CAPITOLUL 8

8. Masive de date. Tablouri

În majoritatea calculelor ştiinţifice se foloseşte noţiunea de vector, matrice, lucrul în spaţii n-dimensionale. Limbajul C a fost prevăzut are posibilitatea de a lucra cu un masive de date de acelaşi tip, masive ce pot fi recunoscute sub un nume unic. Li se mai spune şi tablouri de date. Există posibilitatea definirii statice de tablouri cu maximum trei dimensiuni. Declararea unui tablou se realizează ca mai jos:

- tablou monodimensional:
  tip-dată nume-tablou[nr-elemente]; (ex: char s[10])
- tablou bidimensional:
  tip-dată nume-tablou[nr-linii][nr-coloane]; (ex: int m[20][20])
- tablou tridimensional:
  tip-dată nume-tablou[coord x][coord y][coord z]; (ex: float punct[10][10][10]).

Aceste declaraţii au ca efect rezervarea unui spaţiu de memorie pentru depozitarea 10 caractere sau 100 de întregi sau 1000 de valori reale pentru cazurile prezentate ca exemplu. Accesul la fiecare zonă a tabloului se face poziţional. Elementele sunt depuse unul după altul şi atunci referirea se face printr-un indice care specifică poziţia elementului respectiv (al câtelea este în şir). Deci pot exista referiri de genul:

s[3]=’2’ sau m[2][2]=30 sau punct[0][0][0]=0.233445.

Indicele poate fi furnizat prin intermediul unei variabile. Acest lucru permite folosirea unor instrucţiuni ciclice pentru prelucrarea datelor din cadrul unui tablou de date. O situaţie mai specială este în cazul tablourilor monodimensionale (şirurilor) de caractere. Cum, în general, prelucrarea acestora nu se face caracter cu caracter, ele fiind de obicei folosite în manipularea informaţiei scrise, limbajul C a fost prevăzut cu o serie de funcţii specializate în prelucrarea globală a lor.

8.1. Tablouri monodimensionale

După cum am spus tablourile monodimensionale pot fi folosite şi pentru calcule matematice în spaţii n-dimensionale, unde se lucrează cu un set de n coordonate. Vom prezenta câteva implementări ale operaţiunilor simple cu vectori.

8.1.1. Prelucrări simple cu vectori

În cazul declarării unui vector se rezervă un spaţiu pentru numărul maxim de elemente care s-ar putea să aibă nevoie utilizatorul programului. De exemplu cazul unui program specializat pentru calcule asupra unui număr de măsurători, utilizatorul poată să facă într-un caz 10 măsurători (o estimare mai grosieră) şi în altul 100. Deci se va defini un vector cu 100 de elemente dar în acelaşi timp trebuie ca programul să ceară specificarea numărului de valori ce trebuie introduse şi să facă procesarea numai asupra lor. Va trebui să folosim o instrucţiune for… deoarece are numărul de paşi cunoscut. În cazul în care se poate specifica nici o limită pentru numărul de valori ce trebuiesc stocate sau prelucrate nu se mai pot folosi tablouri definite static ci liste înlănţuite. Mai jos prezentăm un program care

citeşte doi vectori de aceeaşi dimensiune de la tastatură şi realizează o serie de operaţiuni simple cu ei afişând de fiecare dată rezultatele.

```
#include <stdio.h>
#include <conio.h>

void main (void)
    {
    int v1[100], v[100],v2[100],i,j,n;
    clrscr();
    printf("Dati dimensiunea vectorilor n=");
    scanf("%d",&n);

    for(i=0;i<n;i++)
        {
        printf("V[%d]=",i);
        scanf("%d",v[i])        }//citirea vectorului v
    for(i=0;i<n-1;i++)
        for(j=i+1;j<n;i++)
         if(v[i]>v[j])
                    {
                    t=v[i];
                    v[i]=v[j];
                    v[j]=t;
                    }
    printf("\nVectorul ordonat:\n")
    for(i=0;i<n;i++) printf("%d ",v[i]);// afisarea vectorului v
    for(i=0;i<n;i++)
        {
        printf("V1[%d]=",i);
        scanf("%d",v1[i]);
        }
    for(i=0;j=0;i<n;i++;j++)
        {
        v2[j]=v1[2*i];
        v2[j+1]=v[2*i+1];
        }
    printf("\nVectorul v interclasat cu v1 este:\n")
    for(i=0;i<n;i++) printf("%d ",v[i]);
    for(i=0;i<n;i++) v2[i]=v1[i]+v[i];
    printf("\nVectorul v sumat cu v1 este:\n")
    for(i=0;i<n;i++) printf("%d ",v[i]);
    getch();
    }
```

