Subprograme

Sumar

Competenţe generale

- *implementarea algoritmilor într-un limbaj de programare*
- *elaborarea algoritmilor de rezolvare a problemelor*
- *aplicarea algoritmilor fundamentali în prelucrarea datelor*
- *identificarea conexiunilor dintre informatică şi societate*

Competenţe specifice

- *utilizarea corectă a subprogramelor predefinite şi a celor definite de utilizator*
- *construirea unor subprograme pentru rezolvarea subproblemelor unei probleme*
- prelucrarea datelor structurate
- recunoaşterea situaţiilor în care este necesară utilizarea unor subprograme
- analiza problemei în scopul identificării subproblemelor acesteia
- *elaborarea unui algoritm de rezolvare a unei probleme din aria currciculară a specialităţii*

2. Declararea, definirea şi apelul subprogramelor

Noţiunea de subprogram este legată de ideea generală a descompunerii unei probleme în subprobleme.

În rezolvarea problemelor apar următoarele situaţii care necesită o rezolvare:

- o secvenţă dintr-un program se repetă;
- există mai multe programe care au nevoie de un anumit calcul.

Apare ideea ca acea secvenţă sau acel calcul să fie scris o dată şi să fie folosit ori de câte ori este nevoie. În astfel de situaţii se folosesc ***subprograme***.

Astfel, un ***subprogram*** reprezintă o parte dintr-un program, identificat prin nume, care se poate lansa în execuţie ori de câte ori este cazul.

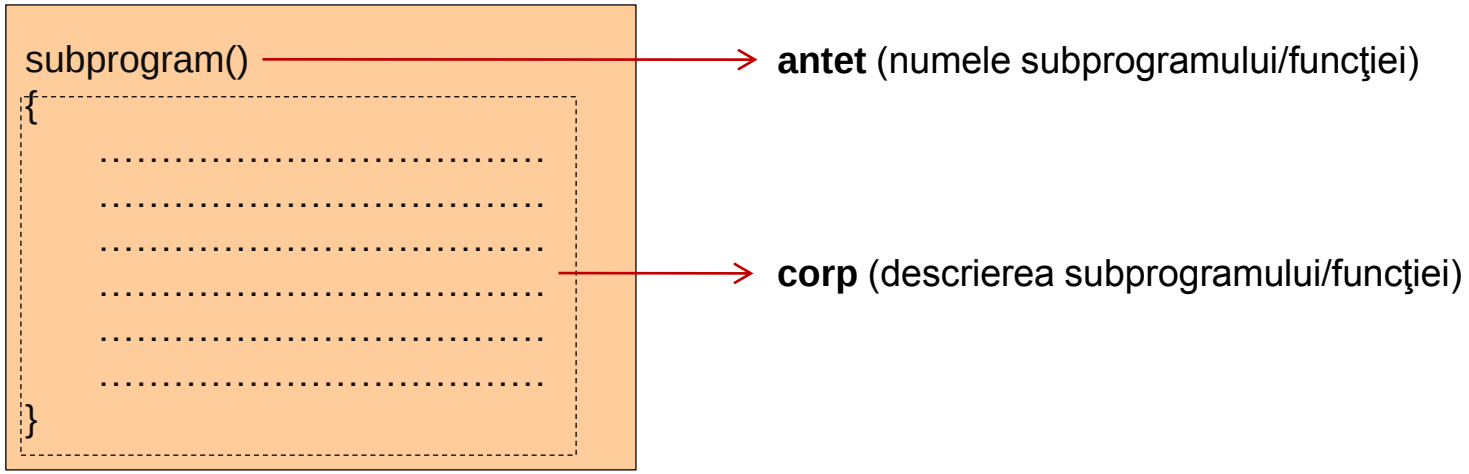
Definiţie

Subprogramul *este o secvenţă de instrucţiuni care rezolvă o anumită sarcină şi care poate fi descris separat de blocul rădăcină şi lansat în execuţie din cadrul unui bloc ori de câte ori este nevoie.*

În C++ subprogramele se numesc ***funcţii***.

Un subprogram, la fel ca şi programul, este format din:

- *antet* şi
- *corpul subprogramului*.



