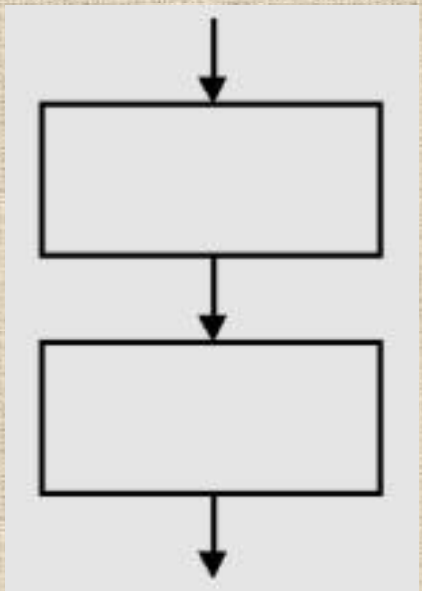
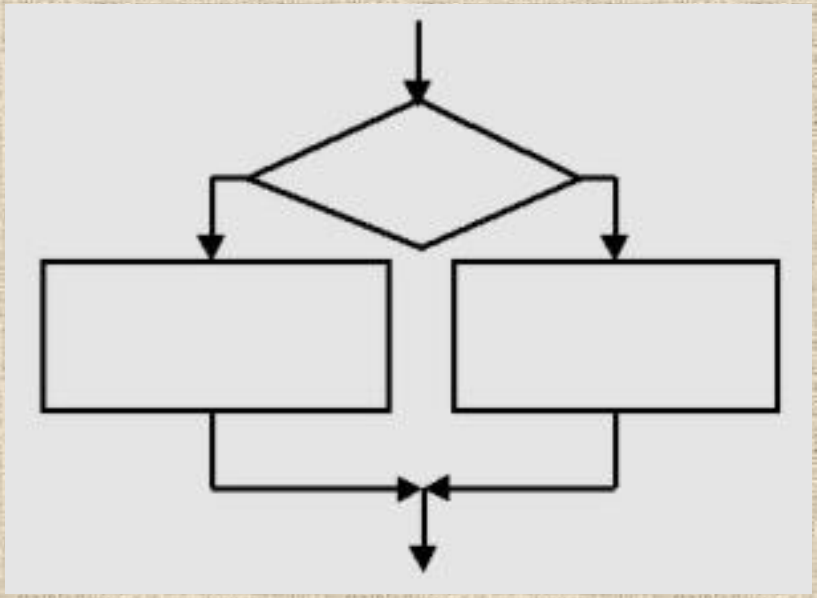
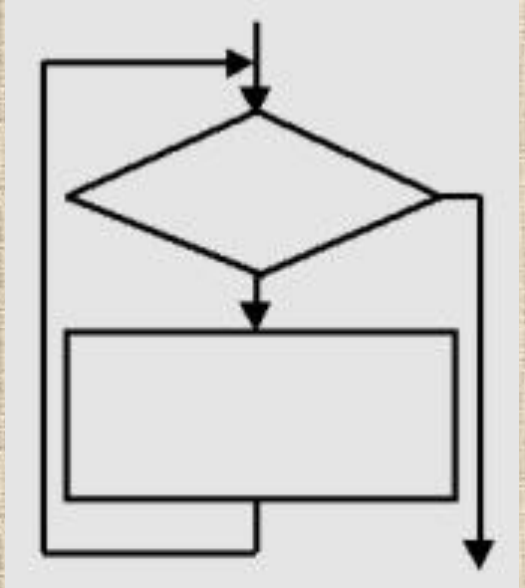
Лекция 5. Управляющие операторы

Базовые конструкции структурного программирования



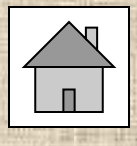




- Линейные
- Ветвление
- Цикл

Управляющие операторы. Ветвления

- Операция условие
- Условный оператор
- Оператор выбора

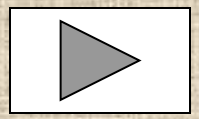


Управляющие операторы. Циклы

Операторы цикла используются для организации многократно повторяющихся вычислений.

Любой цикл состоит

- **тела цикла**, то есть тех операторов, которые выполняются несколько раз,
- **начальных установок**,
- **блока модификации параметра цикла**,
- **проверки условия выхода из цикла**, которое может размещаться:
  - ✔ либо до тела цикла (тогда говорят о цикле с **предусловием**),
  - ✔ либо после тела цикла (цикл с **постусловием**).



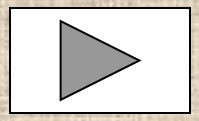
Управляющие операторы. Циклы

Один проход цикла называется **итерацией**.

Переменные, принудительно изменяющиеся в цикле и использующиеся при проверке условия выхода из него, называются **параметрами цикла**.

Замечание:

- Нельзя передавать управление извне внутрь цикла.
- Выход из цикла возможен как при выполнении условия выхода, так и по специальным операторам передачи управления.



Управляющие операторы. Циклы

- цикл с предусловием
- цикл с постусловием
- цикл с параметром

Линейная программа

Задача: Идет **k**-я секунда суток. Определить, сколько целых часов (**h**) и целых минут (**m**) прошло с начала суток.

Сколько секунд в минуте? *1 мин. = 60 сек.*

Сколько минут в часе? *1 час = 60 мин.*

Сколько секунд в часе? *1 час = 3600 сек.*

Как вычислить число полных часов? **k=18125**

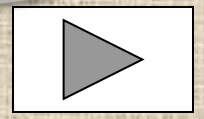
h=k/3600 *целочисленное деление (5 ч)*

Как вычислить число полных минут?

m=(k%3600)/60

С чего начать программу?

- объявить переменные *int h,m; long k;*
- ввести значения секунд k *scanf ("%ld", &k);*
- вычислить *оператор присваивания*
- вывести результат *printf*



```
#include <stdio.h>
#include <stdlib.h>
int main( )
{long k;    int h,m;
printf("Vvedite vremya v secyndah:");
scanf("%ld", &k);
h=k/3600;          m=(k%3600)/60;
printf("This %d clock %d minyt.\n", h, m);
system("Pause");
}
```

```
D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям\
Vvedite vremya v secyndah:9564
This 2 clock 39 minyt.
Для продолжения нажмите любую клавишу . . .
```

Операция условие

– единственная операция имеющая три операнда.

Формат операции:

выражение1 ? выражение2 : выражение3

Операция реализует алгоритмическую структуру ветвления.

Действие: Сначала вычисляется значение выражения1, которое, как правило, представляет собой некоторое условие.

Если оно истинно, т.е. **≠0**, то вычисляется выражение2 и полученный результат становится результатом операции. В противном случае в качестве результата берется значение выражения3.

Операция условие

Примеры:

1. Вычисление абсолютной величины переменной X можно организовать:

Y=X<0 ? –X : X;

2. Выбор большего значения из двух переменных ***a*** и ***b***:

max=(a<=b) ? b : a;

3. Заменить большее значение из двух переменных ***a*** и ***b*** на **1**:

(a>b) ? a : b =1;

Си/С++ позволяет ставить условную операцию слева от знака операции присваивания.

