Programarea calculatoarelor
Limbajul C

CURS 6

Pointeri

Introducere

- *Definiție:*

Un pointer este o variabilă care păstrează *adresa unei date*, nu valoarea datei.

- Un pointer poate fi utilizat pentru referirea diferitelor date şi structuri de date. Schimbând adresa memorată în pointer, pot fi manipulate informații situate la diferite locații de memorie.
- Programul şi datele sale sunt păstrate în memoria RAM (*Random Access Memory*) a calculatorului.

Declarare şi operatori

- *Declarația* unei variabile pointer
  - se foloseşte operatorul de indirectare *:
  - tip_referit * var_pointer;
- *Inițializarea* cu adresa unei variabile:
  - tip_referit var,* var_pointer;
  - var_pointer=&var; // operator de referențiere
- *Valoarea* de la adresa indicată de pointer:
  - *var_pointer // este de tip_referit
  - *Operator de dereferențiere (indirectare)*

Operatori

- *Spațiul* ocupat de o variabilă pointer:
  - sizeof(var_pointer)
  - valoarea expresiei este 2 (în modelul small), oricare ar fi tip_referit
  - expresile de mai jos conduc la aceeaşi valoare 2:
    - sizeof(&var)
    - sizeof(tip_referit *)
- *Tipărirea* valorii unui pointer: se foloseşte prototipul %p, valoarea apărând sub forma a patru cifre hexa

Operatori

- *Adresa* unei variabile pointer este un pointer, la fel ca adresa unei variabile de orice alt tip:
  - &var_pointer

- *Observație:* Un pointer poate fi utilizat doar după inițializare, care se face prin atribuirea adresei unei variabile sau prin alocare dinamică.

Exemplu

```
int *p, n=5, m;
p=&n;
m=*p;
m=*p+1;
```

```
int *a,**b,c=1,d;
a=&c;
b=&a;
d=**b;          //d=1
```

```
int *p;
float x=1.23, y;
p=&x;
y=*p; //valoare eronata pentru y
```

Operatii cu pointeri

- Adunarea (scăderea) unei constante la un pointer

  tip*p;

  p++; ↔ p=p+sizeof(tip);

  p--; ↔ p=p-sizeof(tip);

  p=p+c; ↔ p=p+c*sizeof(tip);

  p=p-c; ↔ p=p-c*sizeof(tip);

- Scăderea a doi pointeri de acelaşi tip

- Compararea a doi pointeri (operații relaționale cu pointeri) = =, !=, <, >, <=, >=

Accesul la memorie

- Pointerul 0 este predefinit ca NULL şi semnifică faptul că nu indică nimic.
- void * înseamna un pointer de tip neprecizat
- Nu putem face operații aritmetice asupra acestor pointeri.
- Pointerii void ne permit păstrarea gradului de generalitate al unui program la maximum.

Pointeri şi vectori

- *Numele unui tablou* este un pointer constant spre primul său element.
- Expresiile de mai jos sunt echivalente:
  - nume_tablou
  - &nume_tablou
  - &nume_tablou[0]
- Şi de asemenea:
  - *nume_tablou
  - nume_tablou[0]

Pointeri şi vectori

- Declarații echivalente:
  - tip v[lim1][lim2]…[limn];
  - tip *…*v;
- Exemplu:
  - int v[10]; /*echivalent cu:*/
  - int *v;
  - v=(int *)malloc(10*sizeof(int)); // cu alocare dinamică!!!
- Referire elemente:
  - v[i]
  - *(v+i)

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

- void f(int *p, int n){
  …
}

- void f(int a[10] , int n){
  …
}

- void f(int a[] , int n) {
  …
}

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);
…
}
```

Exemplu

```
#include <stdio.h>
#include <stdlib.h>
#define N 5
int citire1(int tab[]){
/*citeste elementele lui tab prin accesarea indexata a elementelor */
      int i=0;
      printf("Introduceti elementele tabloului:\n");
      while(scanf(“%d”,&tab[i]!=EOF) i++;
      return i;
}
void tiparire1(int *tab, int n){
/* tipareste elementele tabloului prin accesarea indexata a elementelor */
      int i;
      printf("Elementele tabloului:\n");
      for(i=0;i<n;i++)
          printf("%d ",tab[i]);
      printf(”\n”);
}
```

Exemplu

```
int citire2(int tab[]){
/* citeste elementele lui tab - accesarea fiecarui element se face printr-un pointer la
      el */
      int *pi;
      pi=tab;
      printf("Introduceti elementele tabloului:\n");
      while(scanf(“%d”,pi)!=EOF) pi++;
      return pi-tab;
}

void tiparire2 (int tab[], int n){
/* tipareste elementele lui tab prin accesare prîn pointeri */
        int *pi;
        printf("Elementele tabloului:\n");
        for (pi=tab; pi<tab+n; pi++)
                printf("%d ",*pi);
        printf(“\n”);
}
```

Exemplu