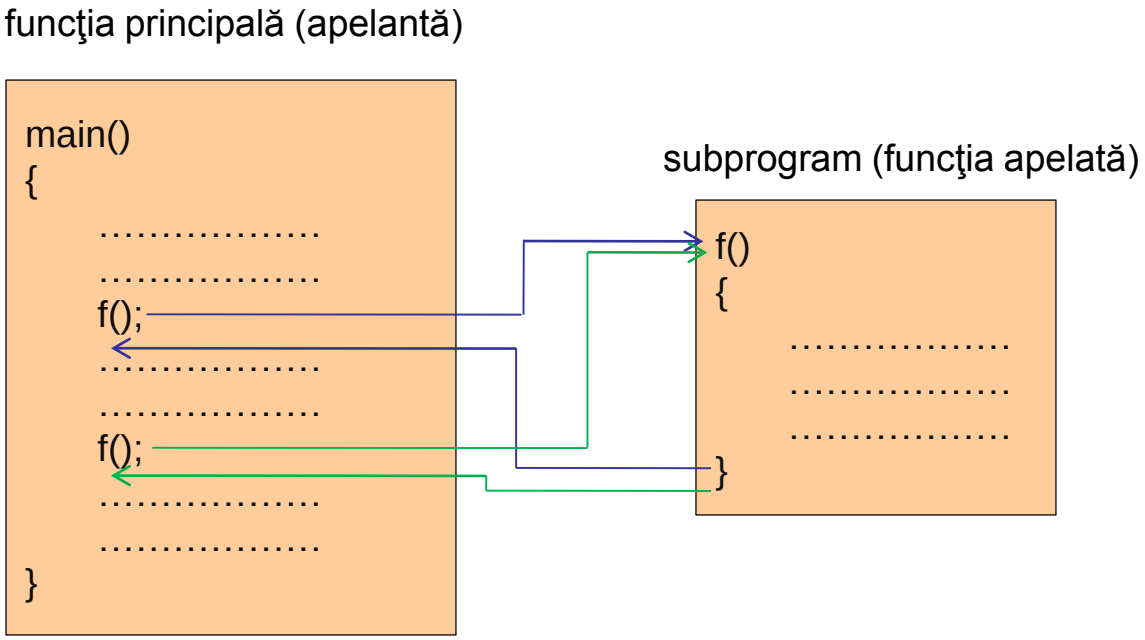
Exemplu

```
int main()
{
    .....
    .....
    .....
}
```

Pentru a executa o funcţie, aceasta trebuie apelată.

Apelul unei funcţii se face scriind numele funcţiei, urmat de paranteze rotunde () şi, eventual, de o listă de parametri.

Parametri sunt date pe care funcţia apelantă le transmite funcţiei apelate.



Exemplu

Să se calculeze şi să se afişeze media aritmetică a două numere reale **`x`** şi **`y`**. Se va folosi o funcţie **`calcul`** care citeşte două numere reale **`x`** şi **`y`**, calculează media lor aritmetică şi afişează rezultatul obţinut.

Declararea, definirea şi apelul subprogramelor

```
1  #include<iostream>
2  using namespace std;
3
4  void calcul()
5  {
6      float x,y,ma;
7      cout<<"x=";cin>>x;
8      cout<<"y=";cin>>y;
9      ma=(x+y)/2;
10     cout<<"media="<<ma;
11 }
12
13 int main()
14 {
15     calcul(); <-- instrucţiunea de apel
16     return 0;
17 }
```

Declararea, definirea şi apelul subprogramelor

```
1   #include<iostream>
2   using namespace std;
3
4   void calcul();
5
6   int main()
7   {
8       calcul();
9       return 0;
10  }
11
12  void calcul()
13  {
14      float x,y,ma;
15      cout<<"x=";cin>>x;
16      cout<<"y=";cin>>y;
17      ma=(x+y)/2;
18      cout<<"media="<<ma;
19  }
```

prototipul subprogramului

3. Variabile locale şi variabile globale

O variabilă este vizibilă numai în interiorul subprogramului în care aceasta a fost declarată, începând din momentul declarării ei.
Variabilele declarate în afara subprogramului sunt vizibile în toate subprogramele care urmează declarării.

Din acest punct de vedere, variabilele se împart în două categorii:
• ***variabile locale (interne)*** – sunt recunoscute din momentul declarării lor şi sunt vizibile numai în interiorul subprogramului în care au fost declarate;
• ***variabile globale (externe)*** – sunt vizibile în toate subprogramele, din momentul declarării lor.

