CURS 2:

Descrierea algoritmilor în pseudocod

=Exemple=

Structura

- Descrierea unor algoritmi simpli
- Specificarea și utilizarea subalgoritmilor

Exemplu 1

Considerăm un tabel cu informații despre studenți

Nr.	Nume	Note			Credite	Stare	Medie
1	A	8	6	7	60		
2	B	10	10	10	60		
3	C	-	7	5	40		
4	D	6	-	-	20		
5	E	8	7	9	60		

Problema: să se completeze coloanele stare și medie folosind regulile

stare = 1 dacă Credite=60 (obs: 1 credit=30 ore de activitate)

stare= 2 dacă Credite este in [30,60)

stare= 3 dacă Credite <30

media aritmetică a notelor se calculează doar dacă Credite =60

Exemplu 1

După completare tabelul va arăta astfel:

Nr.	Nume	Note			Credite	Stare	Medie
1	A	8	6	7	60	1	7
2	B	10	10	10	60	1	10
3	C	-	7	5	40	2	-
4	D	6	-	-	20	3	-
5	E	8	7	9	60	1	8

stare = 1 dacă Credite=60

stare= 2 dacă Credite este in [30,60)

stare= 3 dacă Credite <30

Exemplu 1

Ce fel de date vor fi prelucrate?

Nr.	Nume	Note			Credite	Stare	Medie
1	A	8	6	7	60		
2	B	10	10	10	60		
3	C	-	7	5	40		
4	D	6	-	-	20		
5	E	8	7	9	60		

Date de intrare: Note si Credite

note[1..5,1..3] : tablou bidimensional (matrice) cu 5 linii și 3 coloane

Descriere în pseudocod: int note[1..5,1..3]

Exemplu 1

Ce fel de date vor fi prelucrate?

Nr.	Nume	Note			Credite	Stare	Medie
1	A	8	6	7	60		
2	B	10	10	10	60		
3	C	-	7	5	40		
4	D	6	-	-	20		
5	E	8	7	9	60		

Date de intrare: Note și Credite

credite[1..5] : tablou unidimensional cu 5 elemente

Descriere in pseudocod: int credite[1..5]

Exemplu 1

Ce fel de date vor fi prelucrate?

Nr.	Nume	Note			Credite	Stare	Medie
1	A	8	6	7	60		
2	B	10	10	10	60		
3	C	-	7	5	40		
4	D	6	-	-	20		
5	E	8	7	9	60		

Date de ieșire: Stare si Medie

stare[1..5], medie[1..5] : tablouri unidimensionale cu 5 elemente

Descriere pseudocod: int stare[1..5]

real medie[1..5]

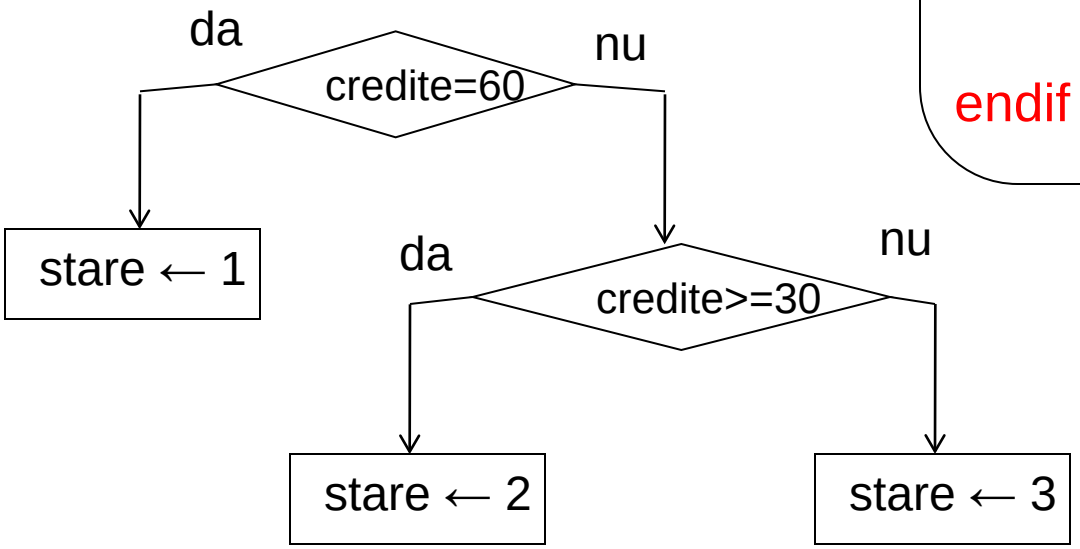
Exemplu 1

Regula pt.determinarea stării asociate unui student

stare = 1 dacă credite=60

stare= 2 dacă credite in [30,60)

stare= 3 dacă credite <30



Descriere în pseudocod:

```
if credite==60 then stare = 1
    else if credite>=30 then stare = 2
            else stare = 3
        endif
endif
```