Mai jos este prezentat codul care implementează algoritmul "merge sort", bazat pe tehnica "împarte şi stăpâneşte", într-o ,manieră recursivă.

```
#include <stdio.h>

int a[100],n; // variabile globale vizibile şi în main şi în interiorul
              //oricărei funcţii definite de utilizator
```

```
    merge_sort(int st,int dr); // prototipurile funcțiilor proprii
    merge(int st,int mij,int dr);
    void main(void)
        {
         int i;
         printf("\n introd dimensiunea sirului de sortat :  ");
         scanf("%d",&n);
         for(i=0;i<n;i++)
         {
         printf("\nV[%d]=",i);
             scanf("%d",&a[i]);
             }
         merge_sort(0,n-1);
         printf("\n sirul ordonat este :\n");
         for(i=0;i<n;i++)
           printf("%3d  ",a[i]);
        }

merge_sort(int st,int dr)
        {
         int mij;
         if(st<dr){
         mij=(st+dr)/2;
         merge_sort(st,mij);
         merge_sort(mij+1,dr);
         merge(st,mij,dr);
         }
        }

merge(int st,int mij,int dr)
        {
         int h,i,j,k;
         int b[100];
         h=st;
         i=st;
         j=mij+1;
         while(h<=mij && j<=dr){
         if(a[h]<=a[j]) {b[i]=a[h];h++;}
         else {b[i]=a[j];j++;}
         i++;
         }
         if(h>mij)
         for(k=j;k<=dr;k++)
                 {b[i]=a[k];i++;}
         else for(k=h;k<=mij;k++)
                 {b[i]=a[k];i++;}
         for(k=st;k<=dr;k++) a[k]=b[k];
```

}

8.1.2. Şiruri de caractere

S-a discutat despre setul de caractere ASCII şi setul de caractere de control. Acestea sunt introduse pentru manipularea unei informații care poate fi doar afişată pe ecran sau la imprimantă. Specific acestei informații este că de obicei manipulează în pachete de mai mult de un caracter. În aceste condiții trebuie folosit un tablou de caractere. Acesta este denumit şir de caractere. El are în plus față de un vector adăugat întotdeauna la sfârşitul datelor utile un caracter special '\0' numit terminator de şir. A fost introdus pentru a uşura scrierea funcțiilor (de bibliotecă sau utilizator) folosite în prelucrarea unui şir de caractere.

Şirurile de caractere au exact aceleaşi caracteristici ca oricare tablou monodimensional.

Inițializarea uni şir de caractere se poate face prin mai multe metode:

- la declarare, fără definirea dimensiunii tabloului

  char s[]=".doc"

  în acest caz dimensiunea tabloului va fi exact numărul de caractere cu care e inițializat şirul.
- la declarare cu definirea dimensiunii tabloului

  char s[20]="doi"

- direct în program : *s="Dati valoarea"*
- prin citire de la tastatură

  cu procedură: gets(s)
  cu funcție: scanf("%s",s);
- prin inițializare caracter cu caracter în cazul unei prelucrări în program

  for(i=0;i<strlen(s1);i++)s2[i]=s1[i]+0x25;

În ultimul caz apare o problemă, în cazul şirului s2 nu este pus automat caracterul '\0' şi deci trebuie OBLIGATORIU introdus explicit. În exemplul de mai sus acest lucru se realizează prin adăugarea instrucțiunii:

s[i]='\0';

Prezentăm mai jos un program simplu ce inițializează în cod, element cu element un şir de caractere apoi afişează diverse poțiuni din el:

```
#include <stdio.h>

void main(void)
        {
        char nume[5];       /* definesc un şir de caractere */

          nume[0] = 'M';
          nume[1] = 'i';
          nume[2] = 'h';
          nume[3] = 'a';
```

```
  nume[4] = 0;     /* caracterul ce marchează sfârşitul */

  printf("Numele este  %s\n",nume);
  printf("O litera din el este %c\n",nume[2]);
  printf("O parte din nume este %s\n",&nume[1]);
getch();
}
```

8.2. Tablouri bidimensionale. Implementarea noțiunii de matrice

Pentru calcule matematice tablourile bidimensionale implementează exact conceptul de matrice şi pot fi manipulate în cod aproape la fel ca în matematică.

