Programarea calculatoarelor
Limbajul C

CURS 6

Pointeri şi tablouri

Pointeri şi funcţii

Pointeri şi vectori

- Exemplu:

```
int i;
double v[100], x, *p;
p=&v[0];          ->corect, neelegant
p=v;
x=v[5];
x=*(v+5);
v++;              ->incorect
p++;              ->corect
```

Obs:

p[4]=2.5 ->corect sintactic, dar nu este alocată memorie pentru p!!!

Transmiterea vectorilor ca argumente funcțiilor

- Apel:

```
void main(void)
{
   int v[10], n;
   …
   f(v,n);
…
}
```

- Sau:

```
void main(void)
{
   int *v, n;
   …
   f(v,n);
…
}
```

Transmiterea matricilor ca parametri funcțiilor

- Declarare funcții:

```
int min( int t[][NMAX], int m, int n){
  ...
}

int min( int *t [NMAX], int m, int n){
  ...
}

int min( int **t, int m, int n){
  ...
}
```

Transmiterea matricilor ca parametri funcțiilor

- Apel funcții:

```
int a[NMAX][NMAX], m, n;
…..
int mimim=min( a, m, n);
    ...
}
```

Exerciții – Pointeri şi vectori

1. Program pentru ordonarea unui vector de numere prin determinarea repetată a valorii maxime dintr-un vector şi schimbarea cu ultimul element din vector. Funcție care determină poziția valorii maxime dintr-un vector.
2. Bubble Sort
3. Determinare valoare minimă dintr-o matrice utilizand o funcție. Care ar fi cea mai potrivită funcție?
4. Funcții cu argument matrice (creare matrice unitate şi afisare matrice). Se impune numărul de coloane fix.

Rezolvari 5.1 Ordonarea unui vector de numere. Funcție care determină poziția valorii maxime din vector.

```
#include<stdio.h>
// determina pozitie maxim în vector de numere
int max ( float x[], int n) {
  int i, imax=0;                    // im = indice maxim
  for (i=1;i<n;i++)
     if ( x[i] > x[imax])           // x[im] este un maxim partial
       imax=i;
  return imax;
}
  // ordonare vector
void sort ( float x[], int n) {
  int imax;
  float aux;
  while ( n>1 ) {
     imax=max(x,n);
```

Rezolvari 5.1 Ordonarea unui vector de numere.

```
      // schimba x[imax] cu x[n-1]
      aux=x[imax];
      x[imax]=x[n-1];
      x[n-1]=aux;
      n--;                  // scade dimensiune vector
  }
}
  // verificare
int main() {
  float t[]= {5,2,9,1,4,6,2,8};
  int i, n = sizeof(t)/sizeof(t[0]);
  sort (t,n);
  for (i=0;i<n;i++)
     printf ("%.2f ",t[i]);
  getchar();
  return 0;
}
```

Rezolvări 5.2 BubbleSort

```
void      bubbleSort ( int *v, int n ) {
    int gata=0, i, aux;
    while (!gata) {
            gata=1;
            for (i=0; i<n-1; i++)
                    if (v[i] > v[i+1]) {
                            gata = 0;
                            aux = v[i];
                            v[i] = v[i+1];
                            v[i+1] = aux;
                    }
    }
}
```

Rezolvari 5.3 Determinare valoare minimă dintr-o matrice

```
float minim (float x[],int n);      // prototip funcție

int main() {
  float a[30][30],am[30],min;
        //am- vector cu minimul de pe fiecare linie
  int i,nl,nc;                  //nl=nr.linii, nc=nr.coloane
  ...                           // citire date
  for (i=0;i<n;i++)
        am[i]=minim(a[i],nc);   // minim din fiecare linie
  min= minim(am,nl);            // cel mai mic minim din linii
  ...
}
```

Rezolvari 5.4 Creare matrice unitate şi afişare matrice

```
 #include<stdio.h>
// generare matrice unitate
     void matrunit (float u[][30], int n) {
       int i,j;
       for (i=0;i<n;i++) {
             for (j=0;j<n;j++)
               u[i][j]=0;
             u[i][i]=1;
       }
     }
 // afisare matrice patratica (cu 30 de coloane declarate)
     void scrmatr (float a[][30], int n) {
       int i,j;
       for (i=0;i<n;i++) {
             for (j=0;j<n;j++)
               printf ("%4.0f",a[i][j]);
             printf("\n");
```

