Introduction à l'informatique

TD 1 : structures de contrôle, premiers exercices

Les exercices marqués d'un pique (♠) sont faits pour creuser le cours et ne seront pas systématiquement corrigés en classe entière, mais n'hésitez pas à demander à votre chargé(e) de TD.

Exercice 1 (Permutation)**.**
Voici le corps du programme **Echange** :

```
int n1, n2;
n1 = 7;
n2 = 4;

n1 = n2;
n2 = n1;

cout << n1 << endl;
cout << n2 << endl;
```

(1) En première lecture, que fait le programme **Echange** ?

(2) L'exécuter pas à pas. Obtient-on le résultat attendu ?

(3) Modifier le programme pour qu'il réponde à ce qui est attendu. Seules des créations de variables et des affectations sont nécessaires pour cela.

Exercice 2 (Division entière et modulo)**.**
Écrire un programme en langage `C++` qui lit un entier `nb` de 4 chiffres au clavier et calcule ensuite :

(1) la valeur du chiffre des unités, des dizaines, des centaines, des milliers de `nb` ;

(2) le nombre `nbInverse` obtenu en inversant les chiffres.

Puis le programme affichera le nombre `nb`, `nbInverse` et la somme des 2 nombres.
Exemple : 1239 inversé 9321 et somme = 10560

Exercice 3 (Premières instructions conditionnelles)**.**

(1) Écrire un programme qui demande à l'utilisateur de saisir un entier n au clavier et qui affiche s'il est pair ou impair.

(2) Écrire un programme qui demande à l'utilisateur de saisir un entier n au clavier et qui affiche la valeur absolue de cet entier.

(3) Écrire un programme qui demande à l'utilisateur de saisir trois entiers : `a, b, c` au clavier et qui affiche le plus grand de ces trois entiers.

Exercice 4 (Autour des boucles)**.**
Écrire un programme qui prend un entier **n** ($n >= 0$) en paramètre et calcule la somme $1^2 + 2^2 + \cdots + n^2$. Il vous est demandé d'écrire un programme pour chaque type de boucle :

(1) **Tant que ... faire ...** (`while`)
(2) **Répéter ... tant que ....** (`do ... while`)
(3) **Pour ... allant de ... à ... faire ....** (`for`)

Remarque : pensez à vérifier que le résultat vaut bien 0 pour $n = 0$.

Exercice 5 (Plus ou Moins)**.**
Écrire un programme pour le jeu « Plus ou Moins » dont le but est de deviner un nombre choisi aléatoirement entre 0 et 1000 par l'ordinateur (on pourra supposer que l'expression `aleaint()` donne un tel nombre aléatoire).

À chaque étape et jusqu'à ce qu'il trouve le bon nombre, le joueur propose un nombre à l'ordinateur qui lui indique si ce nombre est inférieur ou supérieur au nombre à trouver. Une fois que le joueur a trouvé le bon nombre, on affiche le nombre d'essais du joueur.

Exercice 6 (Petites questions ... logiques)**.**

(1) Étant donnés trois entiers a, b et c, calculer la valeur d'un booléen *resultat* tel que *resultat* soit vrai si l'un des trois entiers est plus grand que 10, ou plus petit d'au moins 2 de chacun des deux autres.
(2) Votre téléphone sonne. Vous répondez généralement à votre téléphone lorsque vous ne dormez pas, sauf si on est le matin (peu après 8h15 par exemple) auquel cas vous ne répondez que s'il s'agit d'un parent proche. Calculer un booléen *resultat* dépendant de trois booléens *estEndormi*, *estLeMatin* et *estParentProche*, dont la valeur est vrai si vous allez répondre.

Exercice ♠ 7 (Conditionnelles et opérations sur les entiers)**.**
On considère une machine à distribuer des sucreries. Le problème consiste à écrire le programme qu'elle exécute pour rendre la monnaie sur une somme, à l'aide de pièces de 50 centimes, 20 centimes, 10 centimes et 5 centimes d'euro, de façon à minimiser le nombre de pièces rendues sachant que l'on connaît la somme due et la somme donnée par le client. On suppose que les sommes sont données en centimes d'euro, qu'il n'y a pas de risque de pénurie de pièces de monnaie, et que les prix sont un multiple de 5 centimes.

Travail à faire : Analyser le problème et proposer un algorithme pour le résoudre. Le comportement de l'algorithme, du point de vue de l'utilisateur, devra être le suivant, par exemple pour une barre chocolatée de 1,30 euros payée avec une pièce de 2 euros on aura :

```
Entrer le prix a payer en centimes d'euro : 130
Entrer la somme versée en centimes d'euro : 200
Il faut rendre :
- 1 pièce(s) de 50 centimes d'euro
- 1 pièce(s) de 20 centimes d'euro
```

Vous remarquerez que les pièces n'intervenant pas dans la transaction ne sont pas citées.

Plus difficile : Comment gérer un nombre limité de pièces en réserve dans la caisse de la machine ?

Annexe : résumé de la syntaxe de base C++

Les exemples suivants résument la syntaxe des instructions de base C++, et précisent les conventions de codage utilisées dans le cadre de ce module : indentation, espacement, documentation au format javadoc [1] et tests.

```
#include <iostream>              // Squelette de programme
using namespace std;
int main () {
    ...
}
```

```
cin >> n;                        // Lit la variable n au clavier
cout << 3*x+1;                   // Affiche la valeur d'une expression
cout << endl;                    // Affiche un saut de ligne
```

```
if ( x == 1 ) {                  // Instruction conditionnelle
    ...;
} else if ( x < 2 and not y > 3 ) {
    ...;
} else {
    ...;
}
```

```
for ( int i=0; i < 10; i++ ) {// Instruction itérative : boucle for
    ...;
}
```

```
while ( i <= 10 ) {              // Instruction itérative : boucle while
    ...;
}
```

```
do {                             // Instruction itérative : boucle do ... while
    ...;
} while ( i <= 10 );
```

```
/** La fonction factorielle
 * @param  n un nombre entier positif
 * @return n!
 **/
int factorielle(int n) {
    int resultat = 1;
    for ( int k = 1; k <= n; k++ ) {
        resultat = resultat * k;
    }
    return resultat;
}

/** Les tests de la fonction factorielle **/
void factorielleTest() {
    ASSERT( factorielle(0) == 1 );
    ASSERT( factorielle(1) == 1 );
    ASSERT( factorielle(2) == 2 );
    ASSERT( factorielle(3) == 6 );
    ASSERT( factorielle(4) == 24);
}
```

1. Pour plus de détails, voir par exemple http://fr.openclassrooms.com/informatique/...