Поим за модули и потпрограми

Поедноставно за решавање на сложените задачи е тие да се расчленат на поедноставни делови, или пак ако еден дел од задачата се извршува повеќе пати, тој да се напише како посебна целина. Секоја ваква целина може да се разгледува независно од другите. Алгоритмот напишан за ваква целина е подалгоритам, а програмата е потпрограма.

Подалгоритмите може да се користат во различни програми, а и на повеќе места во ист алгоритам. **Ова е овозможено преку листа на формални и вистински аргуманти.** Општ запос наподалгоритам е:

име на подалгоритмот(листа на формални аргументи);

1. Поделба на алгоритмите според бројот на излезните аргументи:

- **функциски подалгоритми**
- **процедурални подалгоритми**

Функциските подалгоритми се наречени функции. Имаат само еден излезен резултат, исто како и функциите, а може да имаат повеќе влезни аргументи. Пр. f(x), g(x,y), t(x,y,z).

Општа синтакса:

именаподалгоритмот(влезни аргументи);

влезни аргументи се влезни формални аргументи.

пр. подалгоритам за наоѓање на поголем од два дадени броја може да се запише:

```
подалгоритам поголем(број1, број2);
почеток
    ако broj1>broj2
       тогаш
              pogolem=broj1
       инаку
              pogolem=broj2
    крај_ако;
    rezultat=pogolem;
крај.
```

Со помошната променлива rezultat резултатот од пресметувањето во подалгоритмот се враќа во главната програма.

Функциски подалгоритам се повикува со:

променлива=именаподалгоритмот(листа на вистински аргументи)

Листата на вистински аргументи мора да е иста со листата на формални аргументи. И тоа, ист да е бројот на аргументи, ист редослед на аргументи, ист тип на аргументи.

Функциските подалгоритми се повикуваат со чекор за доделување.

пр. р=поголем(а,б).

Со ова, листата на формални аргументи broj1 и broj2 се заменува со листата на вистински аргументи а и б.

Со. н=поголем(р,с)

подалгоритмот се повикува со вистинските параметри р и с и најголемата вредност се сместува во променливата н.

Па алгоритмот за наоѓање најголем од три дадени броја е:

```
алгоритам најголем;
почеток
    читај а,б,с;
    р=поголем(а,б);
    н=поголем(р,с);
    печати н;
```

крај.

Реализација на функциските подалгоритми е со функциски потпрограми или **функции.** Тие може да бидат:

2. Видови на функции:

- **вградени функции**
- **кориснички функции**

Вградените функции се наоѓаат во програмски библиотеки.

Функциите се градбени единки на програмите. Тие овозможуваат произволно структурирање на кодот и поголема модуларност. Кај сите програми кои ги разгледавме досега, наредбите кои требаше да се извршат ги пишувавме во една функција - main(). Тоа е основниот дел кој го содржат сите програми - main() е првата функција која се повикува при стартот на една програма напишана во програмскиот јазик C++. Па и главната програма во C++ всушност претставува функција.

Повик на одредена функција претставува наредба до процесорот за запирање на тековното извршување на наредби, запишување на моментната локација во програмата и повикување на соодветната функција. По завршување на повиканата функција, програмата продолжува со извршување од местото од каде е функцијата повикана.

Основната синтакса за креирање на функции е следната:

```
tip ime(parametar1, parametar2, parametar3, parametar4, ...)
{
  naredba1;
  naredba2;
  naredba3;
  .....
  naredbaN;
}
```

Притоа, tip е типот на податок кој се враќа како резултат од функцијата (или void, ако функцијата не враќа резултат), parametar1, parametar2, ..., итн, се параметрите со кои се повикува функцијата (тип и име), одделени со запирка, додека naredba1, naredba2, ..., naredbaN, ги претставуваат наредбите кои се извршуваат при секој повик на функцијата. Листата на параметри е незадолжителна и можно е повикување на функција без параметри - на пример, main().

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

int max(int b1,int b2)

{
      int k;
            if (b1>b2) k=b1;
            else k=b2;

      return k;
      }

int main()
      {
int a,b,c,n,p,b1,b2;
cin>>a>>b>>c;
            p=max(a,b);
            n=max(p,c);

      cout << n << endl;
```

```
return 0;
}
```
пр. к!=к*(к-1)*...*1

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

int factorial(int n)
{
    int res,i;
        res = 1;
    for (i=2; i<=n; i++)
        res*= i;
    return res;
}

int main()
{
    int n;
        cin>>n;
    int f = factorial(n);

    cout << f << endl;

    return 0;
}
```

функцијата factorial(int n) е декларирана пред функцијата main(). Во C++, тоа е задолжително - не може да се повика функција која не е претходно декларирана.

Функцијата која ја повикавме е дефинирана како "int factorial(n)", што означува дека таа треба да врати резултат цел број. Тоа го правиме со наредбата од линија 10 - функцијата враќа резултат res=24, ако н=4. Овој резултат се запишува во променливата f (дефинирана во main()). По ова, програмата го продолжува извршувањето од линија 19 - каде што обележавме дека треба да се вратиме по повикот на функцијата factorial(int n). Резултатот од извршувањето на една функција се запишува (како директно да сме внеле вредност) на местото од каде се повикала таа функција. Во овој случај, 24, резултатот од извршувањето на factorial(4), директно се внесува во променливата f ("int f = 24;").

