Chapitre 5:Les sous-programmes

Leçon 1
L'analyse modulaire

I-Introduction:

Afin de faciliter la résolution d'un problème complexe et/ou de grande taille, on doit le décomposer en sous problèmes indépendants et de taille réduite.

II-L'analyse modulaire:

➔ ***1-Définition:***

L'analyse modulaire consiste à diviser un problème en sous problème de difficultés moindres. ces derniers sont aussi soumis à cette division jusqu'à ce qu'on arrive à un niveau abordable de difficulté.

➔ ***2-Intérêts :***

- ✔ Plus d'organisation en séparant les difficultés et les taches.
- ✔ S'occuper d'un seul problème à la fois.
- ✔ En cas d'erreur la division en module permet de savoir quel module à corriger
- ✔ Plus facile à faire évoluer.
- ✔ Permet d'éviter la répétition d'un même traitement dans un programme.

➔ ***Notion de sous-programme:***

C'est la décomposition modulaire d'un programme en plusieurs sous-programmes. Un sous programme est appelé aussi une procédure ou une fonction. C'est une portion de texte analogue à un programme, déclaré dans un programme ou dans un sous programme et dont la partie instruction peut être exécutée plusieurs fois au cours du traitement du programme grâce à des appels.

➔ ***Exemple d'analyse modulaire:***

étude de fonction mathématique.

Leçon 2
Les procédures

1-Définition:

Les procédures sont des sous-programmes qui peuvent avoir plusieurs résultats

2-Vocabulaire et syntaxe:

Déclaration:

En analyse:

DEF PROC nom(paramètres formels: type)		
S	L.D.E	O.U
	Résultat= **traitement** **Fin nom**	

En algorithme:

0) DEF PROC nom (paramètres formels: type)
1) Traitement
2) FinNom

en Pascal:

Procedure nom (paramètres formels: type) ;
Déclaration des variables locales;
Begin
Traitement;
End;

Appel de la procédure:

Proc nom_procédure (paramètres effectifs)

L'appel d'une procédure doit se trouver dans une instruction d'appel et ne peut pas être dans une expression comme c'est le cas d'une fonction

Remarque: Il est possible de définir un sous-programme sans paramètres. La communication avec l'appelant se produit grâce aux ressources (objets) communes partagées entre l'appelé et l'appelant.

Leçon 3
Déclaration, accès aux objets et mode de transmission

I-Déclaration et accès aux objets

1-Les objets locaux: tous les objets (constantes, types, variables et sous-programme) déclarés dans un sous-programme sont dits locaux à celui-ci.

2-Les objets globaux: les objets utilisés dans un sous-programme et non déclarés dans celui-ci sont des objets globaux déclarés ailleurs.

3-Accès aux objets: tous les objets locaux d'un sous-programme sont inaccessible: par le programme principal, par les sous-programmes déclarés au même niveau que le sous-programme considéré, par le sous programme qui englobent les sous-programmes considéré.

II- Les paramètres et leur mode de transmission:

On distingue deux types de paramètres:

1-Les paramètres formels: qui figurent dans la définition de la procédure.

2-Les paramètres effectifs: qui figures dans l'appel de la procédure et qui sont manipulés par celle-ci.

Remarque: Les paramètres formels et les paramètres effectifs doivent s'accorder de point de vue nombre et ordre et leurs types doivent être identique ou compatible, selon le mode de passage des paramètres.

3-Mode de passage des paramètres: il existe deux modes de passage des paramètres: le mode par valeur et le mode par variable. Pour le cas de la fonction, nous définissions seulement le mode par valeur.

➔ **Mode de passage par valeur:**

-Permet au programme appelant de transmettre une valeur au sous-programme appelé.
-Le transfert d'information est effectué dans un seul sens : du programme appelant vers le sous-programme appelé.
-Au moment de l'appel, la valeur du paramètre effectif est copiée dans la variable locale désignée par les paramètres formels correspondants.

Remarque : Toute modification du paramètre formel est sans conséquence sur le paramètre effectif

➔ **Mode de passage par variable:**

-Le passage de paramètres par variables permet au programme appelant de transmettre une valeur au sous-programme appelé et inversement.
-Dans l'entête de la procédure, on doit précéder les paramètres formels transmis par variable par le mot clé VAR.

