Лекция 24 Введение в объектно-ориентированное программирование

Объекты и классы

Основными понятиями ООП в C++ является **объект**.

Объект это некая программная единица, объединяющая в себе свойства (атрибуты) и поведение (состояние и функционирование).

То есть объекты могут отличаться значениями свойств и, конечно, должны отличаться именами.

Среда `Borland C++ Builder` предлагает набор визуальных компонентов для создания графического интерфейса приложений `Windows`.

Технология визуального программирования позволяет пользователю визуально наблюдать в процессе разработки основные компоненты программы. С помощью мыши можно создавать и перемещать отдельные компоненты, изменять их размер, а также изменять свойства объекта.

Язык `Visual C++` предлагает свой широкий набор процедур для функционирования приложения.

При этом программирования не требуется, система сама формирует исходный текст программы и автоматически вносит в него исправления. При такой технологии задача программиста сводится к программированию определенных функций для объектов и внесению изменений в автоматически созданный текст программы. Тем самым существенно сокращается время разработки программ.

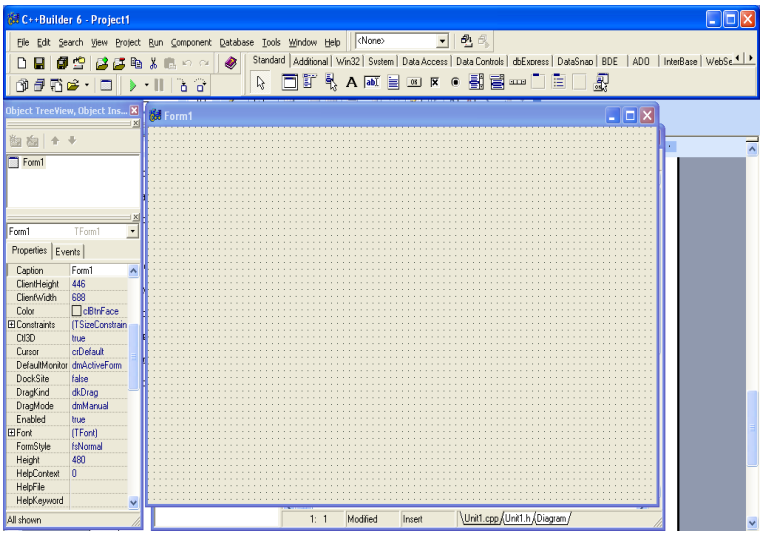

Рис. 1 Интегрированная среда разработки

После запуска C++ Builder на экране появляется окно (Рис. 1). То, что мы видим на экране, называется интегрированной средой разработки (IDE — Integrated Development Environment).

Пользователю сразу доступны следующие компоненты среды.

1. **Команды главного меню.**
2. **Панель инструментов**.
3. **Палитра компонентов.**
4. **Инспектор объектов**.

5. **Редактор форм**.
6. **Редактор кода**

Команды главного меню управляют файлами.

С помощью команды File/New... (Новый...) можно создавать различные виды приложений. При этом запускается Мастер приложений, который в диалоговом режиме позволяет программисту сгенерировать требуемый элемент.

Важным понятием при разработке программ на C++ Builder является понятие проекта. Проектом называется файл, содержащий информацию о файлах, из которых строится готовое приложение, и о том, каким образом это приложение строить. В проект может входить достаточно много файлов различных форматов. В начале работы по умолчанию предоставлен стандартный проект. На его основе строится приложение.

Просматривать проект, добавлять и удалять из него элементы можно, используя команды меню View (Вид) и Project (Проект).

По команде Project/Options... (Проект/Параметры...) можно требуемым образом настроить параметры проекта.

Меню **Run** (Выполнить) содержит команды запуска и отладки (пошаговое выполнение, установка контрольных точек, инспектирование объектов, просмотр переменных и т. д.) сгенерированного приложения.

Панель инструментов позволяет осуществлять быстрый вызов наиболее часто используемых команд.

Палитра компонентов позволяет пользователю визуально выбрать любой объект из библиотеки классов VCL (Visual Component Library) и разместить его в своем приложении.