Условный оператор

Формат условного оператора:

- **полная** форма

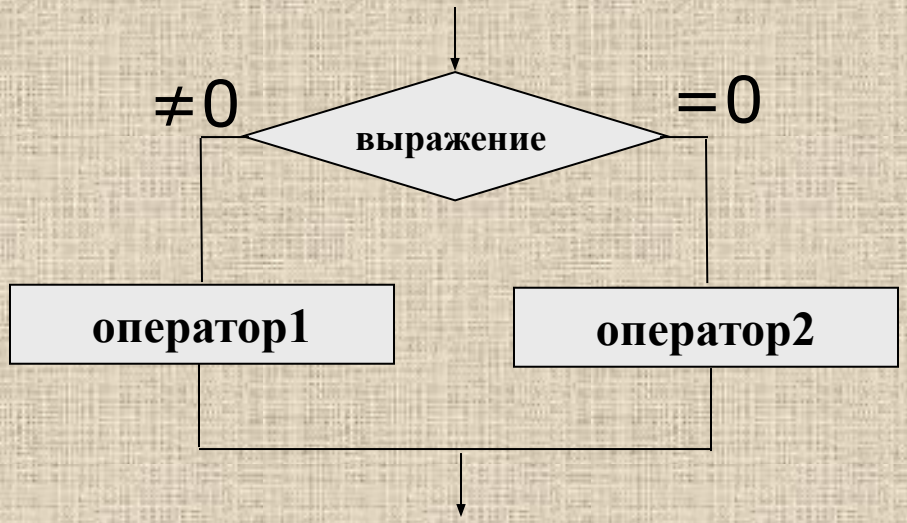
в скобках

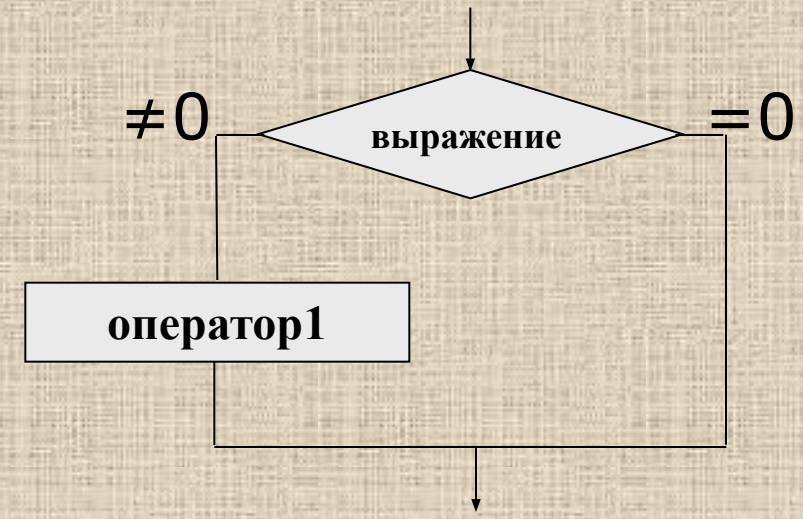
; обязательна

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

- **неполная** форма

if (***выражение***) оператор1;





Условный оператор

Выбор большего значения из двух переменных ***a*** и ***b***:

- операция условие:

max=(a>b) ? a : b;

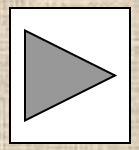
- **полная** форма условного оператора:

if (a>b) max=a**; else max**=b;

- **неполная** форма условного оператора:

max=b;

if (a>b) max=a;



<Выражения>: некоторые примеры

```
if ( a + 7 ) оператор1; else оператор2;
```

```
if ( a > 7 && a < 3 )оператор1;else оператор2;
```

```
if (    ( a >  3 && a <  7 )
     || ( a >  9 && a < 11 )
     || ( a > 15 && a < 17 )  ) . . .
```

```
if ( a = b/7 ) оператор1; else оператор2;
```

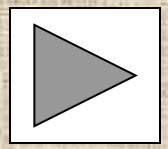
```
if ( a == b/7 ) оператор1; else оператор2;
```

Условный оператор

Пример:

Вычислить значение функции знак ***x:***

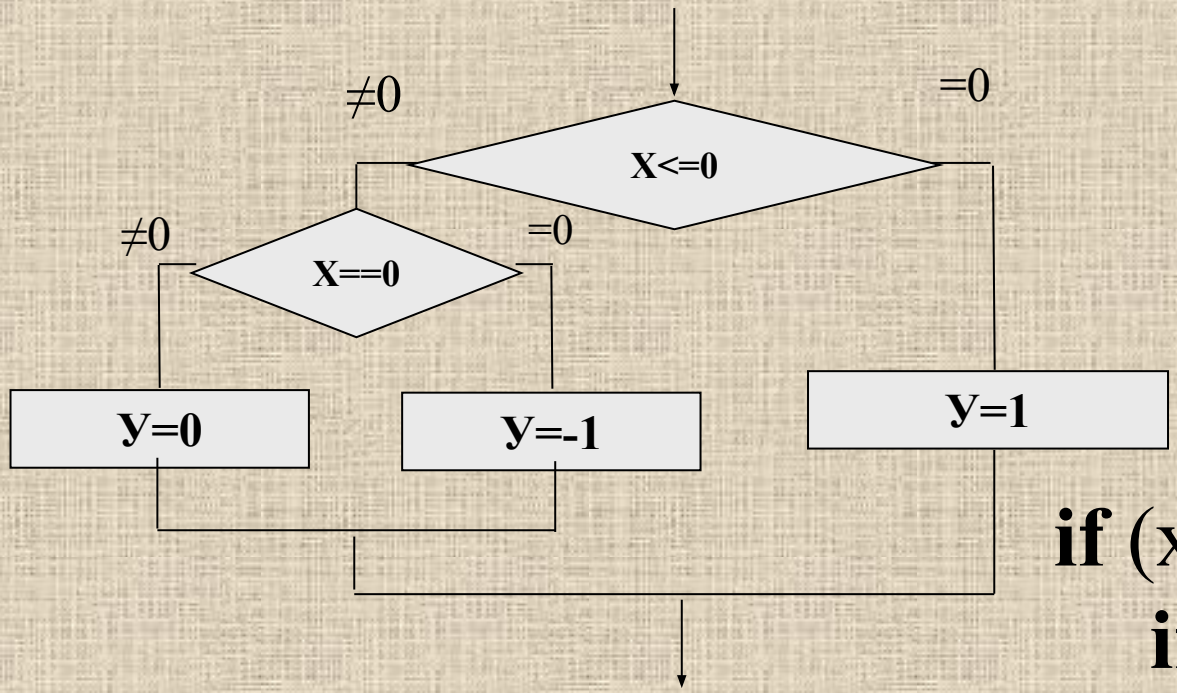
$$y=sign(x)=\begin{cases} -1, & \text{если } x < 0, \\ 0, & \text{если } x = 0, \\ 1, & \text{если } x > 0. \end{cases}$$



Условный оператор

1)

С полным вложенным ветвлением:

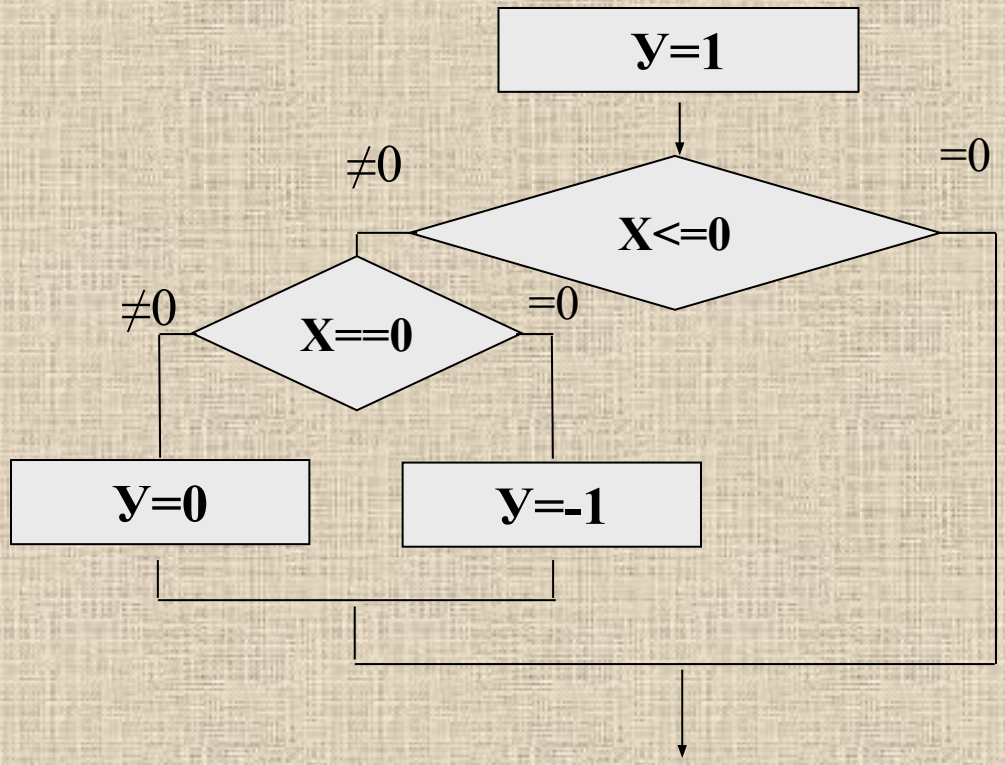



```
if (x<=0)
    if (x==0) y=0;
    else y=-1;
else y=1;
```

Условный оператор

2)

С неполным ветвлением:



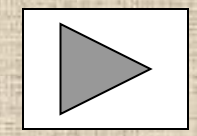

```
y=1;
if (x<=0)
    if (x==0) y=0;
    else y=-1;
```

Составной оператор

Для создания сложных управляющих композиций иногда **последовательность операторов** необходимо указывать **как один оператор**.

