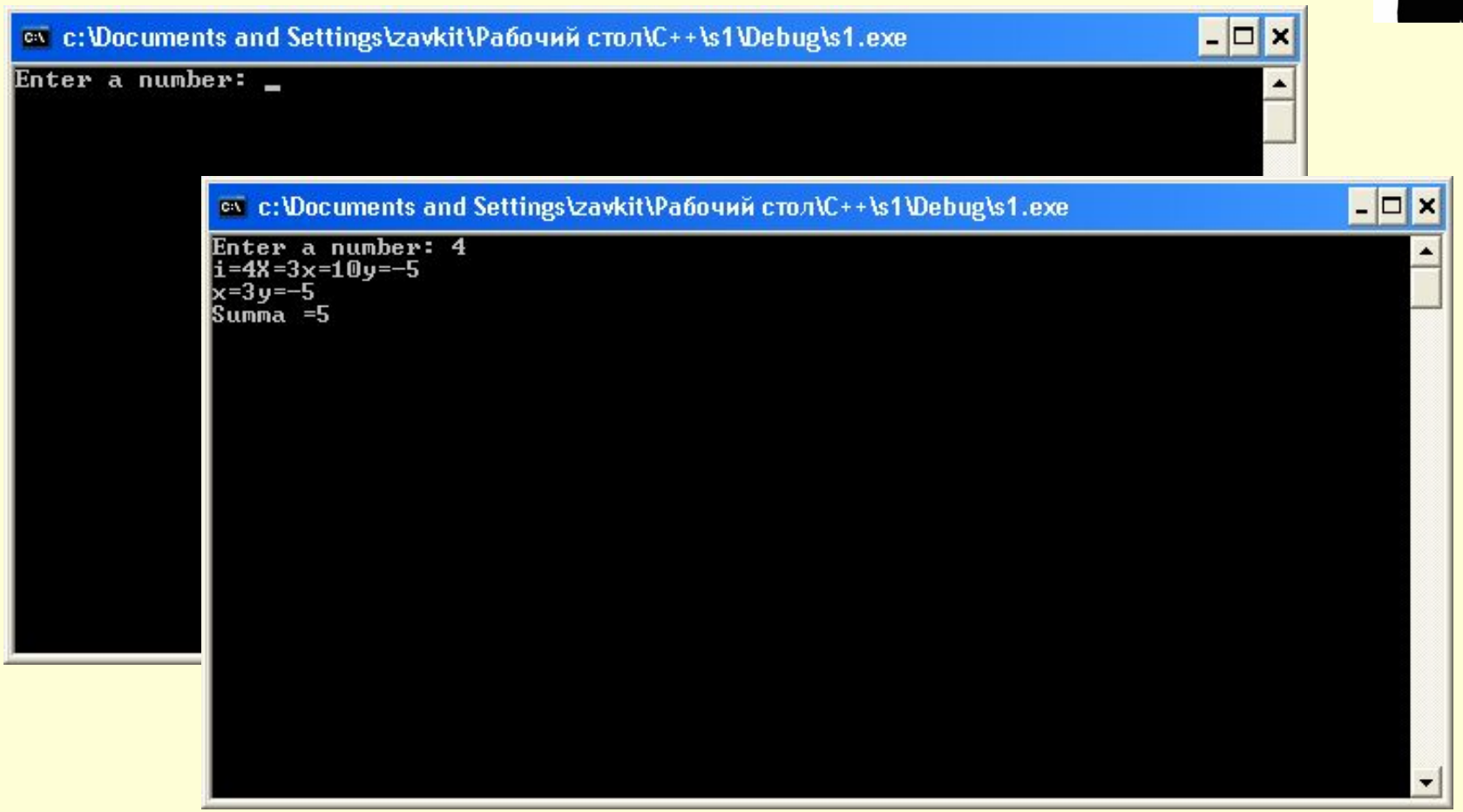
Примеры программ. Основные операторы C++

Ввод данных с помощью функции cin.
Вывод данных с помощью функции cout

```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include  <conio.h>
 using namespace std;

int _tmain()
{int i = 0;
int X=3;
int x=10;
int y=-5;
cout<< "Enter a number: ";
cin>>i;
   cout<<"i="<<i;
   cout<<"X="<<X;
cout<<"x="<<x<<"y="<<y<<
"\n";

cout<<"x="<<X<<"y="<<y<<
endl;
cout<<"Summa ="<<x+y;
   getch();
}
```

Результаты работы программы

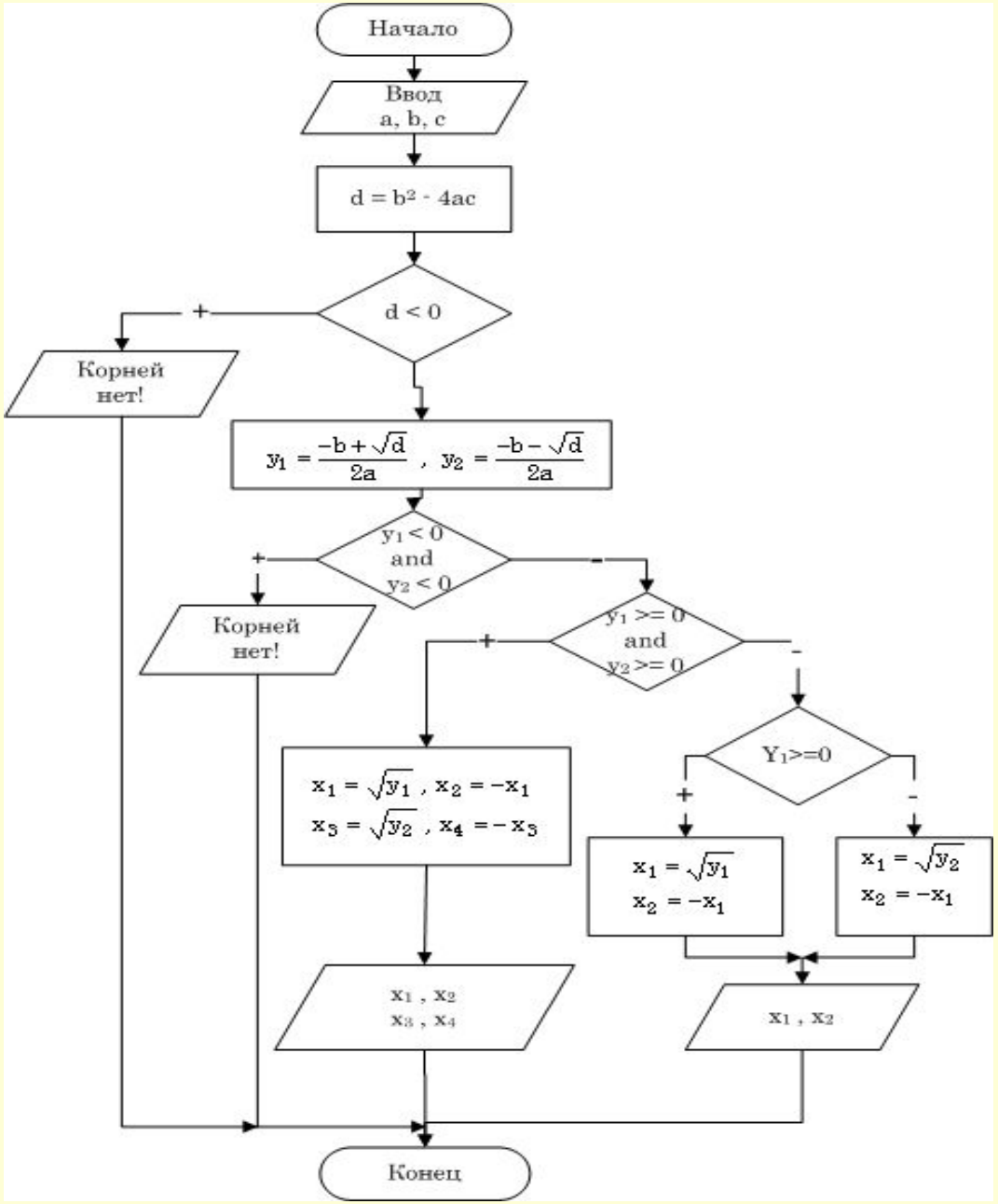


Заданы коэффициенты a, b и c биквадратного уравнения ax4 + bx2 + c = 0. Решить уравнение.

Дано:
a, b, c – коэффициенты биквадратного уравнения.
Найти:
x1, x2, x3, x4 – корни уравнения.

Для решения биквадратного уравнения необходимо заменой $y = x^2$ привести его к квадратному и решить это уравнение.

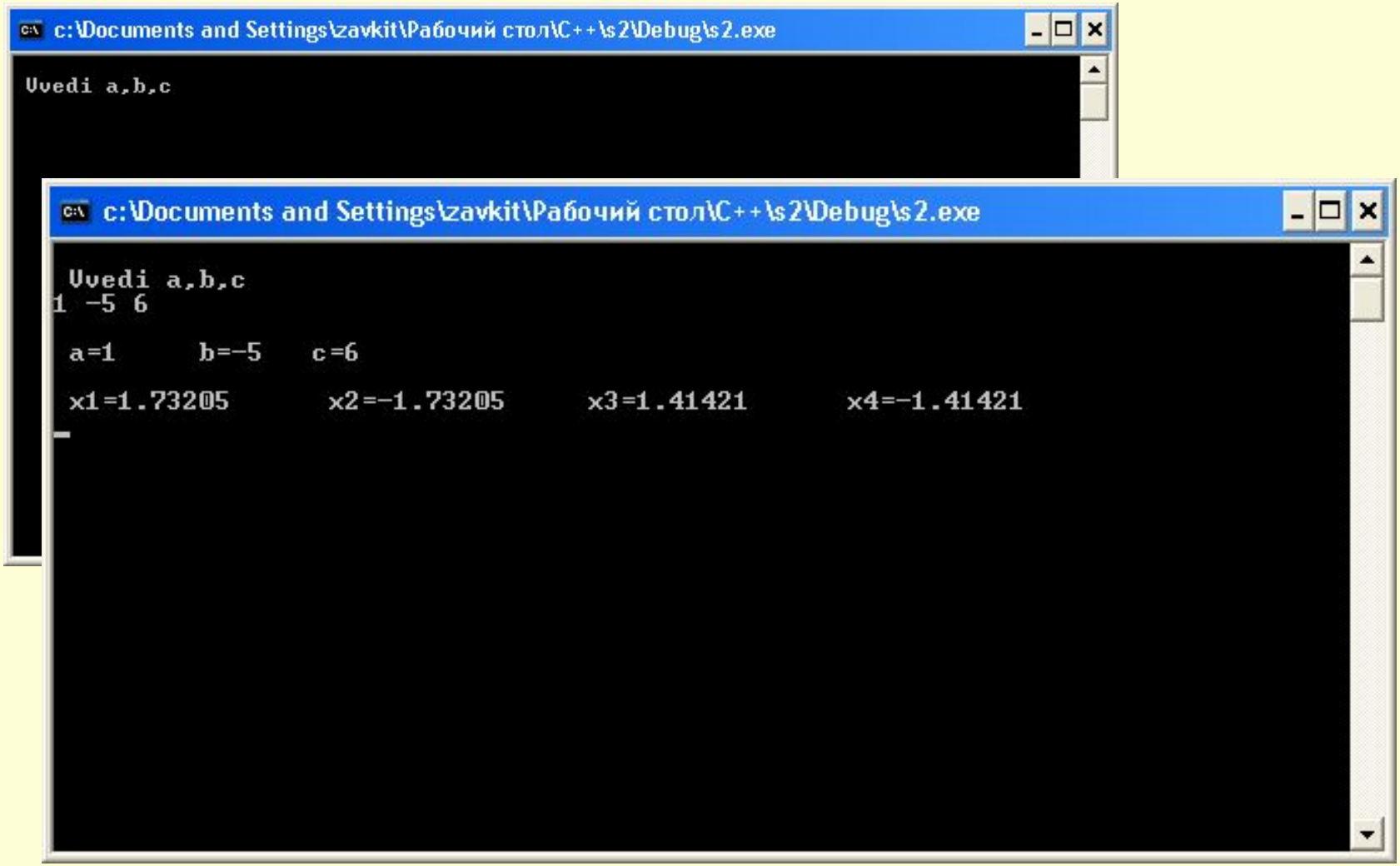
Блок - схема


Начало
Ввод
a, b, c
d = b² - 4ac
d < 0
+
Корней
нет!
$y_1 = \frac{-b+\sqrt{d}}{2a}$, $y_2 = \frac{-b-\sqrt{d}}{2a}$
$y_1 < 0$
and
$y_2 < 0$
+
Корней
нет!
-
$y_1 >= 0$
and
$y_2 >= 0$
+
$x_1 = \sqrt{y_1}$, $x_2 = -x_1$
$x_3 = \sqrt{y_2}$, $x_4 = -x_3$
-
$Y_1>=0$
+
$x_1 = \sqrt{y_1}$
$x_2 = -x_1$
-
$x_1 = \sqrt{y_2}$
$x_2 = -x_1$
x1 , x2
x3 , x4
x1 , x2
Конец

