الجمهورية التونسية
وزارة التعليم العالي والبحث العلمي
جامعة قفصة
المعهد العالي لإدارة المؤسسات

1ère année Licence en informatique de gestion

« Business Computing »

Matière

Algorithmique et Structure de Données I

Enseignant

Ali ZIDANII

ali.zidanii@gmail.com

ali.zidani@isaeg.u-gafsa.tn

Site web : http://ali-zidanii.e-monsite.com

2024 - 2025

Plan

- Les structures de données et les structures simples
- Les structures conditionnelles
- Les structures itératives
- Les tableaux et les enregistrements
- Les fonctions et les procédures
- La récursivité
- Les pointeurs

Les structures de données et les structures simples

Définition d'un algorithme

Un algorithme est une suite structurée et finie d'actions (ou d'instructions ou d'opérations) pour résoudre un problème donné.

Ces opérations regroupent :

Les opérations arithmétiques et logiques;

L'opération de lecture des données de la mémoire;

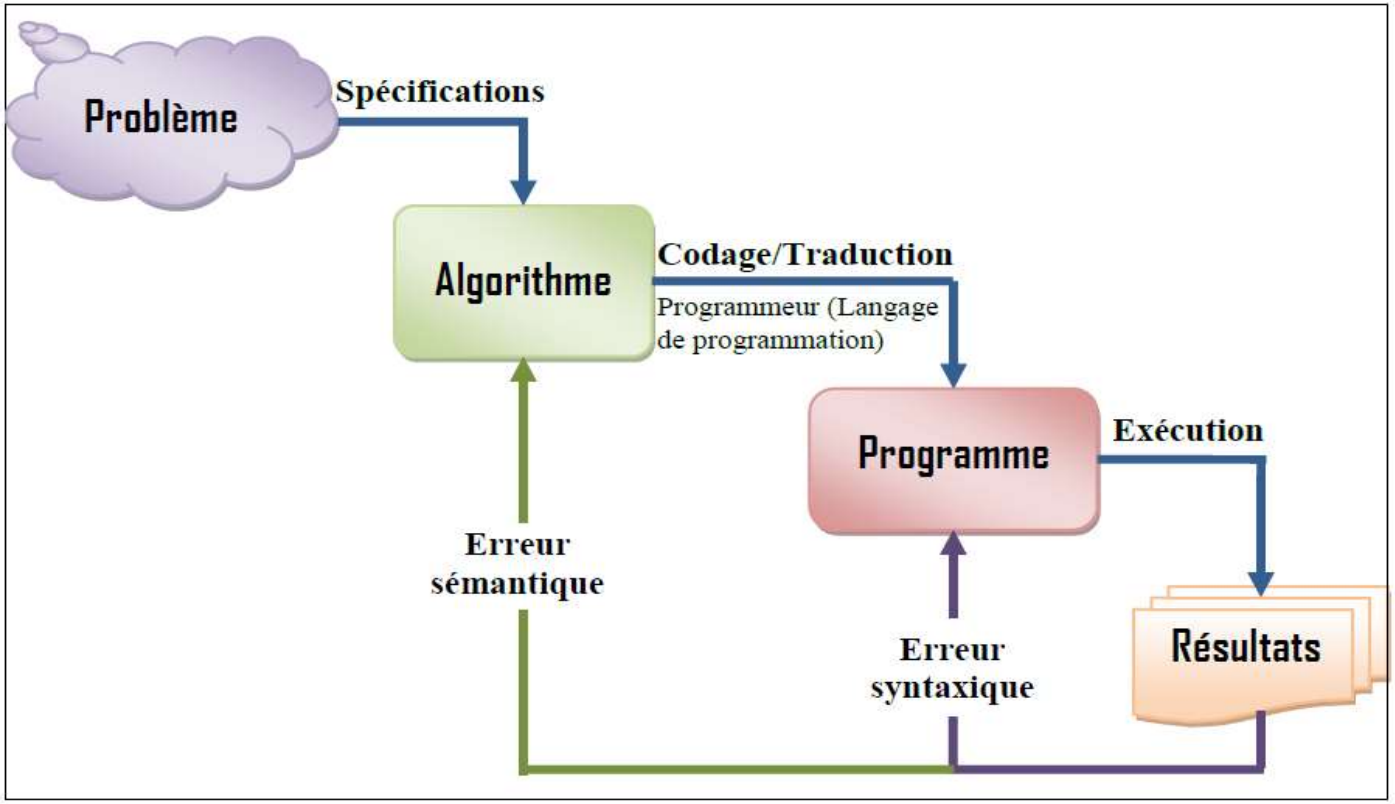
L'opération d'écriture des données dans la mémoire;

L'opération de comparaison;

La répétition d'une ou plusieurs opérations plusieurs fois ou un nombre fini de fois.

