INF203 - Travaux pratiques, séance 5
C : types, tests, boucles, parcours de tableaux

Testons des entiers

`[TP5]` Compilez le programme `exemple_generation.c` à l'aide de la commande :

```
clang exemple_generation.c
```

Puis exécutez quelques fois le programme avec la commande :

```
./a.out
```

• Modifiez le programme afin qu'il affiche `Trop grand` ou `Trop petit` ou `Youpi` selon que l'entier généré est supérieur, inférieur, ou égal à `42`. Compilez à nouveau et ré-exécutez le programme plusieurs fois.
Dans la suite, il ne vous sera pas rappelé de (compiler et) tester vos programmes, ce sera considéré comme évident.

• Lorsque ceci est au point, vous allez utiliser ce que vous avez fait pour tester non pas un seul entier généré, mais plusieurs à la suite. Modifiez le programme pour :
- choisir une valeur entière strictement supérieure à 1 ;
- stocker cette valeur dans une variable n ;
- répéter n fois à l'aide d'une boucle la génération et le test de l'entier généré ;
- réessayer en changeant la valeur de n.

Indentez votre programme pour le rendre lisible : ajoutez, si vous ne l'avez pas fait, un caractère de tabulation ou un nombre fixe d'espaces au début de chaque ligne à l'intérieur de la boucle `for`. Si vous avez des doutes sur ce qui est attendu, demandez à votre enseignant.
[a] Joignez le listing (le code source) de votre programme à votre compte-rendu. ■

Provoquons un débordement

1. Lisez le contenu du fichier `deborde_char.c`, et prévoyez ce qui va se passer lors de l'exécution.
   [b] Quelle est la taille en octets et en bits d'une variable de type `unsigned char` ? Quel est le plus grand entier représentable avec le type `unsigned char` ? ■

2. Créez une copie de `deborde_char.c` sous le nom `deborde_short.c`. Modifiez ce fichier afin qu'il serve à tester des variables de type `unsigned short int` (au lieu de `unsigned char`).
   [c] Répondez aux mêmes questions que ci-dessus, pour le type `unsigned short int`. ■

3. Effectuez la même expérience pour le type `unsigned int` (donnez à l'exécutable le nom `deborde_int`).
   [d] Mêmes questions pour le type `unsigned int`. ■

La commande `time` permet de mesurer le temps d'exécution d'une commande. Renommez en `deborde_int` votre programme `a.out` obtenu avec `deborde_int.c`, puis exécutez :

```
time ./deborde_int
```

Trois lignes s'affichent : notez celle qui commence par "`user`", elle indique le temps d'exécution de votre programme, en minutes et secondes.
[e] En arrondissant à la seconde, quel est le temps d'exécution de `deborde_int` ? Et pour `deborde_short` ? Qu'en pensez-vous ? ■
[f] Et avec un `unsigned long long`, que cela donnerait-il ? Pensez-vous devoir essayer pour avoir la réponse ? ■

Nous reviendrons en cours et en TD sur cette expérience : retenez les résultats que vous avez obtenus lors de cette expérience de débordements.

Rangeons !

Compilez le programme `exemple_generation_tableau.c` puis exécutez le plusieurs fois. Lisez ensuite le texte du programme, et vérifiez que vous comprenez. Si ce n'est pas le cas, demandez à votre enseignant.

Affichages successifs

Modifiez le programme pour qu'il affiche le tableau non pas à la fin, mais à chaque ajout d'un nouvel élément dans le tableau.

Tri du tableau à la volée

• Ajoutez à votre programme (avant la fonction `main`) la fonction suivante, qui échange dans le tableau `Tab` passé en paramètre, les valeurs des éléments d'indices `i` et `j`, passés eux aussi en paramètre :

```
void echanger(long Tab[], int i, int j) {
        long tmp;
        tmp = Tab[i];
        Tab[i] = Tab[j];
        Tab[j] = tmp;
       }
```

En-dessous de la fonction `echanger`, écrivez la fonction `inserer` qui met à sa place l'élément `val` dans la séquence triée (par ordre croissant) `Tab[0..nb-1]`, selon l'algorithme suivant :
— placer l'élément `val` en position `nb` dans `T` ;
— échanger `val` avec son voisin de gauche tant qu'il n'est pas à sa place.
La fonction `inserer` utilisera la fonction `echanger` que vous venez de recopier.

```
/* inserer a sa place l'entier val dans la sequence triee Tab[0..nb-1] */
void inserer(long Tab[], int nb, int val) {
        /* A COMPLETER */
        }
```

• Remplacez l'instruction `T[i] = valeur` de la fonction `main` par un appel à la fonction `inserer`.

[g] Avec quels paramètres *effectifs* avez-vous appelé la fonction `inserer` ?
Lorsque le programme est au point, joignez-le à votre compte-rendu. ■

Les commandes de la semaine : head et tail

Rappel : `head [-n nbl | -c nbc]` affiche le début de ses entrées, selon l'option choisie :
— les `nbl` premières lignes (option par défaut, `nbl`=10)
— ou les `nbc` premiers caractères.
De même la commande `tail` affiche la fin de ses entrées.
[h] Comparez le comportement des commandes `tail -n 4` et `tail -n +4` sur un même fichier. ■

[i] Écrivez un script qui prend en argument le nom d'un fichier, et affiche les lignes de ce fichier 4 par 4 en séparant les blocs de 4 lignes par une ligne de pointillés. Joignez son listing à votre compte-rendu. ■