ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ

«Визуализация» - отображение данных или процессов с помощью графических примитивов → «Визуальное программирование» - это подход, подразумевающий создание кода программы с помощью манипулирования графическими объектами.

На практике в этот термин - иное понятие: визуальное построение интерфейса программы, а не самого кода.
Создаются только заготовки необходимых функций и процедур, а задача создания тела функций по-прежнему - на разработчика программы.

В данном курсе под «Визуальным программированием» - последнее определение, т.е. в качестве объекта визуализации рассматривается *процесс построения интерфейса приложения путем размещения на формах компонентов и настройкой их свойств и поведения*.

Среда программирования: Visual Basic, Delphi, Borland Builder, Microsoft Visual C++, **Visual Studio** и др**.**

Объект - основная единица в ООП, объединяет в себе описывающие его свойства и действия объекта (процедуры) - методы.

Проект - результат процессов программирования и проектирования, объединяет программный код и графический интерфейс. Включает программные модули форм и самостоятельные программные модули в виде отдельных файлов. Проект может быть запущен на выполнение только из системы ООП.

Событие - изменение некоторого состояния, распознаваемое объектом. Для реакции на это изменение могут быть описаны некоторые методы - обработчики, обрабатывающие события в программном коде.

Обработчик события - процедура, которая начинает выполняться после реализации определенного события (щелчка).

Приложение интегрирует программный код и графический интерфейс в одном исполнимом файле, он может запускаться непосредственно в ОС.

Т.е., проект работает только из системы ООП и состоит из нескольких файлов, а приложение работает из ОС и состоит из одного файла

Этапы создания проектов и приложений в системах ООП

Создание графического интерфейса проекта. На форму помещаются элементы управления, обеспечивающие взаимодействие проекта с пользователем.

Установка значений свойств объектов графического интерфейса. В режиме конструирования задаются значения свойств формы и элементов управления, помещенных ранее на форму.

Создание и редактирование программного кода. Создаются заготовки обработчиков событий (двойной щелчок мышью по элементу - вызов заготовки обработчика наиболее часто используемого события для этого элемента). В редакторе программного кода производится ввод и редактирование программного кода обработчиков событий.

Сохранение проекта. Так как проекты включают в себя несколько файлов, рекомендуется для каждого проекта создать отдельную папку на диске.

Компиляция проекта в приложение. Сохраненный проект может выполняться только в самой системе программирования. Чтобы преобразовать проект в приложение, которое может выполняться непосредственно в среде ОС, необходимо выполнить компиляцию проекта, в процессе которой приложение сохранится в исполнимом файле (.exe).

Решаемая на ПК задача реализуется в виде прикладной программы - приложения. В основе разработки приложения лежит **проект**. Центральная часть проекта **- форма**, на нее помещаются необходимые для решения конкретной задачи **компоненты**.

Последовательность создания приложения:
проект - формы - компоненты.

Приложение собирается из элементов: форм, программных модулей, внешних библиотек, картинок, пиктограмм и др. Каждый элемент размещается в отдельном файле и имеет строго определенное назначение.
Набор всех файлов, необходимых для создания приложения, называется проектом .
Компилятор последовательно обрабатывает файлы проекта и строит из них выполняемый файл.

Типы основных файлов проекта:

- ***Файлы описания форм*** - текстовые файлы, описывающие формы с компонентами. В этих файлах запоминаются начальные значения свойств, установленные в окне свойств.

- ***Файлы программных модулей*** - текстовые файлы, содержащие исходные программные коды. В них пишут методы обработки событий, генерируемых формами и компонентами.

-***Главный файл проекта*** - текстовый файл, содержащий главный программный блок.

Файл проекта подключает все используемые программные модули и содержит операторы для запуска приложения. Этот файл среда создает и контролирует сама.

Процесс создания приложения от постановки задачи до получения готового выполняемого файла:

Назначение и внутреннее устройство файлов проекта

1. Файлы описания форм

- текстовый файл, описывающий форму. В нем сохраняются значения свойств формы и ее компонентов, установленные в окне свойств во время проектирования приложения. Количество файлов равно количеству используемых в приложении форм

(если используется только одна форма, то файл только один).

2. Файлы программных модулей

Каждой форме в проекте соответствует свой программный модуль (unit), содержащий все относящиеся к форме объявления и методы обработки событий, написанные на ЯП.

Программные модули размещаются в отдельных файлах (.PAS,.CPP). Их количество может превышать количество форм, так как программные модули могут и не относиться к формам, а содержать вспомогательные процедуры, функции, классы и др.

3. Главный файл проекта

Чтобы компилятор знал, какие конкретно файлы входят в проект, необходимо организующее начало - это ***файл проекта*** - главный программный файл на ЯП, который подключает все файлы модулей, входящих в проект.

Для каждого проекта существует только один такой файл.

По команде **File / New / Application** начинается разработка нового приложения, среда автоматически создает файл проекта.

По мере создания новых форм содержимое этого файла видоизменяется автоматически.

После окончания работы в файле будет находиться перечень программных модулей, которые будут поданы на вход компилятору.

Просмотр содержимого файла приложения – команда **Project / View Source**. В редакторе кода появится новая страница с текстом.

В проект могут входить логически автономные элементы: точечные рисунки (BMP-файлы), значки (ICO-файлы), файлы справки (HLP-файлы) и т.п., но ими управляет сам программист.

Управление проектом

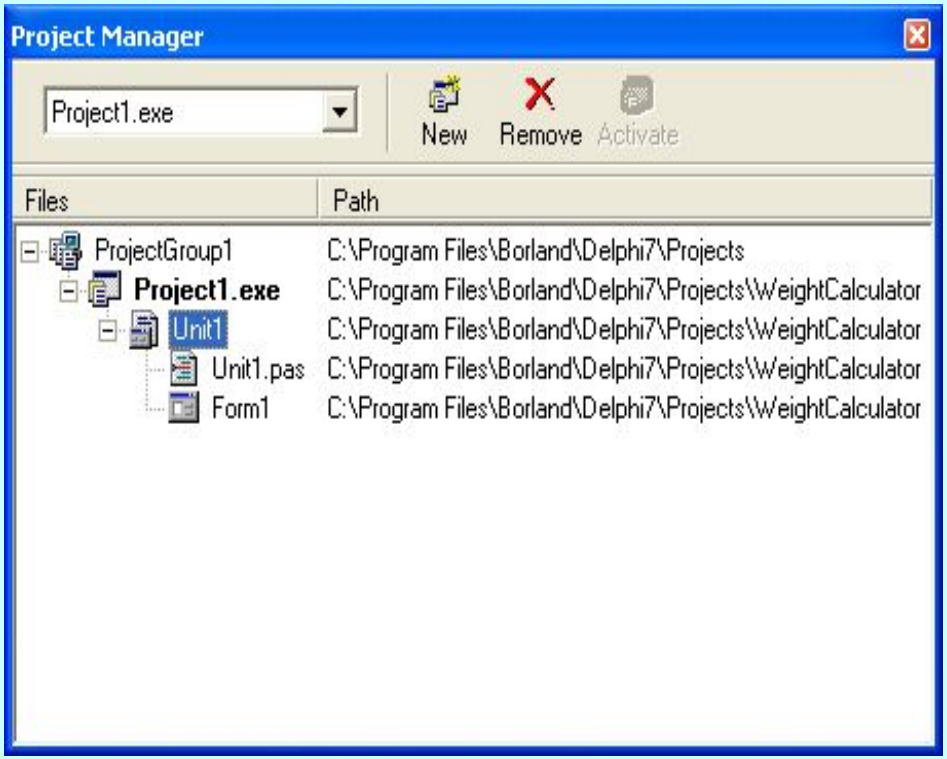
1. Создание, сохранение и открытие проекта

При запуске среды автоматически создается новый проект.

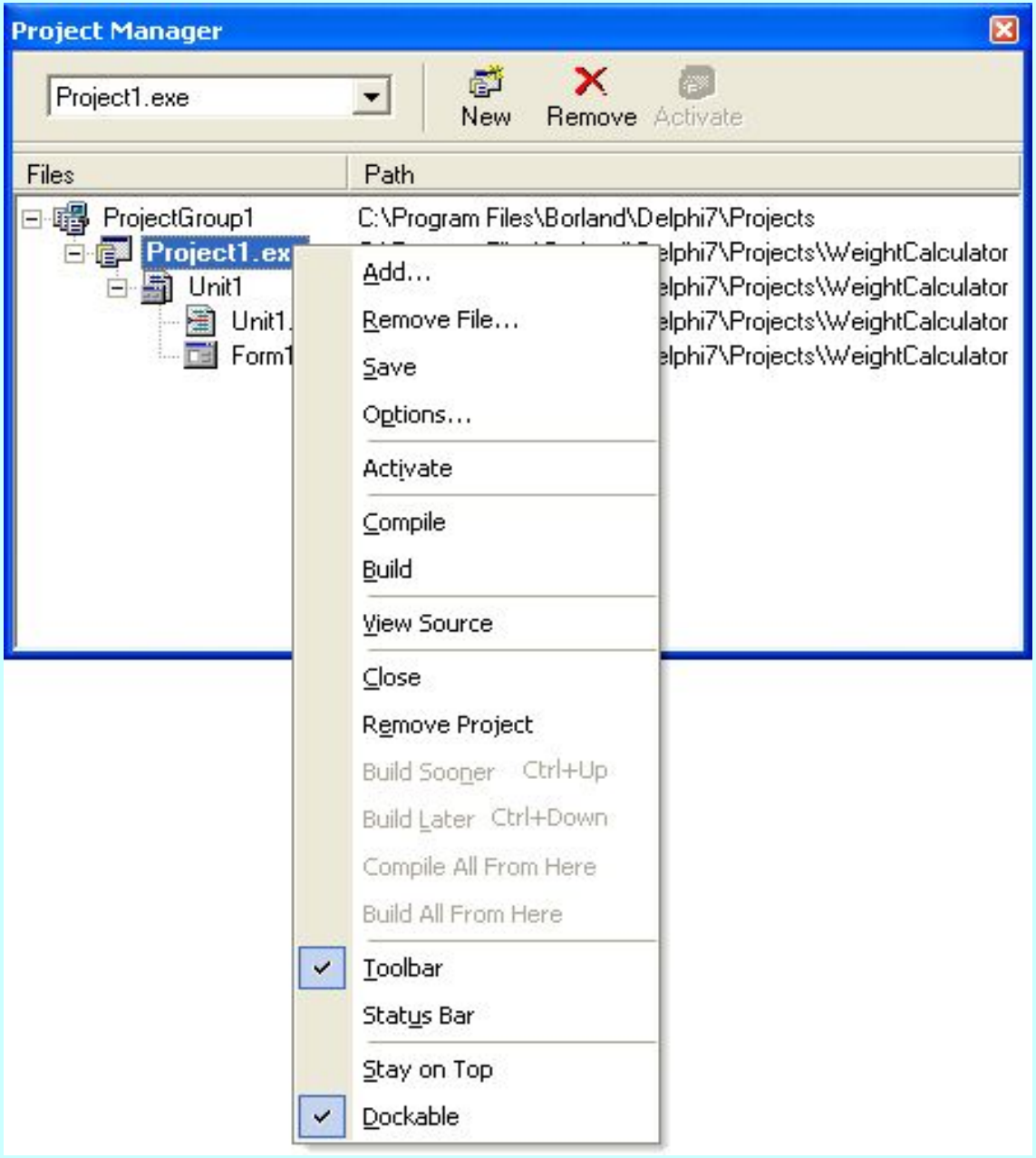
Создать новый проект - команда меню **File / New / Application**.
Старый проект будет закрыт, вместо него создан новый, в него среда всегда помещает чистую форму.
В процессе разработки приложения вы добавляете на форму компоненты, пишите обработчики событий, добавляете в проект дочерние формы, т.е. проектируете приложение.
Сохранить проект - команда **File / Save All**.
Заменить имя проекта другим именем – **File / Save Project As...**
Заменить имя модуля - **File / Save As...**
Открытие ранее сохраненного на диске проекта - **File / Open...**

