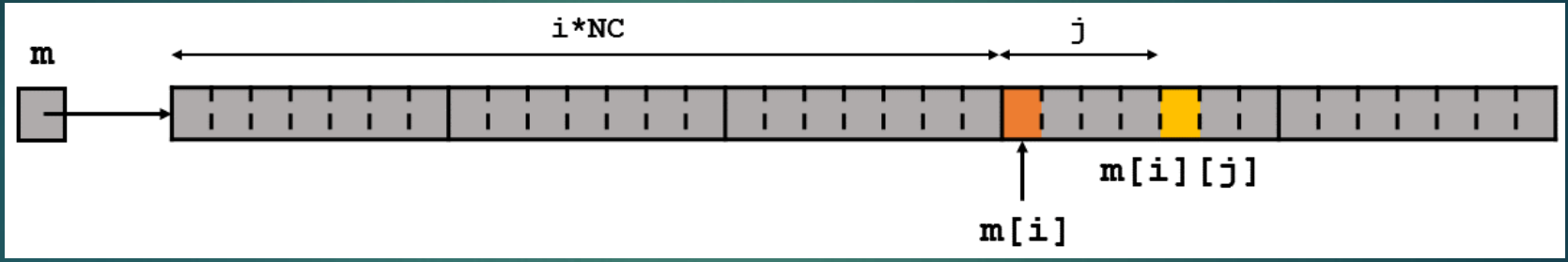
I2010 - Langage C

Chap. 4: Pointeurs et tableaux mutli-dimensionnels

Matrice statique


```
double m[NL][NC];
```

- En mémoire contiguë, ligne par ligne



Matrice statique

Limitations des tableaux statiques :

- la taille doit être connue à la compilation
- si ce n'est pas le cas, il faut fixer une taille maximale

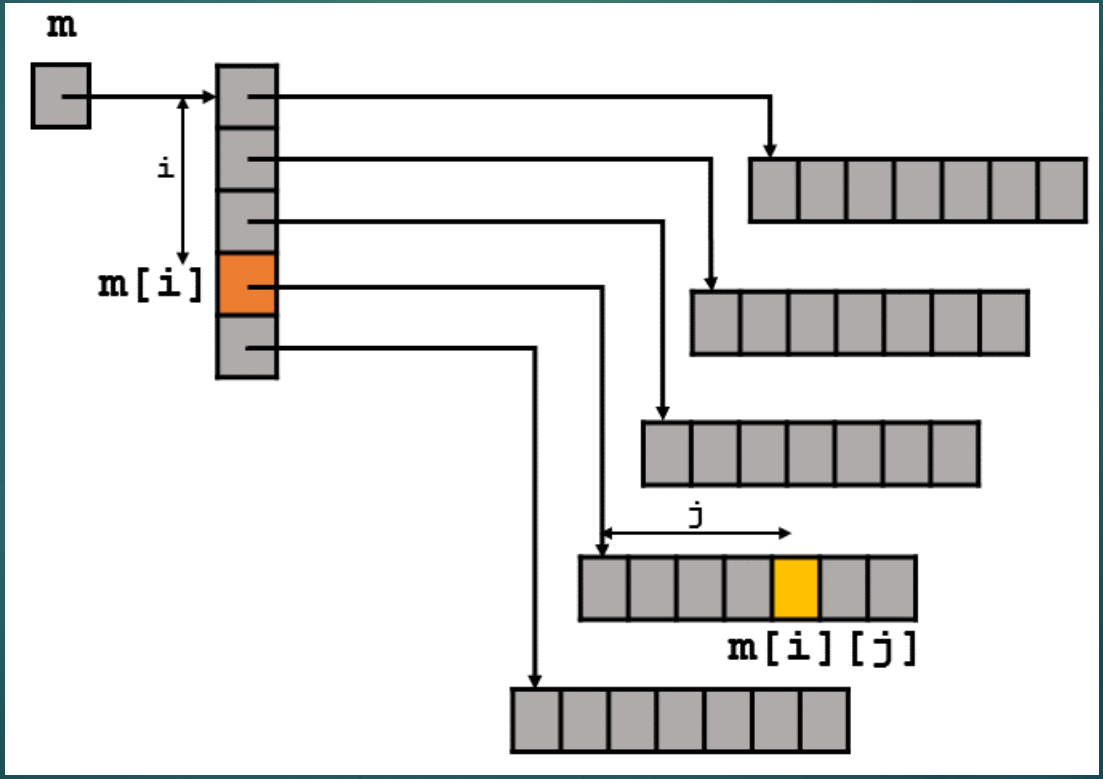
  ⇒ limitation du domaine de validité du programme (borne maximale fixée)

