Résumé sur les sous-programmes

I. Introduction

Lorsque l'on progresse dans la conception d'un algorithme, ce dernier peut prendre une taille et une complexité croissante. De même des séquences d'instructions peuvent se répéter à plusieurs endroits.

Un algorithme écrit d'un seul tenant devient difficile à comprendre et à gérer dès qu'il dépasse deux pages. La solution consiste alors à découper l'algorithme en plusieurs parties plus petites. Ces parties sont appelées des sous-algorithmes.

Le sous-algorithme est écrit séparément du corps de l'algorithme principal et sera appelé par celui-ci quand ceci sera nécessaire.
Il existe deux sortes de sous-algorithmes : les procédures et les fonctions.

II. Les procédures

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 un algorithme ou dans un autre sous-algorithme.
Une procédure renvoie plusieurs valeurs (pas une) ou aucune valeur.

II.1. Déclaration d'une procédure

Syntaxe (Algorithme) :

```
Procédure <Nom_Procedure>(<P1>:<T1> ; <P2>:<T2>;… <Pn>:<Tn>);
<Déclaration>;
Début
  <instruction 1>;
  <instruction 2>;
  .......
  .......
  <instruction N>;
Fin;
```

<P1>, <P2>, … <Pn> : Les paramètres éventuels de la procédure.

<T1>, <T2>, …, <Tn> : Les types respectifs des paramètres de la procédure.

Syntaxe (en langage C) :

```
void  <Nom_Procedure>(<T1>  <P1>, <T2>  <P2>,… <Tn>  <Pn>)

    {
       <Déclaration>;

      <instruction 1>;
      <instruction 2>;
       .......
       .......
      <instruction N>;
    }
```

II.2. L'appel d'une procédure

Pour déclencher l'exécution d'une procédure dans un programme, il suffit de l'appeler.

L'appel de procédure s'écrit en mettant le nom de la procédure, puis la liste des paramètres, séparés par des virgules.

A l'appel d'une procédure, le programme interrompt son déroulement normal, exécute les instructions de la procédure, puis retourne au programme appelant et exécute l'instruction suivante.

Syntaxe :

Nom_proc(liste de paramètres)

Exemple :

Algorithme	Langage C

Procédure carre(a : entier ; **var** b : entier)
Début
b=a*a ;
Fin.
	void carre(int a, int *b)
	{
	*b=a*a;
	}

Un programme qui utilise la procédure précédente

```
#include<stdio.h>
void carre(int a, int *b)
{
        *b=a*a;
}
int main ()  //Début du programme principale
{
   int n, x;
   printf("Introduire n");
   scanf("%d",&n);
   carre(n, &x);
   printf("carre =%d \n", x);
   return 0;
} //Fin du programme principale
```

III. Les fonctions

Les fonctions sont des sous algorithmes admettent des paramètres et retournant un seul résultat (une seule valeur) de type simple qui peut apparaitre dans une expression, dans une comparaison, à la droite d'une affectation, etc.

III.1. Déclaration d'une fonction

Syntaxe (en algorithme) :

```
Fonction <Nom_Fonciton>(<P1>:<T1>; <P2>:<T2>;… <Pn:Tn) : <type_fonction>;
<Déclarations>; //Variables Locales
Début
  <instruction 1>;
  <instruction 2>;
  .......
  .......
  <instruction N>;
  retourner <resultat>;
Fin;
```

<P1>, <P2>, … <Pn> : les paramètres éventuels de la fonction.

<T1>, <T2>, …, <Tn> : les types respectifs des paramètres de la fonction.

<type_fonction> : type de la fonction. C'est-à-dire, le type de résultat de la fonction.

Syntaxe (en langage C) :

```
<type_fonction> <Nom_Fonction>(<T1> <p1>, <T2> <p2>, … , <Tn> <pN>)
{
  <Déclaration>;

  <instruction 1>;
  <instruction 2>;
  <instruction 3>;
  return <resultat>;
}
```

III.2. L'appel d'une fonction

Pour exécuter une fonction, il suffit de faire appel à elle en écrivant son nom suivi des paramètres effectifs. C'est la même syntaxe qu'une procédure.

A la différence d'une procédure, la fonction retourne une valeur. L'appel d'une fonction pourra donc être utilisé dans une instruction (affichage, affectation,…) qui utilise sa valeur.

Syntaxe :

Nom_Fonction(liste de paramètres)

Exemple :

Algorithme	Langage C

Fonction carre(a : entier): entier ;
Début
retourner a*a ;
Fin.
	int carre(int a)
	{
	return (a*a);
	}

Un programme qui utilise la fonction précédente

```
#include<stdio.h>
int carre(int a)
{
        return (a*a);
}
int main ()  //Début du programme principale
{
   int n, x;
   printf("Introduire n");
   scanf("%d",&n);
   x=carre(n);
   printf("carre =%d \n", x);
   return 0;
}  //Début du programme principale
```

Rappel :

Les *paramètres formels* sont les paramètres utilisés dans la déclaration des procédures et fonctions. Par contre, les *paramètres effectifs* sont les paramètres utilisés lors de l'appel aux procédures et fonctions.