

Recursia

Material didactic pentru Informatică
(În corespondență cu curriculum-ul la Informatică)
Clasa a XI-a

Obiectivele lecției:

- O1-** să cunoască noțiunea de apel recursiv și modul de execuție a algoritmilor recursivi;
- O2-** să cunoască avantajele și neajunsurile recursiei;
- O3-** să explicați modul de alocare a memoriei și de transmitere a controlului la execuția algoritmilor recursivi;
- O4-** să elaboreze programe, în care se utilizează recursia.

Recursia

se definește ca o situație, în care un subprogram se autoapelează, fie direct, fie prin intermediul altei funcții sau proceduri.

Subprogramul care se autoapelează se numește recursiv.

Recursia

este utilă pentru programarea unor calcule repetitive. Repetiția este asigurată prin execuția unui subprogram, care conține un apel la el însuși: când execuția ajunge la acest apel, este declanșată o nouă execuție ș.a.m.d.

Tipuri de recursivitate

Subprograme direct recursive – un subprogram Q în corpul căruia există cel puțin un autoapel (Q apelează pe Q) se numește subprogram direct recursiv.

Subprograme indirect recursive - două subprograme A și B se numesc indirect recursive dacă se apelează reciproc.

Important!!!

Orice *subprogram* *recursiv* trebuie să includă *condiții de oprire* a procesului repetitiv.

Reguli pentru scrierea programelor recursive

În procesul derulării calculelor trebuie să existe:

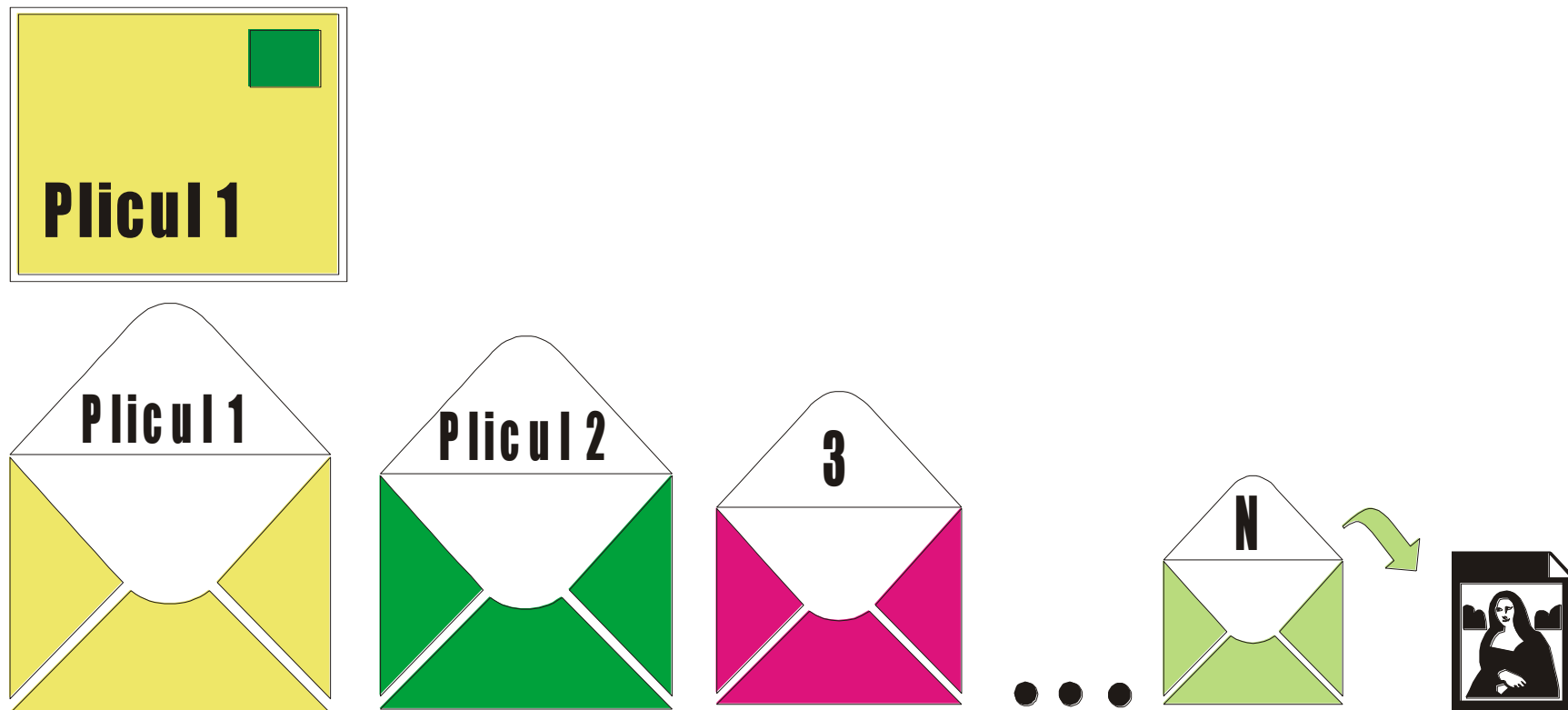
- Cazuri elementare, care se rezolvă direct;
- Cazuri care nu se rezolvă direct, însă procesul de calcul în mod obligatoriu progresează spre un caz elementar.