Текст программы

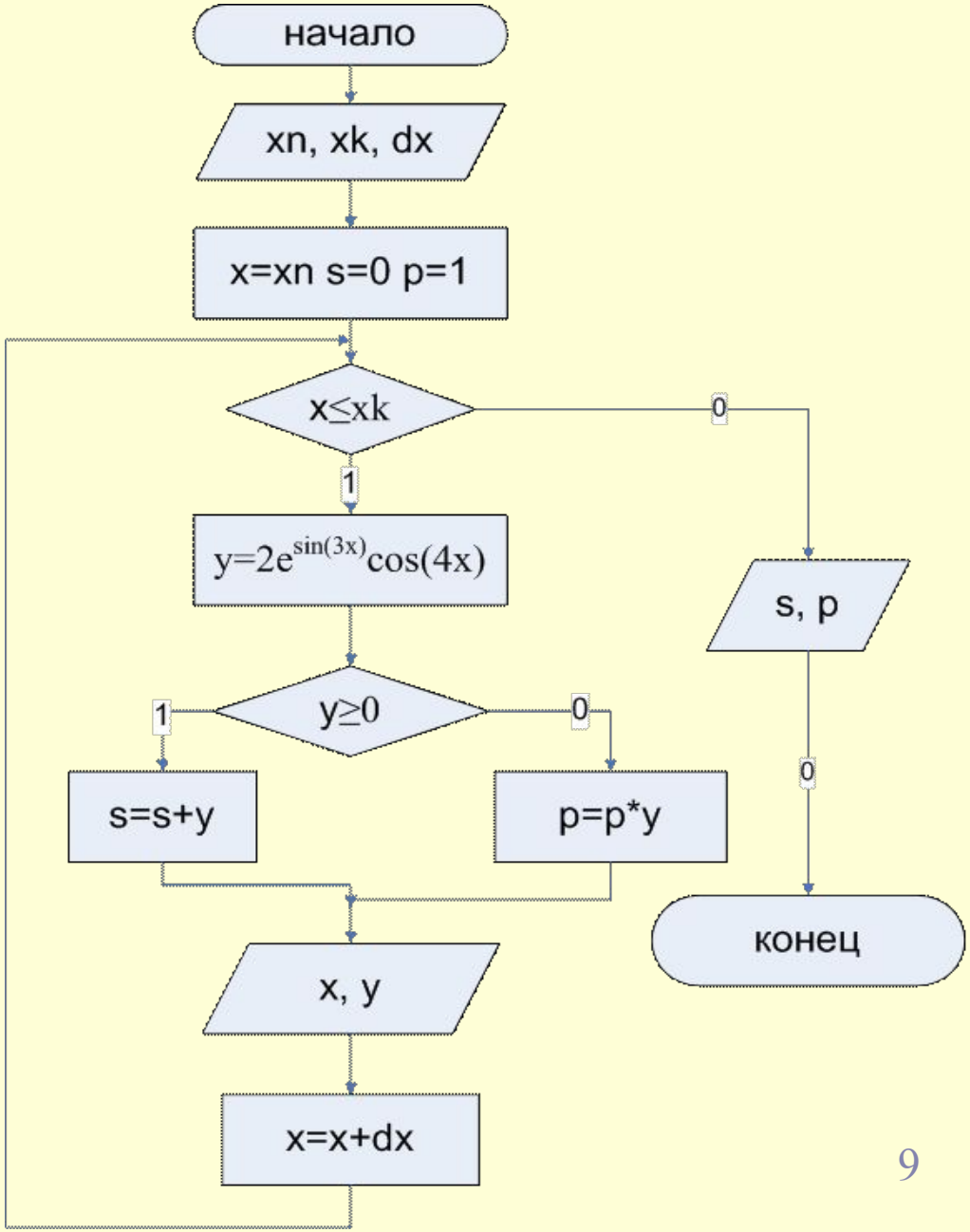
```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include  <conio.h>
#include  <math.h>

using namespace std;
int _tmain()
{ float sqrtf(float x);
float a,b,c,d,y1,y2,x1,x2,x3,x4;
printf("\n Vvedi a,b,c\n");
scanf("%f%f%f",&a,&b,&c);
printf("\n a=%g\tb =%g\tc=%g\n",
a,b,c);
d=b*b-4*a*c;
If (d<0) printf("No \n");
else { y1=(-b+sqrtf(d))/2/a;
y2=(-b-sqrtf(d))/2/a;
if ((y1<0) || (y2<0))
printf("No\n");
else if((y1>=0)||(y2>=0))
{ x1=sqrtf(y1); x2=-x1;
x3=sqrtf(y2); x4=-sqrtf(y2);
printf("\n x1=%g \t x2=%g \t
x3=%g \t 4=%g\n",x1,x2,x3,x4);
} else
if (y1>=0)
{ x1=sqrtf(y1); x2=-x1;
printf("\n x1=%g \t x2=%g \n",
x1,x2); }
else
{ x1=sqrtf(y2); x2=-x1;
printf("\n x1=%g \t x2=%g \n",
x1,x2);
} }
getch(); }
```

Результаты


c:\Documents and Settings\zavkit\Рабочий стол\C++\s2\Debug\s2.exe
Uvedi a,b,c
c:\Documents and Settings\zavkit\Рабочий стол\C++\s2\Debug\s2.exe
Uvedi a,b,c
1 -5 6
a=1 b=-5 c=6
x1=1.73205 x2=-1.73205 x3=1.41421 x4=-1.41421

Составить таблицу значений функции $y=2e^{\sin(3x)}\cos(4x)$ на отрезке [xn; xk] с шагом dx. Найти сумму положительных y и произведение отрицательных y.

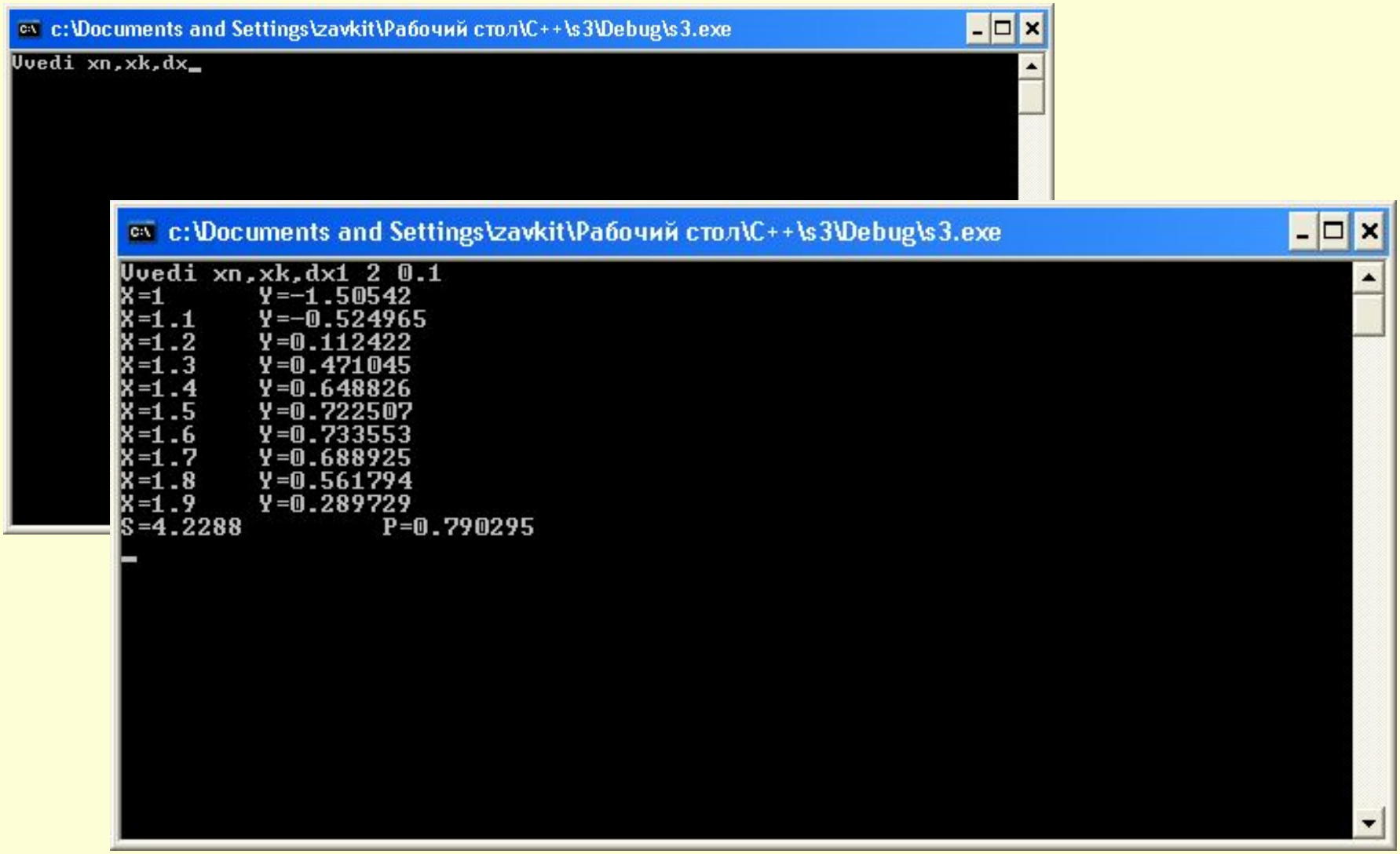
Блок - схема


начало
xn, xk, dx
x=xn s=0 p=1
x≤xk
0
1
y=2e^sin(3x)cos(4x)
s, p
y≥0
1
0
s=s+y
p=p*y
0
конец
x, y
x=x+dx

