Технология программирования задач с циклами.

- ***Цикл*** – многократное повторение одного и того же участка программы
- ***Циклом*** называется блок кода, который для решения задачи требуется повторить несколько раз.

Каждый цикл состоит из

- блока проверки условия повторения цикла
- тела цикла

- Цикл выполняется до тех пор, пока блок проверки условия возвращает истинное значение.
  Тело цикла содержит последовательность операций, которая выполняется в случае истинного условия повторения цикла. После выполнения последней операции тела цикла снова выполняется операция проверки условия повторения цикла. Если это условие не выполняется, то будет выполнена операция, стоящая непосредственно после цикла в коде программы.

Виды циклов

1. Счётный оператор цикла
2. Оператор цикла с предусловием
3. Оператор цикла с постусловием

В языке C++ имеется три вида операторов цикла:

- for - оператор цикла
  с параметром - счетчиком
  (счетный оператор цикла)
  (или арифметический оператор цикла).
- while - оператор цикла
  с предварительным условием
  (с предусловием);
- do-while - оператор цикла
  с последующим условием
  (с постусловием);

Оператор цикла for

применяется при заранее известном количестве повторений.

При этом некоторая переменная, называемая параметром цикла, должна последовательно принимать значения от начального до конечного.

- Цикл ***for*** организуется с помощью специальной переменной, которая называется *параметром цикла*.
- *Параметр цикла* - это числовая переменная, которая управляет работой цикла. Она изменяется по закону арифметической прогрессии, что обеспечивает повторение цикла нужное количество раз.

Параметры цикла

- Для определения количества повторений заранее должны быть известны:
- начальное значение параметра - $t_{нач}$;
- конечное значение параметра - $t_{кон}$;
- шаг изменения параметра - Δt.
- Тогда *количество повторений цикла* определится по формуле:

$$n = \left[\frac{t_{кон} - t_{нач}}{\Delta t} \right] + 1$$

Структура цикла *for* на C++

имеет 4 блока, выполняющиеся в следующей последовательности:

1. - *блок инициализации* - параметру цикла присваивается начальное значение;
2. - *условие* выхода из цикла (или, напротив - условие повторения цикла) - проверка параметра на конечное значение
3. - *тело цикла* основные действия, которые повторяются каждый раз, на каждом витке цикла;
4. - *блок изменения* параметра цикла на величину шага.

Блок-схема арифметического цикла и общий вид и работа цикла for

Параметр цикла

Начальное значение

Изменение параметра цикла

for(<п.цк.> = <н.з.>; <*условие выполнения цикла*>; <*изм. п.цк.*>)

← ***тело цикла***

<*оператор*>;

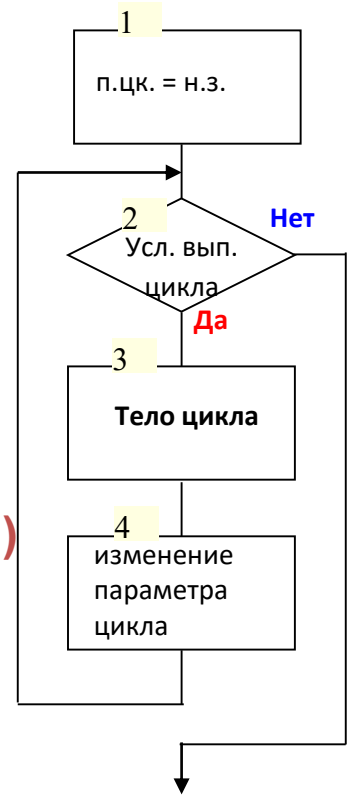
Любой оператор

Блок инициализации

for(<*п.цк.*> = <н.з.>; <*условие выполнения цикла*>; <*изм. п.цк.*>)

{<*оператор1*>; <*оператор2*>; … <*операторn*>; }

тело цикла



Пример

```
#include <iostream>;
#include <stdio.h>;
using namespace std;
int main()
{
	int sum = 0, i;
	for (i=1; i<10; i++) sum+=i;
	cout << sum << endl;
	getchar();
}
```

В C++ допускается объявление переменных прямо в строке инициализации цикла for. В этом случае, предыдущий пример программы примет вид

```
#include <iostream>;
#include <stdio.h>;
using namespace std;
void main() {
	int sum = 0;
	for (int i=1; i<10; i++) sum+=i;
	cout << "sum=" << sum << endl;
	getchar();
}
```

Обращение к переменной i, объявленной в цикле, вне цикла приведет к ошибке (область видимости переменной ограничивается циклом).

Пример

```
#include <iostream>;
#include <stdio.h>;
using namespace std;
void main() {
   int sum = 0;
   for (int i=1; i<10; i++) sum+=i;
   cout << "sum=" << sum << "i="<< i << endl; //в этой
                                              //строке ошибка
   getchar();
}
```

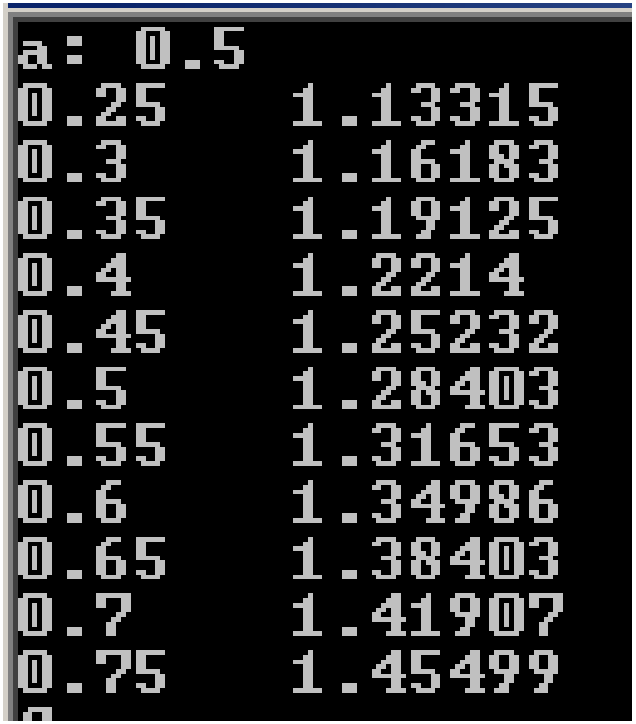
Пример . Допустим, что в группе из N человек собираются профсоюзные взносы по m рублей. Рассчитать, какую сумму группа перечислит в профсоюзный фонд.

