

INFORMATICĂ

LIMBAJUL PASCAL



MATERIAL ELABORAT CORESPUNZÂND
CERINȚELOR DE BACALAUREAT 2016

© 2016 PRESSTERN SOLUTIONS

Cuprins

Gândirea algoritmică	1
Problema căutării	1
Problema intrării într-un cabinet medical.....	3
Tipuri de date.....	4
Identificatori.....	4
Constante.....	5
Variabilă.....	6
Tipul Integer	6
Tipul Real.....	6
Tipul Char	7
Tipul Boolean.....	7
Tipul String.....	8
Vectorul.....	10
Matricea (tablou bidimensional)	11
Înregistrarea (record)	12
Fișierul text	13
Structuri de date de tip liste.....	14
Proceduri	17
Media aritmetică.....	17
Ariile unor figuri geometrice	18
Șir de numere în fișier	20
Laturi în triunghi	21
Interschimbarea a două linii într-o matrice	22
Funcții.....	24
Media aritmetică al perechii de numere	25
Suma cifrelor unui număr.....	26
Număr prim	26
Elementele prime ale unui șir.....	27
Ordonarea a trei numere	28
Unghiuri în grade și radiani.....	29
Perechi de numere	31

Frecvențele caracterelor într-un șir	32
Platouri de lungime maximă într-un șir.....	33
Cel mai mare divizor comun	35
Recursivitate	38
Conceptul de recursivitate.....	38
Inversarea unui cuvânt.....	38
Șirul Fibonacci	39
Cel mai mare divizor comun	40
Funcția Ackermann.....	41
Numărare	41
Anagrame	42
Generare.....	44
Conversie.....	45
Codul Gray.....	45
Șirul mediilor aritmetico-geometrice al lui Gauss	47
Evaluarea unei expresii aritmetice.....	48
Metoda Divide Et Impera.....	51
Cel mai mare divizor comun	51
Problema turnurilor din Hanoi	53
Problema tăieturilor.....	54
Algoritmi de căutare. Căutarea binară.....	56
Merge Sort – sortare prin înreclasare	57
QuickSort - Sortare Rapidă	60
Metoda BackTracking	62
Descrierea generală a metodei	62
Probleme reginelor.....	62
Generarea elementelor combinatorice.....	64
Partițiile unei mulțimi.....	68
Partițiile unui număr natural.....	70
Plata unei sume cu monede de valori date.....	71
Paranteze.....	73
Comis-voiajor	74
Backtracking în plan.....	77
Labirint.....	78
Fotografie	80
Cel mai lung prefix	82

Gândirea algoritmică

Problema căutării

Există și cazul care, pentru o aceeași problemă putem prezenta două soluții, în care una este mai rapidă ca alta. De pildă, fie un șir de numere naturale oarecare, de exemplu 4,2,10,1,8,15,7. Vrem să testăm dacă un număr dat (să zicem numărul 8) se află în această secvență sau nu.

O astfel de căutare, prin parcurgerea de la stânga la dreapta a întregii secvențe de numere, până se găsește numărul dorit sau se epuiează toate elementele din secvență, se numește căutarea secvențială. Dacă avem un șir S de n elemente (notat $S[1..n]$), atunci căutarea secvențială a lui x în S se descrie prin:

Căutare_secvențială($x, S[1..n]$) înseamnă

Început

Fie `elemental_curent = primul_element;`

Atât timp cât (`elemental_curent <> x`) și (`pozitia elementului current <= pozitia ultimului element (n)`) execută

Început

Dacă `elemental_curent = x` atunci mesaj('găsit')

Altfel treci la următorul element

Sfârșit

Sfârșit.

În continuare să presupunem că numerele erau deja ordonate crescător 1, 2, 4, 7, 8, 10, 15. În acest caz particular, căutarea secvențială a unui număr cum este 8 nu este prea eficientă, deoarece 8 se află în a doua jumătate a secvenței, 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 și trebuie să oprim căutarea.

Căutarea în jumătatea aleasă se face tot la fel, deci se va înjumătăți și această zonă...

Procedeu anterior se numește căutare binară, deoarece, de fiecare dată, o secvență de numere este divizată în două jumătăți. Putem descrie căutarea binară a unui număr x într-o secvență $S[1..n]$ astfel.

Căutarea_binară($x, S[1..n]$) înseamnă

Început

Stabilește elemental_curent ca fiind elemental din mijlocul secvenței;

Dacă $n=1$ atunci

Dacă $x = \text{elemental_curent}$ atunci mesaj('găsit')

Altfel mesaj('negăsit')

Altfel

Început

Împarte secvența în cele două jumătăți ($S[1..mijloc]$ și $S[mijloc+1..n]$);

Dacă $\text{elemental_curent} = x$ atunci mesaj('găsit')

Altfel

Dacă $\text{elemental_curent} < x$ atunci

Căutare_binară($x, S[mijloc+1..n]$)

Altfel căutarea_binară($x, S[1..mijloc]$)

Sfârșit

Sfârșit.

Problema intrării într-un cabinet medical

Dacă un om vrea să intre pentru consult la un medic, atunci, mai întâi va bate la ușă: dacă medicul răspunde prin „poftim!”, atunci va intra, altfel va aștepta până când pacientul dinăuntru va ieși; abia după ce cabinetul va fi liber, va intra.

Intrare_la_medic înseamnă

Început

Bate_la_ușă;

Dacă_răspunsul = ,poftim!' atunci

Întră_în_cabinet;

Altfel

Început

Atât_timp_cât_cabinetul_este_ocupat_de_alt_pacient

execută

Citeste_un_ziar;

Sfârșit

Sfârșit.

O instrucțiune compusă este formată dintr-o secvență de instrucțiuni, încadrate de cuvinte început și sfârșit, care pot conține, la rândul lor, blocuri de alternativă și repetiție.

Programarea este tehnica realizării de algoritmi descriși prin proceduri și programe. Ea devine o artă, atunci când se folosesc cele trei elemente de structurate și se numește 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_ușă;

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

Sfârșit

Sfârșit.

Proceduri

Media aritmetică

Realizați un program care calculează și afișează media aritmetică a două numere reale x și 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 conține toate acțiunile algoritmului: citirea numerelor, calcul și afișarea 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

Scrieți un program care, în funcție de dorința utilizatorului, calculează și afișează: aria unui pătrat de latură L , sau aria unui cerc de rază r , sau aria unui triunghi cu baza b și înălțimea h . Pentru calcul și afișarea 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 inaltd 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);

```

Funcții

La fel ca și procedurile, funcțiile efectuează anumite operații, dar în plus a funcție „întoarce” o anumită valoare. Tipul valorii returnate se precizează la sfârșitul antetului funcției, fiind precedat de caracterul „două puncte”. Valoarea pe care o returnează o funcție poate fi folosită apoi în modul apelant și în celelalte module componente.

Sintaxa:

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

<id_fct> - identificatorul

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

<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

Realizați un program care citește de la tastatură două perechi de numere (x_1, x_2) și (y_1, y_2), calculează media aritmetică a numerelor fiecărei perechi, apoi determină cea mai mare dintre mediile obținute.

```
Program 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.
```