Текст программы

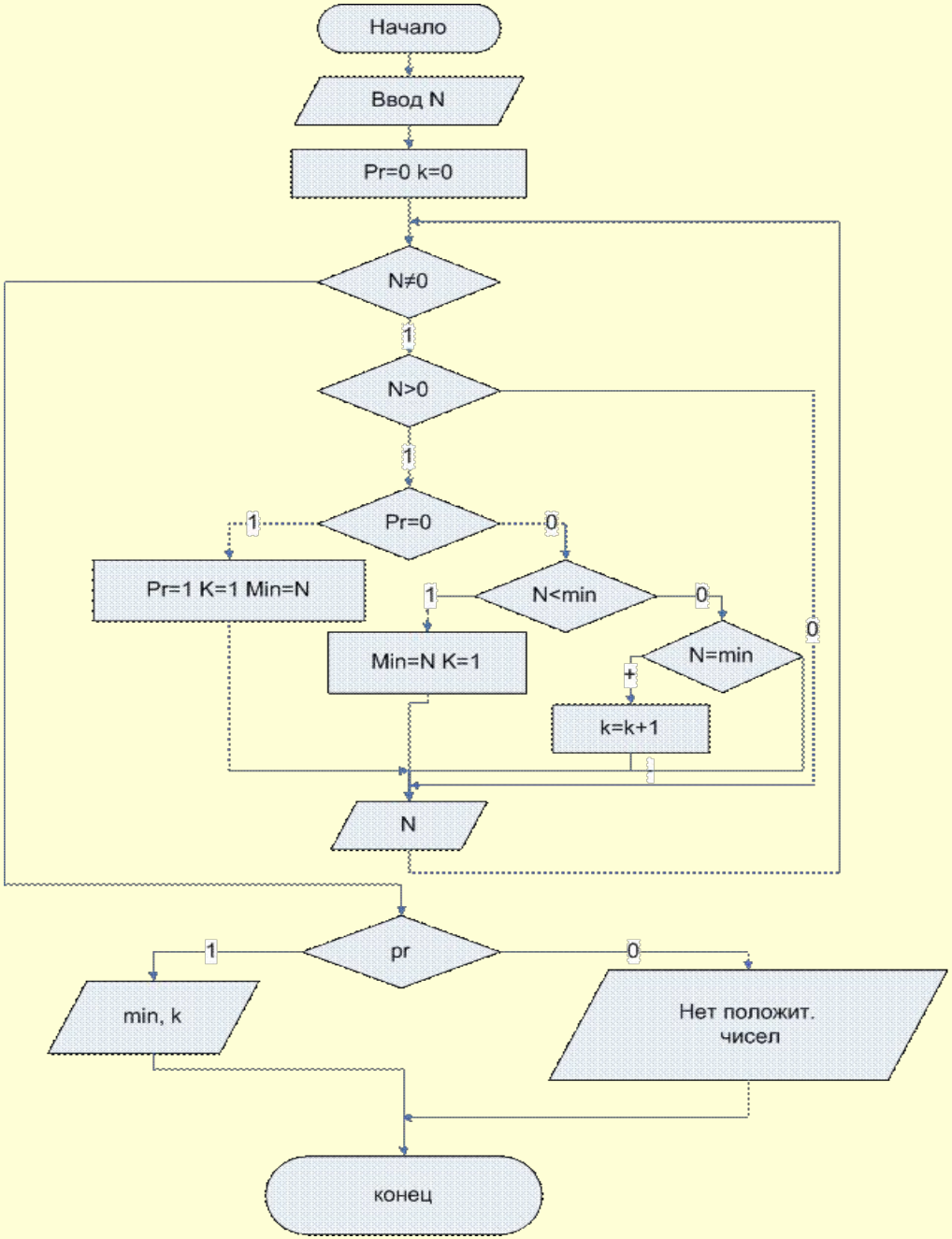
```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include  <conio.h>
#include  <math.h>

using namespace std;

int _tmain()
{
float xn, xk, dx, x, y, s, p;
printf("Vvedi xn,xk,dx");
scanf("%f%f%f",&xn,&xk,&dx);
for(s=0,p=1,x=xn;x<=xk;x+=dx)
{
y=2*exp(sin(3*x))*cos(4*x);
printf("X=%g\t Y=%g\n",x,y);
if (y>=0) s+=y;
else p*=y;
}
printf("S=%g\t P=%g\n",s,p);
getch();
}
```

Результаты


c:\Documents and Settings\zavkit\Рабочий стол\C++\s3\Debug\s3.exe
Vvedi xn,xk,dx
c:\Documents and Settings\zavkit\Рабочий стол\C++\s3\Debug\s3.exe
Vvedi xn,xk,dx1 2 0.1
X=1 Y=-1.50542
X=1.1 Y=-0.524965
X=1.2 Y=0.112422
X=1.3 Y=0.471045
X=1.4 Y=0.648826
X=1.5 Y=0.722507
X=1.6 Y=0.733553
X=1.7 Y=0.688925
X=1.8 Y=0.561794
X=1.9 Y=0.289729
S=4.2288 P=0.790295

Вводится последовательность целых чисел, 0 – конец последовательности. Найти минимальное среди положительных, если таких значений несколько, определить, сколько их.

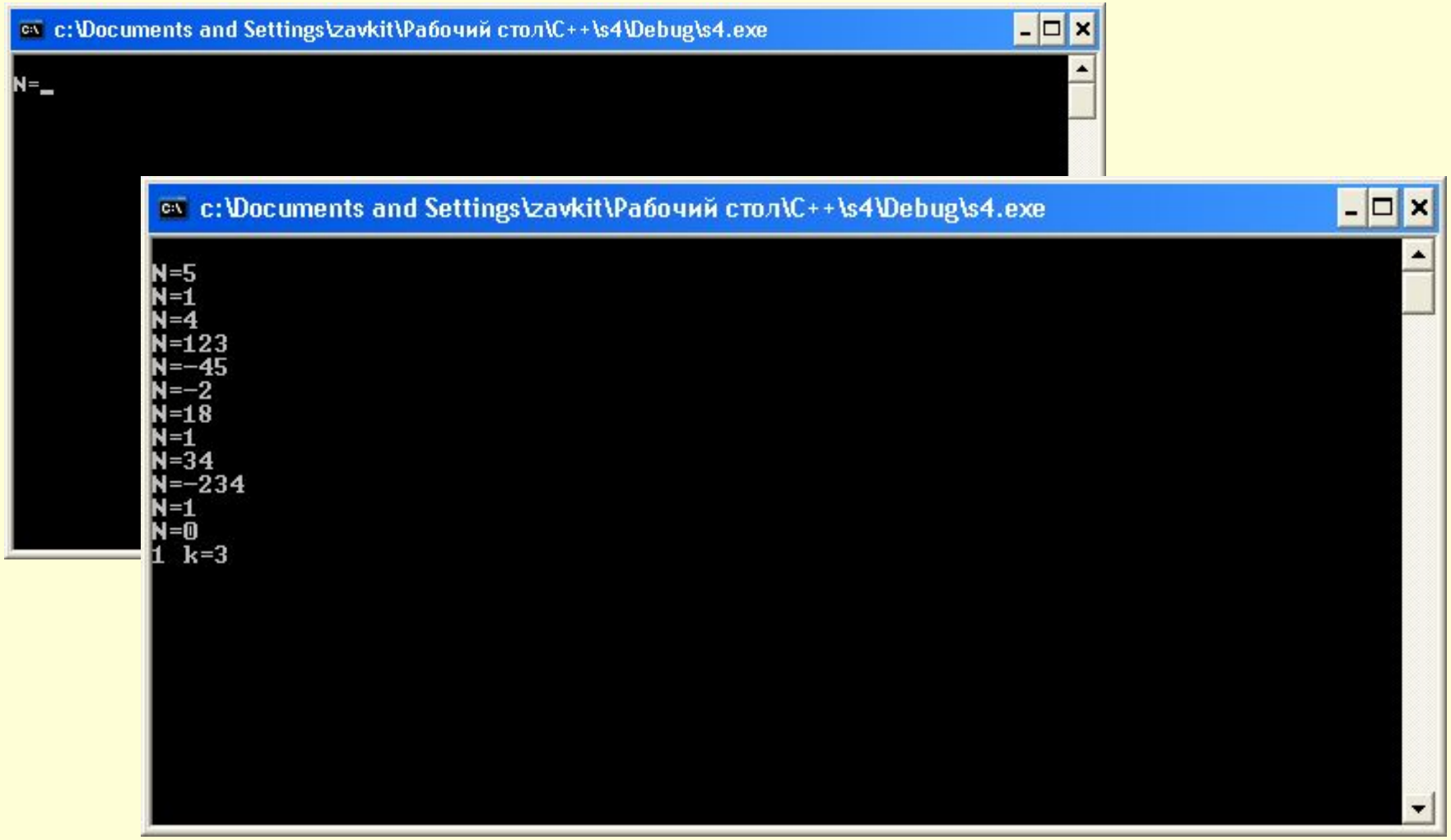
Блок - схема


Начало
Ввод N
Pr=0 k=0
N≠0
1
N>0
1
Pr=0
1
0
Pr=1 K=1 Min=N
1
N<min
0
Min=N K=1
N=min
+
k=k+1
0
N
pr
1
0
min, k
Нет положит. чисел
конец

Текст программы

```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <windows.h>
#include <conio.h>
using namespace std;

int main(int argc, char* argv[])
{ int N,pr,k,min;
cout<<"\nN=";
cin>>N;
for(pr=k=0;N!=0;cout<<"N=",cin>>N)
if(N>0)
if (pr==0)
{ pr=1; min=N; k=1; }
else
if (N<min) { min=N; k=1; }
else
if (N==min) k++;
if (pr)
{cout<<min;
cout<<"k=";
cout<<k<<endl;
}
else
cout<<"В последовательности
нет положительных чисел";
 getch();
}
```