2. Окно управления проектом

При создании приложения программист должен знать, на какой стадии разработки он находится, иметь представление о составе проекта, уметь быстро активизировать нужный файл, добавить в проект новый файл или удалить ненужный, установить параметры компиляции, и т.д. Для этого в среде Delphi - окно управления проектом **Project Manager**.



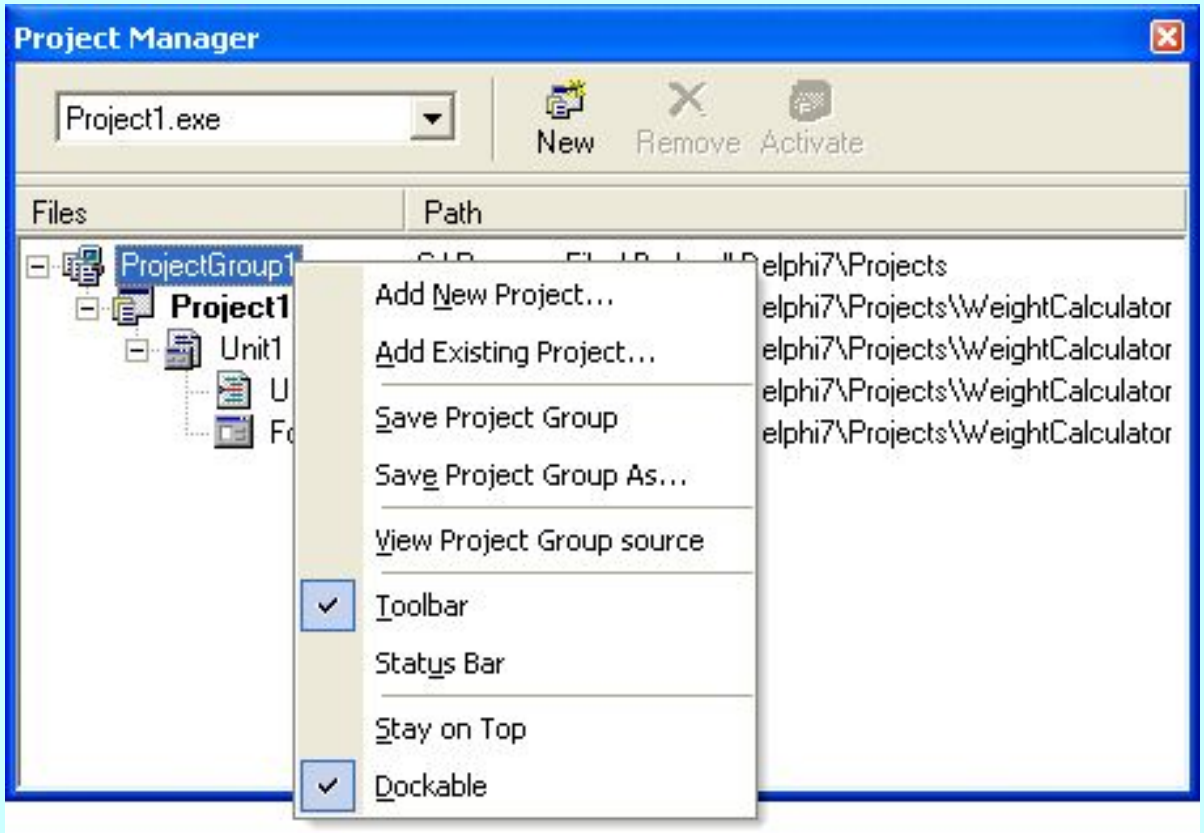
Фактически это визуальный инструмент для редактирования главного файла проекта. Вызов окна:
View / Project Manager
- на экране появится окно, в котором проект представлен в виде дерева



Управление проектом - с помощью контекстного меню (правая кнопки мыши по элементу Project1). Управление модулем – по элементу Unit1

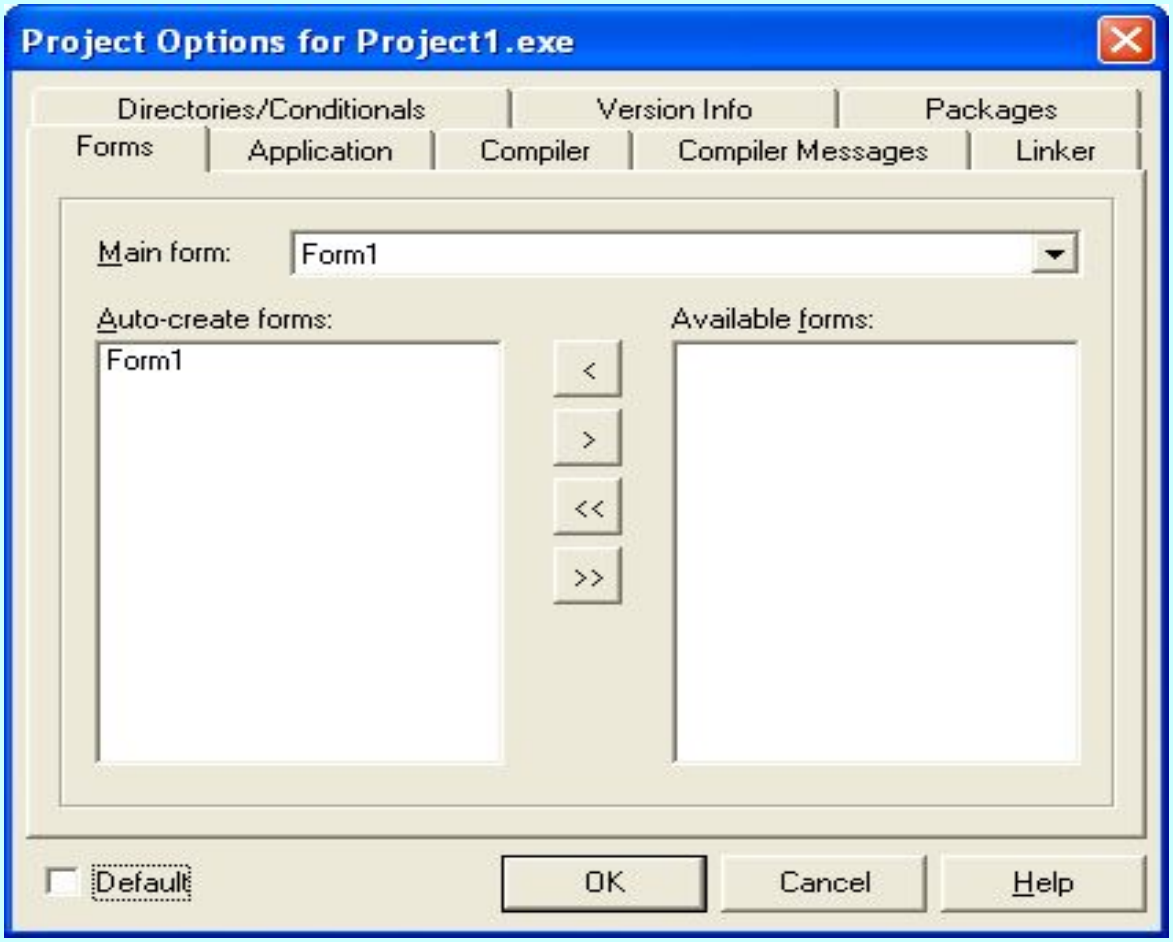
3. Группы проектов

На практике несколько проектов могут быть логически связаны между собой (проект DLL связан с проектом приложения).
Можно объединить проекты в группу. Для этого в окне управления проектом имеется корневой элемент ProjectGroup1, подчиненными элементами которого и являются логически связанные проекты.
Порядок элементов определяет очередность сборки проектов.
Изменить порядок - команды **Build Sooner** и **Build Later.**



4. Настройка параметров проекта

Проект имеет много параметров для управления процессом компиляции и сборки приложения. Установка параметров - в окне **Project Options**.Диалоговое окно параметров проекта состоит из нескольких вкладок.



5. Компиляция и сборка проекта

- могут выполняться на любой стадии разработки проекта.

Компиляция - получение объектных модулей из исходных текстов программных модулей.