Метод накопления суммы

```
#include<iostream>
using namespace std;
int main()
{int N=20,m=100,s=0;
for(int i=1; i<=N; i++)
s+=m;
cout<< "Summa = "<<s<<endl;
system("pause");
    return 0;
}
```

5. Пример программы с использованием счетного оператора цикла

$y = e^{ax}$, $x \in [0{,}25; 0{,}75]$, $\Delta x = 0{,}05$



Кол-во циклов

$$n = \frac{\text{к.з.} - \text{н.з.}}{\text{шаг}} + 1 = \frac{0.75 - 0.25}{0.05} + 1 = 10 + 1 = 11$$

Программа с использованием счетного оператора цикла

$y = e^{ax}$, $x \in [0{,}25; 0{,}75]$, $\Delta x = 0{,}05$

```
int main( )
{
   double a, y; int n, i;
   cout << "a: ";  cin >> a;
   double x,  xn = 0.25, xk = 0.75, dx = 0.05;
    n = (xk - xn) / dx+1;
   cout <<  "\ni" << setw(5) << "x" << "          y\n\n";

   for (x=xn, i = 1 ; i < =n  ; i++, x += dx)
   {
      y = exp(a * x);

     cout << left <<  setw(5) << i << setw(7) << x << y << endl;

      x += dx;
    }
  return 0;
}
```

```
a: 0.5

i     x       y

1     0.25    1.13315
2     0.3     1.16183
3     0.35    1.19125
4     0.4     1.2214
5     0.45    1.25232
6     0.5     1.28403
7     0.55    1.31653
8     0.6     1.34986
9     0.65    1.38403
10    0.7     1.41907
11    0.75    1.45499
```

\\ setw - Задает ширину отображаемого поля.

Параметр цикла вещественного типа

$y = e^{ax}$, $x \in [0{,}25; 0{,}75]$, $\Delta x = 0{,}05$

```
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

int main( )
{
    double a, y;
    cout << "a: ";  cin >> a;
 double xn = 0.25, xk=0.75, dx=0.05;

for(double x = xn; x < = xk; x += dx)
    {
      y = exp(a * x);
      cout << left << setw(7) << x << y << endl;
     }
  return 0;
}
```

```
a: 0.5
0.25   1.13315
0.3    1.16183
0.35   1.19125
0.4    1.2214
0.45   1.25232
0.5    1.28403
0.55   1.31653
0.6    1.34986
0.65   1.38403
0.7    1.41907
Для продолжения нажм
```

Вещественные значения НЕЛЬЗЯ сравнивать на «равно»: в силу **приближённого** представления в цифровом ПК вещественных чисел.

Параметр цикла вещественного типа

$y = e^{ax}$, $x \in [0{,}25; 0{,}75]$, $\Delta x = 0{,}05$

```
#include <iostream>
#include <iomanip>
#include <cmath>

using namespace std;

int main( )
{
    double a, y;
    cout << "a: ";  cin >> a;
 double xn = 0.25, xk=0.75, dx=0.05;

for(double x = xn; x < xk + dx/2; x += dx)
    {
      y = exp(a * x);
      cout << left << setw(7) << x << y << endl;
     }
  return 0;
}
```

```
a: 0.5
0.25    1.13315
0.3     1.16183
0.35    1.19125
0.4     1.2214
0.45    1.25232
0.5     1.28403
0.55    1.31653
0.6     1.34986
0.65    1.38403
0.7     1.41907
0.75    1.45499
```

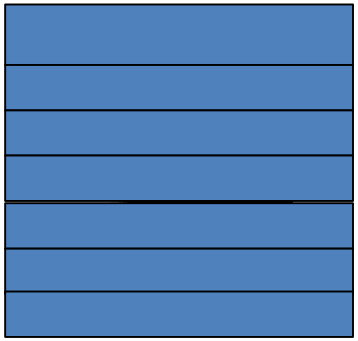
Использование нескольких переменных управления циклом

Цикл for является одним из наиболее гибких операторов, т.к. допускает большое кол-во разнообразных вариантов.

Например, допустимо использовать несколько переменных управления.

Пример:

```
for (x = 0, y = 10;   x <= y;   ++x, --y)
cout << x << ' ' << y <<'\n';
```



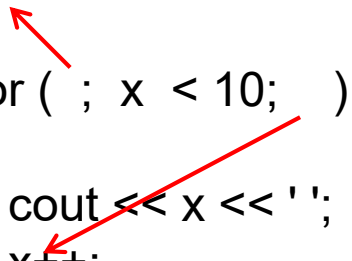
Пропущенные секции в операторе for

Пример (отсутствует секция приращения):
Цикл, который должен выполняться до тех пор, пока с клавиатуры не будет введено число 123.

```
int   x;

for (x = 0;  x != 123;    )
{
   cout << "enter number: ";
   cin  >> x;
}
```

Пример (отсутствуют секции инициализации и приращения):

```
x = 0;

for (  ;  x  < 10;    )
{
   cout << x << ' ';
   x++;
}
```



Пример (отсутствуют все секции – бесконечный цикл):

```
for (   ;    ;    )
{
   // …
}
```

Например: кроты запасли в своей норке **S** штук зерен. С определенной периодичностью они обновляют запасы своих норок **ds1= a | sin (3t+2) |** и поедают их с величиной **ds2 = 5·10^3**. Хитрые же мыши с другой периодичностью иногда обворовывают, а иногда и возвращают награбленное у кротов на величину **ds3 = b (sin (5t-4))**. Запустить процесс заполнения/опустошения норки.

Для выхода из такого цикла необходимо в теле цикла использовать оператор break, размещенный внутри условного оператора if.

```
double s=2.3e20; int t=0; float a=35,b=52;
for (   ;    ;    )
{t++; ds1 = a*abs(sin(3*t)+2;  ds2=5e3 ; ds3 = b *(sin (5t-4));
s+=ds1-ds2- ds3;
If (s<=0) break;
}
```

Пример (отсутствует тело цикла):
(бестелесые циклы весьма полезны)

```
int i;
int sum = 0;

// суммирование чисел от 1 до 10
for ( i = 1  ; i <= 10  ;  sum += i++  ) ;  // цикл без тела цикла
```

Чтобы понять смысл оператора sum += i++ (это обычная запись для C++)
необходимо разобрать его на составные части:

```
sum = sum + i;
i = i + 1;
```

2.3 Табулирование функции счетным оператором

Написать программу печати таблицы значений функции

$$y = \begin{cases} t, & x < 0 \\ tx, & 0 \le x < 10 \\ 2t, & x \ge 10 \end{cases}$$

для аргумента, изменяющегося в заданных пределах с заданным шагом. Если $t > 100$, должны выводиться целые значения функции.

Исходными данными являются начальное значение аргумента Xn, конечное значение аргумента Xk, шаг изменения агрумента dX и параметр t. Все величины — вещественные. Программа должна выводить таблицу, состоящую из двух столбцов — значений аргумента и соответствующих им значений функции.

Словесный алгоритм задачи

В словесной форме алгоритм можно сформулировать так:

1. Ввести исходные данные.
2. Взять первое из значений аргумента.
3. Определить, какому из интервалов оно принадлежит.
4. Вычислить значение функции y по соответствующей формуле.
5. Если t > 100, преобразовать значение y в целое.
6. Вывести строку таблицы.
7. Перейти к следующему значению аргумента.
8. Если оно не превышает конечное значение, повторить шаги 3–7, иначе закончить выполнение.

