RECURSIVITATE

Procesul recursiv este procesul care, în timpul execuţiei, generează apariţia unor procese identice, aflate în legătură directă cu procesul ce le generează.Un proces poate fi descris printr-un subprogram

În limbajul C++ se pot defini funcții procedurale (void) sau operand. Ambele tipuri de funcții se pot autoapela dar în mod diferit. În cazul funcţiilor procedurale, autoapelul se realizează prin apelul funcţiei respective, din interiorul ei iar în cazul funcțiilor operand se utilizează instrucțiunea return:

```
void f(int n)
{ if (n!=0)
{ cout<<n<<' ';
f(n-1); }
}
int main()
{ f(5);}
```

```
int f (int n)
{ if (n!=0) return n+f(n-1);}int main()
{ cout<<f(5);}
```

Se calculează suma 1+2+3+4+5

Se vor afișa numerele 5 4 3 2 1

Într-o funcție recursivă recurența este realizată prin autoapelul funcției. În algoritmul recursiv sunt necesare două elemente:

- formula de recurență (autoapelul)
- o valoare inițială cunoscută (condiția de terminare a apelului).

Fiecare autoapel al funcției duce spre soluția finală care corespunde valorii inițiale cunoscute.

Exemplu: Factorialul unui număr n. Funcția matematică recursivă a acestui proces este:

fact(n)=n,

Funcția recursivă este:

```
int fact(int n)
{if(n==0)
      return 1;
   else return n*fact(n-1);
}
```

unde n == 0 reprezintă condiția de terminare a autoapelului iar return n*fact(n-1) conține autoapelul.

Pentru memorarea parametrilor, subprogramele folosesc zona de memorie numită stivă. La apelul subprogramului se vor depune în stivă, în ordine, parametrii transmişi și se rezervă spaţiu

pentru variabilele locale. În cazul în care subprogramul se autoapelează pe un al doilea nivel, se depun din nou parametrii transmişi şi se rezervă un nou spaţiu pentru variabilele locale. În acest exemplu, calculele se efectuează la revenirea din apel. Valorile au fost reţinute în stivă, iar la revenire au fost folosite pentru înmulţire.

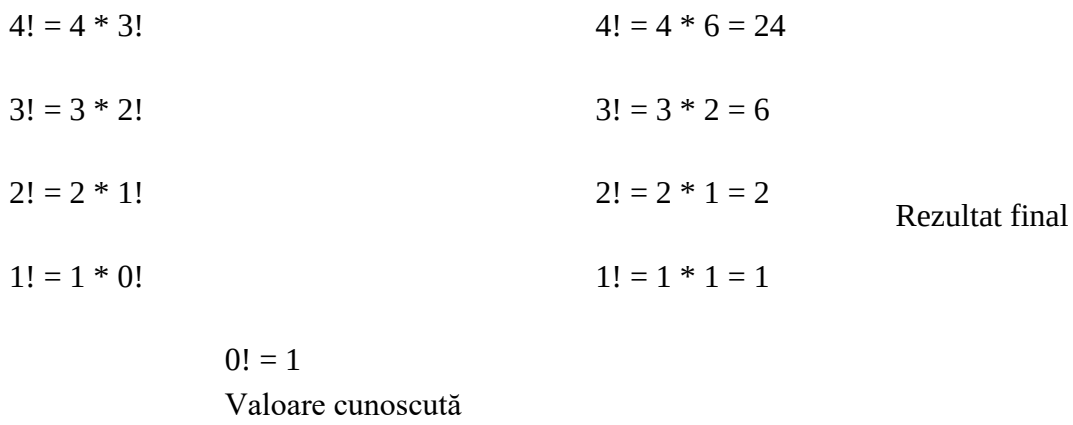

Figura 2.1. Factorialul unui număr n

Regulile pentru construirea unei funcții recursive sunt următoarele:

– Corpul unei funcții recursive cuprinde două elemente :cazul general al soluției care conține operații necesare pentru reducerea dimensiunii problemei, cea mai importantă operație fiind autoapelul. În exemplul funcției fact cazul general al soluției este format din instrucțiunea return n*fact(n-1) și cazul de bază al soluției care conține o operație care rezolvă un caz particular al problemei, în exemplul funcției fact, cazul de bază al soluției este instrucțiunea return 1;

– Într-o funcție recursivă este obligatoriu să existe o condiție de oprire a repetiției exprimată printr-o expresie logică. În exemplul funcției fact , expresia logică este (n= =0). Aceasta expresie logică este folosită pentru a face trecerea de la cazul general la cazul de bază al soluției.

