1 Типы, переменные, стандартный ввод-вывод. Игра «Утраченный клад»

Программирование игр — взыскательная отрасль. В этой сфере и разработчик, и машина должны работать на пределе возможностей. Но игра здесь действительно стоит свеч. Итак, в этой главе вы познакомитесь с основами C++ — важнейшего языка для программирования первоклассных игр. В частности, вы научитесь:

- отображать вывод в окне консоли;
- выполнять арифметические вычисления;
- использовать переменные для хранения данных, их извлечения и манипуляций с ними;
- получать пользовательский ввод;
- работать с константами и перечислениями;
- работать со строками.

Введение в C++

На языке C++ постоянно работают миллионы программистов по всему миру. Это один из популярнейших языков для написания компьютерных программ и важнейший язык, на котором создаются крупнобюджетные компьютерные игры.

Язык C++ изобретен Бьерном Страуструпом и непосредственно основан на языке C. Фактически почти весь язык C сохранился в C++ в виде своеобразного подмножества. Однако C++ предлагает улучшенные способы решения многих задач и включает инновационные возможности, которых не было в языке C.

Использование языка C++ при программировании игр

Существует ряд причин, по которым разработчики игр активно пользуются языком C++. Рассмотрим некоторые из них.

- **Он быстр.** Грамотно написанные программы на C++ могут работать просто молниеносно. Одной из основных проектных характеристик языка C++ была высокая производительность. Если же в вашей программе требуется добиться просто запредельной производительности, то C++ позволяет работать и с *ассемблером*. Ассемблер — это самый низкоуровневый человекочитаемый язык программирования, взаимодействующий непосредственно с аппаратным обеспечением компьютера.
- **Он гибок.** C++ — это мультипарадигмальный язык, поддерживающий различные стили программирования, в том числе *объектно-ориентированное программирование*. Но, в отличие от многих современных языков программирования, C++ не имеет жесткой привязки к какой-либо парадигме программирования.
- **Он хорошо поддерживается.** Поскольку язык C++ уже очень давно используется в игровой индустрии, по нему доступно множество ресурсов. Это и графические API, и возможности 2D и 3D, и игровая физика, и звуковые движки. Программист, работающий с C++, может использовать готовый код, значительно ускоряя разработку новых игр.

Создание исполняемого файла

Исполняемым называется файл, используемый для запуска программы — не только игры, но и любого приложения. Исполняемый файл в несколько этапов создается из *исходного кода*, написанного на C++. Исходный код — это набор инструкций, написанных на C++. Этот процесс проиллюстрирован на рис. 1.1.

1. Сначала программист открывает *редактор* и пишет в нем исходный код на языке C++. Получается файл, обычно имеющий расширение `.cpp`. Такой инструмент напоминает обычный текстовый редактор. В этой программе программист может писать, редактировать и сохранять исходный код.
2. Сохранив файл с исходным кодом, программист запускает *компилятор* C++. Это программа, считывающая исходный код и преобразующая его в *объектный файл*. Обычно объектные файлы имеют расширение `.obj`.
3. Далее компоновщик, также именуемый *редактором связей*, связывает объектный файл с нужными внешними файлами и таким образом создает *исполняемый файл*, обычно имеющий расширение `.exe`. Теперь пользователь (геймер) может запустить программу, просто открыв исполняемый файл.

СОВЕТ

Здесь я описал процесс в упрощенном виде. Чтобы создать сложное приложение на C++, требуется собрать в одну программу множество файлов исходного кода, написанных программистом (нередко целой командой программистов).

Рис. 1.1. Создание исполняемого файла на основе исходного кода C++

Чтобы автоматизировать этот процесс, программисты обычно используют универсальный инструмент, называемый IDE *(интегрированная среда разработки)*. Как правило, в составе IDE есть редактор, компилятор, компоновщик, а также другие инструменты. Существует популярная (при этом свободно распространяемая) IDE для операционной системы Windows. Она называется Visual Studio Express 2013 для Windows ПК. Подробнее почитать об этой IDE, а также скачать ее можно по адресу www.visualstudio.com/downloads/download-visual-studio-vs.

