Variabile indexate

Variabila indexata este cunoscuta si sub denumirea de masiv (tablou), si este definita ca fiind *o structură de date omogenă,* din punct de vedere al tipului elementelor sale, elementele fiind accesibile în mod direct, conform poziţiei ocupate în tablou.

Un element al unui tablou este specificat printr-o *variabilă cu indici* (*variabilă indexată*), formată pe baza identificatorului asociat datei de tip tablou şi a unor expresii aritmetice ce precizează valoarea indicilor, prin care se determină în mod unic elementul referit.

- Se întâlneşte de asemenea noţiunea de **vector**, care este de fapt un tablou unidimensional, deci un tablou ale cărui elemente sunt selectate pe baza unui singur indice.

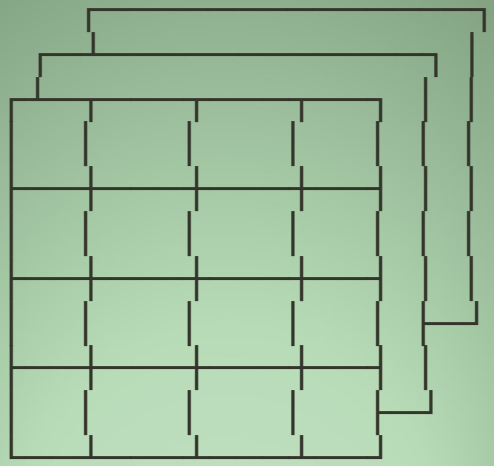
- Pentru masivul bidimensional este folosită noţiunea de **matrice**. Se pot defini şi masive (tablouri) multidimensionale.

OBSERVATIE

1. un tablou multidimensional este tratat ca un tablou ale cărui elemente sunt la rândul lor tablouri.
2. In declaraţia unui tablou, parantezele pătrate se asociază de la stânga la dreapta:
   - 3 – nr. planurilor
   - 4 – nr. liniilor din fiecare plan
   - 4 - nr. coloanelor din fiecare plan

Exemplu:

```
int t[3][4][4];
```



Variabile indexate

Declararea variabilelor indexate se face la fel cu cel al variabilelor simple, cu precizarea suplimentară a dimensiunii.

Exemplu:

```
#define N 80
#define M 80
int   sir[ N ];
float mat[ N ][ M ], vect[ N ];
```

In exemplul prezentat s-au declarat:

- sir - vector de N întregi,
- mat- matrice cu N linii şi M coloane având elemente reale.

Sintaxa generală de declarare a unei variabile indexate este:

```
<tip> <identificator> [<exp_ct_1>] [< exp_ct_2>]...[< exp_ct_n>];
```

Referirea la un element al masivului se face conform următoarei sintaxe generale:

```
<identificator>[<expresie_int1>][<expresie_int2>]...
```

unde

<expresie_intx> - este o expresie a cărei valoare trebuie să fie întreagă şi pozitivă.

Exemplu:

```
mat[ 0 ][ 0 ] = 0;
mat[ i ][ j ] = sir[ 2 ];
vect[ i+1 ] += 1;
vect[ i++ ] = vect[ 0 ];
```

Variabile indexate

Iniţializarea tablourilor

La fel ca şi iniţializarea variabilelor simple (atomice), iniţializarea tablourilor este permisă odată cu declararea lor.
Prin astfel de declaraţii se specifică tipul variabilei, se determină alocarea unei zone de memorie corespunzătoare acestui tip şi se iniţializează variabila cu expresia de după semnul egal.

Exemplu:

```
int x[] = {1, 2, 3};
char s[] = {'B', 'y', '\0'};
```

are ca efect alocarea pentru tabloul unidimensional a 3 x 2 octeţi (*considerând că o variabilă int se reprezintă pe doi octeţi*), cu conţinutul 1, 2, 3
(x[0]=1, x[1]=2, x[2]=3).

Exemplu:

```
int A[ 4 ][ 3 ] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
```

are ca efect alocarea a 12 întregi, elementele matricii având valorile:

A[0][0]=1
A[0][1]=2
A[0][2]=3
A[1][0]=4
.....

Observatie

Se observă că matricea este memorată "*pe linii*" (considerând elementele tabloului bidimensional A în ordinea în care apar în memorie, ultimul indice variază cel mai repede).

Variabile indexate

Exemplu:

```
int A[4][3] = { {1,2,3}, {4,5,6}, {7,8,9}, {10,11,12} };
```

Astfel, se pune în evidenţă faptul că matricea A are 4 elemente, fiecare element fiind un tablou unidimensional de 3 elemente.