В каждый момент времени требуется хранить одно значение функции, поэтому для него достаточно завести одну переменную вещественного типа. Шаги 3–7 повторяются многократно, поэтому для их выполнения надо организовать цикл.

Решение задачи

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
        float xn, xk, dx, t , y;
        printf("Enter xn, xk, dx,t \n");// \n -переход на
                                        // строку = endl
        cin>>xn>>xk>>dx>>t;
        for (float x=xn;x<=xk;x+=dx)
        {
                if(x<0) y=t;
                else if (x>=0 && x<10) y=t*x;
                else y=2*t;
          if (t>100) printf("%9.2f   %9d    \n", x,(int)y);
          else printf("%9.2f   %9.2f    \n", x, y);
        }
        system("pause");
        return 0;
}
```

```
Enter xn, xk, dx,t
-1 12.1 0.5 102
    -1.00          102
    -0.50          102
     0.00            0
     0.50           51
     1.00          102
     1.50          153
     2.00          204
     2.50          255
     3.00          306
     3.50          357
     4.00          408
     4.50          459
     5.00          510
     5.50          561
     6.00          612
     6.50          663
     7.00          714
     7.50          765
     8.00          816
     8.50          867
     9.00          918
     9.50          969
    10.00          204
    10.50          204
    11.00          204
    11.50          204
    12.00          204
Для продолжения нажмите
```

Замечание о внутренних переменных

- Переменная x описана ВНУТРИ цикла, после его завершения, переменная x УДАЛЯЕТСЯ из памяти и мы не можем получить доступ к ее значению!!!
- Область видимости этой переменной – только тело оператора for.

В программу введена вспомогательная переменная x, которая последовательно принимает значения от Xn до Xk с шагом dX. Она определена непосредственно перед использованием. Это является хорошим стилем, поскольку снижает вероятность ошибок (например, таких, как использование неинициализированной переменной).

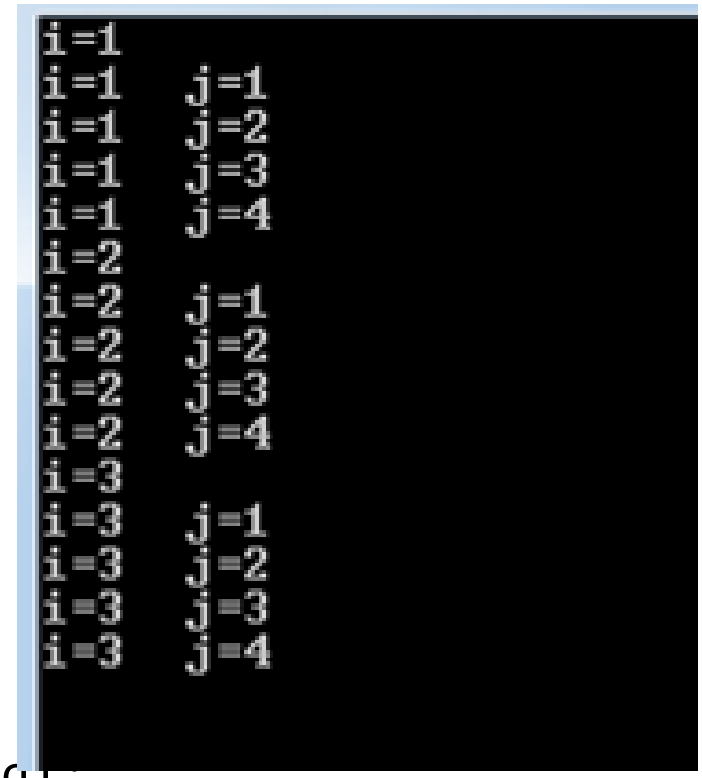
Вложенные циклы for

Синтаксис вложенных циклов **for**

```
//Внешний цикл
for (/*инициализирующее выражение */ ; /* условное выражение */;
      /* модифицирующее выражение */ )
{
      /*один оператор или блок операторов*/;
      // Внутренний цикл
      for (/*инициализирующее выражение */ ; /* условное выражение */;
            /* модифицирующее выражение */ )
      {
                  /*один оператор или блок операторов*/;
       }
}
```

Пример 1

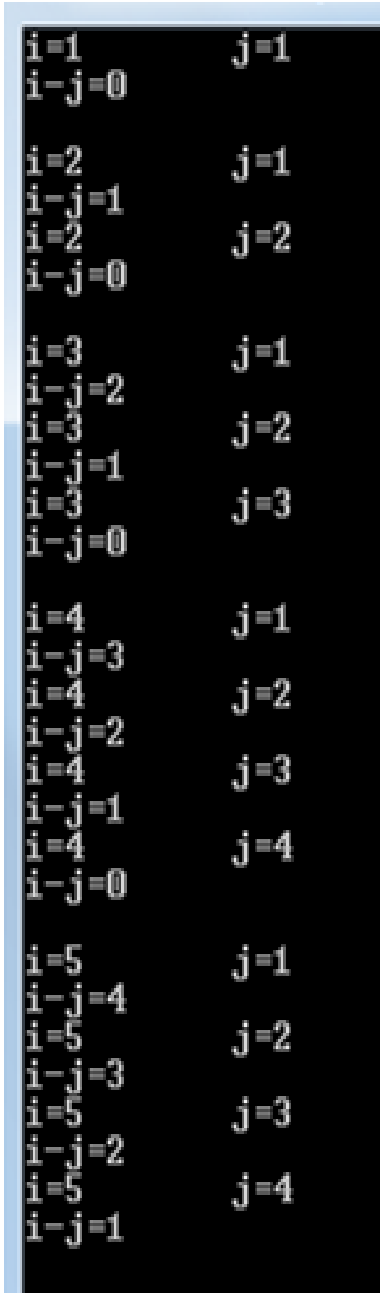
```
#include <iostream>;
#include <stdio.h>;
using namespace std;
void main() {
	int sum = 0;
	for (int i=1; i<4; i++) {
		cout << "i=" << i << endl;
		for (int j=1; j<5; j++)
		  cout << "i=" << i << "  j="<< j << endl;
	}
	getchar();
}
```



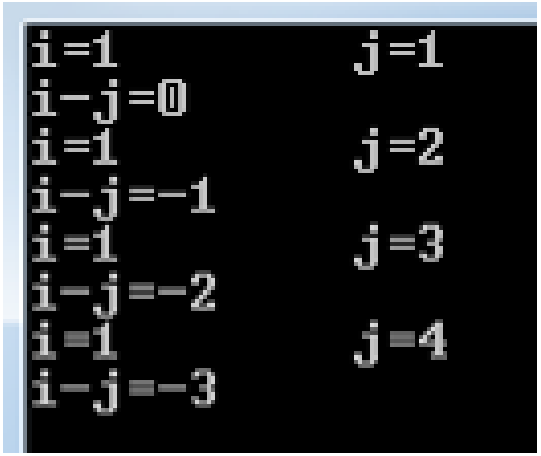
Оператор break

Оператор **break** (разрыв) вызывает немедленный выход из циклов, организуемых с помощью операторов **for, while, do-while, switch**; управление передается на оператор, следующий за законченным.

