

Cuprins

Gândirea algoritmică	1
Problema căutării.....	1-2
Problema intrării într-un cabinet medical	2-3
Tipuri de date	3
Identificatori	3-4
Constante.....	4
Variabilă.....	4
Tipul Integer.....	4
Tipul Real.....	4-5
Tipul Char	5
Tipul Boolean.....	5
Tipul String	5-6
Vectorul.....	6-7
Matricea (tablou bidimensional).....	7-8
Înregistrarea (record).....	8-9
Fișierul text.....	9
Structuri de date de tip liste	9-11
Cum putem implementa o listă?.....	10
Crearea unei liste	10
Afișarea elementelor listei	10
Inserarea unui element în listă	10-11
Ștergerea unui element din listă	11
Proceduri	11-16
Media aritmetică.....	11-12
Ariile unor figuri geometrice.....	12-13
Șir de numere în fișier	13-14
Laturi în triunghi	14-15
Interschimbarea a două linii într-o matrice.....	15-16
Funcții	16-27
Media aritmetică al perechii de numere.....	17-18
Suma cifrelor unui număr.....	18
Număr prim	18-19
Elementele prime ale unui șir.....	19-20
Ordonarea a trei numere	20
Unghiuri în grade și radiani	20-21
Perechi de numere	21-21
Frecvențele caracterelor într-un șir.....	22-24
Platouri de lungime maximă într-un șir.....	24-25
Cel mai mare divizor comun	25-27
Recursivitate	27-37
Conceptul de recursivitate	27
Inversarea unui cuvânt.....	27-28
Șirul Fibonacci	28
Cel mai mare divizor comun	28-29
Funcția Ackermann	29

Numărare.....	29–30
Anagrame.....	30–31
Generare.....	31–32
Conversie.....	32
Codul Gray.....	33–34
Șirul mediilor aritmetico-geometrice al lui Gauss.....	34–35
Evaluarea unei expresii aritmetice.....	35–37
Metoda Divide Et Impera.....	37–45
Cel mai mare divizor comun.....	37–39
Problema turnurilor din Hanoi.....	39
Problema tăieturilor.....	39–41
Algoritmi de căutare. Căutarea binară.....	41–42
Merge Sort – sortare prin inreclasare.....	42–44
QuickSort - Sortare Rapidă.....	44–45
Metoda BackTracking.....	45–63
Descrierea generală a metodei.....	45–46
Probleme reginelor.....	46–47
Generarea elementelor combinatorice.....	47–50
Generarea permutărilor.....	47–48
Generarea aranjamentelor.....	48–49
Generarea combinărilor.....	49–50
Partițiile unei mulțimi.....	50–52
Partițiile unui număr natural.....	52–53
Plata unei sume cu monede de valori date.....	53–54
Paranteze.....	54–55
Comis-voiajor.....	55–58
Backtracking în plan.....	58
Labirint.....	58–60
Fotografie.....	60–62
Cel mai lung prefix.....	62–63

Comandă versiunea completă, tipărită de
www.fituici-bacalaureat.ro

www.fituici-bacalaureat.ro

```

For j:=1 To n Do
Begin
  Temp:=a[i1, j];
  A[i1,j]:=a[i2,j];
  A[i2,j]:=temp;
End;
End;

Procedure afisare(a:matric; n,m:Integer);
Begin
  For i:=1 To n Do
  Begin
    For j:=1 To m Do
      Write(a[i,j]);
      WriteLn;
    End;
  End;
End;

Begin
  Citire;
  Afisare(a,n,m);
  Repeat
    WriteLn('Dati liniile care se vor interschimba:');
    ReadLn;
  Until (L1 in [1..m] and (L2 in [1..m]));
  Interschimbare(L1,L2);
  Afisare(a,n,m);
  ReadLn;
End.

```

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_fact> - 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 (x1,x2) și (y1,y2), 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.

```

Suma cifrelor unui număr

Scrieți o funcție 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

Scrieți 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.