-O funcție recursivă trebuie să asigure condiția de consistență, modificarea parametrilor și sau a variabilelor locale, de la o activare la alta trebuie să asigure condiția de ieșire din recursivitate.

Observații:

-în elaborarea algoritmilor recursivi se aplică raţionamentul că ce se întâmplă la un nivel, se întâmplă la orice alt nivel;

-pentru orice algoritm recursiv există unul iterativ care rezolvă aceeaşi problemă;

-nu întotdeauna alegerea unui algoritm recursiv reprezintă un avantaj.
Exemple de functii recursive:
Suma numerelor naturale <= n

```
int suma (int n)
{ if(n==0)return 0;
   else return suma (n-1) + n; }
int main()
{ int n;
cin>>n;
cout<<suma(n);
return 0;
}
```

Probleme propuse:
-Determinati produsul numerelor pare mai mici sau egale cu n
-Detrminati media aritmetica a numerelor divizibile cu 3 mai mici sau egale cu n

Suma a n numere citite de la tastatura

```
int suma (int n)
{int a;
    if(n==0)return 0;
    else {cin>>a;
        return suma (n-1) + a;}
}
int main()
{
    int n; cin>>n;
    cout<<suma(n);
    return 0;
}
```

Probleme propuse:
-Determinati produsul numerelor cu 2 cifre
-Determinati cate numere impare s-au citit
-Determinati suma numerelor care nu sunt multipli numarului 5

Suma cifrelor unui numar

```
int suma (int n)
{
    if(n==0)return 0;
    else
        return suma (n/10) + n%10;
}
int main()
{
    int n; cin>>n;
    cout<<suma(n);
    return 0;
}
```

Probleme propuse:

-Determinati produsul cifrelor mai mari decat 2
-Determinati cate cifre 0 contine numarul n
-Determinati cifra maxima din n

Suma elementelor unui vector

```
int v[50],n,i;

int vector(int v[50], int n)
{
    if(n==0) return 0;
    else return v[n-1]+vector(v, n-1);
}
int main()
{
    int i;
    cin>>n;
    for (i=0; i<n; i++)
         cin>>v[i];
    cout<<vector(v,n);
    return 0;
}
```

PROBLEME PROPUSE:

1. Să se scrie o **funcție C++ recursivă** care determină factorialul unui număr transmis ca parametru și întoarce rezultatul prin intermediul unui parametru de ieșire.
2. Să se scrie o **funcție C++ recursivă** care determină suma cifrelor unui număr natural `n` transmis ca parametru și întoarce rezultatul prin intermediul unui parametru de ieșire.
3. Să se scrie o **funcție C++ recursivă** care să returneze numărul de cifre egale cu zero ale unui număr natural transmis ca parametru.
4. Să se scrie o **funcție C++ recursivă** cu trei parametri `n`, `k`, `c` și întoarce prin parametrul `c` numărul de cifre ale lui `n` care sunt mai mari sau egale decât `k`.
5. Să se scrie o **funcție C++ recursivă** care să returneze cifra maximă a unui număr natural transmis ca parametru.
6. Să se scrie o **funcție C++ recursivă** care primește ca parametri un număr natural `n` și o cifră `c` și returnează numărul obținut prin eliminarea din `n` a tuturor aparițiilor lui `c`.

PROBLEME VARIANTE BACALAUREAT

1.Fisierul **bac.in** contine pe prima linie un număr natural **n** (**3<n<1000**), iar pe următoarea linie, un sir de **n** numere naturale distincte, de cel mult nouă cifre fiecare. Numerele din sir sunt separate prin câte un spaTiu si cel puTin trei dintre ele au ultima cifră
egală cu **5**.
a) Scrieti un program **C/C++** care citeste toate numerele din fisier si, utilizând un algoritm eficient din punct de vedere al timpului de executare si al memoriei utilizate, determină si afisează pe ecran cele mai mari trei numere din sir care au ultima cifră egală cu **5**. Numerele determinate sunt afisate în ordine crescătoare, separate prin câte un spaTiu.**(6p.)**

Exemplu: dacă fisierul **bac.in** are continutul alăturat, pe ecran se vor afisa, în această ordine, numerele: **15 25 85**
10
97 5 11 1 8 6 85 3 25 15
b) Descrieti succint, în limbaj natural (3-4 rânduri), algoritmul utilizat la punctul **a)** si
justificaTi eficienTa acestuia.

