Université de Franche Comté Licence 2

TP d'algorithmique des données n° 1

Initiation au C

Ce TP doit être fini avant la prochaine séance de TP, éventuellement sur votre temps de travail personnel.

Vous devez rendre ce TP avant 23h30 la veille de votre prochain TP. Pour cela, vous devez déposer une archive .zip contenant 3 fichiers sur Moodle.

- **un fichier source pour le dernier programme de l'exercice 4.**
- **un fichier source pour chacun des 2 programmes demandés dans l'exercice 5.**

Vos fichiers doivent être commentés et les commentaires doivent être en français.

Le Langage C est un langage compilé, de même que Java. Cependant, le langage C ne s'appuie pas sur l'utilisation d'une machine virtuelle et **un exécutable C ne sera fonctionnel que sur la machine où il a été compilé**.

Exercice 1 Compilation - Exécution

1. Connectez-vous sous linux.
2. Créez un répertoire pour vos TPs d'algorithmique. Pour cela utiliser la commande `mkdir TPalgo`.
3. Placez vous dans ce répertoire par la commande `cd TP1`. Créez un répertoire pour le TP1 avec `mkdir TP1`. Placez vous dans TP1 par la commande `cd TP1`
4. Télécharger le fichier `tp1.c` dans le répertoire TP1. Ouvrez-le avec le logiciel emacs par la commande `emacs tp1.c &` Vous devez lire :

```
#include<stdio.h>
#include<stdlib.h>

int main(){
    printf("Hello\n");
}
```

5. Compiler en tapant dans le terminal la commande
   `gcc tp1.c -o tp1`
6. Exécuter avec la commande `./tp1`

Explications : Les deux premières lignes de ce programme sont des inclusions de bibliothèques,

- La première `stdio.h` contient les fonctions classiques gérant les entrées/sorties, c'est-à-dire permettant notamment de lire ce que tape l'utilisateur ainsi que d'afficher.
- La seconde `stdlib.h` (non utilisée dans le code ci-dessus) contient des fonctions usuelles.

Ensuite le programme se découpe en plusieurs fonctions dont une seule s'appelle `main()` et qui est celle qui est exécutée par le programme (comme en Java). Ici il n'y a qu'une instruction `printf()` qui permet d'afficher dans le terminal.

Comme en Java, **toutes les instructions se terminent par un ;**

Exercice 2 Variables, boucles, conditions...
Tout se passe comme en Java, compilez et exécutez le code suivant.

```
#include<stdio.h>
#include<stdlib.h>

int main(){
    int i;
    for(i=0;i<5;i++){
       if( i != 5){
          printf("Hello\n");
       }
    }
}
```

On dispose aussi des boucles `while` et `do`. Les types de bases sont `int`, `float`, `double`, `char`. **Attention il n'y a cependant pas le type `boolean`, ni le type `string`.**

Exercice 3 Afficher - Saisir

Afficher se fait grâce à la fonction `printf()` dont l'usage est un peu technique. Cette fonction prend comme argument une chaîne de caractères entre guillemets, puis éventuellement des noms d'identifiants de variables. Lors de l'exécution la chaîne de caractères est affichée ainsi que la valeur de chaque variable à l'endroit indiqué. Les variables doivent apparaître dans l'ordre de leur apparitions dans la chaîne de caractères. Un entier est marqué grâce à `%d`, un float grâce à `%f`, un caractère grâce à `%c`. `\n` permet un retour à la ligne.

Voici un exemple pour clarifier :

1. Exécutez le programme suivant :

   ```
   #include<stdio.h>
   #include<stdlib.h>

   int main(){
       int i; int j; float k; char c;
       i=1; j=2; k=3.4; c='A';
       printf("la variable i vaut %d \n", i);
       printf("la variable j vaut %d et la variable k %f \n", j,k);
       printf("La première lettre de l'alphabet est %c\n", c);
   }
   ```

2. Écrivez un programme qui affiche en une seule instruction la valeur des quatre variables.

   Il est à noter qu'il existe de très nombreuses options permettant de formater la façon d'afficher. Nous en verrons des exemples plus tard.

