Сколько раз выполняется цикл?

```
1)   a=1;
     for( i=1; i<=3; i++ ) a=a+1;
```

```
2)  a=1;
    for( i=3; i<=1; i++ ) a=a+1;
```

```
3)   a=1;
     for( i=1; i<=3; i-- ) a=a+1;
```

```
4)  a=1;
    for( i=3; i>=1; i-- ) a=a+1;
```

Сколько раз выполняется цикл?

```
5)   a = 4; b = 6;
      while ( a < b ) a = a + 1;
```

```
6)   a= 4; b= 6;
     while ( a < b ) a = a + b;
```

```
7)    a= 4; b= 6;
      while ( a > b ) a ++;
```

```
8)    a= 4; b= 6;
      while ( a < b ) b = a - b;
```

```
9)    a= 4; b= 6;
      while ( a < b ) a --;
```

Какая задача решается в этом фрагменте программы?

```
10)
n = 2;
for( k=1; k<=10; k++ )
   {
   cout << n << endl;
   n *= 2;
   }
```

Функции

Лекция 5

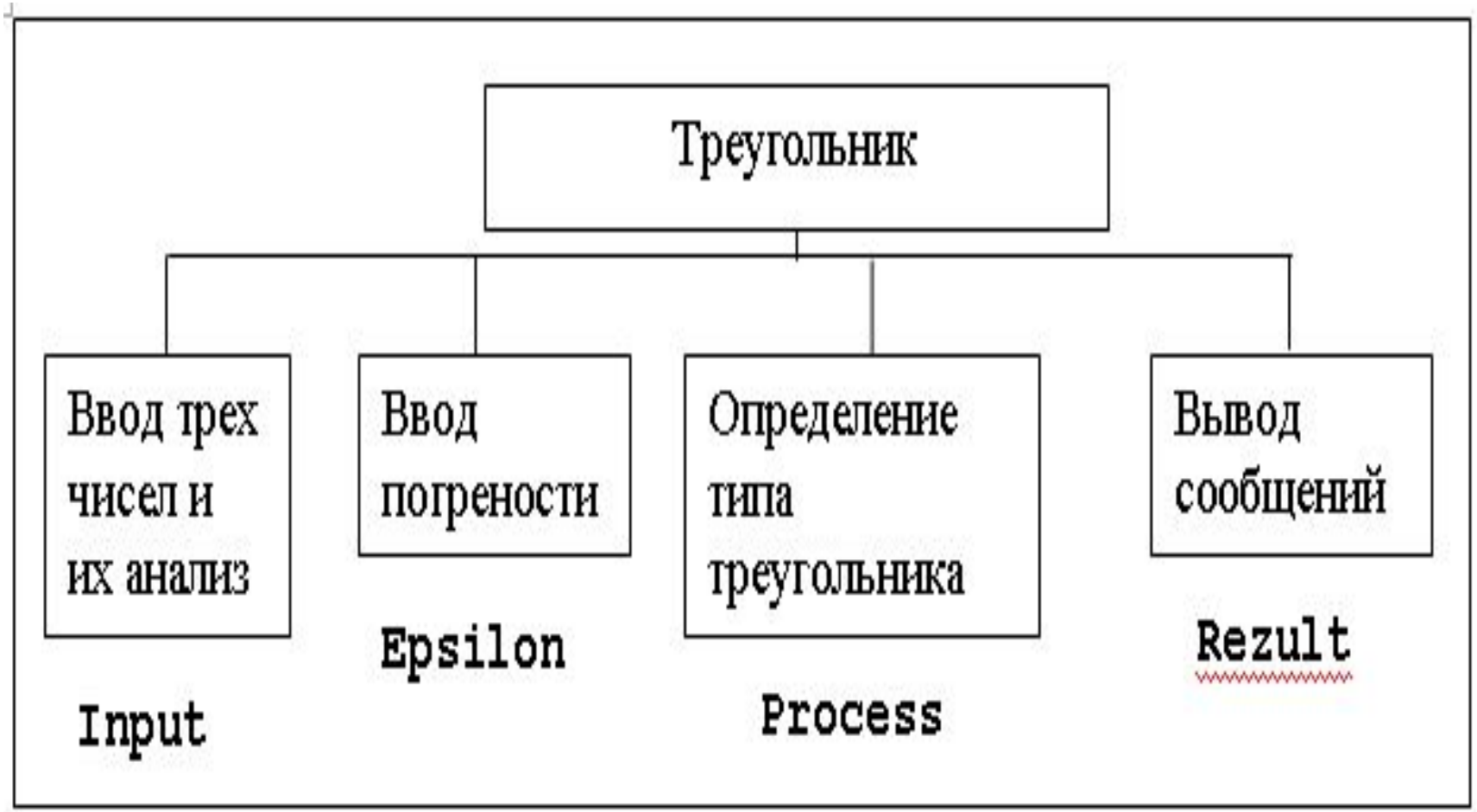
- Деление программы на функции является базовым принципом структурного программирования.

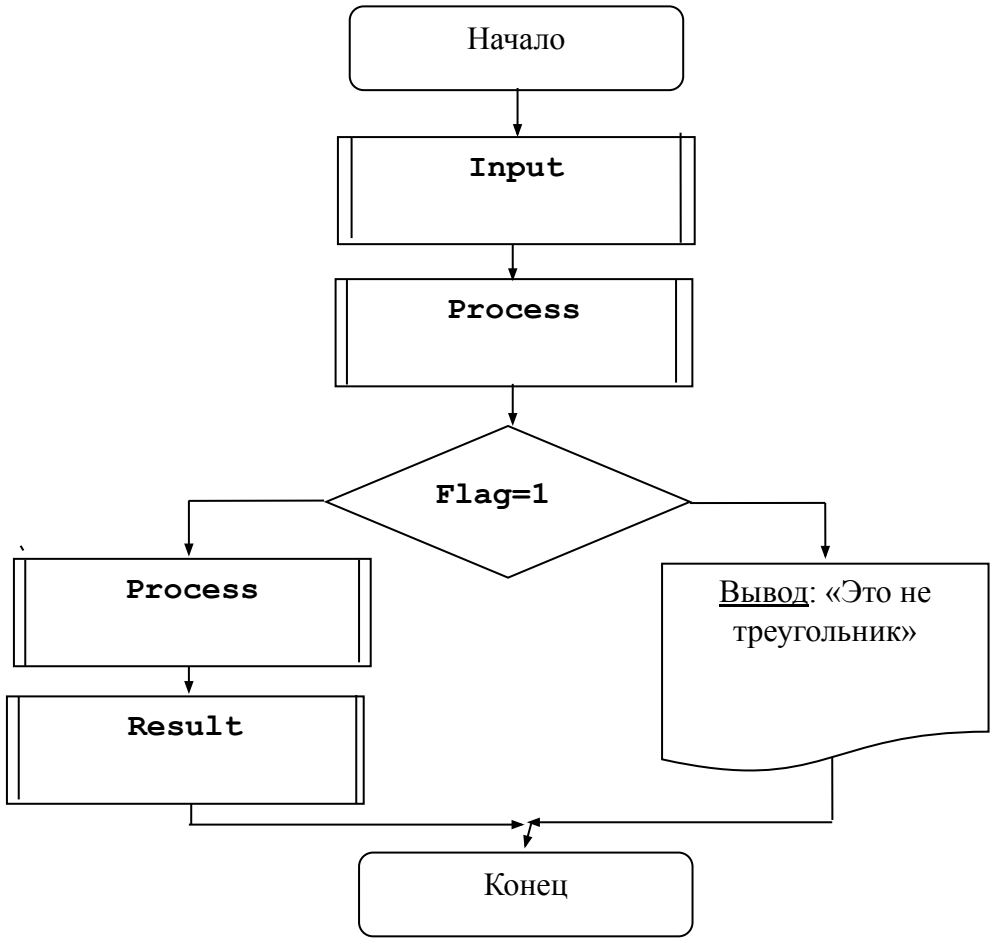
Основные свойства и достоинства структурного программирования

