Подпрограммы

Подпрограмма(функция) – именованная, логически законченная группа операторов, которую можно один раз описать, а затем вызвать внутри функции main() по имени необходимое кол-во раз.

Объявление функции:

тип <имя функции>(список формальных параметров)

```
{
        <тело функции>
        return < значение>;
}
```

тип <имя функции>(список формальных параметров) – заголовок функции. Не является самостоятельной структурной единицей. После заголовка функции ; не ставится!!!

Если функция имеет тип - void (пустой, ничего не возвращающий), оператор - return отсутствует!!

Если функция имеет тип отличный от void (int, double, char, bool) оператор return прерывает выполнение функции и в момент ее вызова передает (возвращает) результат в точку вызова функции. В этом случае тип функции и тип возвращаемого результата должен совпадать.

Функция, имеющая тип отличный от void, может быть вызвана в выражении, или использована с функцией printf() и потоком cout.

Если функция имеет тип void – она не может быть вызвана в выражении, или использована с функцией printf() и потоком cout. Она просто вызывается по имени.

Пример

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

int max (int a, int b)
 {
     if (a>b) return a;  else return b;
 }
int main ()
      {
            int x,y;
            cout<<" vvedite x=";
            cin>>x;
            cout<<" vvedite y=";
            cin>>y;
            cout<<"maks="<<max(x,y)<<endl;
            return 0;
      }
```

Мы описываем функцию, которая находит максимальное из двух чисел. При вызове функции (cout<<max(x,y)) значение максимального из чисел будет выведено на экран монитора.

Формальные и фактические параметры.

Параметры определяют начальные условия для выполнении функции и служат для обмена данными между ними.

Параметрами могут быть переменные, константы, массивы, строки и т.д.

Формальные параметры определяют имена, под которыми

данные будут передаваться из подпрограммы в подпрограмму.
Фактические параметры – это конкретные данные.
В момент вызова функции устанавливается связь между формальными и фактическими параметрами по кол-ву, типу и порядку следования.
Формальные параметры описаны в заголовке функции, фактические подставляются вместо формальных при ее вызове. В предыдущем примере формальные параметры- a и b, фактические- x и y.

Прототипы функций

В языке C++ для создания правильного машинного кода программы до первого вызова функции необходимо сообщить имя функции, тип возвращаемого результата, кол-во и тип параметров.

Если функция описана до функции main() (как в нашем примере), то заголовок функции служит ее объявлением.

Однако, описание функции можно сделать после функции main() или вообще в отдельном файле. В этом случае в качестве объявления функции используется ***прототип***

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

int max (int, int) ;   - прототип функции
int main ()
    {
        int x,y;
        cout<<" vvedite x=";
        cin>>x;
        cout<<" vvedite y=";
        cin>>y;
```

```
        cout<<"maks="<<max(x,y)<<endl;
        return 0;
    }
int max (int a, int b)
 {
    if (a>b) return a;  else return b;
 }
```

Способы передачи параметров функции

В языке С++ существует 3 способа передачи параметров функции – по значению, по ссылке и через указатель. На примере одной задачи рассмотрим каждый из этих способов

1. *Пердача параметров по значению*

   В момент вызова функции в стеке выделяется место для ее работы. Если параметр передается по значению, то в стек помещается копия этого параметра. И какие бы изменения с этим параметром внутри функции не произошли, его значение из функции передано не будет. Такие параметры обычно служат начальными условиями для работы функции

```
    # include <iostream>
    using namespace std;
int kol(int c)
{
  int i=1, d=0;
  int k=16;
 do
   { if (i%c==0) d+=1;
     i+=1;
```

```
    }
  while(i<k);
  return d;
}

int main()
{
  cout<<''perviy rez=''<<  kol(2)<<endl;
  cout<<"vtoroy rez=",kol(4);
  return 0;
}
```

Функция kol(int c) имеет 1формальный параметр c, который передается по значению. Функция вызвана 2 раза. При первом вызове функции формальный параметр - c будет заменен фактическим -2, второй раз 4. Функция имеет тип int, следовательно она будет возвращать результат типа int в точку вызова функции.

Постарайтесь сами ответить на вопрос что напечатает ЭВМ.

2. Пердача параметров по ссылке

Если параметр передается по ссылке, то в стек помещается адрес этого параметра. В этом случае после завершения работы функции значение этого параметра будет предано по соответствующему адресу.

```
    #include <iostream>
using namespace std;
void kol(int c ,int &d)
{  int k=16;
   int i=1;
   d=0;
   do
   { if (i%c==0) d+=1;
```

```
     i++;
    }
    while (i<k);
}
int main()
{
    int d1,d2;
    kol(2,d1);
    kol(4,d2);
    cout<<"perviy resultat"<<d1<<endl;
     cout<<"vtoroy resultat"<<d2<<endl;
     return 0;
}
```

Функция kol(int c, int &d) имеет 2 параметра. с -передается по значению, а d - по ссылке. Функция имеет тип void, соответственно ее можно только вызвать по имени.

При выполнении программы на экран монитора будет выведен такой же результат, как и в предыдущем случае.

3.Пердача параметров через указатель

Как и в предыдущем случае, в стек помещается адрес параметра. Однако, код программы будет более сложным, т.к. при работе с указателями необходимо будет использовать операцию разименования.

```
#include <iostream>
using namespace std;
void kol(int  ,int *);
int main()
{
```

```
    int d1,d2;
    kol(2,&d1);
    kol(4,&d2);
    cout<<"perviy resultat"<<d1<<endl;
     cout<<"vtoroy resultat"<<d2<<endl;
     return 0;
}
void kol(int c ,int *d)
{
    int i=1;
    *d=0;
    do
    { if (i%c==0)*d+=1;
      i++;
    }
    while (i<k);
}
```

Обратите внимание на использование прототипа в данной программе

Передача массивов в качестве параметров функции

Передачу массивов в качестве параметров функции можно осуществить только через указатель!!!!!

Существует несколько способов записи

1. void (int n, int *x, int *y);
2. void (int n, int x[], int y[]);

Здесь x, y – массивы.

Наиболее предпочтителен вариант 2, т.к. визуально сразу понятно, что мы передаем массив, а не переменную через указатель.

```
#include <iostream>
using namespace std;
float sa(int x[], int n, int k);
int main()
{ int m1[4]={1,5,2,9};
  int m2[6]={3,9,4,8,3,6};
  float r1,r2;
  r1=sa(m1,4,3);
  r2=sa(m2,6,4);
  cout<< ''rez1=''<<r1<<''   rez2="<<r2);
  return 0;
}
float sa(int x[],int n, int k)
{ int i;
  float s=12;
  for (int i=0;i<n;i++)
      if (x[i]%k==0) s/=k;
  return s;
}
```