ALOCARE DINAMICĂ

Nicoi Alexandru - Duță Andrei

08 Ianuarie 2021

1 Alocare, citire și afișare

- ALOCARE: Folosim alocarea dinamică a memoriei pentru a manipula eficient materia. De acum înainte, în toate problemele de programare, veți folosi această metodă de alocare. Pentru a aloca un vector folosind metoda alocării dinamice, avem nevoie de instrucțiunile specifice limbajului C (malloc, calloc și realloc)

```
int n;
scanf("%d",&n);

/// ALOCARE DINAMICA NORMALA CU MALLOC
int *v = (int*)malloc(n * sizeof(int));

/// ALOCARE DINAMICA CU INITIALIZARE ELEMENTE CU 0 FOLOSIND
    CALLOC
int *v_cu_zero = (int*)calloc(n,sizeof(int));

/// REALOCARE PENTRU ADAUGARE INCA UN ELEMENT
n++;
v = (int*)realloc(v,n*sizeof(int));
v[n-1] = 7; /// poate fi orice altceva setat
```

- ACCESARE: Se realizează ca un vector normal.

  Exemplu de accesare element pentru afișare:

```
for(int i = 0 ; i < n ; i++)
 printf("%d ",v[i]);
```

- ELIBERARE MEMORIE: Cand terminăm lucrul pe un vector, eliberăm memoria folosită de acest vector folosind funcția "free". Utilizarea acesteia reprezintă o bună practică în toate problemele pe care le veți rezolva la orice materie de programare, precum și în diferite proiecte din care veți face parte.

```
    free(v);
```

2 Transmiterea ca parametru a tablourilor în subprograme

- TABLOURI UNIDIMENSIONALE (VECTORI): Într-o problemă complexă lucrul în "main()" nu este recomandat. De aceea, vom folosi subprograme (funcții) specifice pentru a avea un control mai bun asupra codului implementat. În cazul limbajului C, transmiterea se face prin intermediul pointerilor, unde definim ca parametru pointer spre vector, iar funcția este apelată prin referința în memorie a vectorului.
  - Program care citește si afișează un vector

```
#include <stdio.h>

void citire(int **v, int *n)
{
    /// int **v <-> int *(*v) = pointer spre vectorul v
    /// int *n = pointer spre intregul n
    /// in orice subprogram de acest tip in care editam
        parametrii
    /// accesam parametrii cu (*p), unde p = numele
        parametrului
    scanf("%d", n); /// &(*n) - echivalente
    (*v) = (int*)malloc((*n) * sizeof(int));
    for(int i = 0 ; i < (*n) ; i++)
        scanf("%d", &(*v)[i]);
}

void afisare(int *v, int n)
{
    /// aici cum nu editam parametrii, ii folosim normal
    for(int i = 0 ; i < n ; i++)
        printf("%d ",v[i]);
}

int main()
{
    int n;
    int *v;
    citire(&v, &n);
    /// verificare date citite
    afisare(v,n);
    return 0;
}
```

- TABLOURI BIDIMENSIONALE (MATRICI): Aici este aproximativ la fel ca la vectori, doar că facem o alocare pentru toată matricea, apoi alocare pe fiecare linie pentru coloane. Prin intermediul alocării dinamice, putem avea matrice cu număr de coloane variabil (adică linia 1 să aibă 4 coloane, linia 2 5 coloane, linia 3 7 coloane etc.)
  - Alocare și citire matrice:

```
int n, m;
scanf("%d %d", &n, &m);
int **a = (int**)malloc(n * sizeof(int*));
///int ** -> DUBLU POINTER PENTRU DUBLA DIMENSIUNE IN MEMORIE
/// n * sizeof(int*) -> n linii <-> n vectori
for(int i = 0 ; i < n ; i++)
{
    a[i] = (int*)malloc(m * sizeof(int));
    /// alocam linia i ca vector de coloane (m coloane)
    for(int j = 0 ; j < m ; j++)
        scanf("%d", &a[i][j]);
}
```

  - Program cu subprograme de citire și afișare:

```
#include <stdio.h>

void citire(int ***a, int *n, int *m)
{
    /// int ***a <-> int *(**a) = pointer spre matricea a
    /// int *n = pointer spre intregul n
    /// int *m = pointer spre intregul m
    /// in orice subprogram de acest tip in care editam
        parametrii
    /// accesam parametrii cu (*p), unde p = numele
        parametrului

    scanf("%d %d", n, m); /// &(*n) , &(*m) - echivalente
    (**a) = (int**)malloc((*n) * sizeof(int*));
    for(int i = 0 ; i < (*n) ; i++)
    {
        (*a)[i] = (int*)malloc((*m) * sizeof(int));

        /// alocam linia i ca vector de coloane (m coloane)

        for(int j = 0 ; j < (*m) ; j++)
            scanf("%d", &(*a)[i][j]);
    }
}
```

```
void afisare(int **a, int n, int m)
{
    /// aici cum nu editam parametrii, ii folosim normal
    for(int i = 0 ; i < n ; i++)
    {
        for(int j = 0 ; j < m ; j++)
            printf("%d ", a[i][j]);
        printf("\n");
    }
}

int main()
{
    int n, m;
    int **a;
    citire(&a,&n,&m);
    afisare(a,n,m);
    system("pause");
    return 0;
}
```