Выражения отношений

В C++ эту возможность отношения реализуют операции отношений. доступны шесть операций отношений для сравнения чисел. Поскольку символы представлены своими ASCII-кодами, эти операции можно также применять и для

сравнения символов. Они не работают со строками в стиле С, но работают с объектами класса string. Каждое сравнивающее выражение возвращает булевское (типа bool) значение true, если сравнение истинно, и false — в противном случае, поэтому

данные операции хорошо подходят для применения в проверочных условиях циклов. (Старые реализации оценивали истинные выражения как 1 и ложные — как 0.)

*< // Меньше чем*

*<= //Меньше или равно*

*== //Равно*

*> //Больше чем*

*>= //Больше или равно*

*! = // Не равно*

*for (х = 20; х > 5; х--) // продолжать, пока х больше чем y*

*for (х = 1; у != х; + + х) // продолжать, пока у не равно х*

*for (cin >> х; х == 0; cin » х) ) // продолжать, пока х равно 0*

*х + 3 > у — 2 // выражение 1*

*соответствует такому:*

*(х + 3) > (у - 2) // выражение 2*

*и не соответствует этому:*

*х + (3 > у) — 2 // выражение 3*

*musicians == 4 // сравнение*

*musicians =4 // присваивание*

**Сравнение строк в стиле С**

Предположим, что вы хотите узнать, хранится ли в символьном массиве слово mate. Если word — имя массива, то следующая проверка не сделает того, что вы ожидаете:

*word == "mate"*

Вспомните, что имя массива — это синоним его адреса. Аналогично, строковая константа в двойных кавычках является синонимом ее адреса. Таким образом, приведенное выражение сравнения не проверяет идентичность строк; оно проверяет,

находятся ли они по одному и тому же адресу. Ответом будет — нет, даже если эти две строки состоят из одинаковых символов. Поскольку C++ обрабатывает строки в стиле С как адреса, вы мало что получите, если попытаетесь воспользоваться операциями отношений для сравнения строк. Вместо этого можете обратиться к библиотеке строк в стиле С и применять для их сравнения функцию strcmp (). Эта функция принимает в виде аргументов два адреса

строк. Это значит, что аргументы могут быть указателями, строковыми константами либо именами символьных массивов. Если две строки идентичны, функция возвращает значение 0. Если первая строка предшествует второй в алфавитном порядке, strcmp () возвращает отрицательное значение, если же первая строка следует за второй в алфавитном порядке, то strcmp () возвращает положительное значение. Говорить "в последовательности сопоставления в системе" будет более точно, нежели "в алфавитном порядке". Это значит, что символы сравниваются в соответствии с их системными кодами. Например, в коде ASCII заглавные буквы имеют меньшие коды, чем строчные, поэтому заглавные буквы предшествуют строчным в порядке сортировки. То есть строка "Zoo" предшествует строке "aviary". Тот факт, что сравнение основано на значениях кодов, также означает, что заглавные и строчные буквы отличаются, поэтому строка "F00" отличается от "foo".

В некоторых языках, таких как BASIC и стандартный Pascal, строки, сохраненные в массивах разных размеров, по определению не равны друг другу. Но строки в стиле С ограничиваются нулевым символом, а не размером содержащего их массива. Это значит, что две строки могут быть идентичными, даже если содержатся в массивах разного размера:

*char big[80] = "Daffy"; //5 букв плюс \0*

*char little [6] = "Daffy"; // 5 букв плюс \0*

Кстати, хотя и нельзя применять операции отношений для сравнения строк, вы можете использовать их для сравнения символов, потому что символы относятся к целочисленным типам. Поэтому следующий код является допустимым, по крайней мере, для наборов символов ASCII и Unicode, для отображения символов по алфавиту:

*for (ch = 'a'; ch <= ' z'; ch++)*

*cout <<*

сопрограмма в листинге 5.11 использует strcmpO в проверочном условии цикла for. Эта программа отображает слово, изменяет его первую букву, отображает его снова и продолжает это делать до тех пор, пока strcmpO не определит, что word

содержит строку "mate". Обратите внимание на включение файла est ring — в нем содержится прототип stremp ().