Палитра компонентов содержит множество вкладок. На Рис. 2. приведены объекты из вкладки **Standard** (Стандартная)

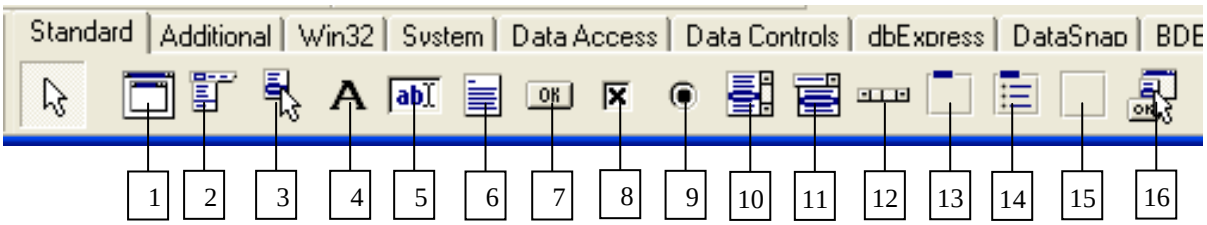

Рис. 2. объекты из вкладки **Standard** (Стандартная)

На вкладке **Standard** (Стандартная) содержатся следующие компоненты:

1. **Frames** создает фреймы.
2. **MainMenu** создает для окна панель с командами главного меню.
3. **PopUpMenu** создает всплывающее меню для компоненты приложения.

4. **Label** помещает на объект текст, который нельзя редактировать.
5. **Edit** помещает на объект область редактируемого ввода в виде одиночной строки.
6. **Memo** размещает область редактируемого ввода из множества строк.
6. **Button** размещает кнопку с названием.

7. CheckBox создает управляющий элемент с двумя возможными состояниями, причем состояние одного элемента не влияет на состояние других подобных элементов объекта.

8. **RadioButton** создает управляющий элемент с двумя возможными состояниями. Из группы подобных элементов объекта только один может находиться в установленном состоянии. Установка другого элемента из группы автоматически приводит к выключению предыдущего элемента.

9. **ListBox** создает область в виде списка текстовых строк для ввода.

10. **ComboBox** создает комбинацию из строки редактируемого ввода и ниспадающего списка текстовых строк для ввода.

11. **ScrollBar** создает линейку прокрутки для окна, списка и т. п.

12. **GroupBox** создает контейнер, объединяющий логически связанные объекты.

13. **RadioGroup** создает контейнер для объединения управляющих элементов **TRadioButton.**

14. **Panel** создает панель инструментов.

15 **Action List**создает список действий.

При подведении указателя мыши к тому или иному компоненту, возникает **всплывающая подсказка**, помогающая понять его назначение.

Инспектор объектов

Используемыми в приложении компонентами можно управлять с помощью свойств, методов и событий.

Свойства объекта определяют его характеристики и поведение. Например, для объекта форма (окно) свойство `Caption` определяет название в виде текста, которое будет отображаться в верхней части окна, `Color` задает цвет окна, `Width` – его ширину и т. д.

Свойства объекта можно менять при помощи инспектора объектов Инспектор объектов содержит две вкладки: Свойства (`Properties`) и События (`Events`). Вкладка Свойства отображает доступные для управления свойства того или иного объекта. Вкладка События отображает перечень событий, связанных с объектом. Объект, для которого требуется изменить свойства, можно выбрать, щелчком мыши на самом объекте или выбрав его из списка в верхней части Инспектора объектов.

Каждое свойство объекта имеет имя (которое нельзя изменить), например, «`COLOR`» и значение, которое можно изменить, например «`red`».

Таким образом, через инспектор объектов можно настроить объект требуемым образом, не прибегая к программированию.

Свойства объекта можно менять и в самой программе. Для обращения к свойству объекта в программе записывают имя объекта и имя свойства, разделенные знаком

`->` (стрелка), например, для объекта с именем `Form1` обращение к свойству `Caption` выглядит так:

```
Forml->Caption="Суперпрограмма!";
```

`Суперпрограмма!` - значение свойства.

После использования данного оператора в программе название окна формы будет `Суперпрограмма!`

С помощью изменения значения свойства `Visible` можно управлять Видимостью окна:

```
Forml->Visible=false; Forml->Visible=true;
```

Первый оператор делает окно невидимым, второй заставляет окно вновь появиться.

Другим важным средством управления объектами являются **методы.** Методы представляют собой функции, которые можно вызывать для управления объектами.