Apare aceeaşi problemă relativ la modul în care alegem dimensiunea maximă a unei matrici ca şi în cazul vectorilor. Există cele două cazuri:

- Utilizatorul ştie exact cât este dimensiunea maximă a matricilor care vor fi folosite, acesta este cazul ideal care este relativ rar.
- Utilizatorul are nişte cerințe vagi şi programatorul prin discuție directă poate afla acest maxim sau poate să-şi facă o idee despre el caz în care va lua o valoare acoperitoare.

În orice caz oricare ar fi dimensiunea unui tablou el are un număr total de elemente care va fi alocat static. Acest număr se poate afla conform formulelor de mai jos:

- tablou monodimensional: Nr_elem=dim_vect*sizeof(tip_data);
- tablou bidimensional: Nr_elem=dim_x*dim_y*sizeof(tip_data);
- tablou tridimensional Nr_elem = dim_x* dim_y *dim_z * sizeof(tip_data).

Deja apare o problemă relativ la spațiul maxim care poate fi ocupat de totalul datelor definite static (adică cum am vorbit până acum), şi anume faptul că chiar sub modelul de compilare HUGE nu se poate aloca mai mult de 1Mo=1024*1024 octeți pentru date. Deci implicit apare automat o limitare asupra dimensiunii maxime care o poate lua un tablou.

Deci indiferent de cerințele utilizatorului nu se poate depăşi această dimensiune. În cazul în care specificațiile depăşesc aceste dimensiuni există două metode:

- metoda swap-ului;
- metoda listelor de ordin n, unde n este conform necesităților.

În general aceste metode sunt mixate chiar şi la nivelul aplicațiilor actuale. Metoda swap-ului se concretizează prin scrierea într-un fişier a datelor ce depăşesc aceste dimensiuni maxime. Practic dispozitivul de stocare devine un fel de prelungire a memoriei operative, având însă dezavantajul că timpii de acces sunt cam de 10 ori mai mari decât în cazul lucrului cu RAM, ceea ce va duce la scăderea vitezei de execuție a respectivei aplicații. Metoda listelor se bazează pe alocarea dinamică care ne permite accesul la toată memoria fizică RAM neocupată de alte programe.

Mai jos vom prezenta un scurt exemplu de folosire a tablouri pentru implementarea de operațiuni simple cu matrici.

```
#include <stdio.h>
#include <conio.h>

void main(void)
    {
    int m1[5][5],m2[5][5],m3[5][5],m4[5][5],i,j,k,m,n,p;
    clrscr();
    printf("n="); scanf("%d",&n);// se poate scrie şi aşa în cod
    printf("m="); scanf("%d",&m);// dar nu este recomandat
    printf("p="); scanf("%d",&p);// am citit dimensiunile matricilor
    for(i=0;i<n;i++)
        for(j=0;j++;j<m)
        {
        printf("\nA[%d,%d]=",i,j);
        scanf("%d",&m1[i][j]);
        }// am terminat de citit prima matrice
    // se observă ca numele sub care o citesc nu are legătură cu
    //modul în care o denumesc în program.
    for(i=0;i<m;i++)
        for(j=0;j++;j<p)
        {printf("\nB[%d,%d]=",i,j);
        scanf("%d",&m2[i][j]);        }// se poate scrie şi aşa în cod
    // deci am citit cele două matrici, acum vom începe procesarea.
    for(i=0;i<n;i++)
        for(j=0;j++;j<p)
        {
        m3[i][j]=0;
        for(k=0;k++;k<m)
                m3[i][j]+=m1[i][k]*m2[k][j];
        }// înmulțirea a două matrici
    printf("Matricea Cnp=Amn*Bnp este:")
    for(i=0;i<n;i++)
        {printf("\n")
        for(j=0;j++;j<p)printf("%d ",m3[i][j]);
        }
    for(i=0;i<n;i++)
        for(j=0;j++;j<p)
        m4[i][j]=m3[j][i];
    printf("Transpusa lui C este ");
    for(i=0;i<p;i++)
        {printf("\n")
        for(j=0;j++;j<n)printf("%d ",m4[i][j]);
        }
    getch();
    }
```

Deoarece tablourile bidimensionale de caractere nu sunt folosite uzual, nu le vom trata.