LES SOUS-PROGRAMMES EN LANGAGE C

SANE Arnaud

Table des matières

Objectifs

- Etre capable de manipuler une fonction en Langage C
- Etre capable de manipuler une procédure en langage C

LES FONCTIONS

Les sous-programmes sont des programmes créés pour exécuter une tâche particulière dans un programme appelé programme principal. ils peuvent être appelés plusieurs fois dans le programme principal. Un sous-programme est défini juste avant le main(). Il existe deux types de sous-programmes à savoir, les fonctions et les procédures.

Les variables définies à l'intérieur d'un sous-programme (procédure ou fonction) sont appelées *variables locales*, elles ne peuvent être manipulées qu'à l'intérieur de ce sous-programme uniquement.

Celles définies juste après les bibliothèques sont dites *variables globales* et peuvent être manipulées dans le programme principal et dans n'importe quel sous-programme.

Définition

Une fonction est un sous-programme qui retourne une valeur dans le programme appelant. Cette valeur retournée est du fait du mot clé *return* spécifié dans la fonction, son type est défini au moment de la création de la fonction.

CREATION D'UNE FONCTION

SYNTAXE :

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.

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 DE FONCTION

Ecrivons un programme qui, à partir d'une fonction retourne la plus grande valeur de deux nombres entiers saisis par l'utilisateur.

RESOLUTION

```
// Déclaration des bibliothèques
#include <stdio.h>
#include <conio.h>
//Création de la fonction maxi avec deux paramètres (paramètres formels)
int maxi(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) ;
}
main()
{
//Déclaration des variables globales
int x1, x2 ;
printf("Saisissez deux nombres entiers svp\n") ;
scanf("%d%d",&x1,&x2) ;
//Appel de la fonction avec des valeurs réelles lors de l'affichage
printf("La plus grande valeur de %d et %d est %d", x1, x2, maxi(x1, x2)) ;
getch() ;
}
```

RECAPITULATIF DU PROGRAMME SANS LES COMMENTAIRES

```
#include <stdio.h>
#include <conio.h>
int maxi(int a, int b)
```

```
{
int c ;
if(a>b)
{
c=a ;
}
else
{
c=b ;
}
return(c) ;
}
main()
{
int x1, x2 ;
printf("Saisissez deux nombres entiers svp\n") ;
scanf("%d%d",&x1,&x2) ;
printf("La plus grande valeur de %d et %d est %d", x1, x2, maxi(x1, x2)) ;
getch() ;
}
```

EVALUATION SUR LES FONCTIONS

II

Exercice 1

Soit le programme ci-dessous qui détermine la puissance d'un nombre entier à partir d'une fonction. La fonction prend deux paramètres, le premier est celui pour lequel nous devons déterminer la puissance et le second est la valeur de la puissance. Compléter les cases vides.

```
#include<stdio.h>
#include<conio.h>
int a,b,c ;
int puissance(int x, int y)
{
int i, n ;
n=1 ;
for(i=1 ;i<=y ;i++)
{
n=   *x ;
}
            ;
}
main()
{
printf("Entrez un nombre entier Svp !") ;
scanf("%d",&a) ;
printf("Saisissez la puissance !") ;
scanf("%d",&b) ;
c=           (a,b) ;
printf("%d à la puissance %d donne %d" ;a,b,c) ;
getch() ;
}
```

LES PROCEDURES

III

Définition

Contrairement à la fonction, la procédure ne retourne pas de valeur, elle effectue en son sein, les différents traitements et les affiche. Elle se caractérise par le mot clé *void* qui précède le nom du sous-programme lors de sa création. On note qu'il n'existe pas de *return* à la fin de la procédure.

CREATION D'UNE PROCEDURE

SYNTAXE :

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.

APPEL D'UNE PROCEDURE

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 :

void Nom_Procedure(valeur1,. . ., valeur n)

EXEMPLE DE PROCEDURE

Ecrivons un programme qui, à partir d'une Procédure retourne le double d'un nombre entier saisi par l'utilisateur.

RESOLUTION

// Déclaration des bibliothèques

#include <stdio.h>

#include <conio.h>

```
//Création de la Procédure double avec un seul paramètre (paramètre formel)
void double(int a)
{
//Déclaration d'une variable locale
int b ;
//définition du bloc d'instructions
b=a*2 ;
printf("Le double de %d est %d",a,b) ;
}
main()
{
//Déclaration des variables globales
int x ;
printf("Saisissez un nombre entier svp\n") ;
scanf("%d",&x) ;
//Appel de la procédure avec une valeur réelle
double(x) ;
getch() ;
}
```

RECAPITULATIF DU PROGRAMME SANS LES COMMENTAIRES

```
#include <stdio.h>
#include <conio.h>
void double(int a)
{
int b ;
b=a*2 ;
printf("Le double de %d est %d",a,b) ;
}
main()
{
int x ;
printf("Saisissez un nombre entier svp\n") ;
scanf("%d",&x) ;
double(x) ;
getch() ;
}
```

EVALUATION SUR LES PROCEDURES

IV

Exercice 1

Soit le programme ci-dessous qui à partir d'une procédure, affiche le périmètre d'un rectangle en fonction de sa longueur et sa largeur.

```
#include<stdio.h>
#include<conio.h>
float x, y;
        ()
{
float p ;
printf("Saisissez la longueur Svp !") ;
scanf("%f",&x) ;
printf("Saisissez la largeur Svp !") ;
scanf("%f",&y) ;
  =(x+y)*2 ;
printf("La longueur vaut %f \n",x) ;
printf("La largeur vaut %f \n",y) ;
printf("Le périmètre vaut    "  ) ;
}
main()
{
perimetre() ;
getch() ;
}
```