Содержание

Глава 2. Основы работы в системе MATLAB®

Глава 3. Математическая библиотека MATLAB® для языка С

Предисловие

Данная книга посвящена изложению вопросов совместного использования богатой библиотеки визуальных компонент Borland C++ Builder и возможностей языка C++ с мощными математическими процедурами MATLAB®. Книга содержит основы программирования в Borland C++ Builder и на MATLAB, описание математических библиотек MATLAB для C/C++ и компиляторов MATLAB (включая последнюю версию). Рассматриваются примеры программ на Borland C++ Builder, которые используют математические библиотеки MATLAB. Хотя книг по основам системы MATLAB и по программированию на C/C++ и на Borland C++ Builder достаточно много, в настоящее время нет ни одной книги, в которой были бы изложены вопросы совместного использования MATLAB и C/C++ и Borland C++ Builder.

Книга предназначена преподавателям и студентам вузов по специальностям, близким к прикладной математике (математическая подготовка читателя предполагается в пределах технического вуза), профессиональным программистам C++, которые сталкиваются с проблемами реализации математических алгоритмов на C++, и MATLAB-программистам, которые хотят использовать гибкость языка C++ и богатую библиотеку визуальных компонент Borland C++ Builder для реализации алгоритмов MATLAB в виде законченных и независимых от MATLAB приложений.

Углубленное программирование на MATLAB предполагает использование языка C для создания MEX-файлов, создание автономных приложений на MATLAB, создание библиотек (dll) функций. Это направление в MATLAB представлено тремя пакетами расширений: MATLAB C Math Library, MATLAB C++ Math Library и MATLAB Compiler – математические библиотеки C/C++ MATLAB и Компилятор MATLAB. На сегодняшний день эти пакеты не отражены ни в одной книге по MATLAB.

Стандартная математическая библиотека C++ содержит около 20 элементарных функций. Поэтому создание программы на C++, где даже немного используется математика, возможно, но требует серьезной работы по программированию математических функций и процедур. Даже такая простая задача, как вычисление ранга матрицы, требует от программиста очень больших усилий. Система MATLAB в своих пакетах MATLAB C/C++ Math Library и MATLAB Compiler предлагает богатую коллекцию математических функций (более 400), которые могут быть эффективно использованы программистами C/C++. При программировании на C++ имеются определенные трудности использования математических библиотек MATLAB. Предлагаемая книга рассказывает, как решить эти проблемы. Освоение технологии использования математических библиотек

MATLAB в Borland C++ Builder позволит создавать полноценные Windows-приложения с развитой графической средой, в которых возможна реализация сложных математических алгоритмов.

Рассмотрим кратко содержание данной книги по главам.

Первая глава представляет краткое введение в программирование на Borland C++ Builder 6. Она предназначена для читателей, которые немного владеют программированием, например на MATLAB, но не на C/C++. Дается общая характеристика системы, ее возможности, инструментальные средства Borland C++ Builder. Рассматриваются файлы, создаваемые Borland C++ Builder при построении проекта, их характеристики и взаимосвязи. Дается краткий обзор программирования на C++.

Вторая глава содержит первоначальные сведения о системе MATLAB. Она предназначена для читателей, которые немного владеют программированием, но не на MATLAB. Глава содержит описание работы с числовыми массивами, массивами символов, массивами ячеек и структур и основы программирования в среде MATLAB.

Глава 3 является введением в Математическую библиотеку C. Данная библиотека представляет собой набор более чем 400 математических подпрограмм, написанных на языке C. Библиотека предлагает определенные типы массивов, аналогичных массивам MATLAB, и свой язык для использования математических функций и массивов в программах на C. Глава содержит детальный анализ ряда тестовых примеров C-программ, в которых используются функции Математической библиотеки C (все эти примеры входят в инсталляционный пакет Математической библиотеки C).

