VECTORI (II)

DESFĂŞURAREA LUCRĂRII

Se vor edita şi apoi executa programele descrise în continuare.

Programul nr. 1

Se citesc 20 de numere întregi de la tastatură şi se memorează într-un vector A. Să se construiască vectorul B ce conţine toate numerele prime din vectorul A.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#define N 20
void main(void)
{
 int A[N];
 int B[N];
 int i,j,k;
 int estePrim;
 clrscr();
 for(i=0;i<N;i++){
   printf("A[%d]=",i);
   scanf("%d",&A[i]);}
 j=0;//index in vectorul B
 for(i=0;i<N;i++){
   //este A[i] numar prim?
   estePrim=1;//initializare semafor estePrim
   for(k=2; k<A[i];k++)
     if(A[i]%k==0){
          estePrim=0;//nu este numar prim
          break;}
   if(estePrim){
     B[j]=A[i];
     j++;}
 }
 //afisare vector B:
 if(j==0){
   printf("Nu sunt numere prime in vectorul A! ");
   getch();
```

#include<iostream>
using namespace std;

int

cout<<"A["<<i<<"]= ";
cin>>A[i]

cout<<"Nu sunt ...A!";

```
   exit(1);}
 printf("Vectorul B:\n");
 for(i=0;i<j;i++)
   printf("%d\n",B[i]);
 getch();
}
```

cout<<"Vectorul B:"<<endl;

cout<<B[i]<<endl;

Programul nr. 2

Să se realizeze un program care realizează produsul scalar a doi vectori

Sursa programului:

```
#include<conio.h>
#include<stdio.h>
  void main(void){

  int A[100],B[100]; //vectori de maxim 100 numere intregi
  int i,dim;
  long int produsScalar=0;

  clrscr();
  printf("Introduceti dimensiunea vectorilor:"); scanf("%d",&dim);
  for(i=0;i<dim;i++){
    printf("A[%d]=",i);
    scanf("%d",&A[i]);
    } //s-a citit vectorul A de la tastatura
  printf("\n");
  for(i=0;i<dim;i++){
    printf("B[%d]=",i);
    scanf("%d",&B[i]);
    } //s-a citit vectorul B de la tastatura
  //calculez produsul scalar:
  for(i=0;i<dim;i++)
   produsScalar=produsScalar+A[i]*B[i];

  //afisez produsul scalar:
printf("Prod.  scalar pentru cei doi vectori este %ld",produsScalar);
  getch();
  }//main
```

Programul nr. 3

Să se citească de la tastatură N numere întregi distincte, ce se vor memora într-un vector A. Dacă se tastează un număr care a fost introdus deja în vectorul A, se repetă tastarea până se va introduce un număr distinct.

Exemplu:

Introduceţi numărul 1: 7
Introduceţi numărul 2: 1
Introduceţi numărul 3: 7
Introduceţi numărul 3: 5
Introduceţi numărul 4: ...

Se observă cum s-a repetat introducerea numărului 3.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#define N 10 //numarul de numere
void main(void)
{
 int A[N];
 int i,j;
 int esteDistinct; //variabila semafor (flag)
 clrscr();
 //citeste primul numar:
 printf("Introduceti numarul 1:  ");
 scanf("%d",&A[0]);
 for(i=1;i<N;i++)
  for(;;){
    printf("Introduceti numarul %d:  ",i+1);
    scanf("%d",&A[i]);
    //este distinct de cele introduse pana acum in A ?
    esteDistinct=1;
    for(j=0;j<=i-1;j++)
      if(A[j]==A[i]){//nu este
              esteDistinct=0;
              break;}
    if(esteDistinct==1)break;
  }//for;;
//Afisarea numerelor tastate:
printf("\n");
for(i=0;i<N;i++)
```

```
   printf("\nA[%d]=%d",i,A[i]);
 getch();
}
```

