**Выражения и Операторы**

В C++ имеется богатый набор операций, с помощью которых в выражениях образуются новые значения и изменяются значения переменных. Поток управления в программе задается с помощью операторов , а описания используются для введения в программе имен переменных, констант и т.д. Заметьте, что описания являются операторами, поэтому они свободно могут сочетаться с другими операторами.

**Выражения**

В C++ имеется большое число операций, и они будут объясняться там, где (и если) это потребуется. Следует учесть, что операции

|  |  |
| --- | --- |
| ~ | дополнение |
| & | И |
| ^ | исключающее ИЛИ |
| | | включающее ИЛИ |
| << | логический сдвиг влево |
| >> | логический сдвиг вправо |

применяются к целым, и что нет отдельного типа данных для логических действий.

Смысл операции зависит от числа операндов; унарное & является операцией взятия адреса, а бинарное & - это операция логического И. Смысл операции зависит также от типа ее операндов: + в выражении a+b означает сложение с плавающей точкой, если операнды имеют тип float, но целое сложение, если они типа int.

В C++ есть операция присваивания =, а не оператор присваивания, как в некоторых языках. Таким образом, присваивание может встречаться в неожиданном контексте; например, x=sqrt(a=3\*x). Это бывает полезно. a=b=c означает присвоение c объекту b, а затем объекту a. Другим свойством операции присваивания является то, что она может совмещаться с большинством бинарных операций. Например, x[i+3]\*=4 означает x[i+3]=x[i+3]\*4, за исключением того факта, что выражение x[i+3] вычисляется только один раз. Это дает привлекательную степень эффективности без необходимости обращения к оптимизирующим компиляторам. К тому же это более кратко.

В большинстве программ на C++ широко применяются указатели. Унарная операция \* разыменовывает\*3 указатель, т.е. \*p есть объект, на который указывает p. Эта операция также называется косвенной адресацией. Например, если имеется char\* p, то \*p есть символ, на который указывает p. Часто при работе с указателями бывают полезны операция увеличения ++ и операция уменьшения --. Предположим, p указывает на элемент вектора v, тогда p++ делает p указывающим на следующий элемент.

**Операторы Выражения**

Самый обычный вид оператора - оператор выражение. Он состоит из выражения, за которым следует точка с запятой. Например:

a = b\*3+c;

cout << "go go go";

lseek(fd,0,2);

Пустой оператор

Простейшей формой оператора является пустой оператор:

;

Он не делает ничего. Однако он может быть полезен в тех случаях, когда синтаксис требует наличие оператора, а вам оператор не нужен.

**Блоки**

Блок - это возможно пустой список операторов, заключенный в фигурные скобки:

{ a=b+2; b++; }

Блок позволяет рассматривать несколько операторов как один. Область видимости имени, описанного в блоке, простирается до конца блока. Имя можно сделать невидимым с помощью описаний такого же имени во внутренних блоках.

Операторы if

Программа в следующем примере осуществляет преобразование дюймов в сантиметры и сантиметров в дюймы; предполагается, что вы укажете единицы измерения вводимых данных, добавляя i для дюймов и c для сантиметров:

#include

main()

{

const float fac = 2.54;

float x, in, cm;

char ch = 0;

cout << "введите длину: ";

cin >> x >> ch;

if (ch == "i") { // inch - дюймы

in = x;

cm = x\*fac;

}

else if (ch == "c") // cm - сантиметры

in = x/fac;

cm = x;

}

else

in = cm = 0;

cout << in << " in = " << cm << " cm\n";

}

Заметьте, что условие в операторе if должно быть заключено в круглые скобки.

Операторы switch

Оператор switch производит сопоставление значения с множеством констант. Проверки в предыдущем примере можно записать так:

switch (ch) {

case "i":

in = x;

cm = x\*fac;

break;

case "c":

in = x/fac;

cm = x;

break;

default:

in = cm = 0;

break;

}

Операторы break применяются для выхода из оператора switch. Константы в вариантах case должны быть различными, и если проверяемое значение не совпадает ни с одной из констант, выбирается вариант default. Программисту не обязательно предусматривать default.

**Оператор while**

Рассмотрим копирование строки, когда заданы указатель p на ее первый символ и указатель q на целевую строку. По соглашению строка оканчивается символом с целым значением 0.

while (p != 0) {

\*q = \*p; // скопировать символ

q = q+1;

p = p+1;

}

\*q = 0; // завершающий символ 0 скопирован не был

Следующее после while условие должно быть заключено в круглые скобки. Условие вычисляется, и если его значение не ноль, выполняется непосредственно следующий за ним оператор. Это повторяется до тех пор, пока вычисление условия не даст ноль.

Этот пример слишком пространен. Можно использовать операцию ++ для непосредственного указания увеличения, и проверка упростится:

while (\*p) \*q++ = \*p++;

\*q = 0;

где конструкция \*p++ означает: "взять символ, на который указывает p, затем увеличить p."

Пример можно еще упростить, так как указатель p разыменовывается дважды за каждый цикл. Копирование символа можно делать тогда же, когда производится проверка условия:

while (\*q++ = \*p++) ;

Здесь берется символ, на который указывает p, p увеличивается, этот символ копируется туда, куда указывает q, и q увеличивается. Если символ ненулевой, цикл повторяется. Поскольку вся работа выполняется в условии, не требуется ни одного оператора. Чтобы указать на это, используется пустой оператор. C++ (как и C) одновременно любят и ненавидят за возможность такого чрезвычайно краткого ориентированного на выразительность программирования.

**Оператор for**

Рассмотрим копирование десяти элементов одного вектора в другой:

for (int i=0; i<10; i++) q[i]=p[i];

Это эквивалентно

int i = 0;

while (i<10) {

q[i] = p[i];

i++;

}

но более удобочитаемо, поскольку вся информация, управляющая циклом, локализована. При применении операции ++ к целой переменной к ней просто добавляется единица. Первая часть оператора for не обязательно должна быть описанием, она может быть любым оператором.

Например:

for (i=0; i<10; i++) q[i]=p[i];

тоже эквивалентно предыдущей записи при условии, что i соответствующим образом описано раньше.

**Описания**

Описание - это оператор, вводящий имя в программе. Оно может также инициализировать объект с этим именем. Выполнение описания означает, что когда поток управления доходит до описания, вычисляется инициализирующее выражение (инициализатор) и производится инициализация. Например:

for (int i = 1; i