Exemplu:

```
int A[][3]={1,2,3,4,5,6,7,8,9,10,11,12};
```

Unde nu s-a precizat prima dimensiune a tabloului.
Lipsa dimensiunii într-o declaraţie de tip în care sunt prezente iniţializări, face ca dimensiunea alocată să fie egală cu dimensiunea zonei iniţializate efectiv.

Observatie

De reţinut ca regulă generală, faptul că numai ***prima dimensiune*** a unui tablou multidimensional poate rămâne neprecizată, în caz contrar, compilatorul nu mai are informaţiile necesare pentru calculul adresei unui element de tablou.

Exemplu:

```
int M[2][3]={{1}, {2}};
int M[2][3]={{1,0,0}, {2,0,0}};
```

Declaratiile sunt echivalente!

Observatie

Pentru tablouri întregi, în cazul în care sunt precizate dimensiunile, iar lungimea zonei iniţializate este mai mică decât această dimensiune, se completează cu 0.

Variabile indexate

TABLOURI. Probleme I

Să se scrie un program care să citească un şir de întregi şi să afişeze elementele pare de pe poziţiile impare ale şirului şi să le numere.

```
Introduceti numarul de elemente 5
vect[0]=1
vect[1]=2
vect[2]=3
vect[3]=4
vect[4]=5

Elementele pare de pe pozitiile impare sunt:
        2       4
Nr. elementelor pare de pe pozitiile impare este: 2
```

```c
#include <stdio.h>
#define N 20
void main( void )
{
    int n, i, nr_par = 0;
    int vect[ N ];
    do
    {   printf("\nIntroduceti numarul de elemente ");
        scanf("%d", &n );
    } while( n <= 0 || n > N );
    for( i = 0; i < n; i++ )
    {
        printf( "vect[%d]=", i );
        scanf( "%d", &vect[ i ] );
    }
    printf( "\nElementele pare de pe pozitiile impare sunt:\n" );
    for( i = 1; i < n; i += 2 )
        if( vect[ i ] % 2 == 0 )
        {
            printf( "\t%d ", vect[ i ] );
            nr_par++;
        }
    printf( "\nNr. elementelor pare de pe pozitiile impare este: %d\n",
        nr_par);
}
```

2019/2020 - Suceava

Variabile indexate

TABLOURI. Probleme II

Să se scrie programul care citește un șir precedat de numărul elementelor. Se vor afișa toate valorile distincte din șir urmate de numărul lor de apariţii.

```c
#include <stdio.h>
#define N 20
void main()
{
    int n, i, j, numar, lim = 0;
    int a[ N ][ 2 ];
    do{ printf("\nIntroduceti numarul de elemente ");
        scanf( "%d", &n );
    } while( n <= 0 || n > N );
    //Plaseaza numarul in a[][]
    for( i = 0 ; i < n ; i++ )
    {   printf( "Elementul %d = ", i + 1 );
        scanf( "%d", &numar );
        for( j = 0; j < lim; j++ ) //Unde este nr.
            if( numar == a[ j ][ 0 ] )
            {   a[ j ][ 1 ]++;
                break;
            }
        if( j == lim ) //Plasa elementul in vector
        {   a[ lim ][ 0 ] = numar;
            a[ lim ][ 1 ] = 1;
            lim++;
        }
    }
    for( i = 0; i < lim; i++ ) //Afiseaza numerele
        printf( "\nNr. %d a aparut de %d ori in sir",
            a[ i ][ 0 ], a[ i ][ 1 ] );
}
```

```
Introduceti numarul de elemente 5
Elementul 1 = 4
Elementul 2 = 2
Elementul 3 = 12
Elementul 4 = 3
Elementul 5 = 4

Nr. 4 a aparut de 2 ori in sir
Nr. 2 a aparut de 1 ori in sir
Nr. 12 a aparut de 1 ori in sir
Nr. 3 a aparut de 1 ori in sir
Process returned 4 (0x4)   execut
```

Variabile indexate

TABLOURI. Probleme II

Tehnica utilizata:

Vom utiliza un tablou cu 2 coloane; în prima coloană se memorează numerele distincte, în a doua coloană, numărul de apariţii.

Programul nu memorează şirul introdus de utilizator, ci reţine în matricea a[][] valorile distincte şi numărul lor de apariţii.

Conform enunţului, rezervarea de memorie pentru stocarea şirului iniţial este complet inutilă.