Programul nr. 4

Se dă un vector A de numere întregi. Să se sorteze în ordine crescătoare, folosind metoda sortării prin interschimbare.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#define N 10
void main(void)
{
/* În această metodă se parcurge vectorul si se compară elementul
curent  cu toate elementele ce se afla in vector dupa el.  Dacă
elementele comparate nu sunt în ordinea corespunzătoare sortării
dorite, se inversează între ele (se comută conţinutul lor). */
 int i,j;
 int A[N];
 int temp;
 clrscr();
 for(i=0;i<N;i++){
  printf("A[%d]=",i);
  scanf("%d",&A[i]);}
 for(i=0;i<N-1;i++)
    for(j=i+1;j<N;j++)
        if(a[i]>a[j]){
          temp=a[i];
          a[i]=a[j];
          a[j]=temp;}
//Afisarea vectorului sortat:
 for(i=0;i<N;i++)
  printf("\nA[%d]=%d",i,A[i]);
 getch();
}
```

Programul nr. 5

Se dă un vector A de numere întregi. Să se sorteze în ordine descrescătoare, folosind metoda bubble sort.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#define N 10
void main(void)
{
/* În această metodă se compară fiecare element cu următorul
element (vecinul său din dreapta). Dacă cele două componente nu
sunt în ordinea corespunzătoare sortării dorite, se inversează între ele
(se comută conţinutul lor). Se repetă acest procedeu până când s-a
parcurs tot vectorul. Deci, pentru vectorul A, se compară A[0] cu
A[1], apoi A[1] cu A[2], A[2] cu A[3] ş.a.m.d. Dacă după
parcurgerea vectorului au fost făcute comutări, înseamnă că încă nu
este sortat şi se reia  parcurgerea lui (cu compararea elementelor
învecinate) începând cu prima componentă.
*/
 int i,j;
 int A[N];
 int temp;
 int existaInversiuni;//variabila semafor
 clrscr();
 for(i=0;i<N;i++){
  printf("A[%d]=",i);
  scanf("%d",&A[i]);}
 //Se ordoneaza descrescator componentele vectorului A:
 //Metoda bubble sort:
 do{
  existaInversiuni=0;
  for(i=0;i<N-1;i++)
   if(A[i]<A[i+1]){
     existaInversiuni=1;
     //se inverseaza A[i] cu A[i+1]:
     temp=A[i];
     A[i]=A[i+1];
     A[i+1]=temp;}//if
 }while(existaInversiuni==1);
 //Afisarea vectorului sortat:
 printf("\n Vectorul A sortat descrescator: ");
 for(i=0;i<N;i++)
  printf("\nA[%d]=%d",i,A[i]);
```

```
 getch();
}
```

Programul nr. 6

Să se implementeze un program care implementează algoritmul de sortare prin inserţie a elementelor unui vector.

Sursa programului:

```
//Sunt doua grupe: cea din stanga este sortata, cea din
//dreapta nu este.
//Initial, grupa din stanga contine primul element din vector:A[0]
#include <stdio.h>
#include <conio.h>
#define N 9
void main()
{
 int A[N]={11111,-1,77,8,111,4,3,55,100};
 int  iNesortatSt;//indexul din stanga grupei nesortate
 int nr;
 int i,j;
 clrscr();
 for(iNesortatSt=1;iNesortatSt<N;iNesortatSt++){
  nr=A[iNesortatSt];
 //inserez pe nr in grupa sortata din stanga, ce cuprinde // elem. de pe
pozitiile 0,1,...,iNesortatSt-1
  if(nr>A[iNesortatSt-1])continue;//nr e pe pozitia care //trebuie
  for(i=0;i<iNesortatSt;i++)
     if(nr<A[i]){
          //ii fac loc: le mutam pe oate de la i spre dreapta:
          for(j=iNesortatSt-1;j>=i;j--)
            A[j+1]=A[j];
          A[i]=nr;
          break;//din for i
     }
 }
 //afisare:
 for(i=0;i<N;i++)
  printf("\n%d",A[i]);
 getch();
}
```

