Laborator 3 - Programarea calculatoarelor
Structuri statice

Tablouri

Un ***tablou*** de elemente reprezintă o structură de date alcătuită din mai multe variabile din acelaşi tip de date.

Tablouri unidimensionale

Tablourile unidimensionale sunt alcătuite dintr-un grup de elemente de acelaşi tip (numit *tip de bază*) şi sunt referite printr-un nume comun. Tablourile sunt stocate în memorie la locaţii consecutive, un tablou ocupând o zonă contiguă de memorie, cu primul element al tabloului aflat la adresa cea mai mică.

Variabilele de tip tablou se definesc astfel:

```
nume_tip nume_var[dimensiune];
```

Exemplu de declaraţie de tablou:

```
int v[4];          // declararea tabloului v, care are 4 elemente
```

Iniţializarea elementelor unui tablou se poate face în următoarele moduri:

```
v[0] = 21;
v[1] = 23;
v[2] = 27;
v[3] = 12;
```

sau:

```
int v[4] = { 21, 23, 27, 12 };
```

În limbajul C, un tablou se poate iniţializa şi fără dimensiune:

```
int d[] = { 1, 4, 6, 3 }; // tabloul va avea dimensiunea 4
```

Dacă nu se specifică dimensiunea unui tablou atunci când se declară, compilatorul va aloca suficientă memorie pentru a păstra elementele matricei respective.

Observaţie: în limbajul C numerotarea elementelor unui tablou începe cu poziţia 0. Astfel, daca avem definiţia:

```
int tablou[10];
```

atunci primul element al tabloului este *tablou[0]*, iar ultimul element al tabloului este *tablou[9]*.

Accesarea unui element al unui tabloului se face folosind ca index poziţia elementului. Astfel, tablou[3] va referi al 4-lea element al tabloului *tablou*.

Cantitatea de memorie necesară pentru memorarea unui tablou este determinată de tipul şi numărul elementelor tabloului. Pentru un tablou unidimensional, cantitatea de memorie ocupată se calculează astfel: mem_ocupată = dimensiune_octeţi * mărime_tablou.

Atenţie! o problemă legată de tablouri este că în C nu se face nici o verificare legată de "marginile" tabloului, astfel încât se pot accesa greşit elemente din afara tabloului. De exemplu, pentru definiţia:

```
int tablou[10];
```

dacă se încearcă accesarea elementului tablou[15] nu se va semnala nici o eroare, returnându-se valoarea de la o locaţie de memorie aflată la o distanţă de 5 locaţii faţă de sfârşitul tabloului, fapt ce poate duce la comportări "bizare" ale programului.

Aceeaşi situaţie, dar faţă de începutul tabloului, se întâmplă la încercarea de a accesa elementul tablou[-5].

Exemplu: Afișarea unui vector cu n elemente citite de la tastatură

```
#include<stdio.h>

int main(void)
{
	int n, v[10], i;
	printf("Dati numarul de elemente: ");
	scanf("%d", &n);
	//Citirea elementelor vectorului
	for (i = 0; i < n; i++)
	{
		printf("v[%d]=", i);
		scanf("%d", &v[i]);
	}
	//Afisarea elementelor vectorului
	for (i = 0; i < n; i++)
		printf("\nv[%d]=%d", i, v[i]);

	return 0;
}
```

Tablouri multidimensionale

Limbajul C oferă posibilitatea folosirii tablourilor multidimensionale. Din punct de vedere semantic, tablourile multidimensionale sunt “tablouri de tablouri”. Un tablou multidimensional se declară în modul următor:

```
nume_tip nume_tablou[dim_1][dim_2]…[dim_n]
```

Exemplu de declarare de tablou multidimensional (bidimensional, în cazul de faţă):

```
int matrice[3][3]     // declararea unui tablou 3 x 3
```

Valorile elementelor unui tablou multidimensional se pot declara atunci când se declară tabloul respectiv:

```
int matrice[3]3] = {{1, 3, 5},{2, 4, 6},{3, 6, 9} };
```

Se poate folosi şi următoarea reprezentare/declarare, pentru uşurarea vizualizării conţinutului tabloului:

```
int matrice[3][3] = {{1, 3, 5},{2, 4, 6}, {3, 6, 9}}
```

