

TD 2 : affectations, instructions conditionnelles**Exercice 1.**

Voici le corps du programme `Échange` :

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

n1 = n2;
n2 = n1;
```

- (1) En première lecture, que fait le programme `Échange` ?
- (2) L'exécuter pas à pas. Obtient-on les valeurs attendues dans les variables `n1` et `n2` ?
- (3) Modifier le programme pour qu'il réponde à ce qui est attendu. Seules des déclarations de variables et des affectations sont nécessaires pour cela.

Exercice 2.

En relisant votre cours après l'amphi, vous avez appris les opérations sur les entiers ainsi que les opérations booléennes. Cela vous sera utile dans cet exercice.

On suppose que l'on a déjà déclaré et initialisé une variable `n` de type entier. On veut diviser `n` par 2 s'il est pair et lui enlever 1 s'il est impair, et stocker le résultat dans une variable `resultat`. Écrire les quelques lignes d'instructions correspondantes.

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 donnés deux entiers `a` et `b` (qu'on suppose déjà déclarés et initialisés), calcule le plus grand des deux et met le résultat dans une variable `max`.
- (2) ♣ Transformer cet algorithme en une fonction nommée `max2` pour pouvoir par la suite calculer le maximum de deux nombres avec un appel à la fonction, par exemple `max2(7, 5)`.
- (3) De même, étant donnés trois entiers, donner un algorithme calculant le plus grand des trois (dans une variable `max`). On pourra utiliser la fonction `max2(a, b)`.
- (4) ♣ Transformer cet algorithme en fonction.
- (5) Spécifier et donner un algorithme qui prend en entrée trois notes de mathématiques, puis deux notes de physique, et calcule la moyenne selon la règle spécifiée.

- (6) ♣ Transformer cet algorithme en fonction.

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 le prix 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.

Par exemple si le prix à payer est de 110 centimes et que le client a donné 200 centimes, il faut lui rendre 1 pièce de 50 centimes et 2 pièces de 20 centimes.

- (1) Quelles sont les entrées / sorties du problème ?
- (2) Écrire un programme pour résoudre le problème. On pourra supposer que les variables `prix` et `somme` contiennent respectivement le prix et la somme donnée par le client. À la fin du programme, la variable `nbpieces50` devra contenir le nombre de pièces de cinquante centimes à rendre, et de même pour les variables `pieces20`, `pieces10` et `pieces5`.
- (3) ♣ Comment gérer un nombre limité de pièces en réserve dans la caisse de la machine ?

Exercice ♣ 5.

Héloïse et Gabriel veulent jouer à Pierre-Feuille-Ciseaux, mais, têtes en l'air, ils ne se souviennent jamais de qui bat quoi.

- (1) On numérote les joueurs par 1 et 2, et on représente un choix par un entier : 1 pour une pierre, 2 pour une feuille, et 3 pour les ciseaux. On suppose que les variables `choix1` et `choix2` ont été déclarées et initialisées respectivement avec les choix du joueur 1 et 2. Écrire un programme dont la sortie, stockée dans une variable `gagnant` contienne le numéro du joueur gagnant (ou zéro si égalité).
- (2) Transformer votre programme sous forme d'une fonction.
- (3) Même chose en représentant un choix par un caractère (type `char`) : 'p' pour pierre, 'f' pour feuille, 'c' pour ciseaux.
- (4) ♣ Même chose en utilisant un type `enum`.

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 ♣ 6 (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 javadoc² 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.

```
if ( x == 1 ) {                               // Instruction conditionnelle (if)
    ...;
}
```

```
if ( x == 1 ) {                               // Instruction conditionnelle (if/else)
    ...;
} 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          // Documentation de la fonction factorielle
 * @param n un nombre entier positif
 * @return n!
 */
int factorielle(int n) {                  // Exemple de déclaration de fonction
    int resultat = 1;
    for ( int k = 1; k <= n; k++ ) {
        resultat = resultat * k;
    }
    return resultat;
}
```

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

```
#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
```

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