Descriere in Python

```
if credite==60:
        stare=1
elif credite>=30:
        stare=2
else:
        stare=3
```

Exemplu 1

Completarea stării pentru toți studenții

Obs: Numărul studenților este notat cu n (în exemplul analizat n=5)

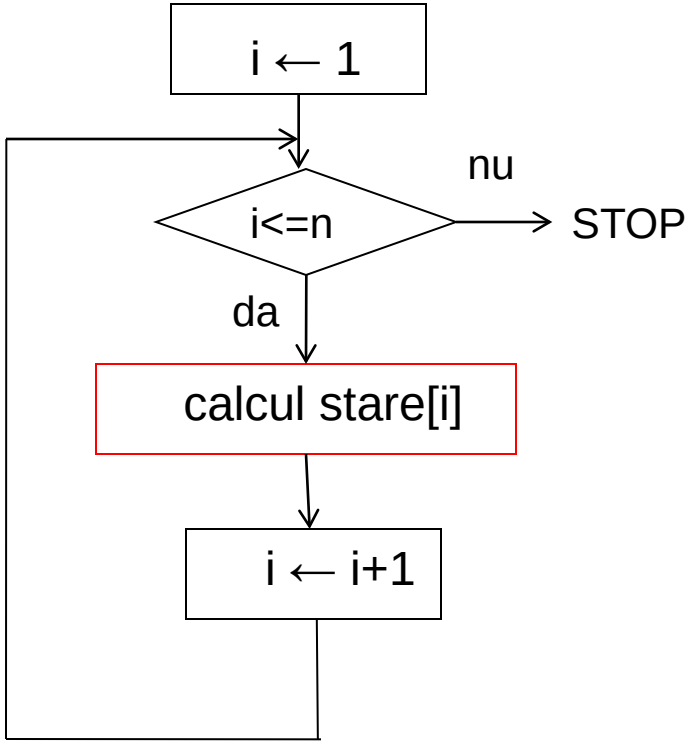
Pas 1: se pornește de la prima linie din tabel (i ← 1)

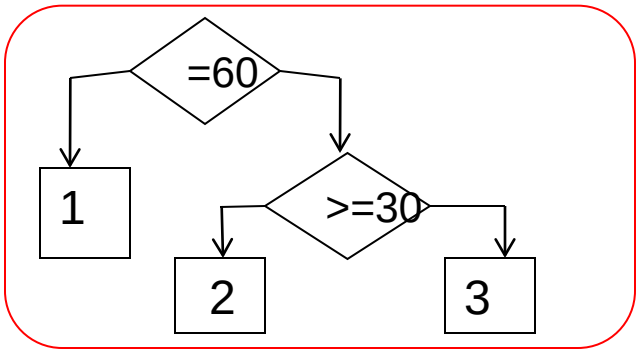
Pas 2: se verifică dacă mai sunt linii de prelucrat (i<=n); dacă nu, se oprește prelucrarea

Pas 3: se calculează starea elementului i

Pas 4: se pregătește indicele următorului element (i ← i+1)