Programul nr. 7

Se citeşte un număr natural x, de la tastatură. Să se copieze primele N numere prime (N – cunoscut), care sunt strict mai mari ca x, într-un vector A. Să se afişeze acest vector.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define N 10
void main()
{
 int x;
 int nrCrt;
 int A[N];
 int i;
 clrscr();
 printf("x=");
 scanf("%d",&x);
 nrCrt=x+1;
 for(i=0;i<N;i++)
   //se introduce in A[i], urmatorul numar prim >= nrCrt:
   //cautam acest numar prim:
   for(;;){
     //este prim nrCrt?
     int este=1; //presupunem ca este
     int j;
     for(j=2;j<=sqrt(nr);j++)
        if(nr%j==0){
           este=0;
           break;}
     if(este==1){
           A[i]=nrCrt;
           nrCrt++;
           break; //iese din for(;;)}
     else nrCrt++;
   }//end for;;
//afisarea vectorului A:
for(i=0;i<N;i++)
   printf("\n%d",A[i]);
getch(); }
```

Programul nr. 8

Fie *a* şi *b* doi vectori de numere întregi, citite de la tastatură. Să se construiască vectorul *c*, obţinut prin concatenarea ("lipirea") celor doi vectori.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#define na 5  //numarul de elemente ale vectorului a
#define nb 7
void main(void)
{
 int i;
 int a[na], b[nb];
 int c[na+nb];//vectorul obtinut prin concatenarea vectorilor a si b
 clrscr();
 for(i=0;i<na;i++){
   printf("a[%d]=",i);
   scanf("%d",&a[i]);}
 for(i=0;i<nb;i++){
   printf("b[%d]=",i);
   scanf("%d",&b[i]);}
 //se copiaza vectorul a in c:
 for(i=0;i<na;i++)c[i]=a[i];
 //In continuarea lui c, se copiaza vectorul b:
 for(i=0;i<nb;i++)c[na+i]=b[i];
 //Afisarea vectorului c:
 for(i=0;i<na+nb;i++)
   printf("\nc[%d]=%d",i,c[i]);
 getch();
}
```

Programul nr. 9

Folosind alocarea dinamica a memoriei, să se memoreze într-un vector, un număr de valori introduse de la tastatură, în vederea unor prelucrări ulterioare.

Pentru alocarea dinamică a memoriei, în limbajul C se folosesc funcţiile **malloc()** şi **free()**. (În limbajul C++, pe lângă acestea, se foloseşte şi operatorul **new**).

Funcţia **malloc()** alocă memorie din zona de memorie dinamică (**heap**) şi returnează adresa de început a zonei alocate. Are ca parametru de intrare, numărul de octeţi solicitaţi.

Funcţia **free()** eliberează memoria alocată anterior din **heap**, pentru o eventuală reutilizare. Ea are ca parametru de intrare, adresa zonei din **heap** anterior alocată printr-un apel al funcţiei **malloc()**.

Prototipurile acestor funcţii sunt:

void *malloc(size_t număr_de_octeţi);

void free(void * p);

Ambele funcţii sunt declarate în fişierul header **stdlib.h** . În prototipul lui **malloc()**, număr_de_octeţi reprezintă numărul de octeţi ceruţi pentru alocare dinamică. Dacă nu mai este suficientă memorie liberă în **heap**, funcţia **malloc()** returnează pointerul **NULL** (valoarea 0).

Tipul **size_t** este definit în **stdlib.h** şi reprezintă un tip generic de întreg fără semn, ce este capabil să ţină cea mai mare cantitate de memorie ce poate fi alocată în urma unui apel al funcţiei **malloc()** .

Alocarea dinamică a memoriei se foloseşte tipic în aplicaţiile cu structuri arborescente (liste înlănţuite, arbori).

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#include <alloc.h> //pentru functiile malloc() si free()
#include <stdlib.h>
void main(void)
{
 int nrElemente;
//adresa de inceput a vectorului in care se memoreaza numerele:
 int * tab;
 int i;
 clrscr();
 printf("Tastati numarul de elemente:");
 scanf("%d",&nrElemente);
 tab=(int * )malloc(nrElemente * sizeof(int) );
 if(tab==NULL){
  printf("Nu exista memorie disponibila!");
  getch();
  exit(1);
 }
 for(i=0;i<nrElemente;i++){
  printf("tab[%d]=",i);
```

```
 scanf("%d",&tab[i]);}
//afisare:
printf("\nComponentele introduse in memoria dinamica sunt:");
for(i=0;i<nrElemente;i++)printf("\n%d",tab[i]);
free(tab);
getch();
}//main
```

4. PROBLEME PROPUSE

1. Se dă un vector de numere întregi. Care este cea mai mare distanţă între două elemente alăturate. Ex. {2,3,17,25}. R: 14.

2. Să se realizeze un program care calculează c.m.m.d.c. a elementelor unui vector.

3. Să se realizeze un program care calculează suma a două numere mari reprezentate ca vectori.

4. Să se scrie un program prin care se calculează şi afişează N! unde N este un număr mare (Exemplu: 50!)