CURS 07 – FP

Tablouri unidimensionale (vectori)

Tablou unidimensional = este o colecție de valori, toate de același tip, având un nume unic și stocate la adrese succesive de memorie

t	10.00	6.50	8.25	6.50	9.80				

Declarare:

```
tip_de_date nume_tablou[număr maxim de elemente];
```

Exemple:

- `int a[100], b[20];`
- `float t[10];`

`dimensiunea fizică`

Accesarea elementelor unui tablou unidimensional:

dimensiunea fizică = 10

`float t[10];`

dimensiunea efectivă = 6

	0	1	2	3	4	5	6	7	8	9	indici/ poziții
t	10.00 ‖ t[0]	6.50 ‖ t[1]	8.25 ‖ t[2]	6.50 ‖ t[3]	9.80 ‖ t[4]	9.25 ‖ t[5]	?	?	?	?	valori

? = valoare reziduală

Tablourile sunt indexate în mod implicit de la poziția 0, iar cel mai mare indice posibil este egal cu *dimensiunea fizică - 1*!

Accesarea elementelor unui tablou se realizează prin indici, respectiv prin expresia *`nume_tablou[indice]`*.

Citirea și scrierea elementelor unui tablou:

```
#include <stdio.h>

int main()
{
    float t[10];
    int i, n;

    //citirea unui tablou:
    //citirea dimensiunii efective (n)
    printf("Numarul de elemente din tablou: ");
    scanf("%d", &n);
    //citirea valorilor celor n elemente
    for(i = 0; i < n; i++)
    {
        printf("t[%d] = ", i);
        scanf("%f", &t[i]);
    }

    //afisarea elementelor unui tablou:
    printf("\nTabloul t:\n");
    for(i = 0; i < n; i++)
        printf("%.2f ", t[i]);
    printf("\n");

    return 0;
}
```

Inițializarea elementelor unui tablou:

```
#include <stdio.h>

int main()
{
    //primele 4 elemente vor fi initializate cu
    //valorile precizate intre acolade, iar restul
    //de 6 elemente vor primi automat valoarea 0
    int t[10] = {10, 20, 30, 40};
    int i, n;

    printf("\nTabloul t:\n");
    for(i = 0; i < 10; i++)
        printf("%d ", t[i]);
    printf("\n");

    return 0;
}
```

Pentru a inițializa toate elementele unui tablou cu 0:

```
int t[10] = {0};
```

Atenție, prin inițializarea

```
int t[10] = {7};
```

doar primul element va fi egal cu 7, restul vor fi automat setate la 0!

Inițializarea elementelor unui tablou declarat fără dimensiune fizică:

```
#include <stdio.h>

int main()
{

    //dimensiunea fizica a tabloului t va fi data de
    //numarul valorilor folosite pentru initializare
    //si NU se mai poate modifica ulterior
    int t[] = {10, 20, 30, 40};

    //numarul efectiv de elemente se determina astfel:
    int n = sizeof(t) / sizeof(t[0]);
    //int n = sizeof(t) / sizeof(int);

    int i;

    printf("Numarul efectiv de elemente: %d\n", n);

    printf("\nTabloul t:\n");
    for(i = 0; i < n; i++)
        printf("%d ", t[i]);
    printf("\n");

    return 0;
}
```

Algoritmi specifici pentru tablouri unidimensionale:

1. calculul unor expresii care implică elementele tabloului

1.1. suma tuturor elementelor tabloului

```
#include <stdio.h>

int main()
{
    int t[100];
    int i, n, suma;

    //citirea tabloului:
    printf("Numarul de elemente din tablou: ");
    scanf("%d", &n);

    for(i = 0; i < n; i++)
    {
        printf("t[%d] = ", i);
        scanf("%d", &t[i]);
    }

    suma = 0;
    for(i = 0; i < n; i++)
        suma = suma + t[i];

    printf("\nSuma elementelor din tablou: %d\n", suma);

    return 0;
}
```

1.2. suma elementelor negative și suma elementelor pozitive

```
#include <stdio.h>

int main()
{
    int t[100];
    int i, n, suma_negative, suma_pozitive;

    //citirea tabloului:
    printf("Numarul de elemente din tablou: ");
    scanf("%d", &n);
```

```
    for(i = 0; i < n; i++)
    {
        printf("t[%d] = ", i);
        scanf("%d", &t[i]);
    }

    suma_negative = suma_pozitive = 0;
    for(i = 0; i < n; i++)
        if(t[i] >= 0)
            suma_pozitive = suma_pozitive + t[i];
        else
            suma_negative = suma_negative + t[i];

    printf("\nSuma elementelor negative: %d\n", suma_negative);
    printf("\nSuma elementelor pozitive: %d\n", suma_pozitive);

    return 0;
}
```