Основни математички функции – библиотека cmath

Во C++ постои стандардна математичка библиотека cmath која содржи многу математички функции. Најчесто користени се:

sqrt(x) – квадратен корен од x
exp(x) – експоненцијална функција e^x
log(x) – природен логаритам од x (основа e)
log10(x) – декаден логаритам д x (основа 10)
fabs(x) – апсолутна вредност од x
ceil(x) – заокружување на x на најмалиот цел број не помал од x пр. 7.8 е 8. пр 5.3 е 6
floor(x) - заокружување на x на најголемиот цел број не поголем од x (целиот дел од реален број) пр.7.8 е 7 пр. 5.3 е 5

round(x) заокружување на бројот пр. 7.8 е 8 пр.5.3 е 5
pow(x,y) – x на степен y
fmod(x,y) – остаток од x/y како реален број пр. 7.8 со 3.2 е 1.4 пр. 5.3 со 4.2 е 1.1
sin(x) – синус од x
cos(x) – косинус од x
tan(x) - тангенс од x
пр. за да се испечати квадратен корен од x: cout<<sqrt(900);

Аргументите на функциите може да бидат константи, променливи или изрази.
пр. c1=4.2, d=3.0, f=4.0
cout<<sqrt(c1+d*f);
се печати 4.0049

Во програми, за да може да се користат вградените математички функции мора да се вклучи библиотеката cmath. Тоа се прави со наредбата # include <cmath>

Имплементација и примена на функции

1. Да се напише програма со која се пресметува и печати s= $\sqrt{1}+\sqrt{3}+\ldots+\sqrt{n}$

```
# include <iostream>
# include <cmath>
using namespace std;
int main()
{
double s;
int i,n;
cin>>n;
s=0;
for (i=1;i<=n;i+=2)
s+=sqrt(i);
cout<<"zbirot na korenite e "<<s<<endl;
return 0;
}
```

2. Да се напише програма со кој се пресметува и печати вредноста на sin за агол од 0 до 360 степени со чекор 10.

```
# include <iostream>
# include <cmath>
using namespace std;
int main()
{
int alfa;
cout<<"alfa sin(alfa)"<<endl;
cout<<"_____"<<endl;
for (alfa=0;alfa<=360;alfa+=10)
cout<<alfa<<" "<< sin(alfa*3.14/180)<<endl;
return 0;
}
```

3. Да се напише програма со која се пресметува тангенс од агол алфа ако се дадени страните а и б.

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

```
int main()
{
double a,b;
cout<<"vnesi gi stranite a i b "<<endl;
cin>>a>>b;
cout<<"tangens e "<<tan(a/b)<<endl;
return 0;
}
```

4. Да се напише програма за пресметување на функцијата
f(x)=cos(x)+4*sin(x)-tan(x)

```
# include <iostream>
# include <cmath>
using namespace std;
int main()
{
double x,y;
cout<<"vnesi ja vrednosta na x "<<endl;
cin>>x;
x1=x*3.14/180;
y=cos(x1)+4*sin(x1)-tan(x1);
cout<<"funkcijata e "<<y<<endl;
return 0;
}
```

Изработка на функции и нивно користење во програми

1. да се напише програма која со потпрограма пресметува збир на природните броеви од 1 до н.

```
#include <iostream>
using namespace std;
int soberi(int a)
{
int s,i;
s=0;
for (i=1; i<=a; i++)
s+=i;
return s;
}
int main()
{
int a,n;
cout<<"vnesi go n ";
cin>>n;
cout<<"zbirot od 1 do "<<n<<" e "<<soberi(n)<<endl;
return 0;
}
```

2. да се напише програма која со потпрограма пресметува збир на н природните броеви.

```
#include <iostream>
using namespace std;
int soberi(int a)
{
int s,i,b;
s=0;
```

```
for (i=1; i<=a; i++)
{
cin>>b;
s+=b;
}
return s;
}
int main()
{
int a,n;
cout<<"vnesi go n ";
cin>>n;
cout<<"zbirot "<<n<<" e "<<soberi(n)<<endl;
return 0;
}
```

3. да се напише програма која со потпрограма пресметува средна вредност на н природни броеви.

```
#include <iostream>
using namespace std;
double sredna(int a)
{
int s,i,b;
double sr;
s=0;
for (i=1; i<=a; i++)
{
cin>>b;
s+=b;
}
sr=s*1.0/a;
return sr;
}
int main()
{
int a,n;
cout<<"vnesi go n ";
cin>>n;
cout<<"srednata vrednost e "<<n<<" e "<<sredna(n)<<endl;
return 0;
}
```

4. да се напише програма која со потпрограма определува нзд со евклидов алгоритам.

```
# include <iostream>
using namespace std;
int nzd(int a, int b)
{
int p,ost;
if (a<b)
{
p=a;
a=b;
b=p;
}
do
{
ost=a%b;
a=b;
b=ost;
} while (b!=0);
```

```
return a;
}
int main()
{
int n,m;
cin>>n>>m;
cout<<" nzd e "<<nzd(n,m);
return 0;
}
```

5. да се напише програма која со потпрограма определува збир на цифрите на природниот број н.

```
# include <iostream>
using namespace std;
int zbir(int n)
{
int s, ost;
s=0;
while (n>0)
{
ost=n%10;
s+=ost;
n=n/10;
}
return s;
}
int main()
{
int n;
cin>>n;
cout<<" zbirot na cifrite e "<<zbir(n);
return 0;
}
```