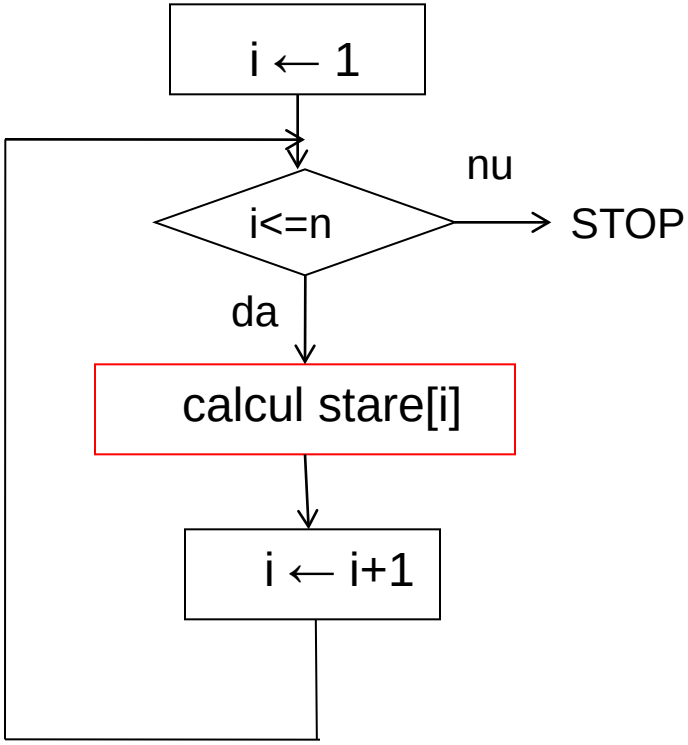
Pas 5: se continuă cu Pas 2





Exemplu 1

Completarea stării pentru toți studenții



Pseudocod:

```
int credite[1..n], stare[1..n], i
i =1
while i<=n do
    if credite[i]==60 then stare[i] =1
      else if credite[i]>=30 then stare[i] =2
                else stare[i] = 3
            endif
    endif
    i = i+1
endwhile
```

Exemplu 1

Simplificarea descrierii algoritmului prin gruparea unor prelucrări in cadrul unui subalgoritm

Pseudocod:

```
int credite[1..n], stare[1..n], i
i = 1
while i<=n do
  stare[i] = calcul(credite[i])
  i = i+1
endwhile
```

Descriere subalgoritm (modul / funcție /procedură/ rutină):

```
calcul (int c)
  int stare
  if c==60 then stare = 1
     else if c>=30 then stare = 2
              else stare = 3
           endif
  endif
 return stare
```

Obs: un subalgoritm descrie un calcul efectuat asupra unor date generice numite parametri

Utilizarea subalgoritmilor

Idei de bază:

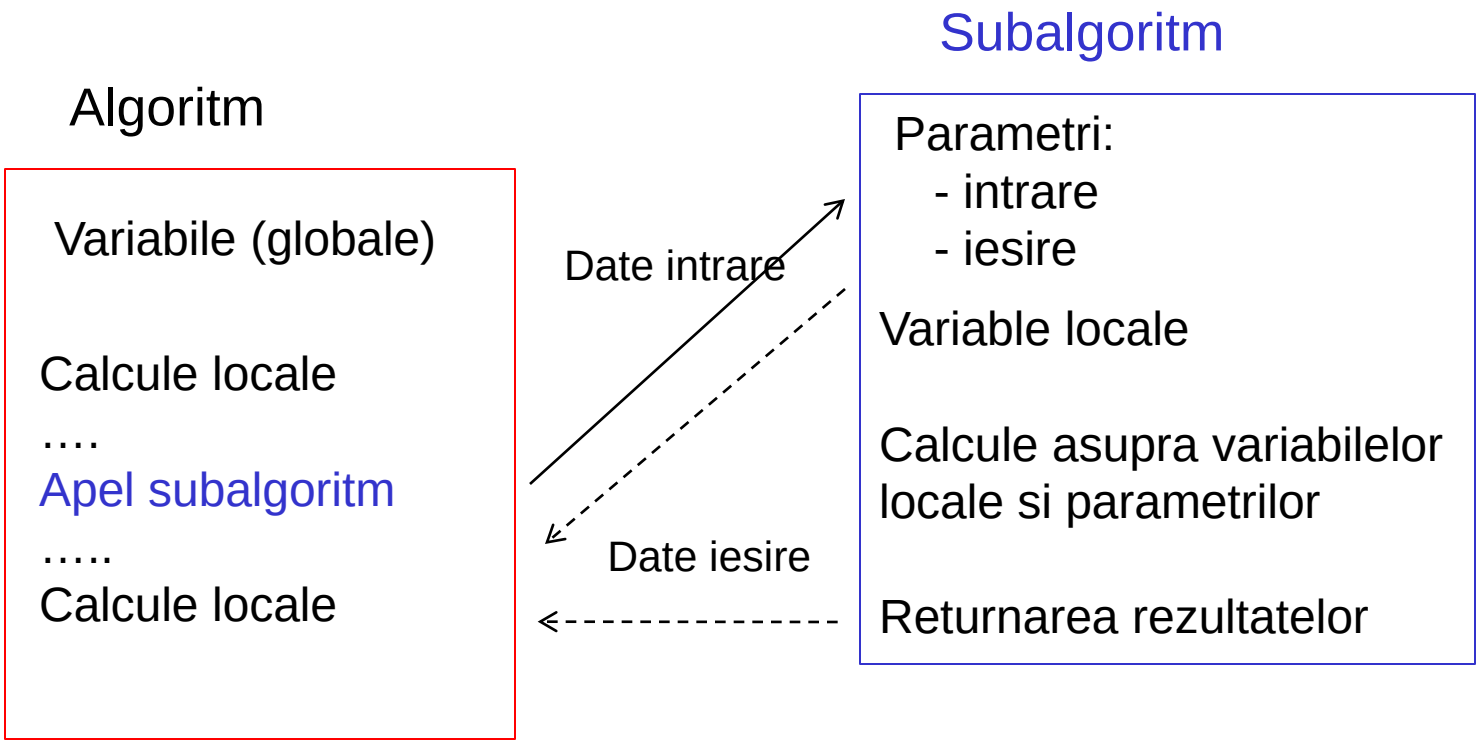
- Problema inițială se descompune în subprobleme
- Pentru fiecare subproblemă se proiectează un algoritm (numit subalgoritm sau modul sau funcție sau procedură)
- Prelucrările din cadrul subalgoritmului se aplică unor date generice (numite parametri) și eventual unor date ajutătoare (numite variabile locale)
- Prelucrările specificate în cadrul subalgoritmului sunt executate în momentul apelului acestuia (când parametrii generici sunt inlocuiți cu valori concrete)

