Автор: Андрей
04.04.2009 17:23 - Обновлено 22.11.2009 11:52

Предисловие

Зачастую в программе некоторые действия должны выполняться при соблюдении некоторых условий, например, если переменная принимает определенное значение, или переменная превосходит какую-то величину, или еще в самых различных случаях. Для осуществления такой проверки и выполнения действий в зависимости от ее результатов служат инструкции if и switch.

Прежде чем перейти к их описанию, позволю себе небольшое лингвистическое отступление. Часто в литературе if называется не инструкцией, а *оператором*. Однако, в языке C++ существует такое понятие, как operator, что как раз логично перевести как "оператор". if же, в числе прочих, относится к понятию statement. Ввиду того, что слово "оператор" уже использовано для обозначения совершенно другого понятия, в моих статьях statement будет переводиться как "инструкция". Таким образом, вопреки довольно распространенной практике, мы будем говорить не про
оператор
if, а про
инструкцию
if.

Инструкция if

Инструкция if может быть представлена в двух видах:

if(условие) инструкция
и
if(условие) инструкция else инструкция

Первый вариант в случае, если выражение в скобках не равно нулю (логическое выражение истинно), выполняет инструкцию, следующую после скобок. Второй же в случае отличия от нуля выражения в скобках также выполняет инструкцию после скобок, а в противном случае - инструкцию, следующую за ключевым словом else. Если необходимо выполнить не одну, а несколько инструкций, то их последовательность заключается в фигурные скобки.

В условных инструкциях используются операторы (в этом случае - именно operator) сравнения == (равенство), != (неравенство), < (меньше), <= (меньше или равно), > (больше) и >= (больше или равно).

Замечу по поводу передаваемого аргумента, что сам if работает с численными

значениями, однако благодаря неявному преобразованию типов выражение в скобках может быть произвольным арифметическим выражением, логическим выражением или выражением с указателями. Результатом преобразовании в int значения булевой величины, равного false, будет ноль, а равного true - единица (величина отличная от нуля, чего условному оператору достаточно). Тип bool обеспечивает более наглядное представление работы с условиями, нежели численное представление. Результатом применения операторов сравнения является значение логического типа.

Указатель, равный нулю - это указатель, который ни на что не ссылается.

В логических выражениях часто используются логические операторы && (и), || (или), ! (не). Оператор "не" - унарный, т.е. выполняет действия над единственным аргументом, стоящим справа (возвращает true, если аргумент равен false, возвращает false, если аргумент равен true). Операторы "и" и "или" - бинарные, т.е. выполняют действия над двумя аргументами, стоящими справа и слева. Оператор "и" возвращает true, если оба аргумента равны true, во всех остальных случаях возвращает false. Оператор "или" возвращает true, если хотя бы один из аргументов равен true, если же оба равны false, оператор возвращает false.

Для иллюстрации всего вышеизложенного рассмотрим следующий пример:

```
    void f(int a, int b, char* p)
{
if(a == b) std::cout << "a равно b\n";
if(p) std::cout << *p;
if(true)
 {
 if(!(a * b - b * b) && (a != b)) std::cout << "b == 0\n";
 else return;
 }
else std::cout << "Такого не может быть!\n";
}
```

Функции передается два целых числа и указатель на символ. Вначале проверяется равенство чисел - if(a == b), и в случае равенства выводится сообщение об этом. Затем проверяется, является ли p указателем на инициализированную переменную, и если так, то выводится объект, на который ссылается указатель (*p). Третий if выполняется всегда, а вложенный - тогда, когда a не равно b и отрицание выражения в скобках отлично от нуля, т.е. выражение в скобках равно нулю. В противном случае функция завершается - else выполняет инструкцию return.

При наличии большого числа вложенных инструкций if с соответствующими else при правильно расставленных фигурных скобках не произойдет никакой путаницы - они будут наглядно объединены, и не будет допущено случайной ошибки, которую компилятор интерпретирует совершенно неожиданным образом. То же самое относится и к конструкциям внутри скобок вложенного if из этого примера - скобки делают длинную строку более читабельной, и гарантируют, что компилятор поймет все именно так, как вы хотели. При отсутствии же скобок может оказаться, что приоритет операторов отличается от того, что вы себе представляли и в результате получится невесть что.

Некоторые условные if-инструкции можно записывать в виде условных выражений.

Например,

```
if(a
```