Например, для того чтобы окно стало невидимым, можно вызвать следующий метод:

```
Forml->Hide();
```

Чтобы сделать окно вновь видимым, надо вызвать другой метод:

```
Forml->Show();
```

Полный перечень свойств и методов объекта из библиотеки VCL приводится в документации и во встроенной справочной системе Borland C++ Builder.

Третье средство управления объектами — это **события.** Для каждого объекта существует свой перечень событий, содержащихся на вкладке Events. Каждое событие, также как и каждое свойство имеет неизменное имя и значение, которое можно изменить.

Используя вкладку Events инспектора объектов, можно изменить реакцию объекта на то или иное событие. Для этого необходимо в инспекторе объектов выбрать нужное событие и дважды на его значении щелкнуть мышью.

После этого открывается окно редактора кода, с автоматически созданной заготовкой функции, которая будет вызываться при возникновении данного события.

Программисту остается написать код тела этой функции. Визуально пример изменения стандартной обработки события активизации окна показан на Рис. 3.

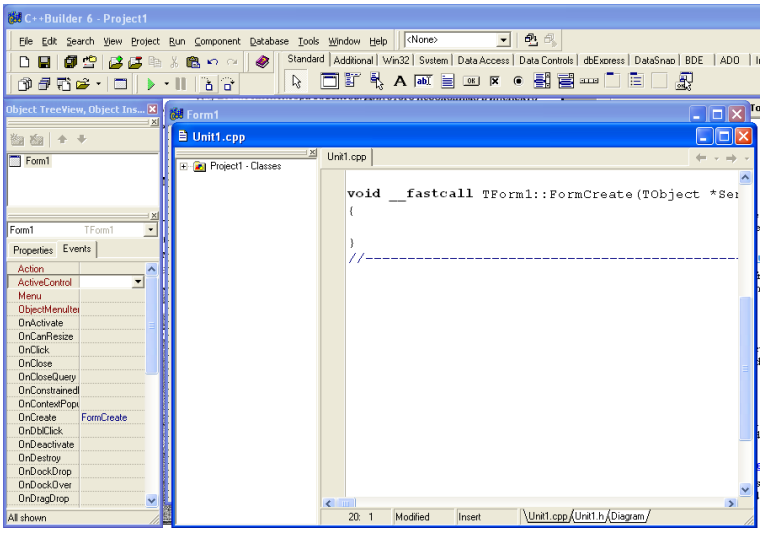

Рис. 3. Изменение стандартной обработки события

С каждым объектом связано событие «по умолчанию». Например, для объекта Button (Кнопка) событие «по умолчанию» - Click (Щелчок). Событие «по умолчанию» содержится первым на вкладке Events. Если дважды щелкнуть мышью на самом объекте, то открывается окно редактора кода, с автоматически созданной заготовкой, куда вписывается тело функции, которая будет вызываться при возникновении события «по умолчанию».

Редактор форм позволяет визуально разместить на форме с помощью мыши требуемые объекты.

Редактор кода представляет собой текстовый редактор, позволяющий программисту создавать фрагменты кода программы, чтобы запрограммировать поведение созданных объектов приложения.

Создание Windows–приложений

C++ Builder при запуске создает по умолчанию проект для Windows–приложения. При этом создаются минимум шесть следующих файлов.

1. Исходный файл проекта с расширением `cpp`. В нем содержится функция для запуска Windows–приложения `WinMain()` и другие команды, использующиеся при старте.

2. Исходный файл главной формы с расширением `cpp`. Содержит объявление объекта главной формы и определение методов, участвующих в управлении формой и ее компонентами.

3. Заголовочный файл главной формы с расширением h. Содержит определение класса главной формы.

4. Файл ресурсов приложения с расширением res. Представляет собой двоичный файл, который описывает графический значок приложения.

5. Файл ресурсов главной формы с расширением `dfm`. Также двоичный файл, который содержит описание графических объектов главной формы.

6. Информационный файл проекта с расширением `bpr`. Представляет собой текстовый файл, содержащий данные об именах исходных файлов и форм проекта, параметрах компилятора, именах подключаемых библиотечных файлов.

Пример 1. Создание Формы с кнопками.

Пусть необходимо создать приложение, интерфейс которого изображен на Рис. 4.