```
#include <iostream>;
#include <stdio.h>;
using namespace std;
void main() {
    int sum = 0;
        for (int i=1; i<6; i++)  {
            for (int j=1; j<5; j++) {
                if (i < j)  break;
                cout << "i=" << i << "\t  j="<< j << endl;
                cout << "i-j=" << i-j << endl;
            }
   cout << endl;
   }
   getchar();
}
```



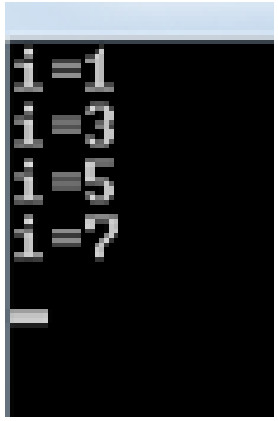
Пример 2




```
#include <iostream>;
#include <stdio.h>;
using namespace std;
void main() {
     int sum = 0, j;
     for (int i=1; i<6; i++)  {
           for (j=1; j<5; j++) {
                 cout << "i=" << i << "\t  j="<< j << endl;
            cout << "i-j=" << i-j << endl;
           }
           if (i < j)  break;
           cout << endl;
     }
     getchar();
}
```

Оператор continue

Оператор **continue** передает управление на следующую итерацию того цикла, в теле которого он находится.



Пример 1

```
#include <iostream>;
#include <stdio.h>;
using namespace std;
void main() {
	int sum = 0;
	for (int i=1; i<8; i++)  {
		if (i % 2 == 0)  continue;
		  cout << "i=" << i << endl;
		}
	getchar();
}
```

Задания для самостоятельной работы

Определите что будет выведено на экран в результате работы следующей программы.

```
#include <iostream>;
#include <stdio.h>;
using namespace std;
void main() {
    int a,b=0,i;
    for (i=1;i<=6;i++)
    {
        a=i+2;
        if (a>=5) b-=a;
        else b+=a;
    }
    cout<< "b="<<b<<endl;
    getchar();
}
```

Наберите текст программы и проверьте правильность ответа.

Задания для самостоятельной работы

Определите что будет выведено на экран в результате работы следующей программы.

```
#include <iostream>;
#include <stdio.h>;
using namespace std;
void main() {
     int a=0,i,j;
     for (i=12;i<=15;i++)
          for (j=4;j>=2;j--)
          if (i%j!=0) continue;
          else
             a+=i/j;
    cout<< "a="<<a<<endl;
    getchar();
}
```

Наберите текст программы и проверьте правильность ответа.

Вычисление суммы и произведение ряда

Вычисление конечной суммы сводится к нахождению суммы заданного количества слагаемых:

$$S = f(1) + f(2) + \ldots + f(n) = \sum_{i=1}^{n} f(i)$$

где i – номер слагаемого; $f(i)$ – слагаемое с номером i.

- Вычисление организуется в виде циклического алгоритма, когда при каждом прохождении цикла номер слагаемого *i* увеличивается на 1, а сумма изменяется на величину *i*-го слагаемого:

$$S_i = S_{i-1} + f(i)$$

Цикл повторяется до тех пор, пока не будут просуммированы все *n* слагаемых. Для того, чтобы начальное содержимое ячейки суммы не исказило результат, сумма предварительно должна быть обнулена

$$S_0 = 0$$

Вывод результата, поскольку он является единственным, осуществляется после окончания работы цикла.

main.cpp

```cpp
#include <iostream>
#include<cmath>
using namespace std;

int main()
{
   float S;
   int i,n;
   S=0;
   cout<<"Введите длину ряда";
   cin >> n;

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

cout<<"Сумма ряда="<<S;

    return 0;
}
```

```
Введите длину ряда5
Сумма ряда=15
```

main.cpp

```cpp
#include <iostream>
#include<cmath>
using namespace std;

int main()
{
   float S,k,x;
   int i,n;
   S=0;
   x=0.5;
   cout<<"Введите длину ряда";
   cin >> n;
for (i=1; i<=n;i++)
    {
    k=1-x*x/(i*i*M_PI*M_PI);
    S+=k;
    }
cout<<"Сумма ряда="<<S<<endl;
cout<<M_PI;

    return 0;
}
```

input

```
Введите длину ряда5
Сумма ряда=4.96293
3.14159

...Program finished with exit code 0
Press ENTER to exit console.
```

Вычисление конечного произведения представляет собой процесс нахождения произведения заданного количества сомножителей по формуле

$$P = f(1) \cdot f(2) \cdot \ldots \cdot f(n) = \prod_{i=1}^{n} f(i)$$

Как и суммирование, вычисление произведения организуется с помощью циклического процесса по рекуррентному соотношению

$$P(i) = P_{i-1} \cdot f(i)$$

В отличие от суммирования начальное значение произведения $P_0 = 1$

Пример. Вычислить факториал числа *N*.

$$y = n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n = \prod_{i=1}^{n} i$$

main.cpp

```
1  #include <iostream>
2  #include<cmath>
3  using namespace std;
4
5  int main()
6  {
7      long n,p;
8      int i;
9      cout<<"Введите длину ряда";
10     cin >> n;
11     p=1;
12 for (i=1; i<=n;i++)
13         p*=i;
14
15 cout<<"Произведение="<<p<<endl;
16
17     return 0;
18 }
19
```

```
Введите длину ряда10
Произведение=3628800
```

Операторы цикла while и do -while

применяются в тех случаях, когда известно

условие выполнения цикла,

а количество повторений может быть заранее не известно.

Оператор цикла с предусловием

Общий вид

Пока

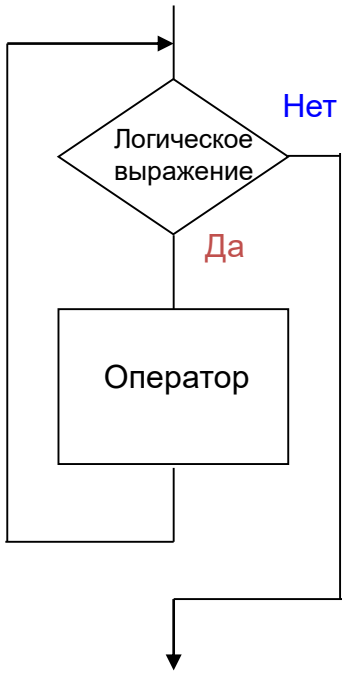
Условие выполнения цикла

while (<*логическое выражение*>)

← *тело цикла*

<*оператор*>;

Любой оператор



- ***Пример 1***. Автомобиль движется со скоростью 5 км/ч и начинает наращивать скорость с ускорением 10 км/ч^2 до тех пор пока не будет достигнута скорость 60 км/час. Определить, за какое время эта скорость будет достигнута.