  ⇒ gaspillage de mémoire (espace mémoire alloué mais inutilisé si taille réelle < taille maximale)

Matrice dynamique

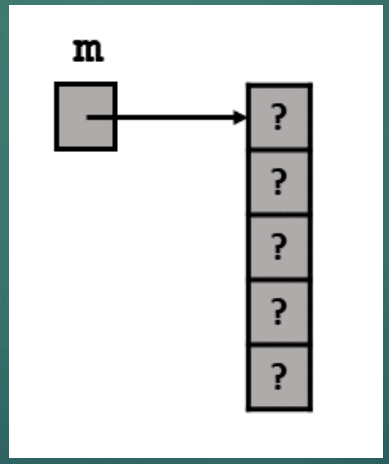

```
double **m;
```

- En mémoire, lignes liées par des pointeurs



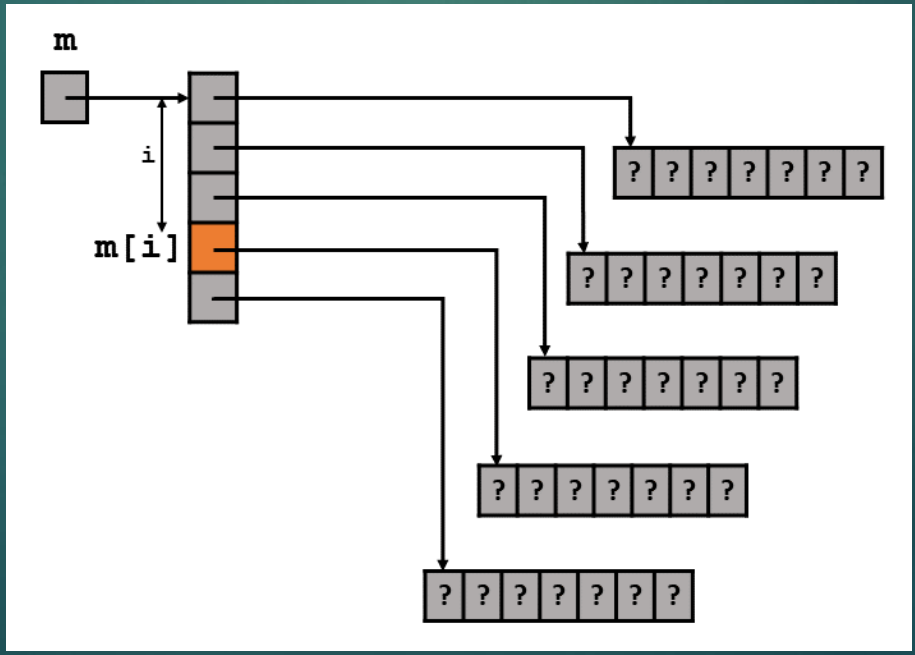
Matrice dyn.: allocation


```
double **m;
m = (double**) malloc(nl * sizeof(double*));
if (m == NULL) exit(1);
```



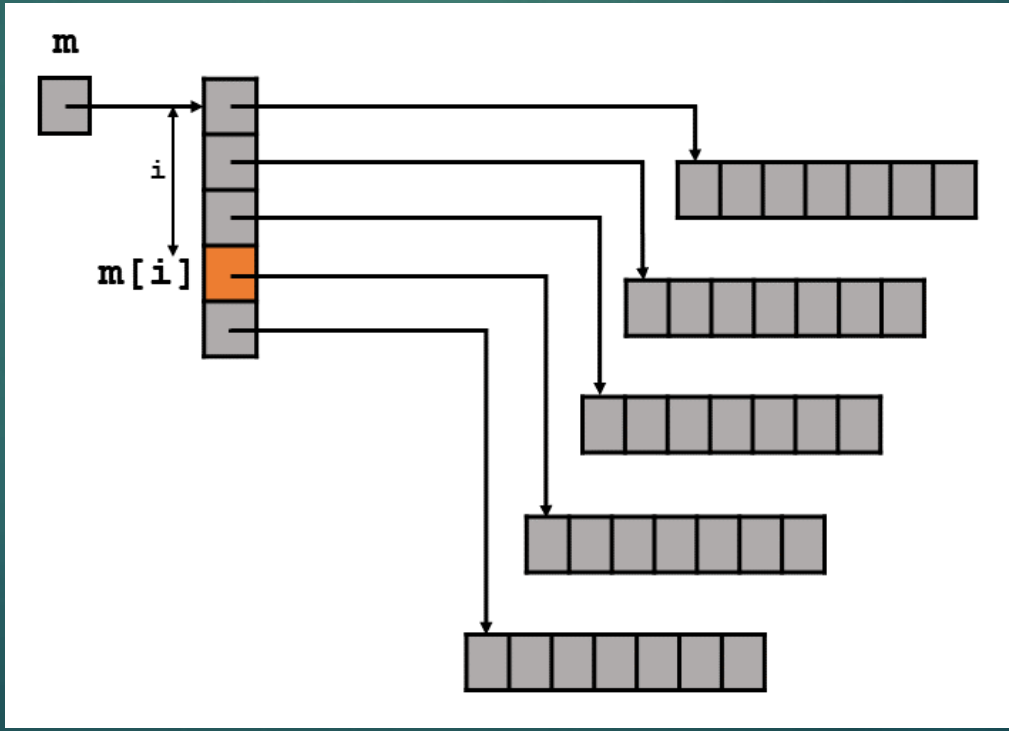
Matrice dyn.: allocation


```
double **m;
m = (double**) malloc(nl * sizeof(double*));
if (m == NULL) exit(1);
for (i = 0; i < nl; i++) {
      m[i] = (double*) malloc(nc * sizeof(double));
      if (m[i] == NULL) exit(1);
}
```



