Travaux Pratiques thème 1 : Boucles/structures de contrôle

Exercice 1 - Se préparer au TP1

1. Dans votre répertoire d'accueil créer un répertoire `Informatique`.
2. Aller dans ce répertoire
3. Créer un sous-répertoire `TP1`.
4. Aller dans ce sous-répertoire.
5. Créer le fichier `hello.c` avec l'éditeur de texte
   `gedit hello.c &` (n'oubliez pas le `&`).
   ou (mieux)
   `emacs hello.c &`

Exercice 2

1. Ecrire dans le fichier `hello.c` grâce à un éditeur de texte :

```
#include <stdio.h>

int main()
{
return(0);
}
```

2. Ecrire dans le `main` une petite instruction permettant d'afficher `Hello World !`
3. Ajouter des commentaires pertinents à ce programme et vérifier l'indentation.
4. Compiler ce programme avec gcc (dans un terminal)

```
gcc -Wall hello.c -o hello
```

5. Lancer l'exécutable (dans un terminal)

```
./hello
```

Exercice 3

Ecrire un programme dans lequel on définit une variable de type entier `n` que l'on initialise à 2 et qu'on l'affiche.

Exercice 4

Même chose que l'exercice précédent mais c'est à l'utilisateur du programme d'initialiser l'entier à la valeur de son choix.

Exercice 5

Ecrire un programme qui demande à l'utilisateur d'initialiser deux entiers. Puis le programme stocke la somme de ces deux entiers dans une autre variable et enfin affiche le résultat.

Exercice 6 PGCD

Ecrire un progamme qui lit deux entiers, vérifie qu'ils sont bien strictement positifs et affiche leur `PGCD`.

Exercice 7 Entier inversé

Écrire un programme qui demande à l'utilisateur de saisir un entier strictement positif, puis affiche cet entier dans le sens inverse. Par exemple, si l'utilisateur saisit 5628, le programme affiche 8265.

Exercice 8 FizzBuzz

Écrire en C un programme qui affiche les nombres de 1 à 100, sauf :
— si le nombre est un multiple de 3 ou se termine par 3, auquel cas il affiche `"Fizz"`
— si le nombre est un multiple de 7 ou se termine par 7, auquel cas il affiche `"Buzz"`
— si le nombre vérifie les 2 propositions, le programme affiche `"FizzBuzz"`
Le tester, et vérifier que l'on obtient bien (pour les 1ères valeurs) `"1 2 Fizz 4 5 Fizz Buzz 8 Fizz 10 11 Fizz Fizz Buzz Fizz 16 Buzz Fizz ...."`

Exercice 9 Retour des fractions egyptiennes

Codez en C la résolution de l'exercice sur les fractions égyptiennes. Dans le cas où votre programme fait des calculs avec des `float`, qu'observez-vous ? Pourquoi ?

Exercice 10 Suite de Fibonacci

La suite de Fibonacci est définie de manière récursive par la relation : $u_n = u_{n-1} + u_{n-2}$.
Cette définition doit être complétée par une condition initiale. Dans notre cas, si n est égal à 0 ou 1 alors : $u_0 = u_1 = 1$.
Ecrire le programme qui calcule le n-ième terme d'une suite de Fibonacci (La valeur de est entrée au clavier).

Exercice 11

Écrire le programme C du jeu qui fabrique un nombre au hasard entre 1 et 100, et le fait ensuite deviner à l'utilisateur. Celui-ci propose une valeur et l'ordinateur répond trop petit ou trop grand jusqu'à ce qu'il trouve la bonne réponse.
Modifiez ensuite le programme pour qu'il affiche le nombre de coups.

Aide : utiliser la fonction `rand()` qui génére un nombre entier aléatoire compris entre `0` et `RAND_MAX`. Pour utiliser cette fonction il faut ajouter `#include <stdlib.h>` au début de votre fichier.

Exercice 12 Traitement d'un entier

Ecrire programme qui affiche le nombre de chiffres 1 contenu dans un entier entré au clavier. Par exemple, pour $n = 1151$, le programme affichera 3.

Exercice 13 Nombres premiers

Nous souhaitons afficher les nombres premiers inférieurs ou égaux à `MAX`. On arrêtera le traitement d'un nombre dès qu'on est sûr qu'il n'est pas premier (i.e. dès que nous avons trouvé un diviseur différent de 1 et de lui même). Pour décider si un nombre `nb` est premier, nous allons déclarer une variable entière `premier` initialisée à 1 que nous utiliserons comme un booléen (1 signifie vrai et 0 signifie faux). Nous allons parcourir tous les nombres à partir de 2 jusqu'à ce que l'on trouve parmi ces nombres un diviseur de `nb` ou que l'on atteigne la valeur `nb/2`. Dans le premier cas, la variable `premier` est mise à 0, dans le deuxième cas, `nb` est premier et le parcours se termine avec la valeur 1 pour premier.

1. Écrivez un programme qui permet de déterminer si un entier, tiré aléatoirement dans l'intervalle $[0, 99]$ est premier. Votre programme affichera à l'écran "`nb` est premier" ou "`nb` n'est pas premier" selon la valeur de la variable premier à l'issue de la boucle (et en remplaçant `nb` par sa valeur !).
2. Modifiez votre programme pour qu'il affiche la liste de tous les nombres premiers inférieurs ou égaux à `MAX`. La valeur de `MAX` sera définie en utilisant la directive #**define**.
3. Modifiez le programme pour qu'il demande à l'utilisateur de saisir une valeur entière m. Le programme affiche ensuite tous les entiers de 2 à m en les séparant par des tabulations ('\t') et en passant à la ligne après chaque nombre premier. Par exemple, pour $m = 25$, on doit obtenir :

```
Saisie d'un entier
25
2
3
4 5
6 7
8 9 10 11
12 13
14 15 16 17
18 19
20 21 22 23
24 25
```

Exercice 14 Suite récurrente d'ordre 2

Écrivez un programme qui calcule le terme de rang n de la suite $U_n = a.U_{n-1} + b.U_{n-2}$ avec $U_0 = 1, U_1 = 2, a = 5$ et $b = 10$.

Exercice 15 Remplissage d'une salle

Nous souhaitons écrire un programme qui permet de placer des spectateurs dans une salle contenant NB_RANG rangées et NB_PLACES places par rangée. Les spectateurs se présentent par groupes d'au plus MAX_GROUPE personnes. Le nombre de personnes composant

un groupe est choisi aléatoirement. Toutes les personnes du groupe doivent être placées avant de passer au groupe suivant. Les rangées sont remplies les unes à la suite des autres. Les groupes de spectateurs sont acceptés tant qu'il y a de la place. Ecrivez le programme qui permet de suivre l'évolution du remplissage de la salle. Voici un exemple d'exécution dans lequel NB_RANG=5, NB_PLACES=5 et MAX_GROUPE=12 :

```
Il y a 25 places disponibles
7 personne(s) a placer
5 personne(s) placee(s) dans la rangee 1
2 personne(s) placee(s) dans la rangee 2
Il reste 18 place(s)
8 personne(s) a placer
3 personne(s) placee(s) dans la rangee 2
5 personne(s) placee(s) dans la rangee 3
Il reste 10 place(s)
2 personne(s) a placer
2 personne(s) placee(s) dans la rangee 4
Il reste 8 place(s)
3 personne(s) a placer
3 personne(s) placee(s) dans la rangee 4
Il reste 5 place(s)
```