Observatie

Instrucţiunile

```
a[ lim ][ 1 ] = 1;
lim++;
```

pot fi înlocuite cu:

```
a[ lim++ ][ 1 ] = 1;
```

2019/2020 - Suceava

Variabile indexate

TABLOURI. Probleme III

Citiţi elementele unui vector şi determinaţi minimul acestor elemente şi poziţia (respectiv poziţiile) acestuia. Intr-o primă versiune a rezolvării programului, se caută minimul valorilor elementelor şirului, după care se parcurge din nou şirul, afişându-se poziţiile în care s-a găsit minimul.

```c
#include <stdio.h>
#define N 20
#define TRUE 1
void main()
{
    int vect[ N ];
    int n, i, min = 32767;
    do
    {   printf( "\nCate elemente are vectorul ? " );
        scanf( "%d", &n );
        if( n > 0 && n <= N ) break;
    }while( TRUE );
    for( i = 0; i < n; i++ )
    {
        printf( "vect[ %d ]= ", i );
        scanf( "%d", &vect[ i ] );
        if( min > vect[ i ] ) min = vect[ i ];
    }
    printf( "\nMinimul este %d si se gaseste in:\n", min );
    for( i = 0; i < n; i++ )
        if( min == vect[ i ] ) printf("\tpozitia %d \n", i );
}
```

```
Cate elemente are vectorul ? 5
vect[ 0 ]= 3
vect[ 1 ]= 1
vect[ 2 ]= 2
vect[ 3 ]= 3
vect[ 4 ]= 1

Minimul este 1 si se gaseste in:
        pozitia 1
        pozitia 4
```

2019/2020 - Suceava

Variabile indexate

TABLOURI. Probleme IV

Să se calculeze valoarea polinomului P(x) pentru o valoare x dată.
P(x) = An*X^n + An-1*X^n-1 + ... +A1*X + A0

```c
#include <stdio.h>
#define N 100

int main( void )
{
    int i, n;
    float a[ N ];
    float x, p;
    do
    {   printf("\nIntroduceti gradul polinomului: ");
        scanf("%d", &n );
    }while ( n < 0 || n > N-1 );
    // Calculul sumei finale ...
    printf("\nIntroduceti coeficientii polinomului\n"
           "P(X) = A[n]*X^n + A[n-1]*X^n-1 + ... +A[1]*X + A[0].\n" );
    for ( i = n; i >= 0; i-- )
    {   printf( "A[%d]= ", i );
        scanf( "%f", &a[ i ] );
    }
    printf( "\nIntroduceti valoare dorita x=" );
    scanf( "%f", &x );
    p = 0;
    for ( i = n; i >= 0; i--)
        p = p + pow( x, i ) * a[ i ];
    printf("\nValoarea polinomului este %5.2f", p );
    return 0;
}
```

```
Introduceti gradul polinomului: 2

Introduceti coeficientii polinomului
P(X) = A[n]*X^n + A[n-1]*X^n-1 + ... +A[1]*X + A[0].
A[2]= 1
A[1]= 2
A[0]= 3

Introduceti valoare dorita x=2

Valoarea polinomului este 11.00
Process returned 0 (0x0)   execution time : 8.703 s
```

Variabile indexate

TABLOURI. Probleme V

Să se citească o matrice n x m de numere întregi. Să se determine un vector de n elemente conţinând suma elementelor de pe fiecare linie.

```
#include <stdio.h>
#define N 100
void main()
{
    int i, j, nl, nc;
    int m[N][N], v[N];
    do
    {   printf("\nIntroduceti numarul de linii si coloane "
               "\nseparate prin virgula: ");
        scanf("%d,%d",&nl,&nc);
    } while( nl < 0 || nl > N || nc < 0 || nc > N );

    for(i=0; i<nl; i++)
    for(j=0; j<nc; j++)
    {   printf("m[ %d ][ %d ]= ",i ,j );
        scanf("%d", &m[ i ][ j ] );
    }
    printf("\nMatricea introdusa este ...");
    for( i = 0; i < nl; i++ )
    {   printf("\n");
        v[ i ] = 0;
        for( j = 0; j < nc; j++ )
        {   v[ i ] += m[ i ][ j ];
            printf(" %5d ", m[ i ][ j ]);
        }
    }
    printf("\nVectorul obtinut este: ");
    for( i = 0; i < nl; i++ )
        printf("\n\t%d", v[ i ] );
}
```

```
Introduceti numarul de linii si coloane
separate prin virgula: 2

Introduceti numarul de linii si coloane
separate prin virgula: 2

Introduceti numarul de linii si coloane
separate prin virgula: 2,3
m[ 0 ][ 0 ]= 1
m[ 0 ][ 1 ]= 2
m[ 0 ][ 2 ]= 3
m[ 1 ][ 0 ]= 1
m[ 1 ][ 1 ]= 2
m[ 1 ][ 2 ]= 3

Matricea introdusa este ...
     1       2       3
     1       2       3
Vectorul obtinut este:
        6
        6
Process ...urned 2 (0...3)  execution ti...
```

Variabile indexate

VARIABILE INDEXATE. TEMA

1. Să se citească un şir de numere întregi a şi să se construiască şirul b care să conţină toate numerele prime din şirul a.