- Преодоление барьера сложности программ.
- Возможность демонстрации правильности программ на различных этапах решения.
- Наглядность.
- Простота модификации.

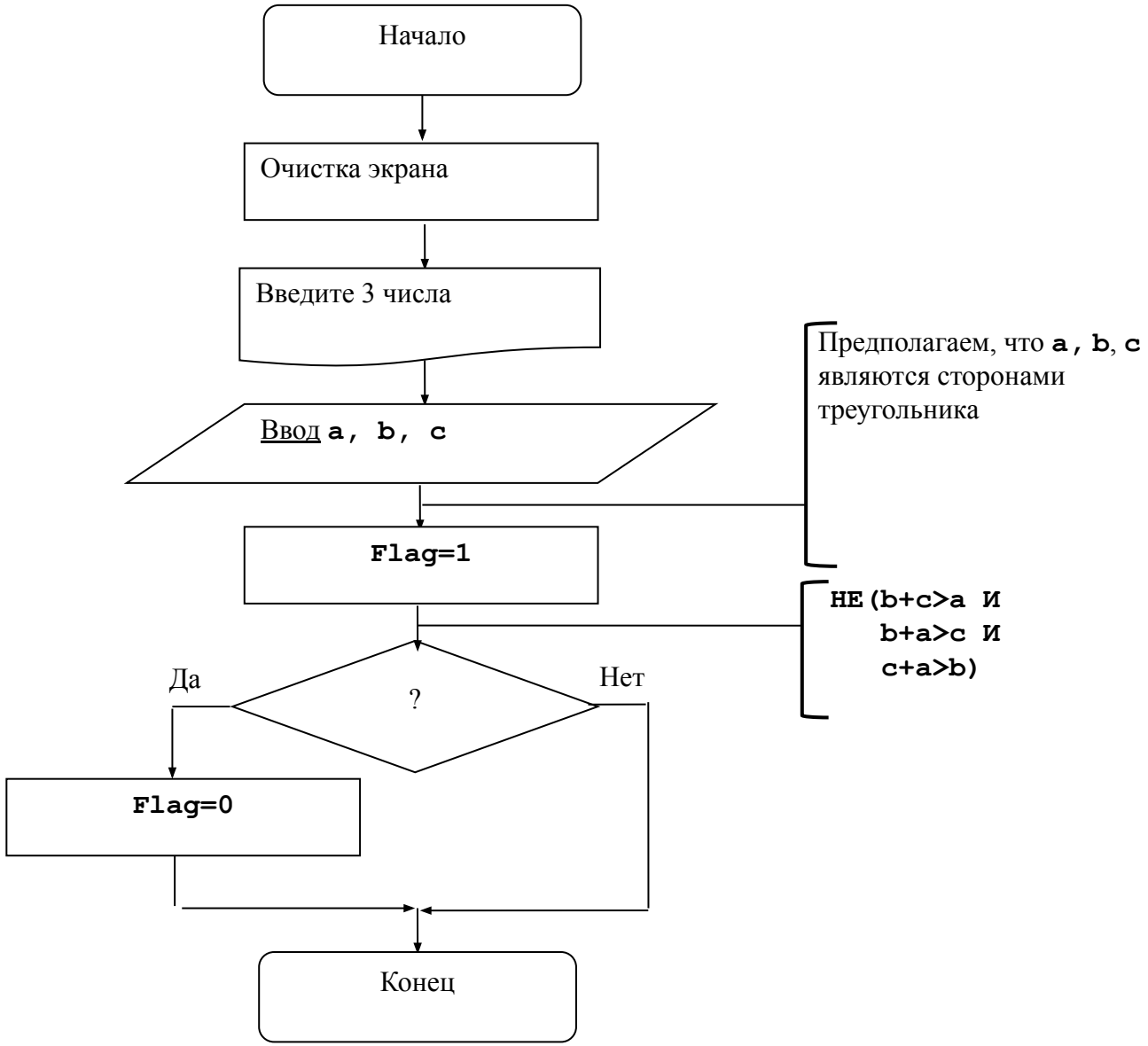
Задача

На основе трех действительных чисел определить вид треугольника.



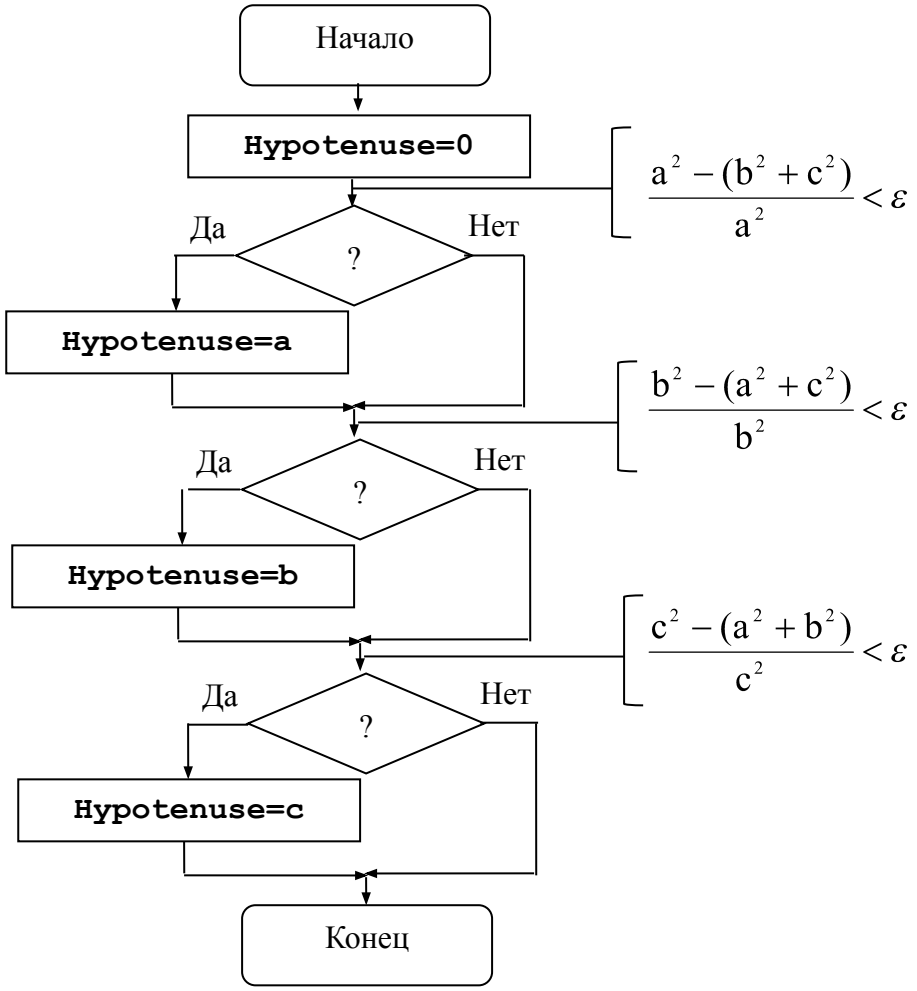

Начало
Input
Process
Flag=1
Process
Result
Вывод: «Это не треугольник»
Конец

Input:

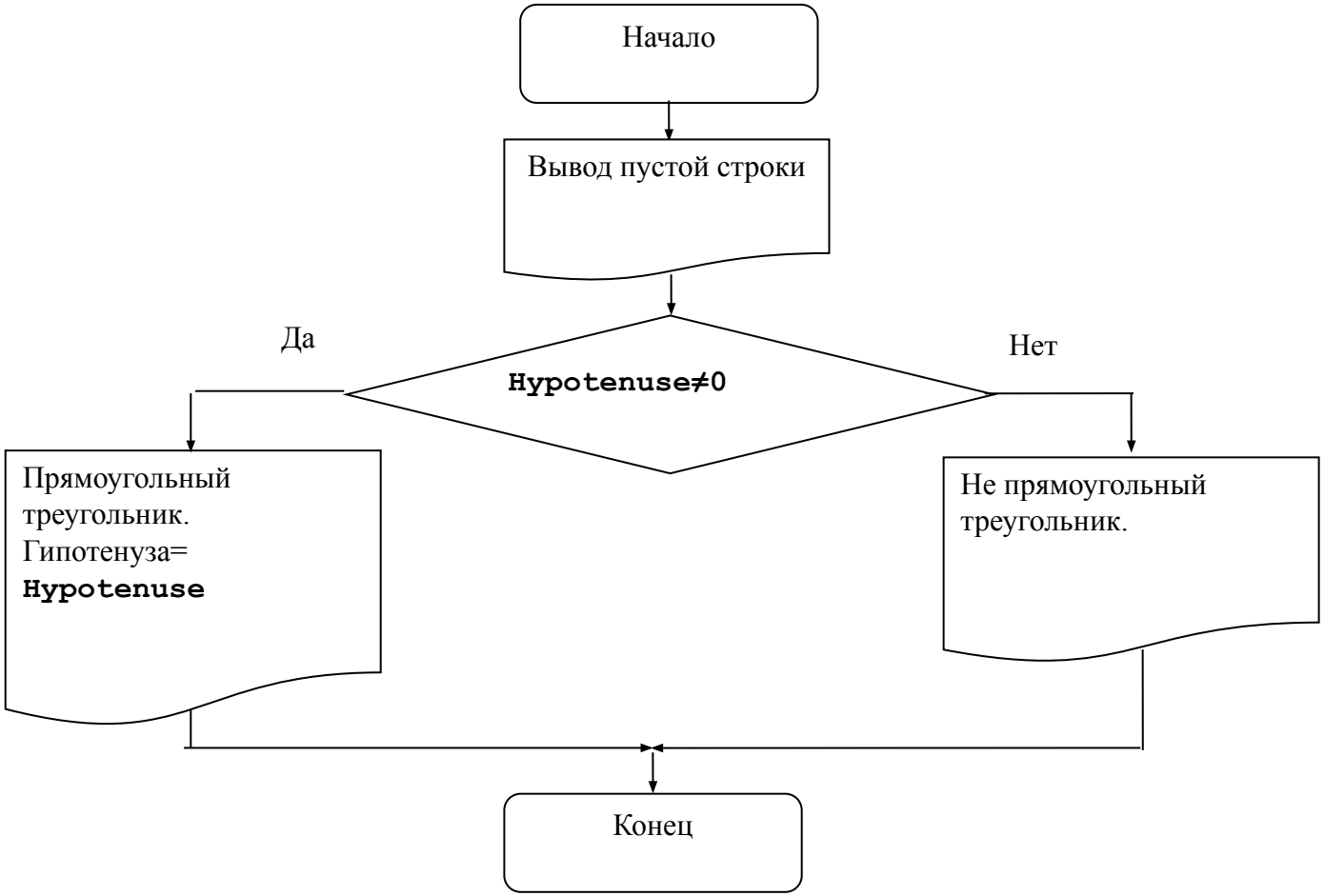

Начало
Очистка экрана
Введите 3 числа
Ввод a, b, c
Предполагаем, что a, b, c являются сторонами треугольника
Flag=1
НЕ(b+c>a И b+a>c И c+a>b)
Да
?
Нет
Flag=0
Конец

Epsilon:

Process:


Начало
Hypotenuse=0
$\frac{a^2-(b^2+c^2)}{a^2}<\varepsilon$
Да
?
Нет
Hypotenuse=a
$\frac{b^2-(a^2+c^2)}{b^2}<\varepsilon$
Да
?
Нет
Hypotenuse=b
$\frac{c^2-(a^2+b^2)}{c^2}<\varepsilon$
Да
?
Нет
Hypotenuse=c
Конец

Rezult:


Начало
Вывод пустой строки
Да
Hypotenuse≠0
Нет
Прямоугольный треугольник. Гипотенуза= Hypotenuse
Не прямоугольный треугольник.
Конец

- Любая последовательность операторов, встречающаяся в программе более одного раза, будучи вынесенной в отдельную функцию, сокращает размер программы.

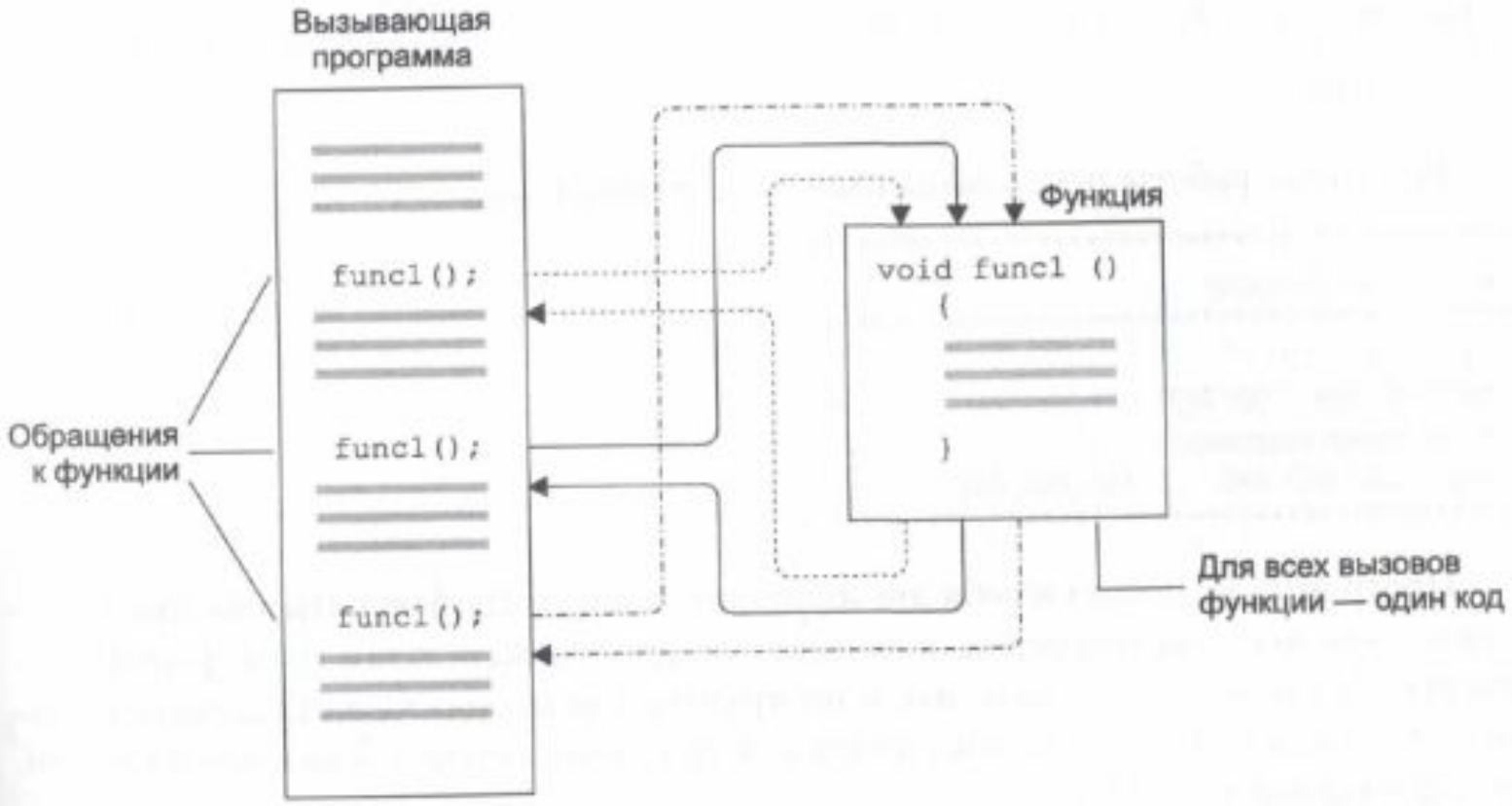


Рис. 5.1. Управление вызовами функции

Объявление и определение функций

- Функция, во-первых, является одним из производных типов C++, а, во-вторых, минимальным исполняемым модулем программы.

- Функция – это именованная последовательность описаний и операторов, выполняющая законченное действие.

- Объявление функции (прототип, заголовок) задает имя функции, тип возвращаемого значения и список передаваемых параметров.

//объявление
тип имя_функции ([список_формальных_параметров]);

- Определение функции содержит, кроме объявления, тело функции, которое представляет собой последовательность описаний и операторов.

//определение
тип имя_функции ([список_формальных_параметров])
{
тело_функции
}
Тело_функции – это блок или составной оператор.

- Внутри функции нельзя определить другую функцию

- Подобно тому как нельзя использовать переменную, не описав её, нельзя обратиться к функции, не указав её необходимые атрибуты

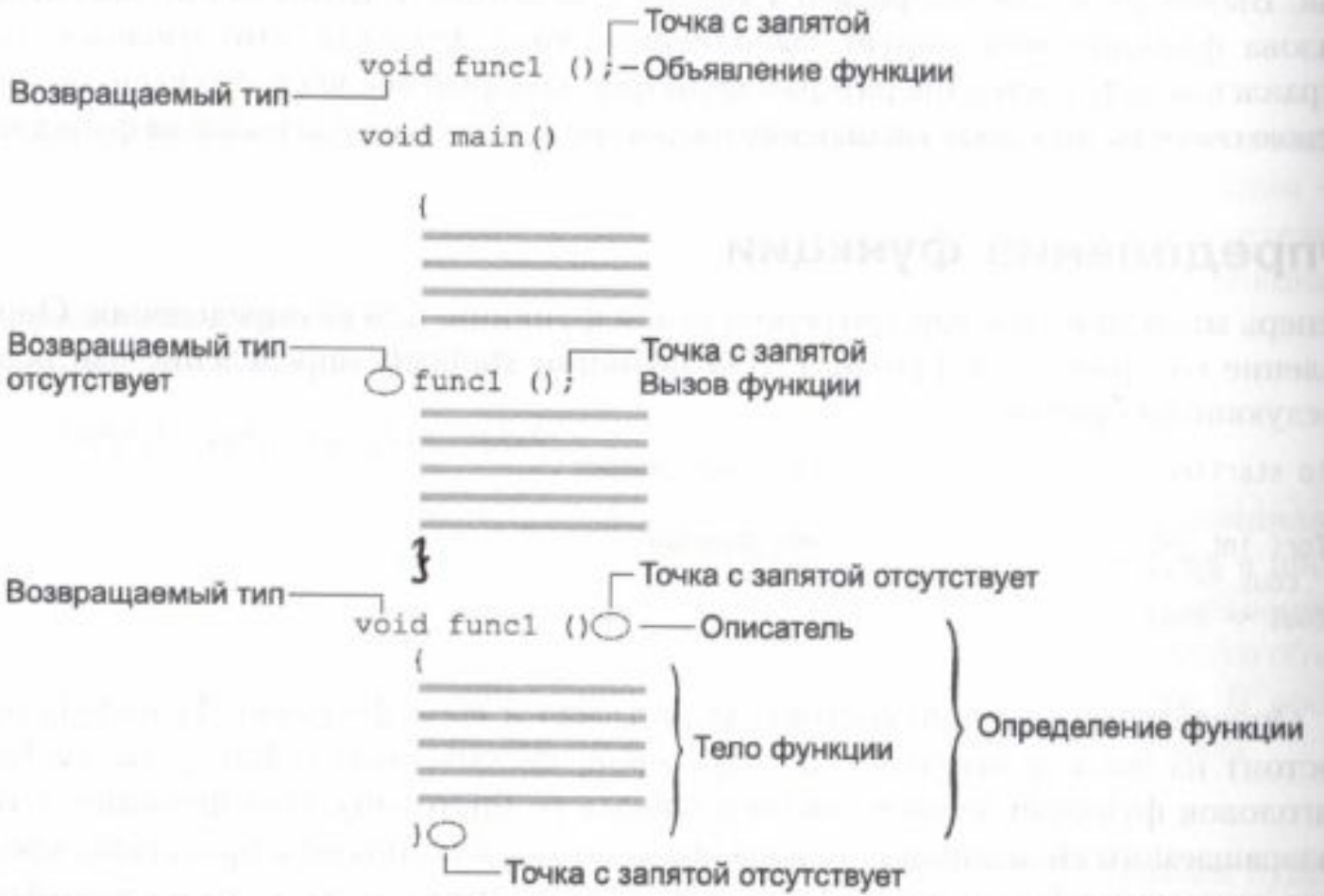


Рис. 5.2. Синтаксис функции

- В теле функции должен быть оператор, который возвращает полученное значение функции в точку вызова. Он может иметь две формы:
  - return выражение;
  - return;
- Первая форма используется для возврата результата, поэтому выражение должно иметь тот же тип, что и тип функции в определении.
- Вторая форма используется, если функция не возвращает значения, т. е. результат описан по типу void.
- Тип возвращаемого значения может быть любым, кроме массива и функции, но может быть указателем на массив или функцию.

Пример:

Вычислить значение *y*:

$$y = \frac{\max(a, \max(b, c)) + 4 * \max(a * b, c - a)}{\max(a, b) + \max(c * a - b, c * b - a)}$$

Удобнее ввести функцию, которая вычисляет максимум из двух чисел: **max (x,z)**.

```
float max (float x, float y) // Заголовок
{  float r; // Внутренняя переменная
     if (x>y) r=x; else r=y;
     return r;                    //тело функции
}
void main ()
{  int a,b;
   float y,c,d;
   scanf("%d%d",&a,&b);
   scanf("%f%f",&c,&d);
y=(max(a,max(b,c))–4*max(d*c–b,a*b+c))/
   (max(a*b–c,c*b)+max(d*a,b-c));
   printf("\ny=%5.2f",y);
}
```

/*Заданы координаты сторон треугольника, если такой треугольник существует, то найти его площадь. */

```
#include <iostream.h>
#include <math.h>
/*функция возвращает длину отрезка, заданного координатами x1,y1 и x2,y2*/
double line(double x1,double y1,double x2,double y2)
{
return sqrt(pow(x1-x2,2)+pow(y1-y2,2));
}
```

```
/*функция возвращает площадь треугольника, заданного длинами сторон a,b,c*/
double square(double a, double b, double c)
{
double s, p=(a+b+c) *0.5;
return s=sqrt(p*(p-a)*(p-b)*(p-c));//формула Герона
}
//возвращает true, если треугольник существует
bool triangle(double a, double b, double c)
{
if(a+b>c&&a+c>b&&c+b>a) return true;
else return false;
}
```

```
void main()
{
double x1=1,y1,x2,y2,x3,y3;
double point1_2,point1_3,point2_3;

cout<<"\nEnter koordinats of triangle:";
cin>>x1>>y1>>x2>>y2>>x3>>y3;

point1_2=line(x1,y1,x2,y2);
point1_3=line(x1,y1,x3,y3);
point2_3=line(x2,y2,x3,y3);

If   (triangle(point1_2,point1_3,point2_3)==true)
cout<<"S="<<square(point1_2,point2_3,point1_3)<<"\n";
    else cout<<"\nTriagle doesnt exist";
}
```

- Список формальных параметров – это те величины, которые требуется передать в функцию.
- Элементы списка разделяются запятыми. Для каждого параметра указывается тип и имя. В объявлении имена можно не указывать.
- При вызове указываются: имя функции и фактические параметры. Фактические параметры заменяют формальные параметры при выполнении операторов тела функции.

- Объявление функции должно находиться в тексте раньше вызова функции, чтобы компилятор мог осуществить проверку правильности вызова.
- Если функция имеет тип не void, то ее вызов может быть операндом выражения.

Задача

- Заданы координаты сторон треугольника, если такой треугольник существует, то найти его площадь

Описания (прототипы) функций

```
double line(double x1,double y1,double x2,double y2);

double square(double a, double b, double c);

bool triangle(double a, double b, double c);
```

```
double line(double ,double ,double ,double);

double square(double , double , double );

bool triangle(double , double , double );
```

Параметры функции

- Существует два способа передачи параметров в функцию:
  - по адресу
  - по значению.

