



Programozás I.

8.

Mutatók használata:
dinamikus memórafoglalás

Dinamikus memóriafooglalás

Dinamikus memóriafooglalás

- A **malloc** (stdlib.h) függvény segítségével kérhetünk az operációs rendszertől egybefüggő memóriaterületet
- A függvény paramétereként át kell adnunk, hogy mekkora területre van szükségünk
- `int* p = (int*)malloc(sizeof(int));`



Típuskonverzió

Típus méretének lekérése

Dinamikus memóriafooglalás

- Nagyon fontos, hogy a **malloc** függvény segítségével lefoglalt memóriaterületeket mindig felszabadítsuk. Erre a **free** függvény használható
 - `free(p);`

Gyakorló feladatok

• 7.5.7 Feladat

- Készítsen egy koordinátát tároló struktúrát
- Hozzon létre egy 5 elemű, koordináta pointereket tároló tömböt
- Menjen végig a tömb elemein, kérdezzen rá, hogy olvasson-e be koordinátát, és ha igen, foglaljon neki helyet és olvassa be az értékét. Ellenkező esetben a memóriacím értéke nulla.
- Jelenítse meg a tömb koordinátára mutató elemeit
- Szabadítsa fel a lefoglalt memóriaterületet

Dinamikus tömbfoglalás

```
int* array = (int*) malloc( sizeof(int) * size );
```

- Egybefüggő területet foglal le
- Egy *size* méretű int tömb *sizeof(int)*size* mennyiségű memóriát
- Elemek elérése:
 - `array[0]`
 - `array[size-1]`
 - `array[4]==*(array+4)`

Gyakorló feladatok

◦ 7.5.2 Feladat

- Olvasson be egy egész számot billentyűzetről
- Foglaljon helyet annyi egész számnak, amennyi az előzőleg beolvasott szám értéke
- Olvasson be és tároljon el annyi egész számot, amennyi az előzőleg beolvasott szám értéke
- Írassa ki a beolvasott számokat
- Szabadítsa fel a lefoglalt memóriaterületet

Mátrixok

Mátrixok

- Képzeljünk el egy 3×7 méretű mátrixot
- Készítsünk egy tömböt, ami a mátrix sorait tárolja (m)
- Tároljunk az m tömbünkben 7 méretű tömböket
- Így $m[0][3]$ a tömb első sorának negyedik eleme

Statikus mátrixok

- Képzeljünk el egy 3x7 méretű mátrixot
 - `int matrix[3][7];`
 - `matrix[2][5]=42;`
- A memóriában összefüggő területen helyezkedik el

Dinamikus mátrixok

```
int** matrix = (int**)malloc( sizeof(int*) * row );    //sor
```

m

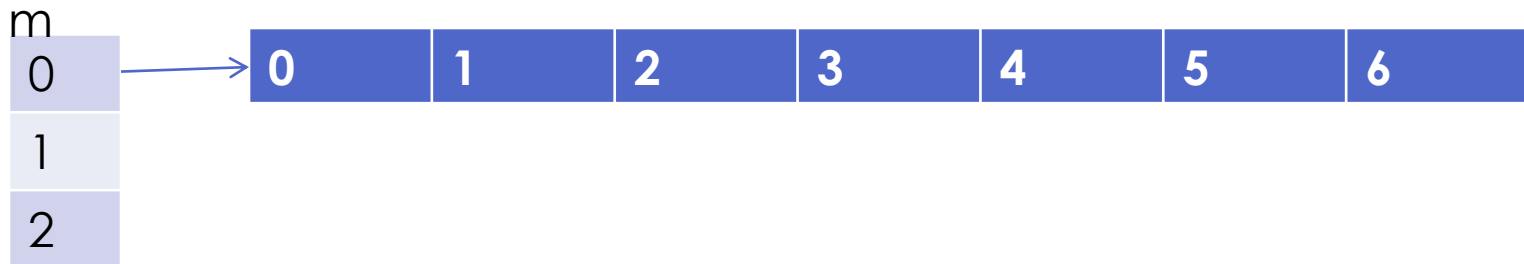
0

1

2

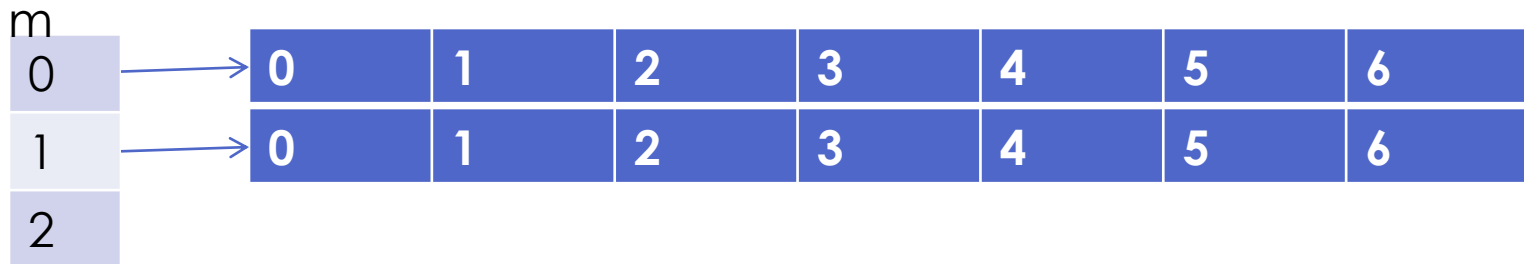
Dinamikus mátrixok

```
int** matrix = (int**)malloc( sizeof(int*) * row );    //sor  
for( i = 0; i < row; i++ )  
{  
    matrix[i] = (int*)malloc( sizeof(int) * col );    //oszlop  
}
```



