Faculté de Sciences de la matière
Université de Djilali Bounaama Khemis Miliana

Introduction à l'algorithmique et la programmation

CHERIEF Houria
houria.cherief1@univ-dbkm.dz

January 7, 2023

Content

Introduction

Notion d'algorithmique
- Définitions
- Structure d'un algorithme et un programme
- Nom d'algorithme et d'un programme
- Notion de variables et les types des données

Les instruction
- L'affectation des variables
- L'instruction de la lecture et de l'écriture
- Instructions conditionnelle : test
- Les instructions itératives : les boucles

Les tableaux : vecteurs et matrices
- Définition
- Déclaration des tableaux
- Remplissage d'un tableaux
- Exercice

Introduction

- