Laborator 10. Pointeri. Exemple de programe

10.1. Definiția unui pointer

Un pointer este o variabilă care are ca valoare o adresă de memorie. Pointerii se folosesc pentru a face referire la date care sunt cunoscute prin adresele lor.

10.2. Declararea unui pointer

Declararea unui pointer se realizează astfel:

```
tip *p;
```

Exemple:

```
int *int_ptr;           // pointer de tip întreg
double *real_ptr;       // pointer de tip real
char *sir;              // pointer de tip caracter
```

În primul exemplu, variabila *int_ptr* este declarată de tip pointer și conține adrese de memorie în care se păstrează date de tip *int*. **int_ptr* reprezintă conținutul zonei de memorie spre care pointează *int_ptr*, conținutul având tipul *int*.

În cel de-al doilea exemplu, *real_ptr* este un pointer către un double, iar în cel de-al treilea exemplu *sir* este un pointer către un caracter.

10.3 Operatori utilizați în relație cu pointerii

Accesul la conținutul unei zone de memorie cu adresa furnizată de pointerul p se face folosind operatorul unar * (asterisc, operator valoare sau operator de indirectare), de forma: *p.

Dacă a este o variabilă, atunci construcția &a reprezintă adresa zonei unde este memorată valoarea variabilei, iar caracterul & este operatorul unar care returnează adresa de memorie a operandului său. El este denumit operator adresă sau de referențiere.

10.4 Tablouri și pointeri

Fie șirul de elemente reale definit astfel:

```
float sir[10];
```

Elementul de indice i din șir poate fi referit printr-una din expresiile:

sir[i] sau *(sir+i).

Numele șirului este un pointer spre primul element din șir.

Fie matricea de elemente întregi definită astfel:

```
int a[10][10];
```

Elementul de indici i și j din matrice poate fi referit printr-una din expresiile:

a[i][j], *(a[i]+j) sau *(*(a+i)+j).

10.5 Alocarea dinamică de memorie

În vederea alocării dinamice se folosește funcția *malloc()*, definită în biblioteca *alloc.h*.

Exemplu: alocarea spațiului de memorie pentru un șir *b* cu *n* elemente:

```
int *b, n, i;
printf("\nn=");
scanf("%d",&n);
// alocare dinamica a spatiului de memorie pt sir
b=(int*)malloc(n*sizeof(int));
```

Exemplu: alocarea spațiului de memorie pentru o matrice pătrată *a* cu *nxn* elemente:

```
int **a,i,j,n,aux;
printf("\nn=");
scanf("%d",&n);
// alocare dinamica a spatiului de memorie pt matrice
a=(int**)malloc(n*sizeof(int*));
```

10.6 Eliberarea (ştergerea) memoriei alocate dinamic

În vederea ştergerii memoriei alocate dinamic se foloseşte funcţia *free()*, definită în biblioteca *alloc.h*.
De exemplu, pentru ştergerea memoriei alocate şirului *b* din exemplul de la paragraful anterior, se foloseşte apelul:

```
// eliberarea zonei de memorie
free(b);
```

iar pentru ştergerea memoriei alocate matricei *a*, se foloseşte:

```
// eliberarea zonei de memorie
free(a);
```

10.7. Exemple de programe

Ex1. Să se scrie un program care citeşte un număr întreg cuprins între 1 şi 7 şi afişează ziua din săptămână corespunzătoare numărului citit. Se vor folosi pointeri.

```
/*Tablouri de pointeri*/
#include <stdio.h>
#include <conio.h>
void main(void)
{
int c;
char* zile[]={
      "zi inexistenta",
      "luni",
      "marti",
      "miercuri",
      "joi",
      "vineri",
      "sambata",
      "duminica"
            };
printf("Introduceti numarul de ordine al zilei (1-7):");
scanf("%d",&c);
switch(c){
      case 1: printf("%s\n",*(zile+1)); break;
      case 2: printf("%s\n",*(zile+2)); break;
      case 3: printf("%s\n",*(zile+3)); break;
      case 4: printf("%s\n",*(zile+4)); break;
      case 5: printf("%s\n",*(zile+5)); break;
      case 6: printf("%s\n",*(zile+6)); break;
      case 7: printf("%s\n",*(zile+7)); break;
      default:printf("%s\n",*zile); break;
      }
printf("Apasati orice tasta pentru continuare\n");
getch();
}
```