Глава 4 посвящена описанию Математической библиотеки MATLAB C++. Данная библиотека основана на Математической библиотеке C, главное, что изменилось, – это синтаксис вызова математических процедур библиотеки. Язык Математической библиотеки C++ стал почти таким же интуитивно понятным и кратким, как в MATLAB. Математическая библиотека C++ использует класс mwArray для представления всех типов массивов MATLAB.

Пятая глава посвящена описанию пакета расширения MATLAB Compiler версии 3 (для MATLAB 6.x). Компилятор MATLAB из m-файлов MATLAB может создать MEX-файлы, C или C++ исходные коды, автономные приложения, библиотеки общего доступа, C-коды S-функций для использования с Simulink, дополнения Excel и COM-объекты. Изложение материала сопровождается обсуждением многочисленных тестовых примеров MATLAB 6 (все эти примеры входят в инсталляционный пакет MATLAB Compiler).

Глава 6 является введением в Компилятор версии 4 (для MATLAB R14) и его использование. Данная версия Компилятора существенно отличается от предыдущей. В частности, Компилятор 4 включает все математические библиотеки MATLAB (они в MATLAB R14 даже не выделены в отдельный пакет расширения). Возможности Компилятора огромны. Компилятор MATLAB поддерживает почти все функциональные возможности MATLAB.

В последней главе 7 на примерах рассматривается создание Windows-приложений на Borland C++ Builder с использованием Математической библиотеки MATLAB C++. Полные тексты этих примеров записаны на компакт-диске, прилагаемом к данной книге.

Книга имеет три приложения. Первое приложение содержит полный перечень функций и процедур Математической библиотеки MATLAB C++. Приложение 2 содержит описание библиотеки классов C++ для Компилятора MATLAB последней версии 4. Третье приложение есть компакт-диск с примерами программ, рассматриваемых в данной книге.

Книга написана при содействии корпорации MathWorks в соответствии с программой MathWorks поддержки книг, посвященных MATLAB®. Авторы благодарят компанию MathWorks за предоставленную возможность использования документации и лицензионного программного обеспечения MATLAB® для написания этой книги.

Глава 1. Основы программирования на Borland C++ Builder

Borland C++ Builder является средством быстрой разработки приложений (Rapid Application Development Environment), позволяющим создавать приложения на языке C++ и использующим в качестве основы библиотеку визуальных компонентов (Visual Component Library), позаимствованную у Borland Delphi. Сведения, приведенные в данной главе, ориентированы на читателя, уже имеющего навыки программирования. В главе будут рассмотрены: основы языка C++, среда разработки C++ Builder, основные приемы, применяемые при проектировании пользовательского интерфейса, а также простые примеры построения математических графиков.

1.1. Среда разработки

C++ Builder представляет собой однооконное приложение (SDI), главное окно которого (верхняя часть) содержит панель меню программы, настраиваемую панель инструментов (левая часть главного окна) и палитру компонентов (правая часть главного окна). Помимо этого, по умолчанию при запуске C++

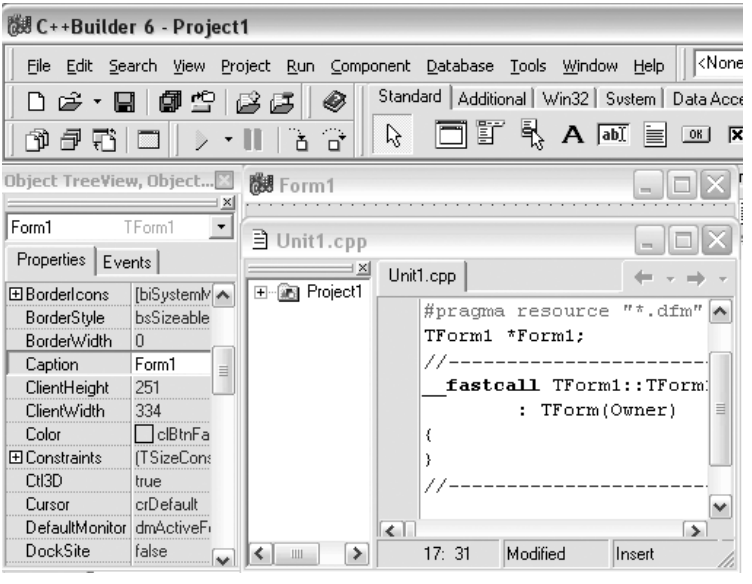

Рис.1.1.1. Среда разработки C++ Builder

Builder отображаются окно дерева объектов (Object Tree View), окно инспектора объектов (Object Inspector), форма нового приложения (например, Form1) и окно редактора кода (например, Unit1.cpp).