Для этой цели служит **составной оператор**. Синтаксически составной оператор в C++ - последовательность операторов взятая в фигурные скобки

```
{ оператор1;
  оператор2;
  …
  операторN;
}
```



Составной оператор

```
#include <iostream.h>
#include <stdlib.h>
int main()
  {    const float fac = 2.54;
       float x, in, cm;
       char ch = 0;
       cout << "Vvedite dliny: "; 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";
  system ("Pause");
}
```

```
D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям\dyu
Vvedite dliny: 35 м
0 in=0 cm
Для продолжения нажмите любую клавишу . . .
```

Составной оператор

Составной оператор иногда необходимо использовать для нарушения порядка выполнения операторов.

Совет: Чтобы сделать программу более читабельной, рекомендуется группировать операторы и конструкции во вложенных операторах if, используя фигурные скобки.

```
int main( )
 {   int t=2, b=7, r=3;
     if (t>b)
        {   if (b<r) r=b;   }
      else  r=t;
      cout << " r="<< r <<endl;
      system ("Pause");
}
```

Получим r=2

Составной оператор

Пример:

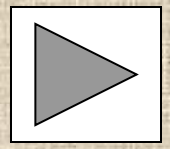
```
int main( )
 {   int t=2, b=7, r=3;
    if (t>b)
       {   if (b<r) r=b;  }
     else  r=t;
     cout << " r="<< r <<endl;
     system ("Pause");
}
```

r=2

Если фигурные скобки опущены, то компилятор связывает каждое ключевое слово *else* с наиболее близким *if*, для которого нет *else*.

```
int main( )
 {   int t=2, b=7, r=3;
    if (t>b)
        if (b<r) r=b;
        else  r=t;
     cout << " r="<< r <<endl;
     system ("Pause");
}
```

Получим r=3

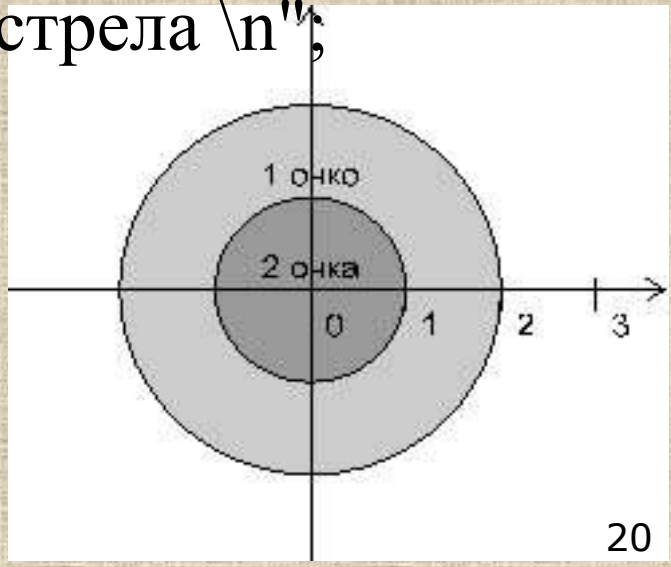


Условный оператор

Пример:

Производится выстрел по мишени, изображенной на рисунке. Определить количество очков.

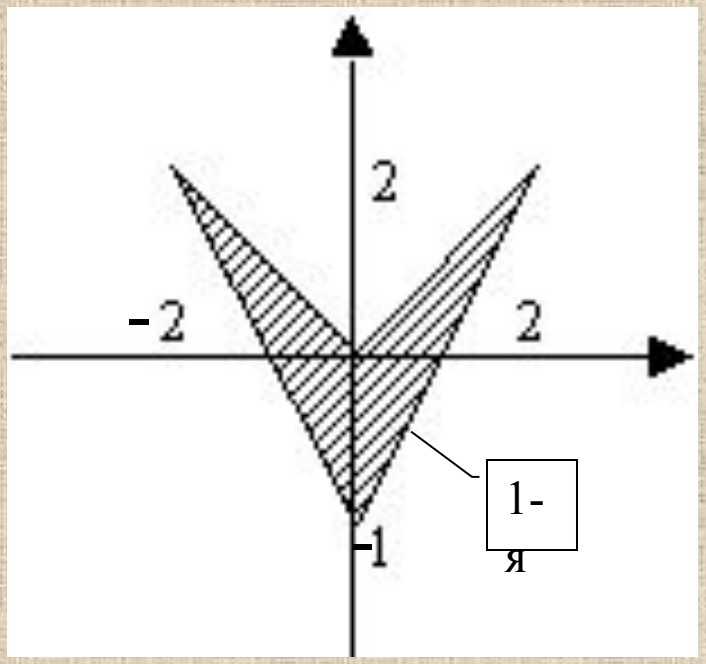
```
#include <iostream>
using namespace std;
int main()
{ float x, y;  int kol;
  cout << "Введите координаты выстрела \n";
  cin >> x >> y;
  if ( x*x + y*y < 1 ) kol = 2;
  else if ( x*x + y*y < 4 ) kol = 1;
    else kol = 0;
  cout << "Очков: " << kol;
```



Условный оператор

Составить условное выражение для:

Задача 2. *Записать логическое выражение, принимающее значение **1**, если точка лежит внутри заштрихованной области, иначе **0**.*



Уравнение прямой проходящей через две точки

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

для 1-ой прямой:

точка (x_1,y_1) координаты (0,-1)

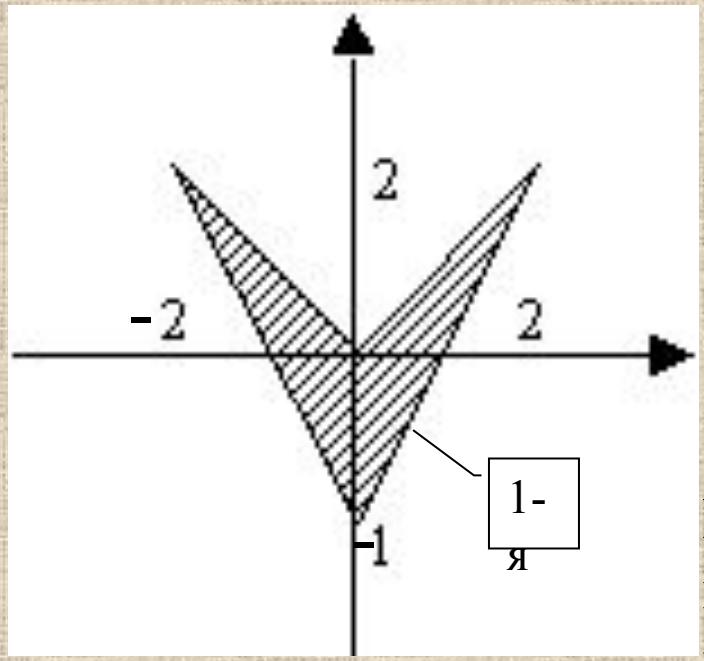
точка (x_2,y_2) координаты (2,2)

$$\frac{x - 0}{2 - 0} = \frac{y - (-1)}{2 - (-1)}$$

Условный оператор

для 1-ой прямой получим:
точка (x_1,y_1) координаты (0,-1)
точка (x_2,y_2) координаты (2,2)

$$\frac{x}{2}=\frac{y+1}{3}$$



$$\frac{x}{2}=\frac{y+1}{3} \Rightarrow y=\frac{3\cdot x}{2}-1$$

Полуплоскость

$$y >= \frac{3\cdot x}{2}-1$$

Проверим:
при (1,2) получим 2>=(3*1.)/2-1=0.5 Истина
при (2, -1) получим -1>=(3*2.)/2-1=2 Ложь