Exemplu: Afișarea unei matrici cu n linii și m coloane cu elementele citite de la tastatură

```
#include<stdio.h>

int main()
{
        int a[10][10], i, j, n, m;
        printf("Dati numarul de linii: ");
        scanf("%d", &n);
        printf("Dati numarul de coloane: ");
        scanf("%d", &m);
        //Citirea de la tastatura a elementelor matricii cu n linii si m
        coloane for (i = 0; i < n; i++)
                for (j = 0; j < m; j++)
                {
                        printf("a[%d][%d]=", i, j);
                        scanf("%d", &a[i][j]);
                }

        //Afisarea elementelor matricii
        for (i = 0; i < n; i++)
        {
                printf("\n"); //trece la rand nou, pentru a afisarea in forma de
                matrice for (j = 0; j < m; j++)
                        printf("%d ", a[i][j]);
        }
```

```
        return 0;
}
```

Constante de tip tablou

```
#include <stdio.h>
#define MAX 100
 int main
()
{
      int tab [MAX];
      int N, i, max;

      printf ("Câte numere doriţi? ");
      scanf ("%d", &N);

      for (i = 0; i < N; i++)
      {
            printf ("Introduceţi elementul %d: ",
            i); scanf ("%d", &tab[i]);
      }

      max = tab [0];

      for (i = 1; i < N; i++)
      {
         if (tab [i] > max)
               max = tab[i];
      }

      printf ("Maximul este: %d \n", max);
      return 0;
}
```

Constantele de tip tablou sunt constante în care se memorează valorile dintr-un tablou de elemente. Forma generală a declaraţiei unui tablou de constante este:

```
const [nume_tip] nume_tablou[dim_1][dim_n] = { val_1, val_2,…,val_n };
```

Exemplu de constantă de tip tablou unidimensional:

```
const int c[3] = { 34, 42, 22 };
```

Exemplu de constantă de tip tablou multidimensional:

```
const int d[2][3] = { {34, 42, 12}, {22, 35, 53} };
```

Exemplu: Următorul program citeşte de la tastatură elementele unui tablou de numere întregi si determină maximul dintre ele:

Generalități despre lucrul cu șiruri de caractere în limbajul C

Limbajul C nu dispune de un tip de date nativ pentru reprezentarea șirurilor de caractere de lungime variabilă. În acest scop se utilizează structuri de date de tip tablou de caractere.

Întrucât şirurile de caractere prelucrate în programe au în general lungime variabilă, s-a stabilit o convenţie prin care ultimul caracter utilizat al unui şir este urmat de un caracter cu valoarea zero ('\0'), numit terminator de şir.

```
char sir [10];
```

Exemplul de mai sus declară un tablou de 10 de elemente de tip caracter. Un asemenea tablou se poate folosi pentru memorarea unui şir de caractere de lungime variabilă, dar de maxim 9 de caractere, întrucât ultimul element este rezervat pentru terminatorul de şir.

Dacă şirul de mai sus conţine valoarea "TEST", conţinutul memoriei rezervate tabloului este următorul:

Index	0	1	2	3	4	5	6	7	8	9
Continut	T	E	S	T	**\0**	-	-	-	-	-

Orice şir de caractere care se prelucrează în program trebuie să dispună însă de o declaraţie de alocare de memorie (static sau dinamic).

Compilatoarele de C permit iniţializarea tablourilor de caractere în momentul declarării acestora cu un şir de caractere:

```
char sir [10] = "Un sir";
```

În urma acestei declaraţii, variabila *sir* va avea următorul conţinut:

Index	0	1	2	3	4	5	6	7	8	9
Continut	U	n		S	i	r	**\0**	-	-	-

Principalele funcții din biblioteca standard C pentru lucrul cu șiruri de caractere

Întrucat nu există tip de date nativ şir de caractere, limbajul nu dispune nici de operatori pentru prelucrarea şirurilor. Există însă o serie de funcţii care asistă programatorul în lucrul cu şirurile de caractere. Aceste funcţii necesită includerea fişierului header **string.h.** Toate funcţiile din această bibliotecă presupun că variabilele de tip şir de caractere asupra cărora operează respectă convenţia de a avea un terminator (caracter cu codul 0).