```
#include<iostream>
using namespace std;
int main()
{int speed = 5, time = 0, count=0;
    while ( speed < 60 )
    {
        cout << count <<"-speed = " << speed
<< " time= "<<time<<endl;
            speed += 10; // приращение скорости
            count++; // подсчёт повторений цикла
            time++;// наращивание времени
    }
     cout<< "final speed "<<speed<<" was
reached in "<< time<< " h"<<endl;
    system("pause");    return 0;}
```

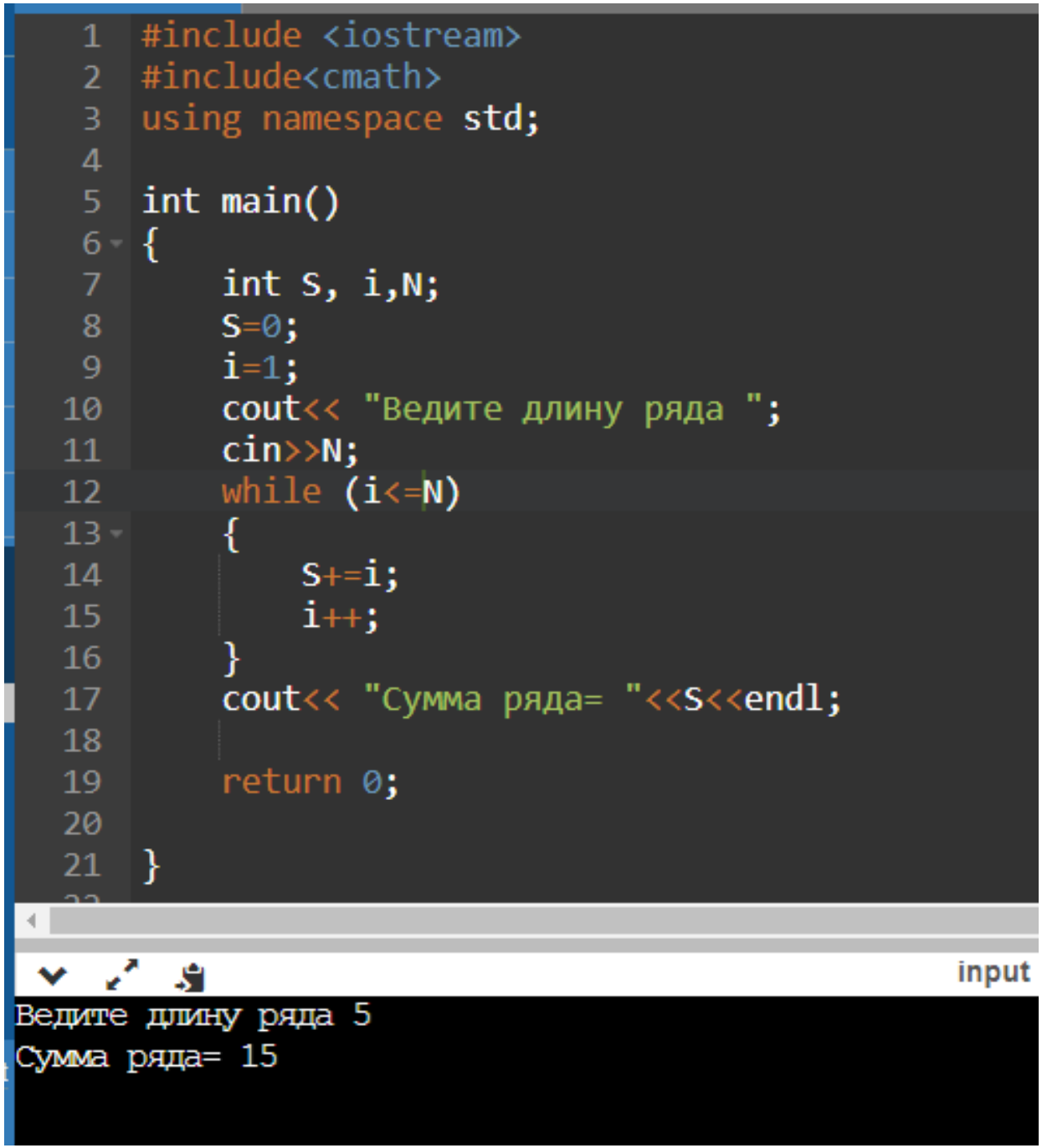
main.cpp

```cpp
#include <iostream>
#include<cmath>
using namespace std;

int main()

{
    int speed = 5, time = 0, count=0;
    while ( speed < 60 )
    {
        cout << count <<"-speed = " << speed << " time= "<<time<<endl;
        speed += 10;
        count++;
        time++;
    }
    cout<< "final speed "<<speed<<" was reached in "<< time<< " h"<<endl;

    return 0;
```

```
0-speed = 5 time= 0
1-speed = 15 time= 1
2-speed = 25 time= 2
3-speed = 35 time= 3
4-speed = 45 time= 4
5-speed = 55 time= 5
final speed 65 was reached in 6 h
```

Анализ программы

- *инициализация* трёх переменных (скорости *speed*, времени *time* и счётчика цикла *count* реализуется до начала цикла;
- *условие* **speed < 60** определяет возможность выполнения цикла, и пока скорость остаётся меньше 60, условие истинно и скорость будет нарщиваться;
- *управление условием* реализуется оператором **speed += 10**;
- *тело цикла* составляют операторы вывода на консоль и операторы приращения счётчика и времени.


#include <iostream>
#include<cmath>
using namespace std;
int main()
{
int S, i,N;
S=0;
i=1;
cout<< "Ведите длину ряда ";
cin>>N;
while (i<=N)
{
S+=i;
i++;
}
cout<< "Сумма ряда= "<<S<<endl;
return 0;
}
input
Ведите длину ряда 5
Сумма ряда= 15

```cpp
#include <iostream>
#include<cmath>
using namespace std;

int main()
{
    int S, i,N,M;
    S=0;
    i=0;
    cout<< "Ведите длину ряда  ";
    cin>>N;
    cout<< "Введите число M  ";
    cin>>M;
    while (S<M)
    {   i++;
        S+=i;
    }
    cout<< "Сумма ряда= "<<S-i<<endl;

    return 0;

}
```

```
Ведите длину ряда  5
Введите число M  16
```

Найти сумму ряда ***N*** натуральных чисел, не превышающих произвольного числа ***M***

Оператор цикла с постусловием

Общий вид

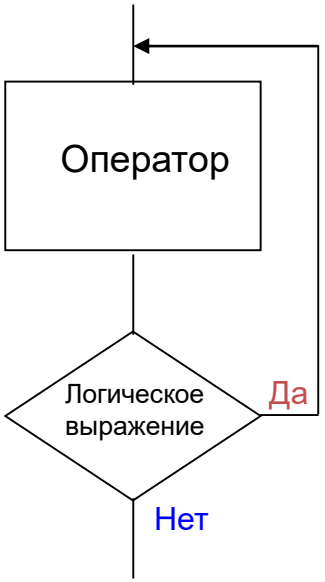
```
do
    <оператор> ;
while(<логическое выражение>);
```

Выполнять

Тело цикла

До тех пор, пока

Условие повторения цикла