```

Elementele prime ale unui șir

Scrieți un program care să se tipărească elementele prime ale unui șir de n numere întregi citit de la tastatură.

```

Program Prim_2;
Type
  vect=array [1..100] of Integer;
Var
  n,j:Integer;
  V:vector;

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

Begin
  Write('n=');
  ReadLn(n);
  For j:=1 To n Do
    Begin
      Write('v[' ,k, ']= ');
      ReadLn(v[k]);
      If test(v[k]) Then
        WriteLn(v[k])
    End;
  End;

```

Comandă versiunea completă, tipărită de
www.fituici-bacalaureat.ro

www.fituici-bacalaureat.ro

```
ReadLn;  
End.
```

Sirul Fibonacci

```
Function Fibo(n:Byte):LongInt;
```

```
Var
```

```
  i:Byte;
```

```
  f0,f1,f2:LongInt;
```

```
Begin
```

```
  If n <= 1 Then
```

```
    f2:=n
```

```
  Else Begin
```

```
    f0 := 1;
```

```
    f1:=1;
```

```
    For i:=2 To n Do
```

```
      Begin
```

```
        f2:=f0+f1;
```

```
        f0:=f1;
```

```
        f1:=f2;
```

```
      End;
```

```
    End;
```

```
  fib:=f2;
```

```
End;
```

```
Function Fibo_Recurs(n:Byte):LongInt;
```

```
Var
```

```
  k:Byte;
```

```
  S:LongInt;
```

```
Begin
```

```
  If n <= 2 Then
```

```
    p:=1
```

```
  Else Begin
```

```
    S:=0;
```

```
    For k:=1 To n-1 Do S:=S+P(k)*P(n-k);
```

```
    P:=S;
```

```
  End;
```

```
End.
```

Cel mai mare divizor comun

```
Program CMMDC_Euclid;
```

```
Var
```

```
  a,b:Word;
```

```
Function CMMDC(x,y:Word):Word;
```

```
Begin
```

```
  If y = 0 Then
```

```

    CMMDC:=x
Else
    CMMDC:=CMMDC(y, x mod y)
End;

Begin
    Write('a= ');
    ReadLn(a);
    Write('b= ');
    ReadLn(b);
    WriteLn('C.M.M.D.C. (' ,a , ' , ' , b , ') = ', CMMDC(a,b));
    ReadLn;
End.

```

Funcția Ackermann

```

Program Ackermann;
Var
    m,n:Byte;
Function ac(x,y:byte):LongInt;
Begin
    If x=0 Then
        ac:=y+1
    Else
        If y=0 Then
            ac:=ac(x-1,1)
        Else
            ac:=ac(x-1, ac(x,y-1))
    End;
End;

Begin
    Write('m=');
    ReadLn(m);
    Write('n=');
    ReadLn(n);
    Write('ac(' ,m , ' , ' , n , ') = ',ac(m,n));
    ReadLn;
end.

```

Numărare

Fie a_1, a_2, \dots, a_n un șir de n ($0 < n < 20$) numere întregi și x un număr întreg. Scrieți o funcție recursivă care să determine numărul de apariții ale lui x în șir.

```

Program Aparitii;
Const

```

Comandă versiunea completă, tipărită de
www.fituici-bacalaureat.ro

www.fituici-bacalaureat.ro

```

    Taie(xs, ys, x[i]-xs, h);
    Taie(x[i], ys, xs+1-x[i], h);
End;
End;

Begin
    Citire;
    Taie(0, 0, Lp, Hp);
    Write(' Placa de arie maxima are coltul stanga jos ( ');
    WriteLn(xmax, ', ', ymax, ') ');
    WriteLn(' Lungimea = ',lmax, '; inaltimea = ',hmax);
    ReadLn;
End.