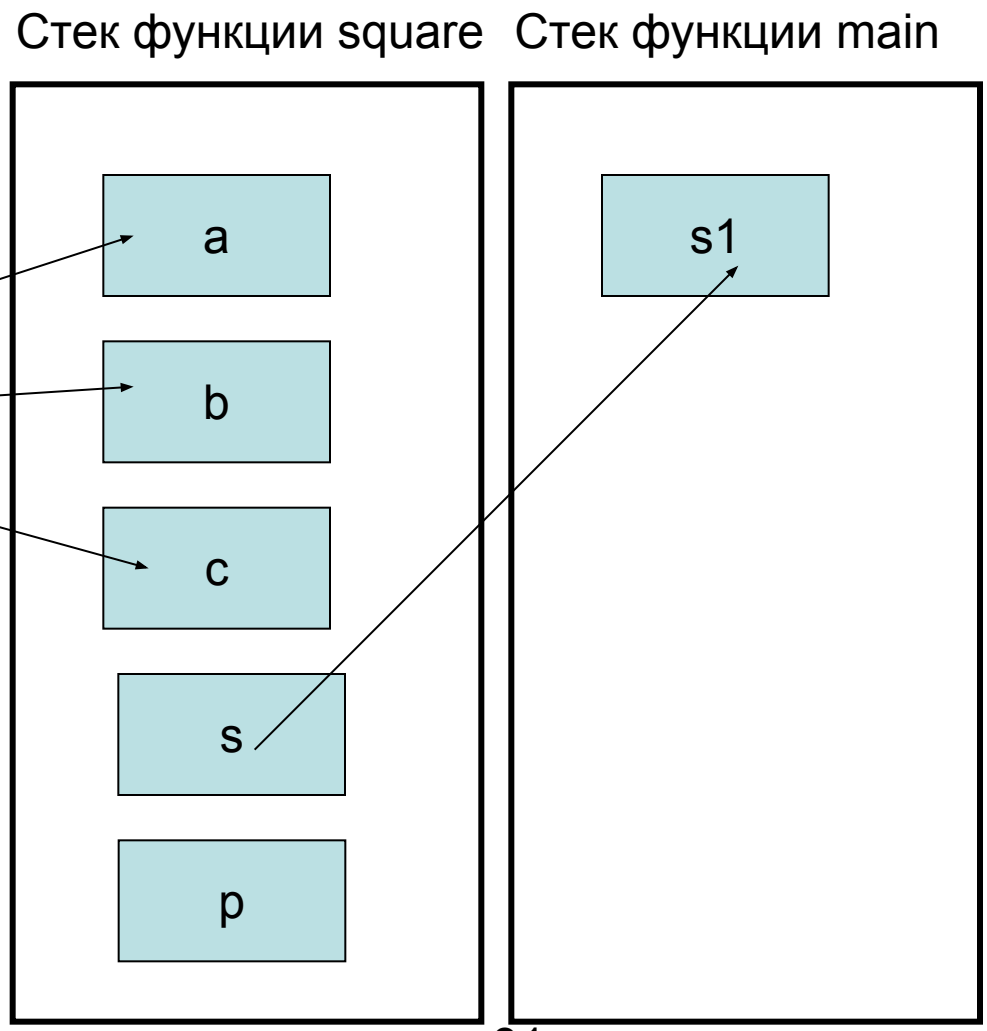
Передача параметров по значению

1. вычисляются значения выражений, стоящие на месте фактических параметров;
2. в стеке выделяется память под формальные параметры функции;
3. каждому формальному параметру присваивается значение фактического параметра, при этом проверяются соответствия типов и при необходимости выполняются их преобразования.

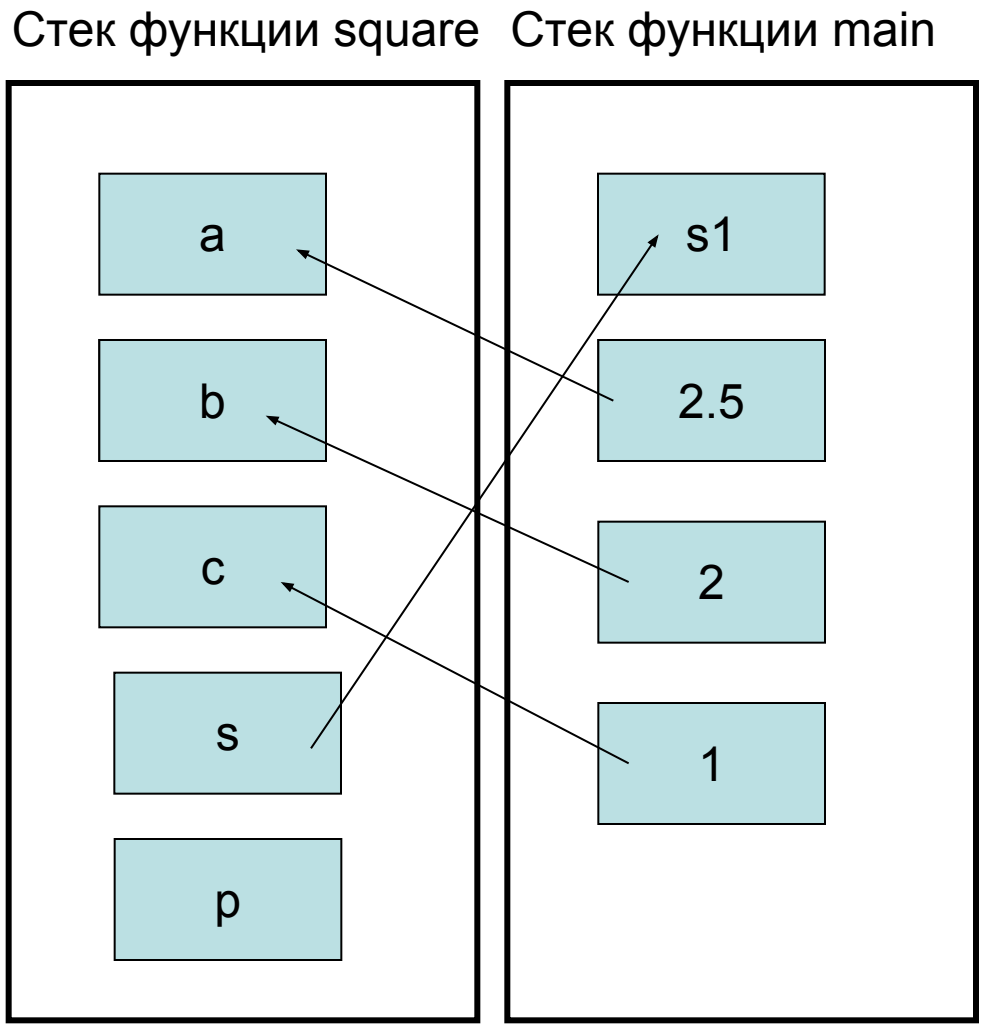
```
//функция возвращает площадь треугольника, заданного длинами сторон a,b,c

double square (double a, double b, double c)
{
double s, p=(a+b+c)/2;
return s=sqrt(p*(p-a)*(p-b)*(p-c));
}
```

```
//вызов функции
double s1=square(2.5,2,1);
```



```
//вызов функции
double a=2.5,b=2,c=1;
double s2=square (a, b, c);
```