Сборка - получение выполняемого файла из объектных модулей.

Компиляция: **Project / Compile <Имя проекта>** или **Ctrl+F9**.

Компилируются все исходные модули, содержимое которых изменялось после последней компиляции. Для каждого программного модуля создается файл (.DCU). Среда компилирует главный файл проекта и собирает (компонует) из DCU-модулей выполняемый файл (имя совпадает с именем проекта).

Полная принудительная компиляция всех программных модулей проекта с последующей сборкой выполняемого файла. При этом не важно, вносились в них изменения после предыдущей компиляции или нет. **Project / Build <Имя проекта>**. В результате тоже создается выполняемый файл, но на это тратится больше времени.

6. Запуск готового приложения

После исправления ошибок и получения выполняемого файла надо выполнить созданное приложение **Run / Run** (или F9). Перед выполнением будет автоматически повторен процесс компиляции (если в проект вносились изменения), и после его успешного завершения приложение запустится на выполнение. В результате вы увидите на экране его главную форму.

Рассмотрим составные части проекта и одновременно основные элементы любого приложения - формы.

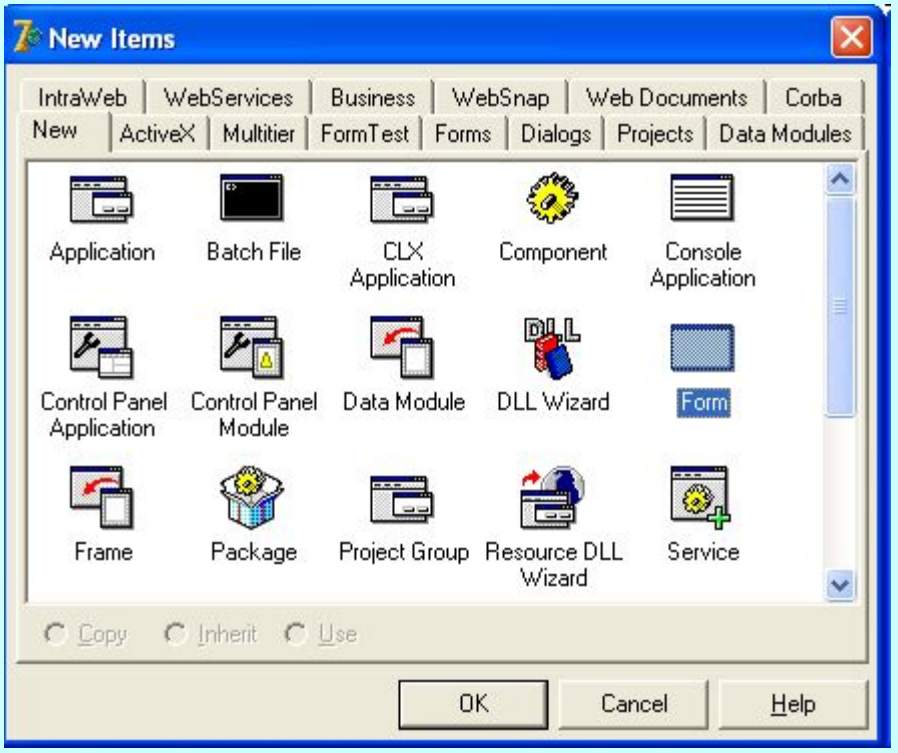
Форма

- главный компонент приложения, имеющий (как и др. компоненты) свойства. При создании новой формы среда сама задает начальные значения свойствам формы, их можно изменить во время проектирования формы (в окне свойств) или во время выполнения приложения (с помощью операторов языка).

Свойства формы: имя объекта (визуального компонента); надпись или изображение; доступность; видимость; положение; размеры; шрифт; и др.

Добавление новой формы в проект: File / New / Form - на экране появится вторая форма, в окне редактора кода - соответствующий новой форме программный модуль.

Добавление новой формы из Хранилища Объектов (содержит заготовки форм, программных модулей и целых проектов): **File / New / Other...**.



Компоненты

- Фундаментальное понятие. Без компонентов все преимущества визуальной разработки приложений исчезают.

2 взгляда на компоненты:

1) Взгляд снаружи, из среды визуальной разработки приложений: компоненты - самодостаточные строительные блоки, которые вы берете из палитры компонентов и переносите на форму для создания собственно приложения.
Примеры: кнопки, списки, надписи и др.

2) Взгляд изнутри, из программы: компоненты - это классы, порожденные прямо или косвенно от класса **TComponent** и имеющие свойства. Экземпляры компонентов - это объекты этих классов, существующие в качестве полей формы.

Объединение этих двух точек зрения дает цельное представление компонентах. При работе с компонентами из среды визуальной разработки приложений вы всегда видите их лицевую сторону. Но как только вы начинаете писать обработчики событий, и управлять компонентами программно, вы соприкасаетесь с программной стороной компонентов, суть которой - объекты. Таким образом, среда (Delphi) обеспечивает симбиоз визуального и объектно-ориентированного программирования.

Компонент = состояние (свойства) + поведение (методы) + обратная реакция (события)

Состояние компонента определяется его свойствами. Свойства бывают изменяемые (для чтения и записи) и неизменяемые (только для чтения). Также свойства могут получать значения либо на этапе проектирования (design-time), либо только во время выполнения программы (run-time).

Первые устанавливаются для каждого компонента в окне свойств и определяют начальное состояние компонента. Во время выполнения приложения эти свойства могут быть изменены программно, изменится внешний вид и поведение компонента.

Вторая группа - это свойства, которые не видны в окне свойств, и управлять которыми можно только программно.

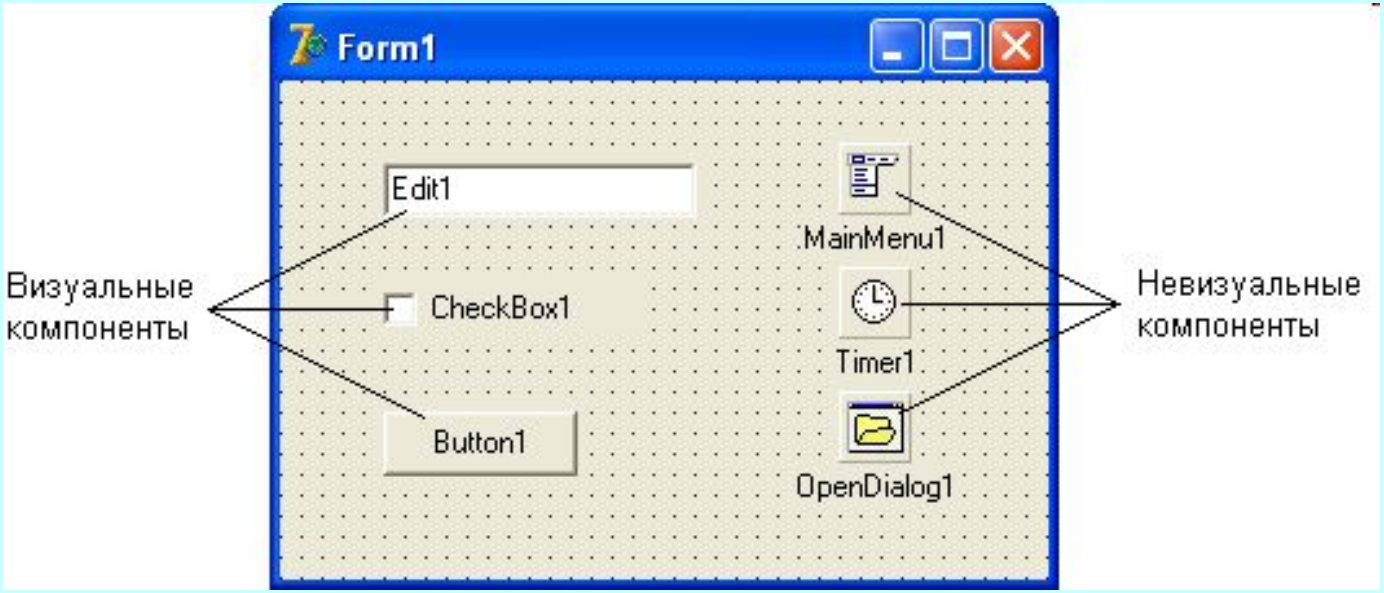
Поведение компонента описывается с помощью его процедур и функций (методов). Вызовы методов компонента помещаются в исходный код программы и происходят только во время выполнения приложения. Методы не имеют под собой визуальной основы.

Обратная реакция компонента - это его **события** . События позволяют, например, связать нажатие кнопки с вызовом метода формы.

Визуальные и невизуальные компоненты

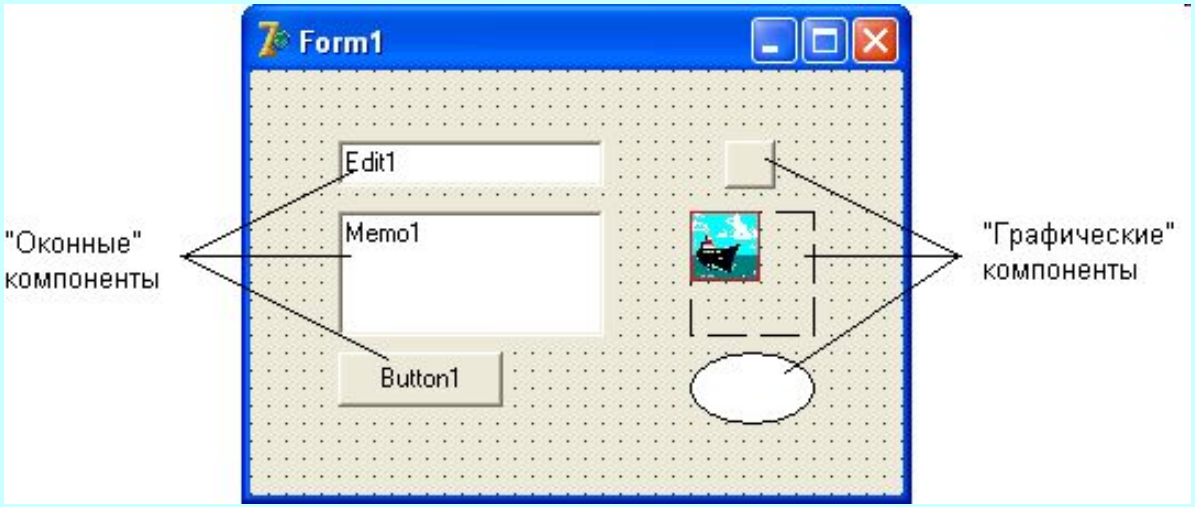
Визуальные компоненты (visual components) - это видимые элементы пользовательского интерфейса: кнопки, метки, блоки списков и др. Они выглядят одинаково и на стадии проектирования, и во время работы приложения.

