STRUCTURI

1. SCOPUL LUCRĂRII

În această lucrare se vor studia structuri şi vectori de structuri.

2. BREVIAR TEORETIC

2.1. Structuri

Structurile sunt folosite pentru a grupa sub acelaşi nume, mai multe date de acelaşi tip sau de tipuri diferite. Sunt denumite şi înregistrări. În limbajul C, pentru a declara o structură, se foloseşte cuvântul cheie **struct**. În mod uzual, numele unei structuri se declară cu ajutorul cuvântului cheie **typedef**.

Sintaxa:

```
typedef struct{
        tip1 membrul1;
        tip2 membrul2;
        ...
                }numeStructra;
```

Exemplu:

```
typedef struct{
        double re;
        double im;
                }complex;
```

După ce a fost definit cuvântul cheie **typedef**, numele asociat structurii poate fi folosit pentru a declara variabile.

Exemplu:

```
complex c1,c2;
angajat a1;
```

2.2. Accesul la componentele unei structuri

Pentru accesarea membrilor unei structuri, pe baza numelui ei, se foloseşte operatorul punct.

Exemple:

```
c1.re = 0.2;
c1.im = 0.7;
modulC1 = sqrt(c1.re* c1.re + c1.im* c1.im);
```

2.3. Vectori de structuri

În mod frecvent se folosesc tablouri ce au drept componente structuri. Astfel, un vector ce conţine 100 de numere complexe, se declară în felul următor:

```
typedef struct{
                double re;
                double im;
                }complex;
complex tab[100];
```

Accesăm un câmp dintr-o structură componentă a unui vector, tot prin intermediul operatorului punct, aflat la dreapta componentei.
Astfel prin instrucţiunea:

```
tab[0].re = 0.5;
```

se atribuie câmpului *re* al primei componente din vectorul *tab* valoarea 5.

3. DESFĂŞURAREA LUCRĂRII

Se vor edita şi apoi executa programele descrise în continuare.

Programul nr. 1

Se defineşte structura punct, ce are două câmpuri x şi y (coordonatele reprezentării în plan a unui punct). Să se scrie un program în care se citesc coordonatele a două puncte, de la tastatură şi se afişează distanţa între aceste puncte.

Sursa programului:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
typedef struct{
                int x;
                int y;
                }punct;

void main(void)
{
 punct P1,P2;
 double d12; //distanta de la punctul 1 la punctul 2
 clrscr();
 printf("Primul punct: x="); scanf("%d",&P1.x);
 printf("\nPrimul punct: y="); scanf("%d",&P1.y);
```

```
printf("\nAl doilea punct: x="); scanf("%d",&P2.x);
printf("\nAl doilea punct: y="); scanf("%d",&P2.y);
//se calculeaza distanta dintre cele doua puncte:
d12=sqrt((P1.x-P2.x)*(P1.x-P2.x)+(P1.y-P2.y)*(P1.y-P2.y));
printf("Distanta dintre cele doua puncte este %lf",d12);
getch();
}
```

Programul nr. 2

Se defineşte în acest program structura complex, ce constă din două numere reale: partea reală şi partea imaginară a unui număr complex. Se va scrie o funcţie în care se calculează modulul unui număr complex, o funcţie în care se calculează conjugatul unui număr complex şi o altă funcţie în care se calculează suma a trei numere complexe.

Sursa programului:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
typedef struct{
                double re;
                double im;
                }complex;
double modulComplex(complex c);
complex sumaComplex(complex c1, complex c2,complex c3);
complex complexConjugat(complex c);
void afisareComplex(complex c);

void main(void){
complex c1,c2,c3,rezultat;
double modul;
clrscr();
printf("Partea reala si partea imaginara a primului numar");
scanf("%lf%lf",&c1.re,&c1.im);
printf("Partea reala si partea imaginara a numarului al doilea");
scanf("%lf%lf",&c2.re,&c2.im);
printf("Partea reala si partea imaginara a numarului al treilea");
scanf("%lf%lf",&c3.re,&c3.im);
modul=modulComplex(c1);
printf("\nModulul primului numar: %lf",modul);
```

```
modul=modulComplex(c2);
printf("\nModulul numarului doi: %lf",modul);
rezultat=sumaComplex(c1,c2,c3);
printf("\nSuma celor trei numere este:\n");
afisareComplex(rezultat);
rezultat=complexConjugat(c1);
printf("\nComplex conjugatul numarului");
afisareComplex(c1);
printf("este: ");
afisareComplex(rezultat);
getch();
}

double modulComplex(complex c)
{
double rez;
rez=sqrt(c.re*c.re+c.im*c.im);
return rez;
}

complex sumaComplex(complex c1,complex c2,complex c3)
{
complex rez;
rez.re=c1.re+c2.re+c3.re;
rez.im=c1.im+c2.im+c3.im;
return rez;
}

complex complexConjugat(complex c)
{
complex rez;
rez.re=c.re;
rez.im=-c.im;
return rez;
}

void afisareComplex(complex c)
{
if(c.im>0)printf("%.3lf+%.3lf",c.re,c.im);
else if(c.im==0)printf("%.3lf",c.re);
```

```
else printf("%.3lf-%.3lf",c.re,-c.im);
}
```

Programul nr. 3

Se citesc de la tastatură mai multe numere de tip complex, care se memorează într-un vector de numere complexe. Să se realizeze un program care să calculeze şi afişeze numărul de modul maxim.