- Efectul unui subalgoritm constă în :
  - Returnarea unuia sau a mai multor rezultate
  - Modificarea valorilor unor parametri (sau a unor variabile globale)

Utilizarea subalgoritmilor

Mecanismul de comunicare intre algoritm si subalgoritmi:

- parametri și valori returnate



Utilizarea subalgoritmilor

Mecanismul de comunicare intre algoritm si subalgoritmi:

- parametri si valori returnate

Algoritm

```
int credite[1..n], stare[1..n], i
i ← 1
while i<=n do
  stare[i] ←calcul(credite[i])
 i ← i+1
endwhile
```

Date intrare

Date iesire

Subalgoritm

```
calcul (int c)
  int stare
  if c==60 then stare = 1
    else if c>=30
            then stare = 2
            else stare = 3
          endif
  endif
 return stare
```

Param. intrare

Var. locala

Rezultat de returnat

Utilizarea subalgoritmilor

- Structura unui subalgoritm:

  <nume subalgoritm> (<parametri formali (generici)>)

  < declarații ale variabilelor locale >

  < prelucrări >

  RETURN <rezultate>

- Apelul unui subalgoritm:

  <nume subalgoritm> (<parametri efectivi>)

Înapoi la Exemplul 1

Pseudocod:

```
int credite[1..n], stare[1..n], i
i = 1
while i<=n do
  stare[i] = calcul(credite[i])
  i = i+1
endwhile
```

Alta variantă:

```
int credite[1..n], stare[1..n], i
for i = 1,n do
  stare[i] = calcul(credite[i])
endfor
```

Subalgoritm (functie) :

```
calcul (int c)
  int stare
  if c==60 then stare = 1
     else if c>=30 then stare = 2
              else stare = 3
           endif
  endif
 return stare
```

Înapoi la Exemplul 1

Python:

```
credite=[60,60,40,20,60]
stare=[0]*5
n=5
i=0
while i<n:
  stare[i]=calcul(credite[i])
  i=i+1
print stare
```

Utilizare for in loc de while:

```
for i in range(5):
  stare[i]=calcul(credite[i])
```

Functie Python:

```
def calcul(c):
    if c==60:
      stare=1
    elif c>=30:
      stare=2
    else:
      stare=3
    return stare
```