Результаты

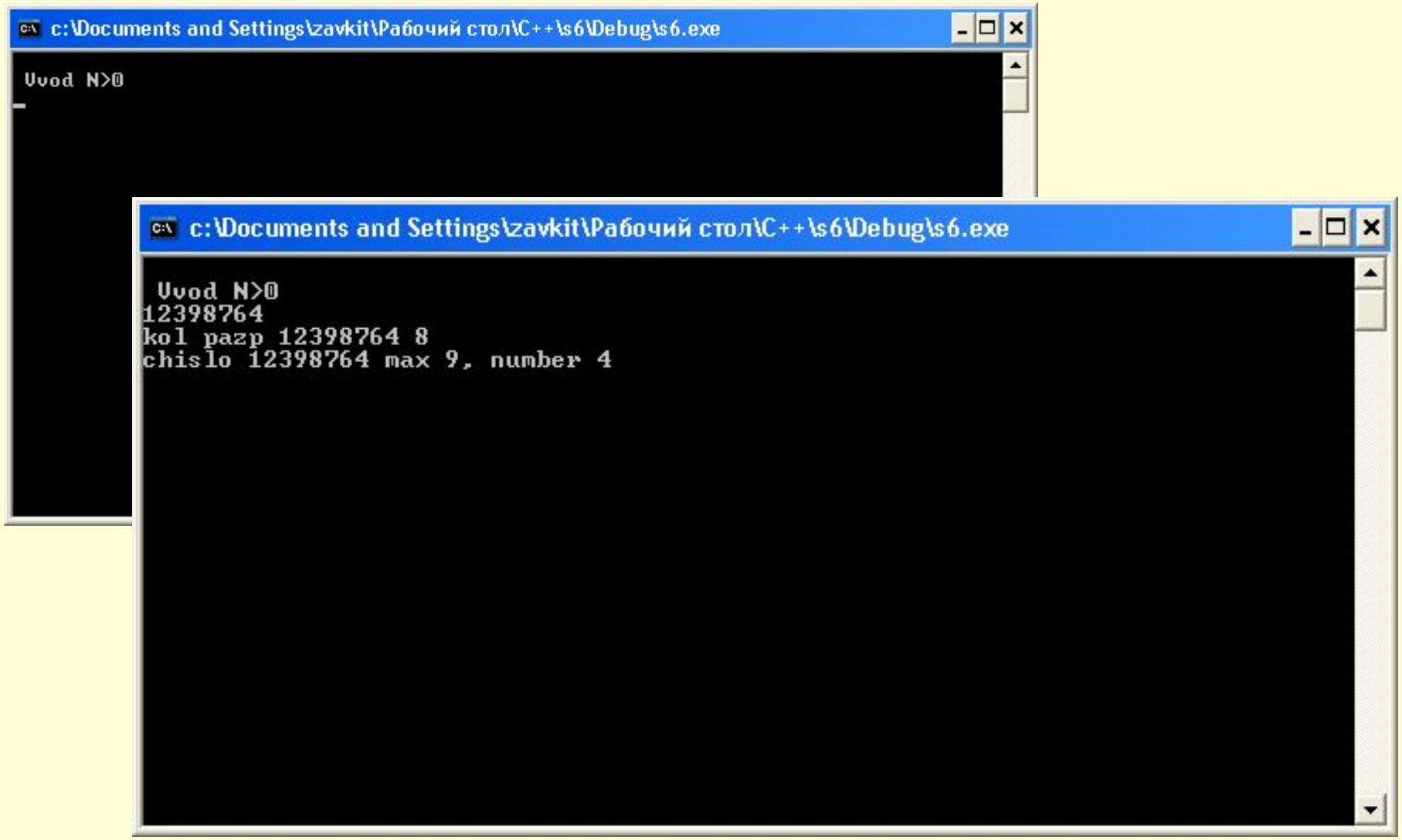

c:\Documents and Settings\zavkit\Рабочий стол\C++\s4\Debug\s4.exe
N=_
c:\Documents and Settings\zavkit\Рабочий стол\C++\s4\Debug\s4.exe
N=5
N=1
N=4
N=123
N=-45
N=-2
N=18
N=1
N=34
N=-234
N=1
N=0
1 k=3

Дано натуральное число N. Определить самую большую цифру и ее позицию в числе (N=573863, наибольшей является цифра 8, ее позиция – четвертая слева).

Текст программы

```
#include "stdafx.h"
#include <iostream>
#include <math.h>
#include <conio.h>

int _tmain()
{
long int N,M,kol=1;
int max,pos,i;
printf("\n Vvod N>0\n");
scanf("%ld",&N);
M=N;
while(M/10>0)
{
kol++;
M/=10;
}
printf("kol pazp %ld %ld \n",
N,kol);
for(M=N,max=-1, pos=1, i=kol; i>1;
i--)
{
if (M%10>max)
{
max=M%10;
pos=i;
}
M/=10;
}
printf("chislo %ld max %d, number
%d\n",
N,max,pos);
getch();
}
```

Результаты

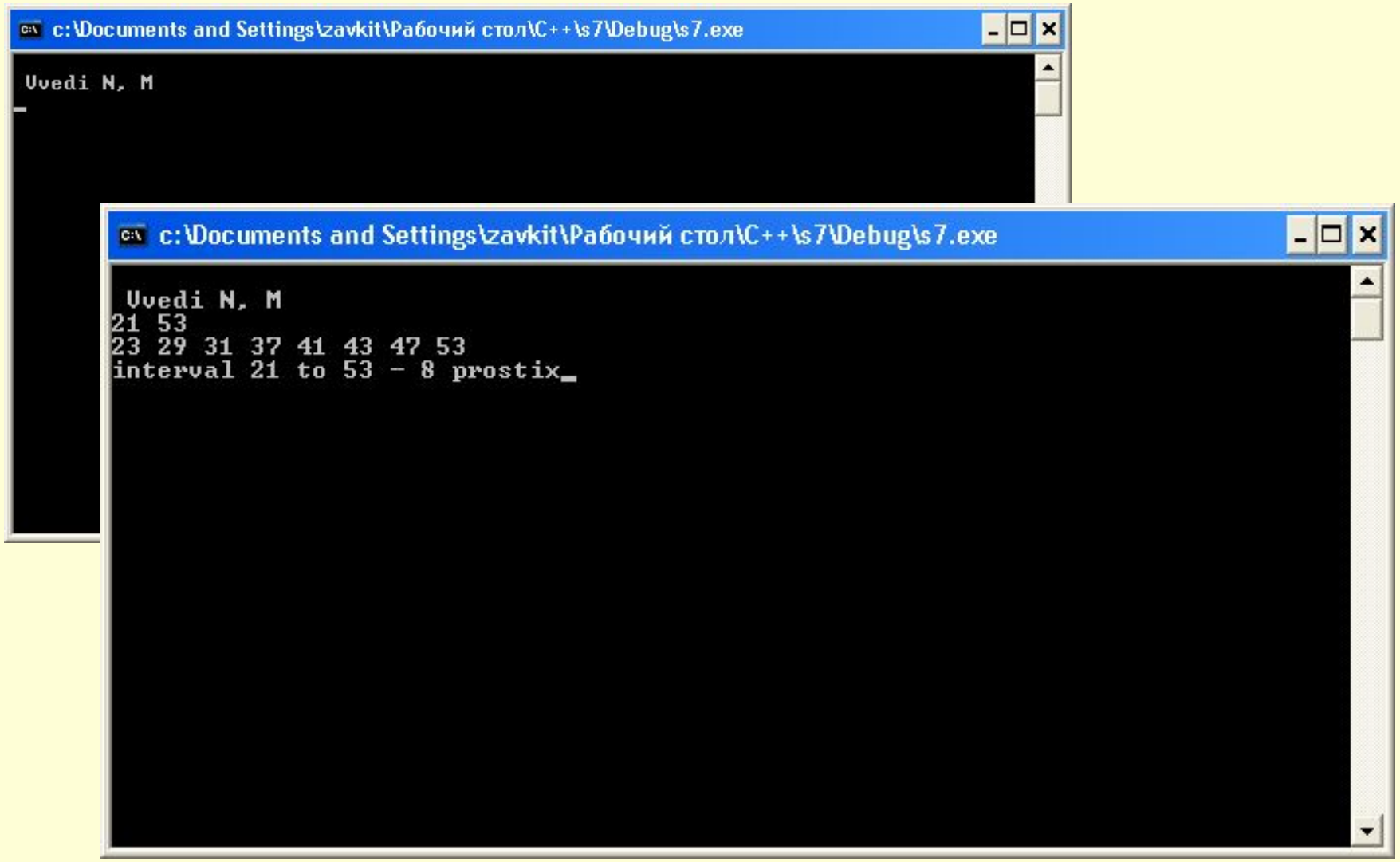

c:\Documents and Settings\zavkit\Рабочий стол\C++\s6\Debug\s6.exe
Vvod N>0
c:\Documents and Settings\zavkit\Рабочий стол\C++\s6\Debug\s6.exe
Vvod N>0
12398764
kol pazp 12398764 8
chislo 12398764 max 9, number 4

Определить количество простых чисел в интервале от N до M, где N и M – натуральные числа.

Текст программы

```
#include "stdafx.h"
#include <stdio.h>
#include <math.h>
#include <conio.h>

int _tmain()
{
unsigned int N,M,i,j,pr,k;
do
{
printf("\n Vvedi N, M\n");
scanf("%u%u",&N,&M);
}while(M<N);
for(k=0,i=N;i<=M;i++)
{
for (pr=1,j=2;j<i/2;j++)
if (i%j==0)
{
pr=0;
break;
}
if (pr==1)
{
printf("%u ",i);
k++;
}
}
if (k)
printf("\ninterval %u to %u - %u
prostix",N,M,k);
else
printf("\ninterval %u to %u -
no",N,M);
getch();
}
```

Результаты


c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
Uvedi N, M
c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
Uvedi N, M
21 53
23 29 31 37 41 43 47 53
interval 21 to 53 - 8 prostix

Использование функций

Подпрограмма – именованная, логически законченная группа операторов языка, которую можно вызвать для выполнения любое количество раз из различных мест программы. В языке **C/C++** подпрограммы реализованы в виде функций. Функция принимает параметры и возвращает единственное скалярное значение.
Функция должна быть описана перед своим использованием. Описание функции состоит из заголовка и тела функции.

```
Заголовок_функции
{
тело_функции
}
```

Использование функций

Заголовок функции имеет вид
type имя_функции ([список параметров])
type – тип возвращаемого функцией значения;
список параметров – список передаваемых в функцию величин, которые отделяются запятыми, каждому параметру должен предшествовать его тип;

В случае, если вызываемые функции идут до функции main, структура программы будет такой.

директивы компилятора

...

Тип_результата f1(Список_переменных)

{ Операторы }

Тип_результата f2(Список_переменных)

{ Операторы }

...

Тип_результата fn(Список_переменных)

{ Операторы}

int main(Список_переменных)

{

Операторы основной функции, среди которых могут операторы вызова функций f1, f2, ..., fn

}

Если вызываемые функции идут после функции main, структура программы будет такой (заголовки функций должны быть описаны до функции main()). Опережающие заголовки функций называют **прототипами** функций.

директивы компилятора

...
Тип_результата f1(Список_переменных);
Тип_результата f2(Список_переменных);

...
Тип_результата fn(Список_переменных);
int main(Список_переменных)
{Операторы основной функции, среди которых могут операторы вызова функций f1, f2, ..., fn }
Тип_результата f1(Список_переменных)
{ Операторы }
Тип_результата f2(Список_переменных)
{ Операторы }
...
Тип_результата fn(Список_переменных)
{ Операторы }

Для того, чтобы функция вернула какое-либо значение, в ней должен быть оператор
return значение;
Для вызова функции необходимо указать имя функции и в круглых скобках список передаваемых в функцию значений.

Передача параметров

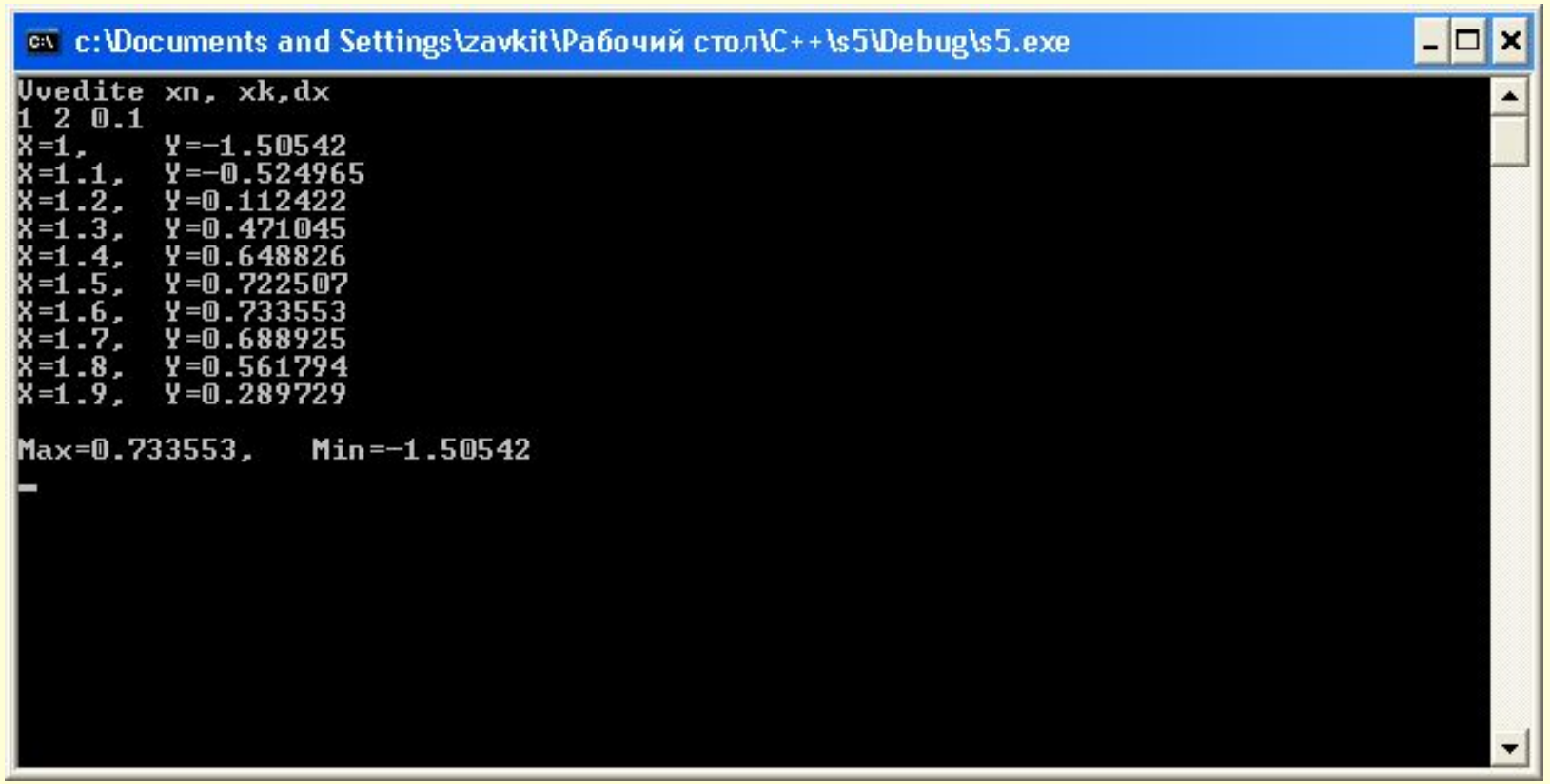
Параметры, указанные в заголовке функции, называются **формальными**. Параметры, передаваемые в функцию, называются **фактическими**.
При обращении к функции фактические параметры передают свое значение формальным и больше не изменяются. **Типы, количество и порядок следования формальных и фактических параметров должны совпадать.** С помощью оператора ***return*** из функции возвращается единственное значение.