Невизуальные компоненты (non-visual components) - они работают, но сами на экране не видны: таймер, компоненты доступа к БД и др. При проектировании они представляются на форме небольшим значком. Могут иметь подписи. Их свойства устанавливаются в окне свойств. Некоторые невизуальные компоненты могут что-нибудь отображать на экране: компонент **MainMenu** отображает на форме полосу главного меню, компонент **OpenDialog** - стандартное диалоговое окно выбора файла.



«Оконные» и «графические» компоненты:

рисуемые оконной системой Windows и графической библиотекой VCL.
«Оконные» (windowed controls) являются специализированными окнами внутри окна формы. Их главное качество - способность получать фокус ввода. Это компоненты: **Button**, **RadioButton**, **CheckBox**, **GroupBox**, и т.д.
Некоторые оконные компоненты (**GroupBox**, **TabControl**, **PageControl**) способны содержать другие визуальные компоненты и называются ***контейнерами*** (container controls). Отображение оконных компонентов обеспечивается операционной системой Windows.
«Графические» (graphical controls) не являются окнами, поэтому не могут получать фокус ввода и содержать другие визуальные компоненты. Они не основаны на объектах ОС Windows, их отображение полностью выполняет библиотека VCL. Это: **SpeedButton**, **Image**, **Bevel** и т.д.



Общие свойства визуальных компонентов:

Left и **Top** - местоположение внутри формы (или компонента-владельца);
Width и **Height** - горизонтальный и вертикальный размеры;
Anchors - позволяет привязать границы компонента к границам формы;
BiDiMode – для чтения текста справа налево;
Caption - надпись компонента;
Constraints - ограничения на размеры компонента;
Color - цвет компонента;
Cursor – вид указателя мыши при наведении на него пользователем;
DragCursor - вид указателя мыши при буксировке объекта над компонентом;
DragKind – поведение компонента при буксировке;
DragMode - режим буксировки компонента по экрану;
Enabled – определяет доступность компонента для пользователя;
Font - шрифт надписи на компоненте;
HelpType - как в файле справки будет осуществляться поиск темы;
HelpContext - содержит номер соответствующей темы в файле справки;
Hint – появляющаяся над компонентом подсказка;
PopupMenu - для привязки контекстного меню к компоненту;
TabOrder - порядковый номер компонента в пределах компонента-владельца;
TabStop - определяет, может ли компонент получать фокус ввода;
Visible - определяет видимость компонента на экране.

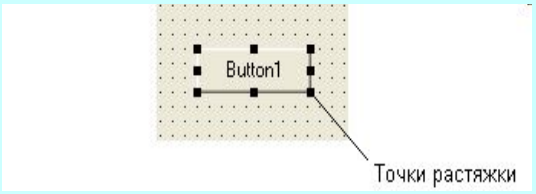
Общие события визуальных компонентов:

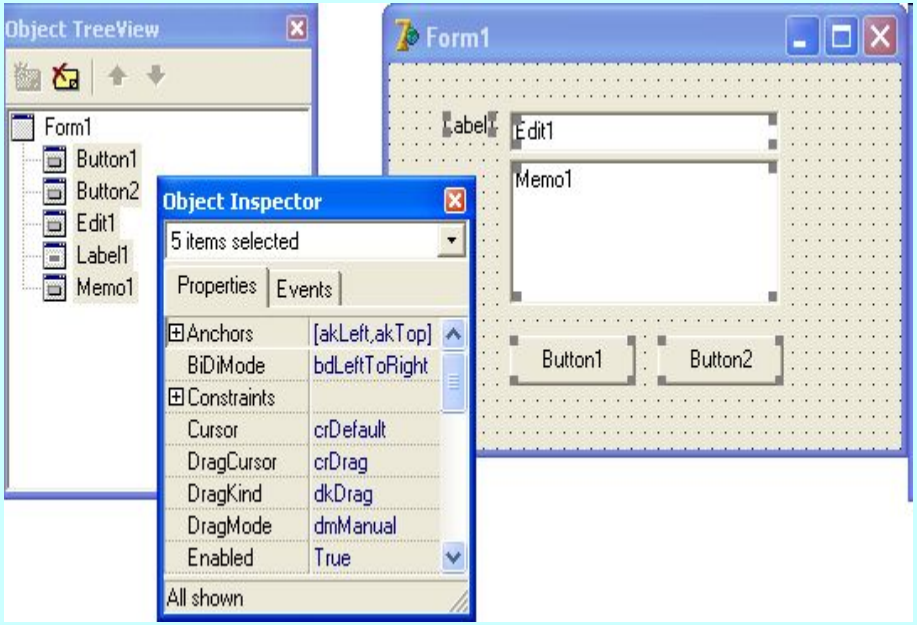
OnClick - происходит в результате щелчка мыши по компоненту.
OnContextPopup - происходит при вызове контекстного меню компонента.
OnDblClick - происходит в результате двойного щелчка мыши по компоненту.
OnEnter - происходит при получении компонентом фокуса ввода.
OnKeyDown - происходит при нажатии пользователем любой клавиши (если компонент обладает фокусом ввода).
OnMouseDown - происходит при нажатии пользователем кнопки мыши, когда указатель мыши наведен на компонент.
OnMouseUp. При перемещении указателя мыши над компонентом, в нем возникает событие **OnMouseMove**, для отслеживания позиции указателя.

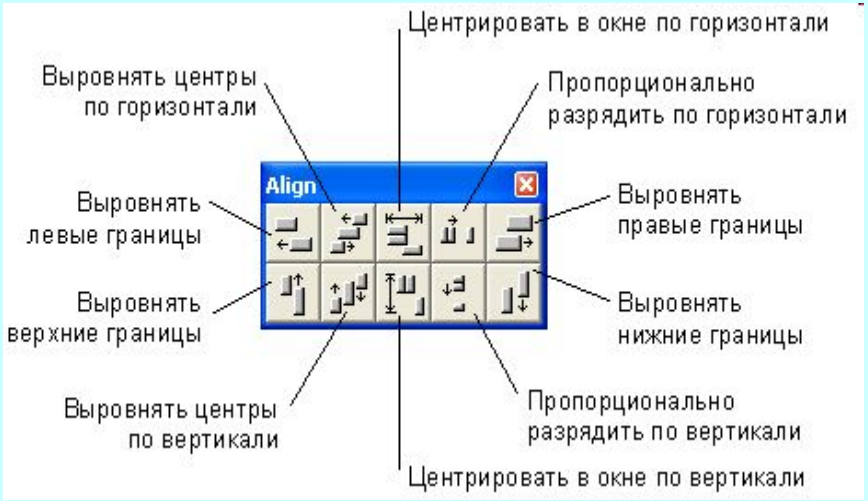
Для организации буксировки и стыковки, в визуальных компонентах существует еще несколько событий:
OnStartDrag - происходит, когда пользователь начинает что-нибудь буксировать.
OnEndDrag - происходит по окончании буксировки объекта и др.

Управление компонентами при проектировании:

- Помещение компонентов на форму из палитры компонентов и их удаление;
- Выделение компонентов на форме;
- Перемещение и изменение размеров компонента.
- Выравнивание компонентов на форме.







Установка свойства для группы компонентов

Компоненты вкладки Standard в Delphi

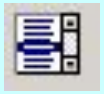
Вкладка Standard палитры компонентов. Компоненты Delphi, размещенные на данной вкладке, имеют самое общее назначение и широко используются при создании любых оконных приложений

Вкладка стандарт палитры компонентов

TFrame - контейнер для размещения других компонентов

TRadioButton - переключатель

TMainMenu - главное меню формы

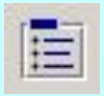


TListBox - список

TLabel - метка

TComboBox – комбинированный список

TEdit - однострочное редактируемое текстовое поле



TRadioGroup – группа переключателей

TMemo -многострочное редактируемое текстовое поле

TButton - кнопка

TScroll – полоса прокрутки

TCheckBox - флажок

TGroupBox – панель группирования

Application - главный объект, управляющий приложением

Рассмотрели внешнюю сторону приложения.

А что происходит внутри?

Над всеми формами и компонентами стоит объект **Application** (класса TApplication), олицетворяющий собой приложение в целом. Это главное "действующее лицо", которое создается в начале выполнения любого приложения. Объект **Application** держит в руках все нити управления: создает главную и второстепенные формы, уничтожает их, обслуживает исключительные ситуации.

Объект **Application** отсутствует в палитре компонентов, поэтому его свойства можно изменять только из программы.

Screen - объект, управляющий экраном

Каждая программа что-то выводит на экран. В среде Delphi экран трактуется как глобальный объект **Screen** класса TScreen, имеющий набор свойств. Многие из них жестко связаны с физическими характеристиками экрана (с "железом"), поэтому в большинстве случаев не доступны для записи.

Mouse - объект, представляющий мышь

Printer - объект, управляющий принтером

Clipboard - объект, управляющий Буфером обмена

Borland C++ Builder для ОС Windows - среда разработки на языке C++ компании «Borland», входит в состав «Borland Developer Studio».

Среда разработана в соответствии с концепцией визуального программирования, обеспечивает:

- скорость визуальной разработки;
- продуктивность повторно используемых компонент в сочетании с мощью языковых средств C++;
- усовершенствованными инструментами;
- разномасштабными средствами доступа к БД.

