**CHAPITRE 1**: Généralités de l’algorithmique

**Introduction**

L’informatique est définie comme étant la science du traitement automatique de l’information en utilisant un outil appelé ordinateur. L’ordinateur est un dispositif électronique capable d’effectuer des traitements rationnels de l’information, de sauvegarder les données et de les restituer en cas de besoin. L’ordinateur a deux composants : **le matériel et le logiciel** (Hard and Soft). **Le logiciel** permet de faire fonctionner le matériel (le système d’exploitation) et **les applications** permettent d’ajouter des fonctionnalités à l’ordinateur.

L’algorithmique est la discipline qui permet de concevoir des solutions informatiques qui peuvent être mis en œuvre à travers l’utilisation des langages de programmation. Toute solution algorithmique est appelé **un algorithme** et il permet de résoudre un problème bien défini.

1. **Structure d’un programme algorithmique**

En algorithmique, un programme est composé de trois parties :

* **La partie en-tête** : Elle permet de donner un nom au programme.
* **La partie déclarative** : Elle permet de déclarer les constantes, de définir les types, les prototypes, les modules et de déclarer les variables
* **La partie corps du programme principal :** Elle contient toutes les directives de base que le programme doit exécuter

Programme identificateur Partie entête

// DECLARATION DE CONSTANTE (S)

// DEFINITION DE TYPE, PROTOTYPE ET MODULES Partie déclarative

// DECLARATION DE VARIABLE

// DEBUT

// CORPS DU PROGRAMME PRINCIPAL Partie corps

// FIN

1. **Notion de types et d’identificateur**
2. **Notion de type**

L’algorithmique manipule des variables typées c’est-à-dire chaque variable est d’un type. Les types sont classés en deux catégories :

* **Les types élémentaires** encore appelés les types de base ou type primitif. Ils sont :
* **Entier**
* **Réel**
* **Caractère**
* **Chaine**
* **Booléen**

L’entier et le réel sont des numériques, l’entier n’a pas de virgules mais le réel en a. Toute valeur caractère doit être entre apostrophe, toute valeur chaine entre guillemets, un booléen ne peut avoir que deux valeurs **vrai ou faux.**

* **Les types composés** au chapitre 4

1. **Notion d’identificateur**

L’identificateur permet de repérer chaque entité du programme. Tout identificateur doit obéir aux règles suivantes :

* **Règle 1 :** Les composants de l’identificateur doivent être de type alphabétique ou alphanumérique.
* **Règle 2 :** L’identificateur ne doit jamais commencé par un chiffre.
* **Règle 3 :** Si l’identificateur est un mot composé alors ses composants ne doivent jamais être séparés ni par l’espace, ni par le tiret du 6 mais il est possible de les coller ou de les séparer par le tiret du 8.
* **Règle 4 :** L’identificateur ne doit pas être dans la liste des mots réservés et il est recommandé à ce que son nom soit en relation avec son contenu.

**REMARQUE :** Les identificateurs associés aux types de données ont permis la création des notions de constante et de variable.

1. **Notion de constante**

Une constante est un identificateur dont la valeur ne change pas.

**Syntaxe**

Const identificateur = valeur

**Exemples**

1. Const ecole = " ISM "
2. Cons anneeIndependance = 1960
3. Const TauxTVA = 0.18
4. **Notion de variable**

Une variable est un identificateur dont la valeur peut changer à tout instant.

**Syntaxe**

Var identificateur(s) : type

La syntaxe de déclaration est la suivante :

**Exemples**

Var nom : chaine

Var nom, prenom : chaine

Var prenom : chaine

1. Var age : entier
2. Var prix : reel
3. Var C : caractère
4. Var Test : booléen
5. **Les instructions d’entrée/sortie**
6. **Les instructions d’entrée**

Elles permettent d’entrer des données dans l’ordinateur en utilisant les mots clés.

Lire (identificateur)

Ou

Saisir (identificateur)

**Exemples**

**NB :** Dans ce cas, l’identificateur est toujours une variable

1. **Les instructions de sortie**

Elles permettent d’envoyer des flux de données vers les unités de sortie essentiellement l’écran en utilisant les mots clés :

Ecrire (expression)

Ou

Afficher (expression)

**Exemples**

1. Ecrire "les étudiants sont des génies"
2. Afficher ("Entrer
3. **Les opérateurs arithmétiques et logiques**
4. **Les opérateurs arithmétiques**

Une opération est l’association d’opérandes et d’opérateurs (opérande = valeur, opération = signe). Une opération est soit binaire soit unaire. Une opération unaire est composée d’une valeur et d’un signe par contre une opération binaire est composée de deux valeurs et d’un signe. Les opérations comme les opérateurs sont soit arithmétiques soit logiques. Les opérateurs arithmétiques de base qui existent en Maths sont également valables en informatique. Ces opérateurs sont résumés dans le tableau suivant :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Variables  Ou  Valeurs | Opérandes | | Opérateurs | | Exemples |
| Signe | Sens | Unaire | Binaire |
| + | Addition |  |  |  |
| - | Soustraction |  |  |  |
| ⁎ | Produit |  |  |  |
| Div | Division entière |  |  |  |
| / | Division réel |  |  |  |
| Mod | Reste de la division entière |  |  |  |
| Sqrt | Racine carrée |  |  |  |
| Sqr | carré |  |  |  |

1. **Les opérateurs logiques**

Ils sont associés à des variables logiques ou à des conditions composées. Les opérateurs logiques de base sont la négation, l’opérateur Et logique, l’opérateur Ou logique.