Исправление ошибок

Рассказывая о создании исполняемого файла из исходного кода C++, я не упомянул об одной детали — об ошибках. Человеку свойственно ошибаться, а программисту — особенно. Даже самые лучшие программисты пишут такой код, в котором и после первой, и после пятой проверки обнаруживаются какие-нибудь ошибки. Программисту приходится исправлять ошибки и все переделывать. Далее перечислены наиболее распространенные типы ошибок, встречающиеся в C++.

- **Ошибки компиляции.** Они возникают на этапе компиляции кода. Поэтому объектный файл не получается. Это могут быть *синтаксические ошибки* — значит, компилятор чего-то «не понял». Зачастую они возникают из-за банальных

опечаток. Компилятор может выдавать предупреждения. Хотя обычно такие предупреждения погоды не делают, то, о чем в них говорится, следует расценивать как ошибки — исправлять и перекомпилировать.

- **Ошибка компоновки.** Такие ошибки возникают в процессе компоновки (связывания) и могут означать, что программа не может найти какие-то данные, ссылки на которые в ней имеются. Обычно для устранения таких ошибок достаточно исправить проблемную ссылку и повторить процесс компиляции/компоновки.
- **Ошибки времени исполнения.** Они возникают при запуске исполняемого файла. Если программа выполнит недопустимую операцию, она может внезапно завершиться. Но существуют и более тонкие ошибки времени исполнения — так называемые *логические ошибки*, в результате которых программа просто начинает действовать непредусмотренным образом. Если вам когда-нибудь доводилось играть в игру, персонаж которой вдруг начинает ходить в воздухе (но по сценарию этого делать не может), то вам встретилась типичная логическая программная ошибка.

НА ПРАКТИКЕ

Как и все разработчики софта, игровые компании напряженно работают, стремясь создавать продукты, начисто лишенные таких ошибок (багов). Последняя линия защиты от багов — это сотрудники отделов обеспечения качества, в просторечии называемые тестировщиками. Тестировщики-геймеры зарабатывают на жизнь, играя в игры, но эта работа далеко не такая классная, как может показаться на первый взгляд. Тестировщик должен проходить одни и те же этапы игры снова и снова, иногда сотни раз подряд, пробуя все возможные и невозможные варианты и тщательно описывая все замеченные аномалии. При такой монотонной работе зарплаты тестировщиков отнюдь не высоки. Однако получить такую работу — замечательный способ закрепиться в штате игровой компании.

Понятие о стандарте ISO

Стандарт ISO (International Organization for Standardization — *Международная организация по стандартизации*) для языка C++ — это определение C++, в точности описывающее, как должен работать этот язык. Кроме того, в данном стандарте ISO описана группа файлов, называемая *стандартной библиотекой*. Стандартная библиотека содержит файлы-кирпичики, позволяющие программисту решать распространенные задачи, например получать ввод и отображать вывод. Стандартная библиотека серьезно облегчает жизнь программисту и предоставляет ему фундаментальный код, избавляя от необходимости постоянно изобретать велосипед. Стандартная библиотека применялась при написании всех программ из этой книги.

СОВЕТ

Стандарт ISO часто называют ANSI (American National Standards Institute — Американский национальный институт стандартов) или ANSI/ISO. Эти аббревиатуры соответствуют разным организациям и комитетам, рассматривавшим и формировавшим этот стандарт. Простейшее обозначение кода C++, соответствующего всем требованиям стандарта ISO, — стандартный C++.

Разрабатывая программы для этой книги, я пользовался интегрированной средой Microsoft Visual Studio Express 2013 для Windows ПК. Компилятор, входящий в состав этой интегрированной среды разработки, безупречно соответствует стандарту ISO, поэтому вы без проблем сможете компилировать, компоновать и запускать любые программы, работая с любым другим современным компилятором. Правда, если вы работаете в Windows, рекомендую пользоваться именно Microsoft Visual Studio Express 2013 для Windows ПК.

