INFORMATICĂ

LIMBAJUL PASCAL

MATERIAL ELABORAT CORESPUNZÂND
CERINȚELOR DE BACALAUREAT 2016

Cuprins

Gândirea algoritmică

Problema căutării

Există şi cazul care, pentru o aceeaşi problem putem prezenta două soluţii, în care una este mai rapidă ca alta. De pildă, fie un şir de numere natural oarecare, de exemplu 4,2,10,1,8,15,7. Vream să testăm dacă un număr dat (să zivem numărul 8) se află în această secvenţă sau nu.

O astfel de căutare, prin parcurgerea de a stânga la dreapta a întregii secvenţe de numere, până se găseşte numărul dorit sau se epuiează toate elementele din secvenţă, se numeşte căutarea secvenţială. Dacă avem un şir S de n elemente (notat S[1..n]), atunci căutarea secvenţială a lui x în S se descrie prin:

```
Căutare_secventială(x,S[1..n]) înseamnă
Început
  Fie elemental_curent = primul_element;
  Atăt timp cât (elemental_curent <> x) si (pozitia elementului current <= pozitia
ultimului element (n)) execută
    Început
      Dacă elemental_curent = x atunci mesaj('găsit')
      Altfel treci la următorul element
    Sfârşit
Sfârşit.
```

În continuare să presupunem că numerele erau déjà ordonate crescător 1, 2, 4, 7, 8, 10, 15. În acest caz particular, căutarea secvenţială a unui număr cum este 8 nu este prea eficientă, deoarece 8 se află în a doua jumătate a secvenţei, deci ar fi de preferat să nu-l căutăm în prima jumătate. Acest procedeu este mai rapid:

- Dacă numărul din mijloc este mai mic decât numărul căutat, atunci căutăm a doua jumătate;
- Dacă numărul din mijloc este mai mare ca numărul căutat, atunci căutăm în prima jumătate;
- Dacă numărul din mijloc este egal cu numărul căutat, înseamnă că am găsit numărul în cauză şi trebuie să oprim căutarea.

Căutarea în jumătatea aleasă se face tot la fel, deci se va înjumătăţi şi această zonă...

Procedeul anterior se numeşte căutare binară, deoarece, de fiecare dată, o secvenţă de numere este divizată în două jumătăţi. Putem descrie căutarea binară a unui număr x într-o secvenţă S[1..n] astfel.

```
Căutarea_binară(x, S[1..n]) înseamnă
Început
 Stabileste elemental_curent ca fiind elemental din mijlocul secventei;
 Dacă n=1 atunci
  Dacă x = elemental_curent atunci mesaj('găsit')
  Altfel mesaj('negăsit')
 Altfel
  Început
   Împarte secventa în cele două jumătăti (S[1..mijloc] si S[mijloc+1..n]);
   Dacă elemental_curent = x atunci mesaj('găsit')
   Altfel
    Dacă elemental_curent < x atunci Căutare_binară(x,S[mijloc+1..n])
    Altfel căutarea_binară(x.S[1..mijloc])
  Sfârşit
Sfârşit.
```

Problema intrării într-un cabinet medical

Dacă un om vrea să intre pentru consult la un medic, atunci, mai întâi va bate la uşă: dacă medicul răspunde prin „poftim!", atunci va intra, atlfel va aştepta până când pacientul dinăuntru va ieşi; abia după ce cabinetul va fi liber, va intra.

```
Intrare_la_medic înseamnă
 Început
  Bate la_uşă;
  Dacă răspunsul = ‚poftim!' atunci
   Întră_în_cabinet;
  Altfel
   Început
    Atât timp cât cabinetul_este_ocupat_de_alt_pacient execută
     Citeste_un_ziar;
   Sfârşit
Sfârşit.
```

O intrucţiune compusă este formată dintr-o secvenţă de intrucţiuni, încadrate de cuvinte început şi sfârşit, care pot conţine, la rândul lor, blocuri de alternativă şi repetiţie.

Programarea este tehnica realizării de algoritmi descrişi prin proceduri şi programe. Ea devine o artă, atunci când se folosesc cele trei elemente de structurate şi se numeşte programare structurată. Pentru a vedea ce ar însemna programare nestructurată, să reconsiderăm procedura de intrare în cabinetul medical:

```
Intare_la_medic înseamnă
Început
  Bate_la_uşă;
  Dacă răspunsul = ‚poftim!' atunci intră_în_cabinet;
  Altfel Început
    Atât timp cât cabinetul_este_ocupat_de_alt_pacient execută
      Citeste_un_ziar;
    Întră_în_cabinet
  Sfârşit
Sfârşit.
```

Tipuri de date

Într-o declaraţie de forma procedure ordonare(n), n reprezintă argumentul procedurii; adică procedura ordonare ordonează n elevi, unde s se va preciza ulterior. Noi convenim ca n să fie număr natural, chiar dacă declaraţia de mai sus nu rezultă acest lucru. În limbajele evoluate de programare, fiecare argument, fiecare variabilă are un anumit tip bine definit, adică poate lua valori dintr-o mulţime precizată de valori.

