**3 Операторы**

В С++ точка с запятой является признаком конца оператора.

**3.1 Пустой оператор**

Пустой оператор состоит из **;**. Он используется там, где по правилам языка должен находиться какой-либо оператор, а по логике пр-граммы там ничего выполнять не надо.

**3.2 Оператор-выражение**

Любое выражение, за которым следует **;** , является оператором. Такой оператор называется **оператор-выражение**.

Примеры:

*i++;*

*a = b + c;*

*c += (a < b)? a:b;*

*x+y; // Здесь результат не используется и будет выдано предупреждение.*

**3.3 Составной оператор**

Составной оператор иначе называют блоком. Он представляет фрагмент текста программы, заключенный в фигурные скобки и, как правило, объединяющий несколько операторов. Составной оператор должен использоваться там, где синтаксис языка требует наличие лишь одного оператора, а логика программы – сразу нескольких:

*{ i=5; c=sin(i∗x); c++;} // Это блок*

**3.4 Объявления**

В С++ объявления являются операторами языка и могут стоять там, где возможен любой другой оператор С++:

*s = 0.3; d /= s; int k = 5;*

*d = s + 2\*k;*

*double f = s + d; f ∗= k;*

**3.5 Условный оператор**

Имеется две формы условного оператора:

1) **if** (**выражение**) **оператор1**

2) **if** (**выражение**) **оператор1 else оператор2**

**Оператор1** выполняется в случае, если **выражение** принимает ненулевое значение. Если **выражение** принимает значение 0 (или ука-затель NULL), то выполняется **оператор2**.

Примеры:

*if (a > b) c = a – b; else c = b – a;*

*if (i < j) i++; else {j = i – 3; i ++;}*

При использовании вложенных операторов **if** текущий **else** всегда относится к самому последнему **if**, с которым еще не сопоставлен ни один **else**.

*void main ( )*

*{*

*int a = 2, b = 7, c = 3;*

*if (a > b) { if (b <c ) c = b;}*

*else c=a;*

*cout<<“c=“<<c<<“.\n”;*

*}*

Здесь результатом будет вывод строки **c=2**.

Если опустить фигурные скобки в операторе **if**, то программа примет вид:

*void main ( )*

*{*

*int a=2, b=7, c=3;*

*if (a > b)*

*if (b < c) c=b;*

*else c = a;*

*cout<<“c=“<<c<<“.\n”;*

*}*

Здесь **else** относится ко второму **if**. В результате выведется строка **c=3**.

**3.6 Оператор выбора *switch***

Этот оператор позволяет передать управление одному из несколь-ких помеченных метками операторов в зависимости от значения цело-численного выражения. Метки оператора **switch** имеют специальный вид:

**case** целая\_константа**:**

Вид оператора **switch**:

*switch (целое\_выражение){*

*[объявления]*

*[case константное\_целое\_выражение1:]*

***...***

*[case константное\_целое\_выражение2:]*

*[операторы]*

*...*

*[case константное\_целое\_выражение m:]*

*...*

*[case константное\_целое\_выражение n:]*

*[операторы]*

*[default:] [операторы] }*

Здесь [ ] означают необязательную часть оператора, а **...** говорит о том, что указанная конструкция может применяться сколько угодно раз. Блок после **switch**( ) называют телом оператора **switch**.

Схема выполнения оператора:

Сначала вычисляется выражение в круглых скобках (назовем его селектором).

Затем вычисленное значение селектора последовательно сравнивается с константным выражением, следующим за **case**.

Если селектор равен какому-либо константному выражению, стоящему за **case**, то управление передается оператору, помеченному соответствующим оператором **case**.

Если селектор не совпадает ни с одной меткой варианта, то управление передается на оператор, помеченный словом **default**.

Если **default** отсутствует, то управление передается следующему за **switch** оператору.

Отметим, что после передачи управления по какой-либо одной из меток дальнейшие операторы выполняются подряд. Поэтому, если необходимо выполнить только часть из них, нужно позаботиться о выходе из **switch**. Это обычно делается с помощью оператора **break**, который осуществляет немедленный выход из тела оператора **switch**.