СОВЕТ

Пошаговые инструкции о том, как написать, сохранить, скомпилировать и запустить программу Game Over с помощью Microsoft Visual Studio Express 2013 для Windows ПК, вы найдете в приложении 1 к этой книге, которое называется «Создание первой программы на языке C++». Если вы пользуетесь другой средой разработки или другим компилятором, почитайте документацию по этим инструментам.

Пишем первую программу на C++

Итак, довольно теории. Переходим к самому интересному: сейчас вы напишете свою первую программу на языке C++. Хотя эта программа и очень проста, она обладает всеми характерными чертами типичной программы. Кроме того, в ней будет продемонстрировано, как выводить текст в окне консоли.

Знакомство с программой Game Over

Как правило, начиная осваивать новый язык, разработчик пишет на нем программу «Hello World». Она отображает на экране слова «Hello World», которые можно перевести как «Привет, мир». Программа Game Over является игровой вариацией этой классической задачи и выводит на экран слова «Game Over!» («Игра окончена!»). На рис. 1.2 эта программа показана в действии.

Рис. 1.2. Ваша первая программа выводит на экран два слова, способные огорчить любого геймера (используется с разрешения компании Microsoft)

Вы можете скачать исходный код этой программы на сайте Cengage Learning (www.cengageptr.com/downloads). Программа находится в каталоге к главе 1 (Chapter 1), имя файла — `game_over.cpp`.

СОВЕТ

Чтобы быстро найти этот код на ресурсе www.cengageptr.com/downloads, можете сразу поискать эту книгу. Например, ее удобно искать по идентификационному номеру (ISBN). ISBN этой книги — 9781305109919.

```
// GameOver
// Первая программа на C++
#include <iostream>
int main()
{
    std::cout << "Game Over!" <<std::endl;
    return 0;
}
```

Комментирование кода

Первые две строки в данной программе — это комментарии:

```
// GameOver
// Первая программа на C++
```

Строки комментариев компилятор игнорирует, поскольку предназначены они для людей, а не для машины. Комментарии помогают другим программистам, которые будут читать ваш код, понять ход ваших мыслей. Но комментарии могут пригодиться и вам самим. Они напоминают, как вы решили ту или иную задачу, тогда как из кода это может быть неочевидно.

Чтобы создать комментарий, достаточно начать строку с двух прямых слешей (//). Весь остальной текст до конца физической строки машина сочтет частью комментария. Таким образом, на одной и той же строке сначала может идти код C++, а затем — комментарий, но после кода до начала комментария должны стоять два прямых слеша.

СОВЕТ

Существует еще одна разновидность комментариев. Они называются «комментарии в стиле C» и могут занимать несколько строк. Такой комментарий необходимо начинать с символов /* и заканчивать символами */. Весь текст между двумя этими маркерами будет считаться частью комментария.

Использование пробелов

Сразу после комментариев в нашей программе идет пустая строка. Компилятор игнорирует пустые строки. На самом деле компилятор игнорирует любое пустое пространство — пробелы, табуляцию и символы перехода на новую строку. Как и комментарии, пробелы предназначены не для машины, а для человека.

При разумном использовании пробелов повышается удобочитаемость программы. Например, можно оставлять пустые строки между большими блоками кода, если весь код в конкретном блоке взаимосвязан. Я также использую пробелы (точнее, табуляцию) в начале каждой из двух строк между фигурными скобками, чтобы немного развести эти скобки.

Включение других файлов

Следующая строка программы — это директива препроцессора. Строки с такими директивами начинаются с символа #:

```
#include<iostream>
```

Препроцессор начинает действовать еще до того, как к работе подключится компилятор. Он выполняет подстановку текста в зависимости от различных директив. В данном случае строка начинается с директивы `#include`, приказывающей препроцессору включить в программу содержимое другого файла.