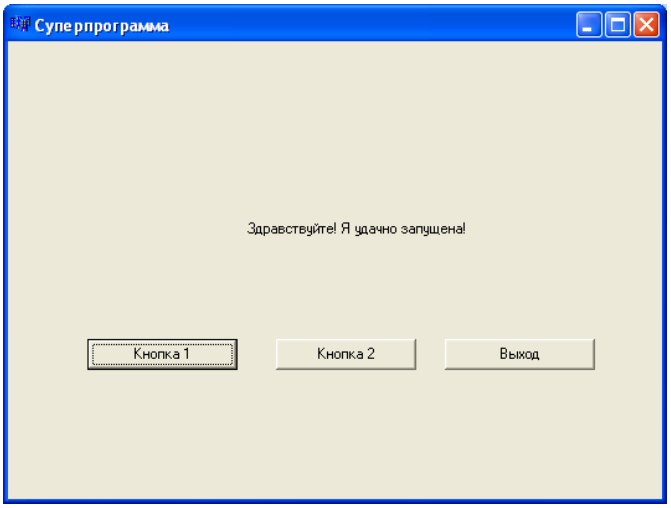

Рис. 4. Интерфейс приложения

Интерфейс приложения содержит следующие элементы:

- приветственную надпись,
- функциональные кнопки.

При запуске приложения отображается приветственная надпись – «Здравствуйте! Я удачно запущена!». Функциональные кнопки размещены в нижней части окна. При нажатии первой кнопки приветственная надпись должно заменяться надписью «Привет от первой кнопки». При нажатии второй кнопки текущая надпись должна быть заменена на «Привет от второй кнопки!» и третья кнопка, если она видна в окне, должна исчезнуть, а если не видна, то она должна появиться. При нажатии на третью кнопку программа должна завершить свою работу. Названия кнопок следующие: «Кнопка 1», «Кнопка 2», «Выход». Название формы – «Суперпрограмма».

Последовательность действий по созданию приложения для данного примера такова:

1.После входа в систему C++ Builder будет по умолчанию создан файл проекта под Windows-приложение.

2.Необходимо сохранить пустой проект в заданную папку командой File/Save Project as... (Сохранить проект как...). В этом случае все файлы проекта будут сохранены в указанной папке. Если не сохранить пустой проект в заданную папку, а сразу начать дизайн и отладку, то файлы проекта будут сохранены в служебных каталогах C++ Builder .

3.В редакторе форм можно визуально просмотреть внешний вид окна нашего приложения. Сейчас в нем еще ничего нет (Рис. 1).

Создание элементов интерфейса приложения

Создание надписей в окне приложения

Надпись в рабочей области окна создается с помощью компонента `Label` следующими действиями:

- нажмите кнопку A в палитре компонентов `Standard`,

- щелчком мыши поместите компонент на форму в нужное место,

- растяните его до желаемых размеров,

- перейдите в окно Object Inspector,

- заполните требуемым текстом значение свойства Caption (Наазвание) этого компонента.

В нашем случае свойство Caption заполняем текстом «Здравствуйте! Я удачно запущена!»

Создание кнопок

Кнопки оболочки создаются с помощью компонента Button. Для создания кнопки:

- нажмите кнопку  в палитре компонентов Standard,

- щелчком мыши поместите компонент Button на форму,

- переместите в выбранное Вами место,

- растяните его до желаемых размеров,

-введите название для кнопки в окошко свойства Caption этого компонента.

В нашем случае поместим на форму три кнопки с соответствующими названиями.

Заменим надпись «Form1» в верхней части окна на «Суперпрограмма». Для этого щелкнем мышью на форме, в верхней части инспектора объектов отобразится имя формы – Form1:Tform1. Заменим значение «Form1» свойства Caption на "Суперпрограмма". Название окна будет изменено. Результат показан на Рис. 5.