```
#include <iostream>
#include<cmath>
using namespace std;
int main()
{
 int S, i,N,M;  S=0;   i=0;
 cout<< "Ведите длину ряда  ";  cin>>N;
cout<< "Введите число M  ";   cin>>M;
  do
 {
      i++;
     S+=i;
}
while (S<M);
cout<< "Сумма ряда= "<<S-i<<endl;
return 0;
}
```

main.cpp

```cpp
#include <iostream>
#include<cmath>
using namespace std;

int main()
{
    int S, i,N,M;
    S=0;
    i=0;
    cout<< "Ведите длину ряда  ";
    cin>>N;
    cout<< "Введите число M  ";
    cin>>M;
    do
    {
        i++;
        S+=i;
    }
    while (S<M);
    cout<< "Сумма ряда= "<<S-i<<endl;

    return 0;
}
```

input

```
Ведите длину ряда  5
Введите число M  16
Сумма ряда= 15


...Program finished with exit code 0
Press ENTER to exit console.
```

Технология программирования задачи с оператором цикла do-while

Составить программу:

Вычислить с заданной точностью сумму членов бесконечного ряда: $$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots = \sum_{n=1}^{\infty} \frac{1}{n}$$

Условие прекращения вычислений

$S_n - S_{n-1} =$ $< \delta$ – малое число больше нуля.

Выбор идентификаторов:

Входной	*Параметр цикла*	*Выходной*
$\delta \rightarrow$ d,	n $\rightarrow$ n,	S $\rightarrow$ s

$$S = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots = \sum_{n=1}^{\infty} \frac{1}{n}$$

$$S_n - S_{n-1} = \frac{1}{n} < \delta$$

```
int  main( )            // Заголовок функции
{        // Начало кода функции
    cout   <<   "d: ";
    double d;
     cin >> d;          0,3
   double s = 0;
   double n = 1;
   do
      {
        s +=  1 / n;
        n ++;
      }  while (1 / n > d);
   cout << "n = " << n << "   s = " << s << endl;
}
```

0 + 1/1 + 1/2 + 1/3

1 + 1 + 1 + 1

// Вывод ответа

1.8333

N	1/N	S
1	1	1.0000
2	0.5	1.5000
3	0.3333	1.8333
4	0.25	

Генерация псевдослучайных чисел средствами языка C++

Функции работы со случайными числами

- *Случайные числа на языке программирования C++* могут быть сгенерированы функцией rand() из стандартной библиотеки **cstdlib.**
- Функция **rand()** генерирует числа в диапазоне от **0** до **RAND_MAX**.
- **RAND_MAX** — ***это константа***, определённая в библиотеке <cstdlib>.
- Для Microsoft Visual Studio: **RAND_MAX = 32767,** но оно может быть и больше, в зависимости от компилятора.

- Для того чтобы масштабировать интервал генерации чисел нужно воспользоваться, операцией нахождения остатка от деления «**%**«.

// пример масштабирования диапазона генерации случайных чисел

rand() % 3 +1 // диапазон равен от 1 до 3 включительно

- Число 3 является масштабируемым коэффициентом. То есть, какое бы не выдал число генератор случайных чисел **rand()** запись **rand() % 3** в итоге выдаст число из диапазона от **0** до **2**. Для того чтобы сместить диапазон, мы прибавляем единицу, тогда диапазон изменится на такой — от **1** до **3** включительно.

```
#include <iostream>
using namespace std;
#include <cstdlib>
int main()
{
cout << "RAND_MAX = " << RAND_MAX << endl; // константа, хранящая
    максимальный предел из интервала случайных чисел
cout << "random number = " << rand() << endl; // запуск генератора
    случайных чисел
 cout <<rand() % 3 +1; // диапазон равен от 1 до 3 включительно
   return 0;
}
```

main.cpp

```cpp
#include <iostream>
using namespace std;
#include <cstdlib>
int main()
{
cout << "RAND_MAX = " << RAND_MAX << endl; // константа, хранящая максим
cout << "random number = " << rand() << endl; // запуск генератора случа
 cout <<rand() % 3 +1; // диапазон равен от 1 до 3 включительно
    return 0;
}

```

input

```
RAND_MAX = 2147483647
random number = 1804289383
2
```

- **rand()** сгенерирует случайное число один раз, при первом запуске программы. В дальнейшем, сколько бы Вы не запускали эту программу, сгенерированное число останется одним и тем же.

Функции работы со случайными числами

- Чтобы функция **rand()** всегда возвращала ***разные числа***, её нужно использовать в паре с функцией **srand().**
- Функция **srand()** получив целый положительный аргумент типа **unsigned** или **unsigned int** (без знаковое целое) выполняет рандомизацию, таким образом, чтобы при каждом запуске программы функция **srand(**) генерировала случайные числа.

Синтаксис: **srand(unsigned int *apг*).**

Аргумент **apг** задаёт то стартовое число, на базе которого и создаётся база случайных чисел.

- Пример:

```
unsigned rand_value = 11;
srand(rand_value); // рандомизация генератора случайных чисел
```

Или srand(11)

main.cpp

```cpp
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
    unsigned rand_value = 11;
    srand(rand_value); // рандомизация генератора случайных чисел
    cout << "rand_value = " << rand_value << endl;
    cout << "1-random number = " << 1 + rand() % 10 << endl; // первый запуск генератора случа
    cout << "2-random number = " << 1 + rand() % 10 << endl; // второй запуск генератора случа

    return 0;
}
```

input

```
rand_value = 11
1-random number = 4
2-random number = 5
sh: 1: pause: not found


...Program finished with exit code 0
```

- В **строке 7** выполняется функция **srand(),** которая принимает в качестве аргумента целое положительное число 11. При первом запуске мы получили случайные числа, и при последующих запусках программы мы видим всё те же числа. Так вот, чтоб каждый раз генерировались новые случайные числа необходимо, что бы менялся аргумент в функции **srand(**).

Чтобы производить рандомизацию автоматически, то есть, не меняя каждый раз аргумент в функции **srand()** нужно воспользоваться функцией time() с аргументом 0.

srand(time(0));

Чтобы использовать функцию time(), необходимо подключить заголовочный файл **<ctime>.**

main.cpp

