Licence Informatique — **Initiation à la Programmation**
Université Bordeaux I — **Travaux Dirigés**

Calcul

- Mettre en œuvre des boucles et des appels de fonctions.
- Utiliser les fonctionnalités de base du débogueur GDB.
- Vérifier la correction d'un calcul à l'aide de constantes.
- Respecter des normes de présentation.
- Écrire un programme principal : utiliser les arguments du main et définir une fonction d'usage.

Un compte-rendu d'une page sur l'exercice 2 vous sera demandé la prochaine séance, on pourra utiliser la commande `a2ps -4 --file-align=fill [les fichiers] -o toto.ps` pour obtenir l'impression de 4 pages par feuille.

PRÉPARATION

1. sortir le necéssaire pour prendre des notes ;
2. se connecter, en cas de problème contacter très rapidement votre enseignant ;
3. recopier à votre racine le répertoire `~wacren/IP/2_calcul` ainsi que son contenu (commande `cp -r ~wacren/IP/2_calcul ~/IP`) ;
4. visiter succintement votre répertoire `~/IP/2_calcul` à l'aide des commandes `cd`, `ls` et `cat`.

Exercice 1.

TRADUCTION D'ALGORITHME

1. Compléter, à partir de l'algorithme itératif vu en ASD, le fichier `fibonacci.c` du répertoire `~/IP/2_calcul/fibonacci/iteratif` ;
2. compiler (depuis emacs) puis tester cette fonction ;
3. déterminer expérimentalement le rang limite de correction de cette fonction ;
4. quitter emacs afin d'éviter toutes confusions ;
5. compléter, à partir de l'algorithme récursif vu en ASD, le fichier `fibonacci.c` du répertoire `~/IP/2_calcul/fibonacci/recursif` ;
6. compiler via emacs ce programme puis tester le programme `fibonacci` ;

UTILISATION D'UN DÉBOGUEUR

7. quitter emacs, puis le relancer ;
8. s'assurer de la présence de l'option `-g` dans la variable `CFLAGS` du fichier `makefile` ;
9. lancer le débogueur GDB sous emacs par la commande `M-x gdb` et avec, pour nom d'exécutable, `~/IP/2_calcul/fibonacci/iteratif/fibonacci` ;
10. lister le programme sous GDB en tapant `l 1` puis une série de fois sur la touche entrée ;
11. désigner un *point d'arrêt* à une ligne de la boucle principale de la fonction fibonacci en tapant `b` *numéro-de-ligne* ;
12. lancer le programme via la commande `run 10000` ;
13. taper `display` *nom-de-variable* pour visualiser la valeur de chaque variable intervenant dans la boucle ;
14. taper `c` pour continuer l'exécution du programme, `s` pour avancer d'un pas (*step*) et ignore <numero de breakpoint> <n> pour avancer de `n` tours de boucle ;
15. poursuivre ainsi l'exécution jusqu'à la détection de l'anomalie puis quitter GDB par la commande `C-x k`) ;

MISE EN PLACE D'ASSERTION

16. utiliser la fonction `assert` du module `assert.h` et une constante afin de prémunir l'utilisateur de toute anomalie ;
17. expliquer pourquoi vous avez choisi la constante `ULONG_MAX` de `limits.h` plutôt que le rang limite déterminé plus haut ;
18. tester le programme ;

INTÉGRATION D'UNE FONCTION

19. intégrer la fonction `fibonacci` au module `combinatoire` présent dans `~/IP/2_calcul/calculer/` (on respectera la présentation du module) ;
20. modifier en conséquence le fichier `~/IP/2_calcul/calculer/calculer.c`, compiler et tester le programme calcul.

Exercice 2. En vous inspirant de la fonction `calculer`, écrire une fonction qui mesure une quantité sur une liste d'entiers passée en paramètre, par exemple :

```
$ mesurer somme 1 2 2 3
8
$ mesurer max -1 2 8 9 4
9
$ mesurer doublon 1 3 5 4 2 1 5 5
4             # il y a 4 paires de doublons à base de 1 et 5
$ mesurer répétition 1 3 5 4 2 1 5 5
2             # il y a 2 éléments qui se répétent (1 et 5)
$ mesurer max 1 2 4 y 1
usage : mesurer option <liste d'entiers>
options disponibles : somme max doublon répétition
```

Programmer les options somme, max, doublon et répétition.