Пример 1:

*int i, d;*

*cout<<“Задайте целое значение i\n”;*

*cin>>i;*

*switch ( i )*

*{*

*case 1: case2: case3: cout<<“ i=“<< i <<“\n”;*

*case 4: cout<<“ i=”<< i <<“ i^2= “<<i∗i<<“\n”;*

*d=3∗i – 4; cout<<“ d=“ << d <<“.\n”;*

*break;*

*case 5: cout<<“i=5.\n”; break;*

*default: cout<<“ Значение i меньше 1 или больше 5.\n”;*

*}*

Если ввести число 2, то будет напечатано

*i=2*

*i=2 i^2=4*

*d=2*

Если **i** равно 4 , то будет выведено

*i=4 i^2=16*

*d=8*

При **i**=5 будет выведено

*i=5*

При всех остальных значениях **i** будет напечатано

*Значение i меньше 1 или больше 5*

Пример 2:

*char sign;*

*int x, y, z;*

*cout<<”Задайте знак операции + – \* / \n”;*

*cin>>sign;*

*cout<<”Задайте* ***x*** *и* ***y*** *\n”;*

*cin>>x>>y;*

*switch (sign)*

*{*

*case ‘+’: z= x + y; break;*

*case ‘–’: z= x – y; break;*

*case ‘\*’: z= x \* y; break;*

*case ‘/’: if ( y = = 0 ){cout<<”Делить на нуль нельзя!\n”; exit(1);}*

*else z = x / y; break;*

*default: cout<<”Неизвестная операция!\n”; exit(1);*

*}*

Здесь exit(1) – вызов функции, который приводит к немедленному прекращению выполнения программы.

**3.7 Оператор цикла *while***

Оператор цикла с **предусловием** имеет вид

while (**выражение**) **оператор**

**Оператор** называют телом цикла.

При выполнении такого оператора сначала вычисляется значение **выражения**. Если оно равно 0, то **оператор** не выполняется и управление передается оператору, следующему за ним. Если значение **выражения** отлично от 0, то выполняется **оператор**, затем снова вычисляется **выражение** и т.д.

Возможно, что тело цикла не выполнится ни разу, если **выражение** сразу будет равно 0.

Пример 1:

*char с;*

*while ( cin.get(c) ) cout<<c;*

Здесь происходит копирование символов, включая пробельные, из потока **cin** (в данном случае из буфера клавиатуры) в поток **cout** (в данном случае на экран дисплея). Здесь функция **get(c)** (член класса) извлекает один символ из входного потока, включая пробельные символы. Возвращает ненулевое значение до тех пор, пока не достигнет конца файла (признак конца файла – ctrl-z).

Пример 2:

*while (1) { операторы ... }*

Это – бесконечный цикл.

Пример 3:

*char c;*

*while (( c = cin.get (c)) = = ‘ ‘ || c = = ‘\n‘ || c = = ‘\t’);*

Этот оператор цикла пропускает при считывании из потока **cin** так называемые пробельные символы. Здесь **get( )** – другая форма функции, считывающей из потока один символ. Она возвращает целое число – код символа, или число -1, если встретился признак конца файла.

**3.8 Цикл с постусловием *do-while***

Этот оператор цикла проверяет условие окончания в конце, после каждого прохода через тело цикла; поэтому тело цикла всегда выполняется по крайней мере один раз.

Вид оператора:

do **оператор** while (**выражение**)

Сначала выполняется **оператор**, затем вычисляется выражение и, если оно отлично от нуля, то оператор выполняется снова и т.д.

Если выражение становится равно нулю, цикл завершается.

Такой цикл удобно, например, использовать при проверке вводимых пользователем данных:

*int input=0;*

*int minvalue=10, maxvalue=150;*

*do { cout <<“Введите значение input \n”; cin >>input;*

*cout <<“ input=“ << input << “\n”; }*

*while ( input < minvalue || input > maxvalue );*

**3.9 Оператор *for***

Этот оператор цикла имеет вид:

for ( **оператор1 выражение1; выражение2** ) **оператор2**

**Оператор1** может быть объявлением, пустым оператором или оператором-выражением.

Наиболее распространенным является случай, когда **оператор1** и **выражение2** являются присваиваниями или обращениями к функциям, а **выражение1** – условным выражением. Этот цикл эквивалентен следующей конструкции:

**оператор1**

while (**выражение1**) { **оператор2 выражение2**; }

Иногда **оператор1** называют инициализатором цикла, а **выражение2** – реинициализатором.