Составить таблицу значений функции $y=2e^{\sin(3x)}\cos(4x)$ на отрезке [xn; xk] с шагом dx.
Найти сумму положительных y и произведение отрицательных y.
Расчет ***y*** оформить в виде функции.

Текст программы

```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include  <conio.h>
#include  <math.h>

using namespace std;
float f(float x, float a, float b, float
c);
int _tmain()
{
float xn, xk, dx, x, y, max,min;
int k=0;
cout<<"Vvedite xn, xk,dx\n";
cin>>xn>>xk>>dx;
for(x=xn;x<=xk;x+=dx)
{
        y=f(x,2,3,4);
cout<<"X="<<x<<",\tY="<<y<<endl;
        k++;        if (k==1)
        {       max=y;
                min=y;        }
        else
        {   if (y>max) max=y;
            if (y<min) min=y;        }}
cout<<endl<<"Max="<<max<<",
\tMin="<<min<<endl;
getch();}
float f(float x, float a, float b, float c)
{
        float y;
        y=a*exp(sin(b*x))*cos(c*x);
        return y;
}
```

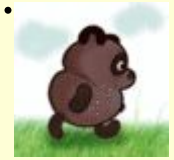
Результаты


c:\Documents and Settings\zavkit\Рабочий стол\C++\s5\Debug\s5.exe
Uvedite xn, xk,dx
1 2 0.1
X=1, Y=-1.50542
X=1.1, Y=-0.524965
X=1.2, Y=0.112422
X=1.3, Y=0.471045
X=1.4, Y=0.648826
X=1.5, Y=0.722507
X=1.6, Y=0.733553
X=1.7, Y=0.688925
X=1.8, Y=0.561794
X=1.9, Y=0.289729
Max=0.733553, Min=-1.50542

Пример

Вводится последовательность целых чисел, 0 – конец последовательности. Найти минимальное среди простых чисел и максимальное, среди чисел, не являющихся простыми.

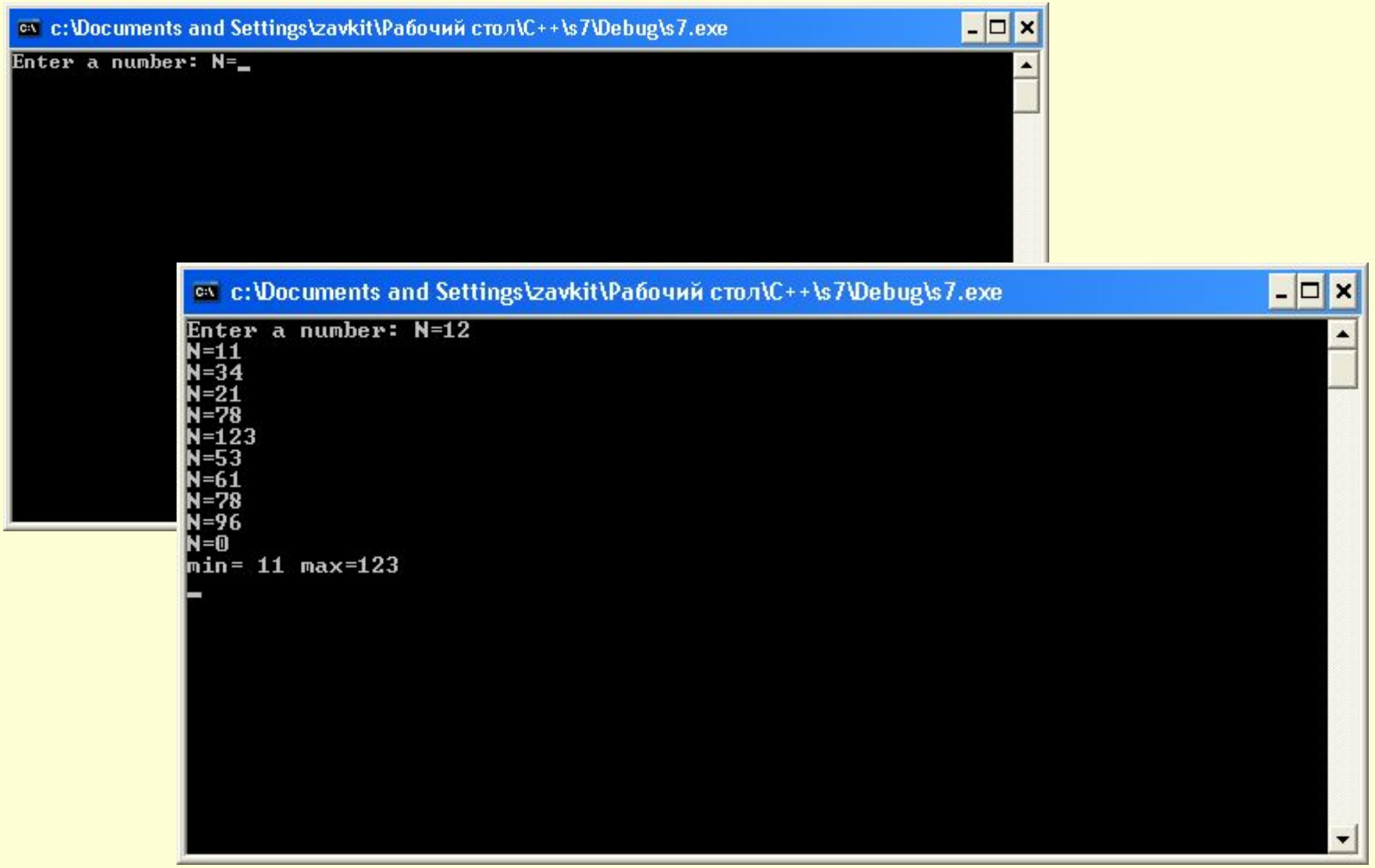
Целое число называется **простым**, если оно делится нацело только на самого себя и единицу. Алгоритм проверки, что число N является простым состоит в следующем: если разделим N без остатка хотя бы на одно число в диапазоне от 2 до N пополам, то число не является простым. Если не найдем ни одного делителя числа, число N – простое. Проверку является ли число N простым оформим в виде отдельной функции с именем **prostoe**. Входным параметром функции будет целое число N, функция будет возвращать значение 1, если число простое и 0 – в противном случае.



Текст программы

```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include  <conio.h>
using namespace std;
int prostoe(int N)
{ int i,pr;
if (N<1) pr=0;
    else
for(pr=1,i=2;i<=N/2;i++)
if (N%i==0) {pr=0;break;}
return pr; }
int _tmain()
{ int kp=0,knp=0,min,max,N;
cout<< "Enter a number: ";
for (cout<<"N=", cin>>N; N!=0;
cout<<"N=", cin>>N)
    if (prostoe(N))
{       kp++;
      if (kp==1) min=N;
      else if (N<min) min=N;
   }
   else
   {       knp++;
      if (knp==1) max=N;
      else if (N>max) max=N;
   }
if (kp>0) cout<<"min= "<<min<<"\t";
else cout<<"Net prostih";
if (knp>0)
cout<<"max="<<max<<endl;
else cout<<"Net ne prostih";
getch();
}
```

Результаты


c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
Enter a number: N=
c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
Enter a number: N=12
N=11
N=34
N=21
N=78
N=123
N=53
N=61
N=78
N=96
N=0
min= 11 max=123

Пример

Вводится последовательность из N целых чисел, найти среднее арифметическое совершенных чисел и среднее геометрическое простых чисел.

Число называется **совершенным**, если сумма всех делителей, меньших его самого равна самому числу.

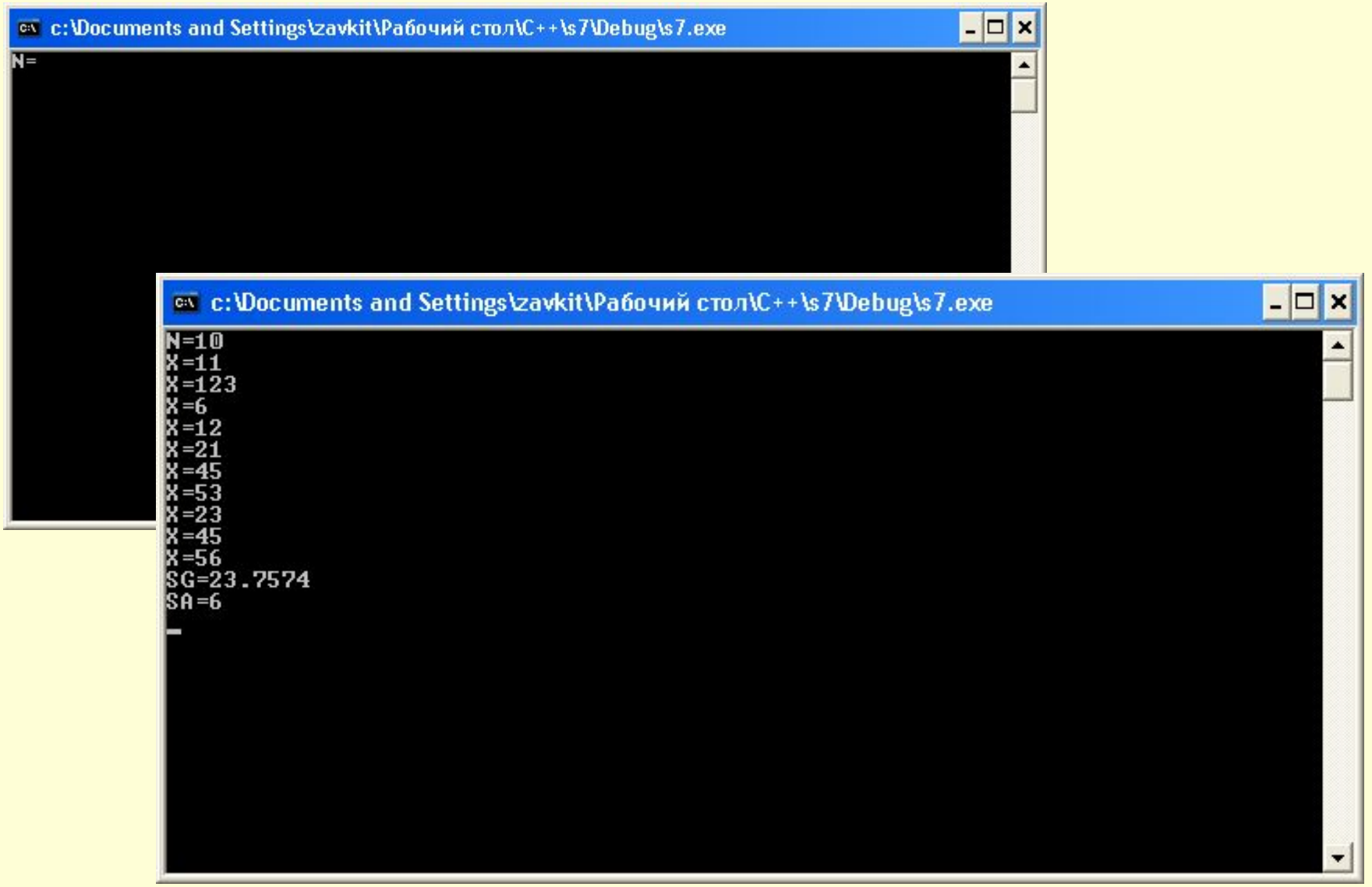
При решении этой задачи понадобятся две функции:

