Бесконечный цикл *for*. Оператор *break*.

Если пропустить второе управляющие выражение, которое определяет проверочное условие цикла for, то предполагается, что условие всегда будет *true*, поэтому цикл будет продолжаться бесконечно, если только не предусмотреть какой-либо другой способ выхода из него. На самом деле можно пропустить все управляющие выражения в скобках после *for*.

// Листинг 14

//использование бесконечного цикла для вычисления среднего

#include <iostream.h>

int main()

{

double value = 0.0; //Введенное значение

double sum = 0.0; //Сумма значений

int i = 0; //Счетчик значений

char indicator = 'n'; //Индикатор продолжения

for(;;)

{

cout <<endl

<<"input number:";

cin >> value; //Читать значение

++i; //Увеличить счетчик

sum +=value; //Добавить текущее к сумме

cout <<endl

<<"do you want input number (for finish press n)?:";

cin >> indicator; //Читать индикатор

if ((indicator == 'n') || (indicator == 'N'))

break; //Выход из цикла }

cout <<endl

<<"AV from " << i

<<"entered values equal " << sum/i

<<endl;

return 0;

}

input number:10

do you want input number (for finish press n)?:y

input number:20

do you want input number (for finish press n)?:y

input number:40

do you want input number (for finish press n)?:y

input number:50

do you want input number (for finish press n)?:n

AV from 4entered values equal 30

Данная программа вычисляет среднюю величину произвольного числа значений. После ввода каждого числа пользователь должен указать, желает ли он ввести еще одно значение, введя символ y или n.

После объявления и инициализации переменных, которые будут использованы, запускается цикл for без управляющих выражений, и следовательно без гарантий завершения. Непосредственно следующий за этим блок – субъект цикла, который должен повторяться.

Блок цикла выполняет следующие действия:

* Читает значение.
* Добавляет прочитанное из *cin* значение к *sum*.
* Проверяет, желаете ли продолжить ввод значений.

Первое действие внутри блока приглашает ввести число и читает введенное значение в переменную *value*. Веденное значение добавляется к *sum*, после чего задается вопрос о продолжении. Предлагается ввести символ *n*, если продолжение нежелательно. Введенный символ помещается в переменную *indicator* для последующей проверки на равенство *n* или *N*. Если равенства нет – цикл продолжается, в противном случае выполняется *break*.

Он приводит к немедленному выходу из цикла с передачей управления оператору, следующему за закрывающейся фигурной скобкой блока цикла.

далее выполняется вывод количества введенных значений и их средней величины.

Использование оператора *continue*

Имеется еще один оператор помимо *break*, служащий для управления циклом – *continue*. Записывается он очень просто:

*continue;*

Выполнение continue внутри цикла немедленно начинает следующую итерацию цикла, пропуская операторы тела цикла, которые оставалось выполнить в текущей операции. Можно продемонстрировать, как это работает, на следующем примере:

// Листинг 15

//использование оператора continue

#include <iostream.h>

int main ()

{

int i = 0, value =0,product = 1;

for(i =1; i<=10 ; i++)

{

cout <<"input integer number: ";

cin >>value;

if (value == 0)

continue;

product \*=value;

}

cout <<"result: " <<product <<endl;

return 0;

}

Результат

input integer number: 4

input integer number: 5

input integer number: 6

input integer number: 0

input integer number: 7

input integer number: 8

input integer number: 9

input integer number: 10

input integer number: 11

input integer number: 12

result: 79833600

Цикл читает 10 чисел с намерением получить произведение всех введенных значений. Оператор if проверяет, не был ли введен 0, и, если это так, то выполняется *continue*, чтобы сразу перейти к следующей итерации. Это делается для того, чтобы произведение введенных чисел не обратилось в ноль при первом же вводе нулевого значения. Понятно, что если ноль будет введен на последней итерации, цикл просто завершится. Безусловно, существуют и другие способы достичь того же результата, но *continue* представляет очень удобное средство, в частности, для сложных циклов, где вам может понадобиться пропускать остаток текущей итерации, начиная с разных точек его тела.

Влияние операторов *break* и *continue* на логику цикла *for* иллюстрирует рисунок.



Очевидно, что в реальных ситуациях необходимо использовать операторы *break* и *continue* с некоторой логикой проверки условий для определения того, когда цикл должен быть завершен, или когда текущая итерация должна быть пропущена. Можно применять операторы *break* и *continue* и с циклами других типов.