1.1.1. Панель инструментов

Рассмотрим панель инструментов, на которой стандартно размещены 16 кнопок быстрого доступа (все эти кнопки соответствуют определенным пунктам меню). Этими кнопками являются: «новый файл», «открыть проект», «сохранить», «сохранить все», «добавить к проекту», «выбрать модуль», «выбрать форму», «запустить», «пауза» и т. д. Панель инструментов можно полностью перенастроить под собственные предпочтения, внеся в нее требуемые комманды меню или удалив ненужные.

1.1.2. Палитра компонентов

Палитра компонентов предназначена для выбора компонентов или элементов управления (таких как ActiveX), размещаемых на форме разрабатывемого программного приложения. Все компоненты C++ Builder располагаются в *палитре компонентов*, выполненной в виде многостраничного блокнота, каждая закладка которого объединяет компоненты в смысловую группу. Особенностью среды разработки C++ Builder является то, что она позволяет создавать собственные компоненты и добавлять их в палитру компонентов, а также настраивать палитры компонентов для разных проектов.

1.1.3. Редактор форм

Формы являются основой для разработки интерфейса приложений C++ Builder. Создание пользовательского интерфейса приложения заключается в добавлении в окно формы компонентов (объектов) C++ Builder и задания некоторой реакции компонента на события. Для помещения компонента в форму необходимо щелкнуть на его пиктограмме в палитре компонентов, а затем щелкнуть на нужном месте в вашей форме, при этом новому компоненту (экземпляру класса) будет автоматически присвоено имя и заполнены некоторые свойства, например отвечающие за местоположение компонента в форме. При этом в заголовочном файле формы появится строка типа `TLabel *Label1` (в `public` разделе объявлений). Если вы хотите разместить несколько копий компонентов одного типа, то, выбрав в палитре необходимый компонент и удерживая клавишу Shift, кликая мышкой, помещайте в форму сколько угодно компонентов. При завершении процесса тиражирования щелкните на быструю кнопку «стрелка» в палитре компонентов.

1.1.4. Редактор кода

Окно редактора кода разделено на расположенный слева проводник классов (ClassExplorer) и непосредственно сам редактор кода справа. Проводник классов позволяет оперативно следить за развитием собственной программы, а также ви-

деть поля и методы используемых программой классов. При необходимости проводник классов может быть закрыт. Все открытые в среде исходные файлы отображаются непосредственно в редакторе кода, который представляет собой окно редактирования текста с закладками, соответствующими именам открытых файлов. В левой части редактора кода расположено поле для установки *точек прерывания* выполнения программы. Они отображаются жирными красными точками. Устанавливать их можно, просто щелкая мышью по полю слева от текста, причем делать это можно и перед запуском, и во время выполнения программы. Недействующие точки останова помечаются крестиком. Также стоит отметить то, что редактор кода C++ Builder позволяет выделять цветом ключевые слова и синтаксические конструкции. При удачном задании цветов (по умолчению оно таковым не является) чтение кода становится намного легче.

1.1.5. Инспектор объектов

Инспектор объектов работает совместно с редактором форм. Инспектор объектов предназначен для установки значений свойств, доступных на этапе разработки. Окно инспектора объектов состоит из: Селектора компонентов, Закладки Properties (свойства) и Закладки Events (события). О свойствах и событиях мы подробнее поговорим при изучении компонентов.

Селектор представляет собой выпадающий комбинированный список, в котором отображены все компоненты формы. Выбранный в списке компонент будет выделен в форме.