- функция **prostoe**,
- функция **soversh**, которая определяет является ли число совершенным; входным параметром функции будет целое число N, функция будет возвращать значение 1, если число совершенным и 0 – в противном случае.

Текст программы

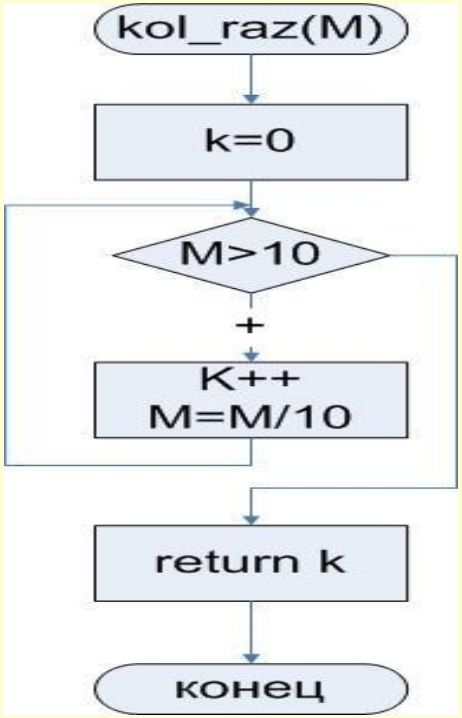
```
#include "stdafx.h"
#include <math.h>
#include <iostream>
#include <fstream>
#include  <conio.h>
using namespace std;
int prostoe(int N)
{ int i,pr;
if (N<1) pr=0;      else
for(pr=1,i=2;i<=N/2;i++)
if (N%i==0) {pr=0;break;}
return pr; }
int soversh(int N)
{ int i,S;
if (N<1) return 0;
     else for(S=0,i=1;i<=N/2;i++)
if (N%i==0) S+=i;
if (S==N) return 1; else return 0; }
int _tmain()
{ int i,N,X,S,kp,ks;  int P;
cout<<"N=“; cin>>N;
for(kp=ks=S=0,P=1,i=1;i<=N;i++)
{     cout<<"X=“;     cin>> X;
     if (prostoe(X))
     { kp++; P*=X; }
     if (soversh(X))
     { ks++;S+=X;  } }
if (kp>0)
cout<<"SG="<<powf((float)P,(float)1/
kp)<<endl;
else
cout<<"Net prostih";
if (ks>0)
 cout<<"SA="<<(float)S/ks<<endl;
else cout<<"Net soversh";
getch(); }
```

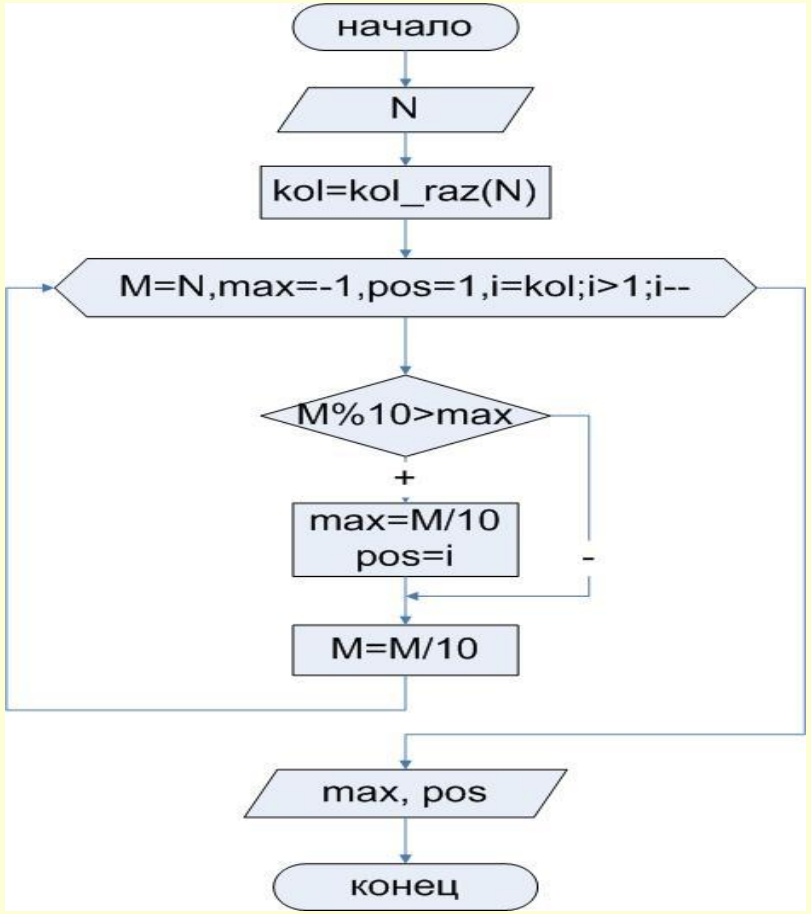
Результаты


c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
N=
c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
N=10
X=11
X=123
X=6
X=12
X=21
X=45
X=53
X=23
X=45
X=56
SG=23.7574
SA=6

Пример

Дано натуральное число N. Определить самую большую цифру и ее позицию в числе (N=573863, наибольшей является цифра 8, ее позиция – четвертая слева)

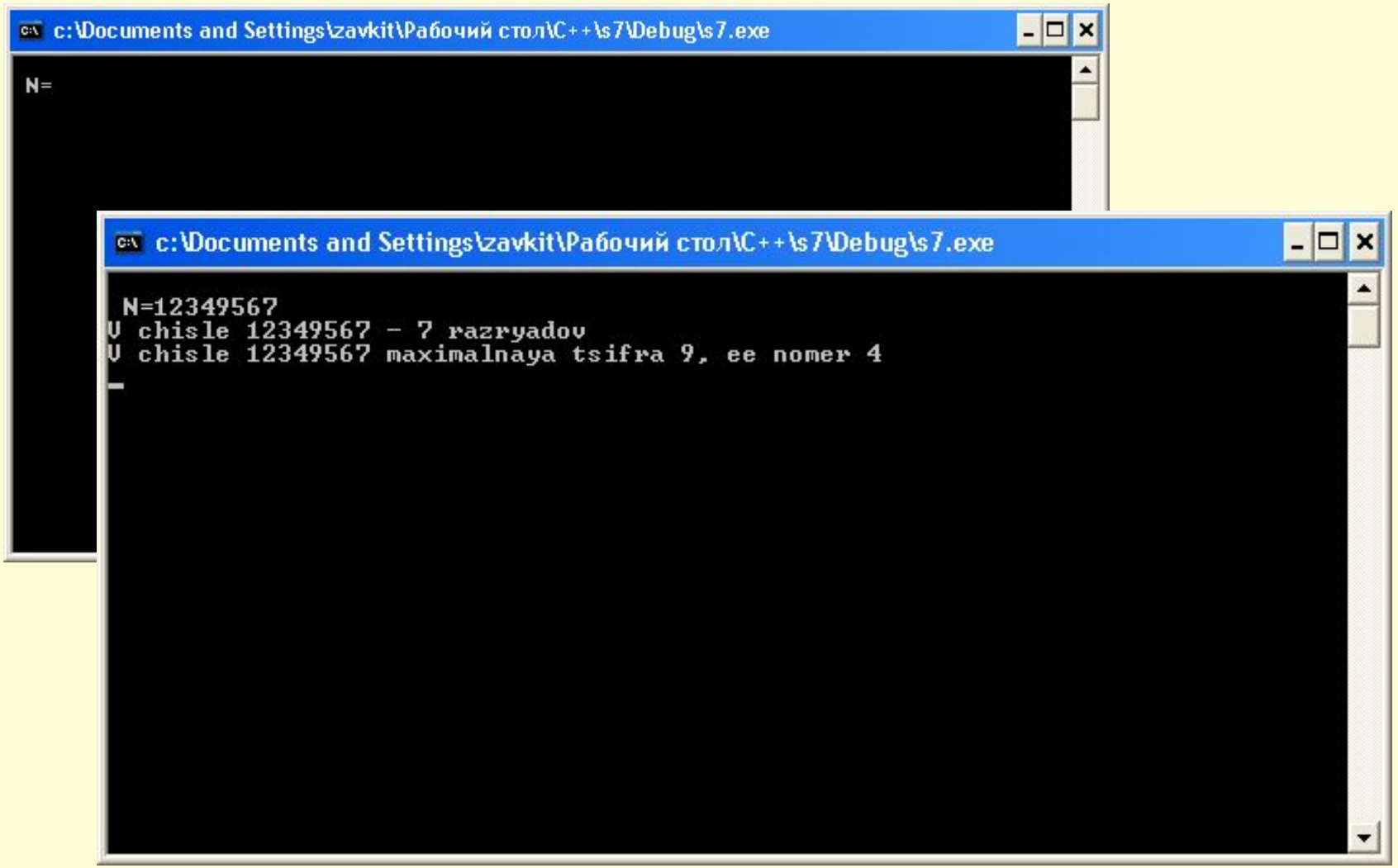




Текст программы

```
#include "stdafx.h"
#include <math.h>
#include <iostream>
#include <fstream>
#include  <conio.h>
using namespace std;
int kol_raz(int M)
{ int k=0; while(M/10>0)
{ k++; M/=10;}return k;}
int _tmain()
{ long int N,M,kol=1;
int max,pos,i;
printf("\n N="); // Ввод числа N.
scanf("%ld",&N);
// Вычисление количества позиций в числе (kol).
kol=kol_raz(N);
printf("V chisle %ld - %ld
razryadov\n“, N,kol);
// Вычисление максим. цифры в числе, и ее номера.
for(M=N, max=-1, pos=1, i=kol;i>1;i--)
{ if (M%10>max)
{
max=M%10;
pos=i;
}
M/=10; }
// Вывод на экран максимальной цифры
// в числе, и ее номера.
printf("V chisle %ld maximalnaya
tsifra %d, ee nomer
%d\n",N,max,pos);
getch();
}
```