2. Fisierul **BAC.TXT** conTine, **în ordine crescătoare**, cel putin două si cel mult **10000** de numere naturale. Numerele sunt separate prin câte un spaTiu si au cel mult **9** cifre fiecare. Cel putin un număr din fisier apare o singură dată.
a) Scrieti un program **C/C++** care citeste toate numerele din fisier si, printr-un algoritm eficient din punct de vedere al timpului de executare si al memoriei utilizate, determină si afisează pe ecran, în ordine strict crescătoare, separate prin câte un spaTiu, numai numerele care apar o singură dată în fisier. **(6p.)**
Exemplu: dacă fisierul are conTinutul de mai jos
1 1 2 2 2 7 10 10 10 10 21
pe ecran se afisează, în această ordine, numerele **7 21**.

b) Descrieti în limbaj natural (3-4 rânduri) algoritmul utilizat la punctul **a)** si justificati
eficienta acestuia

3. Fisierul **BAC.TXT** contine un sir de cel putin **11** si cel mult un milion de numere naturale, despătite prin câte un spaŃiu. Fiecare număr are cel puŃin două si cel mult nouă cifre. Primul termen al sirului are numărul de ordine **1**, al doilea are numărul de ordine **2** etc. Se citeste sirul din fisier si se cere ca, utilizând un algoritm eficient din punct de vedere al timpului de executare, să se determine si să se afiseze pe ecran numărul de ordine al unui termen al sirului care este precedat în fisier de un număr maxim de valori care au cifra zecilor egală cu a sa. Dacă sunt mai mulŃi termeni cu această proprietate, se afisează numărul de ordine doar al unuia dintre ei.
Exemplu: dacă fisierul **BAC.TXT** contine numerele
12 36 265 18 139 19 32 34 112 14 68
pe ecran se afisează **10** (numărul de ordine al termenului **14**).
a) Descrieti în limbaj natural algoritmul utilizat, justificând eficienta acestuia.
b) Scrieti programul **C/C++** corespunzător algoritmului descris.

4. Se citesc de la tastatură două numere naturale **s1** si **s2** (**0<s1<=18**, **0<=s2<=18**) si se cere scrierea în fisierul **BAC.TXT**, fiecare pe câte o linie, în ordine strict crescătoare, a tuturor numerelor naturale cu exact

5 cifre, pentru care suma primelor două cifre este egală cu **s1**, iar suma ultimelor două cifre este egală cu **s2**. Pentru determinarea numerelor indicate se utilizează un algoritm eficient din punct de vedere al timpului de executare.

Exemplu: dacă **s1=8**, iar **s2=7**, atunci **35725** este unul dintre numerele care respect proprietatea cerută (**3+5=8** si **2+5=7**).

a) DescrieTi în limbaj natural algoritmul utilizat, justificând eficienTa acestuia.

b) ScrieTi programul **C/C++** corespunzător algoritmului descris.

1. Să se determine cea mai mare cifră a unui număr întreg, de cel mult 9 cifre, citit de la tastatură.

```
#include<iostream> //CIFRA MAXIMĂ
using namespace std;
int max(long x);
int main()
 {long n; //n-numărul dat
   cout<<"n=";cin>>n;
   cout<<"cifra maxima:"<<max(n);
 }
int max(long x) //x-numarul citit, transmis ca parametru
 {int a,b;
 if (x==0) //condiţia de opriere a apelului recursiv al funcţiei
    return 0;
  else
   {
 a=x%10; //în variabila a reţinem ultima cifră a numărului
 b=max(x/10); //se reapelează funcţia pentru numărul iniţial din care s-a eliminat
//ultima cifră prin impărţirea la 10, rezultatul se depune în varibila b
 if(a>b) //se compară valorile obţinute în a şi b
 return a; //se returnează valoarea cea mai mare
 else
 return b;
   }
 }
```

2. Să se construiască, utilizând un algoritm recursiv, numarul format prin inversarea cifrelor unui număr natural n de cel mult 9 cifre, citit de la tastatură.

```
#include<iostream> //INVERSARE CIFRE NUMĂR
using namespace std;
void inv(long n,long &m);
int main()
{long n,m=0; //n-numărul de inversat, m-numărul inversat
  cout<<"n=";cin>>n;
 inv(n,m); //apelul funcţiei
  cout<<endl<<m;
```

```
}
void inv(long n,long &m) //m-parametru transmis prin referinţa pentru a prelua valoarea
{ //calculată în funcţie
 if (n!=0) //condiţia de oprire a apelului recursiv
    {m=m*10+n%10;
 inv(n/10,m); //autoapelarea funcţiei
    }
}
```