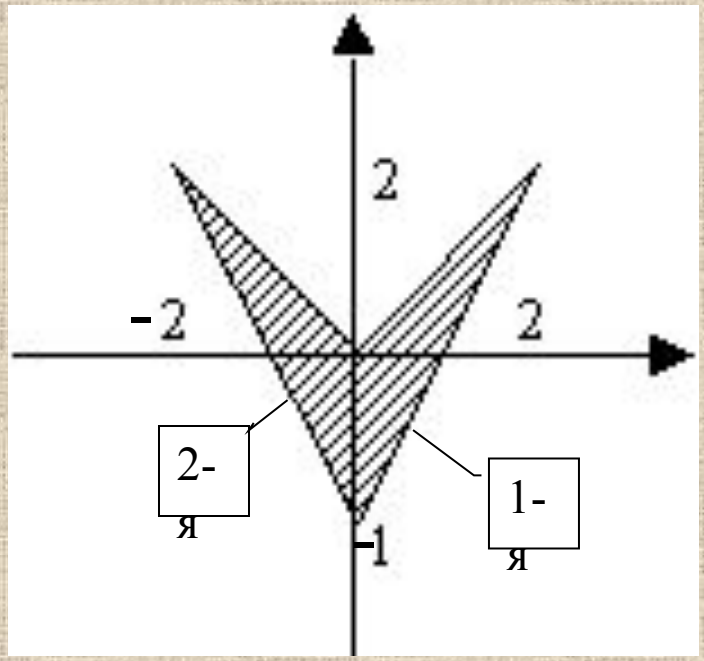
$$y=y_1+\frac{y_2-y_1}{x_2-x_1}\cdot(x-x_1)$$

Условный оператор

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1} \cdot (x - x_1)$$

для 2-ой прямой получим:
точка (x_1,y_1) координаты (0,-1)
точка (x_2,y_2) координаты (-2,2)

$$y = -\frac{3 \cdot x}{2} - 1$$



$$y >= -\frac{3 \cdot x}{2} - 1$$

Проверим:
при (-1,2) : 2>=-(3*-1.)/2-1=0.5 Истина
при (-2, -1) :-1>=-(3*-2.)/2-1=2 Ложь

аналогично составить для остальных прямых.
Пересечение полуплоскостей – конъюнкция
(y>=3*x/2-1 && y>=-3*x/2-1 && ??? && ???)

Выражения …

```c
#include <stdio.h>
#include <stdlib.h>
int main( )
{int x;
printf("Vvedite x:  ");
scanf("%d", &x);
if (x) printf("True\n"); else printf("False\n");
system("Pause");
}
```

```
D:\ковчег\Алутина\Программирование\К занятиям\Задания к зан
Vvedite x:  -7
True
Для продолжения нажмите любую клавишу . . .
```

Выражения …

```
#include <stdio.h>
#include <stdlib.h>
int main( )
{int x;
printf("Vvedite x:  ");
scanf("%d", &x);
if (x=1) printf("True\n"); else printf("False\n");
system("Pause");
}
```

```
D:\ковчег\Алутина\Программирование\К занятиям\Задания к заня
Vvedite x:  0
True
Для продолжения нажмите любую клавишу . . .
```

Выражения …

Пример:

if (x= (x==y))printf("True\n");else printf("False\n");

```
#include <stdio.h>
#include <stdlib.h>
int main( )
{int x, y;
printf("Vvedite x and y:  ");
scanf("%d%d", &x, &y);
if (x= (x==y))printf("True\n");else printf("False\n");
system("Pause");
}
```

```
D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям\chasyi.e
Vvedite x and y:  0 0
True
Для продолжения нажмите любую клавишу . . .
```

Выражения …

Примеры:

Консольный ввод/вывод

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
int main( )
{char ch;
printf("Vvedite simbol :  ");
if (ch = getch()== 'g')printf("True %d \n", ch);
else printf("False %d \n"
system("Pause");
}
```

Нажата клавиша ‘g’

```
D:\ковчег\Алутина\Программ...вание\К занятиям\Задания к зан
Vvedite simbol :  True 1
Для продолжения нажмите любую клавишу . . .
```

Оператор выбора

В [ ] необязательная часть

switch (*выражение*){

case константное_выражение_1:
[список_операторов_1]

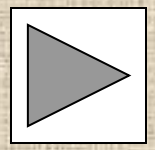
case константное_выражение_2:
[список_операторов_2]

...

case константное_выражение_n:
[список_операторов_n]

[default: операторы **]**

}



Оператор выбора

```
switch ( выражение ){
    case константное_выражение_1:
    [список_операторов_1]
    case константное_выражение_2:
    [список_операторов_2]
    ...
    case константное_выражение_n:
    [список_операторов_n]
    [default: операторы ]
}
```

Действие оператора:

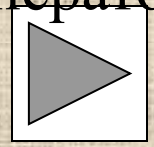
1. Вычисляется выражение в скобках после **switch.**

2.Полученное значение последовательно сравнивается с константами после слова **case**, при первом совпадении значений выполняются все нижестоящие операторы после **:**

3.Если ни с одной из констант совпадения не произошло, то выполнятся операторы после слова **default**.

Замечание: **1.** Строка **default** может отсутствовать.

2. Чтобы обойти выполнение операторов на последующих ветвях необходимо принять специальные меры, используя оператор выхода (**break**).

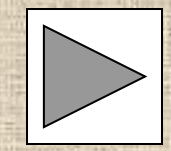


Оператор выбора

Вычислить значение выражения **a+b** или **a-b** или **a*b** или **a/b**

…

```
int main()
{  int a, b, res;      char op;
   cout << "\nВведите 1й операнд : ";        cin >> a;
   cout << "\nВведите знак операции : ";   cin >> op;
   cout << "\nВведите 2й операнд : ";        cin >> b;
   bool f = true;
   switch (op)
     { case '+': res = a + b; break;
       case '-': res = a - b; break;
       case '*': res = a * b; break;
       case '/': res = a / b; break;
       default : cout <<"\n Неизвестная операция";   f = false;
      }
  if (f) cout << "\nРезультат : " << res;
}
```



Оператор выбора

```
#include <iostream.h>
#include <stdlib.h>
int main()
{   int a, b, res;    char op;
    cout << "\nVvedite 1 operand : ";          cin >> a;
    cout << "\nVvedite operaziu : ";    cin >> op;
    cout << "\nVvedite 2 operand : ";          cin >> b;
    bool f = true;
    switch (op)
      { case '+': res = a + b; break;
        case '-': res = a - b; break;
        case '*': res = a * b;
        case '/': res = a / b;  break;
        default : cout <<"\n Неизвестная операция";    f = false;
      }
   if (f) cout << "\nRezultat : " << res<<endl;
   system("Pause");
}
```

Убрали break

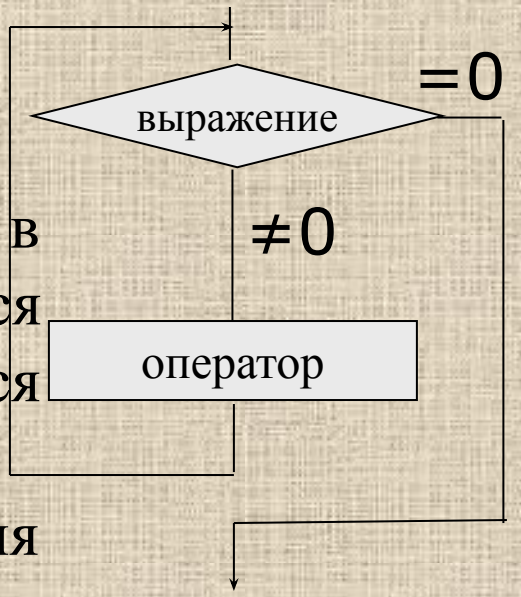
```
D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям\switch.exe