2. Citiţi elementele unui vector şi calculaţi suma elementelor care aparţin intervalului deschis (a,b) cu a şi b valori citite de la tastatură.

3. Scrieţi programul care construieşte, pornind de la un şir de numere întregi, două şiruri: unul cu elementele pare şi celălalt cu elementele impare ale şirului iniţial.

4. Fiind date tablourile unidimensionale a şi b, fiecare avînd n elemente, să se scrie programul care construieşte tabloul c obţinut din tablourile a şi b astfel: primul element din a, al doilea din b, şi aşa mai departe.

5. Determinaţi numărul elementelor negative, nule şi pozitive dintr-o matrice.

6. Dându-se o matrice de numere întregi să se reordoneze liniile matricii în ordinea descrescătoare a elementelor de pe prima coloană.

7. Să se scrie un program care realizează înmulţirea a două matrici.

8. Să se citească o matrice şi să se calculeze suma elementelor aflate în triunghiul superior mărginit de cele 2 diagonale.

Variabile indexate

Siruri de caractere. Declarare

Pentru memorarea şirurilor de caractere, în C, se folosesc tablourile unidimensionale de caractere. Declaraţia, in acest caz, se face conform sintaxei:

char id_sir[*nr_elem*];

Pentru marcarea sfârşitului de şir, după ultimul caracter se adaugă un octet cu valoarea 0. Pentru a sugera că este vorba de un caracter, se foloseşte secvenţa '\0'. In cazul particular al folosirii unui şir vid, primul element din tablou este chiar indicatorul de sfârşit de şir:

```
sir[ 0 ] = 0; /* sau sir[0] = '\0'; */
```

Deci, un tablou de *nr_elem* caractere poate memora şiruri cu lungimi cuprinse între 0 şi nr_elem - 1 caractere, deoarece trebuie să "*încapă*" şi indicatorul de sfârşit de şir '\0' în zona de memorie rezervată tabloului.

Observatie

Acest terminator permite determinarea simplă a sfârşitului şirului.

Exemplu:

```
int n = 0;
char my_str[] = "Un test";
while( my_str[ n++ ]) ;  //while (my_string[n++]);
printf( "Lungimea este: %d\n", n-1 );
```

Variabile indexate

Siruri de caractere. Initializare

O succesiune de caractere încadrată între ghilimele reprezintă o constantă şir de caractere (ghilimelele nu fac parte din şirul de caractere).
La întâlnirea constantei, compilatorul rezervă zona de memorie necesară, o iniţializează cu codurile ASCII ale şirului şi adaugă automat, după acestea, indicatorul de sfârşit de şir '\0'.

Exemplu:

```
printf("ABCDE");
printf("%d",n);
scanf("%f",&t);
```

Constante şir se întâlnesc în apelul funcţiilor printf(...s) sau scanf(...)

Alte forme de initializare a sirurilor de caractere

Exemplu:

```
char s[]="ABCDE";
char *s="ABCDE";
```

In acest caz se alocă pentru tabloul s[ ], 6 octeţi, dintre care primii 5 conţin codurile ASCII ale literelor A,B,C,D,E şi al şaselea indicatorul de sfârşit de şir '\0'.

Exemplu:

```
char s[]= { 'A', 'B',
'C', 'D', 'E', '\0' };
```

Efect identic ...

Variabile indexate

Siruri de caractere. Operare

*/*Metoda 1*/*

In situaţia în care constanta şir depăşeşte lungimea unei linii de terminal, fie în cadrul unei iniţializări, fie în cazul apelului unui printf(), continuarea ei pe linia următoare se face adăugând un '\' la sfârşitul liniei curente.

*/*Metoda 2*/*

O alta varianta consta in divizarea constantei sir initiala (lunga) in doua subsiruri alaturate. In C, doua constante sir alaturate (separate doar prin statii, tab, enter) sunt conacatenate.

