Leçon 4 : Les Procédures et les Fonctions en C++

Table des matières

1- Les Procédures

I

Définition : 1.1- Définition

Une procédure est une série d'instructions regroupées sous un nom, qui permet d'effectuer des actions par un simple appel de la procédure dans le programme. Elle se caractérise par le mot clé *void* qui précède le nom du sous-programme lors de sa création.

Syntaxe : 1.2- Création

void Nom_Procedure([type1 Paramètre1,. . .,type n Paramètre n])

{

Bloc d'instructions ;

}

REMARQUE :

void: pour signifier qu'aucune valeur n'est retournée.

Nom_Procedure : Désigne le nom de la procédure.

type1 à type n : Encore appelé arguments, ils déterminent les éventuels paramètres de la procédure.

Paramètre1 .. Paramètre n : ce sont des paramètres formels c'est à dire paramètres définis lors de la création de la procédure, ils ne contiennent pas de valeur réel.

Bloc d'instructions : il représente l'ensemble des instructions devant être exécutées dans le sous-programme.

NB : les crochets définis dans les paramètres de la procédure désignent le fait que ces derniers soient facultatifs car une procédure peut ne pas avoir de paramètres.

1.3- Appel d'une procédure

Toute procédure créée est appelée dans le programme principal à partir de son nom suivi de valeurs réelles (paramètres effectifs). Ces valeurs réelles (généralement saisie par l'utilisateur) remplaceront les paramètres formels définis lors de la création de la procédure.

Syntaxe :

Nom_Procedure(valeur1,. . ., valeur n)

Exemple : 1.4- Application

En utilisant une procédure, écrire un programme en langage C++ permettant de faire la somme de deux nombres.

```
#include <iostream> //permet d'utiliser le cout
using namespace std; // permet d'utiliser le endl
/*** Déclaraction de la procédure ******/
void somme(int nb1, int nb2) // entête de la procédure avec les paramètres qui vont récuperer les données du programme princpal
{
 int som; // variable qui va recevoir le resultat.
 som = nb1 + nb2; // calcul de la somme
 cout<<nb1<<" + "<<nb2<<" = "<<som; // affiche du résultat.
}
//Programme principal
main()
{
    int nb1,nb2; // Variable globale
   //  Récupere les deux nombres
    cout<<"Donner le premier nombre"<<endl;
    cin>>nb1;
    cout<<"Donner le deuxième nombre"<<endl;
    cin>>nb2;
// Appel de la procédure avec les différents paramètre.
 somme(nb1,nb2);


}
```

Exercice

II

Énoncé :

Écrire une procédure qui affiche à l'écran une ligne de 10 arobase.

Solution :

```
{
"@" ;
}
{
i;
( ; 10; )
{
arobase ;
}
}
```

2- Les Fonctions

Définition : 2.1- Définition

La fonction est un sous-programme admettant des paramètres et retournant un seul résultat et un seul de type simple qui peut apparaître dans une expression, dans une comparaison, à la droite d'une affectation, etc. La valeur retournée est du fait du mot clé *return* spécifié dans la fonction.

Syntaxe : 2.2- Déclaration

TYPE Nom_Fonction([type1 Paramètre1,. . .,type n Paramètre n])

{

Bloc d'instructions ;

return (valeur retournée) ;

}

REMARQUE :

TYPE : Il définit le type de la valeur retournée.

Nom_Fonction : Désigne le nom de la fonction.

type1 à type n : Encore appelé arguments, ils déterminent les éventuels paramètres de la fonction.

Paramètre1 .. Paramètre n : ce sont des paramètres formels c'est à dire paramètres définis lors de la création de la fonction, ils ne contiennent pas de valeur réel.

Bloc d'instructions : il représente l'ensemble des instructions devant être exécutées dans le sous-programme.

return : retourne une valeur de même type que celui qui précède le nom de la fonction.

NB : les crochets définis dans les paramètres de la fonction désignent le fait que ces derniers soient facultatifs car une fonction peut ne pas avoir de paramètres.

2.3-Appel d'une fonction

Toute fonction créée est appelée dans le programme principal à partir de son nom suivi de valeurs réelles (paramètres effectifs). Ces valeurs réelles (généralement saisie par l'utilisateur) remplaceront les paramètres formels définis lors de la création de la fonction.

SYNTAXE :

variable = Nom_Fonction(valeur1,. . ., valeur n)

REMARQUE :

variable reçoit la valeur retournée par la fonction.