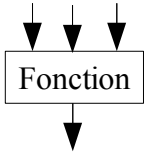
Remarque : Toute modification du paramètre formel entraîne automatiquement la modification de la valeur du paramètre effectif.

Leçon 4

Les fonctions

1-Définition:

Une fonction est un sous-programme qui renvoie une valeur de type **simple**, ce type sera le **type** de la fonction.



2-Syntaxe:

Déclaration d'une fonction:

En analyse:

DEFFN nom (paramètres formels: type): Résultat		
S	L.D.E	O.U
	Résultat= **Nom ← résultat calculé** Fin nom	

En algorithme:

0) DEFFN nom (paramètres formels: type) : Type_Resultat
1) Traitement
2) Nom ← résultat calculé
3) FinNom

En Pascal:

Function nom (paramètres formels: type) : Type_Resultat;
Déclaration des variables locales;
Begin
Traitement;
Nom:=RésultatCalculé;
End;

Appel de la fonction:

Analyse

Y←FN f(x) (x est un paramètre effectif)

TDO:

objet	type/nature	rôle
f	Fonction	f(x)=.....

Remarques:

- L'appel de la fonction se fait à l'aide de FN
- Le préfixe FN est ajouté devant le nom de la fonction que nous avons crée; ça nous aidera à nous rappeler qu'il faudra analyser.
- F est l'appelant.
- Dans l'analyse :

DEFFN nom f(X:entier):type_entier {x paramètre formel}

- X est déjà déclarés au niveau de l'entête de la fonction.
- F est une fonction ayant un seul paramètre formel x. Il est possible qu'une fonction ait plusieurs paramètres.
- Les variables déclarés dans la fonction sont appelés variables locales à la fonction f.
- Une fonction est constituée de trois partie:

1) ***La partie entête de la fonction*** ou nous trouvons son nom qui est suivit entre parenthèses des paramètres en entrée et de leur mode de passage, puis du type du résultat.

2) ***La partie déclaratives*** ou tous les objets locaux de la fonction sont déclarés.

3) ***La partie instruction*** ou nous trouvons les instructions propres à la fonction. Il est bon de rappeler que ces instructions sont exécutées à la demande de l'appelant par une instruction d'appel.

3-Transmission du résultat de la fonction:

- ✔ Une fonction possède un type, c'est celui du résultat quelle calcule.
- ✔ On appelle une fonction en utilisant simplement son nom suivi de la liste des paramètres effectifs séparé par des virgule (,)
- ✔ Un appel de fonction figure obligatoirement dans une expression sous forme suivante:

V ←FN nom de la fonction (liste des paramètres effectifs)

Exemple: y ← FN f(x)

- ✔ Il doit y avoir nécessairement dans la partie instruction de la fonction au moins une affectation explicite ayant l'identificateur de la fonction à gauche du symbole d'affectation.

4-Définition d'une fonction:

- ✔ Lors de l'utilisation d'une fonction, il faut:
- ✔ Spécifier le type de la fonction
- ✔ Déclarer, si nécessaire, une variable locale de même type que la fonction (pour faire les calcules intermédiaires)
- ✔ Affecter le résultat de calcul de la fonction au nom de la fonction, obligatoirement, avant la fin du bloc.

5-Mode de passage:

Pour le cas de la fonction, nous définissant seulement le mode de passage par valeur.

Remarques:

- Dans la partie instruction de la fonction, les paramètres formels transmis par valeur ne doivent en aucun cas se trouver à gauche du symbole d'une instruction d'affectation.
- Même si par erreur la valeur du paramètre formel transmis par valeur est modifié dans l'appelé au retour après exécution, les paramètres effectifs gardent la même valeur original transmise lors de l'appel.
- En effet, au niveau de l'appelé on manipule qu'une copie.
- Un identificateur peut cacher un autre. (un objet local à un sous programme a le même nom qu'un objet global (homonyme)).
- L'ordre des sous programmes est important, lorsque un sous programme fait appel à un autre, alors ce dernier doit être définit avant.