Exemplu 1

Să se calculeze şi să se afişeze media aritmetică a două numere reale **x** şi **y**. Se va folosi o funcţie **calcul** care citeşte două numere reale **x** şi **y**, calculează media lor aritmetică şi afişează rezultatul obţinut. Toate variabilele folosite sunt variabile globale.

Variabile locale şi variabile globale

```
1   #include<iostream>
2   using namespace std;
3
4   float x,y,ma;
5
6   void calcul()
7   {
8       cout<<"x=";cin>>x;
9       cout<<"y=";cin>>y;
10      ma=(x+y)/2;
11      cout<<"media="<<ma;
12  }
13
14  int main()
15  {
16      calcul();
17      return 0;
18  }
```

variabile globale

Exemplu 2

Să se calculeze şi să se afişeze media aritmetică a două numere reale **`x`** şi **`y`**. Se va folosi o funcţie ***`calcul`*** care calculează media lor aritmetică şi afişează rezultatul obţinut. Valorile variabilelor **`x`** şi **`y`** sunt citite în funcţia apelantă.

Variabile locale şi variabile globale

```
1   #include<iostream>
2   using namespace std;
3
4   float x,y,ma;
5
6   void calcul()
7   {
8       ma=(x+y)/2;
9       cout<<"media="<<ma;
10  }
11
12  int main()
13  {
14      cout<<"x=";cin>>x;
15      cout<<"y=";cin>>y;
16      calcul();
17      return 0;
18  }
```

variabile globale

4. Parametri transmişi prin valoare

Prin intermediul parametrilor, funcţia schimbă informaţii cu blocul apelant. Parametri sunt reprezentaţi atât în antetul subprogramului cât şi în instrucţiunea de apel prezentă în blocul apelant.

La fiecare apel al unei funcţii, blocul (funcţia) apelantă poate transmite date funcţiei apelate, pe care funcţia apelantă le va folosi atunci când aceasta este executată. Aceste date se numesc ***parametri***.

Parametrii sunt de două tipuri:

- ***parametri formali*** – apar în antetul funcţiei şi nu au valoare atunci când sunt definiţi;
- ***parametri actuali (efectivi)*** – apar în apelul funcţiei.

Observaţie

Parametri actuali trebuie să corespundă ca ***număr***, ***ordine*** şi ***tip*** cu parametri formali.

Exemplu

Să se calculeze şi să se afişeze media aritmetică a două numere reale **`x`** şi **`y`**. Se va folosi o funcţie ***`calcul`*** care primeşte ca parametri două numere reale **`x`** şi **`y`**, calculează media lor aritmetică şi afişează rezultatul obţinut.

Parametri transmişi prin valoare

```
1   #include<iostream>
2   using namespace std;
3
4   void calcul(float x, float y)
5   {
6       float ma;
7       ma=(x+y)/2;
8       cout<<"media="<<ma<<endl;
9   }
10
11  int main()
12  {
13      float x,y;
14      cout<<"x=";cin>>x;
15      cout<<"y=";cin>>y;
16      calcul(x,y);
17      calcul(3,4);
18      calcul(x+y,x-y);
19      return 0;
20  }
```

parametri formali – parametri transmişi prin valoare

parametri actuali (efectivi)

```
void calcul(float x, float y)
```

```
     calcul(     3.0   ,   4.25)
```

5. Parametri transmişi prin referinţă

În cadrul parametrilor transmişi prin referinţă, în lista de parametri formali, declararea parametrului este precedată de simbolul **ampersand** (sau **epershand,** "**&**"). În acest caz, blocul apelant transmite blocului apelat adresa la care este stocat în memorie parametrul actual.

În cazul parametrilor transmişi prin referinţă, parametri formali sunt parametri de intrare-ieşire, deoarece sunt folosiţi pentru a transmite date dinspre blocul apelant către blocul apelat, dar şi invers, la terminarea execuţiei, blocul apelat transmite date către blocul apelant.

Exemplu

Să se calculeze şi să se afişeze media aritmetică a două numere reale **`x`** şi **`y`**. Se va folosi o funcţie **`calcul`** care primeşte ca parametri două numere reale **`x`** şi **`y`** şi care furnizează prin al treilea parametru media lor aritmetică.

