# Programmation Impérative – TP 4
# Compilation séparée, préprocesseur et fichiers

Guillaume Mercier
mercier@enseirb-matmeca.fr
Augusta Mukam
augusta.mukam@u-bordeaux.fr
Philippe Swartvagher
philippe.swartvagher@enseirb-matmeca.fr

2023 – 2024

Pour les exercices demandant d'écrire une fonction, on écrira également une fonction `main()`, pour s'assurer que le code écrit compile et que la fonction a le comportement attendu.

## 1 Code de César à partir d'un unique fichier source

Dans le TP 2, les fichiers sources `code_cesar.c` et `decode_cesar.c` ne différent que de quelques lignes.

Utiliser les mécanismes que propose le préprocesseur pour n'avoir qu'un seul fichier source qui peut donner à la fois le programme `code_cesar` et `decode_cesar` suivant comme on le compile :

```
cc -DCODE code_cesar.c -o code_cesar
cc -DDECODE code_cesar.c -o decode_cesar
```

## 2 Macro de préprocesseur `SWAP`

Écrire une macro de préprocesseur `SWAP` qui échange les valeurs de deux variables de même type.

Exemple d'utilisation :

```
int a = 5, b = 8;

printf("a = %d ; b = %d\n", a, b); // a = 5 ; b = 8
SWAP(int, a, b);
printf("a = %d ; b = %d\n", a, b); // a = 8 ; b = 5
```

```
SWAP(int, a, b);
printf("a = %d ; b = %d\n", a, b); // a = 5 ; b = 8
```

Pourquoi la macro a besoin du type des variables comme paramètre ?

# 3 Jouons avec la compilation séparée

Reprenez le code que vous avez écrit dans l'exercice sur la représentation des dates dans le TP précédent. On va déplacer tout le code relatif à `struct date_s` (structure, fonctions qui manipulent cette structure, ...) dans un fichier `date.c`. Gardez le reste du code (la fonction `main()` surtout) dans un fichier `date_test.c`.

**Question 1** Quel(s) changement(s) est-il nécessaire d'apporter au code maintenant reparti dans deux fichiers ? A-t-on besoin d'un fichier supplémentaire ? Comment compile-t-on `date_test.c` ?

**Question 2** Pour abstraire la façon dont est défini `struct date_s`, on veut maintenant masquer ses membres, pour que seul le code dans le fichier `date.c` puisse connaître les membres de la structure. Comment faire ? Que faut-il changer ? A-t-on besoin d'introduire une fonction supplémentaire ?

**Question 3** Compilez de façon séparée les différents fichiers source :

```
cc -c date_test.c -o date_test.o
cc -c date.c -o date.o
cc date_test.o date.o -o date_test
```

Affichez la table des symboles des fichiers objets. Qu'observez-vous ?

```
nm date_test.o
nm date.o
```

**Question 4** Modifiez la fonction `date_is_bissextile()` pour qu'elle appelle une fonction `year_is_bissextile()` que vous définirez juste au-dessus de `date_is_bissextile()`. Recompilez `date.c` seul en fichier objet et affichez sa table des symboles. Qu'observez-vous ?

**Question 5** Quelles commandes `cc` a-t-on besoin de réexécuter pour que le programme `date_test` utilise la nouvelle version de `date.c` ?

**Question 6** Appelez `year_is_bissextile()` dans `date_test.c` sans ajouter son prototype à `date.h`. Recompilez ce qu'il est nécessaire de compiler (sans `-Werror` cette fois). Qu'observez-vous ?

**Question 7** Transformez la fonction `year_is_bissextile()` en une fonction statique. Recompilez ce qu'il est nécessaire de compiler. Qu'observez-vous ?

**Question 8** Enlevez l'appel `year_is_bissextile()` dans `date_test.c` et transformez la fonction `year_is_bissextile()` en une fonction `inline`. Recompilez ce qu'il est nécessaire de compiler. Qu'observez-vous ?

**Question 9** On souhaite changer la façon dont la date est stockée au sein de la structure : au lieu d'avoir trois champs de type `int`, on souhaite maintenant avoir un tableau de trois `int`. Dupliquez `date.c` en `date_tab.c` et appliquez les changements nécessaires.

- Que peut-on utiliser au lieu des valeurs numériques `0`, `1` et `2` pour accéder aux cases du tableau ?
- Faut-il aussi modifier d'autres fichiers ?
- Que faut-il recompiler pour avoir le programme `date_test` qui utilise cette seconde version de l'interface ?
- Aurait-on pu faire ce changement de façon aussi transparente pour les codes qui utilisent l'interface avec la version qui rendait publique la définition de la structure (avant la question 2) ?

# 4 Afficher un fichier en numérotant ses lignes

Écrire un programme qui affiche le contenu du fichier texte passé en paramètre en préfixant chaque ligne par le numéro de ligne. Les lignes du fichier à afficher peuvent avoir n'importe quelle longueur.

On affichera les messages d'erreur à l'aide de la fonction `perror`.

Exemple d'exécution :

```
./cat_numbered_lines cat_numbered_lines.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char* argv[])
5 {
[...]
```

**Bonus** Aligner les numéros de ligne sur la droite, pour obtenir un affichage comme suit :

```
./cat_numbered_lines cat_numbered_lines.c
[...]
 8     {
 9         printf("Usage : %s <fichier>\n", argv[0]);
10         return EXIT_FAILURE;
11     }
[...]
```

# 5 Code de César sur un fichier

Reprendre l'exercice du TP précédent sur le code de César, mais cette fois-ci l'appliquer directement à un fichier.

Les fichiers à chiffrer ou déchiffrer pourront être passés au programme de deux manières différentes :

- comme paramètre du programme ;
- sur l'entrée standard, en faisant une redirection lorsque le programme est lancé.

Exemple d'exécution :

```
./code_cesar 3 code_cesar.c > code1
./code_cesar 3 < code_cesar.c > code2
diff code1 code2  # ne doit rien afficher
./decode 3 code1 > decode1
./decode 3 < code2 > decode2
diff decode1 code_cesar.c  # ne doit rien afficher
diff decode2 code_cesar.c  # ne doit rien afficher
```

Que se passe-t-il si on exécute la ligne suivante ?

```
./code_cesar 3
```

# 6 Compilation séparée et manipulation de matrices

Reprenez dans le TP précédent les deux implémentations qui stockent les matrices de deux manières différences (tableau de tableaux ou unique tableau).

**Question 1** Dans le même esprit que l'exercice 1, utilisez les mécanismes que propose le préprocesseur pour pouvoir choisir à la compilation quelle méthode de stockage va être utilisée.

**Question 2** Dans le même esprit que l'exercice 2, déplacez tout le code relatif à l'interface de manipulation des matrices dans un fichier séparé.