Pour saisir un entier dans une variable `i`, l'instruction est
`scanf(“%d”,&i);`
Pour saisir un réel dans une variable `k`, l'instruction est
`scanf(“%f”,&k);`
Pour saisir un caractère dans une variable `c`, l'instruction est
`scanf(“%c”,&c);`

3. Tester le code suivant plusieurs fois :

```
#include<stdio.h>
#include<stdlib.h>

int main(){
  int j; float x; char d;
  printf("entrer un entier \n");
  scanf("%d",&j);
  printf("entrer un réel \n");
  scanf("%f",&x);
  printf("entrer un caractère \n");
  scanf("%c",&d);
  printf("%f %c %d\n",x,d,j);
}
```

4. Afin de corriger le problème, rajoutez ce code après chaque scanf :

```
scanf("%*[^\n]s");
getchar();
```

Ce code vide le buffer jusqu'au premier saut de ligne. Ainsi le prochain scanf se passe correctement. Nous vous renvoyons à cette page pour de plus amples informations :

`http://www.siteduzero.com/tutoriel-3-133769-la-saisie-securisee-avec-scanf.html`

Cette page est extraite de ce guide gratuit :

`http://www.siteduzero.com/tutoriel-3-14189-apprenez-a-programmer-en-c.html#part_14188`

Exercice 4 Quelques programmes

1. Écrivez un programme C qui demande à l'utilisateur d'entrer la valeur de deux réels et qui en retourne la moyenne.
2. Écrivez un programme C qui demande à l'utilisateur d'entrer un entier `n` et qui, si ce `n` est strictement positif, affiche la liste des entiers de 1 à `n`.
3. Écrivez un programme C qui demande à l'utilisateur d'entrer un entier `n` et qui, si ce `n` est strictement positif, affiche

```
1
1 2
1 2 3
1 2 3 4
...
1 2 3 4 ... n
```

Ce dernier programme doit être déposé sur Moodle avant 23h30 la veille de votre prochain TP.

Exercice 5 Fonction, Génération aléatoire

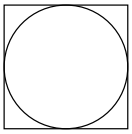
En C les fonctions se déclarent comme les méthodes statiques en JAVA, sans y mettre les mots *public static*. Par exemple la fonction suivante calcule la somme de deux entiers :

```
int somme(int i, int j){
  return i+j;
}
```

On utilise pour la génération aléatoire la fonction `rand()` qui retourne un entier entre 0 et `RAND_MAX` une constante du système.

Cette fonction `rand()` doit être initialisée une seule fois, dans le programme principal `main` en utilisant la fonction `srand(time(NULL))` qui fait appel à la bibliothèque `time.h`

1. Le fichier `hazard.c` permet d'afficher trois entiers compris entre 1 et 10. Testez-le.
2. Écrivez un programme C qui choisit aléatoirement un entier entre 0 et 100 et qui le fait deviner à l'utilisateur en lui répondant *trop grand* ou *trop petit*. **Ce programme doit être déposé sur Moodle avant 23h30 la veille de votre prochain TP**.
3. On considère la figure suivante :

   Elle représente un cercle d'un mètre de rayon dans un carré de 2 mètres de côté. L'aire du cercle vaut donc π m^2 et celle du carré 4 m^2. Si on tire une flèche au hazard dans le carré, la probabilité qu'elle soit dans le cercle est donc de $\pi/4$. Cela suggére une méthode d'approximation pour calculer π. En effet, on considère un cercle de rayon 1 centré sur le point (0,0) et un carré centré de rayon sur ce même point. Si on tire au hazard `n` points dans le carré, en moyenne `n` $*\pi/4$ de ces points seront dans le cercle. Donc le nombre de ces points dans le cercle multiplié par 4 et divisé par `n` est une approximation de π.

   (a) Proposez et testez une fonction qui retourne au hazard un nombre réel entre -1 et 1.

   (b) Si on ajoute la bibliothèque `math.h` en tête du programme : `#include <math.h>`, la formule `sqrt(n)` retourne la racine carré de `n` et `pow(a,b)` retourne a^b. Sachant que la distance du centre du carré au point de coordonnées (x, y) est $\sqrt{(x^2 + y^2)}$, écrivez la fonction `danscercle` qui prend comme argument deux réels `a` et `b` et qui indique si le point de coordonnées `(a,b)` est dans le cercle.

   Attention Lors de la compilation avec la librairie `math.h`, il faut ajouter l'option `-lm` on écrit donc `gcc -lm -o tp2 tp2.c`

   (c) Proposez un programme qui demande un entier `n` et qui donne une approximation de π en regardant combien de `n` points au hazard dans le carré sont dans le cercle. **Ce programme doit être déposé sur Moodle avant 23h30 la veille de votre prochain TP**.