TD 2 : affectations, instructions conditionnelles

Exercice 1.
Voici le corps du programme `Échange` :

```
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 `Échange` ?
Correction. D'après son nom, le programme devrait échanger les valeurs de `n1` et `n2`.

(2) L'exécuter pas à pas. Obtient-on le résultat attendu ?
Correction. En fait le programme affiche
`n1 : 4`
`n2 : 4`

(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.
Correction.

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

temporaire = n2;
n2 = n1;
n1 = temporaire;

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

Exercice 2.
Écrire le corps d'un programme qui lit un entier n saisi au clavier et affiche s'il est pair ou impair.

Correction. En `C++`, si n et k sont des entiers, alors n/k est le quotient de la division euclidienne de n par k, et n%k est le reste de la division euclidienne de n par k (voir Cours 1, partie E. Expressions arithmétiques)

```
cin >> n;
if (n % 2 == 0) {
  cout << "Le nombre " << n << " est pair" << endl;
} else {
  cout << "Le nombre " << n << " est impair" << endl;
}
```

Exercice 3.
Lors de la fin d'un semestre dans une Université X, les enseignants sont amenés à calculer la moyenne générale des notes de physique et de mathématiques selon une règle précise : la meilleure note des trois épreuves de mathématiques est comptée coefficient 3, et la meilleure note des deux épreuves de physique est comptée coefficient 2 ; les autres notes ne comptent pas.

Un enseignant d'informatique (est-ce son métier ?) est chargé de concevoir un algorithme prenant en entrée les trois notes de mathématiques et les deux notes de physique, et donnant la moyenne générale suivant la règle énoncée ci-dessus.

(1) Spécifier et écrire un algorithme qui, étant donné deux entiers a et b renvoie le plus grand des deux.

Correction. `max2(int a, int b)` :

```
// Entrées: deux nombres a et b
// Sortie: le maximum de a et b
int max;
if (a < b) {
  max = b;
} else {
  max = a;
}
return max;
```

(2) De même, étant donnés trois entiers, donner un algorithme renvoyant le plus grand des trois.

Correction. `max3(int a, int b, int c)` :

```
// Entrées: trois nombres a, b et c
// Sortie: le maximum de a, b et c
int max;
if (a <= c and b <= c) {
  max = c;
} else if (a <= b and c < b) {
  max = b;
} else if (a > b and a > c) {
  max = a;
}
return max;
```

(3) Spécifier et donner un algorithme qui prend en entrée trois notes de mathématiques, puis deux notes de physique, et renvoie la moyenne selon la règle spécifiée.

Correction. Programme global `moyenne(int m1, int m2, int m3, int p1, int p2)` :

```
        // Entrées: m1, m2, m3: les notes de math,
        //          p1, p2: les notes de physique
        // Sortie: la moyenne
        return (max3(m1, m2, m3) * 2 + max2(p1, p2) * 3) / 5.0
```

Exercice 4.
On considère une machine qui distribue 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 1,10 euros payée avec une pièce de 2 euros on aura :

```
Entrer le prix a payer en centimes d'euro : 110
Entrer la somme versée en centimes d'euro : 200
Il faut rendre :
- 1 pièce(s) de 50 centimes d'euro
- 2 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 ?

Correction. Rendre la monnaie

```
int prix, somme, rendu, npieces;
cout << "Entrez le prix à payer en centimes d'euro: ";
cin >> prix;
cout << "Entrez la somme versée en centimes d'euro: ";
cin >> somme;
rendu = somme - prix;
npieces = 0;
npieces = rendu / 50;
rendu   = rendu % 50
if (npieces > 0) {
  cout << npieces << " pièce(s) de 50 centimes" << endl;
}
npieces = rendu / 20;
rendu = rendu % 20;
if (npieces > 0) {
  cout << npieces << " pièce(s) de 20 centimes" << endl;
}
npieces = rendu / 10;
rendu = rendu % 10
if (npieces > 0) {
  cout << npieces << " pièce(s) de 10 centimes" << endl;
}
npieces = rendu / 5;
rendu = rendu % 5
if (npieces > 0) {
  cout << npieces << " pièce(s) de 5 centimes" << endl;
}
```

L'exercice suivant est tiré du **Projet Euler**[1], une série de défis, de difficulté croissante, mêlant mathématiques, algorithmique, et programmation. Chaque problème possède une unique solution qu'il s'agit de découvrir par soi-même, ce qui permet d'accéder à un forum consacré aux différentes approches menant à sa résolution. L'ensemble constitue une sorte de parcours initiatique, en ce sens que la résolution des énigmes les plus simples octroie progressivement au joueur des mécanismes préparant à celle des plus complexes.

Exercice ♣ 5 (Projet Euler, problème 19)**.**
Les informations suivantes te sont données, mais tu préfères peut-être rechercher cela par toi-même.

- Le 1er janvier 1900 était un lundi.
- 30 jours comptent septembre, Avril, juin et novembre.
- Tous les autres en ont trente-et-un,
- Sauf février le plaisantin, Qui en compte vingt-huit, sans bluff.
- Et les années bissextiles, vingt-neuf.
- Une année bissextile a lieu chaque année divisible par 4, mais pas multiple de 100 sauf si divisible par 400.

Combien de dimanches sont tombés le premier jour du mois au cours du vingtième siècle (du 1er janvier 1901 au 31 décembre 2000) ?

1. `http://projecteuler.net/` ; voir `http://submoon.freeshell.org/fr/sphinx/euler.html` pour les énoncés en français

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 [2] et tests. Complétez cette feuille à la main au verso comme vous le souhaitez. Ce sera le seul document autorisé au partiel et à l'examen.

```
#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 );
```

2. Pour plus de détails sur javadoc, voir par exemple `http://fr.openclassrooms.com/informatique/cours/presentation-de-la-javadoc/les-tags-javadoc-1`

```
    ASSERT( factorielle(2) == 2 );
    ASSERT( factorielle(3) == 6 );
    ASSERT( factorielle(4) == 24);
}
```