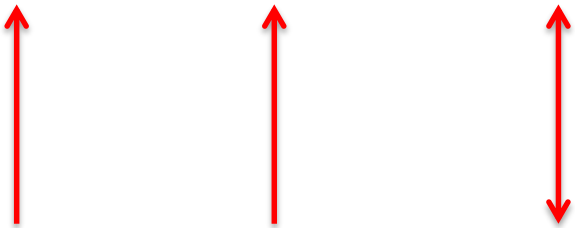
Parametri transmişi prin referinţă

```
1   #include<iostream>
2   using namespace std;
3
4   void calcul(float x, float y, float &ma)
5   {
6       ma=(x+y)/2;
7   }
8
9   int main()
10  {
11      float x,y,ma;
12      cout<<"x=";cin>>x;
13      cout<<"y=";cin>>y;
14      calcul(x,y,ma);
15      cout<<"media="<<ma;
16      return 0;
17  }
```

parametri transmişi prin valoare

parametru transmis prin referinţă

```
void calcul(float x, float y, float &z)
```



```
     calcul(float a, float b, float  c)
```

6. Funcţii care returnează o valoare

Funcţiile care returnează o valoare se numesc şi ***funcţii operand***.

O ***funcţie operand*** este un subprogram care returnează un rezultat prin chiar numele său, şi eventual şi alte rezultate, prin intermediul parametrilor.

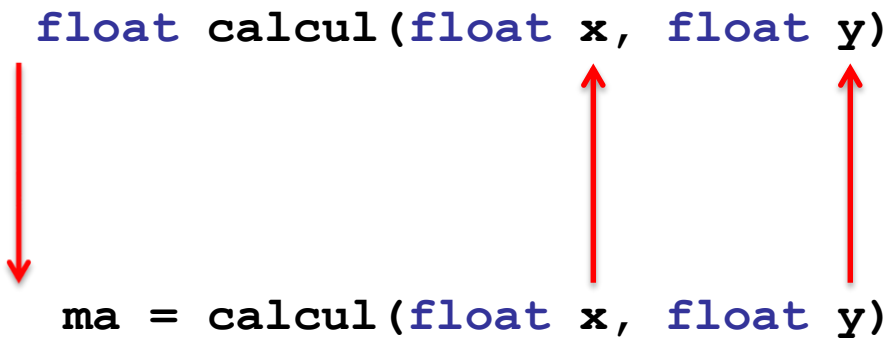
Tipul unei funcţii operand este dat de tipul valorii returnate de către funcţie.

Exemplu

Să se calculeze şi să se afişeze media aritmetică a două numere reale **`x`** şi **`y`**. Se va folosi o funcţie **`calcul`** care primeşte ca parametri două numere reale **`x`** şi **`y`** şi care returnează media lor aritmetică.

Funcţii care returnează o valoare

```
1  #include<iostream>
2  using namespace std;
3
4  float calcul(float x, float y)
5  {
6      float ma;
7      ma=(x+y)/2;
8      return ma;
9  }
10
11 int main()
12 {
13     float x,y,ma;
14     int ma1;
15     cout<<"x=";cin>>x;
16     cout<<"y=";cin>>y;
17
18     ma=calcul(x,y);
19     cout<<"media="<<ma<<endl;
20
21     cout<<"media="<<calcul(x,y)<<endl;
22
23     ma1=calcul(x,y);
24         cout<<"media="<<ma1<<endl;
25
26     if(calcul(x,y)>=0)
27         cout<<"rezultat pozitiv";
28      else
29         cout<<"rezultat negativ";
30
31      return 0;
32 }
```


float calcul(float x, float y)
ma = calcul(float x, float y)

Fişă de lucru:

- Aplicaţii subprograme

8. Bibliografie şi webografie

1. Miloşecsu M., *Informatica. Manual pentru clasa a X*, Editura Didactică şi Pedagogică, Bucureşti, 2005
2. Mateescu G, Moraru P., *Informatica. Manual pentru clasa a X*, Editura Donaris, Sibiu, 2006
3. Popescu C., *Culegere de probleme de informatică*, Editura Donaris-Info, Sibiu, 2002
4. Ministerul Educaţiei, Cercetării şi Tineretului, Centrul Naţional pentru Curriculum şi Evaluare în Învăţământul Preuniversitar, *Proba scrisă la informatică. Examenul de bacalaureat – Variante (1-100)* , Bucureşti 2008
5. http://www.cplusplus.com/doc/tutorial/functions/
6. http://www.cplusplus.com/doc/tutorial/functions2/
7. http://infoscience.3x.ro/c++/subprograme.htm