```
int main(){
    int tab1[N], tab2[N], n, m;
    system("cls");
    n=citire1(tab1);
    tiparire1(tab1,n);
    m=citire2(tab2);
    tiparire2(tab2,m);
    getchar();
    return 0;
}
```

Transmiterea matricilor ca parametri funcțiilor

- Declarare funcții:

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

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

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

Transmiterea matricilor ca parametri funcțiilor

- Apel funcții:

```
int a[NMAX][NMAX], m, n;
…..
if (minmax( 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. Determinare valoare minimă dintr-o matrice utilizand o funcție. Care ar fi cea mai potrivită funcție?
3. Funcții cu argument matrice (creare matrice unitate şi afisare matrice). Se impune numărul de coloane fix.

Rezolvari 6.1 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 6.1 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;
}
```

Rezolvari 6.2 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 dîn fiecare linie
  min= minim(am,nl);            // cel mai mic minim dîn linii
  ...
}
```

Rezolvari 6.3 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 6.3 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 funcții

- 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 funcții

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 ununi tip de dată.
- să lucrăm cu zone de memorie alocate dinamic

Exerciții rezolvate – pointeri şi funcții

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

Rezolvare 6.4. 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 6.4. 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);
  getchar();
  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. Se tipăresc elementele a două matrici de întregi, cu acelaşi număr de coloane.
   Adăugați câte o funcție pentru fiecare din prelucrările:
   - citeşte o matrice
   - returnează suma elementelor unei matrici
   - însumează două matrici într-o a treia
   - înmulțeşte două matrici într-o a treia.

2. Program pentru citirea unui vector de întregi şi extragerea elementelor distincte într-un al doilea vector, care se va afisa. Se vor utiliza funcții. Ce funcții trebuie definite?

Exerciții propuse

3. Să se scrie o funcție de desenare a unei bare orizontale compuse din n caractere ch. Să se utilizeze această funcție pentru afişarea unei histograme ale cărei valori sunt memorate într-un vector (se va crea o funcție). Program de testare a funcțiilor scrise anterior.

4. 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.

Rezolvare 1. Tiparire elemente matrice

```
#include <stdio.h>
void tip ( int t[][10], int nl, int nc, char *nume){
        int i,j;
       printf("Elementele matricii %s:\n",nume);
       for(i=0;i<nl;i++){
         for (j=0;j<nc;j++)
         printf("%d ",t[i][j]);
       putchar('\n');
       } // for i
}
int main(){
       int t1[40][10]={{4,5},{3,4}}, t2[50][10]={{9,15,8},{13,99,14}};
       tip (t1, 2, 2, "t1");
       tip (t2, 2, 3, "t2");
       getchar();
       return 1;
}
```

Rezolvare 2 Extragere elemente distincte dintr-un vector

```
#define MAX 30
#include <stdio.h>
/* cauta pe x în vectorul a*/
int gasit(int v[],int n, int x){
      int m=0,i;
      for (i=0;i<n; i++)
            if (v[i]==x)  return i;
      return -1;
}

void main () {
       int a[MAX];            /* un vector de intregi*/
       int b[MAX];             /* aici se pun elementele distincte dîn a*/
       int n,m,i,j; /* n=dimensiune vector a, m=dimensiune vector b*/
```

Rezolvari 2 Extragere elemente distincte dintr-un vector

```
        printf("Numar de elemente vector n="); scanf("%d",&n);
         printf ("Introducere %d numere intregi:\n",n);
         /* citire vector*/
         for (i=0;i<n;i++) scanf("%d",&a[i]);

        m=0;
        for (i=0;i<n;i++)
              if(gasit(b,m,a[i])==-1) b[m++]=a[i];

        /* scrie vector b*/
         printf("Elementele distincte sunt:");
         for (j=0;j<m;j++)
           printf ("%5d",b[j]);
}
```

Rezolvare 3 Desenarea unei bare orizontale compuse din n caractere ch

```
#define M 20
#include <stdio.h>
 /* Desenarea unei bare orizontale dîn n caractere ch*/
void bar1 (int n, char ch) {
        int i;
        for (i=0;i<n;i++)
                putchar(ch);
        putchar('\n');
}

/*funcție de tip "void" cu argument de tip vector
 Desenare histograma pe baza unui vector de intregi*/
void hist (int a[], int n) {
     int i;
```

Rezolvare 3 Desenarea unei bare orizontale compuse din n caractere ch

```
for (i=0;i<n;i++)
	bar1(a[i],'*');
}

void main () {
	int a[M],n,i;
	scanf("%d",&n);
	for(i=0;i<n;i++)
		scanf("%d",&a[i]);
	hist (a,n);
}
```