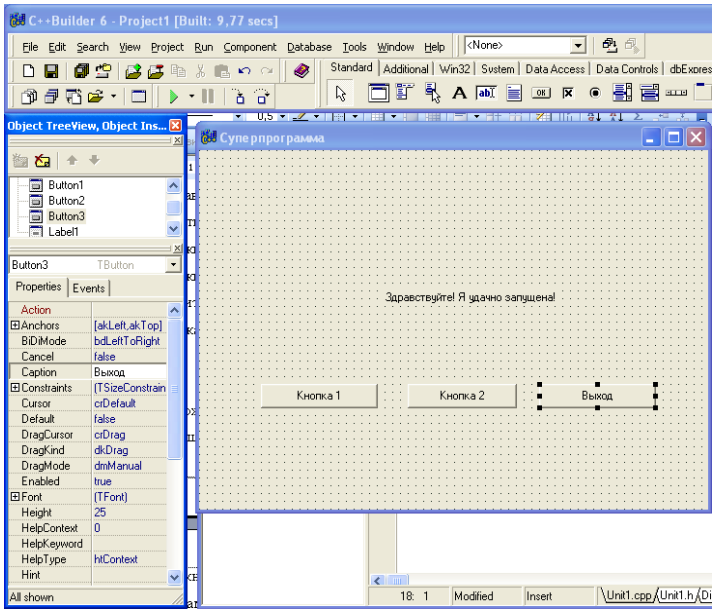

Рис. 5. Изменение свойств компонентов через Инспектор объектов

Создание поведения объектов приложения

Запрограммируем действия, выполняемые в программе при нажатии на кнопки.

6. Дважды щелкнем в редакторе форм на первой кнопке. После этого откроется окно редактора кода, в котором находится автоматически созданная заготовка функции, выполняемой при возникновении события «по умолчанию» – нажатии кнопки.

Между фигурными скобками введем фрагмент программы, заменяющий текст на метке

"Здравствуйте! Я удачно запущена" на текст «Привет от первой кнопки!».

Он будет следующим:

```
Label1->Caption="Привет от первой кнопки!";
```

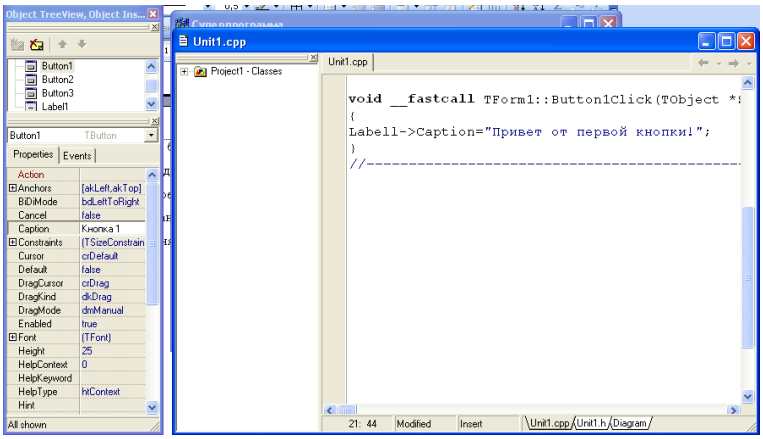

Рис. 6. Программное изменение свойства объекта

Теперь нужно запрограммировать поведение второй кнопки. Для этого нужно снова перейти на форму. Чтобы перейти из окна редактора кода в окно формы используется кнопка  Toggle Form/Unit («Переключиться на редактор форм или редактор кода») или клавиша F12. Нажав эту кнопку, вернемся в редактор форм.

Для второй кнопки аналогичным образом введем следующий программный код:

```
Label1->SetTextBuf("Привет от второй кнопки");
if (Button3->Visible==true) Button3->Hide(); else Button3->Show();
```

Заметим, что в этом случае для управления объектом Label1 использован метод SetTextBuf(), позволяющий заменять текст сообщения.

Результат достигается аналогичный, как и для первой кнопки, разница состоит в подходах к управлению объектом.

Чтобы управлять видимостью третьей кнопки, анализируется ее свойство Visible, и в зависимости от его значения true (истина) или false (ложь) вызывается метод Hide() (Скрыть объект) или Show() (Показать объект).

Условие совпадения свойства Visible со значением true задается при помощи двойного равенства.

Для третьей кнопки аналогичным образом введем код функции, закрывающей пнриложение:

```
Close();
```

На этом создание приложения закончено. Осталось убедиться в его работоспособности. На панели инструментов нажмем кнопку "Запустить программу на решение"  . После этого программа будет откомпилирована и, если нет ошибок, сгенерированный exe-файл будет автоматически запущен на выполнение. На экране появится готовое приложение. Нажимая кнопки на форме, остается убедиться в правильности его работы.