разрушается при закрытии пользователем этого диалогового окна.

Если же объект был создан динамически вызовом метода new, то для его разрушения следует вызвать метод delete.

Класс CButton предоставляет ряд методов, включая следующие:

Create - метод создает Windows-кнопку для объекта CButton и при успешном завершении возвращает ненулевое значение. При вызове метода кнопке могут быть присвоены следующие флажки стиля (оконного объекта):

WS\_CHILD - устанавливается всегда;

WS\_VISIBLE - видимая кнопка;

WS\_DISABLED - недоступная кнопка;

WS\_GROUP - используется для образования групп кнопок: устанавливается для первой кнопки группы;

WS\_TABSTOP - устанавливается для включения кнопки в табулированный порядок.

GetCheck - метод определяет состояние кнопки (флажка или радиокнопки), созданной как BS\_AUTOCHECKBOX, BS\_AUTORADIOBUTTON, BS\_AUTO3STATE, BS\_CHECKBOX, BS\_RADIOBUTTON, BS\_3STATE, и возвращает одно из следующих значений:

0 - кнопка не включена (не отмечена);

1 - кнопка включена (отмечена);

2 - кнопка находится в неопределенном состоянии (только для кнопок, имеющих флажки стиля BS\_3STATE или BS\_AUTO3STATE).

Для командных кнопок метод возвращает значение 0.

GetState - метод определяет текущее состояние кнопки. Возвращаемое значение формируется как комбинация набора значений, выделяемых с помощью следующих масок:

0x0003 - определяет состояние кнопки-флажка или радиокнопки:

0 - кнопка не включена (не отмечена);

1 - кнопка отмечена;

2 - состояние кнопки не определено.

0x0004 - определяет состояние выделения кнопки: при значении 0 - кнопка не нажата (пользователь щелкнул на ней левой кнопкой мыши и держит ее);

0x0008 - определяет фокус: при значении 1 - кнопка находится в фокусе.

SetBitmap- метод "растровое изображение, отображаемое на данной кнопке".

**Элемент управления группа( Group Box)**

Группа или групповой блок (Group Box) – это элемент управления, предназначенный для группировки других элементов управления, например, радиокнопок, служит для отображения информации. Используется в комбинации с другими элементами управления.

**Элемент управления окно со списком (List Box)**

Элемент управления List Box "окно списка", называемый иногда просто списком, используется для работы с информацией, отображаемой в виде списка. Окно списка может быть реализовано как список с единичным или с множественным выбором.

Окно списка можно связать как с переменной типа CString, так и с объектом класса CListBox.

Свойство Selection позволяет устанавливать тип списка:

Single (по умолчанию) - пользователь может выделить только один элемент списка

Multiple - пользователь может одновременно выделить несколько элементов списка. Выделение элементов (или снятие выделения) осуществляется при щелчке или двойном щелчке мыши.

Extended - пользователь может одновременно выделить несколько элементов списка. Выделение элементов (или снятие выделения) осуществляется перемещением мыши при нажатой кнопке. None - пользователь не может выделить ни одного элемента списка.

Класс CListBox предоставляет широкий набор методов для работы со списком, включая следующие:

GetCount - метод возвращает количество элементов в окне списка.

SetTopIndex - метод прокручивает окно списка к указанному элементу, отображая его первым видимым элементом.

GetText- метод возвращает строку по указанному индексу элемента списка.

SetCurSel- метод выделяет указанный элемент списка и при необходимости прокручивает окно списка так, чтобы выделенный элемент стал видимым.

GetSelCount - метод возвращает общее количество выделенных элементов в окне списка с множественным выбором.

AddString - метод добавляет в окно списка новый элемент, содержащий указанную строку.

DeleteString - метод удаляет из окна списка строку с соответствующим индексом.

InsertString - метод вставляет в указанное место окна списка новый элемент, содержащий заданную строку.

FindString - метод выполняет поиск указанной строки в окне списка.

ResetContent - метод удаляет все элементы окна списка.

**Элемент управления комбинированный список (Combo Box)**

Класс CComboBox реализует функционирование комбинированного окна, иногда также называемого комбинированным окном списка.