Important!!!

La orice apel de subprogram în memoria calculatorului vor fi depuse următoarele informații:

- valorile curente ale parametrilor transmiși prin valoare;**
- locațiile (adresele) parametrilor-variabilă;**
- adresa de retur, adică adresa instrucțiunii ce urmează după apel.**

Mecanismul general



Recursia - un proces care se realizează prin apelarea unei forme mai simple ale sale.

M a r i a G u t u

Descriere

Fie P o problemă în care cere calculul valorii v ,

$v \in Q = \{Q_0, Q_1, Q_2, \dots, Q_k, \dots, Q_n, \dots\}$

Q_0 , este cunoscut, sau poate fi calculat direct.

Oricare $Q_i \in Q$ poate fi exprimat prin elementul Q_{i-1} și o expresie calculabilă.

Soluția recursivă

Se determină formula de calcul a elementului Q_i exprimat prin Q_{i-1} .

Se exprimă consecutiv Q_i prin Q_{i-1}
 Q_{i-1} prin Q_{i-2}
...
 Q_1 prin Q_0

Se calculează (dacă e necesar) Q_0

Se calculează Q_1 folosind Q_0
 Q_2 folosind Q_1
...
 Q_{n-1} folosind Q_{n-2}
 Q_n folosind Q_{n-1}

Example

Alcătuiți o funcție recursivă ce calculează n^k .

$$n^k = n * n^{k-1}$$

$$n^0 = 1$$

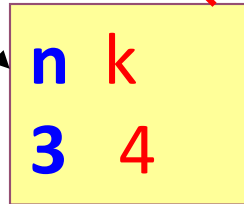
$$3^4 = 3 * 3^3$$

$$3^3 = 3 * 3^2$$

$$3^2 = 3 * 3^1$$

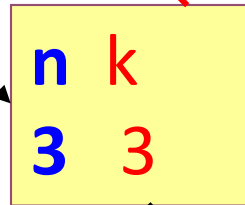
$$3^1 = 3 * 3^0$$

$$3^0 = 1$$

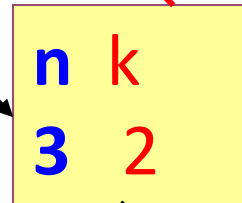


$\Leftarrow 81$

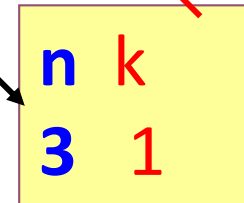
$\Leftarrow 27$



$\Leftarrow 9$



$\Leftarrow 3$



$\Leftarrow 1$



M a r i a G u t t u

Example

Funcția recursivă ce calculează n^k este:

```
Function F (k:Integer):Longint;  
Begin  
  If k=0 Then F:=1  
    Else F:=n*F(k-1);  
End;
```

Example

Alcătuiți o funcție recursivă ce calculează factorialul unui număr natural N .

Example

Funcția recursivă ce calculează factorialul unui număr natural N.

```
Function F (N:Integer):Longint;  
Begin  
If N=0 Then F:=1  
    Else F:=F(N-1)*N  
End;
```


Example

Alcătuiți o funcție recursivă ce determină numărul Fibonacci de pe locul N .