Vvedite 1 operand : 7

Vvedite operaziu : *

Vvedite 2 operand : 5

Rezultat : 1
Для продолжения нажмите любую клавишу . . .
```

```
#include <iostream.h>
#include <stdlib.h>
#include <iomanip.h>
int main()
{  float x, res=1; int n;   bool f = true;
    cout << "\nVvedite x: ";          cin >> x;
    cout << "\nVvedite n<= 10: ";    cin >> n;
    switch (n)
      { case 10: res = res*x;
        case 9: res = res*x;
        case 8: res = res*x;
        case 7: res = res*x;
        case 6: res = res*x;
        case 5: res = res*x;
        case 4: res = res*x;
        case 3: res = res*x;
        case 2: res = res*x;
        case 1: res = res*x;break;
        default : cout <<"\n False n";   f = false;
}
   if (f) cout << "\nРезультат : " << setw(15)<<setprecision(12)<<res<<endl;
 system("Pause");
 }
```

D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям\switch1.exe

```
Vvedite x: 9

Vvedite n<= 10: 9

Шхчеы№Ср€ :       387420480
Для продолжения нажмите любую клавишу . . .
```

Цикл с предусловием



Общий формат цикла с предусловием:

while (выражение) оператор;

Сначала вычисляется значение выражения в скобках, если оно истинно (не 0), то выполняется оператор тела цикла, затем снова вычисляется выражение и все повторяется.

Если выражение ложно (значение выражения равно 0) цикл заканчивает работу, управление передается следующему после цикла оператору.

Вместо оператора – тела цикла можно использовать составной оператор, тогда формат имеет вид:

```
while (выражение)
{ оператор1;
}
```

Тело цикла может ни разу не выполниться, если выражение сразу принимает значение 0.

Цикл с предусловием

```
#include <iostream.h>
#include <stdlib.h>
int main()
{ long int F=1;   int i=1, N;
    cout << "Vvedite
```

Проверить границы для типа unsigned long

Последнее правильное

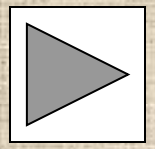
```
Vvedite N
12

12!=479001600
Для продолжения нажмите любую клавишу . . .
```

```
 system("Pause");
}
```

D:\ковчег\Алутина\Программирование\К занятиям\Задания к

Почему?

```
Vvedite N
17

17!=-288522240
Для продолжения нажмите любую клавишу . . .
```



Цикл с предусловием

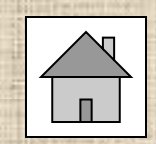
Тот же пример. Составной оператор тела цикла можно записать одним оператором присваивания:

```
#include <iostream.h>
#include <stdlib.h>
int main()
{ long int F=1;  int i=1, N;
  cout << "Vvedite N\n";
  cin >> N ;
  while ( i<=N )
  F=F*i++;
  cout << "\n"<<N<< "!="<<F<< endl;
  system("Pause");
}
```

Более лаконично:
F*=i++;

Эквивалентно двум операторам: F=F*i; i=i+1;

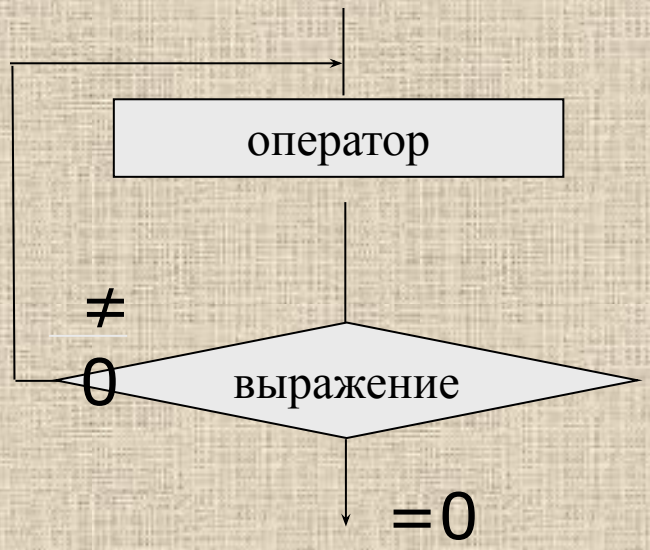
Предлагаем еще вариант
while (i++<=N)
F*=i;



Цикл с постусловием

do оператор **while** (выражение);

Сначала выполняется простой или составной оператор – тело цикла, а затем вычисляется выражение. Если значение выражения не равно 0 (истинно), тело цикла выполняется еще раз, и так далее, пока значение выражения не станет равным нулю или в теле цикла не будет выполнен какой-либо оператор передачи управления. Тип выражения должен быть арифметическим или приводимым к нему.



Тело цикла с постусловием хотя бы один раз выполняется.

Цикл с постусловием

Вводить значение переменной *answer* и печатать текст пока ***answer*** не равно ***y***. Использовать библиотеку вывода кириллицы.

Для работы функции setlocale необходимо подключение заголовочного файла #include <clocale>

Вызов функции setlocale устанавливает *русскую локаль,* после чего русские константы выводятся в консольное окно по-русски при использовании и printf, и cout.

```
        cin >> answer;
    }while (answer != 'y');
}
```

D:\ковчег\Алутина\Программирование\К занятиям\

```
Купи слона! n
Купи слона! f
Купи слона! d
Купи слона! a
Купи слона!
```

Пример - вычисление суммы ряда с заданной точностью

$$1 - 1/x + 1/x^2 - 1/x^3 + 1/x^4 - \ldots$$

Суммой ряда называется предел (***lim)*** к которому стремится последовательность частичных сумм ряда, если такой ***lim*** существует.

Известно, что знакопеременный ряд сходится, если $|r_n| > |r_{n+1}|$, где r_n - ***n***-й член ряда.

Доказано, что $|s-s_n| <= |r_{n+1}| < \varepsilon$

s – сумма ряда

s_n – сумма ***n*** первых членов ряда

ε – точность

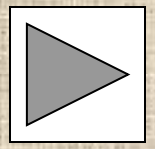
```
R= -R/x;
```

Найти значения суммы ряда для **ε** = 0,1; 0,01; 0,001; 0,0001 …

Оформить вывод значения суммы ряда до 9-12 знаков после десятичной точки.

В общем случае вид ***n***-го члена ряда задан в постановке задачи.

Проверить ряд на сходимость.



Пример - Вычисление суммы ряда

$$1 - 1/x + 1/x^2 - 1/x^3 + 1/x^4 - \ldots$$

```
#include <iostream.h>
#include <math.h>
int main(){
   double x, S, R, eps;
   cout << "\nВведите аргумент x и точность eps: ";
   cin >> x >> eps;
   if ( fabs(x)>1 )
      {  S=R=1;
      while ( fabs(R)> eps )
         { R= -R/x;
           S += R;
         }
      cout << "\nСумма ряда= " << S;
      }