Exemple : 2.4- Application

En utilisant une fonction , écrire un programme en langage C++ permettant de renvoie le plus grand de deux nombres différents.

```
#include <iostream> //permet d'utiliser le cout
using namespace std; // permet d'utiliser le endl

//Création de la fonction maxi avec deux paramètres (paramètres formels)
int max(int a, int b)
{
 //Déclaration d'une variable locale
int c;
//définition du bloc d'instructions
if(a>b)
{
 c=a;
}
else
{
c=b;
}
//valeur retournée est c
return(c);
}
// Programme principal
main()
{
//Déclaration des variables globales
int nb1, nb2;
 cout<<"Donner le premier nombre"<<endl;
 cin>>nb1;
 cout<<"Donner le deuxième nombre"<<endl;
 cin>>nb2;
 cout<<"La plus grande valeur de "<<nb1<<" et "<<nb2<<" est "<<max(nb1,nb2);
}

```

Exercice

IV

Énoncé :

Écrire une fonction qui calcul un nombre à la puissant n.

Solution :

```
puissance( nbre, int )
{
int i;
double p;
p=1;
( ; n; )
{
p nbre;
}
;
}

{
int exp;
double puis,nb;
"Donner le nombre" endl;

Donner l'exposant" endl;

puis =

"Le resultat de " " exposant " " est : "
}
```

3- Les Passages de Paramètres

V

3.1- Passage par valeur

Dans ce type de passage, le paramètre formel reçoit uniquement une copie de la valeur du paramètre effectif. *La valeur du paramètre effectif ne sera jamais modifiée*.

Autrement dit à l'appel de du sous-programme, *les paramètres formels sont remplacés par les valeurs réelles. Toute modification des paramètres formels dans le sous-programme n'entraîne aucune modification des valeurs réelles (paramètres effectifs) à la sortie.*

3.2- Passage par référence ou par adresse

Dans ce type de passage, la procédure ou la fonction utilise l'adresse du paramètre effectif. *Lorsqu'on utilise l'adresse du paramètre, on accède directement à son contenu*. Ainsi, à l'appel de la fonction ou de la procédure, les paramètres formels sont remplacés par les valeurs réelles. *Toute modification des paramètres formels dans le sous-programme modifie les valeurs réelles (paramètres effectifs) à la sortie.*

Remarque :

Pour utiliser l'adresse du paramètre effectif il faut mettre à la suite du type formel un <<&>>

Exemple : 3.3- Application

En utilisant les sous-programme, écrire un programme qui fait la somme de deux entiers.

```
1 #include <iostream> //permet d'utiliser le cout
2 using namespace std; // permet d'utiliser le endl
3  // Procédure pour recupérer les valeurs en utilisant le passage par réference
  car ici les paramètre effectifs doivent être modifiés
4 void recupdonne(int& nb1, int& nb2)
5 {
6     cout<<"Donner le premier nombre"<<endl;
7     cin>>nb1;
8     cout<<"Donner le deuxième nombre"<<endl;
9     cin>>nb2;
10 }
11  // fonction pour faire le calcul en utilisant le passage par valeur car ici les
  paramètre effectifs ne seront pas  modifiés
12 int calcul(int nb1, int nb2)
13 {
14     int som;
15     som = nb1 + nb2;
16     return som;
17 }
```

```
18 // Procédure pour afficher le resultat les valeurs en utilisant le passage par
   valeur car ici les paramètre effectifs ne seront pas  modifiés
19 void affichage(int nb1, int nb2,int som)
20 {
21     cout<<nb1<<" + "<<nb2<<" = "<<som;
22 }
23 //Programme principal
24 main()
25 {
26  int A,B,resu; // Déclaration des variables globales
27  recupdonne(A,B); // appel de la procédure recupdonne avec les paramètres
   effectifs
28  resu = calcul(A,B);// appel de la fonction calcul avec les paramètres effectifs
29  affichage(A,B,resu); // appel de la procédure affichage avec les paramètres
   effectifs
30
31 }
```

Exercice

VI

Énoncé :

En utilisant les sous-programme, écrire un programme qui calcule le périmètre et l'aire d'un rectangle.

Solution :

```
recupdim( longu, larg)
{
Donner la longueur" endl;
Donner la largeur" endl;
}
calculperi( longu, larg)
{
double peri;
= (longu+larg)
}
calculaire( longu, larg)
{
double aire;
= longu
}
affichage( peri, aire)
{
"Le périmètre est " endl;
"L'aire est " endl;
```

```
}
main()
{
double lar,lon,P,A;

A=calculaire(lon,lar);

}
```