Здесь я включаю файл `iostream`, входящий в состав стандартной библиотеки, поскольку в этом файле содержится код, упрощающий отображение вывода. Я заключаю имя файла в угловые скобки — это сигнал компилятору искать этот файл среди тех, что поставляются вместе с компилятором. Файл, включаемый в ваши программы таким образом, называется *заголовочным*.

Определение функции main()

Следующая непустая строка — это заголовок функции `main()`:

```
intmain()
```

Функция — это единица в составе программного кода. Функция должна выполнить определенную работу и вернуть значение. В данном случае код `int` означает, что функция должна вернуть целочисленное значение. Во всех заголовках функций после имени функции ставятся две круглые скобки.

Во всех программах на языке C++ должна быть функция `main()` — это своеобразная исходная точка программы. Здесь начинается действие.

Следующая строка отмечает начало функции:

```
{
```

А самая последняя строка — конец функции:

```
}
```

Все функции начинаются и заканчиваются фигурной скобкой, а весь код между ними относится к функции. Код между двумя фигурными скобками называется блоком, обычно он снабжается отступами, чтобы было понятнее, что этот блок — цельная единица. Блок кода, образующий целую функцию, называется телом функции.

Отображение текста в стандартном выводе

Первая строка в теле функции `main()` содержит слова `GameOver!`, затем идет новая строка. Все это мы видим в окне консоли:

```
std::cout<< "GameOver!" <<std::endl;
```

Слова `GameOver!` — это *строка* или *последовательность символов*, точнее, последовательность символов, которые отображаются при печати. Технически данная последовательность является *строковым литералом* — то есть перед нами в буквальном смысле находятся именно эти символы в кавычках.

`cout` — это объект, определенный в файле `iostream` и используемый для отправки данных в поток стандартного вывода. В большинстве программ (включая эту) поток стандартного вывода — это просто окно консоли на экране компьютера.

Я применяю *оператор вывода* (`<<`) для отправки строки в `cout`. Оператор вывода можно сравнить с воронкой: он берет все, что наливают в горлышко, и перенаправляет эту информацию в указанное место. Эта строка направляется в стандартный вывод, то есть в окно консоли на экране.

Перед объектом `cout` я ставлю префикс `std`, чтобы подсказать компилятору, что имею в виду объект `cout` из стандартной библиотеки. `std` — это *пространство имен*. Пространство имен можно сравнить с региональным телефонным кодом. Оно обозначает группу, к которой относится тот или иной объект. Пространство имен `std` отделяется от собственно имени `cout` оператором разрешения видимости. Перед пространством имен ставится *оператор разрешения* (`::`).

Наконец, я отправляю в стандартный вывод код `std::endl`. `endl` также является объектом из пространства имен `std` и определяется в файле `iostream`. Операция отправки `endl` в стандартный вывод аналогична тому, как если бы мы нажали клавишу Enter в окне консоли. Кроме того, если бы я отправил в окно консоли еще одну последовательность символов, то она началась бы с новой строки.

Не исключаю, что запомнить весь этот материал сразу сложно. Поэтому изучите рис. 1.3, где наглядно представлены взаимосвязи всех описанных элементов.

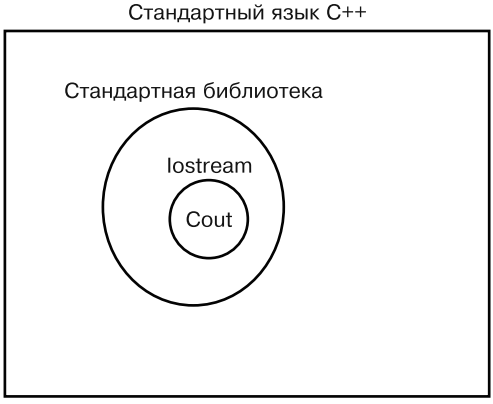

Рис. 1.3. В стандартной реализации языка C++ имеется набор файлов, называемый «стандартная библиотека». В стандартной библиотеке есть файл iostream, в котором определяются различные сущности, в том числе объект cout