   else  cout << "\nРяд расходится!";
}
```

Оператор цикла с п

Общий формат цикла:

for (выражение_1 ; выражение_2 ; выражение_3)
оператор;

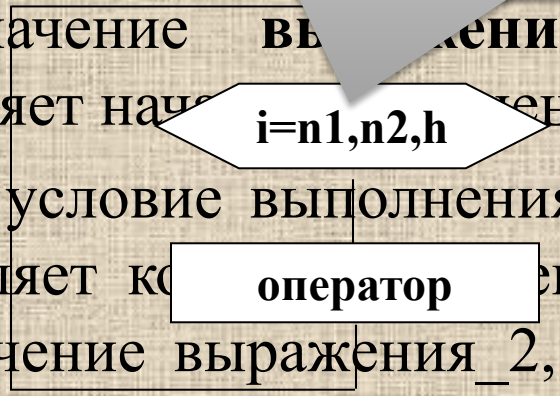
Заголовок цикла с параметром

n1 – начальное значение, n2 – конечное значение параметра цикла, h – шаг изменения пар

в заголовке цикла ; не ставится

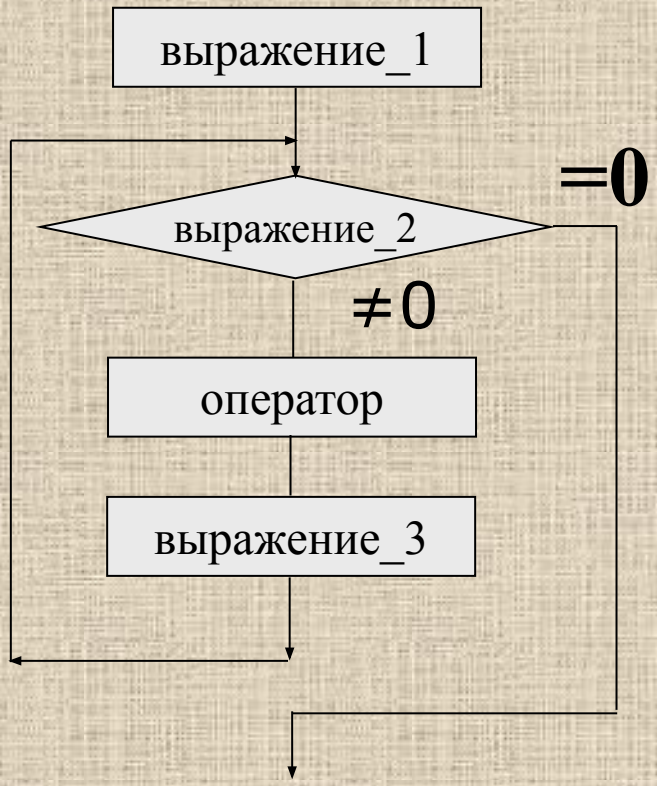
Действие:

1. Вычисляется значение **выражения_1**, которое (как правило) определяет нач ение параметра цикла.
2. **Выражение_2** – условие выполнения цикла, которое (как правило) определяет ко ение параметра цикла. Вычисляется значение выражения_2, если не 0 (истинно), то выполняется оператор – тело цикла. Если значение выражения_2 равно 0 (ложно), цикл заканчивает работу.
3. **Выражение_3** (как правило) определяет изменение параметра цикла.
4. Перейти на шаг 2 – вычисление выражения_2, т.е. прове условия.



Оператор цикла с параметром (счетный цикл)

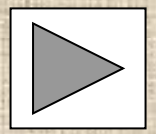
for (выражение_1; выражение_2; выражение_3)
оператор;



Оператор цикла с параметром (счетный цикл)

```
#include <iostream.h>
int main()
{
   int i; double a=2;
   for (i=1; i<5; i++)
  cout << a*i <<"\n";
}
```

i	i<5	Экран
1	(1<5) +	2
2	(2<5) +	4
3	(3<5) +	6
4	(4<5) +	8
5	(5<5) - цикл заканчивает работу	

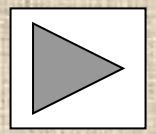


Оператор цикла с параметром (счетный цикл)

```
for (int i = 1, s = 0; i<=100; i++)
      s += i;
```

Что вычисляется в цикле?

i=1	**s=0**	**(1<=100) +**
2	1	(2<=100) +
3	3	(3<=100) +
4	6	(4<=100) +
5	10	(5<=100) +
…	…	+
100	Сумма чисел от 1 до 100	(100<=100) +
101		(101<=100) -



Оператор цикла с параметром (счетный цикл)

1. Если условие (т.е. выражение_2) изначально ложно

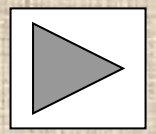
```
for ( i = 1; i<1 ; i++)
    оператор;
```

цикл не выполнится ни разу.

2. Если условие (т.е. выражение_2) всегда истинно

```
for ( i = 1; i>0 ; i++)
    оператор;
```

цикл будет повторятся бесконечно (зацикливание).



Оператор цикла с параметром (счетный цикл)

3. В операторе цикла с параметром может отсутствовать одно из выражений ; должны

инициализация i в цикле не требуется

i инициализируется до цикла

выраж 1, то не выполняется инициализация
парамет икла. В э м случае инициализация параметра должна
быть выпо ена до заголовка цикла с параметром.

```
{   in i=1; double a=2;
   for (   ;  i<5 ;  i++)
      cout<<a*i;
}
```

б) Если выражение_3 отсутствует в заголовке цикла, тогда изменение параметра цикла должно быть выполнено в теле цикла.

При этом следует избегать изменение параметра цикла одновременно и в выражении_3, и в теле цикла, так как это может привести к трудно контролируемым ошибкам.

Оператор цикла с параметром (счетный цикл)

в) если отсутствует выражение_2, то есть отсутствует условие проверки окончания цикла, в этом случае результат проверки всегда – истина, получим бесконечный цикл.

выражение_2 отсутствует

```
{double a=2;
   for (int i=1;  ; i++)
      cout<<a*i <<endl;
}
```

г) если все три выражения отсутствуют.

бесконечный цикл

```
for (  ;   ;  )
      оператор;
```

Оператор цикла с параметром (счетный цикл)

Используя операцию «запятая» в выражениях заголовка цикла с параметром можно задавать несколько операторов.

Например, в выражении_1 инициализировать значение нескольких переменных перед вычисление факториала).

Порядок операторов не важен

```
for (F=1, i=1; i<=N ; i++)
    F=F*i;
```

Порядок операторов важен!

в выражение_3 может быть [illegible] операт

; обязательна

```
for (F=1, i=1; i<=N ; F=F*i, i++) ;
```

Оператор тела цикла формально отсутствует, так как внесен в выражение_3 заголовка

```
for (F=1, i=0; i<=N ; i++, F=F*i) ;
```

В данном варианте необходимо изменить начальное значение параметра цикла

Оператор цикла с параметром (счетный цикл)

В качестве третьего выражения можно использовать любое правильно составленное выражение. Какое бы выражение мы ни указали, его значение будет меняться при каждой *итерации*:

```
for (x=y=1; y<=75; y=5*x++)
    printf("%10d %10d\n",x,y);
```

Обратите внимание, что в спецификации цикла проверяется значение ***y***, а не ***x***. В каждом из трех выражений, управляющих работой цикла for, могут использоваться любые переменные.

! Хотя ***этот пример*** и правильный, он ***не может служить иллюстрацией хорошего стиля программирования.*** Программа выглядела бы гораздо понятнее, если бы мы не смешали процесс изменения переменной цикла с алгебраическими вычислениями.

Цикл с параметром

Протабулировать значение функции y=x^2+1;

```
#include <stdio.h>
```

```
#include <stdio.h>
#include <stdlib.h>

int main(){
    float X1, Xk, Dx;
    printf("Vvedite diapazon and step izmeneniy argymenta: ");
    scanf("%f%f%f", &X1, &Xk, &Dx);
    printf("|    X    | Y=x^2+1 |\n");
    for (float X = X1;  X <= Xk; X +=Dx )
        printf("|  %5.2f  |  %5.2f  |\n", X, X*X + 1);
    printf("|  %5.2f  |  %5.2f  |\n", X, X*X + 1);
    system("Pause");
}
```

Компилятор | Ресурсы | Журнал компиляции | Отладка | Результаты поиска | Закрыть

Строка	Файл	Сообщение
	D:\ковчег\Алутина\Программиров...	In function `int main()':
11	D:\ковчег\Алутина\Программиров...	name lookup of `X' changed for new ISO `for' scoping
9	D:\ковчег\Алутина\Программиров...	using obsolete binding at `X'

```
|   2.50  |   7.25  |
|   3.00  |  10.00  |
|   3.50  |  13.25  |
|   4.00  |  17.00  |
|   4.50  |  21.25  |
|   5.00  |  26.00  |
Для продолжения нажмите любую клавишу . . .
```

Оператор цикла с параметром (счетный цикл)

