ALOCAREA DINAMICA A MEMORIEI PENTRU STRUCTURI DE DATE (TABLOURI)

4/29/2013

Sa ne reamintim :

- **Elementele unui tablou de memorie pot avea tipuri diferite.**
- **Prin ce se identifica un element al unui tablou de memorie**
- **Cum se realizeaza citirea unui tablou de memorie**
- **Ce tip de variabila poate memora adrese ale unor zone de memorie?**
- **Care sint cele 2 moduri de incarcare a unei variabile pointer?**
- **Care este rolul functiilor :**

  void *malloc(unsigned n);

  void * calloc(unsigned n, unsigned d);

- **Care este rolul functiei void free(void *p);**

A

```
char *str = malloc(30);          // Aloca memorie pentru 30 de caractere
```

```
void  *malloc(unsigned n);
void  * calloc(unsigned n, unsigned d);
```

B

calloc -este specializata pentru memorie organizata ca un vector
malloc - nu tine cont de structura memoriei.
masura de siguranta - inmultirea dintre numarul de elemente si dimensiunea tipului de date poate conduce la overflow

Calloc vs Malloc 1

.2.

Calloc vs Malloc 2

```
98   // Aloca memorie pentru n numere intregi si initializeaza zona cu zero
99
100  int *a= calloc(n, sizeof(int));
101
102
103  int i;
104  int *a = (malloc(n * sizeof(int));
105  for (i = 0; i < n; i++) {
106      a[i] = 0;
107  }
```

1

2

.3.

```cpp
#include <iostream>// vector alocat dinamic
#include <cstdlib>
using namespace std;

int main() {
  int n,i;

  int w[100]; // 100 x 4 octeti
  int *v;
  cout<<"Numar elem vectorul w, n=" ; cin>>n;
  for (i = 1; i <= n; i++)    cin>>*(w+i);  //cin>>w[i];
  for (i = 1; i <= n; i++) cout<<*(w+i)<<' ';  //cout<<w[i]<<' ';

  cout<<"\nNumar elemente vectorul v, n=" ; cin>>n;
  v =(int*) calloc(n, sizeof(int));
  // v = malloc(n * sizeof(int));
  for (i = 1; i <= n; i++)    cin>>*(v+i);  //cin>>v[i];
  for (i = 1; i <= n; i++) cout<<*(v+i)<<' ';  //cout<<v[i]<<' ';
  free(v);                     // Eliberarea memoriei
  return 0;
}
```

```
C:\Users\a\Desktop\alocare-dinamica\bin\Debug\alocare-dinamica.exe
Numar elem vectorul w, n=3
1 2 3
1 2 3
Numar elemente vectorul v, n=3
4 5 6
4 5 6
Process returned 0 (0x0)   execution time : 14.166 s
Press any key to continue.
```

Vector alocat dinamic

.4.

```cpp
#include <iostream> //matrice alocata dinamic
#include <cstdlib>
using namespace std;
int main() {
  int n, i, j;
  int **mat; // Adresa matrice
  cout<<"numar linii/coloane= "; cin>>n;
  mat = (int **)malloc(n * sizeof(int *));  // Alocare memorie ptr matrice

  for (i = 0; i < n; i++) {
    mat[i] =(int *) calloc(n, sizeof(int));  }

  for (i = 0; i < n; i++) // completare matrice
    for (j = 0; j < n; j++)
      *(*(mat+i)+j)=n*i+j+1; //mat[i][j] = n * i + j + 1;

 for (i = 0; i < n; i++) // Afisare matrice
  { for (j = 0;j < n; j++)
       cout<< *(*(mat +i)+j)<<' '; //cout<<mat[i][j]<<' ';
   cout<<endl;}
 return 0;
}
```

```
C:\Users\a\Desktop\alocare-dinamica\bin\Debug\alocare-dinamica.exe
numar linii/coloane= 3
1 2 3
4 5 6
7 8 9

Process returned 0 (0x0)   execution time : 2.062 s
Press any key to continue.
```

Matrice alocata dinamic 1

.5.

```
35  #include <iostream> //matrice alocata dinamic
36  #include <cstdlib>
37  using namespace std;
38
39  void afisare_matrice(int **a, int n, int m) //Functia de afisare a matricei
40  {
41   for (int i = 0; i < n; i++) // Afisare matrice
42    {
43      for (int j = 0;j < m; j++)
44        cout<<a[i][j]<<' ';
45      cout<<endl;}
46  }
47
48 int main() {
49    int n, i, j;
50    int **mat; // Adresa matrice
51    cout<<"numar linii/coloane= "; cin>>n;
52    mat = (int **)malloc(n * sizeof(int *));  // Alocare memorie ptr matrice
53
54    for (i = 0; i < n; i++) {
55      mat[i] =(int *) calloc(n, sizeof(int));  }
56
57    for (i = 0; i < n; i++) // completare matrice
58      for (j = 0; j < n; j++)
59        *(*(mat+i)+j)=n*i+j+1; //mat[i][j] = n * i + j + 1;
60    afisare_matrice(mat,n,n);
61
62   return 0;
63  }
```

```
C:\Users\a\Desktop\alocare-dinamica\bin\Debug\alocare-dinamica.exe
numar linii/coloane= 3
1 2 3
4 5 6
7 8 9

Process returned 0 (0x0)   execution time : 2.062 s
Press any key to continue.
```

Matrice alocata dinamic 2

.6.

```
1
2   #include <iostream>
3   using namespace std;
4   int main()
5   { int m,n,i,j,(*adr)[10];
6   adr=new int[10][10];
7   cout<<"m="; cin>>m;
8   cout<<"n="; cin>>n;
9   for(i=0;i<m;i++)
10      for (j=0;j<n;j++)    cin>>adr[i][j];
11  for(i=0;i<m;i++)
12      { for (j=0;j<n; j++)   cout<<adr[i][j]<<" ";
13      cout<<endl;}
14  }
15
```

Utilizarea operatorului **new**

Matrice alocata dinamic 3

.7.

Utilizarea operatorului

```
24  #include <iostream>
25  using namespace std;
26
27  void* Citire(int n,  int m)
28  { int i,j, (*adr1)[10]=new int[10][10];
29    for(i=0;i<n;i++)
30          for (j= 0;j<m;j++)  cin>>adr1[i] [j] ;
31    return adr1;
32  }
33
34  void Tipar( int n,int m,int(*adr1)[10])
35  { int i,j;
36    for(i=0;i<n;i++)
37      { for (j=0;j<m;j++)
38              cout<<adr1 [i] [j] <<" ";
39        cout<<endl;  }
40  }
41
42  void* Suma( int n, int m,int(*adr1)[10],int(*adr2)[10])
43  { int i,j,(*adr)[10]=new int[10][10];
44    for (i=0;i<n;i++)
45      for (j=0;j<m;j++)
46          adr[i][j]=adr1[i][j]+adr2[i][j];
47    return adr;
48  }
```

```
54  int main()
55  { int m,n,(*adr)[10],(*adr1)[10],(*adr2)[10];
56      cout<<"n="; cin>>n;
57      cout<<"m="; cin>>m;
58      adr1=(int(*)[10]) Citire(n,m);
59      adr2=(int(*)[10]) Citire(n,m);
60      adr=(int(*)[10]) Suma(m,n,adr1,adr2); Tipar(m,
61  }
```

Matrice alocata dinamic 4