Recursia

Material didactic

M a r i a G u t u

Obiectivele lecției:

O1- să cunoască noţiunea de apel recursiv şi modul de execuţie a algoritmilor recursivi;
O2- să cunoască avantajele şi neajunsurile recursiei;
O3- să explicaţi modul de alocare a memoriei şi de transmitere a controlului la execuţia algoritmilor recursivi;
O4- să elaboreze programe, în care se utilizează recursia.

Recursia

se defineşte ca o situaţie, în care un subprogram se autoapelează, fie direct, fie prin intermediul altei funcţii sau proceduri.

Subprogramul care se autoapelează se numeşte recursiv.

Recursia

este utilă pentru programarea unor calcule repetitive. Repetiţia este asigurată prin execuţia unui subprogram, care conţine un apel la el însuşi: când execuţia ajunge la acest apel, este declanşată o nouă execuţie ş.a.m.d.

Tipuri de recursivitate

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

Subprograme indirect recursive - două subprograme A şi 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!!!

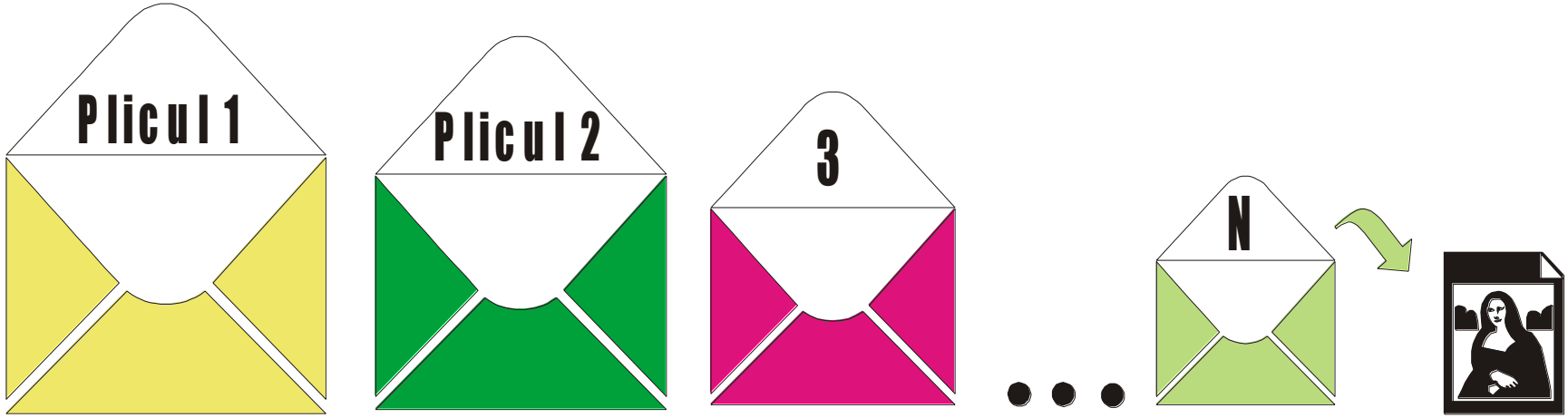
La orice apel de subprogram în memoria calculatorului vor fi depuse următoarele informaţii:

- valorile curente ale parametrilor transmişi prin valoare;

- locaţiile (adresele) parametrilor-variabilă;

- adresa de retur, adică adresa instrucţiunii ce urmează după apel.

Mecanismul general



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

Descriere

Fie P o problemă în care cere calculul valorii v,

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

Q_0, este cunoscut, sau poate fi calculat direct.

Oricare $Q_i \in Q$ poate fi exprimat prin elementul Q_{i-1} şi 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