```
#include <stdio.h>
#include <clocale>
#include <stdlib.h>
int main()
{  setlocale( LC_ALL, "rus" );
   for ( int i = 1; i < 6; ++i )
       printf( " %d ", i );
   printf( "вышел зайчик погулять \n
   system("Pause");
}
```

```
D:\ковчег\Алутина\Программирование\К зан
 1  2  3  4  5 вышел зайчик погуля
Для продолжения нажмите любую клав
```

```
#include <stdio.h>
#include <clocale>
#include <stdlib.h>
int main()
{  setlocale( LC_ALL, "rus" );
   for ( int i = 1; i < 6; ++i ) {
       printf( " %d ", i );
   printf( "вышел зайчик погулять \n" );  }
   system("Pause");
}
```

```
D:\ковчег\Алутина\Программирование\К занятиям\Задан
 1 вышел зайчик погулять
 2 вышел зайчик погулять
 3 вышел зайчик погулять
 4 вышел зайчик погулять
 5 вышел зайчик погулять
Для продолжения нажмите любую клавишу . . .
```

```
{  setlocale( LC_ALL, "rus" );
   for ( int i = 1; i < 6; ++i ) {
       printf( " %d ", i );
   printf( "вышел зайчик погулять \n" );  }
   system("Pause");
}
```

Оператор цикла с параметром (счетный цикл)

```
#include <stdio.h>
#include <clocale>
#include <stdlib.h>
int main()
{   int num=1;
    setlocale( LC_ALL, "rus" );
    for ( printf("Вводите числа! ");  num != 5;  )
          scanf("%d", &num);
    printf("Это как раз то, что я хочу! \n");
 system("Pause");
}
```

```
D:\ковчег\Алутина\Программирование\К занятиям\Задания к зан
Вводите числа! 5
Это как раз то, что я хочу!
Для продолжения нажмите любую клавишу . . .
```

Что будет на экране?

Оператор цикла с параметром (счетный цикл)

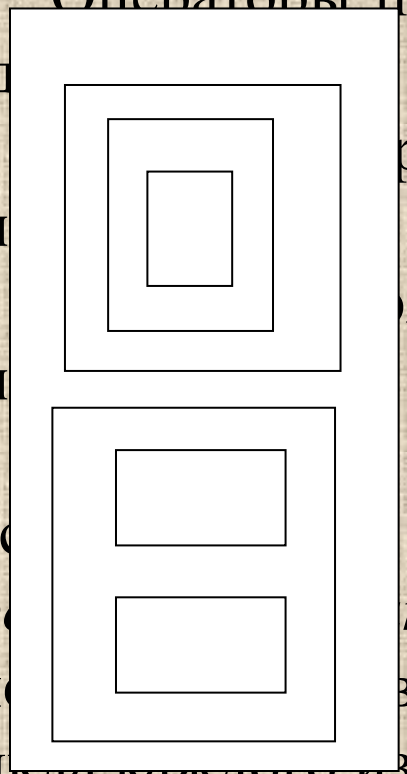
```
#include <stdio.h>
#include <clocale>
#include <stdlib.h>
int main()
{   char ch;
    setlocale( LC_ALL, "rus" );
    for (ch='я';  ch>='a'; ch-- )
       printf("ASCII код символа

 system("Pause");
}
```

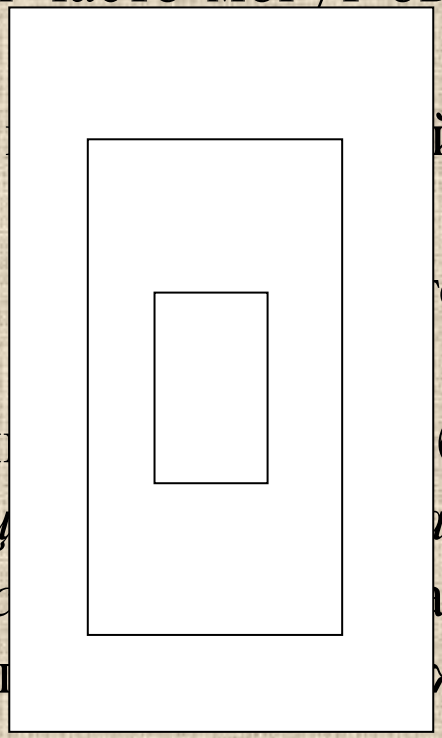
```
ASCII код символа я  равен  -1
ASCII код символа ю  равен  -2
ASCII код символа э  равен  -3
ASCII код символа ь  равен  -4
ASCII код символа ы  равен  -5
ASCII код символа ъ  равен  -6
ASCII код символа щ  равен  -7
ASCII код символа ш  равен  -8
ASCII код символа ч  равен  -9
ASCII код символа ц  равен  -10
ASCII код символа х  равен  -11
ASCII код символа ф  равен  -12
ASCII код символа у  равен  -13
ASCII код символа т  равен  -14
ASCII код символа с  равен  -15
ASCII код символа р  равен  -16
ASCII код символа п  равен  -17
ASCII код символа о  равен  -18
ASCII код символа н  равен  -19
ASCII код символа м  равен  -20
ASCII код символа л  равен  -21
ASCII код символа к  равен  -22
ASCII код символа й  равен  -23
ASCII код символа и  равен  -24
ASCII код символа з  равен  -25
ASCII код символа ж  равен  -26
ASCII код символа е  равен  -27
ASCII код символа д  равен  -28
ASCII код символа г  равен  -29
ASCII код символа в  равен  -30
ASCII код символа б  равен  -31
ASCII код символа а  равен  -32
Для продолжения нажмите любую клавишу .
```

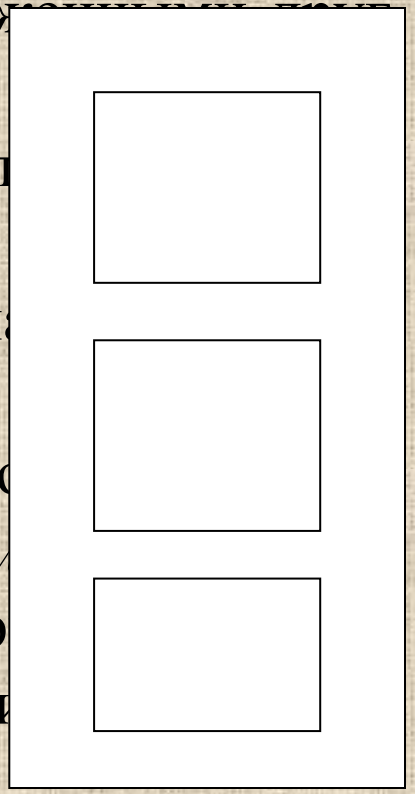
```
ASCII код символа х  равен  -11
ASCII код символа ц  равен  -10
ASCII код символа ч  равен  -9
ASCII код символа ш  равен  -8
ASCII код символа щ  равен  -7
ASCII код символа ъ  равен  -6
ASCII код символа ы  равен  -5
ASCII код символа ь  равен  -4
ASCII код символа э  равен  -3
ASCII код символа ю  равен  -2
ASCII код символа я  равен  -1
Для продолжения нажмите любую клавишу
```

Вложенные циклы

Операторы повторения часто могут быть вло
в д
ржащий й цикл
вн
жащийся го цикла
вн
и внешни быть лю
рас видов: *ц иметром*
пр клом с пос авила ор
вн внутреннег же, как и
цикла каждого из этих видов.







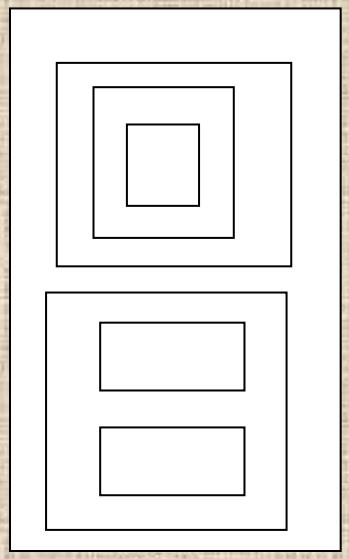
При построении вложенных циклов необходимо соблюдать условие: ***все операторы внутреннего цикла должны полностью лежать в теле внешнего цикла.***

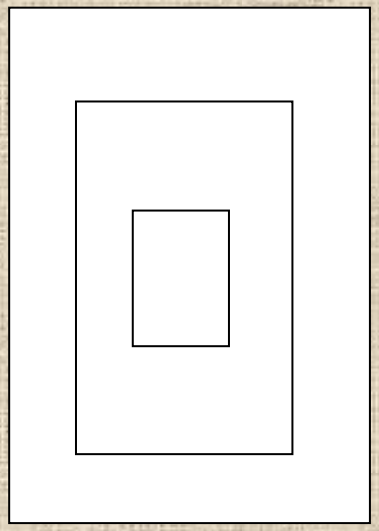
Некоторые структуры вложенных циклов представлены на рисунке.

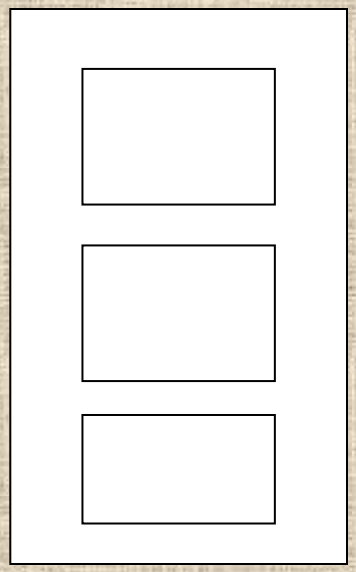
Вложенные циклы

Параметры циклов разных уровней не изменяются одновременно. Сначала все свои значения изменит параметр цикла низшего уровня вложенности при фиксированных значениях параметров циклов с высшим уровнем.

Затем изменяется на один шаг значение параметра цикла следующего уровня, и снова полностью выполняется самый внутренний цикл, и т.д. до тех пор, пока параметры циклов не примут все требуемые значения.