1.3. determinarea minimului și maximului dintr-un tablou

```
#include <stdio.h>

int main()
{
    int t[100];
    int i, n, minim, maxim;

    //citirea tabloului:
    printf("Numarul de elemente din tablou: ");
    scanf("%d", &n);

    for(i = 0; i < n; i++)
    {
        printf("t[%d] = ", i);
        scanf("%d", &t[i]);
    }

    //initializam minimul si maximul curent
    //cu prima valoare din tablou
    minim = maxim = t[0];

    //parcurgem restul tabloului element cu element
    //si actualizam valorile minim si maxim cand este necesar
    for(i = 1; i < n; i++)
        if(t[i] < minim)
            minim = t[i];
```

```
        else
            if(t[i] > maxim)
                maxim = t[i];

    printf("\nMinimul elementelor: %d\n", minim);
    printf("\nMaximul elementelor: %d\n", maxim);

    return 0;
}
```

Variantă (cu inițializări forțate)

```
#include <stdio.h>
//fisierul header limits.h contine valorile minime si maxime
//ale tuturor tipurilor de date primitive
#include <limits.h>

int main()
{
    int t[100];
    int i, n, min_t, max_t;

    //n = dimensiunea efectiva
    printf("Numarul de elemente din tablou (n): ");
    scanf("%d", &n);

    printf("\nElementele tabloului:\n");
    for(i = 0; i < n; i++)
    {
        printf("t[%d] = ", i);
        scanf("%d", &t[i]);
    }

    min_t = INT_MAX;
    max_t = INT_MIN;
    for(i = 0; i < n; i++)
    {
        if(t[i] < min_t)
            min_t = t[i];
        if(t[i] > max_t)
            max_t = t[i];
    }

    printf("\nValoarea minima: %d\n", min_t);
    printf("\nValoarea maxima: %d\n", max_t);

    printf("\nTabloul:\n");
    for(i = 0; i < n; i++)
        printf("%d ", t[i]);

    return 0;
}
```

Variantă:

Să se afișeze cea mai mare valoare negativă (sau cea mai mică pozitivă) dintr-un tablou (dacă există).

```
#include <stdio.h>
//fisierul header limits.h contine valorile minime si maxime
//ale tuturor tipurilor de date primitive
#include <limits.h>

int main()
{
    int t[100];
    int i, n, min_t, max_t;

    //n = dimensiunea efectiva
    printf("Numarul de elemente din tablou (n): ");
    scanf("%d", &n);

    printf("\nElementele tabloului:\n");
    for(i = 0; i < n; i++)
    {
        printf("t[%d] = ", i);
        scanf("%d", &t[i]);
    }

    max_t = INT_MIN;
    for(i = 0; i < n; i++)
        if(t[i] < 0 && t[i] > max_t)
            max_t = t[i];

    if(max_t == INT_MIN)
        printf("\nNu exista valori strict negative in tablou!\n");
    else
        printf("\nValoarea negativa maxima: %d\n", max_t);

    min_t = INT_MAX;
    for(i = 0; i < n; i++)
        if(t[i] > 0 && t[i] < min_t)
            min_t = t[i];

    if(min_t == INT_MAX)
        printf("\nNu exista valori strict pozitive in tablou!\n");
    else
        printf("\nValoarea pozitiva minima: %d\n", min_t);

    printf("\nTabloul:\n");
    for(i = 0; i < n; i++)
        printf("%d ", t[i]);

    return 0;
}
```

Atenție, nici unul dintre programe de mai sus NU necesită utilizarea tablourilor!

1.4. determinarea numărului de apariții ale valorii maxime dintr-un tablou

Varianta 1 (două parcurgeri ale tabloului)

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

int main()
{
    int t[100];
    int i, n, nr_max, max_t;

    //n = dimensiunea efectiva
    printf("Numarul de elemente din tablou (n): ");
    scanf("%d", &n);

    printf("\nElementele tabloului:\n");
    for(i = 0; i < n; i++)
    {
        printf("t[%d] = ", i);
        scanf("%d", &t[i]);
    }

    //calculez maximul din tablou
    max_t = t[0];
    for(i = 1; i < n; i++)
        if(t[i] > max_t)
            max_t = t[i];