```cpp

#include <iostream>
#include <ctime>
using namespace std;

int main()
{
    srand( time( 0 ) ); // автоматическая рандомизация
    cout << "rand_value = " << 1 + rand() % 10 << endl;
    return 0;
}
```

input

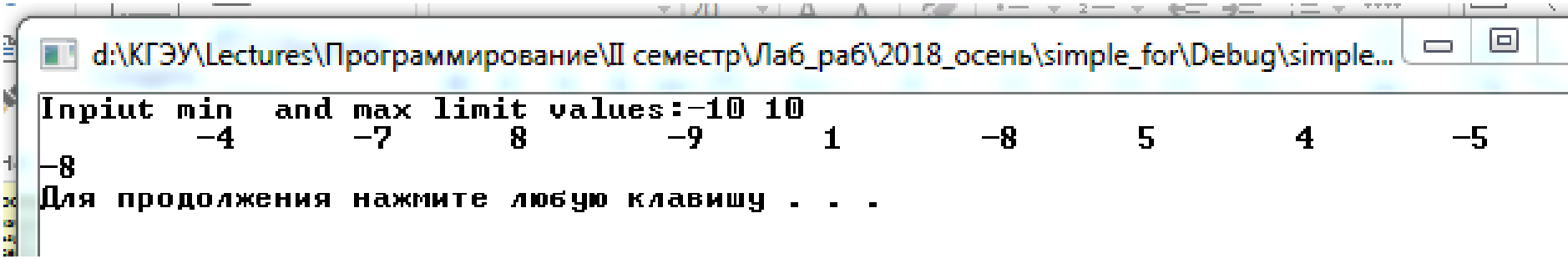
```
rand_value = 6


...Program finished with exit code 0
Press ENTER to exit console.
```

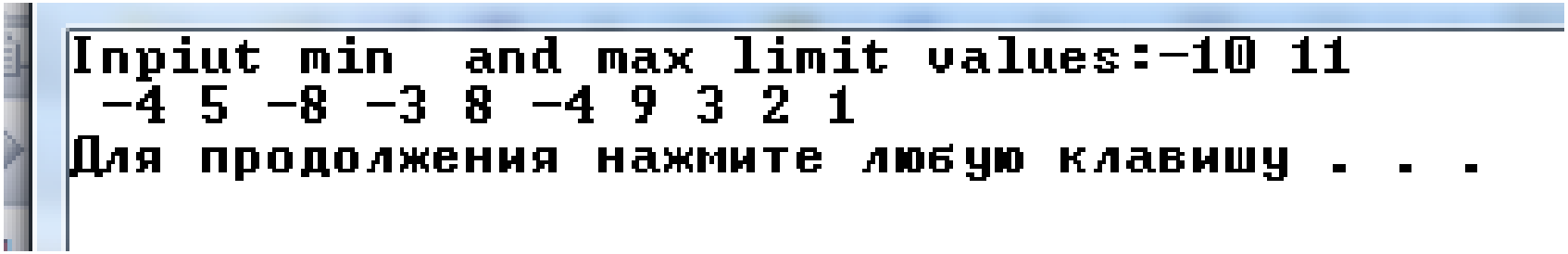
Особенности работы функции srand()

- Чаще всего в качестве передаваемой величины в функцию **srand()** используют системное время в секундах, т.к. это число будет всегда разным, а соответственно, мы будем получать на выходе из **rand()** разные случайные числа.
- Чтобы передать в функцию srand() текущее системное время, можно использовать библиотечную функцию **time(),** описанную в библиотеке <**ctime**>.
- Для того, чтобы эта функция возвращала текущее время в секундах (секунды отсчитываются от 00:00:00), нужно вызывать ее с параметром **NULL**: srand(time(NULL)); или **srand(time(0));.**
- Если нет необходимости в формировании всегда разных случайных чисел, то функцию **srand** можно задать с любой константой или вовсе не включать её в программу.

Пример 1. Инициализация массива случайными числами в заданном диапазоне значений: от -10 до 10.

tf[i] = rand( ) % (r_max - r_min+1) + r_min;





```
#include <iostream>
#include <ctime>
using namespace std;

// функция инициализации массива случайными числами
int main()
{
int tf [10], nf, r_min, r_max;
nf = 10;
cout<< “Inplut min  and max limit values:”; cin>>r_min>>r_max;
  srand( (unsigned int) time( NULL ) );   // рандомизация генератора
  for (int  i = 0;   i < nf;   i++ )          // n - количество чисел
    tf[i] = rand( ) % (r_max - r_min+1)  +  r_min;  // формирование случайного числа
for (int  i = 0;   i < nf;   i++ ) cout<<'\t'<< tf[i]; cout<<endl;
return 0;
}
```

Создание базы случайных чисел на основе Time (NULL)

Функция ***rand*** генерирует случайные числа, возвращает псевдослучайное целое число в диапазоне от 0 до ***RAND_MAX***. ***RAND_MAX*** это константа, определенная в <cstdlib>. По умолчанию её значение может изменяться, в зависимости от реализации, но, как правило, макрос RAND_MAX меньше значения 32767 не бывает.

Пример 1. Определить количество цифр в числе *N*, заданным случайным образом.

```
#include<iostream>
#include<cmath>
#include<ctime>
using namespace std;
int main()
{int Number,M,N, count;
srand(10);
cout<< " What is the maximal value of the Number?";
cin>>M;
N=rand()%M;
Number=N;
```

```
// Метод - цикл с делением
count = (Number == 0) ? 1 : 0;
        while (Number != 0)
         {
            count++;
            Number /= 10;
        }
cout<<"\nCounts of digits in the number
"<<N<<" is equal "<<count<<endl;
```

```
// Метод - десятичный логарифм и округление
// хорош для очень больших чисел.
N=rand()%M;
Number=N;
count=(Number == 0) ? 1 :
(int)ceil(log10(abs(Number) + 0.5));
cout<<"\nCounts of digits in the number "<<N<<"
is equal "<<count<<endl;
system("pause");
return 0;
}
```