Ex2. Să se scrie un program care determină elementul maxim şi poziţia acestuia dintr-un şir *x* de numere reale. Pentru accesarea elementelor şirului *x* se vor folosi pointeri.

```
/* Maximul elementelor unui sir*/
#include <stdio.h>
#include <conio.h>
#define MAX 100
```

```
void main(void)
{
int n,i,conditie,pozmax;
double *p;
double x[MAX],max;
do
{
  printf("Valoarea lui n=");
  scanf("%d",&n);
  conditie=n<=0||n>MAX;
  if(conditie)
      printf("dimensiune eronata: %d\n",n);
}
while(conditie);
for(i=0;i<n;i++){
      printf("Elementul x[%d]=",i);
      scanf("%lf",x+i);
                  }
max=*x;
pozmax=0;
for(p=x+1,i=1;i<n;p++,i++)
      if(max<*p){
            max=*p;
            pozmax=i;
                  }
printf("Valoarea maxima din sir este x[%d]=%lf\n",pozmax,max);
printf("Apasati orice tasta pentru continuare\n");
getch();
}
```

Ex3. Să se scrie un program care calculează produsul scalar a doi vectori *x* şi *y* având elemente numere reale. Se vor folosi pointeri.

```
/*Produsul scalar a doua siruri*/
#include<stdio.h>
#include<conio.h>
#define MAX 100
void main(void)
{
int i,n;
double *p,*q;
double x[MAX],y[MAX], s;
printf("Valoarea lui n=");
scanf("%d",&n);
while(n<=0||n>MAX){
      printf("Dimensiune eronata: %d\n",n);
      printf("Introduceti alt n: ");
      scanf("%d",&n);
                  }
for(i=0;i<n;i++){
      printf("Elementul x[%d]=",i);
      scanf("%lf",x+i);
                  }
for(i=0;i<n;i++){
      printf("Elementul y[%d]=",i);
      scanf("%lf",y+i);
                  }
for(p=x,q=y,i=0;i<n;i++,p++,q++)
      s+=*p* *q;
printf("(x,y)=%g\n",s);
printf("Apasati orice tasta pentru continuare\n");
getch();
}
```

Ex4. Să se scrie un program care calculează transpusa unei matrice pătrate $A_{n\times n}$. Se vor folosi pointeri.

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
void main (void)
{
      int **a,i,j,n,aux;
      printf("\n n=");
      scanf("%d",&n);
// alocare dinamica a spatiului de memorie pt matrice
      a=(int**)malloc(n*sizeof(int*));
// introducere matrice
      printf("Introduceti matricea, linie cu linie:\n");
      for(i=0;i<n;i++)
        for(j=0;j<n;j++)
            scanf("%d",*(a+i)+j);
// calcul matrice transpusa
       for(i=0;i<n;i++)
          for(j=i+1;j<n;j++)
          {
            aux=*(*(a+i)+j);
            *(*(a+i)+j)=*(*(a+j)+i);
            *(*(a+j)+i)=aux;
          }
       // afisare
       puts(" MATRICEA TRANSPUSA ESTE: ");
       for(i=0;i<n;i++)
        {
         for(j=0;j<n;j++)
           printf("%d ",*(*(a+i)+j));
         putchar('\n');
         }
       getch();
       // eliberarea zonei de memorie
       free(a);
   }
```

10.8 Exerciții propuse spre rezolvare

1. Să se scrie un program prin care să se calculeze valoarea matricei sumă: $S = A + B$. Pentru adresarea elementelor se vor folosi pointeri.
2. Să se scrie un program care calculează produsul a două matrice. Pentru adresarea elementelor se vor folosi pointeri. Produsul dintre două matrici se poate calcula doar dacă numărul de linii al primei matrici este egal cu numărul de coloane al celei de-a doua matrici. Dacă se citesc elementele matricelor *A[m][n]* şi *B[n][l]*, matricea produs dintre *A* şi *B* va fi de forma *P[m][l]*. Formula de calcul este:

$$P_{[i][j]} = \sum_{k=0}^{n-1} A_{[i][k]} \cdot B_{[k][j]}$$

3. Să se scrie un program care citeşte o variabilă întreagă şi pozitivă *n* şi construieşte matricea identitate I_n. Pentru adresarea elementelor matricei se vor folosi pointeri.
4. Să se scrie un program care citeşte un şir de numere reale A_n şi construieşte matricea care are pe diagonala principală elementele din şirul A şi valori zero în rest. Pentru adresarea elementelor şirului şi matricei se vor folosi pointeri.
5. Să se scrie un program care citeşte o variabilă întreagă şi pozitivă *n* şi construieşte matricea pătrată de dimensiuni *nxn*, care conține valori aleatoare întregi, cuprinse în intervalul 0..100. Se va utiliza funcția *rand*(). Pentru adresarea elementelor matricei se vor folosi pointeri.