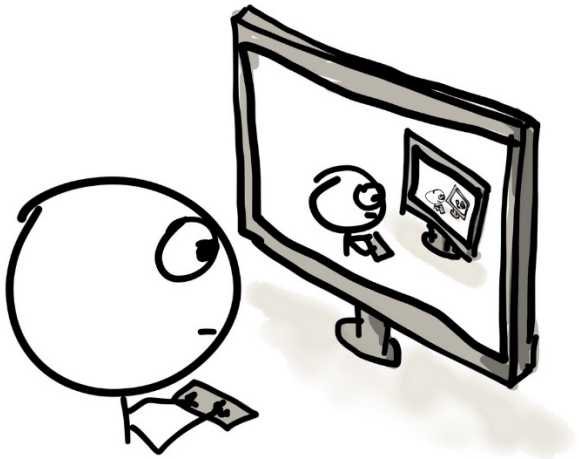
Tehnici de programare

Recursivitate recursivă

ovidiu.banias@aut.upt.ro

Recursivitate



- “A recursive process is one in which

objects are defined in terms of other objects of the same type”

- “A recursive definition is used to define an object in terms of itself ”

- Recursiv – “care poate fi repetat in mod nelimitat”

- O funcție este recursivă dacă se apelează pe ea însăşi pe baza unei formule de recurență

Recursivitate

- www.google.com/ncr : recursion

- Un pom fructifer este un pom care rodeşte

- Un şir de caractere este format prin concatenarea a două şiruri de caractere

- Progresia aritmetică : x[k]=x[k-1]+c, c= ct
- Progresia geometrică : x[k]=x[k-1]*c, c= ct

Recursivitate. Matroska recursivă

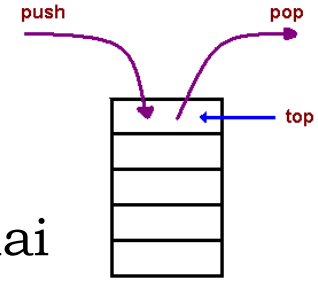
Problemă: valoarea unei păpuşi Matroska este dată de valoarea tuturor păpuşilor ce se află în interiorul ei + 10.

Găsiți o modalitate de calcul şi o formulă de recurență

- Putem calcula valoarea unei păpuşi fără să o deschidem?
- De ce NU?
- Fară a deschide păpuşile pe rând nu se poate afla valoarea primei păpuşi
- Putem face o paralelă între această problemă şi lucrul cu stiva?
- Care este condiția de terminare? Care este formula de recurență?

Recursivitate. Descompunerea în subprobleme

- Care sunt paşii de calcul a unei expresii matematice cu paranteze?



- Se execută operațiile de prioritate maximă, apoi următoarele,...
- **Metodă**: descompunerea problemei de calcul în subprobleme mai mici prin executarea succesivă a operațiilor (subproblemă) în funcție de prioritate
- Rolul recursivității : rezolvă o problemă prin descompunerea ei in subprobleme mai mici si rezolvarea acestor probleme, apoi rezolvarea problemei initziale

- Uneori este mult mai uşor de a formula o rezolvare printr-o formulă de recurență matematică:

$$M(i) = 10 + \sum_{j=i+1}^{n} M(j) \mid i = \overline{1,n}$$

Recursiv vs. Iterativ

```
int Sum(int n)
{
    if (n<2) return 1;
    else return n+Sum(n-1);
}
```

```
int Sum(int n)
{
    int s=0;
    for (i=1;i<=n;i++)
        s=s+i;
    return s;
}
```

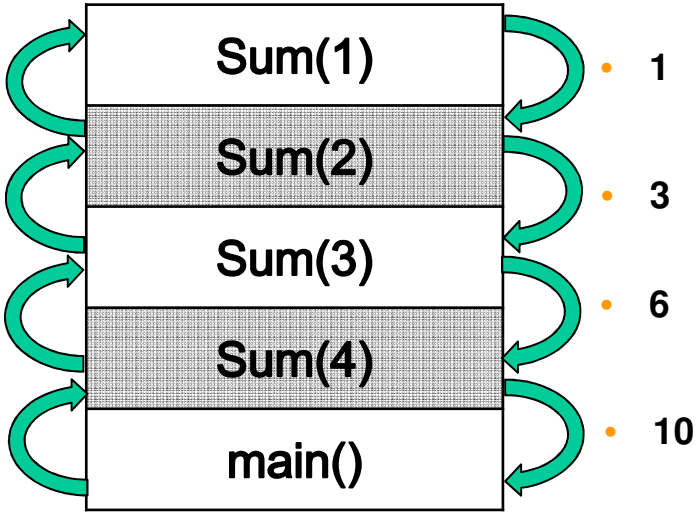
Avantaje și dezavntaje *în funcție de problemă*

- (D) Mai mult cod de scris
- (D) Necesită uneori structuri de date auxiliare
- (A) Nu foloseşte segmentul de stivă
- (A) Mai puțin cod de scris
- (A/D) Foloseşte segmentul de stivă. Folosit fără atenție poate umple stiva → eroare
- (D) Uneori poate fi mai lent decât alg. iterativ

Segmentul de stivă

```
int Sum(int n)
{
  if (n<2) return 1;
  else return n+Sum(n-1);
}

int main(){
    Sum(4);
}
```



Segmentul de stivă (RAM)

- Corpul funcțiilor din seg. de stivă se execută după principiul LIFO (Last In First Out)
- Parametrii funcției recursive + variabilele locale se salvează în memorie în segmentul de stivă
- !!! Foarte mare atenție ca algoritmul să nu cicleze la infinit → Stiva se umple → Segmentation fault
- Trebuie să existe o condiție de terminare : (n<2) în cazul de față
- Care ar fi efectul adăugării unei variabile int k în corpul funcției?

Exemplificare pe cod

- Suma primelor n numere naturale
- Calcul Factorial(n)
- Calcul suma cifrelor unui număr
- Calcul funcție Manna-Pnueli
- Calcul Cmmdc(a,b)
- Calcul Fibonacci(n)
- Inversarea cifrelor unui număr

Exemplu de recursivitate ineficientă

Recursiv

```
int Fibo(int n)
{
  if (n>2)
    return Fibo(n-1)+Fibo(n-2);
  else
    return n;
}
```

Iterativ

```
int Fibo(int n)
{
  int v[100];
  v[1]=1;v[2]=2

  for(i=3;i<n;i++)
    v[i]=v[i-1]+v[i-2];

  return v[n];
}
```

- Funcția recursivă este mai uşor de scris şi nu necesită declarații auxiliare de variabile/vectori
- Care este complexitatea celor 2 algoritmi?
- De ce funcția recursivă este mai lentă decât cea iterativă?
- !!! Atenție la problemele în care se fac apeluri redundante de funcții recursive cu aceeaşi parametri