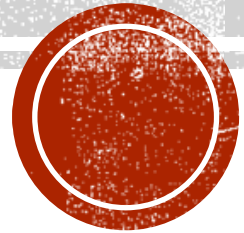
ITERATIVITATE SAU RECURSIVITATEA

CARLASUC SABINA CL.XI ,,C''

I.P.T.L ,,SPIRU HARET''

PROFESOR: GUTU MARIA



CUPRINS:

- Introducere
- Studiu comparativ
- Avantajele si dezavantajele
- Exemple subprograme recursive
- Exemple subprograme iterative

MULTE PROBLEME DE O REALĂ IMPORTANŢĂ PRACTICĂ POT FI REZOLVATE CU AJUTORUL UNOR METODE STANDARD, DENUMITE TEHNICI DE PROGRAMARE:

- Recursia (directa si indirecta)
- trierea
- metoda reluării
- metodele euristice

Studiul comparativ al iterativității şi recursivității este reprezentat în tabel

nr. crt	Caracteristici	Iterativitate	Recursivitate
1	Necesarul de memorie	mic	Mare
2	Timpul de execuție	Acelaş	
3	Structura programului	Complicată	Simplă
4	Volumul de muncă	mare	Mic
5	Testarea şi depănarea	simplă	complicată

AVANTAJELE\DEZAVANTAJELE:

- Dimensiunea stivei trebuie aleasa astfel incat sa poata permite memorarea elementelor pentru toate iteratiile.
- Din punctul de vedere al memoriei solicitate, o varianta recursiva necesita un spatiu de stiva suplimentar pentru fiecare apel fata de varianta iterativa.
- Recursivitatea poate fi inlocuita prin iteratie atunci cand recursivitatea este prea adanca sau cand limbajul de programare nu permite implementarea de apeluri recursive.
- Iterativitatea minimizeaza complexitatea unor algoritmi.
- Deseori, variantele iterative necesita folosirea explicita a structurii de tip stiva, generand astfel un cod extrem de laborios. In aceste situatii se considera solutia recursiva mai eleganta, datorita simplitatii sale.
- Exista o legatura stransa intre recursivitate si structurile de date de tip stiva, arbore, etc folosite in limbajele Borland Pascal si C++ pentru reprezentarea functiilor/procedurilor recursive (insasi definitia stivelor, arborilor, listelor realizandu-se recursiv).

EXEMPLE SUB.ITERATIVE:

1) Calculul sumei

```
…
unsigned n;
void citire()
{cout<<"n=";cin>>n;}
float suma(unsigned n)
{unsigned i;
long p=1;
float s=1;
for(i=1;i<=n;i++)
{p*=2;
s+=(float)1/p;}
return s;}
void main()
{clrscr();
citire();
cout<<"suma pentru n="<<n<<"
este="<<suma(n);
getch();}
```

2) Egalitatea a 2 string-uri

```
var a,b:string;
egal:boolean;
i:byte;
begin
writeln('introduceti sirurile :');
readln(a); readln(b);
egal:=true;
if length(a)<>length(b) then egal:=false
else
for i:=1 to length(a) do
if a[i]<>b[i] then egal:=false;
if egal then writeln('sunt egale')
else writeln('nu sunt egale')
end.
```

3) Suma cifrelor unui numar

…

```
int suma(int n)
{ int s=0;
while(n!=0)
{ s=s+n%10;
n=n/10;}
return s;}
void main()
{ int n;
cout<<" n = "; cin>>n;
int n1=n;
cout<<" Suma cifrelor numarului
"<<n1<<" = "<< suma(n);}
```

EXEMPLE

SUB.RECURSIVE:

1) Câte perechi de iepuri se pot obţine în n luni dintr-o singură pereche ştiind că:

la momentul iniţial, iepurii din prima pereche sunt nou-născuţi;

fiecare nouă pereche de iepuri devine fertilă după o lună;

fiecare pereche produce o pereche de descendenţi în fiecare lună;

nici un iepure nu moare.

