PIT – Exercices C

1 Entrées-Sorties, Variables, Schéma séquentiel

Rappelons que les entrées/sorties sont les interactions du programme avec l'extérieur ; en général, l'entrée (la lecture) se fait par le clavier et la sortie (l'écriture) sur l'écran (mais on peut imaginer d'autres moyens de communications pour le programme : ficher, carte réseau, etc.). En C, les fonctions `printf` pour lire et `scanf` pour écrire sont déclarées dans l'en-tête `stdio.h`.

La fonction `printf` permet d'afficher à l'écran un message ou le contenu d'une variable.

```
#include <stdio.h>
int main(){
  printf("trop␣super\n");                          //trop super
  const int nbDeFois = 1000;
  printf("et␣encore␣%d␣fois␣plus\n", nbDeFois); //et encore 1000 fois plus
  return 0;
}
```

La lettre suivant le caractère % indique à la fonction comment afficher le contenu de la variable en fonction de son type :

- d : entier
- f : flottant
- c : caractère
- s : chaîne de caractères terminée par '\0'

La fonction `scanf` permet de lire des données frappées au clavier.

```
#include <stdio.h>
int main(){
  int n; char p;
  printf("Entrez␣un␣entier␣suivi␣d'un␣caractere␣:␣");
  scanf ("%d␣%c", &n, &p);
  printf("J'ai␣lu:␣%d␣et␣%c\n", n, p);
  return 0;
}
```

La fonction `scanf` utilise les mêmes conventions de codage que celles de `printf`. Le caractère & signifie « adresse de la variable » (on reverra ça en détails dans la matière AGP).

1. Écrire un programme de facturation appelé `facture.c`. Il demande le coût de matériel et le coût de main d'oeuvre (en nombres entiers), puis calcule et affiche le prix HT et TTC, pour une taxe au taux de 19.6% du prix HT. Si l'utilisateur rentre des coûts s'élevant à 150 et 90, le programme devra afficher ceci :

```
Cout materiel : 150
Cout main d'oeuvre : 90
Prix HT : 240.000000
Taxe : 47.040001 (taux de 19.600000)
Prix TTC : 287.040001
```

Rappelons qu'à l'exécution d'un programme, le système d'exploitation commence par appeler la fonction `main`. Les paramètres passés au programme sont stockés dans un tableau représenté par un paramètre formel de la fonction `main`, généralement appelé `argv`. Ainsi `argv[0]` est une chaine de caractères correspondant au nom du programme, `argv[1]` est une chaine de caractères correspondant au premier paramètre, `argv[2]` est une chaine de caractères correspondant au deuxième paramètre, etc. Le nombre de paramètres passés au programme est stocké dans un autre paramètre formel de la fonction `main`, généralement appelé `argc`.

```
#include <stdio.h>
int main(int argc, char* argv[]){
  printf("Nom du programme: %s\n", argv[0]);
  printf("Parametre 1: %s\n", argv[1]);
  return 0;
}
```

Le programme ci-dessus pourrait être copié dans un fichier nommé `sourceExemple.c`, puis compilé et appelé avec un paramètre (`toto`) :

```
gcc sourceExemple.c -o executableExemple
./executableExemple toto
```

Le résultat serait alors le suivant :

```
Nom du programme: ./executableExemple
Parametre 1: toto
```

2. Modifier le programme `facture.c` pour que les coûts de matériel et de main d'oeuvre soient passés comme paramètres à l'exécution du programme. La fonction `atoi`, déclarée dans l'entête `stdlib.h`, permet d'obtenir un entier à partir des caractères numériques d'une chaîne de sorte que `atoi(argv[1])` renvoie l'entier passé comme premier paramètre au programme.

2 Schéma conditionnel

Dans l'exercice précédent, les instructions sont toujours toutes exécutées les unes à la suite des autres. C'est comme une recette ou un assistant d'installation Windows : on clique sur suivant, suivant, suivant. Mais il est parfois nécessaire d'exécuter une instruction seulement quand une condition est remplie. Par exemple, si deux paramètres sont nécessaires au bon fonctionnement du programme, mais qu'aucun n'est fourni, une erreur se produit. En C, les structures `if ...else` et `switch` permettent de faire de tels branchements.

1. Modifier le programme `facture.c` pour que la facture soit affiché si et seulement si deux paramètres sont passés au programme en considérant la valeur de la variable `argc`. Dans le cas contraire, un message avertissant l'utilisateur qu'il est obligatoire de fournir deux paramètres à l'exécution du programme est affiché.

2. Écrire un programme qui affiche “Divisible par 7” si et seulement si l'entier donné par l'utilisateur est divisible par 7. L'opérateur % (modulo) permet d'obtenir le reste d'une division entière de sorte que l'expression `(n % 7) == 0` est vraie si `n` est divisible par 7.
3. Considérons les tarifs d'affranchissement suivants :
   <20g, tranche 1 : 0,5 euro
   entre 20 et 50g (exclu), tranche 2 : 1 euro
   entre 50 et 80g (exclu), tranche 3 : 1,2 euro
   80g et au-delà, tranche 4 : 1,5 euro
   Écrire un programme `lettre.c` qui, à partir du poids de votre lettre, détermine et indique dans quelle tranche se situe votre lettre (suite de `if` imbriqués), puis détermine et affiche le tarif correspondant (`switch` en fonction du numéro de tranche).