Les structures de données

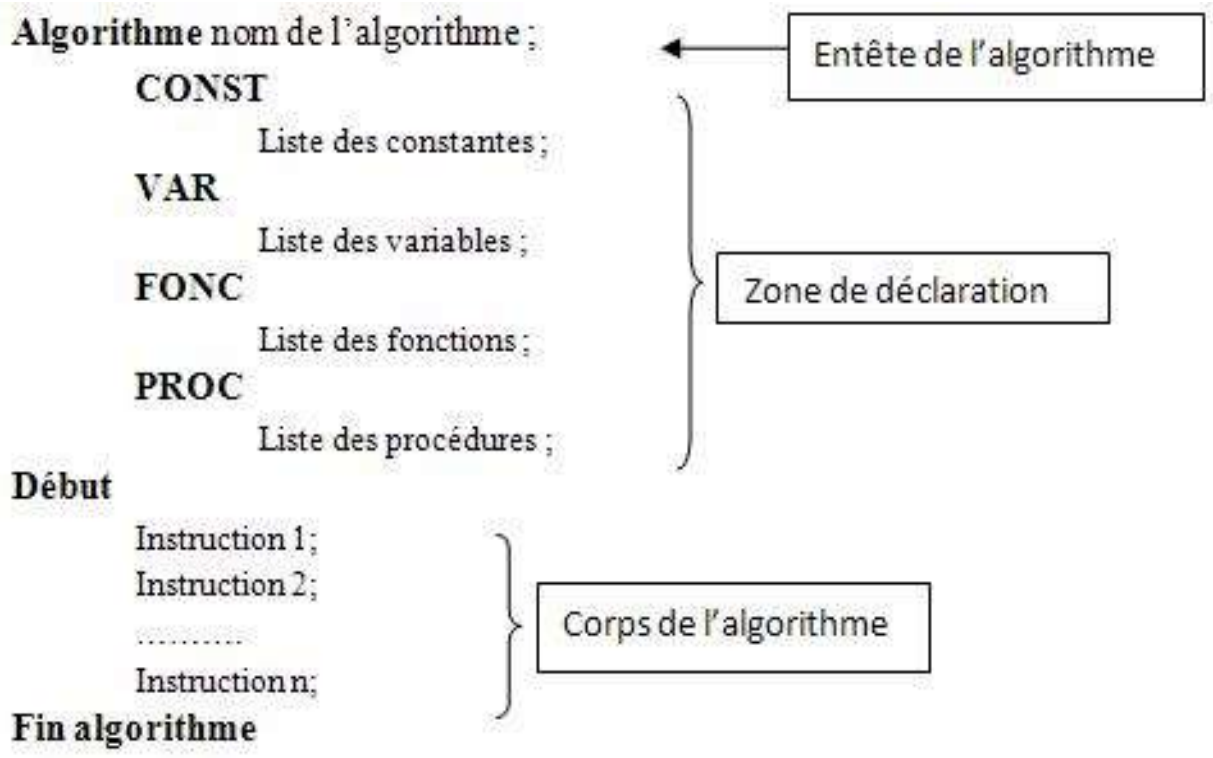
- **Cycle de vie d'un programme**



Les structures de données

Structure générale d'un algorithme

Il peut être présenté sous la structure suivante :



Les structures de données

Dont :

- **L'en-tête**, permet d'identifier l'algorithme (nom facultatif).
- **Les déclarations**, c'est une liste des objets utilisés et manipulés dans le corps de l'algorithme ; ***cette liste est placée en début d'algorithme***.
- **Le corps**, contient les tâches (instructions, opérations…) à exécuter. Une instruction peut être une structure simple (exemple : affectation ou action d'écriture/lecture) ou structure conditionnelle (exemple : expression de comparaison) ou structure répétitive.

Remarques :

- ❑ Un algorithme est écrit en utilisant un langage de description d'algorithme (LDA).
- ❑ L'algorithme ne doit pas être confondu avec le programme.
- ❑ Tout algorithme peut contenir des commentaires pour mieux l'interpréter.