    //numar de cate ori apare max_t in tablou
    nr_max = 0;
    for(i = 0; i < n; i++)
        if(t[i] == max_t)
            nr_max++;

    printf("\nMaximul este %d si apare de %d ori in tablou\n", max_t,
            nr_max);

    printf("\nTabloul:\n");
    for(i = 0; i < n; i++)
        printf("%d ", t[i]);

    printf("\n");

    return 0;
}
```

Varianta 2 (o parcurgere a tabloului)

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

int main()
{
    int t[100];
    int i, n, nr_max, max_t;

    //n = dimensiunea efectiva
    printf("Numarul de elemente din tablou (n): ");
    scanf("%d", &n);

    printf("\nElementele tabloului:\n");
    for(i = 0; i < n; i++)
    {
        printf("t[%d] = ", i);
        scanf("%d", &t[i]);
    }

    //calculez simultan maximul din tablou si
    //numarul sau de aparitii
    max_t = t[0];
    nr_max = 1;
    for(i = 1; i < n; i++)
        if(t[i] == max_t)
            nr_max++;
        else
            if(t[i] > max_t)
            {
                nr_max = 1;
                max_t = t[i];
            }

    printf("\nMaximul este %d si apare de %d ori in tablou\n", max_t,
nr_max);

    printf("\nTabloul:\n");
    for(i = 0; i < n; i++)
        printf("%d ", t[i]);

    printf("\n");

    return 0;
}
```

Varianta 3 (fără tablou)

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

int main()
{
    int i, n, x, nr_max, max_t;

    //n = dimensiunea efectiva
    printf("Numarul de valori (n): ");
    scanf("%d", &n);

    //citesc separat prima valoare x
    printf("\nValoare: ");
    scanf("%d", &x);

    max_t = x;
    nr_max = 1;

    //citesc restul de n-1 valori
    for(i = 1; i < n; i++)
    {
        printf("Valoare: ");
        scanf("%d", &x);

        if(x == max_t)
            nr_max++;
        else
            if(x > max_t)
            {
                nr_max = 1;
                max_t = x;
            }
    }

    printf("\nMaximul este %d si apare de %d ori\n", max_t, nr_max);

    printf("\n");

    return 0;
}
```

1.5. determinarea pozițiilor pe care apare valoarea maximă dintr-un tablou

Exemplu: t = (8, -7, 12, 6, 12, 12, -100, 9, 12, 8) => maximul este 12 și apare pe pozițiile 2, 4, 5 și 8.

Varianta 1 (afișare directă):

```
#include <stdio.h>

int main()
{
    int t[100];
    int i, n, maxim;

    //citirea tabloului:
    printf("Numarul de elemente din tablou: ");
    scanf("%d", &n);

    for(i = 0; i < n; i++)
    {
        printf("t[%d] = ", i);
        scanf("%d", &t[i]);
    }

    //calculam maximul elementelor din tablou
    maxim = t[0];
    for(i = 1; i < n; i++)
        if(t[i] > maxim)
            maxim = t[i];

    printf("\nMaximul elementelor: %d\n", maxim);

    //parcurgem inca o data tabloul si afisam pozitiile
    //pe care apare maximul
    printf("\nPozitiile pe care apare maximul:\n");
    for(i = 0; i < n; i++)
        if(t[i] == maxim)
            printf("%d, ", i);

    printf("\b\b \n");

    return 0;
}
```

Varianta 2 (construirea unui tablou auxiliar care va conține pozițiile pe care apare maximul):

Exemplu:

1. t = (8, -7, 12, 6, 12, 12, -100, 9, 12, 8) => maximul este 12
2. determinăm pozițiile maximului și le salvăm într-un tablou având aceeași dimensiune fizică: pozmax = (2, 4, 5, 8)

pozmax	0	1	2	3	4	5	6
	2	4	5	8			
dim_pozmax = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4							

```
#include <stdio.h>

int main()
{
    //pozmax = tablou auxiliar care va contine
    //indecsii elementelor egale cu maximul in tablou
    int t[100], pozmax[100];

    //dim_pozmax = dimensiunea efectiva a
    //tabloului poz_max
    int i, n, maxim, dim_pozmax;

    //citirea tabloului:
    printf("Numarul de elemente din tablou: ");
    scanf("%d", &n);

    for(i = 0; i < n; i++)
    {
        printf("t[%d] = ", i);
        scanf("%d", &t[i]);
    }