Комбинированное окно представляет собой элемент управления, объединяющий окно списка с окном редактирования или со статическим элементом управления. Окно списка при этом может иметь как распахнутое (отображаемое постоянно), так и свернутое состояние (отображаемое только при щелчке пользователя на стрелке вниз).

Выбранный элемент списка отображается в окне редактирования или в окне статического элемента управления.

Метод GetLBText определяет строку текста, соответствующую указанному индексу элемента окна списка комбинированного окна.

Метод GetCurSel возвращает индекс выделенного элемента окна списка комбинированного окна.

**Элемент управления линейный индикатор (Progress bar).**

Линейный индикатор представляет собой прямоугольник, заполняющийся синими квадратиками (блоками). Чем больше заполнен прямоугольник, тем ближе завершение процесса. Обычно этот элемент управления используют в качестве индикатора хода выполнения некоторого процесса в приложении, например загрузка или сохранение файла.

Класс CProgressCtrl реализует функционирование линейного индикатора

Класс CProgressCtrl имеет ряд методов, которые позволяют инициализировать элемент управления и манипулировать им.

Create() Создает элемент управления.

OffsetPos() Продвигает зону заполнения на заданное количество блоков.

SetPos() Устанавливает текущую величину параметра элемента.

SetRange() Устанавливает максимальное и минимальное значения параметра элемента.

SetStep() Устанавливает приращение параметра элемента, который соответствует одному блоку.

StepIt() Добавляет один блок в зону заполнения.

**Элемент управления линейный регулятор (Slider Control)**

Когда пользователь должен ввести некоторую величину, не выходящую за пределы заданного диапазона используется линейный регулятор.

Класс CSliderCtrl, реализующий функционирование линейного регулятора, имеет ряд методов, которые позволяют инициализировать элемент управления и манипулировать им.

Create() Создает элемент управления.

ClearSel() Очищает выполненную ранее установку регулятора

ClearTics() Очищает установленные пометки регулятора

GetChannelRect() Считывает размер регулятора элемента управления

GetLineSize() Считывает установленное значение приращения параметра элемента управления в ответ на нажатие клавиши управления курсором "вверх" и "вниз".

GetNumTics() Считывает количество пометок элемента управления

GetPageSize() Считывает размер страницы элемента управления (значение приращения параметра при управлении клавишами <PgUP> и <PgDn>

GetPos() Считывает текущее значение параметра регулятора

GetRange() Считывает диапазон значений параметра элемента управления (максимальное и минимальное значения)

GetRangeMax() Считывает максимальное значение диапазона допустимых значений параметра элемента управления

GetRangeMin() Считывает минимальное значение диапазона допустимых значений параметра элемента управления

GetSelection() Считывает текущий выбор диапазона допустимых значений параметра элемента управления.

GetThumbRect() Считывает размер прямоугольника бегунка элемента управления

GetTic() Считывает положение ближайшей пометки элемента управления.

GetTicArray() Считывает положение всех пометок элемента управления

GetTicPos() Считывает координаты пометки элемента управления

SetLineSize() Устанавливает значение приращения параметра элемента управления в ответ на нажатие клавиши управления курсором "вверх" и "вниз"

SetPageSize() Устанавливает значение приращения параметра элемента управления в ответ на нажатие клавиши управления курсором <PgUP> и <PgDN

SetPos() Устанавливает положение регулятора элемента управления

SetRange() Устанавливает диапазон значений параметра элемента управления (минимальное и максимальное и значение)

SetRangeMax() Устанавливает максимальное значение диапазона допустимых значений параметра элемента управления

SetRangeMin() Устанавливает минимальное значение диапазона допустимых значений параметра элемента управления

SetSelection() Устанавливает поддиапазон допустимых значений параметра элемента управления

SetThumbRect() Устанавливает размер прямоугольника бегунка элемента управления

SetTic() Устанавливает положение пометки элемента управления

SetTicFreq() Устанавливает интервал между пометками элемента управления

Если вы включили в свое приложение линейный регулятор, то должны установить диапазон значений и интервал между пометками.

Если предусматривается возможность управления регулятором с помощью клавиш, то необходимо установить значение приращения параметра элемента управления в ответ на нажатие клавиш управления курсором "вверх" и "вниз" и размер страницы элемента управления, при управлении клавишами <PgUP> и <PgDn> .