Интегрированная среда разработки объединяет:
Редактор форм, Инспектор объектов, Палитру компонент, Администратор проекта и полностью интегрированные Редактор кода и Отладчик - инструменты быстрой разработки программных приложений, обеспечивающие полный контроль над кодом и ресурсами.

Borland C++ Builder

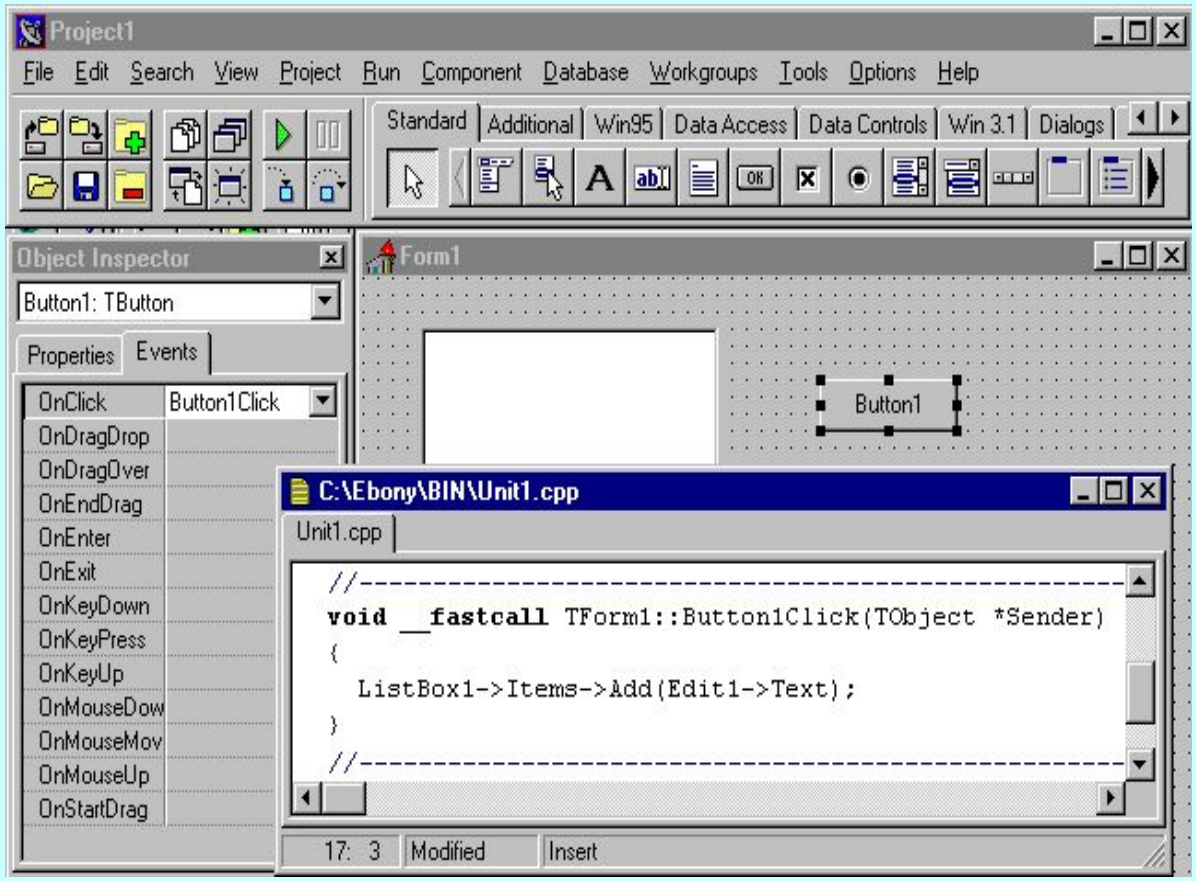
- средство быстрой разработки приложений на языке C++, используя среду разработки и библиотеку компонентов Delphi.

Среда разработки C++ Builder

- это SDI-приложение, главное окно содержит настраиваемую панель инструментов и и палитру компонентов.

При запуске C++ Builder появляются окно инспектора объектов и форма нового приложения.

Под окном формы приложения находится окно редактора кода.



Основа приложений – **формы**.

Создание ПИ заключается в добавлении в окно формы элементов объектов C++ Builder – **компонентов**.

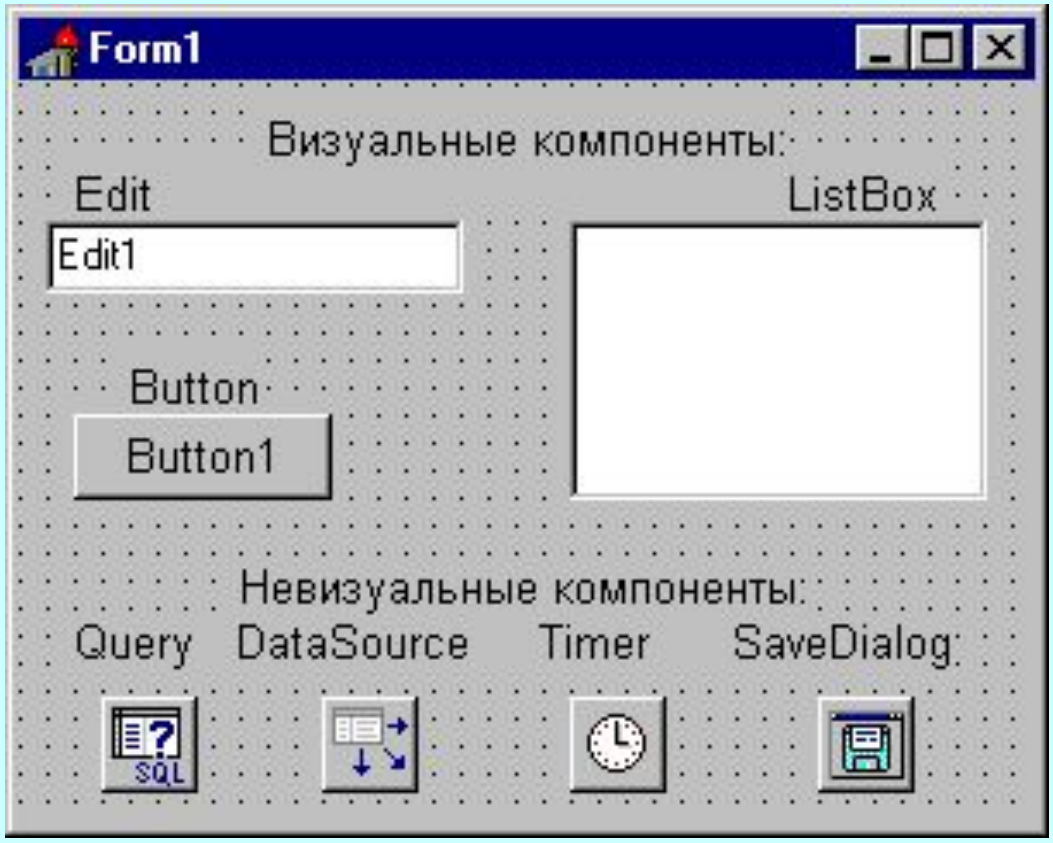
Компоненты располагаются **на палитре компонентов**, выполненной в виде многостраничного блокнота.

Важная особенность - C++ Builder позволяет создавать собственные компоненты, настраивать палитру компонентов, создавать различные версии палитры компонентов для разных проектов.

Компоненты C++ Builder:

видимые (визуальные) и невидимые (невизуальные).
Визуальные компоненты появляются во время выполнения так же, как и во время проектирования.
Примеры: кнопки и редактируемые поля.

Невизуальные компоненты появляются во время проектирования как пиктограммы на форме. Они никогда не видны во время выполнения, но обладают функциональностью (обеспечивают доступ к данным, вызывают стандартные диалоги и др.)

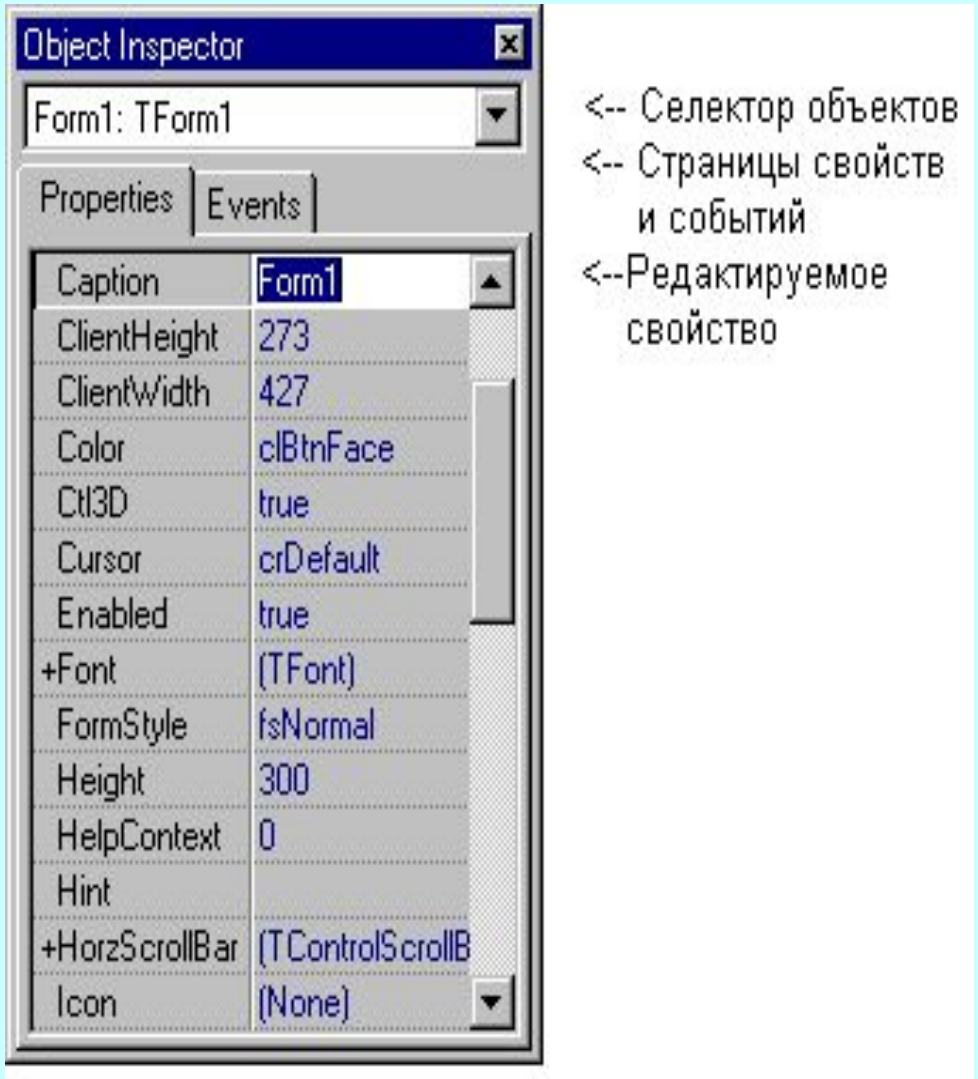


Добавление компонента в форму: выбрать нужный компонент в палитре и щелкнуть ЛКМ в нужном месте проектируемой формы.

