LIFAP1 – TD 5 : Sous-programmes et passage de paramètres

Objectifs : Comprendre la différence entre les modes de passage des paramètres : donnée ou donnée / résultat
Comprendre la différence entre paramètres formels et paramètres effectifs

Recommandations : Pour chacun des algorithmes que vous écrirez vous préciserez le mode de passage des paramètres (**donnée** ou **donnée / résultat**) et vous écrirez le programme principal appelant les sous-programmes que vous aurez écrits.

Données (passage par valeur)
- Le sous-programme dispose d'une copie de la valeur.
- Il peut la modifier, mais l'information initiale dans le code appelant n'est pas affectée par ces modifications.

Syntaxe en C/C++ : type nom ;

Résultats ou **données / résultats** (passage par adresse)
- Le sous-programme dispose d'une information lui permettant d'accéder en mémoire à la valeur que le code appelant cherche à lui transmettre.
- Il peut alors modifier cette valeur, le code appelant aura accès aux modifications faites sur la valeur.

Syntaxe en C/C++ : type **&** nom ;

1. Un nombre **parfait** est un nombre naturel n non nul qui est égal à la somme de ses diviseurs stricts (n exclus). Exemple : 6 = 1 + 2 + 3
   a. Écrire en langage algorithmique une fonction booléenne qui retourne vrai si un entier n passé en paramètre est un nombre parfait, faux sinon.

```
Fonction parfait (n : entier) : booléen
Précondition : n > 0
Donnée : n
Résultat : booléen
Description : retourne vrai si n est parfait, faux sinon
Variable locale : res : booléen, i, som : entiers
Début
  som ← 0
  Pour i allant de 1 à n-1 par pas de 1 faire
      Si (n modulo i ) = 0  Alors som ← som + i
      Fin si
  Fin pour
  Retourner ( n = som)
Fin

Remarque : dès que som > n, on peut s'arrêter avec un tantque
```

   b. Écrire en langage algorithmique le programme principal permettant d'afficher la liste des nombres parfaits compris entre 1 et 10000. On utilisera le résultat renvoyé par la fonction précédente.

```
Début
  Variables locales : i : entier
  Pour i allant de 1 à 10000 par pas de 1 faire
    Si parfait(i)   Alors   Afficher (i, "est un nombre parfait")
                            Afficher (saut de ligne)
    Fin si
  Fin pour
Fin
```

2. Soit le programme suivant. Identifiez et notez :
   a. le(s) paramètre(s) formel(s) / le(s) paramètre(s) effectif(s)
   b. le(s) paramètre(s) en donnée / le(s) paramètre(s) en donnée / résultat
   c. Qu'est censé faire ce programme ?
   d. Quelle(s) modification(s) faudrait-il apporter pour obtenir un résultat plus logique ?

```
#include <iostream>
using namespace std ;

void mystere (int a, int b, int &c, int d)
{c=a+b;
 d=a*b;
}

int main (void)
{int e,f,g,h;
 cout<<"donnez une valeur";
 cin>>e;
 cout<<"donnez une valeur";
 cin>>f;
 mystere(e,f,g,h);
 cout<<" valeur "<<g<<" valeur :"<<h<<endl;
 return 0;
}
```

Profitez de ce premier exercice pour faire quelques rappels de cours en donnant les définitions…

Rappels de cours (définition) :

Paramètre formel : variable utilisée dans le corps du sous-programme qui reçoit une valeur de l'extérieur (ils font partie de la description de la fonction)
Paramètre effectif : il s'agit de la variable (ou valeur) fournie lors de l'appel du sous-programme (valeurs fournies pour utiliser la fonction et valeurs renvoyées)
Copie de la valeur du paramètre effectif vers le paramètre formel correspondant lors de l'appel.
Paramètres formel et effectif ont des noms différents

Données (passage par valeur) : le sous-programme dispose d'une copie de la valeur.
Il peut la modifier, mais l'information initiale dans le code appelant n'est pas affectée par ces modifications.
Syntaxe en C/C++ : type nom ;
Résultats ou données / résultats (passage par adresse) : le sous-programme dispose d'une information lui permettant d'accéder en mémoire à la valeur que le code appelant cherche à lui transmettre. Il peut alors modifier cette valeur, le code appelant aura accès aux modifications faites sur la valeur.
Syntaxe en C/C++ : type & nom ;

Paramètres formels : a, b, c et d
Paramètres effectifs : e, f, g, h
Paramètres en donnée : a, b, d
Paramètres en donnée / résultat : c

Le programme est censé calculer et retourner la somme et le produit de deux variables a et b. La somme est stockée dans la variable c et le produit dans la variable d. Pour obtenir le résultat attendu, il faut passer le paramètre formel d en donnée / résultat sinon la valeur calculée dans la procédure est perdue définitivement.

3. Écrivez l'algorithme d'une procédure effectuant la permutation circulaire de trois variables : a=5 b=8 et c=2 donne après exécution : a=2 b=5 et c=8

```
Procédure permutation_circulaire (a : entier, b : entier, c : entier)
Précondition : aucune
Données / Résultats : a, b et c
Description : effectue la permutation circulaire des 3 variables a, b et c
Variable locale : tampon : entier
Début
  tampon ← c
  c← b
  b← a
  a← tampon
Fin permutation_circulaire

Appel :
Début
  Variables locales : v1, v2, v3 : entier
  Afficher ('première valeur')
  Saisir (v1)
  Afficher ('deuxième valeur')
  Saisir (v2)
  Afficher ('troisième valeur')
  Saisir (v3)
  permutation_circulaire (v1,v2,v3)
  Afficher ('nouvelles valeurs : ', v1, ' ', v2, ' ', v3)
Fin
```