3 Schéma itératif

Dans les exercices prédédents, les instructions, quand elles sont exécutées, le sont une et une seule fois. Mais il est parfois nécessaire d'accomplir une même tâche plusieurs fois de suite. En C, les structures `for`, `while` et `do ...while` permettent de répéter un bloc d'instructions en boucle : un certain nombre de fois dans le cas du `for` ou tant qu'une (respectivement jusqu'à ce qu'une) condition est satisfaite dans le cas du `while` (respectivement `do ...while`).

3.1 `for`

1. Écrire un programme qui affiche la liste des paramètres passés à l'appel du programme en ligne de commande.
2. Écrire un programme qui affiche les entiers naturels de 1 à n (n est passé comme paramètre à l'appel du programme).
3. Écrire un programme qui n'affiche que les entiers divisibles par 3 ou par 7 parmi les n premiers entiers naturels. Calculer et afficher le nombre d'entiers divisibles par 3 ou par 7 parmi les n premiers entiers naturels. Pour cela, vous pourrez, à chaque fois qu'un entier divisible par 3 ou 7 est détecté, ajouter 1 à la valeur d'une variable initialisée à zéro et appelée par exemple `compteur`. Pour information, il y a 43 nombres divisibles par 3 ou 7 entre 1 et 100.
4. Écrire un programme `multiplication.c` qui calcule et affiche le résultat de la multiplication de a par b en ajoutant b fois a à la valeur d'une variable initialisée à zéro et appelée par exemple `somme`. Comparer le résultat en évaluant directement $a \times b$.
5. Calculer la somme des n premiers termes de la série harmonique, c'est-à-dire la somme $1 + 1/2 + 1/3 + \ldots + 1/n$.

3.2 `while` et `do ...while`

1. Écrire un programme `division.c` qui calcule la division entière a/b par soustractions successives. Comparer le résultat en évaluant directement a/b.
2. Dans le programme `lettre.c`, ajouter le code nécessaire pour simuler le paiement de l'affranchissement. L'utilisateur insère des pièces de 20, 50 centimes ou 1 euro en tapant respectivement les valeurs 0.2, 0.5 ou 1. Il doit en ajouter tant que le montant payé est inférieur au

tarif d'affranchissement. Le programme s'arrête quand le montant total est égal ou supérieur au tarif (il n'y a pas de rendu de monnaie).

3. Dans une boucle d'interaction, afficher la racine carrée de nombres saisis par l'utilisateur. Dès le début ou après chaque calcul, le programme attend la saisie d'un nombre, s'arrête lorsqu'il reçoit la valeur 0 et refuse les valeurs négatives. La bibliothèque `math.h` contient une fonction `float sqrt(float)`. Compiler en ajoutant l'option -lm (qui signifie en gros « lier avec bibliothèque math»).

4 Fonctions

Les programmes sont généralement décomposés en sous-ensembles d'instructions, rassemblés dans des fonctions. Cette décomposition fonctionnelle devient un réflexe avec l'expérience. Au début, cela aide de se demander quelles tâches on ferait soi-même, à la place de la machine. Par exemple, le programme décrit dans `lettre.c` simule le paiement de l'affranchissement d'une lettre en fonction de son poids. Une première tâche consiste à calculer le numéro de tranche en fonction du poids, une deuxième tâche consiste à déterminer le tarif d'affranchissement en fonction du numéro de tranche, une troisième consiste à payer. Ces trois tâches peuvent être écrites sous forme de fonctions.

1. Modifier le programme `lettre.c` en ajoutant et en appelant les fonctions `obtenirTranche`, `calculerTarif` et `effectuerPaiement`.
2. Modifier le programme `multiplication.c` (respectivement `division.c`) de sorte que la multiplication (respectivement division) par additions (respectivement soustractions) successives soit réalisée dans une fonction.
3. Ecrire la fonction `factorielle` qui prend en entrée un entier n et retourne le nombre $n! = 1 \times 2 \times \ldots \times n$. Appeler cette fonction dans une boucle afin d'afficher les factorielles des entiers de 1 à 30.

5 Jeu final

1. Simuler un master mind. Le programme connaît un entier mystère de 4 chiffres que l'utilisateur doit deviner. Quand l'utilisateur donne un entier, chaque chiffre est comparé à ceux de l'entier mystère. Le programme affiche (a) le nombre de chiffres communs à l'entier mystère et (b) le nombre de chiffres identiques aux chiffres de même position dans l'entier mystère (avec bien sûr, $a \geq b$). Ces indices orientent l'utilisateur pour les essais suivants et le jeu termine quand l'entier donné par l'utilisateur est exactement l'entier mystère.