Citirea şi afișarea unui sir de caractere de la tastatură

Pentru citirea unui şir de caractere se poate utiliza funcţia *scanf* cu specificatorul de format %s. Pentru afişarea unui şir de caractere se poate utiliza funcţia printf, cu acelaşi specificator de format:

```
char sir [80];
printf ("Introduceti un sir: ");
scanf ("%s", sir);
printf ("Ati tastat: %s \n", sir);
```

Întrucât variabila şir este declarată ca şi tablou, numele acesteia reprezintă un pointer către primul element, de aceea nu se mai foloseşte operatorul adresă (&) în apelul scanf.

Utilizarea funcţiei scanf pentru citirea şirurilor de caractere are un dezavantaj major: nu se pot citi şiruri care conţin spaţii sau tab. De aceea, se recomandă utilizarea funcţiei *gets*:

```
char sir [80];
 printf ("Introduceti un sir: ");
gets (sir);
printf ("Ati tastat: %s \n", sir);
```

Lungimea unui șir de caractere

Pentru a determina lungimea unui şir de caractere se foloseşte funcţia:

```
int strlen (char *sir)
```

Funcţia returnează lungimea şirului de caractere primit ca şi parametru.

Exemplu:

```
char sir [80];
int n;
printf ("Introduceti un sir: ");
gets (sir);
n = strlen (sir);
printf ("Lungimea sirului este: %d \n", n);
```

Copierea conținutului unui șir de caractere

Pentru a copia conţinutul unei variabile sau constante de tip şir de caractere într-o variabilă tot de tip şir de caractere trebuie utilizată funcţia:

```
char *strcpy (char *destinatie, char *sursa)
```

Exemplu:

```
char sir1[80], sir2[80];
printf ("Introduceti un sir: ");
gets (sir1);
strcpy (sir2, sir1);
/* nu este permisa atribuirea sir2 = sir1 ! */
 printf ("Ati tastat: %s \n", sir2);
```

Este sarcina programatorului să se asigure că destinaţia are suficient spaţiu alocat pentru a memora toate caracterele din variabila sursă (inclusiv terminatorul de şir).

Compararea alfabetică a doua șiruri de caractere

Pentru compararea a două şiruri de caractere nu se pot aplica operatorii relationali între cele doua şiruri (<, >, <=, >=, ==, !=), întrucât aceştia au ca efect compararea adreselor în memorie ale celor două şiruri. De aceea trebuie utilizată această funcţie:

```
int strcmp (char *s1, char *s2)
```

Rezultatul funcţiei este:

- negativ dacă şirul s1 este mai mic decât s2 din punct de vedere al conţinutului;
- zero dacă s1 şi s2 au acelaşi conţinut;
- mai mare ca zero, dacă şirul s1 este mai mare decât şirul s2 din punct de vedere al conţinutului;

Exemplu:

```
char sir1 [80], sir2 [80];
int x;
printf ("Introduceti primul sir: ");
gets (sir1);
printf ("Introduceti al doilea sir: ");
gets (sir2);
 x = strcmp (sir1, sir2);
 if (x > 0)
            printf ("%s > %s \n", sir1, sir2);
  else
       if (x == 0)
             printf ("%s == %s \n", sir1, sir2);
       else
             printf ("%s < %s \n", sir1, sir2);
```

Căutarea unui subșir într-un șir

Pentru căutarea unui subsir într-un şir se utilizează functia *strstr*:

```
char *strstr (char *sir, char *subsir)
```

Funcţia returnează un pointer la prima apariţie a conţinutului variabilei *subsir* în conţinutul variabilei *sir* sau valoarea NULL dacă subsirul nu apare în şirul căutat.

Exemplu:

```
char sir [80], subsir [80];
char *p;
printf ("Introduceti sirul: ");
gets (sir1);
printf ("Introduceti subsirul cautat: ");
gets (sir2);
p = strstr (sir, subsir);
 if (p != NULL)
      printf ("%s contine %s \n", sir, subsir);
else
     printf ("%s nu contine %s \n", sir, subsir);
```

Concatenarea a doua șiruri de caractere

Pentru concatenarea a două şiruri de caractere se foloseşte funcţia:

```
char *strcat (char *dest, char *sursa)
```

Din nou, este responsabilitatea programatorului să se asigure că destinaţia are suficient spaţiu alocat pentru a memora toate caracterele din variabila sursă (inclusiv terminatorul de şir).

