# Travaux Pratiques de programmation n°4

**Cours de programmation impérative**

*—Licence MPI L1 S2 - Info 121—*

## Fonctions et procédures: passage de paramètres

Cette séance de travaux pratiques est dédiée au passage de paramètres. On rappelle que en C++ le mode par défaut est **par valeur**, les arguments sont alors **recopiés**. Le C++ fournit un autre mode dit **par référence** où l'on fait précéder le paramètre formel du symbole `&`. Voici un exemple : la procédure suivante ajoute 2 à la variable passée en paramètre :

```
void incremente(int &i) {
    i = i + 2;
}
```

▸ **Exercice 1.** La fonction suivante retourne un entier positif donné par l'utilisateur :

```
int litpositif(string nom) {
  int resultat;
  do {
    cout << "Donner la valeur de l'entier positif " << nom << " : ";
    cin >> resultat;
  } while (resultat < 0);
  return resultat;
}
```

1. Modifier la fonction `litpositif` pour en faire une procédure qui renvoie l'entier positif lu et le nombre d'erreurs (entiers négatifs) faites par l'utilisateur.
2. Écrire un programme qui utilise la procédure précédente, affiche l'entier lu et le nombre d'essais.

▸ **Exercice 2. (Transactions bancaires)**

Nous nous occupons de la gestion de comptes bancaires. Il faut nous assurer qu'aucune somme d'argent n'est créée ou perdue dans les virements et que les comptes restent toujours au-dessus du solde autorisé. Nous disposons des comptes tels que définis ci-dessous :

```
1   #include <iostream>
2   using namespace std;
3
4   int compte1 = 1000;
5   int compte2 = 2000;
6   int compte3 = 1500;
7   int compte4 = 3000;
8
9   void etat_comptes();
10  bool virement(int compte_orig, int compte_dest, int somme);
```

```
11
12  void etat_comptes() {
13    cout << "Etat des comptes : " << endl;
14    cout << "Compte n 1 : " << compte1 << endl;
15    cout << "Compte n 2 : " << compte2 << endl;
16    cout << "Compte n 3 : " << compte3 << endl;
17    cout << "Compte n 4 : " << compte4 << endl;
18  }
19
20  int main() {
21    bool v;
22    etat_comptes();
23    v = virement(compte1, compte2, 100);
24    etat_comptes();
25    if (v) { cout << "Virement effectué !" << endl };
26    else   { cout << "Virement impossible !"<< endl };
27    return 0;
28  }
```

1. Définir et appeler la fonction :

   `bool virement(int compte_orig, int compte_dest, int somme)`

   qui verse si possible (c'est-à-dire si le versement ne rend pas le solde de `compte_orig` négatif) un montant d'argent égal à `somme` depuis le `compte_orig` vers le `compte_dest`. Cette fonction doit retourner `true` si et seulement si le versement a été effectué et renvoie `false` sinon.

2. Vérifier ensuite les valeurs de comptes qui ont été passées en paramètres de la fonction `virement`. Une fois cela fait, nous constaterons que les valeurs sont inchangées, alors qu'un virement devait normalement les avoir changées. Ceci est dû au fait que les paramètres ont été passés par valeur à la fonction.

3. Pour résoudre le problème constaté à la question précédente, nous allons transformer la fonction en procédure et passer les paramètres par référence. Modifier la fonction `virement` en une procédure :

   `void virement(int &compte_orig, int &compte_dest, int somme, bool &virement_ok)`

   Cette procédure doit renvoyer `true` dans le paramètre `virement_ok` si et seulement si le versement a été effectué et renvoie `false` sinon. Vous devrez aussi modifier le corps de la procédure pour utiliser les paramètres `compte_orig` et `compte_dest`. Appelez enfin cette procédure (dans un programme `main`).

▸ **Exercice 3. (Nombres complexes)**

On rappelle qu'un nombre complexe est un nombre qui s'écrit $a + ib$ où $a$ est appelé partie réelle et $b$ partie imaginaire. On demande de :

1. Écrire une procédure `saisie` qui saisit un nombre complexe.
2. Écrire une procédure `affiche` qui affiche un nombre complexe.
3. Écrire une procédure `somme` qui calcule la somme de deux nombres complexes.
4. Écrire une procédure `produit` qui calcule le produit de deux nombres complexes.
5. Écrire une fonction `norme_carre` qui retourne le carré de la norme d'un nombre complexe. On rappelle que
$$|a + ib|^2 = a^2 + b^2 \,. \tag{1}$$
6. Écrire une procedure `inverse` qui retourne l'inverse d'un nombre complexe. On rappelle que
$$\frac{1}{a + ib} = \frac{a - ib}{|a + ib|^2} \,. \tag{2}$$
7. L'algorithme de calcul de la racine carrée du TP 2 fonctionne en général encore sur les nombres complexes : la suite
$$u_0 := a, \qquad u_{n+1} := \frac{u_n + a/u_n}{2} \tag{3}$$
converge vers une racine carrée de $a$. Écrire une procédure `racine` qui calcule la racine carrée d'un nombre complexe. On considère que l'approximation $u$ est correcte si
$$\frac{|u^2 - a|}{|a|} < 10^{-6} \,, \qquad \text{c'est-à-dire si} \qquad \frac{|u^2 - a|^2}{|a|^2} < 10^{-12} \,. \tag{4}$$