Компонент появится на форме, далее его можно перемещать, менять размеры и др. характеристики.

Характеристики компонентов:

свойства, события, методы



При выборе компонента из палитры и добавления его к форме, инспектор объектов автоматически покажет свойства и события, которые могут быть использованы с этим компонентом.

В верхней части инспектора объектов - выпадающий список для выбора нужного объекта из имеющихся на форме

Свойства

- атрибуты компонента, определяют его внешний вид и поведение.

Свойства могут иметь значение, устанавливаемое по умолчанию (высота кнопок).

Свойства компонента отображаются на странице свойств (Properties).

Инспектор объектов отображает опубликованные (published) свойства компонентов.

Компоненты могут иметь и общие (public), опубликованные свойства, которые доступны только во время выполнения приложения.

Инспектор объектов используется для установки свойств во время проектирования.

Список свойств - на странице свойств инспектора объектов.

Определить свойства можно:

1. во время проектирования
2. написать код для видоизменения свойств компонента во время выполнения приложения.

Определение свойств компонента

во время проектирования:

- выбрать компонент на форме,
- открыть страницу свойств в инспекторе объектов,
- выбрать определяемое свойство и изменить его с помощью редактора свойств

(простое поле для ввода текста, числа; выпадающий, раскрывающийся список; диалоговая панель и др.).

События

Страница событий (Events) инспектора объектов показывает список событий, распознаваемых компонентом

(программирование для ОС с графическим ПИ предполагает описание реакции приложения на события, сама ОС занимается опросом ПК для выявления наступления события).

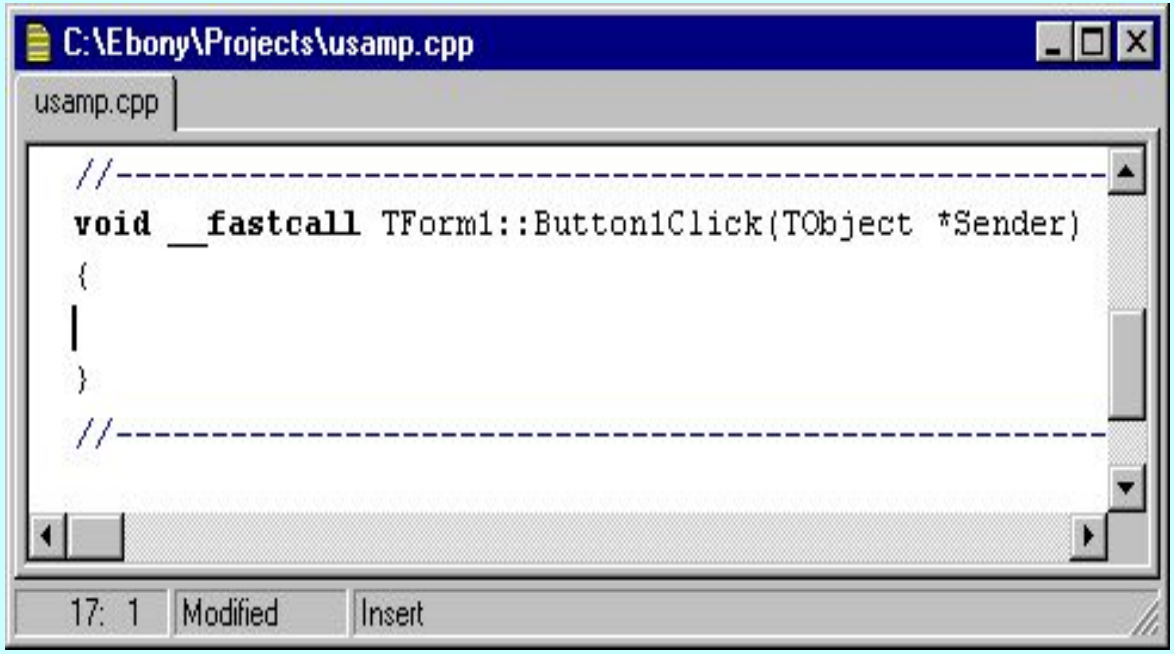
Каждый компонент имеет свой набор обработчиков событий.

В C++ Builder пишут функции - обработчики событий и связывают события с этими функциями.

Создавая обработчик события, вы поручаете программе выполнить написанную функцию, если это событие произойдет.

Создание обработчика события:

- выбрать на форме необходимый компонент,
- открыть страницу событий инспектора объектов, дважды щелкнуть ЛКМ на колонке значений рядом с событием (C++ сгенерирует прототип обработчика событий и покажет его в редакторе кода. Автоматически генерируется текст пустой функции, редактор откроется в том месте, где следует вводить код).
- ввести код, который должен выполняться при наступлении события.



Обработчик событий может иметь параметры, они указываются после имени функции в круглых скобках.

Методы

Метод – это функция, связанная с компонентом, объявляется как часть объекта.

Вызов метода при создании обработчика событий: **->**, например: Edit1->Show();

При создании формы обязательно генерируются связанные с ней модуль и заголовочный файл (*.h).

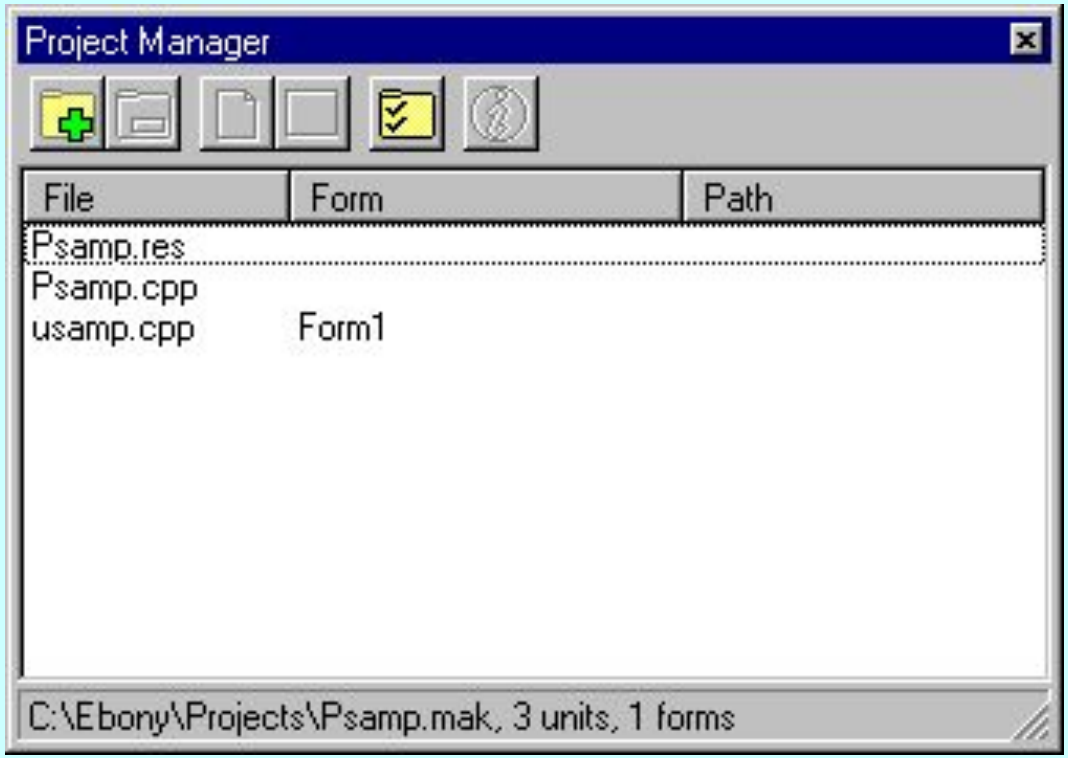
При создании нового модуля он не обязан быть связан с формой (например, если в нем содержатся процедуры расчетов).

Имена формы и модуля можно изменить, желательно сделать это сразу после создания, пока на них не появилось много ссылок в других формах и модулях.

Менеджер проектов

Файлы, образующие приложение (формы и модули) собраны в проект.

Менеджер проектов показывает списки файлов и модулей приложения и позволяет осуществлять навигацию между ними.



Вызов менеджера проектов:
View/Project Manager

По умолчанию имя созданного проекта – ***Project1.cpp***

По умолчанию проект первоначально содержит файлы для одной формы и исходного кода одного модуля.