Obs: indentarea liniilor este foarte importantă în Python

Înapoi la Exemplul 1

Nr.	Nume	Note			Credite	Stare	Medie
1	A	8	6	7	60	1	
2	B	10	10	10	60	1	
3	C	-	7	5	40	2	
4	D	6	-	-	20	3	
5	E	8	7	9	60	1	

Calcul medie: pentru studenții având starea=1 trebuie calculată media aritmetică a notelor

Notele studentului i se află pe linia i a matricii note (se pot specifica prin note[i,1..m])

Înapoi la Exemplul 1

Calculul mediei

```
int note[1..n,1..m], stare[1..n]
real medie[1..n]
…
for i = 1,n do
  if stare[i]==1
     medie[i] = calculMedie(note[i,1..m])
  endif
endfor
```

Linia i a matricii = tablou uni-dimensional

Funcție pt calcul medie

```
calculMedie(int  v[1..m])
real suma
int i
suma = 0
for i = 1,m do
    suma = suma+v[i]
endfor
suma = suma/m
return suma
```

Înapoi la Exemplul 1

Calculul mediei (exemplu Python)

```
note=[[8,6,7],[10,10,10],[0,7,5],[6,0,0],[8,7,9]]
stare=[1,1,2,3,1]
medie=[0]*5
for i in range(5):
  if stare[i]==1:
    medie[i]=calculMedie(note[i])
print medie
```

Obs: `range(5) = [0,1,2,3,4]`

In Python indicii tablourilor încep de la 0

Functie pt. calculul mediei (Python)

```
def calculMedie(note):
    m=len(note)
    suma=0
    for i in range(m):
        suma = suma+note[i]
    suma=suma/m
    return suma
```

Pauză ... de ciocolată

Am o tabletă de ciocolată pe care doresc să o rup în bucățele (în cazul unei tablete 4x6 sunt 24 astfel de bucățele). Care este numărul de mișcări de rupere necesare pentru a separa cele 24 de bucățele ? (la fiecare mișcare pot rupe o bucată în alte două bucăți – doar de-a lungul uneia din liniile separatoare ale tabletei)

Pauză ... de ciocolată

Am o tabletă de ciocolată pe care doresc să o rup în bucățele (în cazul unei tablete 4x6 sunt 24 astfel de bucățele). Care este numărul de mișcări de rupere necesare pentru a separa cele 24 de bucățele ? (la fiecare mișcare pot rupe o bucată în alte două bucăți – doar de-a lungul uneia din liniile separatoare ale tabletei)

Răspuns: 23 (în cazul unei tablete de mxn numărul de mișcări este mxn-1)

Cum putem demonstra ?

Pauză ... de ciocolată

Prin inducție matematică (pentru o tabletă cu N=nxm bucățele)

Caz particular: bucata întreagă (1) nu necesită nici o rupere (0)

Ipoteză: Prespunem că pentru orice K<N sunt necesare și suficiente K-1 mișcări.

Pentru a obține N bucăți se procedează astfel:

- Se rupe tableta în două bucăți (cu K_1<N respectiv K_2<N bucățele, $K_1+K_2=N$) – o mișcare
- Se rupe fiecare dintre cele două bucăți în bucățele ($K_1-1+K_2-1=K_1+K_2-2$ mișcări)
- Total: $K_1+K_2-2+1=K_1+K_2-1=N-1$ mișcări

Exemplu 2 – cel mai mare divizor comun

Problema: Fie a și b două numere reale. Să se determine cel mai mare divizor al lui a și b: cmmdc(a,b)

Metoda lui Euclid (varianta bazată pe împărțiri):

- Se calculează restul r al împărțirii lui a (deîmpărțit) la b (împărțitor)
- Inlocuiește
  - valoarea deimpărțitului (a) cu valoarea împărțitorului (b),
  - valoarea împărțitorului (b) cu valoarea restului r și calculează din nou restul împărțirii lui a la b
- Procesul continuă până se obține un rest egal cu 0
- Restul anterior (care este evident diferit de 0) va fi cmmdc(a,b).

Exemplu 2 – cel mai mare divizor comun

Cum funcționează metoda?