```
What is the maximal value of the Number?10000

Counts of digits in the number 71 is equal 2

Counts of digits in the number 6899 is equal 4
Для продолжения нажмите любую клавишу . . .
```

Пример 2. Паук находится на плоскости в точке с координатами x=50 и y=50. Каждую секунду он делает шаг влево, вправо, вниз или вверх с равной вероятностью. Смоделируйте движение паука с помощью генератора случайных чисел.
Координаты x(t) и y(t) сохраните в одномерных массивах. Напечатайте траекторию паука в виде таблицы, которая содержит в клеточке с координатами x и y символ “.”, если там паук не был, “+”, если паук там побывал 1 раз, “*”- для двух раз, “#” - для трёх и символ “@” -, если он побывал больше раз.

Блок инициализации

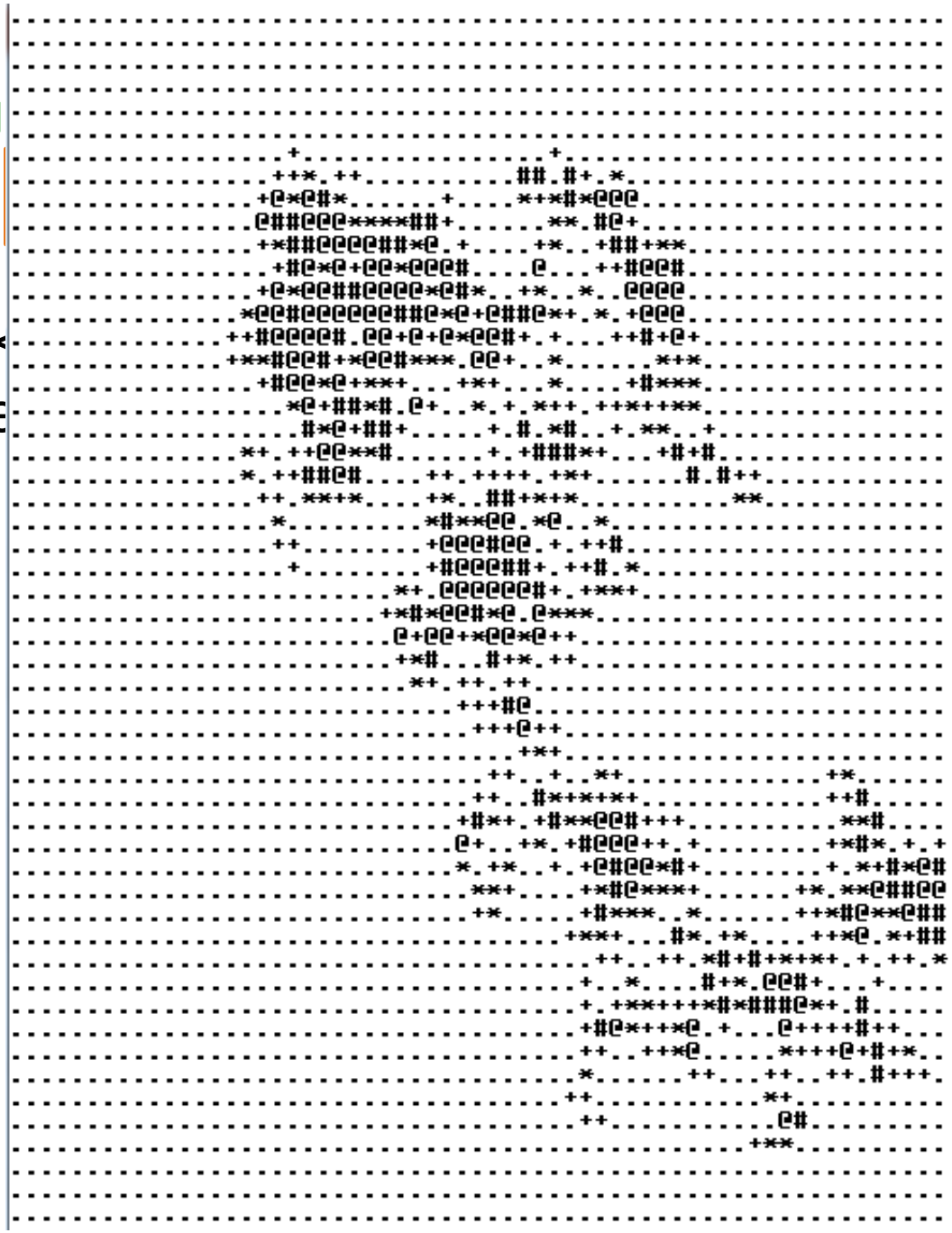
```
#include<iostream>
#include<ctime>
#include<iomanip>
using namespace std;
int main()
{
cout << " Случайное блужданиепо плоскости." << endl;
const int N=61, K=1550, J=1;
char buf[N+1]; ];// массив символов ~ количество посещени
int xy[N][N];// массив для хранения количества посещений
клетки N,N
int x=N/2;// ставим курсор в середину консоли
int y=N/2;
```

```
// Заполнение двумерного массива значениями при
// моделировании движения паука

for(int m=0; m<N; m++)
                for(int n=0; n<N; n++)
                xy[m][n]=0;
        xy[x][y]=1;
        // запись координат движения паука в массив
        for(int k=1; k<K; k++)
        {
           for(int j=0; j<J; j++)
          x+=rand()%3-1;//формирование сл.чисел от -1 до 1
           for(int j=0; j<J; j++)
              y+=rand()%3-1;
```

```
//проверка выхода к границам
        if(x<0)x=0;
        if(x>N-1)x=N-1;
        if(y<0)y=0;
        if(y>N-1)y=N-1;
        xy[x][y]+=1; // отметка о посещении точки в массиве
        }
```

```
// Заполнение символьного массива
        for(int m=0; m<N; m++)
        {
                for(int n=0; n<N; n++)
                        switch(xy[m][n])
                        {
                        case 0: buf[n]='.'; break;
                        case 1: buf[n]='+'; break;
                        case 2: buf[n]='*'; break;
                        case 3: buf[n]='#'; break;
                        default: buf[n]='@';
                        }
                buf[N]='\0';// ставим конец строки
```

```
// Выводим

        cout<
        syste
return 0;
}
```



Краткие итоги

- **Оператор for на** C++ состоит из четырёх секций: инициализации, условия, тела цикла, приращение. Любая из секций может быть опущена с соблюдением синтаксиса, позволяя гибко строить алгоритм, используя этот компактный и многофункциональный оператор.
- **Условные операторы while и do-while** по эффективности эквиваленты оператору **for,** удобны при описании условных алгоритмов.
- **Оператор for** необходим при работе с элементами массивов, при вычислении сумм конечных рядов, при любых алгоритмах, где используется счётчик при заранее известном количестве итераций.
- **При алгоритмизации задач с бесконечными рядами** можно использовать любые типы циклов, однако **while** и **do-while** многими полагаются как более наглядные.
- **Рекуррентные формулы** при вычислении сумм длинных рядов не только позволяют избегать операций прерывания, но и существенно экономят процессорное время.
- **Для оценки времени** удобно пользоваться функций **clock.**
- **Формула для формирования числа в заданном диапазоне значений**

```
tf[i] = rand( ) % (r_max - r_min+1)  +  r_min;
```

Контрольные задания. Ответы обосновать.

1. Сколько итераций сделает данный цикл? :

for(int k = 9, s = -3; k > s; k /= 1.5; s *= -1,5);

Можно ли узнать значение **s** после завершения цикла?

2. Чему будет равно значение **c** после выполнения операторов:

c=044; while(~(0x7|0xc)^c) c << 2; cout << c << endl;

3. Проанализируйте фрагмент программы и напишите, что будет выведено на консоль:

int k=0,z=0xCAF; for(int m=5,k=2;m>2; m--)

{z << k; k++; cout << z << '\t' << k << endl;} cout << k;

4. Какой результат выведет этот оператор?

cout << sizeof(2.*35/7e0) << endl;

5. В каком диапазоне значений будет сформировано число по оператору **x+=rand()%5 - 2; ?**