Добавление модулей (форм) к проекту:

ПКМ - New Form

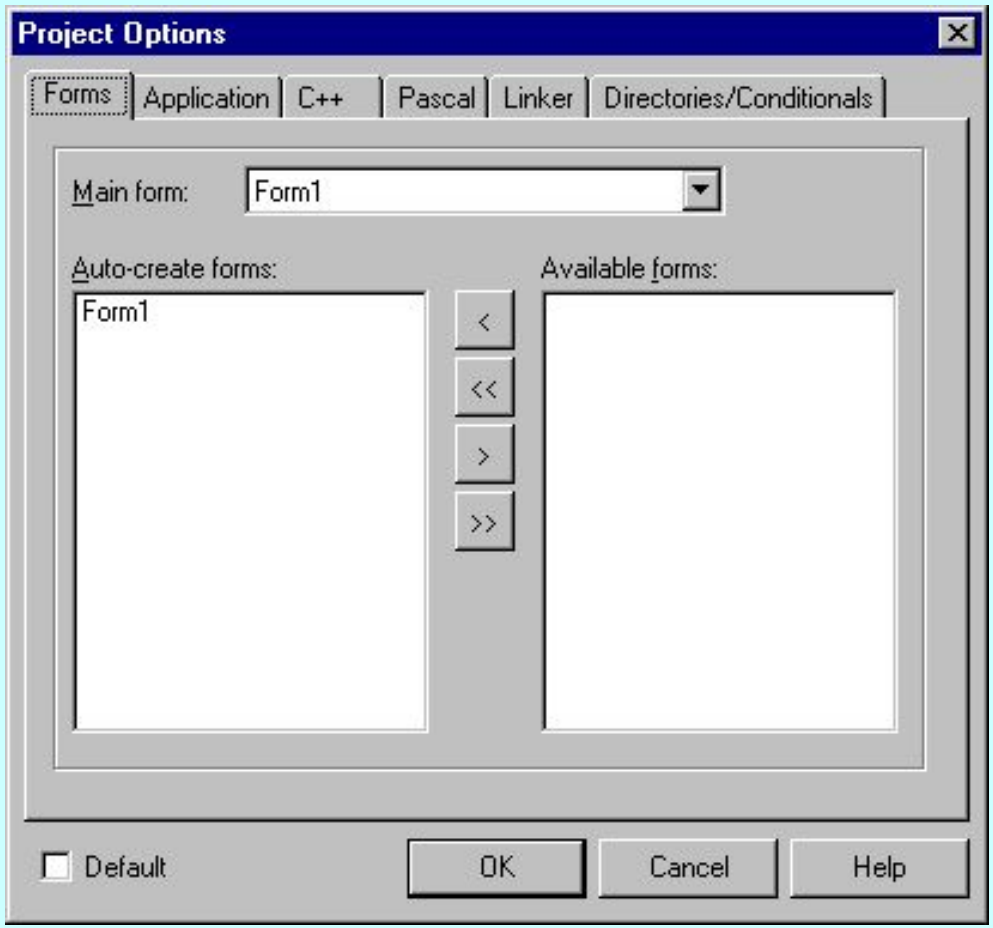
Добавление существующих форм и модулей: кнопка ***Add*** контекстного меню менеджера проектов, выбрать модуль (форму), которую нужно добавить.

Формы и модули можно удалить в любой момент в течение разработки проекта.

Так как форма всегда связана с модулем, нельзя удалить одно без удаления другого, кроме случая, когда модуль не имеет связи с формой.

Удаление модуля из проекта: кнопка ***Remove*** менеджера проектов.

Кнопка ***Options*** в менеджере проектов - откроется диалоговая панель опций проекта, можно выбрать главную форму приложения, определить, какие формы будут создаваться динамически, параметры компиляции модулей и компоновки.

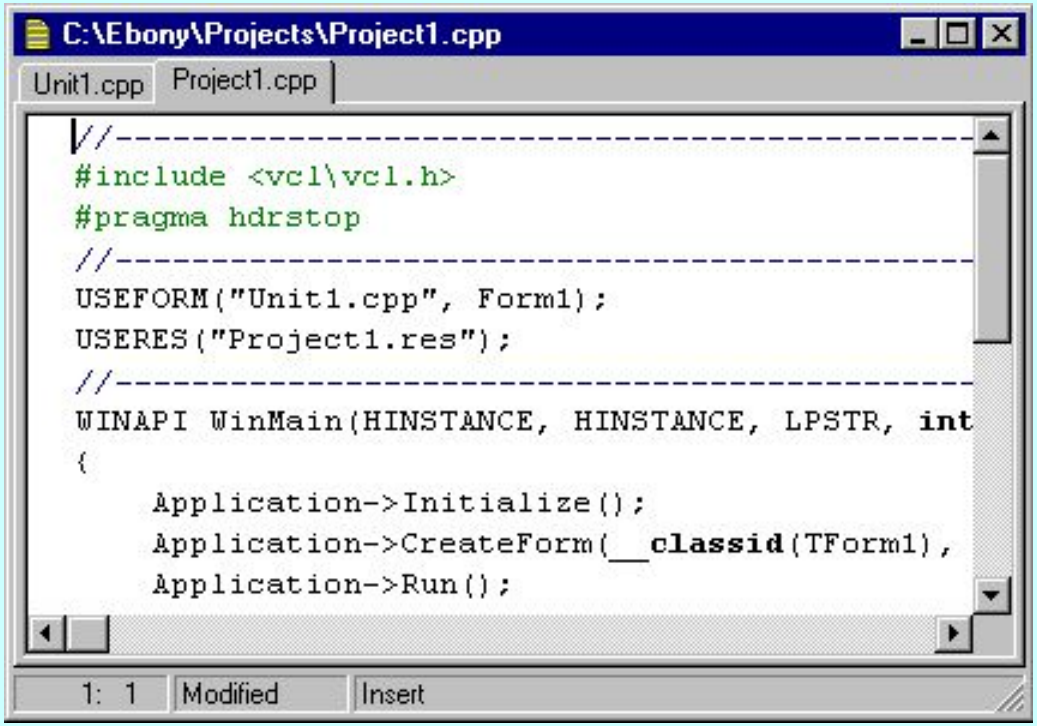


Важный элемент C++ Builder - контекстное меню для быстрого доступа к наиболее часто используемым командам. C++ Builder обладает встроенной системой контекстно-зависимой помощи, доступной для любого элемента интерфейса и являющейся обширным источником справочной информации о C++ Builder.

Создание приложений в C++ Builder

1ый шаг - создание проекта. Файлы проекта содержат сгенерированный автоматически исходный текст, он становится частью приложения, когда оно скомпилировано и подготовлено к выполнению.

Создание нового проекта: ***File/New Application***



C++ Builder создаст:

- файл проекта (Project1.cpp)
- make-файл Project1.mak.

При внесении изменений в проект (добавление новой формы) C++ Builder обновляет файл проекта.

Проект (приложение) имеют несколько форм.

Добавление формы к проекту создает дополнительные файлы:

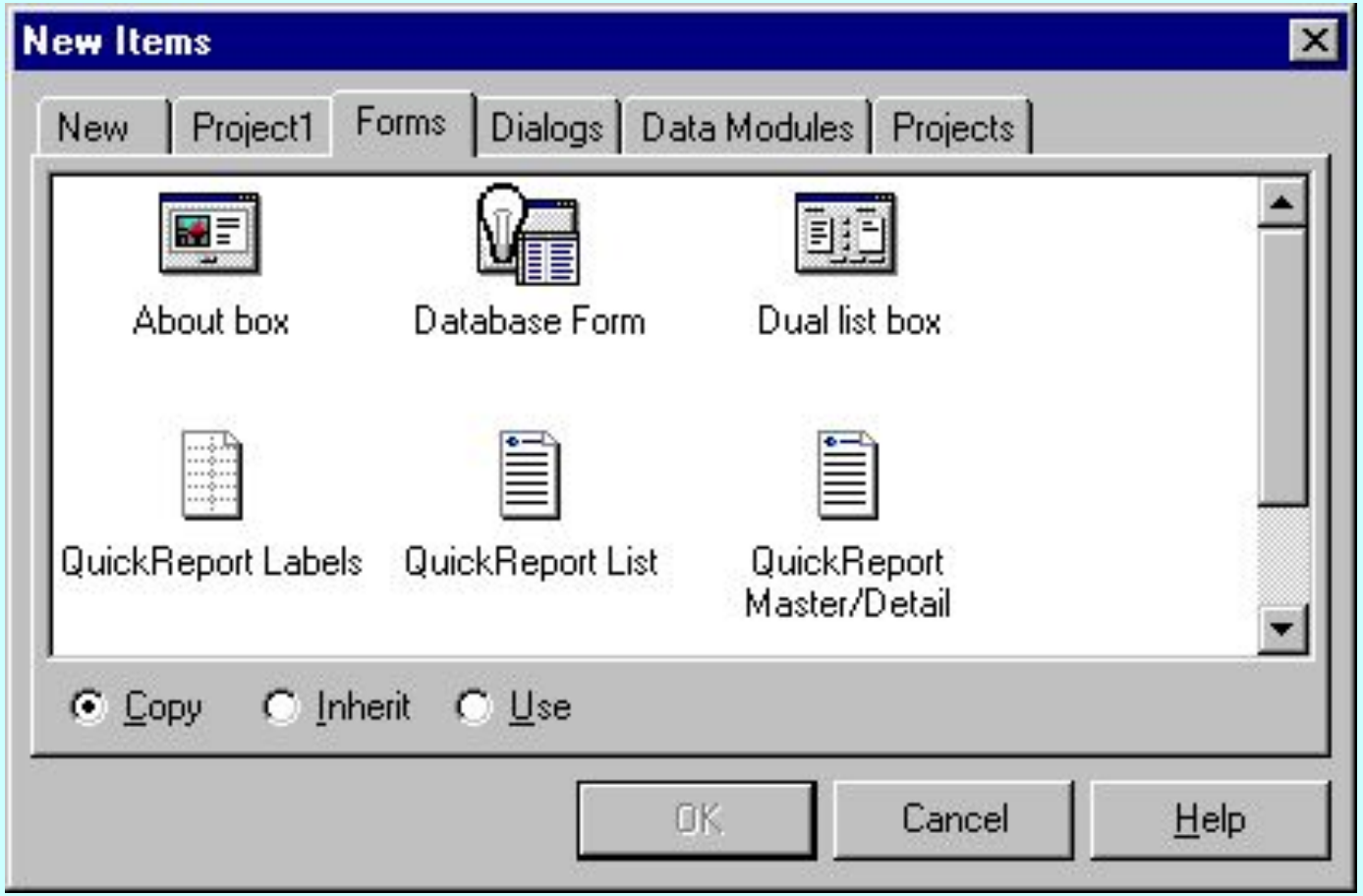
- Файл формы ***.DFM*** - информация о ресурсах окон для конструирования формы
- Файл модуля ***.CPP*** - код на C++.
- Заголовочный файл ***.H*** - описание класса формы.