Закладка *Properties* отображает все свойства текущего компонента, доступные на этапе разработки. Если вы выбрали в форме несколько компонентов, то инспектор объектов отобразит их общие *свойства*. В левом столбце вы видите имя свойства, а в правом – установленное значение. Практически все значения можно поменять прямым вводом значения в правом столбце, но во многих случаях редактор свойств содержит список возможных значений. У некоторых свойств (например, Font) стоит знак плюс. Это означает, что данное свойство представляет собой другой объект, также содержащий набор свойств. Щелкнув на плюс, вы раскроете весь набор. Для свойств, представляющих собой объекты, есть второй способ редактирования свойств: нужно щелкнуть в столбце значений на многоточие, при этом раскроется соответствующее окно редактора свойств. Для редактирования некоторых свойств можно пользоваться только многоточием.

Закладка *Events* содержит список событий, которые может обрабатывать данный компонент. Использование этой закладки заключается в том, чтобы соответствующим событиям назначить имена функций, которые будут выполняться при возникновении этого события. Это позволяет нескольким компонентам, в случае указания одной и той же функции, одинаково реагировать на одно и то же *событие*. Если вы хотите написать отдельный код для этого события, то самый простой вариант – это просто дважды щелкнуть в столбце значений соответствующего события. При этом автоматически будет создана функция для этого события.

1.1.6. Менеджер проекта

Файлы, образующие программное приложение (формы и модули), собраны в проект. Менеджер проекта показывает список файлов и модулей приложения, позволяя осуществлять навигацию между ними. Менеджер проектов можно вызвать, выбрав пункт меню View/Project Manager. По умолчанию вновь созданный проект получает имя Project1.cpp.

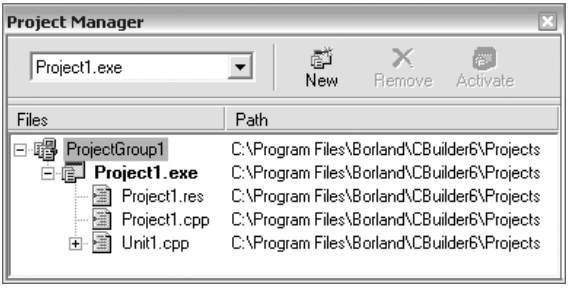

Рис. 1.1.2. Менеджер проекта

Первоначально проект содержит файлы для одной формы и исходного кода одного модуля. Однако большинство проектов содержат несколько форм и модулей. Можно добавлять существующие формы и модули к проекту: для этого нужно щелкнуть правой кнопкой мыши по проектному файлу (в окне менеджера проекта отображается жирным шрифтом) и выбрать пункт Add контекстного меню менеджера проектов, затем выбирать модуль или форму, которую нужно добавить. Аналогичным образом формы и модули можно удалять в любой момент из разработки проекта. Однако, из-за того что форма с модулем всегда связаны, нельзя удалять одно без удаления другого, за исключением случая, когда модуль изначально не имеет связи с формой. Удалить модуль из проекта можно, используя кнопку Remove менеджера проектов или через контекстное меню.

Если выбрать пункт меню **Options** в менеджере проектов, откроется диалоговая *панель опций* проекта (рис. 1.1.3), в которой можно выбрать главную форму приложения, определить, какие формы будут создаваться динамически, каковы будут параметры компиляции и компоновки модулей.

1.1.7. Контекстное меню

Важным элементом среды разработки C++ Builder является контекстное меню, появляющееся при нажатии на правую клавишу мыши и предлагающее быстрый доступ к наиболее часто используемым командам.