    //calculam maximul elementelor din tablou
    maxim = t[0];
    for(i = 1; i < n; i++)
        if(t[i] > maxim)
            maxim = t[i];

    printf("\nMaximul elementelor: %d\n", maxim);
```

```
    //parcurgem inca o data tabloul si
    //construim tabloul pozmax

    //initial, tabloul pozmax este vid
    dim_pozmax = 0;
    for(i = 0; i < n; i++)
        if(t[i] == maxim)
        {
            //adaugam indicele i pe care apare
            //maximul dupa ultimul element din pozmax
            pozmax[dim_pozmax] = i;

            //actualizam numarul de elemente din pozmax
            dim_pozmax++;
        }

    printf("\nIndecsii elementelor maxime:\n");
    for(i = 0; i < dim_pozmax; i++)
        printf("%d, ", pozmax[i]);

    printf("\b\b \n");

    return 0;
}
```

Varianta 3 (construirea unui tablou auxiliar care va conține pozițiile pe care apare maximul folosind o singură parcurgere a tabloului):

t = (4, 3, 4, 7, 4, 7, 5, 3, 7)

maxim = 4
dim_pozmax = 1
pozmax = (0)

maxim = 4
dim_pozmax = 2
pozmax = (0, 2)

maxim = 7
dim_pozmax = 1
pozmax = (3)

maxim = 7
dim_pozmax = 2
pozmax = (3, 5)

maxim = 7
dim_pozmax = 3
pozmax = (3, 5, 8)

```
#include <stdio.h>

int main()
{
    //pozmax = tablou auxiliar care va contine
    //indecsii elementelor egale cu maximul in tablou
    int t[100], pozmax[100];

    //dim_pozmax = dimensiunea efectiva a
    //tabloului poz_max
    int i, n, maxim, dim_pozmax;

    //citirea tabloului:
    printf("Numarul de elemente din tablou: ");
    scanf("%d", &n);

    for(i = 0; i < n; i++)
    {
        printf("t[%d] = ", i);
        scanf("%d", &t[i]);
    }

    //construim direct tabloul pozmax
    //initial, maximul este primul element
    maxim = t[0];
    pozmax[0] = 0;
    dim_pozmax = 1;
    for(i = 1; i < n; i++)
        //daca valoarea curenta este strict mai mare
        //decat maximul curent
        if(t[i] > maxim)
        {
            //actualizam maximul curent
            maxim = t[i];
            //"golim" tabloul pozmax si introducem doar
            //indexul elementului curent (noul maxim curent)
            pozmax[0] = i;
            dim_pozmax = 1;
        }
        else
            //daca valoarea curenta este egala cu maximul curent
            if(t[i] == maxim)
            {
                //adaugam indexul elementului curent in pozmax
                pozmax[dim_pozmax] = i;
```

```
                dim_pozmax++;
            }

    printf("\nMaximul elementelor: %d\n", maxim);

    printf("\nIndecsii elementelor maxime:\n");
    for(i = 0; i < dim_pozmax; i++)
        printf("%d, ", pozmax[i]);

    printf("\b\b \n");

    return 0;
}
```

Varianta 4 (construirea unui tablou auxiliar care va conține pozițiile pe care apare maximul fără a mai păstra valorile introduse):

```
#include <stdio.h>

int main()
{
    //pozmax = tablou auxiliar care va contine
    //indecsii elementelor egale cu maximul in tablou
    int pozmax[100];

    //dim_pozmax = dimensiunea efectiva a
    //tabloului poz_max
    int i, n, maxim, dim_pozmax, x;

    //citirea valorilor
    printf("Numarul de valori: ");
    scanf("%d", &n);

    //citim separat prima valoare pentru a
    //putea initializa corect maximul
    printf("Valoare: ");
    scanf("%d", &x);

    maxim = x;
    pozmax[0] = 0;
    dim_pozmax = 1;

    //construim direct tabloul pozmax,
    //folosind doar valoarea curenta x
    for(i = 1; i < n; i++)
    {
        printf("Valoare: ");
        scanf("%d", &x);
```

```
        if(x > maxim)
        {
            maxim = x;
            pozmax[0] = i;
            dim_pozmax = 1;
        }
        else
            if(x == maxim)
            {
                pozmax[dim_pozmax] = i;
                dim_pozmax++;
            }
    }

    printf("\nMaximul elementelor: %d\n", maxim);

    printf("\nIndecsii elementelor maxime:\n");
    for(i = 0; i < dim_pozmax; i++)
        printf("%d, ", pozmax[i]);

    printf("\b\b \n");

    return 0;
}
```

Observație:

Dacă într-un program parcurgem o singură dată un tablou și prelucrăm doar valoarea elementului curent, atunci tabloul poate fi eliminat și înlocuit cu citiri repetate ale valorii curente!