При добавлении новой формы, файл проекта автоматически обновляется.

Добавление форм к проекту:***File/New Form***.

Появится пустая форма, которая добавится к проекту.

ИЛИ: меню ***File/New***, выбрать страницу ***Forms*** и нужный шаблон из репозитория объектов.



Компиляция текущего проекта: ***меню Compile, пункт Compile***.

Компиляция проекта и создание исполняемого файла:
меню Run, пункт Run.

Компоновка проекта является инкрементной (перекомпилируются только изменившиеся модули).

Если при выполнении приложения возникает ошибка времени выполнения, C++ Builder делает паузу в выполнении программы и показывает редактор кода с курсором, установленным на операторе, являющемся источником ошибки.

Следует перезапустить приложение (Run), закрыть приложение, затем вносить изменения в проект.

В этом случае уменьшится вероятность потери ресурсов Windows.

Пример: создание простейшего приложения:

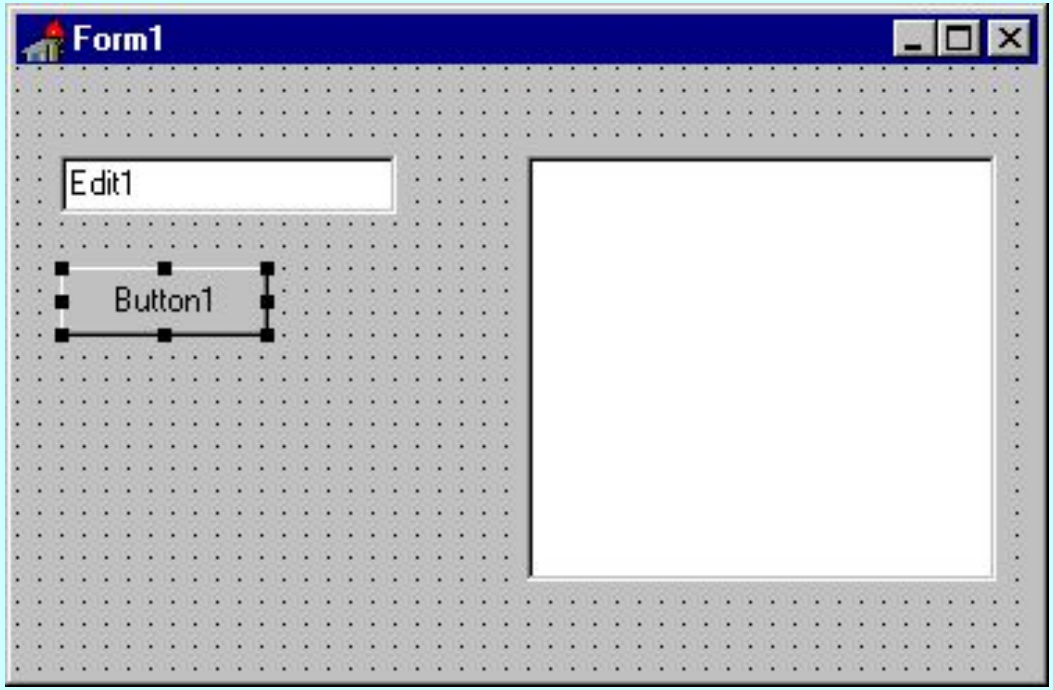
вводит текст в редактируемое поле, добавляет текст к списку при нажатии мышью на кнопку.

Создадим проект: ***File/New Application***

сохраним его главную форму под именем **samp1.cpp**,

а сам проект под именем **samp.mak**.

Поместим на форму компоненты Button, Edit и ListBox со страницы Standard палитры компонент:

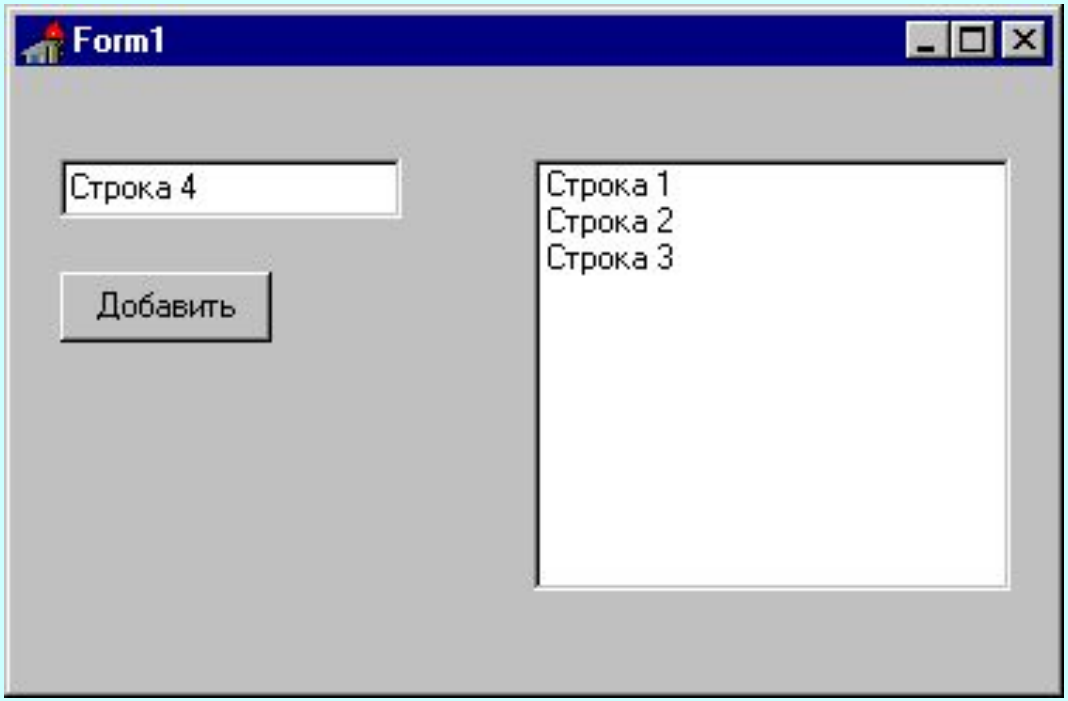


- Выберем на форме компонент ***Edit***, удалим текущее значение свойства ***Text*** и установим свойство ***Caption*** для ***Button1*** равным "***Добавить***".
- Добавим обработчик события ***OnClick*** для кнопки ***Добавить:*** выбрать эту кнопку на форме, открыть страницу событий в инспекторе объектов и дважды щелкнуть мышью на колонке справа от события ***OnClick***. В строке ввода появится имя функции.

C++ Builder сгенерирует прототип обработчика событий и покажет его в редакторе кода.

Введем следующий код:

```
void __fastcall TForm1::
    Button1Click(TObject *Sender)
    {
    if (!(Edit1->Text == ""))
    {
    ListBox1->Items->Add(Edit1->Text);
    Edit1->Text = "" ;
    }
    }
```

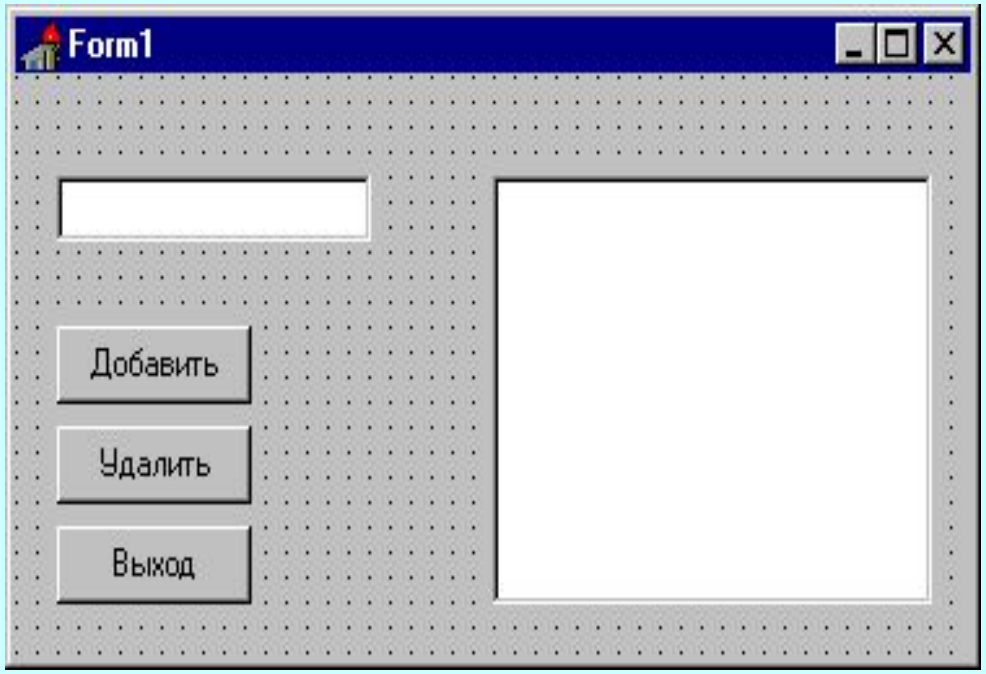


Откомпилируем приложение (Run), введем что-либо в редактируемое поле, нажмем мышью на кнопку Добавить и убедимся, что вводимые строки добавляются к списку.

Модифицируем приложение:

добавим кнопки ***Удалить*** и ***Выход***.

Добавим еще две кнопки, изменим их свойство ***Caption*** и создадим обработчики событий, связанных с нажатием на эти кнопки:



Для кнопки Удалить:

```
void __fastcall TForm1::
Button2Click(TObject *Sender)
{
if (!(ListBox1->ItemIndex == -1))
ListBox1->Items->
Delete(ListBox1->ItemIndex);
}
```

Для кнопки Выход: *Close();*

Сохраним,скомпилируем приложение, протестируем его.