În algoritmii simpli, putem folosi următoarele tipuri de date:

- Integer = mulţimea numerelor întregi
- Real = mulţimea numerelor reale
- Char = caracter
- String = şir de caractere
- Boolean = logic

Byte	0	255	1 Byte
Word	0	65 535	2 Byte
Shortint	–128	127	1 Byte
Integer	–32 768	32 767	2 Byte
Longint	–2 147 483 648	2 147 483 647	4 Byte

Proceduri

Media aritmetică

Realizaţi un program care calculează şi afişează media aritmetică a două numere reale x şi y.

```
Program Media_aritmetica;
Var
  media,x,y:Real;
Begin
  Write('Dati numerele:');
  ReadLn(x,y);
  Media:=(x+y)/2;
  WriteLn('Media='Media:8:2);
  ReadLn;
End.
```

Cu Procedure...

```
Program Media_aritmetica;
Var
  media,x,y:Real;

Procedure media_calcul;
Begin
  Media:=(x+y)/2;
  WriteLn('Media=',Media:8:2);
End;

Begin
  Write('Dati numerele:');
  ReadLn(x,y);
  media_calcul;
  ReadLn;
End.
```

Procedura Media_calcul conţine toate acţiunile algoritmului: citirea numerelor, calcul şi afişarea mediei aritmetice.

```
Program Media_aritmetica;
Var
 media,x,y:Real;

Procedure media_calcul;
Begin
 Write('Dati numerele:');
 ReadLn(x,y);
 Media:=(x+y)/2;
 WriteLn('Media=',Media:8:2);
End;

Begin
 media_calcul;
 ReadLn;
End.
```

Ariile unor figuri geometrice

Scrieţi un program care, în funcţie de dorinţa utilizatorului, calculează şi afişează: aria unui pătrat de latură L, sau aria unui cerc de rază r, sau aria unui triunghi cu baza b şi înălţimea h. Pentru calcul şi afişarea fiecăreia din cele trei arii, se va folosi câte o procedură.

```
Program aria;
Const
 pi=3,14159;
Var
 opt:Integer;

Procedure aria_patrat;
Var
 L:Integer;
Begin
 Write('Latura L=');
 ReadLn(L);
 If L>0 Then
  WriteLn('Aria =',(L+L):6)
 Else
  WriteLn('Date incorecte!');
End;
```

```
Procedure aria_cerc;
Var
 r:Integer;
Begin
 Write('Raza r=');
 ReadLn(r);
 If r>0 Then
  WriteLn('Aria =',(pi*r*r))
 Else
  WriteLn('Date incorecte!');
End;

Procedure aria_triunghi;
Var
 b,h:Integer;
Begin
 Write('baza b si inalt h= ');
 ReadLn(b,h);
 If (b>0) and (h>0) Then
  WriteLn('Aria -', (b*h)/2)
 Else
  WriteLn('Date incorecte!');
End;

Begin
 WriteLn('Dati iptiunea dvs. ');
 WriteLn('[1:patrat, 2:cerc , 3:triunghi] ');
 ReadLn(opt);
 Case opt Of
  1:aria_patrat;
  2:aria_cerc;
  3:aria_triunghi;
  Else
   WriteLn('Date incorecte!');
 End;
 ReadLn;
End.
```

Funcţii

La fel ca şi procedurile, funcţiile efectuează anumite operaţii, dar în plus a foncţie „întoarce” o anumită valoare. Tipul valorii returnate se precizează la sfârşitul antetului finţiei, fiind precedat de caracterul „două puncte”. Valoarea pe care o returnează o funcţie poate fi folosită apoi în modul apelant şi în celelalte module componente.

Sintaxa:

```
Function <id_fct> (<L1>:<tip1>; <L2>:<tip2>; ...; <Ln>:<tipn>): <tipr>;
  ...
  {declaraţii de variabile locale}
Begin
  ...
  {corpul funcţiei}
End;
```

<id_fact> - identificatorul

<L1>,<L2>,...,<Ln> - sunt liste de paramateri

<tip1>,<tip2>,...,<tipn> - tipurile parametrilor

<tip_r> - tipul valorii returnate

Exemplu:

```
Program Media;
Var
  x,y:Real;

Function calcul(a,b:Real):real;
Begin
  Calcul:=(a+b)/2;
End;

Begin
  Write('x,y=');
  ReadLn(x,y)
  ReadLn('Media=',calcul(x,y));
  ReadLn;
End.
```

Media aritmetică al perechii de numere

Realizaţi un program care citeşte de la tastatură două perechi de numere (x1,x2) şi (y1,y2), calculează media aritmetică a numerelor fiecărei perechi, apoi determină cea mai mare dintre mediile obţinute.

```
Progam media_2;
Var
 m1,m2,x1,x2,y1,y2:Real;
Function calcul(x,y:Real):Real;
Begin
 Calcul:=(x+y)/2;
End;

Begin
 Write('x1,x2=');
 ReadLn(x1,x2);
 M1:=calcul(x1,x2);
 Write('y1,y2=');
 ReadLn(y1,y2);
 M1:=calcul(y1,y2);
 If M1>M2 Then
  WriteLn(M1)
 Else
  WriteLn(M2);
 ReadLn;
End.
```

Suma cifrelor unui număr

Scrieţi o funcţie care returnează suma cifrelor unui număr natural x dat ca paramaetru.

```
Program suma;
Var
 x:LongInt;
Function suma(x:LongInt):Integer;
Var
 d,s:Integer;
Begin
 D:=x;
```

```
 S:=0;
 Repeat
  S:=S+d mod 10;
  D:=d div 10;
 Until d=0;
 Suma:=S;
End;

Begin
 Write('Numărul:');
 ReadLn(x);
 WriteLn('Suma cifrelor lui ',x, ' este : ',suma(x));
 ReadLn;
End.
```

Număr prim

Scrieţi un program care testează dacă un număr natural x dat ca parametru este prim sau nu, returnând true sau false.

```
Program Prim;
Var
 x:Integer;
Function test(x:Integer):Boolean;
Var
 i:Integer;
Begin
 Test:=True;
 For i:=2 To x div 2 Do
 If x mod i = 0 Then
  Test:=False;
End;

Begin
 Write('x=');
 ReadLn(x);
 If test(x) Then
  WriteLn('Numarul este prim!')
 Else
  WriteLn('Numarul nu este prim!');
End.
```