```
#include <iostream.h>
#include <stdlib.h>
int main()
{   int i, j, k;
    for (i = 1; i < 10; i++)      // Внешний цикл
    { cout << "\n";
       for (j = 1; j<10; j++) // Внутренний цикл
            {  k=i*j;
                if (k<10) cout<<"  "; // два пробела
        else cout<<" "; // один пробел
               cout<<k;
            }
    }
  cout<<endl<<endl;
  system("Pause");
    }
```

D:\ковчег\Алутина\Программирование\К занятиям\Задания к зан

```
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32 36
 5 10 15 20 25 30 35 40 45
 6 12 18 24 30 36 42 48 54
 7 14 21 28 35 42 49 56 63
 8 16 24 32 40 48 56 64 72
 9 18 27 36 45 54 63 72 81

Для продолжения нажмите любую клавишу . . .
```

чная таблица		
	j<10	k=i*j
	+	1
	+	2
	…	…
)	-	
	+	2
	+	4
	…	…
	+	18
	-	
	+	3
	…	…
	+	27
)	-	

Операторы передачи управления

В C++ есть пять операторов, изменяющих естественный порядок выполнения вычислений:

- оператор выхода из цикла и переключателя **break**;
- оператор перехода к следующей итерации цикла **continue**;
- оператор возврата из функции **return**;
- оператор безусловного перехода **goto**;

Оператор возврата из функции **return** завершает выполнение функции и передает управление в точку ее вызова. рассмотрим его вместе с функциями позже.

Исключительную ситуацию (или просто ***исключение***) генерирует либо программист с помощью оператора **throw**, либо сама среда выполнения.

Это происходит, когда во время выполнения программы возникают ошибки, например, деление на ноль или переполнение. Механизм обработки исключений, реализованный в C++, позволяет реагировать на подобные ошибки и так избегать аварийного завершения.

```
#include <iostream.h>
#include <stdlib.h>
int main()
{   int i, j, k;
    for (i = 1; i < 10; i++)      // Внешний цикл
    { cout << "\n";
       for (j = 1; j<10; j++) // Внутренний цикл
            {  k=i*j;
               if (k>35) break;
                if (k<10) cout<<"  "; // два пробела
                else cout<<" "; // один пробел
               cout<<k;
            }
    }
  cout<<endl<<endl;
  system("Pause");
    }
```

D:\ковчег\Алутина\Программирование\К занятиям\Задания к зан

```
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32
 5 10 15 20 25 30 35
 6 12 18 24 30
 7 14 21 28 35
 8 16 24 32
 9 18 27

Для продолжения нажмите любую клавишу . . .
```

ная таблица

10	k=i*j	k>35
-	1	-
-	2	-
.	…	-
-	3	-
.	…	-
-	4	-
.	…	-
-	36	+
-	5	-
.	…	-
-	35	-
-	40	+
-	6	-
-	36	+
-	7	-

Решение

```
#include <iostream.h>
#include <stdlib.h>
int main()
{  int i, j, k=0;
    for (i = 1; i < 10; i++)     // Внешний цикл
    { cout << "\n";
       for (j = 1; j<10; j++) // Внутренний цикл
            {  if (k==i*i) continue;
               k=i*j;
               if (k<10) cout<<"  "; // два пробела
               else cout<<" "; // один пробел
               cout<<k;
            }
    }
  cout<<endl<<endl;
  system("Pause");
    }
```

```
D:\ковчег\Алутина\Программирование\К занятиям\Задания к заня

 1
 2  4
 3  6  9
 4  8 12 16
 5 10 15 20 25
 6 12 18 24 30 36
 7 14 21 28 35 42 49
 8 16 24 32 40 48 56 64
 9 18 27 36 45 54 63 72 81

Для продолжения нажмите любую клавишу . . .
```

Трассировочная таблица

$k==i^2$	k=i*j
0==1(-)	1
1==1(+)	
1==1(+)	
…	
1==4(-)	2
2==4(-)	4
4==4(+)	
…	
4==9(-)	3
3==9(-)	6
6==9(-)	9
9==9(+)	
…	
9==16(-)	4
4==16(-)	8

Оператор цикла с параметром

```
#include <stdlib.h>
#include <iostream.h>
int main()
{   int i;
    for (i = 1; i <= 100; i++)
     {  if ( i%2 )    continue;
        cout <<"\t"<< i ;
     }
    system ("PAUSE");
}
```

Что будет выведено на экран?

D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям\Павловская\Безымян...

```
        2       4       6       8       10      12      14      16      18
20      22      24      26      28      30      32      34      36      38
40      42      44      46      48      50      52      54      56      58
60      62      64      66      68      70      72      74      76      78
80      82      84      86      88      90      92      94      96      98
100Для продолжения нажмите любую клавишу . . .
```

Пример - Вычисление суммы ряда

$$\mathrm{sh}\ x = 1 + x^3/3! + x^5/5! + x^7/7! + \ldots$$

```
#include <iostream.h>
#include <math.h>
int main(){
   const int MaxIter = 500;
   double x, eps;
   cout << "\nВведите аргумент и точность: ";
   cin >> x >> eps;
   bool ok = true;
   double y = x, ch = x;
   for (int n = 0; fabs(ch) > eps; n++){
      ch *= x * x /(2 * n + 2)/(2 * n + 3);
      y += ch;
      if (n > MaxIter){ok = false; break;}
   }
   if (ok) cout << "\nЗначение функции: " << y;
   else    cout << "\nРяд расходится!";
}
```

Пример - вычисление квадратного корня

```
#include <stdio.h>
#include <math.h>
int main(){
   double X, Eps;
   double Yp, Y = 1;
   printf("Введите аргумент и точность: ");
   scanf("%lf%lf", &X, &Eps);
   do{
      Yp = Y;
      Y = (Yp + X/Yp)/2;
   }while (fabs(Y - Yp) >= Eps);
   printf("\n %lf %lf", X, Y);
}
```

schikl4.cpp

$$y_n = \frac{1}{2}(y_{n-1} + x/y_{n-1})$$