**Листинг 5.11. compstrl. ерр**

*// compstrl.срр — сравнение строк с использованием массивов*

*#include <iostream>*

*#include <cstring> // прототип для strcmpO*

*int main ()*

*{*

*using namespace std;*

*char word[5] = "?ate";*

*for (char ch = 'a'; stremp (word, "mate"); ch+ + )*

*{*

*cout « word « endl;*

*word[0] = ch;*

*}*

*cout « "After loop ends, word is " « word « endl; // вывод word по завершении цикла*

*return 0;*

Ниже показан вывод программы из листинга 5.11:

*?ate*

*aate*

*bate*

*cate*

*date*

*eate*

*fate*

*gate*

*hate*

*iate*

*jate*

*kate*

*late*

*After loop ends, word is mate*

**Замечания по программе**

Программа в листинге 5.11 содержит несколько интересных моментов. Один из них, конечно же — проверка цикла. Вы хотите, чтобы цикл продолжался до тех пор, пока word не совпадает с "mate". To есть до тех пор, пока strcmp () сообщает, что строки не одинаковы. Наиболее очевидный способ сделать это выглядит следующим образом:

strcmp(word, "mate") != О // строки не одинаковы Этот оператор имеет значение 1 (true), если строки не одинаковы, и значение О (false), если они совпадают. Но как насчет самого вызова strcmp (word, "mate") ? Он возвращает ненулевое значение (true), если строки отличаются, и 0 (false) — если строки эквивалентны. По сути, функция возвращает true, если строки разные, и false, если они одинаковы. Вы можете использовать только саму функцию вместо всего сравнивающего выражения. Это даст тот же результат при меньшем объеме кода. К тому же, это — традиционный способ применения strcmp () в языках С и C++.

**Проверка на эквивалентность или порядок**

*Функцию strcmp () можно применять для проверки строк в стиле С на эквивалентность или порядок. Следующее выражение истинно, если strl и str2 идентичны:*

*strcmp(strl, str2) == О*

*Выражения*

*strcmp(strl, str2) != О*

*и*

*strcmp(strl, str2)*

*истинны, когда strl и str2 не идентичны. Показанное ниже выражение истинно, если strl по порядку предшествует str2:*

*strcmp(strl, str2) < О*

*И, наконец, следующее выражение истинно, когда strl следует за str2:*

*strcmp(strl, str2) > О*

Таким образом, функция strcmp () может играть роль операций ==, ! =, < и >, в зависимости от того, как будет составлено проверочное условие. Далее в compstrl .cpp используется операция инкремента для прохода переменной ch по всему алфавиту:

ch++ Операции инкремента и декремента можно применять в отношении символьных переменных, т.к. тип char на самом деле является целочисленным, поэтому данная операция в действительности изменяет целочисленный код, хранящийся в

переменной. К тому же обратите внимание, что использование индекса массива упрощает изменение отдельных символов в строке:

word[0] = ch;

**Сравнение строк класса string**

Жизнь станет немного легче, если вместо строк в стиле С использовать строки класса string, поскольку этот класс позволяет применять операции отношений для выполнения сравнений. Это становится возможным благодаря определению функций класса, которые "перегружают", или переопределяют, операции. показано, как включать это средство в классы, но с практической точки зрения все, что вам нужно знать сейчас — это то, что с объектами класса string можно

использовать операции сравнения. В листинге 5.12 представлен преобразованный код из

листинга 5.11, в котором вместо массива char применяется объект string.

**Листинг 5.12. compstr2 . срр**

*// compstr2.cpp — сравнение строк с использованием класса string*

*#include <iostream>*

*#include <string> // класс string*

*int main ()*

*{*

*using namespace std;*

*string word = "?ate";*

*for (char ch = 'a'; word != "mate"; ch+ + )*

*{*

*cout « word « endl;*

*word[0] = ch;*

*}*

*cout « "After loop ends, word is " « word << endl;*

*return 0;*

*}*

Замечания по программе

В листинге 5.12 приведенное ниже проверочное условие использует операцию сравнения с объектом string в левой части и строкой в стиле С — в правой части: word != "mate" Способ перегрузки операции != классом string позволяет применять ее, если хотя бы один из операндов является объектом string; второй операнд при этом может быть либо объектом string, либо строкой в стиле С. Класс string позволяет использовать объект string либо в качестве одиночной сущности, как в выражениях сравнения, либо в качестве агрегатного объекта, допускающего нотацию массивов для извлечения индивидуальных символов. Как видите, одного и того же результата можно достигнуть как с помощью строки в стиле С, так и с помощью объектов string, но программирование с объектами string проще и намного понятней. И, наконец, в отличие от большинства циклов for, которые вы видели до сих пор, два последних цикла не имеют счетчиков. То есть они не выполняют блок

операторов определенное количество раз. Вместо этого каждый из циклов проверяет определенное условие (равенство слову "mate"), которое сигнализирует о необходимости завершения. Для программ C++ более типично применять в таких случаях

цикл while, поэтому давайте рассмотрим его в следующем разделе.