Любая из трех частей может быть опущена, хотя точка с запятой обязательно должна оставаться. Если отсутствует проверка, то есть **выражение1**, то считается, как будто **выражение1** отлично от 0, так что

for ( ;; ){ ... } – бесконечный цикл и его надо каким-либо образом прервать.

Пример1:

*int n=20, s=0;*

*for ( int i = 1; i <= n; i++ ) s+ = i∗i;*

Здесь вычисляется сумма квадратов целых чисел от 1 до 20.

Пример2:

*double s, sum, den = 0.85, eps = 1e-10;*

*for ( s=1, sum=0; s>eps; s∗=den ) sum+=s;*

Здесь вычисляется сумма геометрической прогрессии 1, 1∗0.85, 1∗0.85∗0.85 и т.д., пока ее очередной член не станет меньше 10-10.

Пример 3. Вычислим и выведем на экран дисплея таблицу функции y=sin(x2) для *x*∈[0; π/2] с шагом π /20.

*#include <iostream>*

*#include <math.h>*

*#include <conio.h>*

*void main()*

*{*

*int n = 10;*

*double x0 = 0, xk = M\_PI\_2, y,*

*h = (xk – x0)/n, xt = xk + h/2;*

*clrscr();*

*cout<<” x y\n”*

*for(double x = x0; x < xt; x+ = h)*

*{*

*y = sin(x\*x);*

*cout.width(4); cout.precision(2);*

*cout<<x;*

*cout.width(10); cout.precision(4);*

*cout<<y<<’\n’;*

*}*

*}*

В этой программе используется константа M\_PI\_2, представляющая значение π/2 и определенная в заголовочном файле math.h. Обращение к функции cout.width(**k**) устанавливает ширину поля следующего вывода в **k** позиций, что позволяет выровнять вид таблицы. Функция cout.precision(**k**) задает число цифр, выводимых после десятичной точки. Функция clrscr(), прототип которой находится в conio.h, очищает экран.

Цикл for предпочтительнее там, где имеется простая инициализация и реинициализация, поскольку при этом управляющие циклом операторы наглядным образом оказываются вместе в начале цикла. Это наиболее очевидно в конструкции

for (i = 0; i < n; i ++),

которая применяется для обработки первых n элементов массива, аналогично оператору цикла for Паскаля. Аналогия, однако, не полная, так как границы цикла могут быть изменены внутри цикла, а понятие управляющей переменной в С++ отсутствует и переменная i сохраняет свое значение после выхода из цикла, какова бы ни была причина этого выхода.

**3.10 Оператор безусловного перехода**

Оператор безусловного перехода имеет вид

**goto метка**;

**Метка** – это имя, за которым следует ‘**:**’. Этот оператор передает управление оператору, помеченному указанной меткой. С его помощью удобно выходить сразу из нескольких вложенных циклов:

*for ( i = 0; i < n; i++)*

*for ( j = 0; j < m; j++)*

*for ( k = 0;k < l; k++) {*

*…*

*операторы;*

*…*

*if ( условие ) goto lab;*

*операторы;}*

*lab:; ...*

С помощью оператора **goto** можно переходить извне в тело блока, если при этом управление не передается через объявления имен, которые присутствуют в этом блоке.

**3.11 Оператор break**

Этот оператор осуществляет выход из тела цикла **for**, **while**, **do-while** или оператора **switch**, в котором он появился. При этом управле-ние передается на первый оператор после цикла.

Оператор не может обеспечить выход сразу из двух или более вложенных циклов.

**3.12 Оператор continue**

Этот оператор осуществляет переход на точку сразу за последним оператором тела цикла без выхода из цикла, так что дальнейшие итерации в цикле будут продолжаться.

Пример вывода четных чисел:

*for ( int num = 0; num < 100; num++ ){*

*if ( num % 2 ) continue;*

*cout << num << “\n“;*

*}*

Когда **num** становится нечетным, выражение num % 2 получает значение 1 и выполняется оператор, который передает управление на следующую итерацию цикла **for**, не выполняя вывода.

**3.13 Оператор return**

Этот оператор завершает выполнение функции, в которой он задан, и возвращает управление в вызывающую функцию.

Управление передается в вызывающую функцию в точку, непосредственно следующую за вызовом.

Если **return** присутствует в функции main(), то он вызывает прерывание выполнения программы.