# I2010 : langage C (17)

## Modularisation et Classes d'allocation

### 1. Modularisation

Reprenez l'exercice écrit lors de la séance précédente et découpez les sources pour regrouper le code des fonctions qui implémentent la pile dans un fichier `pile.c` et le code qui gère l'évaluation de l'expression en notation polonaise inverse dans un fichier `npi.c`.

Comme expliqué dans le syllabus, un fichier reprenant les définitions de types et les prototypes des fonctions devra être défini ; ce sera `pile.h`.

Ecrivez aussi un makefile

### 2. Fonction statique

Ajoutez une fonctionnalité de réinitialisation dans l'implémentation de la pile.

Pour ce faire définissez une fonction `viderPile` qui devra être une fonction statique dans `pile.c`

La fonction `reinitPile` qui elle sera visible à l'extérieure de `pile.c`, devra faire appel à la fonction statique `viderPile`.

L'interface sera dès lors :

| | | |
|---|---|---|
| `Pile` | `initPile();` | qui renvoie une pile vide |
| `int` | `push(Pile *, int);` | qui met l'entier sur la pile et le renvoie |
| `int` | `pop(Pile *);` | qui retire l'entier du sommet de la pile et le renvoie |
| `int` | `pileVide(Pile);` | qui teste si la pile est vide |
| `Pile` | `reinitPile(Pile *);` | qui vide la pile avant de renvoyer une pile vide |

## 3. *Portée et visibilité d'un identificateur*

Qu'affiche le programme qui suit? Celui-ci est constitué d'un seul fichier :

```
#include <stdio.h>
int   i = 1, next(int), last(int), new(int), reset(void);
main()
{
      int   i, j;
      i = reset();
      for ( j = 1; j <= 3; j++ ) {
            printf("i = %d j = %d\n",i, j);
            printf("next(i) = %d\n", next(i));
            printf("last(i) = %d\n", last(i));
            printf("new(i + j) = %d\n", new(i + j));
      }
}
int next(int j)
{
      return j = i++;
}
int last(int j)
{
      return j = i--;
}
int new(int i)
{
      static int j = 5;
      return i = j += i;
}
int   reset()
{
      return i;
}
```