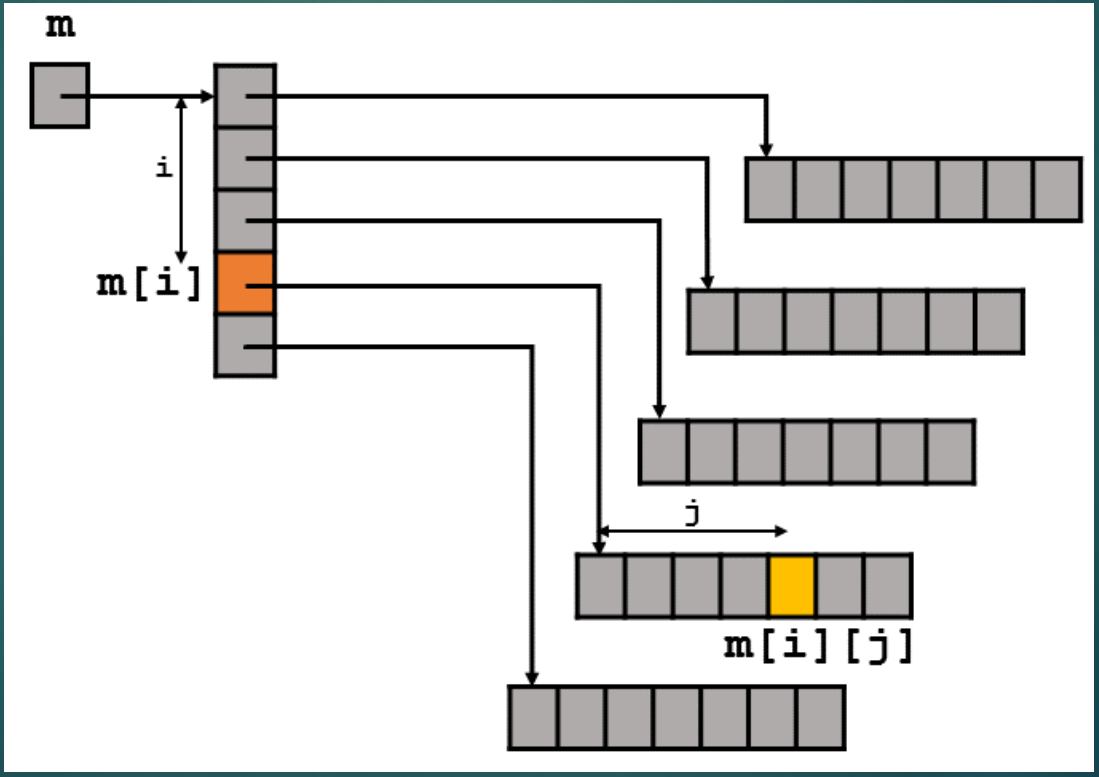
Matrice dynamique: libération mémoire

```
for (i = 0; i < nl; i++)
      free(m[i]);
free(m);
```



Matrice dynamique: déréférencement

`m[i][j]`



Débuggage

- Les fautes classiques liées à la manipulation des pointeurs provoquent des erreurs d'exécution : « **segmentation fault** »

  = tentative d'accès à un emplacement mémoire qui n'est pas alloué au programme

- Difficile d'identifier la cause d'une *segfault*

  ⇒ Utilisation d'un débugger :

  - **gdb** (*GNU Debugger*) pour UNIX / Linux
  - **lldb** (*Low Level Debugger*) pour macOS