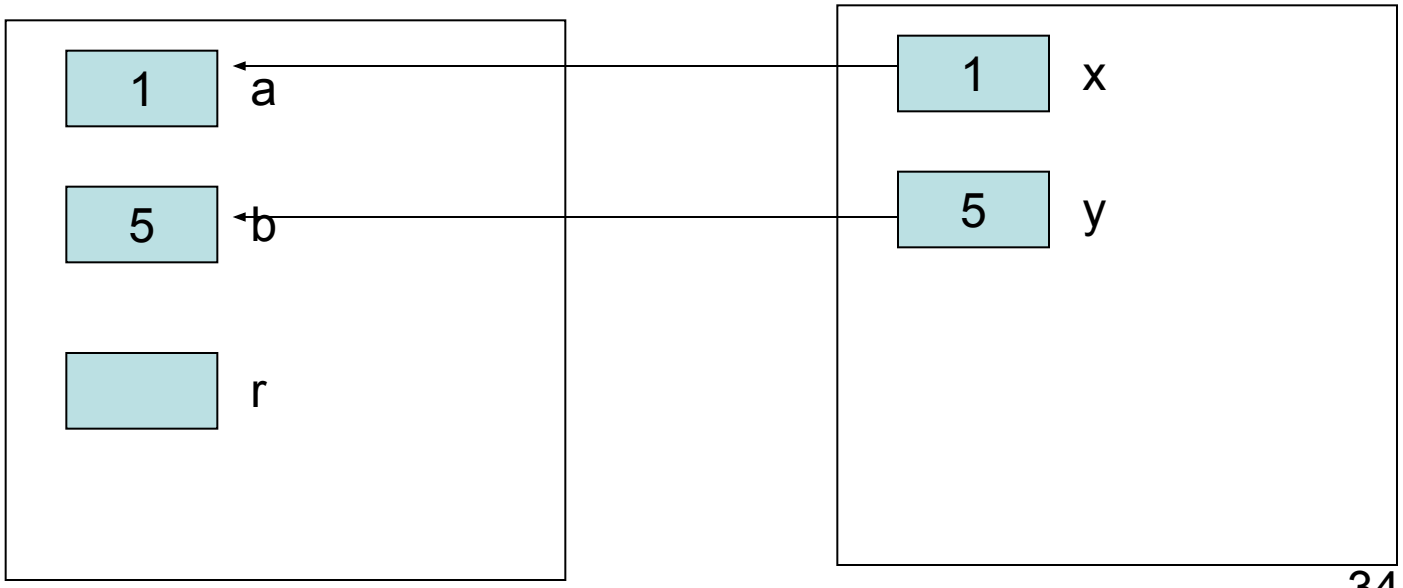
Таким образом, в стек заносятся копии фактических параметров, и операторы функции работают с этими копиями. Доступа к самим фактическим параметрам у функции нет, следовательно, нет возможности их изменить.

Передача параметров по адресу

- В стек заносятся копии адресов параметров, следовательно, у функции появляется доступ к ячейке памяти, в которой находится фактический параметр и она может его изменить.

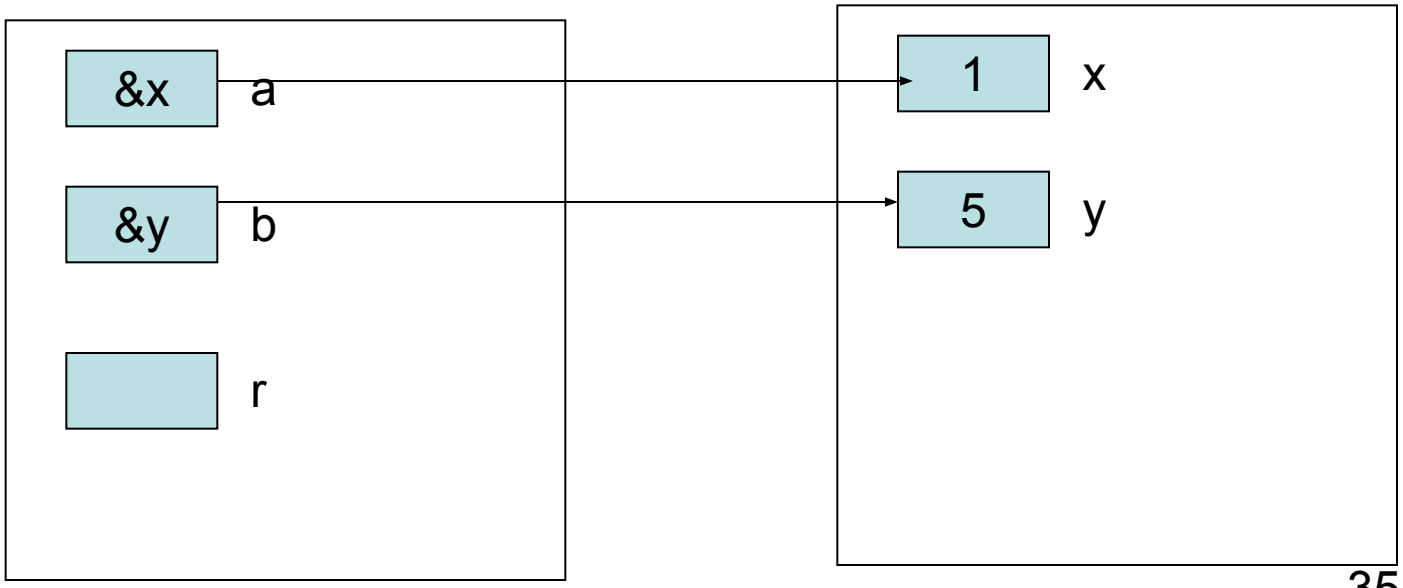
```
void Change (int a,int b)     //передача по значению
{
        int r=a;
        a=b;
        b=r;
}

//вызов функции
int x=1,y=5;
Change(x,y);
cout<<”x=”<<x<<” y=”<<y;
```



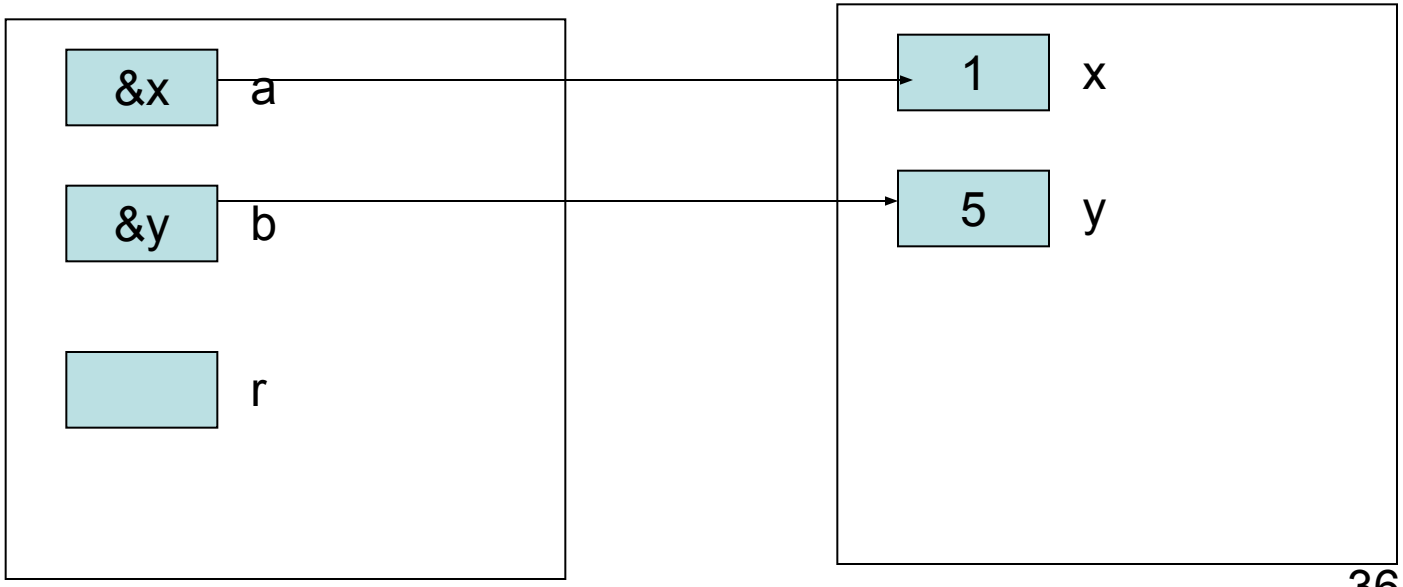
```
void Change (int* a, int* b) //передача по адресу
{
int r=*a;
*a=*b;
*b=r;
}

//вызов функции
int x=1,y=5;
Change(&x,&y);
cout<<"x="<<x<<" y="<<y;
```



```
void Change (int& a, int& b)      //передача по адресу
{
int r=a;
a=b;
b=r;
}

//вызов функции
int x=1,y=5;
Change(x,y);
cout<<”x=”<<x<<” y=”<<y;
```



