LIFAP1 – TD 4 : 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.

Soit le programme suivant :

```
#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;
}
```

1. 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 ?

2. 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.

3. Écrivez l'algorithme d'une procédure permettant d'effectuer la division euclidienne de deux entiers a et b. On appellera q le quotient et r le reste de cette division. On rappelle la formule de la division : a = b*q + r, avec r < b.

4. Écrivez l'algorithme d'une fonction perimetre_cercle permettant de retourner le périmètre d'un cercle en fonction de son rayon (passé en paramètre). Écrivez ensuite une fonction aire_cercle qui retourne l'aire d'un cercle. On souhaite maintenant écrire un sous-programme (qui utilise les deux fonctions précédentes) permettant à partir du rayon d'un cercle de calculer son périmètre et sa surface. Écrivez l'entête de ce sous programme de deux manières différentes.

5. Écrivez l'algorithme d'une fonction qui à partir de deux entiers n et p calcule le nombre de combinaisons de p éléments pour un ensemble de n éléments. Rappel : $C_n^p = n ! / (p ! * (n-p) !)$.
   Transformez cette fonction en procédure puis traduisez en langage C.

6. Un nombre entier est dit "doublon" si le produit de ses diviseurs est multiple de la somme de ses diviseurs.
   Exemple : n = 6. Les diviseurs de n sont : 1, 2, 3, 6. La somme des diviseurs est 12 et le produit des diviseurs est 36 (= 3 * 12). Le produit des diviseurs de `n` est donc un multiple de la somme des diviseurs de `n` donc `n` est un nombre doublon.
   a. Ecrire l'algorithme d'un sous-programme `somme_produit` permettant de calculer et "renvoyer" au programme principal la **somme des diviseurs et le produit des diviseurs** d'un nombre `n` passé en paramètres.
   b. Ecrire l'algorithme d'une **fonction booléenne** `verifie_doublon` qui retourne vrai si un entier passé en paramètres est un doublon, faux sinon. On utilisera pour cela le sous-programme écrit dans la question précédente.

Pour s'entraîner

1. Ecrire l'algorithme d'une procédure permettant à partir des trois coefficients a, b et c d'un polynôme du second degré, de calculer et retourner (si elles existent) les racines.