Les structures de données

Déclaration des constantes

Elles représentent des chiffres, des nombres, des caractères, des chaînes de caractères, …etc. dont la valeur ne peut pas être modifiée au cours de l'exécution de l'algorithme.

Mot clé : CONST

Exemple : **CONST** pi =3.14 ; coefalgo=3; creditalgo=6;

Déclaration des variables

Elles peuvent stocker des chiffres des nombres, des caractères, des chaînes de caractères,… dont la valeur peut être modifiée au cours de l'exécution de l'algorithme.

Mot clé : VAR

Exemples : **VAR** x, y : entier;

ch : chaîne de caractère ;

Les structures de données

Remarques

Les constantes et les variables sont définies dans la partie déclarative par les caractéristiques suivantes :

- **L'identificateur** : c'est le nom de la variable ou de la constante. Il est composé de lettres et de chiffres
- **Le type** : il détermine la nature de la variable ou de la constante (entier, réel, booléen, chaîne de caractères…)
- **Le contenu ou la valeur**: qui doit être compatible au type de la constante ou la variable.

Les structures de données

Les types de données

Un type de données est l'ensemble des valeurs que peuvent prendre une variable, une constante, une expression, ou qui peuvent être générés par une fonction.

Toute variable qui apparaît dans un programme doit être associée à un et un seul type par une déclaration.

- ***Les types de base:***

L'entier (1, 7, 999),

Le réel (3.14),

Le booléen (VRAI ou FAUX),

Le caractère (symbole, lettre alphabetique , etc ..),

La chaîne de caractères ou chaîne (« bonjour »).

Les structures de données

▶ *Les types énumérés (ensemble)*

Les types énumérés sont définis par le programmeur. Ils vous permettent de créer vos propres types, qui consistent en un ensemble de symboles.

Vous créez d'abord l'ensemble de symboles et leur donner un nouveau nom de type.

Exemple:

Type *jour = (Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche);*

Var jr : jour;

Le premier symbole de cet ensemble a la valeur 0, et chaque symbole successif est plus grand de un.

NB. Les valeurs du type énuméré ne peuvent pas être introduites à partir du clavier ou affichées à l'écran.

Les structures de données

- ***Les types intervalles***

Un type peut-être défini comme un intervalle de n'importe quel type scalaire défini ou prédéfini.

```
Type undouze = 1..12;

Var mois : undouze;
```

Les structures de données

Exemple :

Tout comme vous pouvez créer votre propre ensemble de type prédéfini, vous pouvez aussi créer un plus petit sous-ensemble ou intervalle d'un ensemble existant qui a été défini précédemment. Chaque intervalle est défini par une borne supérieure et une borne inférieure.

Type jour = (*Lundi, Mardi, Mercredi, Jeudi, Vendredi, Samedi, Dimanche)*
Type Weekday = *Lundi..Vendredi* ; /*intervalle de jour*/

Type Weekend = *Samedi.. Dimanche* ; /*intervalle de jour*/

Type Heures = 0..24; /*intervalle d'entier*/

Type Lettre = 'A'..'Z'; /*intervalle de caractères*/

Les structures de données

- **Application :**

Lesquels de ces exemples sont corrects ?

Type Points = 0,5..4,1;

Type Number = entier;

Type Alphabet = 'Z'..'A';

Réponses

AUCUN n'est correct !!

On ne peut pas avoir d'intervalle de type réel.

On ne peut pas le faire, on doit avoir des bornes numériques.

On ne peut pas le faire, on doit écrire Alphabet= 'A'..'Z' parce que 'A' vient avant 'Z'.

Les structures de données

- **Les opérateurs :**
- **Opérateurs sur les entiers et les réels**

Arithmétiques		Comparaisons	
+	Addition	>	Supérieur
-	Soustraction	<	Inférieur
*	Multiplication	≥	Supérieur ou égal
/	Division	≤	Inférieur ou égal
DIV	Division entière	=	Egal
MOD	Reste de la division entière de deux nbres	≠	différent
^	Puissance		

Les structures de données

- **Les opérateurs :**
- **Opérateurs logiques (sur les entiers et les booléens)**

Fonctions logiques	
Et (AND)	Fonction ET
Ou (OR)	Fonction OU
OuEx (XOR)	Fonction OU exclusif
Non (NOT)	Fonction NON
NonEt	Fonction NON ET
NonOu	Fonction NON OU
>>	Décalage à droite
<<	Décalage à gauche