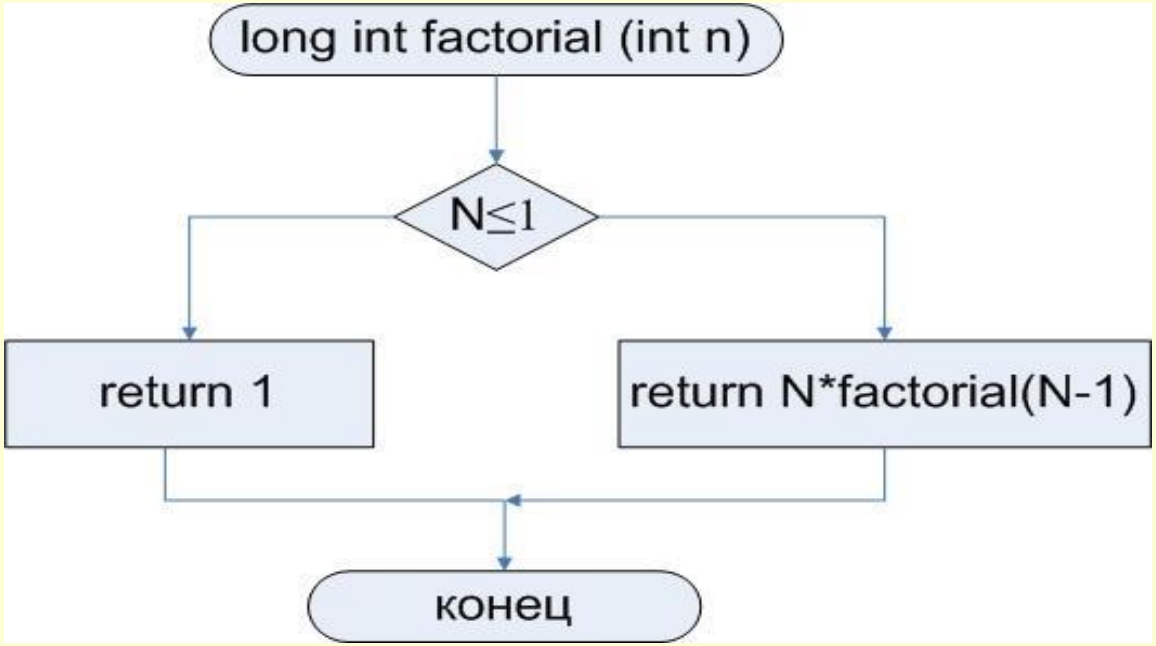
Результаты


c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
N=
c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
N=12349567
V chisle 12349567 - 7 razryadov
V chisle 12349567 maximalnaya tsifra 9, ee nomer 4

Рекурсивные функции

Под рекурсией в программировании понимается вызов функции из тела ее самой. В рекурсивных алгоритмах функция вызывает саму себя до выполнения какого-то условия.

long int factoial(int n)
предназначена для вычисления факториала числа **n**.

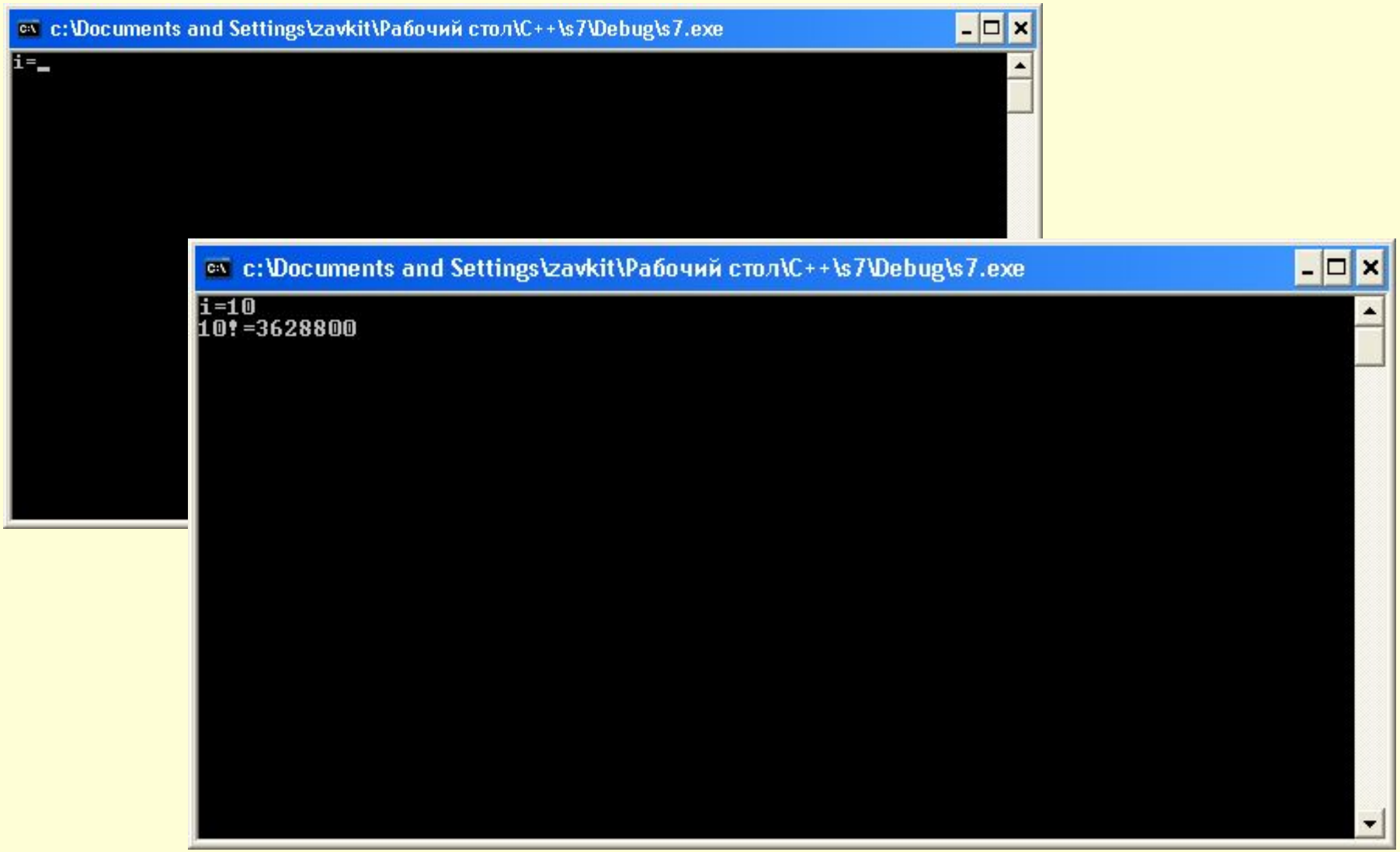


Текст программы

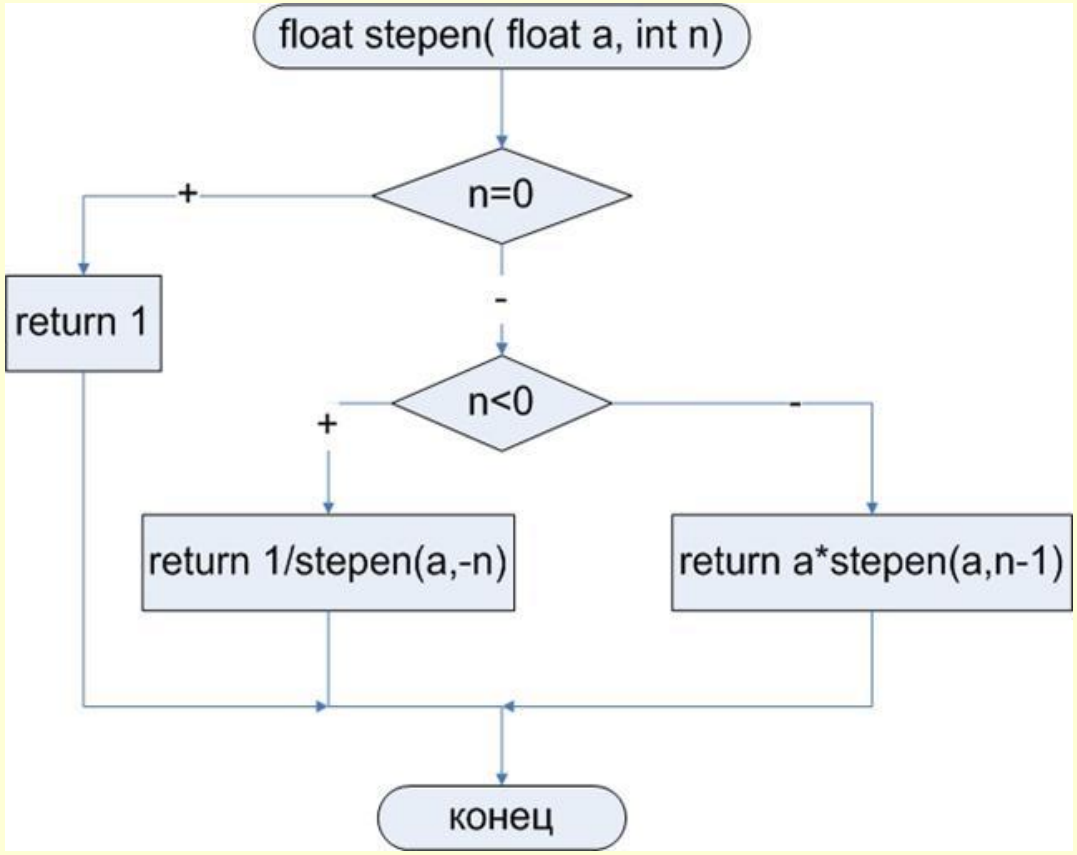
```
#include "stdafx.h"
#include <math.h>
#include <iostream>
#include <fstream>
#include  <conio.h>

using namespace std;

long int factorial(int n)
{
if (n<=1)
    return(n);
else
    return(n*factorial(n-1));
}
```

```
int main()
{
int i;
long int f;
cout<<"i=";
cin>>i;
f=factorial(i);
cout<<i<<"!="<<f<<endl;
getch();
}
```

Результаты


c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
i=
c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
i=10
10!=3628800

float stepen(float a, int n) предназначена для возведения числа **a** в степень **n**.



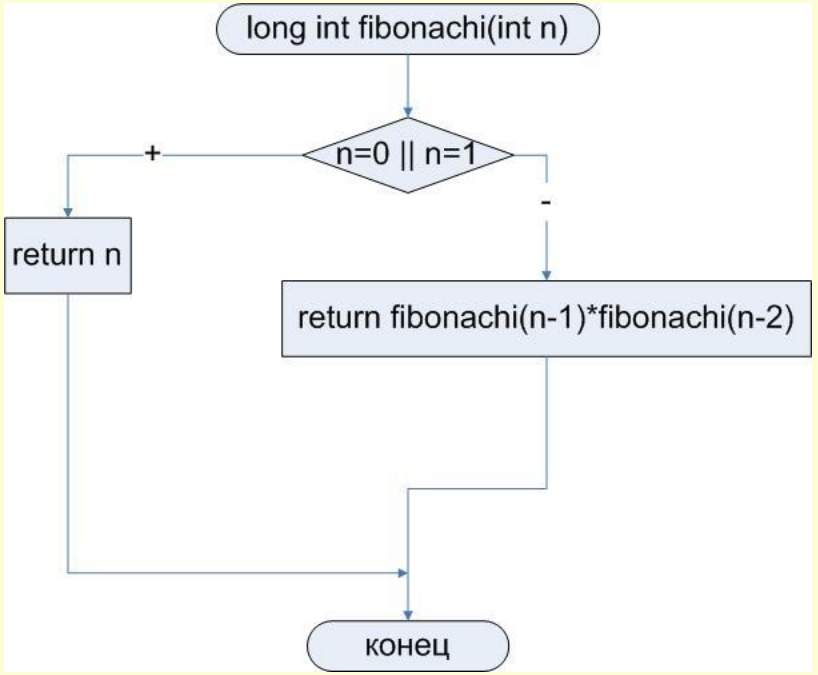
Текст программы

```
#include "stdafx.h"
#include <math.h>
#include <iostream>
#include <fstream>
#include  <conio.h>

using namespace std;

float stepen(float a, int n)
{
if (n==0)
   return(1);
else
    if (n<0)
       return(1/stepen(a,-n));
    else
        return(a*stepen(a,n-1));
}
```

```
int main()
{
int i;
float s,b;
long int f;
cout<<"b=";
cin>>b;
cout<<"i=";
cin>>i;
s=stepen(b,i);
cout<<"s="<<s<<endl;
getch();
}
```

long int fibonachi(int n) предназначена для вычисления n-го числа Фибоначчи.