1. **L’opérateur de la négation**

La négation est encore appelée **opérateur de complémentation** ou **d’inversion.** Soit A, une variable logique, la négation de A est notée :

NON (A) = NOT (A) = A = ! A

|  |  |
| --- | --- |
| A | NON (A) = NOT (A) = A = ! A |
| FAUX | VRAI |
| VRAI | FAUX |

1. **L’opérateur Et logique**

**Et logique** est un opérateur binaire. Soit A et B deux variables logiques, A **Et logique** B est noté :

A **ET** B = A and B = A ⁎ B = A. B

Table de vérité de l’opérateur Et logique

|  |  |  |
| --- | --- | --- |
| A | B | A Et B = A and B = A ⁎ B= A. B |
| FAUX | FAUX | FAUX |
| FAUX | VRAI | FAUX |
| VRAI | FAUX | FAUX |
| VRAI | VRAI | VRAI |

Considérons juste que VRAI = 1, FAUX = 0 et  **Et = ⁎**

1. **L’opérateur Ou logique**

C’est un opérateur binaire. Soit A et B, des variables logiques, A **Ou logique** B est noté :

A **Ou** B = A or B = A + B

Table de vérité de l’opérateur **Ou** logique

|  |  |  |
| --- | --- | --- |
| A | B | A **Ou** B = A or B = A + B |
| FAUX | FAUX | FAUX |
| FAUX | VRAI | VRAI |
| VRAI | FAUX | VRAI |
| VRAI | VRAI | VRAI |

Considérons juste que VRAI = 1, FAUX = 0 et **Ou** = +

1. Les opérateurs de comparaison

Les opérateurs de comparaison de base qui existent en Mathématiques sont toujours valables en informatique. Ces opérateurs sont résumés dans le tableau :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Variables ou valeurs | | Opérateurs | | Exemples |
| A | B | Signes | Sens |
| < | Inférieur | A < B |
| ≤ ou <= | Inférieur ou égal | A ≤ B ; B ≤ B |
| = | Egalité | A = B |
|  |  |  |
|  |  |  |

1. **L’affectation-**

Elle consiste à donner une valeur à une variable. Son symbole est :

**Exercice d’application**

Ecrire un programme qui permet la saisie des informations relatives à une vente. Le programme affiche les données de la vente, calcule le montant sans remise, le montant avec remise et affiche les résultats. Une vente est caractérisée par : le code de la vente, le nom du produit vendu, le prix unitaire du produit, la quantité vendue, le nom du vendeur et la date de la vente.

**FORMULES**

* **Montant sans remise** = prix unitaire ⁎ qté vendue
* **Montant avec remise** = (montant sans remise) 1 – 20%

**Chapitre 2** : Les structures conditionnelles

Généralement dans un programme, l’exécution de certaines actions peut être liée à des conditions. Les structures conditionnelles permettent de tester les conditions et de donner des directives à exécuter au cas où ces conditions sont soit vérifiées soit non vérifiées. Il existe trois (3) structures conditionnelles :

* **La structure conditionnelle simple**
* **La structure alternative**
* **La structure à choix multiple**

**Ⅰ. La structure conditionnelle simple**

Elle permet de tester une condition et de donner les actions à exécuter au cas où la condition est vérifiée.

***SYNTAXE :***

Si (condition) Alors

<action(s)>

Fin Si

**Ⅱ. La structure alternative**

1. **Contexte d’utilisation**

Elle permet de tester une condition et de définir toutes les actions à exécuter au cas où la condition aurait pris une valeur quelconque. Toutes les alternatives sont prises en compte. Si la condition est vérifiée, les actions du "Alors" sont exécutées et si elle n’est pas vérifiée, les actions du "Sinon" sont exécutées.

***SYNTAXE :***

Si (Condition) Alors

<Actions(s)>

Sinon

<Actions B, C(s)>

Fin Si

**Ⅲ. La structure à choix multiple**

Elle propose une liste d’options possibles et vous laisse le choix à l’utilisateur de faire son choix. Les actions à exécuter dépendent de la valeur choisie. Elle est matérialisée en algorithmique par le mot clé suivant :

**Chapitre** 3 : Les structures répétitives

Dans un programme, il est fréquent de répéter l’exécution d’un ou de plusieurs traitements. Pour ce faire, il faut utiliser les structures itératives encore appelées structures répétitives ou boucle. Les boucles sont classées en trois catégories :

* La boucle "**pour"**
* La boucle "**répétée"**
* La boucle "**tant que"**

**I) La boucle « pour »**

**1) Contexte d’utilisation**

La boucle **pour**  doit être utilisée que si on connait le nombre d’itérations c’est-à-dire il faut le nombre de fois que les actions de la boucle doivent être exécutées.

**SYNTAXE**

Chapitre : Les tableaux d’enregistrements

Définition

Un tableau d’enregistrement est un tableau dans lequel chaque cellule contient un enregistrement. Chaque cellule d’un tableau d’enregistrement est décomposée en sous-cellules et le nombre de sous-cellules est égal au nombre de champs de l’enregistrement.

Un tableau d’enregistrement peut symboliser un vecteur d’enregistrement ou une matrice d’enregistrement.

**SYNTAXE**

**Type** NomEnregistrement = Structure

Début

Champ (s) : type (s)

Fin

**Type** NomTableauEnregistrement = Tableau [1…Taille]