1.1.8. Контекстная помощь

Разумеется, C++ Builder обладает встроенной системой контекстно-зависимой помощи, доступной для любого элемента интерфейса и являющейся обшир-

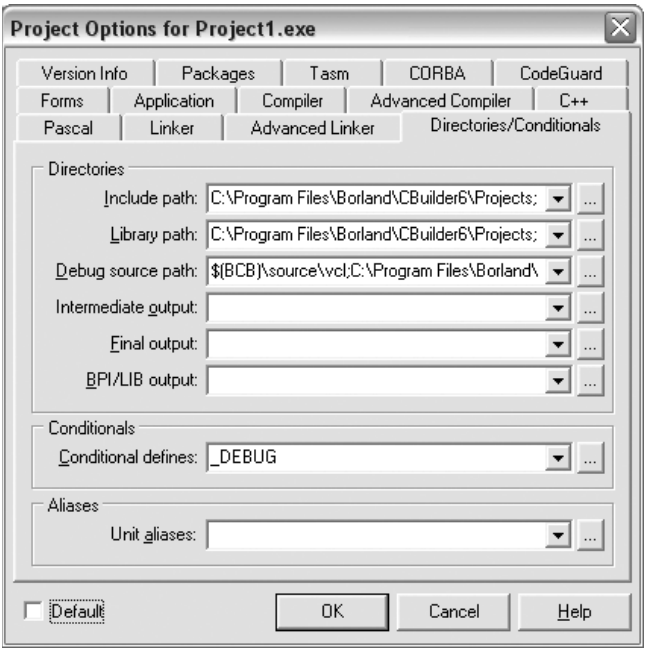

Рис. 1.1.3. Установка опций проекта

ным источником справочной информации о C++ Builder. Достаточно установить курсор на неизвестный элемент или часть кода и нажать клавишу <F1>. Особенно полезна помощь во время написания программ, так как в ней описаны все объекты библиотеки визуальных компонентов (VCL). Практически все описанные свойства и методы классов сопровождаются примерами использования.

1.2. Компоненты C++ Builder

Под компонентами C++ Builder понимаются классы C++, которые размещены в палитре компонентов и могут быть перенесены в разрабатываемую форму. Компоненты C++ Builder разделяются на визуальные и невизуальные. *Визуальные компоненты* во время выполнения разработанного приложения отображаются на форме точно так же, как и во время проектирования. Этот класс компонент, исходя из названия, необходим для представления данных на экране компьютера. Примерами таких компонент являются поля редактирования, статический текст, кнопки, списки и т. д. Следует также отметить, что во время выполнения программы, при необходимости, они могут быть видимыми и невидимыми. *Невизуальные компоненты* во время исполнения программы не отображаются, а в процессе проектирования отображаются на форме в виде соответствующих иконок. Такие компоненты обладают определенной функциональностью (например, вызывают стандартные диалоги открытия или закрытия файлов, обрабатывают системные события, обеспечивают доступ к базам данным и т. д.)

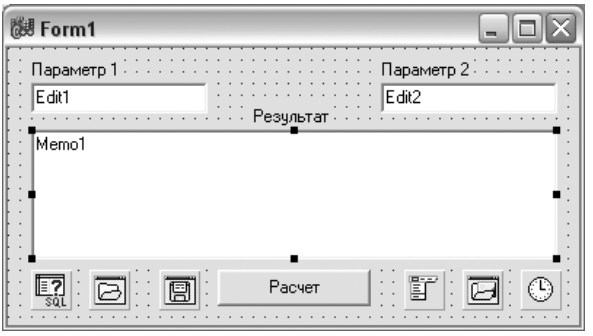

Рис. 1.1.4. Пример отображения визуальных и невизуальных компонентов

1.2.1. Свойства компонентов

Свойства являются атрибутами компонента, определяющими его внешний вид и поведение во время работы программы. Многие свойства компонента, отображаемые в колонке свойств инспектора объектов, имеют значения, устанавливаемые по умолчанию (например, высота кнопок). Инспектор объектов отображает только опубликованные (`published`) свойства компонентов. Помимо published-свойств, компоненты могут и чаще всего имеют общие (`public`) свойства, которые доступны только во время выполнения приложения. Можно определять свойства во время проектирования или написать код для видоизменения свойств компонента во время выполнения приложения.