Rezolvari 5.4 Creare matrice unitate şi afisare matrice

```
      }
    }
      // utilizare
int main () {
      float x[30][30]; int n;
      for (n=2;n<=10;n++) {
            matrunit(x,n);
            scrmatr(x,n);
            getchar();                    // asteapta tasta "Enter“
              }
    return 0;
}
```

Transmiterea parametrilor

- Transmiterea parametrilor se face prin valoare - valorile parametrilor actuali sunt depuse pe stiva, la apelul unei funcții, fiind prelucrate ca parametri formali de către funcție;
- Modificarea parametrilor formali nu afectează deci parametrii actuali!
- Dacă parametrul este un tablou, cum numele este echivalent cu pointerul la tablou, funcția poate modifica valorile elementelor tabloului, primind adresa lui. A se observa că trebuie să se transmită ca parametri şi dimensiunea/dimensiunile tabloului/matricii.
- Dacă parametrul este şir de caractere, dimensiunea tabloului de caractere nu trebuie să se transmită, sfârşitul şirului fiind indicat de caracterul terminator '\0'.

Pointeri şi funcţii

- Transmiterea implicită a argumentelor se face prin valoare!
- Parametrii nemodificabili!

Exemplu:

```
#include<stdio.h>
void schimbare(int x, int y)  //se vor crea copii ale variabilelor x şi y
{    int tmp;
     tmp=x;
     x=y;
     y=tmp;          //în cadrul copiilor variabilelor se face inversarea
}                    //dar la revenire copiile variabilelor x şi y se distrug
```

Pointeri şi funcţii

continuare exemplu:

```
void main(void)
{
    int x=5, y=7;
    schimbare(x,y);              //transmitere prin valoare
    printf("%d %d\n",x,y);       /*valorile rămân
                  nemodificate adică se va afişa 5 7*/
}
```

Transmiterea prin referință

- Transmiterea prin referință – pointeri.
- Se pot modifica valorile de la adresele trimise ca parametri!
- Nu se pot modifica adresele trimise!

Transmiterea prin referință

```
#include<stdio.h>
void schimbare(int *x, int *y)  //se vor crea copii ale variab. x şi y
{
    int tmp;
    tmp=*x;
    *x=*y;
    *y=tmp;        //se face inversarea asupra zonei originale
}
void main(void)
{
     int x=5, y=7;
     schimbare(&x, &y);  //transmitere prin adresă
     printf("%d %d\n", x, y);
     /*valorile sunt inversate adică se va afişa 7 5*/
}
```

CONCLUZII

Pointerii permit:

- să realizăm modificarea unor valori trimise ca parametrii unei funcții
- să accesăm mult mai eficient tablourile.
- oferă mijlocul de a accesa indirect o valoare a unui tip de date.
- să lucrăm cu zone de memorie alocate dinamic

Exerciții rezolvate – pointeri şi funcții

1. Program pentru determinarea elementelor minim şi maxim dintr-un vector într-o aceeaşi funcție. Funcția nu are tip (void)!

Rezolvare 6.1. Determinare minim şi maxim dintr-un vector

```
#include<stdio.h>
void minmax ( float x[], int n, float* pmin, float* pmax) {
      float xmin, xmax;
      int i;
      xmin=xmax=x[0];
      for (i=1;i<n;i++) {
          if (xmin > x[i])  xmin=x[i];
          if (xmax < x[i]) xmax=x[i];
       }
       *pmin=xmin;
       *pmax=xmax;
   }
```

Rezolvare 6.1. Determinare minim şi maxim dintr-un vector

```
  // utilizare funcție
int main () {
  float a[]={3,7,1,2,8,4};
  float a1,a2;
  minmax (a,6,&a1,&a2);
  printf("%f %f \n",a1,a2);
  return 0;
}
```

Observație

```
  // NU!!!
int main () {
  float a[]={3,7,1,2,8,4};
  float *a1, *a2;
  minmax (a,6,a1,a2);
  printf("%f %f \n",*a1,*a2);
  getchar();
  return 0;
}
```

Exerciții propuse

1. Să se scrie o funcție care calculează valorile unghiurilor unui triunghi, în funcție de lungimile laturilor. Funcția va fi scrisă în două variante:
   - cu 6 argumente: 3 date şi 3 rezultate
   - cu 2 argumente de tip vector.