Numerele Fibonacci sunt definite prin următoare relație de recurență:

$$F_0 = 0, F_1 = 1, F_i = F_{i-1} + F_{i-2}, \text{ pentru } i \geq 2$$

Astfel, fiecare număr **Fibonacci** este suma celor două numere Fibonacci anterioare, rezultând secvența:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

```
program fibonacci_iterativ;  
Var n,z,x,y,i:Integer;  
begin  
  Readln(n);  
  if n=0 then y:=0 else  
  begin  
    x:=0;  
    y:=1;  
    for i:=1 to n-1 do  
    begin  
      z:=x+y;  
      x:=y;  
      y:=z;  
    end;  
  end;  
  Writeln('Al ',n,' lea numar fibonacci este ',y);  
end.
```

Example

Funcția recursivă ce determină numărul Fibonacci de pe locul N.

*Function **Fib** (N:Integer):Longint;*

Begin

*If N=0 Then **Fib**:=0*

*Else If N=1 Then **Fib**:=1*

*Else **Fib**:=**Fib**(N-1)+**Fib**(N-2)*

End;

Studiul comparativ al iterativității și recursivității

Nr	Caracteristici	Iterativitate	Recursivitate
1	Necesarul de memorie		
2	Timpul de execuție		
3	Structura programului		
4	Volumul de muncă necesar pentru scrierea programului		
5	Testarea și depănarea programului		

Lucrare practică

M a r i a G u t u

Problema Nr. 1

Scrieți un subprogram recursiv care calculează suma primilor N termeni:

$$S(n)=1+2+3+4+...+n.$$

Intrare: Numărul N se citește de la tastatură.

Ieșire: Rezultatul se afișează la ecran.

Problema Nr. 2

Scrieți o funcție recursivă ce inversează un șir de caractere.

Intrare: Șirul S se citește de la tastatură.

Ieșire: Șirul inversat se afișează la ecran.

Problema Nr. 2

```
Program String_invers;  
Var s: string;  
      i: integer;  
Function sir(i1: integer):integer;  
Begin  
  if i1=0 then sir="" else sir:=s[i1]+sir(i1-1);  
End;  
Begin  
  Writeln('Introdu sirul'); readln(s);  
  i:=length(s);  
  Writeln('sirul inversat este: ', sir(i));  
End.
```


Concluzii

La **apeluri recursive**, **spațiul ocupat din memorie va crește rapid, crescând depășirea capacității de memorare a calculatorului. Astfel de cazuri pot fi evitate, înlocuind recursia prin iterație.**

Concluzii

Recursia este deosebit de utilă în cazurile în care, elaborarea unor algoritmi nerecursivi este foarte complicată.

Extindere: nivel 1

Scrieți un subprogram recursiv care calculează suma, produsul primilor N termeni:

$$S(n)=1+3+5+\dots+(2n-1);$$

$$S(n)=2+4+6+\dots+2n;$$

$$P(n)=1 \times 3 \times 5 \times \dots \times (2n-1);$$

$$P(n)=2 \times 4 \times 6 \times \dots \times 2n.$$

Extindere: nivel 2

- 1. Fie A un vector. Scrieți un subprogram recursiv care va calcula elementul minim din acest vector.**
- 2. Fie B o matrice cu n linii și m coloane. Scrieți un subprogram recursiv care va calcula:**
 - a) elementul maxim din tablou;**
 - b) elementu minim de pe ultima coloană.**
- 3. Fie X un vector ce conține numele elevilor din clasă. Scrieți un subprogram recursiv care va afișa lungimea maximă a numelui din șir.**

Literatura recomandată:

1. Gremalschi, A.(2008). Informatică: Manual pentru cl. A 11-a, Editura Știința. 192 p.
2. Braicov, A.(2005). Turbo Pascal: culegere de probleme, Editura Prut Internațional. 232 p.
3. Sacara, A.(2012). Informatică: culegere de probleme pentru clasele a IX-a – a XII-a, Editura Epigraf. 88 p.
4. Creangă-Andrunache, E.(2001). Informatica: probleme Pascal, Editura Paragon. 219 p.
5. Cercez, E.(2002). Informatica: culegere de probleme pentru liceu, Editura Polirom. 237 p.

Mult succes!

M a r i a G u t u