Dinamikus mátrixok

```
int** matrix = (int**)malloc( sizeof(int*) * row );    //sor  
for( i = 0; i < row; i++ )  
{  
    matrix[i] = (int*)malloc( sizeof(int) * col );    //oszlop  
}
```



Dinamikus mátrixok

```
int** matrix = (int**)malloc( sizeof(int*) * row );    //sor
for( i = 0; i < row; i++ )
{
    matrix[i] = (int*)malloc( sizeof(int) * col );    //oszlop
}
```

m

0	→	0	1	2	3	4	5	6
1	→	0	1	2	3	4	5	6
2	→	0	1	2	3	4	5	6

Dinamikus mátrixok

```
int** matrix = (int**)malloc( sizeof(int*) * row );    //sor
for( i = 0; i < row; i++ )
{
    matrix[i] = (int*)malloc( sizeof(int) * col );    //oszlop
}
```

matrix[2][5] = 42; //3. sor 6. oszlop

m								
0	→	0	1	2	3	4	5	6
1	→	0	1	2	3	4	5	6
2	→	0	1	2	3	4	5: 42	6

Dinamikus mátrixok

```
for( i = 0; i < row; i++ )  
{  
    free(matrix[i]); //oszlop  
}
```

```
free(matrix); //sor
```

m								
0	→	0	1	2	3	4	5	6
1	→	0	1	2	3	4	5	6
2	→	0	1	2	3	4	5: 42	6

Gyakorló feladatok

◦ 7.5.4 Feladat

- Olvasson be két egész számot billentyűzetről
- Foglaljon helyet egy mátrixnak, amely dimenziói az előzőleg beolvasott értékek
- Töltse fel a mátrixot billentyűzetről olvasott értékekkel
- Keresse meg és írja ki a mátrix legnagyobb és legkisebb elemét
- Jelenítse meg a mátrix tartalmát
- Szabadítsa fel a lefoglalt memóriaterületet

Operátorok


- () Precedenciák
 - `*array[i] != (*array)[i]`

Változó méretű tömbök

Változó méretű tömbök

- Mi történik, ha nem tudjuk előre, mennyi számra lesz szükségünk?
- Egy megoldás: foglaljunk nagy helyet, az biztos elég.
- Másik megoldás: foglaljunk kevesebb helyet, és ha később több kell, foglalunk még.

Tömb méretének növelése

regi 

6	12	23	-8	-23	233
---	----	----	----	-----	-----

meret = 6

Tömb méretének növelése

```
int *uj=(int*)malloc(sizeof(int)*(meret+1)); // régi méret + 1
```



Tömb méretének növelése

```
int *uj=(int*)malloc(sizeof(int)*(meret+1)); // régi méret + 1
for( i = 0; i < meret; i++ )
{
    uj[i]=regi[i];      // az elemeket átmásoljuk
}
```

regi →

6	12	23	-8	-23	233
---	----	----	----	-----	-----

meret = 6

uj →

6	12	23	-8	-23	233	
---	----	----	----	-----	-----	--

Tömb méretének növelése

```
int *uj=(int*)malloc(sizeof(int)*(meret+1)); // régi méret + 1
for( i = 0; i < meret; i++ )
{
    uj[i]=regi[i];           // az elemeket átmásoljuk
}
meret++;                    // a méretet növeljük
```

regi →

6	12	23	-8	-23	233
---	----	----	----	-----	-----

meret = 7

uj →

6	12	23	-8	-23	233	
---	----	----	----	-----	-----	--

Tömb méretének növelése

```
int *uj=(int*)malloc(sizeof(int)*(meret+1)); // régi méret + 1
for( i = 0; i < meret; i++ )
{
    uj[i]=regi[i];    // az elemeket átmásoljuk
}
meret++;             // a méretet növeljük
free(regi)           // a régit kitöröljük
```

regi →

meret = 7

uj →

6	12	23	-8	-23	233	
---	----	----	----	-----	-----	--

Tömb méretének növelése

```
int *uj=(int*)malloc(sizeof(int)*(meret+1)); // régi méret + 1
for( i = 0; i < meret; i++ )
{
    uj[i]=regi[i];    // az elemeket átmásoljuk
}
meret++;             // a méretet növeljük
free(regi)           // a régit kitöröljük
regi=uj;             // mutatót átállítjuk
```



Gyakorló feladatok

◉ 7.5.14 Feladat

- ◉ Készítsen egy programot, ami számokat kér be addig, amíg a bekért szám nem nulla
- ◉ A bekért számokat (a nulla kivételével) tárolja el egy tömbben
- ◉ A tömb mérete mindig pontosan akkora legyen, amennyi beolvasott számot tárol
- ◉ A beolvasás után irassa ki a tömb elemeit
- ◉ Keresse meg és közölje a tömb legkisebb elemét