1.2.2. События компонентов

Закладка событий инспектора объектов (Events) показывает список всех возможных событий, распознаваемых компонентом. Программирование в операционных системах с графическим пользовательским интерфейсом (в частности, Microsoft Windows) предполагает описание реакций создаваемого приложения на те или иные события, в то время как сама операционная система занимается постоянным опросом компьютера с целью выявления наступления какого-либо события. Каждый компонент имеет свой собственный набор событий. Создание программ в C++ Builder в основном и заключается в написании функций (называемых обработчиками событий), связываемых с событиями компонентов, помещенных в форму компонентов. Создавая обработчик того или иного события, вы поручаете программе выполнить написанную функцию, если это событие произойдет.

Для того чтобы добавить *обработчик событий*, необходимо выбрать компонент на форме с помощью мыши (или Селектор компонентов Инспектора объектов), которому необходим обработчик событий, затем открыть страницу событий инспектора объектов и дважды щелкнуть левой клавишей мыши на колонке значений рядом с событием, чтобы заставить C++ Builder сгенерировать прототип обработчика событий и показать его в редакторе кода. При этом автоматически

генерируется текст пустой функции, и редактор открывается в том месте, где следует вводить код. Курсор позиционируется внутри операторных скобок { и }. Далее нужно ввести код, который должен выполняться при наступлении события. Обработчик событий может иметь параметры, которые указываются после имени функции в круглых скобках.

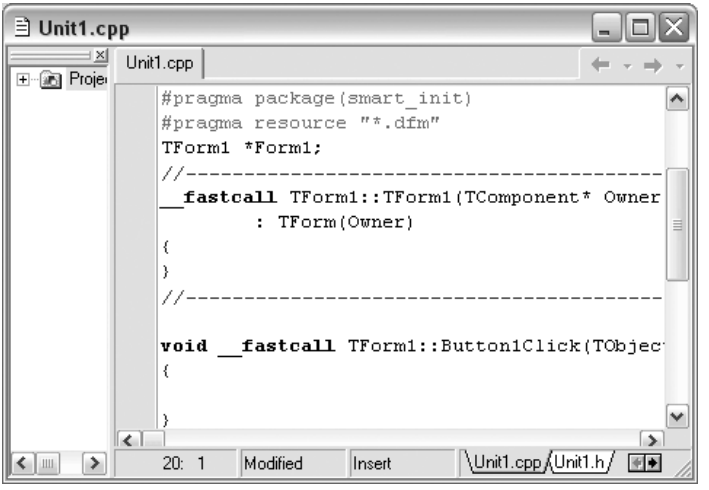

Рис. 1.1.5. Пример пустой функции обработчика событий

1.2.3. Методы компонентов

Методами компонента являются все функции, которые связаны с компонентом и которые объявляются в классе C++, описывающем компонент. Создавая обработчик события, можно вызывать методы компонентов, описанных в модуле формы, используя нотацию «->», например:

```
Edit1->Show();
```

1.3. Основы C++ как языка создания программ в C++ Builder

Этот раздел представляет собой краткий обзор языка программирования C++, являющегося основным языком создания программ в C++ Builder. Эта глава всего лишь является справочным пособием для людей, обладающих знанием других процедурных языков программирования, а приведенные в примерах конструкции языка программирования C++ демонстрируют основные особенности языка.

1.3.1. Комментарии

Часто бывает полезно вставлять в программу игнорируемый компилятором текст, который предназначается в качестве комментария для человека, стараю-

щегося впоследствии разобраться с тем, что же ваша программа все-таки делает. В C++ это можно сделать одним из двух способов:

Символы `/*` начинают комментарий, заканчивающийся символами `*/`. Вся эта последовательность символов эквивалентна символу пропуска (например, символу пробела). Чаще всего они используются для многострочных комментариев, но иногда и для изъятия частей программы (отключения части функциональности кода) при отладке. Следует помнить, что комментарии `/* */` не могут быть вложенными.

Символы `//` указывают на наличие однострочного комментария, который заканчивается в конце той же самой строки, на которой появились эти символы. Используется он для коротких комментариев. Его также можно использовать для отключения работы единичной строки кода программы.