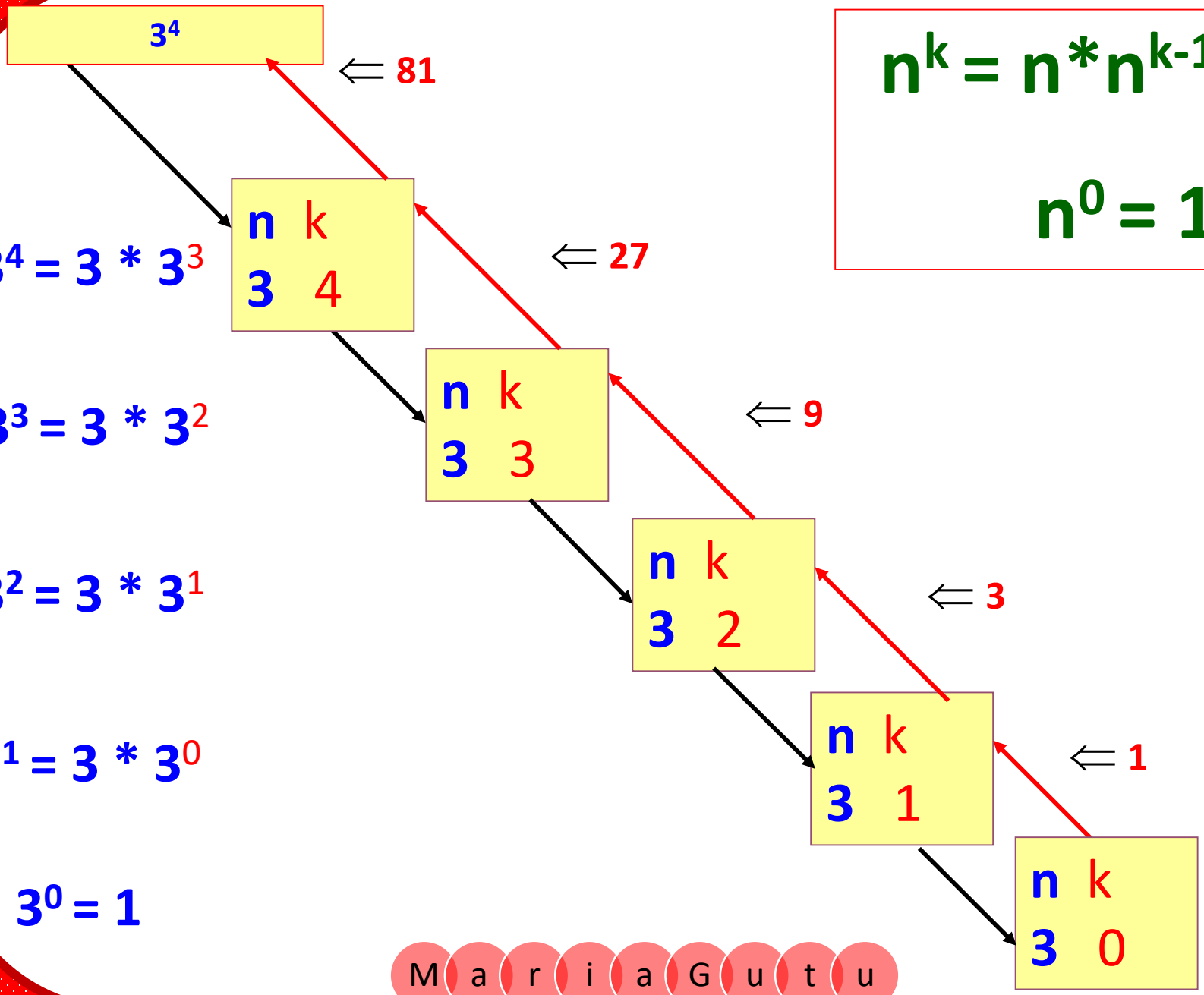
...

Q_{n-1} folosind Q_{n-2}

Q_n folosind Q_{n-1}

Exemple

Alcătuiți un program recursiv ce calculează n^k .


3^4
⇐ 81
n k
3 4
$3^4 = 3 * 3^3$
⇐ 27
n k
3 3
$3^3 = 3 * 3^2$
⇐ 9
n k
3 2
$3^2 = 3 * 3^1$
⇐ 3
n k
3 1
$3^1 = 3 * 3^0$
⇐ 1
n k
3 0
$3^0 = 1$
$n^k = n*n^{k-1}$
$n^0 = 1$
M a r i a G u t u

Exemple

Programul recursiv ce calculează n^k:

```
#include <stdio.h>
int putere(int n, int k){
        if (k==0){ return 1;}
        else { return n * putere(n, k-1); }
}
int main(){
int a = 2, b = -2;
int p = putere(a, b);
printf("p=%g",p);
return 0;}
```

Exemple

Programul recursiv ce calculează n^k:

```
#include <stdio.h>
float putere(int n, int k){
        if (k==0){ return 1;}
        else {
                if ( k > 0 ) { return n * putere(n, k-1);}
                else {
                        if (k<0){return (1./n * putere(n, k+1));}
                }       }}
int main(){
int a = 2, b = -2;
float p = putere(a, b);
printf("p=%g",p);
return 0;}
```

Exemple

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

Exemple

Funcția recursivă ce calculează factorialul unui număr natural N.

```
#include <stdio.h>
        long factorial(int n){
                if (n==1){ return 1;}
                else { return n * factorial(n-1);}
}
int main(){
        int x = 4;
        long f = factorial(x);
        printf("f=%ld",f);
return 0;}
```

Studiul comparativ al iterativităţii şi recursivităţii

Nr	Caracteristici	Iterativitate	Recursivitate
1	Necesarul de memorie	Mai scăzut	Mai înalt
2	Timpul de execuţie		Mai eficient
3	Structura programului		Mai compact
4	Volumul de muncă necesar pentru scrierea programului		
5	Testarea şi depănarea programului		

Lucrare practică

M a r i a G u t u

Problema Nr. 1

Scrieți un program recursiv care calculează suma primilor N termeni:

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

Intrare: Numărul N se citește de la tastatură.
Ieşire: Rezultatul se afişează la ecran.

Concluzii

La apeluri recursive, spaţiul ocupat din memorie va creşte rapid, crescând depăşirea capacităţii de memorare a calculatorului. Astfel de cazuri pot fi evitate, înlocuind recursia prin iteraţie.

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+...+(2n-1);

S(n)=2+4+6+...+2n;

P(n)=1 x 3 x 5 x ... x (2n-1);

P(n)=2 x 4 x 6 x ... x 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.**

Mult succes!

M a r i a G u t u