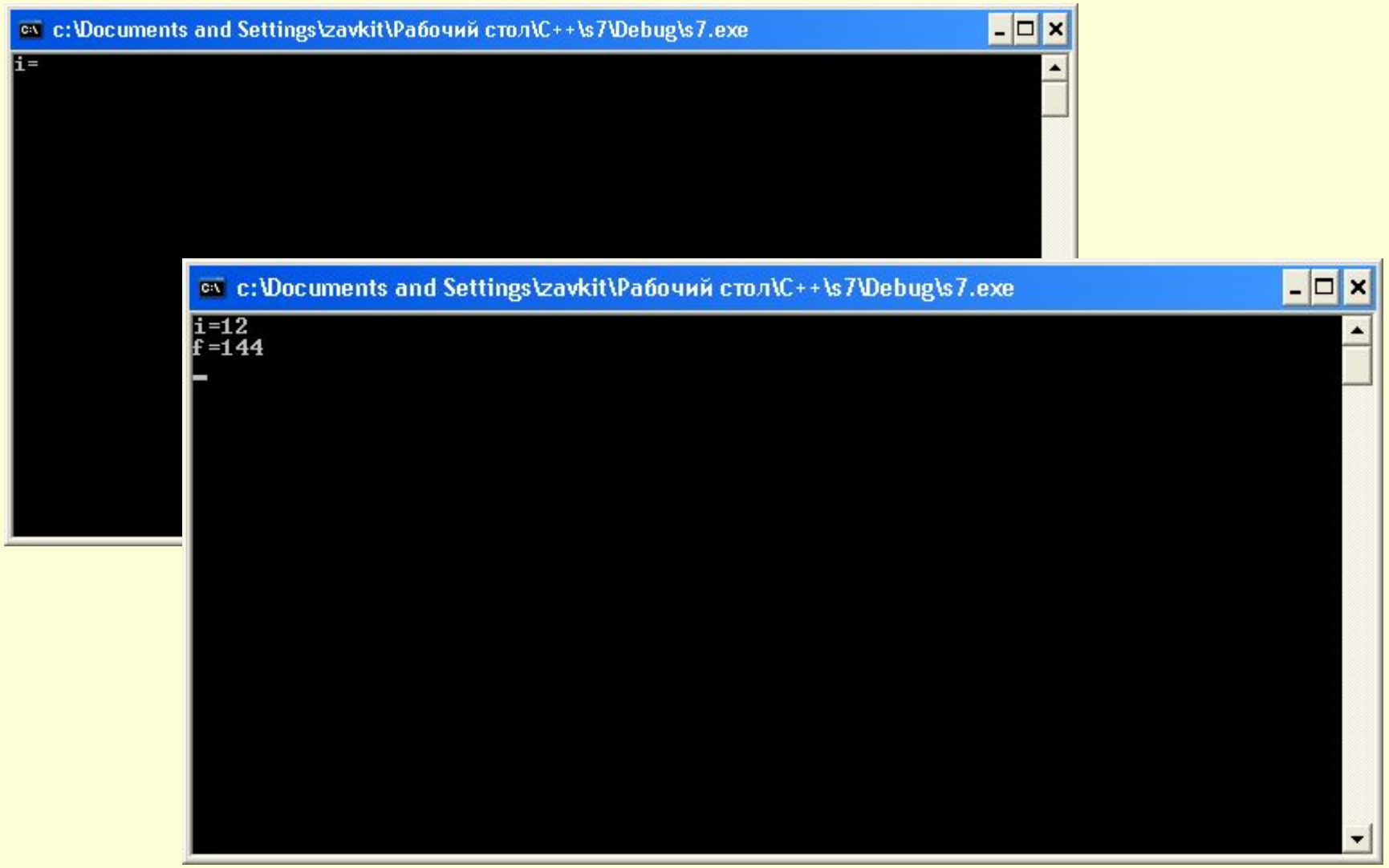
Если нулевой элемент последовательности равен 0, первый – 1, а каждый последующий равен сумме двух предыдущих, то это последовательность чисел Фибоначчи (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...).



Текст программы

```
#include "stdafx.h"
#include <math.h>
#include <iostream>
#include <fstream>
#include  <conio.h>

using namespace std;

long int fibonachi(unsigned int n)
{
if ((n==0)||(n==1)) return(n);
else
return(fibonachi(n-1)+fibonachi(n-2
));
}
int main(int argc, char* argv[])
{
int i;
long int f;
cout<<"i=";
cin>>i;
f=fibonachi(i);
cout<<"f="<<f<<endl;
getch();
}
```

Результаты


c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
i=
c:\Documents and Settings\zavkit\Рабочий стол\C++\s7\Debug\s7.exe
i=12
f=144

Область видимости переменных в функциях C/C++, расширение области видимости переменных

Рассмотрим особенности использования локальных и глобальных переменных в программах на C++:

1. Область видимости и использования локальной переменной ограничена функцией, где она определена.
2. Глобальные переменные объявляются вне любых функций и их областью видимостью является весь файл.
3. Одно и тоже имя может использоваться при определении глобальной и локальной переменной. В этом случае в функции, где определена локальная переменная действует локальное описание, вне этой функции «работает» глобальное описание.

Из функции, где действует локальное описание переменной можно обратиться к глобальной переменной с таким же именем, используя оператор расширения области видимости **::переменная**.

Рассмотрим это на примере

```
float pr=100.678; int prostoe (int n) { int pr=1,i;
if (n<0) pr=0; else      for (i=2;i<=n/2;i++)
    if (n%i==0){pr=0;break;}
// Вывод локальной переменной
cout<<"local pr="<<pr<<endl;
// Вывод глобальной переменной
cout<<"global pr="<<::pr<<endl;
return pr; }
int main()
{ int g;
cout<<"g=“; cin>>g;
if (prostoe(g)) cout<<"g – prostoe";
else cout<<"g – ne prostoe";
getch(); }
```

Результаты работы программы

g=7
local pr=1
global pr=100.678
g - prostoe
Press any key to continue

Перегрузка и шаблоны функций

Язык C++ позволяет связать с одним и тем же именем функции различные определения, т. е. возможно существование нескольких функций с одним и тем же именем. У этих функций может быть разное количество параметров или разные типы параметров. Создание двух или более функций с одним и тем же именем называется **перегрузкой имени функции**. Перегруженные функции следует создавать, когда одно и то же действие следует выполнить над разными типами входных данных, а иногда одна и та же функция над разными типами входных данных выполняется с помощью разных алгоритмов.

Перегрузка и шаблоны функций

Функция возведения в степень неопределена при 0^0 и при возведении отрицательного x в дробную степень

$$n = \frac{k}{m}$$

в случае четного m. Пусть наша функция в этих случаях будет возвращать 0.

```
#include "stdafx.h"
#include <iostream.h>
#include <math.h>
float pow(float a, int k, int m)
{
cout<<"function 1\t";
if (a==0) return (0);
else
    if (k==0) return(1);
    else
    if (a>0)
     return(exp((float)k/m*log(a)));
    else
        if (m%2!=0)
return (-(exp((float)k/m*log(-a))));
}
float pow(float a, int n)
{
if (a==0)
{cout<<"function 2\t";return (0);}
else
if (n==0)
{cout<<"function 2\t";return (1);}
else
    if (n<0) return(1/pow(a,-n));
    else
        return(a*pow(a,n-1));
}
int pow(int a, int n)
{
if (a==0)
 {cout<<"function 3\t";return (0);}
else
if (n==0)
{cout<<"function 3\t";return (1);}
else
    if (n<0) return(1/pow(a,-n));
    else
        return(a*pow(a,n-1));
}
```

```
int main()
{
float a;
int k,n,m;
cout<<"a=";
cin>>a;
cout<<"k=";
cin>>k;
cout<<"s="<<pow(a,k)<<endl;
cout<<"s="<<pow((int)a,k)<<endl;
cout<<"a=";
cin>>a;
cout<<"k=";
cin>>k;
cout<<"m=";
cin>>m;
cout<<"s="<<pow(a,k,m)<<endl;
return 0;
}
```

Результаты работы программы

```
a=5.2
k=3
function 2      s=140.608
function 3      s=125
a=-8
k=1
m=3
function 1      s=-2
Press any key to continue
```

Если перегрузку можно применять при использовании различных алгоритмов решения задачи при различных типах исходных данных и просто при различных типах исходных данных, то при использовании различных типов исходных данных можно применять **шаблоны**.

Шаблон – это особый вид функций, который начинается со служебного слова template, за которым в угловых скобках (<>) следует список используемых в функции типов данных. Каждый тип предваряется служебным словом class. Можно сказать, что в случае шаблона в качестве параметров выступают не только переменные, но их типы.

Рассмотрим пример шаблона поиска наименьшего из четырех чисел.

```
#include "stdafx.h"
#include <iostream.h>
//Определяем абстрактный тип данных с
// помощью служебного слова Type
template <class Type>
// Определяем функцию с использованием
// типа данных Type
Type minimum(Type a, Type b, Type c, Type d)
{
Type min=a;
if (b<min) min=b;
if (c<min) min=c;
if (d<min) min=d;
return min;
}
int main()
{
int ia,ib,ic,id,mini;
float ra,rb,rc,rd,minr;
cout<<"Vvod 4 thelih chisla\t";
cin>>ia>>ib>>ic>>id;
//Вызов функции minimum, в которую
// передаем 4 целых значениями
mini=minimum(ia,ib,ic,id);
cout<<"\n"<<mini<<"\n";
cout<<"Vvod 4 vecshestvenih chisla\t";
cin>>ra>>rb>>rc>>rd;
//Вызов функции minimum, в которую
// передаем 4 вещественных значениями
minr=minimum(ra,rb,rc,rd);
cout<<"\n"<<minr<<"\n";
return 0;
}
```

Использование значений формальных параметров по умолчанию

В C++ существует возможность задать значение некоторых формальных параметров по умолчанию, если такой параметр будет отсутствовать в вызове функции, то будет работать значение по умолчанию. Формальные параметры со значениями по умолчанию должны быть самыми последними в списке.

Использование значений формальных параметров по умолчанию

```
float stepen(float a, int n=3)
{    if (n==0) return(1);
   else
       if (n<0) return(1/stepen(a,-n));
       else
          return(a*stepen(a,n-1)); }
int main()
{     int a;
   long int f;
   cout<<"a=“;    cin>>a;
   f=stepen(a,5);
   cout<<"f="<<f<<endl;
   f=stepen(a);
   cout<<"f="<<f<<endl;
   return 0; }
```