3. Să se determine cel mai mare divizor comun a două numere naturale citite de la tastatură cu algoritmul lui EUCLID.

```
#include<iostream> //CMMDC
using namespace std;
int cmmdc(int a,int b);
int main()
{int a,b;
  cout<<"a="; cin>>a;
  cout<<"b="; cin>>b;
 cout<<"c.m.m.d.c("<<a<<","<<b<<")="<<cmmdc(a,b); //apelul funcţiei
}
int cmmdc(int a,int b) //a şi b numerele penru care se calculeaza cmmdc
{ if (b) //condiţia de oprirea a reapelului
 return cmmdc(b,a%b); //autoapelarea funcţiei pentru împărţitor şi rest
 else
   return a;
}
```

4. Să se determine al n-lea element al şirului lui Fibonacci.

```
 0 dacă n=0;
fibon=   { 1 dacă n=1;
             fibon-2+fobon-1 în rest

#include<iostream> //FIBONACCI
using namespace std;
int n;
int fibo(int n);
int main()
{int i;
  cout<<"n=";cin>>n;
  for (i=1;i<=n;i++)
    cout<<fibo(i)<<" ";}
int fibo(int n)
{if ((n==1)||(n==2))
   return 1;
 else
```

1.Contorizare cifra 2

```
void cntk(int n, int &k)
{
   if(n!=0)

   {if(n%10==2)

    k=k+1; cntk(n/10,k);}
}
```

2.Factorial prin parametru

```
void fact(int n, int &f)
{
   if(n!=0)

   {

    f=f*n;; fact(n-1,f);}
}
```

3.Suma cifrelor prin parametru

```
void sumcif(int n, int &s)
{
   if(n!=0)

   {

    s=s+n%10; sumcif(n/10,s);}
}
```

4.Cifra maxima

```
int  cifmax(int n)
{ int a,b;
   if(n==0) return 0;
else
   {a=n%10; b=cifmax(n/10);
   if(a>b) return a;
    else return b;

   }
}
```

Comparatie functii recursive

Functiile recursive pot returna rezultatele prin numele functiei (operand) sau pot utiliza un parametru (procedural)

1.Factorialul unui numar

```
#include <iostream>
using namespace std;
long fact(int n)
{
   if(n==0) return 1;
   else return n*fact(n-1);
}
int main()
{  int n;
  cin>>n;
  cout << fact(n);
  return 0;
}
```

```
//rezultat transmis prin parametru
#include <iostream>

using namespace std;
void fact(int n,int &f)
{
   if(n!=0){  f=f*n;
   fact(n-1,f);}
}
int main()
{  int n,f=1;
  cin>>n;
  fact(n,f);
  cout<<f;
  return 0;
}
```

2.Suma cifrelor unui numar

```
#include <iostream>

using namespace std;
int sumcif(int n)
{
   if(n==0) return 0;
   else return sumcif(n/10)+ (n%10);
}
int main()
{   int n;
   cin>>n;
   cout<<sumcif(n);
   return 0;
}
```

```
#include <iostream>

using namespace std;
void sumcif(int n, int &s)
{
   if(n!=0)
      { s=s+n%10;
        sumcif(n/10,s);}
}
int main()
{   int n,s=0;
   cin>>n;
   sumcif(n,s);
   cout<<s;
   return 0;
}
```

3.Determinarea divizorilor

```
//Suma divizorilor
#include <iostream>

using namespace std;

int div(int n,int d)
{

   if(d<=n/2)
   {

    if(n%d==0)
   return div(n,d+1)+d;
   else return div(n,d+1);
}
else return 0;

}
int main()
{
   int n;
   cin>>n;
   cout<<div(n,2);
   return 0;
}
```

```
//Afisarea divizorilor
#include <iostream>

using namespace std;

void div(int n,int d)
{
   if(d<=n/2)
   {

    if(n%d==0)
        cout<<d<<" ";
   div(n,d+1);
}

}
int main()
{
   int n;
   cin>>n;
   div(n,2);
   return 0;
}
```