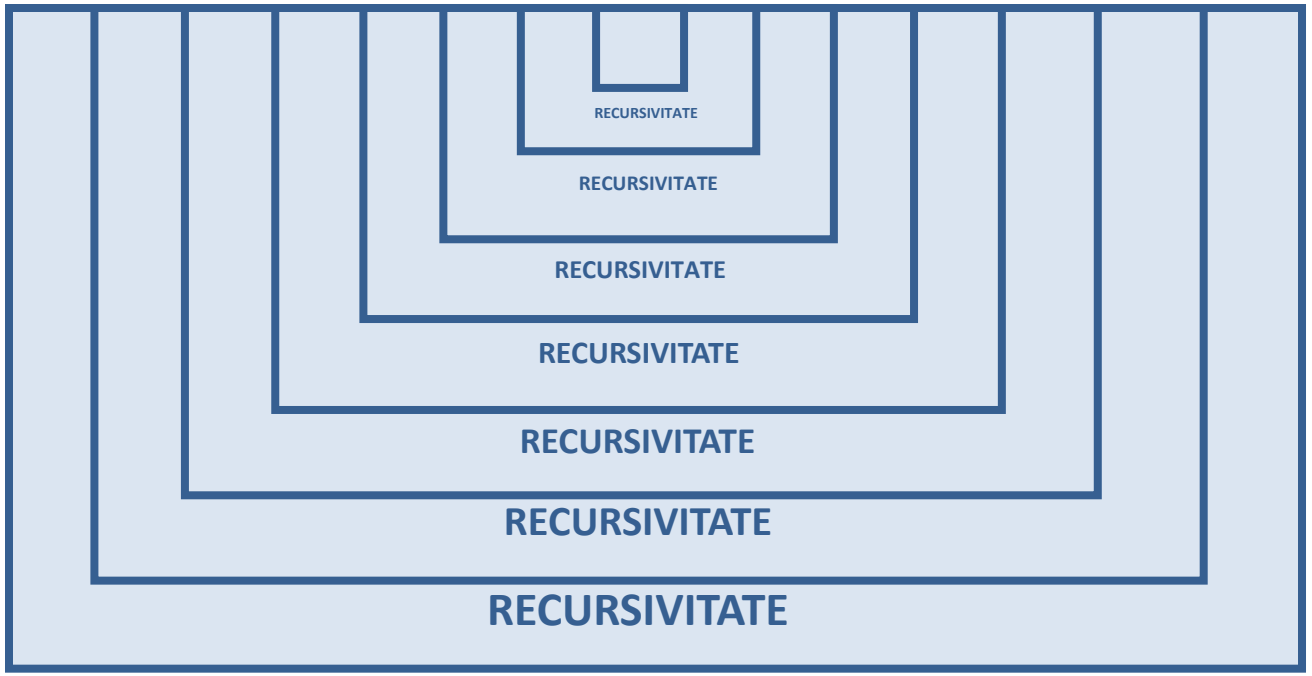
RECURSIVITATE
RECURSIVITATE
RECURSIVITATE
RECURSIVITATE
RECURSIVITATE
RECURSIVITATE
RECURSIVITATE

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*
- *aplicarea mecanismului recursivităţii prin crearea unor subprograme recursive (definite de utilizator)*
- *compararea dintre implementarea recursivă şi cea iterativă a aceluiaşi algoritm*
- analiza problemei în scopul identificării subproblemelor acesteia
- descrierea metodei de rezolvare a unei probleme în termeni recursivi

Recursivitatea este un mecanism general de scriere a problemelor.

Un algoritm recursiv se caracterizează prin proprietatea că se autoapelează, adică din interiorul lui se apelează pe el însuşi.
Atunci când se scrie un algoritm recursiv este suficient să se gândească ce se întâmplă la un anumit nivel, pentru că la orice nivel se întâmplă exact acelaşi lucru.

Într-un algoritm recursiv, pentru a realiza un anumit calcul sunt necesare două elemente:

1. o formulă de recurenţă;
2. o valoare iniţială cunoscută.

Un exemplu de recursivitate este în definirea formală a numerelor naturale din cadrul teoriei mulțimilor:

- baza recursiei este faptul că 1 este număr natural;
- în plus, orice număr natural are un succesor, care este de asemenea un număr natural.

Un alt exemplu ar fi definirea conceptului de strămoş al unei persoane:

- un părinte este strămoșul copilului (“baza”);
- părinții unui strămoș sunt și ei strămoși (“pasul de recursie”).

• recurenţă = repetiţie, revenire;

• formulă de recurenţă = formulă care exprimă orice termen dintr-un şir, în funcţie de termenii precedenţi;

Exemplu

Să se calculeze suma primelor n numere naturale.

$S(n)=1+2+3+\ldots+(n-1)+n$

Formula matematică:

$S(n)=S(n-1)+n$ // formula de recurenţă

$S(0)=0$ //valoarea iniţială cunoscută

$$S(n) = \begin{cases} 0, & \text{pentru } n = 0 \\ S(n-1) + n, & \text{pentru } n > 0 \end{cases}$$

$S(4)=1+2+3+4=10$

astfel:

$S(4)=S(3)+4$
$S(3)=S(2)+3$
$S(2)=S(1)+2$
$S(1)=S(0)+1$
$S(0)=0$

$\Rightarrow$

$S(0)=0$
$S(1)=S(0)+1=0+1=1$
$S(2)=S(1)+2=1+2=3$
$S(3)=S(2)+3=3+3=6$
$S(4)=S(3)+4=6+4=10$

Un algoritm recursiv se implementează folosind o funcţie recursivă.

Se numeşte ***funcţie recursivă*** o funcţie care din corpul ei se apelează pe ea însăţi. Un subprogram se numeşte ***recursiv*** dacă se autoapelează.

Orice funcţie recursivă trebuie să îndeplinească două condinţii:

1. să se poată executa cel puţin o dată fără să se autoapeleze;
2. toate apelurile să se producă astfel încât să se tindă spre îndeplinirea condiţiei de execuţie fără apeluri.

Din afara funcţiei recursive, se face un prim apel la funcţia recursivă, după care funcţia se autoapelează de un anumit număr de ori.

Pentru orice algoritm iterativ există un algoritm recursiv echivalent (rezolvă acceaşi problemă) şi invers, pentru orice algoritm recursiv există unul iterativ echivalent.

Exemplu

Să se calculeze suma primelor **n** numere naturale.

$$S(n)=1+2+3+...+(n-1)+n$$

```
1   #include <iostream>
2
3   using namespace std;
4
5   int S(int n)
6   {
7       if(n==0)
8           return 0;
9       else
10          return S(n-1)+n;
11  }
12
13  int main()
14  {
15      cout <<S(4)<< endl;
16      return 0;
17  }
18
```

Autoapelarea se poate realiza în două moduri:

1. *direct* – în corpul funcţiei apare explicit un apel recursiv;
2. *indirect* – în corpul funcţiei apare apelul unei alte funcţii care, la rândul său, apelează direct sau indirect funcţia recursivă.

Recursivitate directă

```
void A()
{
      …
      A();
      …
}

void main()
{
      …
      A();
      …
}
```

Recursivitate indirectă

```
void A()
{
      …
      B();
      …
}

void B()
{
      …
      A();
      …
}

void main()
{
      …
      A();
      …
}
```

Fişă de lucru

- Subprograme recursive recursive
- Aplicaţii subprograme reczursive

5. 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://ro.wikipedia.org/wiki/Recursivitate
6. http://en.wikipedia.org/wiki/Recursion