INFORMATICĂ

LIMBAJUL PASCAL

MATERIAL ELABORAT CORESPUNZÂND CERINȚELOR DE BACALAUREAT 2016

FOLOSIREA FIȚUICILOR ESTE O FRAUDĂ.
NU RECOMANDĂM UTILIZAREA LOR ÎN TIMPUL EXAMENELOR!

Cuprins

Gândirea algoritmică 1–14
Problema căutării.....1–2
Problema intrării într-un cabinet medical 2–3
Tipuri de date 4
Identificatori.....4–5
Constante.....5
Variabilă.....5
Tipul Integer.....5
Tipul Real.....6
Tipul Char 6
Tipul Boolean.....6
Tipul String 7–8
Vectorul.....9–10
Matricea (tablou bidimensional).....10–11
Înregistrarea (record).....11
Fişierul text.....12
Structuri de date de tip liste 13–14
Cum putem implementa o listă?.....13
Crearea unei liste 13–14
Afişarea elementelor listei 14
Inserarea unui element în listă 14
Ştergerea unui element din listă.....14

Proceduri.....15–21
Media aritmetică.....15–16
Ariile unor figuri geometrice.....16–18
Şir de numere în fişier 18–19
Laturi în triunghi 19–20
Interschimbarea a două linii într-o matrice.....20–21

Funcţii.....22–35
Media aritmetică al perechii de numere.....23
Suma cifrelor unui număr.....24
Număr prim 24–25
Elementele prime ale unui şir 25–26
Ordonarea a trei numere 26–27
Unghiuri în grade şi radiani.....27–28
Perechi de numere 29–30
Frecvenţele caracterelor într-un şir.....30–31

Platouri de lungime maximă într-un şir 31–33
Cel mai mare divizor comun 33–35

Recursivitate 36–48
Conceptul de recursivitate 36
Inversarea unui cuvânt.....36
Şirul Fibonacci 37
Cel mai mare divizor comun 38
Funcţia Ackermann 38–39
Numărare.....39–40
Anagrame 40–41
Generare 41–42
Conversie.....42–43
Codul Gray 43–44
Şirul mediilor aritmetico-geometrice al lui Gauss 44–45
Evaluarea unei expresii aritmetice.....46–48

Metoda Divide Et Impera 48–59
Cel mai mare divizor comun 49–51
Problema turnurilor din Hanoi.....51
Problema tăieturilor.....52–53
Algoritmi de căutare. Căutarea binară 54–55
Merge Sort – sortare prin inreclasare.....55–57
QuickSort - Sortare Rapidă 58–59

Metoda BackTracking.....60–83
Descrierea generală a metodei.....60
Probleme reginelor 60–62
Generarea elementelor combinatorice 62–66
  Generarea permutărilor 62–63
  Generarea aranjamentelor 64–65
  Generarea combinărilor 65–66
Partiţiile unei mulţimi.....66–68
Partiţiile unui număr natural.....68–69
Plata unei sume cu monede de valori date.....69–71
Paranteze 71–73
Comis-voiajor.....73–76
Backtracking în plan.....76
Labirint.....76–79
Fotografie 79–81
Cel mai lung prefix.....81–83

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.
```

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);
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.
```

Sau

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;
```

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.
```