1: $a=bq_1+r_1$, $0<=r_1<b$

2: $b=r_1q_2+r_2$, $0<=r_2<r_1$

3: $r_1=r_2q_3+r_3$, $0<=r_3<r_2$

…

i: $r_{i-2}=r_{i-1}q_i+r_i$, $0<=r_i<r_{i-1}$

…

n-1: $r_{n-3}=r_{n-2}q_{n-1}+r_{n-1}$, $0<=r_{n-1}<r_{n-2}$

n : $r_{n-2}=r_{n-1}q_n$, $r_n=0$

Observații:

- la fiecare pas deîmpărțitul ia valoarea vechiului împărțitor iar noul împărțitor ia valoarea vechiului rest

- secvența de resturi este un șir strict descrescător de numere naturale, astfel că există o valoare n astfel încât $r_n=0$ (metoda este finită)

- utilizând aceste relații se poate demonstra ca r_{n-1} este într-adevar cmmdc(a,b)

Exemplu 2 – cel mai mare divizor comun

Cum funcționează metoda?

1: $a=bq_1+r_1$, $0<=r_1<b$

2: $b=r_1q_2+r_2$, $0<=r_2<r_1$

3: $r_1=r_2q_3+r_3$, $0<=r_3<r_2$

…

i: $r_{i-2}=r_{i-1}q_i+r_i$, $0<=r_i<r_{i-1}$

…

n-1: $r_{n-3}=r_{n-2}q_{n-1}+r_{n-1}$, $0<=r_{n-1}<r_{n-2}$

n : $r_{n-2}=r_{n-1}q_n$, $r_n=0$

Demonstratie:

- din ultima relație rezultă că r_{n-1} divide pe r_{n-2} , din penultima relație rezultă că r_{n-1} divide pe r_{n-3} s.a.m.d.
- rezultă astfel că r_{n-1} divide atât pe a cât și pe b (deci este divizor comun)
- pt a arăta că r_{n-1} este cmmdc considerăm că d este un alt divizor comun pentru a și b; din prima relație rezultă că d divide r_1 ; din a doua rezultă că divide pe r_2 s.a.m.d.
- din penultima relație rezultă că d divide pe r_{n-1}

Deci orice alt divizor comun il divide pe r_{n-1} => r_{n-1} este cmmdc

Exemplu 2 – cel mai mare divizor comun

Algoritm
(varianta WHILE):

```
cmmdc(int a,b)
int d,i,r
d = a
i = b
r = d MOD i
while r!=0 do
    d = i
    i = r
    r = d MOD i
endwhile
return i
```

Algoritm :
(varianta REPEAT)

```
cmmdc(int a,b)
int d,i,r
d = a
i = b
repeat
    r = d MOD i
    d = i
    i = r
until r=0
return d
```

Exemplu 2 – cmmdc al unei secvențe de valori

- Problema: să se determine cmmdc al unei secvențe de numere naturale nenule

- Exemplu:

  cmmdc(12,8,10)=cmmdc(cmmdc(12,8),10)=cmmdc(4,10)=2

- Date de intrare: secvența de valori $(a_1,a_2,..., a_n)$
- Date de ieșire (rezultat): cmmdc $(a_1,a_2,..., a_n)$

- Idee:

  Se calculează cmmdc al primelor două elemente, după care se calculează cmmdc pentru rezultatul anterior și noua valoare …

  … e natural să se utilizeze un subalgoritm care calculează cmmdc

Exemplu 2 – cmmdc al unei secvențe de valori

- Structura algoritmului:

```
cmmdcSecventa(int a[1..n])
int d,i
d = cmmdc(a[1],a[2])
for i = 3,n do
  d = cmmdc(d,a[i])
endfor
return d
```

```
cmmdc (int a,b)
int d,i,r
d = a
i = b
r = d MOD i
while r<>0 do
      d = i
      i = r
      r = d MOD i
endwhile
return i
```

Exemplu 3: problema succesorului

Se consideră un număr constituit din 10 cifre distincte. Să se determine elementul următor din secvența crescătoare a numerelor naturale constituite din 10 cifre distincte.

Exemplu: x= 6309487521

Data de intrare: tablou unidimensional cu 10 elemente ce conține cifrele numărului: [6,3,0,9,4,8,7,5,2,1]

Care este următorul număr (în ordine crescătoare) ce conține 10 cifre distincte?