```

Algoritmi de căutare. Căutarea binară

```

Program Cautare_Binara;
Const
    DimMax=100;
Type
    Indice = 0..DimMax;
    Vector = array[Indice] of Integer;
Var
    a:Vector;
    N,poz:Indice;
    X:Integer;

Procedure Citire;
Var
    i:Indice;
Begin
    Write('n= ');
    ReadLn(n);
    WriteLn('Introduceti elementele ');
    For i:=1 To n Do Read(a[i]);
    ReadLn;
    Write(' x=');
    ReadLn(x);
End;

Function Cautare(prim, ultim:Indice):Indice;
Var
    mijloc:Indice;
Begin
    If prim>ultim Then
        Cautare:=0;
    Else

```

```

Begin
  Mijloc:=(prim+ultim) div 2;
  If x= a[mijloc] Then
    cautare:=mijloc
  Else
    If x<a[mijloc] Then
      Cautare:=cautare(prim, mijloc-1)
    Else
      Cautare:=Cautare(mijloc+1, ultim);
    End;
  End;
End;

```

```

Begin
  Citire;
  Poz:=cautare(1,n);
  If poz =0 Then
    WriteLn(x, ' nu se gaseste in sir! ')
  Else
    WriteLn(x, ' apare pe pozitia',poz);
  End.

```

Merge Sort – sortare prin inreclasare

```

Program Sortare_Merge_sort;
Const
  Nmax=20;
Type
  Indice = 1..Nmax;
Var
  a:Array[Indice] of Integer;
  N:Indice;

Procedure Citire;
Var
  fin:text;
  I:Indice;
Begin
  Assign(fin, 'sort.in');
  Reset(fin);
  ReadLn(fin,n);
  For i:=1 To n Do Read(fin,a[i]);
  ReadLn(fin);
  Close(fin);
End;

Procedure Afisare;
Var

```

```

i:Integer;
Begin
  WriteLn(' Vectorul sortat este : ');
  For i:=1 To n Do Write(a[i], ' ');
  WriteLn;
  ReadLn;
End;

Procedure Interclasare(p, m, q:Indice);
Var
  i, j, k:Indice;
  B:array[Indice] of Tipe Element;
Begin
  i:=p;
  j:=m+1;
  k:=1;
  While (i<=m) and (j<=q) Do
  Begin
    If a[i]<a[j] Then
    Begin
      b[k]:=a[i];
      Inc(i);
    End
    Else
    Begin
      b[k]:=a[j];
      Inc(j);
    End;
    Inc(k);
  End;
  While i<=m Do
  Begin
    b[k]:=a[i];
    Inc(i);
    Inc(k);
  End;
  While j<=q Do
  Begin
    b[k]:=a[j];
    Inc(j);
    Inc(k);
  End;
  For i:=p To q Do a[i]:=b[i-p+1];
End;

Procedure Msort(p,q:Indice);

```

```

Var
  m:Indice;
Begin
  If q>p Then
  Begin
    M:=(p+q) div 2;
    MSort(p,m);
    MSort(m+1,q);
    InterClasare(p, m, q);
  End
End;

Begin
  Citire;
  MSort(1, n);
  Afisare;
End.

```

QuickSort - Sortare Rapidă

```

Program QuickSort;
Const
  Nmax = 20;
Type
  Indice=1..Nmax;
Var
  a:Array[Indice] od Integer;
  N:Indice;

Procedure Citire;
Var
  fin:Text;
  I:Indice;
Begin
  Assign(fin, 'sort.in');
  Reset(fin);
  ReadLn(fin, n);
  For i:=1 To n Do Read(fin,a[i]);
  ReadLn(fin);
  Close(fin);
End;

Procedure Afisare;
Var
  i:Integer;
Begin
  WriteLn(' Vectorul sortat este : ');

```

Comandă versiunea completă, tipărită de
www.fituici-bacalaureat.ro

www.fituici-bacalaureat.ro