Локальные переменные

- Переменные, которые используются внутри данной функции, называются локальными. Память для них выделяется в стеке, поэтому после окончания работы функции они удаляются из памяти.
- Нельзя возвращать указатель на локальную переменную, т. к. память, выделенная такой переменной, будет освобождаться.

```
int* f()
{
        int a;
        …
        return &a;// ОШИБКА!
}
```

Глобальные переменные

- Глобальные переменные – это переменные, описанные вне функций. Они видны во всех функциях, где нет локальных переменных с такими именами.

```
int a,b;          //глобальные переменные
void change()
{
      int r;//локальная переменная
      r=a;
      a=b;
      b=r;
}
void main()
{
      cin>>a,b;
      change();
      cout<<”a=”<<a<<”b=”<<b;
}
```

Подставляемые (inline) функции

- Спецификатор inline определяет для функции так называемое внутреннее связывание, которое заключается в том, что компилятор вместо вызова функции подставляет команды ее кода. При этом может увеличиваться размер программы, но исключаются затраты на передачу управления к вызываемой функции и возврата из нее.
- Подставляемыми не могут быть:
  - рекурсивные функции;
  - функции, у которых вызов размещается до ее определения;
  - функции, которые вызываются более одного раза в выражении;
  - функции, содержащие циклы, переключатели и операторы переходов;
  - функции, которые имеют слишком большой размер, чтобы сделать подстановку.

```
/* функция возвращает расстояние от
   точки с координатами (x1,y1) (по
   умолчанию центр координат) до точки с
   координатами (x2,y2)*/
inline float Line(float x1,float y1,
                  float x2=0,float y2=0)
{
return sqrt(pow(x1-x2)+pow(y1-y2,2));
}
```

Функции с переменным числом параметров

- В C++ допустимы функции, у которых при компиляции не фиксируется число параметров, и, кроме того, может быть неизвестен тип этих параметров. Количество и тип параметров становится известным только в момент вызова, когда явно задан список фактических параметров. Каждая функция с переменным числом параметров должна иметь хотя бы один обязательный параметр. Определение функции с переменным числом параметров:

```
тип имя (явные параметры, . . . )
{
      тело функции
}
```

Существует два подхода:

- известно количество параметров, которое передается как обязательный параметр;
- известен признак конца списка параметров.

Задача

Найти среднее арифметическое последовательности чисел, если известно количество чисел.

```
//Найти среднее арифметическое последовательности
//чисел, если известно количество чисел
#include <iostream.h>
float sum(int k, . . .)
//явный параметр k задает количество чисел
{
int *p=&k;//настроили указатель на параметр k
int s=0;
for(;k!=0;k--)
s+=*(++p);
return s/k;
}
```

```
void main()
{
//среднее арифметическое 4+6
cout<<”\n4+6=”<<sum(2,4,6);
//среднее арифметическое 1+2+3+4
cout<<”\n1+2++3+4=”<<sum(4,1,2,3,4);
}
```

```
/*Найти среднее арифметическое последовательности чисел, если
    известен признак конца списка параметров */

#include<iostream.h>
int sum(int k, …)
{
int *p = &k;  //настроили указатель на параметр k
int s = *p;          //значение первого параметра присвоили s
for(int i=1;p!=0;i++)  //пока нет конца списка
s += *(++p);
return s/(i-1);
}
void main()
{
//находит среднее арифметическое 4+6
cout<<”\n4+6=”<<sum(4,6,0);
//находит среднее арифметическое 1+2+3+4
cout<<”\n1+2++3+4=”<<sum(1,2,3,4,0);
}
```

Рекурсия

- Рекурсией называется ситуация, когда какой-то алгоритм вызывает себя прямо (прямая рекурсия) или через другие алгоритмы (косвенная рекурсия) в качестве вспомогательного. Сам алгоритм называется рекурсивным.
- Рекурсивное решение задачи состоит из двух этапов:
  - исходная задача сводится к новой задаче, похожей на исходную, но несколько проще;
  - подобная замена продолжается до тех пор, пока задача не станет тривиальной, т. е. очень простой.

Задачи

- Вычислить факториал (n!), используя рекурсию.
- Вычислить степень, используя рекурсию.

Задача 1. Вычислить факториал (n!), используя рекурсию.

Исходные данные: n

Результат: n!

Рассмотрим эту задачу на примере вычисления факториала для n=5. Более простой задачей является вычисление факториала для n=4. Тогда вычисление факториала для n=5 можно записать следующим образом:

5!=4!*5.

Аналогично:

4!=3!*4;

3!=2!*3;

2!=1!*2 ;

1!=0!*1

Тривиальная (простая) задача:

0!=1.

Можно построить следующую *математическую модель*:

$$f(n) = \begin{cases} 1, \mathrm{n} = 0 \\ f(n-1) * n, \mathrm{n} \geq 1 \end{cases}$$

```
#include <iostream.h>
int fact(int n)
{
    if (n==0)return 1;      //тривиальная задача
    return (n*fact(n-1));
}

void main()
{
    cout<<"n?";
    int k;
    cin>>k;  //вводим число для вычисления факториала

//вычисление и вывод результата
    cout<<k<<"!="<<fact(k);
}
```

Задача 2. Вычислить степень, используя рекурсию.

Исходные данные: x,n

Результат: xn

Математическая модель:

$$pow(x,y) = \begin{cases} 1, \mathrm{n} = 0 \\ pow(x, n-1) * x, \mathrm{n} \geq 1 \end{cases}$$

```
#include <iostream.h>
int pow( int x,int n)
{
    if(n==0)return 1;//тривиальная задача
    return(x*pow(x,n-1));
}
void main()
{
    int x,k;
    cout<<"n?";
    cin>>x; //вводим число
    cin>>k; //вводим степень
        //вычисление и вывод результата
    cout<<x<<"^"<<k<<"="<<pow(x,k);
}
```