Exemplu:

```
/*Constanta originala*/
printf("Acesta este un exemplu cu o constanta sir foarte mare");
/*Metoda 1*/
printf( "Acesta este un exemplu cu o \
constanta sir foarte mare" );
/*Metoda 1*/
printf("Acesta este un exemplu cu o "
       "constanta sir foarte mare"
      );
```

Efecte identice !

Variabile indexate

Siruri de caractere. Observatie

In acest moment, puteţi face distincţia între caracterul 'A' şi şirul "A".
Astfel, caracterul 'A' se reprezintă în memorie pe un octet, pe când şirul "A" se reprezintă pe 2 octeţi.

Siruri de caractere. Functii de citire

Pentru citirea unui şir de caractere se pot folosi funcţiile:

char sir[20];

- ***scanf("%s", sir);*** - cu specificatorul de format %s. Functia va determina citirea caracterelor introduse de la tastatură până se întâlneşte caracterul '\n' (linie nouă), memorand in sir doar pana la primul separator (spatiu, tab, enter);

- ***gets(sir);*** - care va determina citirea caracterelor introduse de la tastatură până se întâlneşte caracterul '\n' (linie nouă). Spre deosebire de scanf, această funcţie permite şi citirea caracterelor spaţiu (blank).

Variabile indexate

Siruri de caractere. Functii de afisare

Pentru afişarea unui şir de caractere se pot folosi funcţiile:

- ***printf("%s", sir);*** - cu specificatorul de format %s. Functia va afisa caracterele din sir, până se întâlneşte caracterul **'\0'** (terminator sir);
- ***puts(sir);*** - determină afişarea şirului pe ecran şi trecerea la linie nouă

Observatie

Deoarece pot ramane caractere in bufferul de tastaura, pentru golirea acestuia se va folosi functia ***flushall()***; `//fflush( stdin );`
Vom folosi acesta functie atunci cand dorim ca apelul functiei gets() sa preia caracterele ce le vom introduce in acel moment si nu caractere ramase de la citiri anterioare.

Exemplu:

```
char sir[ 80 ];
/* .... */
printf("\Introdu sirul: " );
flushall(); //fflush( stdin );
gets( sir );
/* .... */
```

Variabile indexate

Siruri de caractere. Probleme I

. Scrieţi programul care citește un șir de caractere și va număra câte spaţii s-au introdus

```c
#include <stdio.h>
#define N 80

int main ( void ) //In Linux este obligatoriu returnarea unei valori
{
    char s[ N ];
    int i, nr_space = 0;
    puts( "\nIntroduceti sirul de caractere dorit" );
    gets( s );
    for( i = 0; s[i]; i++ )
        if( s[ i ] == ' ' )
            nr_space++;
    printf( "\n S-au introdus %d blank-uri", nr_space );
    return 0;
}
```

```
Introduceti sirul de caractere dorit
Acesta este un test!

 S-au introdus 3 blank-uri
Process returned 0 (0x0)   execution time
                           ntinue
```

Variabile indexate

SIRURI. TEMA

1. Scrieţi un program care citeste un şir de caractere şi afişeaza şirul, astfel încât după fiecare caracter să urmeze câte o virgulă.

2. Scrieţi un program care citeste un şir de caractere şi apoi să afişeze la rând nou câte un subşir care se obţine din precedentul prin îndepărtarea primei litere.

3. Se vor citi:
   - n = numărul de şiruri de caractere ce vor fi citite;
   - respectiv, cele n şiruri de caractere.

Se va calcula suma lungimilor acestor şiruri. Nu se va folosi memorie decât pentru un singur şir.

4. Scrieţi un program care determina dacă un cuvânt dat se găseşte într-o propoziţie dată. Cuvântul şi propoziţia vor fi citite de la tastatură.

5. Să se determine dacă un şir de caractere, reprezentând o propoziţie, este "*negativă sau afirmativă*". Veţi considera că o propoziţie este *negativă* dacă în conţinutul ei apare cuvântul "nu" sau şirul "n-" de un număr impar de ori.

6. Să se citească mai multe şiruri de caractere terminate prin <CR> şi să se afişeze şirul de lungime maximă.

2019/2020 - Suceava

Analiza proprie a referatului trimis!

REFERAT printf() / scanf()

Analizati urmatoarele doua materiale si comparati cu referatele trimise de catre dv.
Ati atins, in referatul trimis, toti acesti specificatori de format?

https://www.cplusplus.com/reference/cstdio/scanf/

https://en.cppreference.com/w/c/io/fscanf