**SUITE ALGORITHMIQUE**

Table des matières

[CHAPITRE1 : PROCEDURES ET FONCTIONS 2](#_Toc91536564)

[I. Fonctions : 2](#_Toc91536565)

[II Procédures : 2](#_Toc91536566)

[CHAPITRE2 : STRUCTURE DE DONNEES 4](#_Toc91536567)

[I. Les tableaux 4](#_Toc91536568)

[II. Les structures : 5](#_Toc91536569)

[III. Les structures et tableau : 6](#_Toc91536570)

# 

# CHAPITRE1 : PROCEDURES ET FONCTIONS

## I. Fonctions :

Une fonction Est un sous-algorithme (un bloc de structures) qui rend l’algorithme plus lisible et plus compréhensible.

On divise pour mieux régner notre algorithme. (Mieux gérer en termes d’exactitude)

***Exemple*** dans la vie réelle pour construire une maison on a deux méthodes :

Méthodes *1* : on fait appel à une seule personne qui fera toutes les tâches (creuser, maçonnerie, peinture, plomberie, boiserie, etc.) => **un seul algorithme**

Méthode 2 : On fait appel à plusieurs personnes, chacune avec sa spécialité (plombier, menuisier, maçon, électricien, chauffagiste, peintre, etc.) => **plusieurs algorithmes**

Les deux méthodes sont justes sauf que la deuxième est plus rapide et plus exacte car chaque personne maîtrise un bloc de fonctions bien défini.

D’où l’utilité des fonctions et procédures en algorithmique

Une fonction est définie par :

-Un objectif bien défini

-Un nom pour pouvoir l’appeler dans l’algorithme quand on en a besoin

-des paramètres propres à elles qu’on ne voit pas ailleurs dans l’algorithme (paramètres locaux)

-Elle est caractérisée par un retour d’expression (avec résultat) pour communiquer avec l’extérieur et c’est ce qu’elle la diffère d’un la procédure

-elle n’affiche jamais de résultat à l’écran ( ses résultats sont utilisés dans l’algorithme)

## II Procédures :

Une procédure est une fonction qui ne fait pas de retour (pas d’instruction retourne) (sans résultat), est un bloc d’instructions nommé, déclaré dans l’algorithme et appelé à chaque fois que l’algorithme en a besoin, elle affiche un résultat à l’écran

***EXEMPLE DE FONCTION*** :

**-Cette fonction prend une variable somme qui est un entier et qui sort un entier comme résultat**

**-Ses paramètres sont x et y**

**Fonction** **somme** (x : entier, y : entier) **: entier** ;

**Var** : somme : entier

**Début**

Ecrire (**"**calcul de la somme en cours… **"**)

**somme** = x + y ;

Retourner **somme**

**FinFonction**

***EXEMPLE D’UNE PROCEDURE :***

**Procedure** **somme** (x : entier, y : entier) **: vide** ;

**Var** : somme : entier

**Début**

Ecrire (**"**calcul de la somme en cours… **"**)

**somme** = x + y ;

écrire (**"La somme des deux nombres est : " + somme)**

**FInProcédure**

**-Cette Procedure somme prend en paramètres deux entiers x et y calcule leur somme et affiche le résultat sur la console.**

**Le Résultat c’est vide (pas d’instruction retourne)**

CORPS De LA PROCEDURE

**DECLARATION DES PROCEDURES ET FONCTIONS :**

**APPEL DES PROCEDURES ET FONCTIONS :**

Quand on veut utiliser une fonction ou une procédure on doit les appeler dans leur corps

# CHAPITRE2 : STRUCTURE DE DONNEES

On utilise des structures de données pour traiter des grosses collections (ensemble d’éléments) de données

On doit structurer nos données pour qu’elles soient faciles à modifier et/ou à utiliser.

La réalité nous exige d’utiliser la complexité des données.

## I. Les tableaux

On l’utilise lorsque les données sont nombreuses et de même type,

SYNTAXE : **tabName : TYPE [ TAILLE]**

**EXEMPLE : Notes : ENTIER [10]**

**Notes : tabName, ENTIER : TYPE, 10 : TAILLE**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INDEX | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| valeur | 12 | 20 | 40 | 95 | 13 | 12 | 8 | 7 | 9 |

Le premier index est de 0 (toujours) c’est lié au langage informatique et à la structure de la mémoire l’ordinateur, le dernier index son rang est de (n-1), n c’est la taille du tableau, dans notre cas n=9 donc le dernier index est de 8. On écrit Tab[5] = 12,

On peut appliquer toutes les instructions qu’on a vues dans l’algorithmique au tableau : affectation, lecture, écriture.

Tab[8] = 10 modifie le contenu de la case N° 9 de 8 vers 10

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| INDEX | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| valeur | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Dans le tableau précédent on a initialisé toutes les valeurs vers 0 par la boucle suivante :

AlgoInitTableau

DEBUT

Pour i = de 0 à n-1 faire

Tab[i] = 0

FinPour

FIN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| T[0, 0] | T[0, 1] | T[0, 2] | T[0, 3] | T[0, 4] |
| T[1, 0] | T[1, 1] | T[1, 2] | T[1, 3] | T[1, 4] |
| T[2, 0] | T[2, 1] | T[2, 2] | T[2, 3] | T[2, 4] **200** |

On écrit Tab [2, 4] = **200**

C’est un tableau de 5 colonnes et 3 lignes.

## II. Les structures :

C’est comme un tableau sauf ici l’identification c’est des caractères alphanumériques.

C’est un moule qui sera le même pour toutes les variables d’une structure donnée.

**Struct Person** {

Nom : car (20) ;

Prenom : (20) ;

Age : Entier ;

20 représente le nombre de personnes avec 20 noms et 20 prénoms (chaque personne a un nom et un prénom

**Déclaration de variable :**

P1 : Person ;

P2 : Person ;

P3 : Person ;

**Utilisation :**

P1.Nom = **"MOUCHENE"**

P1.Prenom = **"Djamel"**

P1.Age = 45

## III. Les structures et tableau :

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | SEBIH | | Salim | | 34 | | |  | | --- | | HASSENA | | Hakim | | 30 | | |  | | --- | | BIZANDRY | | Fortuna | | 22 | |
|  |  |  |

Struct Person{

Nom : car(20) ;

Prenom : car (20)

Age : Entier ;

}

Persons : Person[20] ;

P1 : Person ;

P1.nom = **"SEBIH"**

P1.prenom=**"Salim" ;**

P1.age= 34

Persons : Person[20] ;

P2 : Person ;

P2.nom = **"BIZANDRY"**

P2.prenom=**"Fortuna" ;**

P2.age= 22

Persons [0] = p1 persons[1] = p2

0 représente le rang des trois informations (nom, prenom et age) de la personne N° 1 dans le tableau