Sursa programului:

```
#include<conio.h>
#include<stdio.h>
#include<math.h>
typedef struct{
                double re;
                double im;
                }complex;

double modulComplex(complex c);
#define N 2
void main(void)
{
 int i,indexMax;
 complex A[N];
 double ModulMaxim;
 double Modul[N];
clrscr();
 for(i=0;i<N;i++){
  printf("Partea reala a numarului %d: ",i);
  scanf("%lf",&A[i].re);
  printf("\nPartea imaginara a numarului %d: ",i);
  scanf("%lf",&A[i].im);
 }
 for(i=0;i<N;i++)
  Modul[i]=modulComplex(A[i]);
 ModulMaxim=Modul[0];
 indexMax=0;
 for(i=1;i<N;i++)
  if(Modul[i]>ModulMaxim){ModulMaxim=Modul[i];indexMax=i;}
 printf("%lf\n",A[indexMax].re);
 printf("%lf",A[indexMax].im);
 getch();
```

```
}

double modulComplex(complex c)
{
 double rez;
 rez=sqrt(c.re*c.re+c.im*c.im);
 return rez;
}
```

Programul nr. 4

Folosind structura punct, se vor citi de la tastatură N puncte, care vor fi introduse într-un vector. Să se realizeze un program care să afişeze daca toate cele N puncte sunt diferite intre ele sau nu.

Sursa programului:

```
#include<stdio.h>
#include<conio.h>
typedef struct {
                int x;
                int y; } punct;
#define N 3 //numarul de puncte
void main(void)
{
 punct p[N];//vectorul de puncte
 clrscr();
 int i,j;
 //citim cele N puncte in vectorul p:
 for(i=0;i<N;i++){
   printf("p[%d].x=",i);
   scanf("%d",&p[i].x);
   printf("p[%d].y=",i);
   scanf("%d",&p[i].y);}

 int suntDiferite=1;//presupunem ca sunt
 for(i=0;i<N-1;i++)
   for(j=i+1;j<N;j++)
     if((p[i].x==p[j].x)&&(p[i].x==p[j].y)){
          suntDiferite=0;
          break;}
 if(suntDiferite)printf("Sunt diferite.");
 else printf("Nu sunt diferite.");
```

```
 getch();
}
```

Programul nr. 5

Folosind structura punct, se vor citi de la tastatură N puncte, care vor fi introduse într-un vector. Să se realizeze un program care să afişeze distanţa maximă dintre două puncte.

```
#include<stdio.h>
#include<conio.h>
#include <math.h>
typedef struct {
                int x;
                int y; } punct;
#define N 3 //numarul de puncte
double dist(punct a, punct b); //distanta intre doua puncte
void main(void)
{
 punct p[N];//vectorul de puncte
 clrscr();
 int i,j;
 //citim cele N puncte in vectorul p:
 for(i=0;i<N;i++){
   printf("p[%d].x=",i);
   scanf("%d",&p[i].x);
   printf("p[%d].y=",i);
   scanf("%d",&p[i].y);}
 //Toate distantele sunt pozitive, deci initializam
 // distMax cu 0:

 double distMax=0;
 //Formam toate perechile de puncte posibile:
 for(i=0;i<N-1;i++)
   for(j=i+1;j<N;j++){
     double distCrt=dist(p[i],p[j]);//distanta intre p[i] si p[j]
     if(distCrt>distMax)distMax=distCrt;}
 printf("Distanta maxima=%lf",distMax);
 getch();
}
double dist(punct a, punct b)
{
```

```
 return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
```

Programul nr. 6

Se defineşte structura elev, ce conţine două câmpuri: numele de familie unui elev (de tipul string) şi nota obţinută de elev (de tipul int). Se citesc de la tastatură şi se memorează într-un vector de structuri elev, elevii unei clase şi notele obţinute de ei. Să se sorteze acest vector în ordinea descrescătoare a notelor elevilor.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
#define NR_ELEVI 20
typedef struct {
                 char nume[41];
                 int nota;
                } elev;
void main(void)
{
 elev tab[NR_ELEVI];
 int i;
 elev temp;
 int existaInversiuni;//variabila semafor
 for(i=0;i<NR_ELEVI;i++){
  printf("\n\nNumele elevului numarul %d: ",i+1);
  scanf("%s",tab[i].nume);
  printf("Nota elevului numarul %d: ",i+1);
  scanf("%d",&tab[i].nota);
 }
 //sortarea, prin metoda bubble sort:
 do{
  existaInversiuni=0;
  for(i=0;i<NR_ELEVI-1;i++)
   if(tab[i].nota<tab[i+1].nota){
     existaInversiuni=1;
     //se inverseaza strucura tab[i] cu structura tab[i+1]:
     temp=tab[i];
     tab[i]=tab[i+1];
     tab[i+1]=temp;}//if
 }while(existaInversiuni==1);
```

```
//Afisarea  elevilor:
 for(i=0;i<NR_ELEVI;i++)
    printf("%s   %d\n",tab[i].numetab[i].nota);
}
```

4. PROBLEME PROPUSE

1. Cu ajutorul structurii punct, definită anterior, să se realizeze un program care afişează dacă trei puncte ale căror coordonate se introduc de la tastatură, sunt sau nu coliniare.
2. Să se scrie o funcţie în care se realizează înmulţirea a doua numere complexe.
3. Se citesc N puncte într-un vector. Să se afişeze dacă sunt coliniare.
4. Folosind structura *elev* definită anterior, citim N elevi într-un vector. Să se afişeze toţi elevii care au nota > 7.