Numărul perechilor de iepuri din fiecare lună este descris de şirul:

Luna 0,1,2,3,4,5,6,7,8

Nr. perechi 1,1,2,3,5,8,13,21,34

Notăm cu U(n) numărul perechilor din luna n. Atunci avem:

1, n=0

U(n) = 1, n=1

U(n-1)+U(n-2), n>=2

Acest şir este chiar şirul lui Fibonacci.

Algoritmul *recursiv* decurge astfel: pentru calculul lui fib(n) sunt calculaţi fib(n-1) şi fib(n-2), ai căror parametri sunt depuşi pe stivă. De exemplu, pentru a calcula fib(10) se calculează fib(9) şi fib(8); fib(9) ca sumă de fib(8) şi fib(7), după care se calculează fib(8) încă o dată. Să ne imaginăm, pentru fib(800), de câte ori se calculează fib(3) ! Deoarece metoda implică determinarea, cu salvările pe stivă aferente, a aceloraşi valori în mod repetat, este de preferat, evident, varianta iterativă, mult mai naturală în această situaţie.

2) Se consideră şirurile definite recurent astel:

A0=a; b0=b; a,b>0:

Să se scrie un program care să citească a,b şi n şi să se calculeze an şi bn.

```
double a,b;
int n;
double bn(int n);
double an(int n)
{if (!n)return a;
else return (an(n-1)+bn(n-1))/2;}
double bn(int n)
{if (!n) return b;
else return sqrt(an(n-1)*bn(n-1));
void main()
{cout<<"a=";
cin>>a;
cout<<"b=";
cin>>b;
cout<<"n";
cin>>n;
cout<
```

Totuşi , fiecare apel al unui suprogram recursiv inseamnă încă o zonă de memorie rezervată pentru execuţia subprogramului (variabilele locale şi instrucţiunile).

3) Calculul Sumei

```
...
unsigned n;
void citire()
{cout<<"n=";cin>>n;}
long putere2(unsigned n)
{if(n==0) return 1;
else return 2*putere2(n-1);}
float suma(unsigned n)
 {if(n==0) return 1;
else
return (float)1/putere2(n)+suma(n-1);}
void main()
{clrscr();
citire();
cout<<"suma pentru n="<<n<<"
este="<<suma(n);
 getch();}
```

4) Egalitatea a 2 string-uri

```
var a,b:string;
function egal(a,b:string):boolean;
begin
if length(a)<>length(b) then egal:=false
else
if a[1]<>b[1] then egal:=false
else
if length(a)=1 then egal:=true
else
egal:=egal(copy(a,2,length(a)-1),
copy(b,2,length(b)-1))
end;
begin
writeln('introduceti sirurile :');
readln(a); readln(b);
if egal(a,b) then writeln('sunt egale')
else writeln('nu sunt egale')
```

5) Suma cifrelor unui numar

```
…
int suma(int n)
{ if(n==0) return 0;
else return n%10 + suma(n/10);}
void main()
{ int n;
clrscr();
cout<<" n = "; cin>>n;
int n1=n;
cout<<" Suma cifrelor numarului
"<<n1<<" = "<< suma(n);}
```

CONCLUZIE:

In alegerea intre metoda recursiva si iterativa in elaborarea unui program trebuie sa se tina seama de :

- eficienta oferita programului de catre fiecare dintre variante,
- relatia dintre timpului de rulare si spatiului de memorie necesar
- nevoia de compactizare a programului.

Alegerea variantei recursive / iterative pentru scrierea unui program presupune:

- cunoasterea fiecarei metode in parte si a particularitatilor sale;
- cunoasterea tehnicii de transformare a recursivitatii in iteratie;
- studiu profund al structurilor de date optime reprezentarii datelor problemei;
- stapanirea tuturor facilitatilor oferite de limbajul de programare in care va fi implementat algoritmul.

BIBLIOGRAFIE:

- http://fmi.unibuc.ro/ro/pdf/2017/admitere/licenta/FMI_Subprograme_si_recursivitate_2017.pdf
- file:///D:/Users/D.Doctor/Downloads/XI_Informatica%20(in%20limba%20romana).pdf
- http://muhaz.org/nvmntul-profesional-si-tehnic-n-domeniul-tic.html?page=3