Les structures de données

Opérateurs sur les caractères et les chaines de caractères

+ : Concaténation ;
= : Egalité ;
≠ : Différent ;
> : Supérieur ;
< : Inférieur

Exemple: 'alpha' + 'numérique' donne après concaténation : 'alphanumérique'

Priorité des opérateurs :

La priorité des différents opérateurs, de la plus élevée à la plus basse, est :

NON
*, /, DIV, MOD, ET
+, -, OU
=, != <>, <, <=, >, >=, DANS OU APPARTIENT (IN)

Remarque :

Au sein d'une même priorité, les opérateurs sont toujours évalués de gauche à droite. Utiliser les parenthèses pour éviter le problème de priorité des opérateurs.

Les structures simples

On distingue trois structures simples:

- L'affectation
- La lecture
- L'écriture

a-

```
Algorithme a1
Var A, B, C : Entier
Début
```

Les structures simples

Exemples :

- L'affectation
- La lecture
- L'écriture

a- Affectation

```
B ← 5;
X ← "g";
F ← 2.5;
```

a- Lecture

```
Lire(B);
Lire(X);
Lire(F);
```

a- Écriture

```
Ecrire(" Donner un entier ");
Ecrire("la valeur de X=" ,X);
Ecrire(" F=", F );
```

Les structures de données et les structures simples

- **Applications:**

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes?

a-

```
Algorithme a1
Var A, B, C : Entier
Début
A ← 5; écrire(A);
B ← 3; écrire(B);
C ← A + B ; écrire(C);
A ← 2 ; écrire(A);
C ← B – A ; écrire(C);
Fin
```

b-

```
Algorithme a2
Variables A, B, C en Entier
Début
A ← 3 ; écrire(A);
B ← 10 ; écrire(B);
C ← A + B ; écrire(C);
B ← A + B ; écrire(B);
A ← C ; écrire(A);
Fin
```

Plan

- Les structures de données
- Les structures conditionnelles
- Les structures itératives
- Les tableaux et les enregistrements
- Les fonctions et les procédures
- La récursivité
- Les pointeurs

Les structures conditionnelles

Les structures algorithmiques fondamentales

Les opérations élémentaires relatives à la résolution d'un problème peuvent, en fonction de leur enchaînement être organisées suivant quatre familles de structures algorithmiques fondamentales.

* structures linéaires
* structures alternatives (Conditionnelle)
* structures de choix
* structures itératives (ou répétitives)

- **Les structures linéaires**

La structure linéaire se caractérise par une suite d'actions à exécuter successivement dans l'ordre énoncé.

Ecrire ("c'est un exemple") ;

Les structures conditionnelles

- **Les structures alternatives ou conditionnelles**

La structure alternative n'offre que deux issues possibles à la poursuite de l'algorithme et s'excluant mutuellement.

On peut rencontrer deux types de structures alternatives :

- ***Structure alternative réduite***

La structure alternative réduite se distingue de la précédente par le fait que seule la situation correspondant à la validation de la condition entraîne l'exécution du traitement, l'autre situation conduisant systématiquement à la sortie de la structure.

- **Notation :**

  SI condition **alors**

  Action ;

  Fin si ;

Les structures conditionnelles

- ***Structure alternative complète***

Dans cette structure l'exécution d'un des deux traitements distincts ne dépend que du résultat d'un test effectué sur la condition qui peut être une variable ou un événement ;

Si la condition est vérifiée seul le premier traitement est exécuté ;

Si la condition n'est pas vérifiée seul est effectué le second traitement.

- **Notation :**

SI condition **alors**

action1 ;

Sinon

action2 ;

Fin si ;

Les structures conditionnelles

- ***Structure alternative généralisée***
- **Notation :**

SI condition1 **alors**

action1 ;

Sinon si condition2

action2 ;

……

Sinon

Action3;

Fin si ;

Les structures conditionnelles

- **Les structures de choix**

La structure de choix permet, en fonction de plusieurs conditions de type booléen, d'effectuer des actions différentes suivant les valeurs que peut prendre une même variable. **RQ: Sinon (default) est facultatif**

- **Notation :**

selon valeur **faire**
  valeur1 : action1 ;
  valeur2 : action2 ;
  ……..
  valeur_N : action_N ;
  sinon action_N+1 ;
Fin selon ;

en C

```
switch (valeur){
case valeur1 : action1; break;
case valeur2 : action2; break;
…….
case valeur_n : action_n; break;
default action_n+1;
}
```