Răspuns:

6309512478

Exemplu 3: problema succesorului

Pas 1. Determină cel mai mare indice i având proprietatea că

x[i-1]<x[i] (se consideră că prima cifră, adică 6, are indicele 1)

Exemplu: x= 6309487521 i=6

Pas 2. Determină cel mai mic x[k] din subtabloul x[i..n] care este mai mare decât x[i-1]

Exemplu: x=6309487521 k=8

Pas 3. Interschimbă x[k] cu x[i-1]

Exemplu: x=6309587421 (aceasta valoare este mai mare decât cea anterioară)

Pas 4. Sortează x[i..n] crescător (pentru a obține cel mai mic număr care satisface cerințele)

Exemplu: x=6309512478 (este suficient să se inverseze ordinea elementelor din x[i..n])

Exemplu 3: problema succesorului

Subprobleme / subalgoritmi:

Identifica: Identifică poziția i a celui mai din dreapta element x[i], care este mai mare decât vecinul său stâng (x[i-1])

Input: x[1..n]
Output: i

Minim: determină indicele celui mai mic element din subtabloul x[i..n] care este mai mare decat x[i-1]
Input: x[i-1..n]
Output: k

Inversare: inversează ordinea elementelor din x[i..n]
Input: x[i..n]
Output: x[i..n]

Exemplu 3: problema succesorului

Structura generala a algoritmului:

```
Succesor(int x[1..n])
int  i, k
i = Identifica(x[1..n])
if i==1
then write “nu exista succesor !"
else
   k = Minim(x[i-1..n])
   x[i-1]↔x[k]
   x[i..n] = Inversare(x[i..n])
   write x[1..n]
endif
```

Observație: In general interschimbarea valorilor a două variabile necesită 3 atribuiri și utilizarea unei variabile auxiliare (la fel cum schimbarea conținutului lichid a două pahare necesită utilizarea unui alt pahar)

```
a ↔b
```

este echivalent cu

```
aux = a
a = b
b = aux
```

Exemplu 3: problema succesorului

```
Identifica(int x[1..n])
int i
i = n
while (i>1) and (x[i]<x[i-1]) do
    i = i-1
endwhile
return i
```

```
Minim(int x[i-1..n])
int j
k = i
for j = i+1,n do
  if  x[j]<x[k] and x[j]>x[i-1] then
      k = j
  endif
endfor
return k
```

Exemplu 3: problema succesorului

```
inversare (int x[left..right])
   int i,j
   i = left
   j = right
  while i<j DO
        x[i]↔x[j]
         i = i+1
         j = j-1
   endwhile
   return x[left..right]
```

Exemplu 3: implementare Python

```
def identifica(x):
  n=len(x)
  i=n-1
  while (i>0)and(x[i-1]>x[i]):
    i=i-1
  return i

def minim(x,i):
  n=len(x)
  k=i
  for j in range(i+1,n):
    if (x[j]<x[k])and (x[j]>x[i-1]):
      k=j
  return k
```

```
def inversare(x,left,right):
  i=left
  j=right
  while i<j:
    x[i],x[j]=x[j],x[i]
    i=i+1
    j=j-1
  return x
```

Obs. In Python interschimbarea a două variable a și b poate fi realizată prin

```
a,b=b,a
```

Exemplu 3: implementare Python

```
# apelul functiilor definite anterior
x=[6,3,0,9,4,8,7,5,2,1]
print "secventa cifrelor din numarul initial:",x

i=identifica(x)
print "i=",i

k=minim(x,i)
print "k=",k
x[i-1],x[k]=x[k],x[i-1]

print "secventa dupa interschimbare:",x
x=inversare(x,i,len(x)-1)
print "secventa dupa inversare:",x
```

Sumar

- Problemele se descompun in subprobleme cărora li se asociază subalgoritmi
- Un subalgoritm este caracterizat prin:
  - Nume
  - Parametri
  - Valori returnate
  - Variabile locale
  - Prelucrări
- Apelul unui subalgoritm:
  - Parametrii sunt înlocuiți cu valori concrete
  - Prelucrările din algoritm sunt executate

Cursul următor…

- Cum se poate verifica corectitudinea algoritmilor
- Introducere in verificarea formală a corectitudinii algoritmilor

Intrebare de final

```
x = 4
y = 6
while y>0 do
   x = x+1
   y = y-1
endwhile
```

Ce valoare va avea variabila x
    după execuția algoritmului de
    mai sus ?

Variante de răspuns:

a) 5
b) 4
c) 6
d) 10
e) 2