Exemplu:

```
char sir1[40], sir2[40], sir [80];
printf ("Introduceti primul sir: ");
gets (sir1);
printf ("Introduceti al doilea sir: ");
gets (sir2);
strcpy (sir, sir1);
strcat (sir, sir2);
printf ("Ati introdus: %s \n", sir);
```

Probleme rezolvate

Exemplu: Programul următor citeşte de la tastatură un șir de caractere, iar apoi transformă literele mici in litere mari.

```
#include <stdio.h>
#include <string.h>
#include <ctype.h> /* pentru toupper */
```

```
int main()
{
char s[20];
int i;
gets(s);
for(i=0;i<strlen(s);i++)
printf("%c",toupper(s[i]));
return 0;
}
```

Exemplu: următorul program citeşte de la tastatură un şir de caractere şi afişează şirul în ordine inversă:

```
#include <stdio.h>
#include <string.h>
int main (void)
{
  char sir [80];
  int k, n;
  printf ("Introduceti sirul: ");
  gets (sir);
  n = strlen (sir);
  for (k = n - 1; k >= 0; k--)
      printf ("%c", sir [k]);
printf ("\n");
return 0;
}
```

Exemplu: următorul program citeşte un şir de caractere reprezentând un număr în baza 2 şi afişează valoarea acestuia în baza 10:

```
#include <stdio.h>
#include <string.h>
int main (void)
{
  char sir [80];
  int k, n, val, p, cifra, valid;
       do {
   do
     {
              printf ("Introduceti sirul: ");
              gets (sir);
             n = strlen (sir);
           }while (n == 0);

        valid = 1;

        for (k = 0; k < n; k++)
             if ((sir [k] != '0') && (sir [k] != '1'))
                        valid = 0;
             if (!valid)
               printf ("Sirul introdus nu este numar in baza 2 !
      \n"); }while (!valid);
      p = 1;     /* puterea lui 2 */
      val = 0; /* valoarea numarului */
```

```
        for (k = n - 1; k >= 0; k--)
        {
           cifra = sir [k] – '0';
           val = val + cifra * p;

           p = p * 2;
        }
        printf ("Numarul %s in baza 10 este: %d \n",sir,val);
        return 0;
}
```

Probleme propuse

1. Să se scrie un program în C care să citească de la tastatură două matrice pătratice de numere întregi, de dimensiune specificată de utilizator şi să afişeze suma celor două matrice.

2. Să se scrie un program în C care, folosind un meniu interactive să conțină următoarele opțiuni:
   a. Citirea unui vector cu n elemente
   b. Afișarea vectorului
   c. Afișarea elementelor de pe poziițile pare
   d. Afișarea produsului elementelor impare
   e. Ieșire

3. Să se calculeze suma elementelor de pe diagonala principală a unei matrice pătratice citite de la tastatură.

4. Să se scrie un program în C care să citească de la tastatură o matrice pătratică. Să se creeze un meniu interactive cu următoarele opțiuni:
   a. Afișarea matricii pătratice
   b. Să se afișeze suma numerelor pare deasupra diagonalei principale
   c. Să se afișeze produsul elementelor impare de pe diagonal secundară
   d. Să se afișeze elementele prime din matrice
   e. Iesire

5. Să se scrie un program C care să citească de la tastatură un cuvânt şi să verifice dacă respectivul cuvânt este palindrom (cuvânt care poate fi citit de la stânga la dreapta şi de la dreapta la stânga fără să-şi piardă sensul: cojoc, capac, rar).

6. Să se scrie un program C care realizează următoarele:
   a. Şterge dintr-un şir de caractere un subşir specificat prin poziţie şi lungime.
   b. Inserează într-un şir începând cu o poziţie dată un al şir.
   c. Citeşte doua cuvinte şi înlocuieşte într-un text introdus de la tastatură toate apariţiile primului cuvânt prin cel de-al doilea.