**Основные конструкции программирования**

**Оператор присваивания**

<переменная>=<выражение>;

Каждая арифметическая операция имеет комбинацию с оператором присваивания.

<операнды>+=<операнда 2>=> складывает значения своих операндов и результат записывается в левый операнд. Аналогичным образом работают: -=, /=, %=,\*=.

**Условный оператор**

Полная форма условного оператора:

if (<условие>)

<оператор 1>;

else <оператор 2>;

Пример.

Найти максимальное значение из 3-ех переменных.

int a,b,c;

cin>>a>>b>>c;

int max;

if (a>b)

max=a;

else max=b;

if (c>max) max=c;

cout<<max;

Условные операторы могут быть вложены один в другой. При этом глубина вложенности не ограничена, но конечна. При вложенности условных операторов else всегда относится к ближайшему перед их if.

**Оператор цикла с предусловием**

Цикл с предусловием считается циклом с входящим условием, так как условие проверяется до начала выполнения цикла.

Операторы, которые выполняются внутри цикла, называют телом цикла.

В цикле с предусловием тело цикла может не выполниться ни разу, может выполниться любое конечное число раз, а также бесконечное число раз, если условие цикла не корректно.

В общем виде структура цикла с предусловием следующая:

While (<условие>)

<тело цикла>;

Пример

Дано натуральное число n. Найти сумму его кратных 3 цифр.

int n,s=0;

cin>>n;

while (n>0)

{

if (n%10%3==0)

s+=n%10;

n/=10;

}

cout<<s<<endl;

Цикл while — цикл с неизвестным заранее числом повторений. Для корректной работы цикла while необходимо, чтобы переменные, входящие в условие, меняли свое значение в теле цикла.

**Оператор цикла с постусловием**

Цикл с выходным условием, при котором условие проверяется после выполнения тела цикла.

Тело цикла выполняется хотя бы один раз.

Структура следующая:

do

{ <тело цикла>;

}

while (<условие>);

Пример:

Дано натуральное n. Найти сумму кратных 3 цифр.

int n,s=0;

cin>>n;

do

{

if (n%10%3==0)

s+=%10;

n/=10;

} while(n);

cout<<s<<endl;

**Оператор цикла с параметром**

Цикл с параметром или счетчиком имеет следующую структуру: for (<начальная инициализация счетчика>; <условие выполнения>; <закон изменения счетчика>) <тело цикла>;

Работа цикла for заключается в следующем:

1 Производится начальная инициализация счетчика.

2 Проверяется условие выполнения цикла, если условие истинно, то выполняется тело цикла.

3 Изменяется значение счетчика по указанному закону.

4 Повторяются шаги 2-3.

Обычно цикл for используют, когда заранее известно количество шагов цикла.

Пример

Дано натуральное число n. Найти n.

int n, f=1;

cin>>n;

for (int i=1;i<=n;i++)

f\*=1;

cout<<f<<endl;

Закон изменения счетчика может представлять собой любое выражение допустимое над таким типом данных.

for (int i=100; i>0; i--) //100, 99, 98.. 1

cout<<i<< « »;

for (int i=1; i<100; i+=3) //1, 4, 7.. 97

cout<<i<< « »;

В цикле for все 3 параметра являются не обязательными, обязательно лишь наличие точек с запятой.

for (; ;)

<тело цикла>;

for (следует) while

<начальная инициализация цикла>;

while (<условие выполнения>)

{ <тело цикла>;

<закон изменения счетчика>;

}

Пример

int n,f=1;

cin>>n;

int i=1;

while (i<=n);

{f\*=i;

i++; }

while → for

for (;<условие>; )

<тело цикла>;

**Оператор выбора**

Оператор выбора используется в том случае, когда нужно сделать выбор из более чем 2-ух альтернатив.

Структура оператора выбора следующая:

switch (<выражение>)

{ case <метка 1>: <оператор 1>;

case <метка 2>: <оператор 2>;

…

case <метка n>: <оператор n>;

default: <оператор n+1>;

}

Работа оператора выбора заключается в следующем:

* вычисляется значение выражения;
* сравнивается со всеми метками;
* если находится метка равная вычисленному значению, то выполняется помеченный оператор.

Альтернатива default является необязательной, она срабатывает в том случае, если не нашлось подходящей метки.

Выражение может быть целочисленного либо символьного типа, а метки должны являться константами также целочисленного, либо символичного типа.

**Операторы break и continue**

Выполнение некоторого оператора, помеченного меткой, не останавливается на следующем case, поэтому каждую метку необходимо заканчивать оператором break, который позволяет завершить выполнение оператора выбора.

Пример

int d;

cout<< «Введи номер дня недели»<< endl;

cin>>d;

switch (d)

{

case 1: cout<< «понедельник»<<endl; break;

case 2: cout<< «вторник»<<endl; break;

….

case 7: cout<< «воскресенье»<<endl; break;

default: cout<< «неверный день недели»<<endl;

Оператор break нарушает естественный ход выполнения программы и обеспечивает выход из текущего оператора цикла, либо оператора выбора.

Пример.

bool f =true;

int x;

cin>>x;

if (x!=1)

{ for (int i=2;i<x;i++)

if (x%1==0)

{

f=false; // значит непростое

break; // выводим из текущего цикла по делителям

if (f)

cout<< «простое»<<endl;

else

cout<< «непростое»<< endl;

}

else

cout<< «не простое»<< endl;

Еще один оператор, который нарушает естественный ход выполнения программы — это оператор continue. Данный оператор позволяет досрочно завершить текущую итерацию цикла и перейти к следующей.

Пример

int x; // элемент последовательности

do

{

cin>>x;

If (x<0)

Continue;

if (x%2==0)

cout<<x;

}

while(x);