Символы `//` можно использовать для того, чтобы закомментировать символы `/*` или `*/`, а символами `/*` можно закомментировать последовательность `//`.

Если вы используете комментарии при отладке кода, не забывайте после доведения программы до рабочего состояния прибирать «строительный мусор», удаляя закомментированные куски кода. По личному опыту не раз убедился, что они вносят существенное замешательство при разборе чужих, да и собственных программ, особенно если это делается спустя годы и в редакторах без подсветки синтаксических конструкций языка.

1.3.2. Типы и описания

Каждая поименованная переменная в программе и каждое выражение имеют тип, определяющий операции, которые могут над ними производиться. Например, описание

```
int MaxSize;
```

определяет, что `MaxSize` имеет тип `int`, то есть `MaxSize` является целочисленной переменной. *Описание переменной* – это оператор, который сообщает программе имя переменной и задает ее тип. Тип определяет правила использования имени или выражения. Для целых определены такие *операции*, как +, –, * и /. Например, после подключения файла `stream.h`, переменная или выражение типа `int` может также быть вторым операндом <<, при выводе в некоторый поток `ostream`.

Для переменных инициализация необязательна, но настоятельно рекомендуется. Тип объекта определяет не только то, какие операции могут к нему применяться, но и смысл этих операций. В C++ есть несколько основных типов и несколько способов создавать собственные.

1.3.3. Основные типы

Основные типы данных строятся из следующих ключевых слов:

```
char      __int8          long
double    __int16         signed
float     __int32         short
int       __int64         unsigned
```

Из этих ключевых слов вы можете создать любые целочисленные типы данных и типы с плавающей точкой, которые больше известны как арифметические типы. К целочисленным типам могут быть применены модификаторы `long`, `short`, `signed` и `unsigned`. Подключаемый файл `limits.h` содержит определения диапазонов значений для всех базовых типов данных.

Целочисленные типы

`char`, `short`, `int` и `long` вместе со своими беззнаковыми вариантами описывают целочисленные типы данных. Целочисленные типы указаны ниже вместе с условиями образования синонимов:

- `char`, `signed char`, будут являться синонимами, если по умолчанию `char` является знаковым;
- `unsigned char`;
- `char`, `unsigned char`, будут являться синонимами, если по умолчанию `char` является беззнаковым;
- `signed char`;
- `int`, `signed int`;
- `unsigned`, `unsigned int`;
- `short`, `short int`, `signed short int`;
- `unsigned short`, `unsigned short int`;
- `long`, `long int`, `signed long int`;
- `unsigned long`, `unsigned long int`.

Модификаторы `signed` и `unsigned` могут быть использованы с `char`, `short`, `int` и `long`. Если при описании переменных используются только ключевые слова `signed` и `unsigned`, то это соответственно означает `signed int` и `unsigned int`.

Ключевые слова `long` и `short` могут использоваться только с `int`, означая соответственно длинное и короткое целое значение.

Так как ANSI C не диктует размера или внутреннего представления таких типов, указывая только формы (`short`, `int` и `long`), то не меняет значимости «`sizeof(char) <= sizeof(short) <= sizeof(int) <= sizeof(long)`». И поэтому вполне законно, что все целочисленные типы могут быть одним и тем же. В 32-битных программах на C++ Builder типы `int` и `long` идентичны и занимают 32 бита. Для типа char, по определению, имеет размер единица. Соотношение между типами с плавающей точкой можно записать «`sizeof(float) <= sizeof(double)`». Предполагать что-либо еще насчет основных типов неразумно. В частности, то, что целое достаточно для хранения указателя, верно не для всех платформ.

Типы с плавающей точкой

Представления и наборы значений для типов с плавающей точкой являются зависящими от реализации, то есть каждая реализация C может определить их по-своему. Авторы C++ Builder использовали форматы с плавающей точкой

Конец ознакомительного фрагмента.
Приобрести книгу можно
в интернет-магазине
«Электронный универс»
e-Univers.ru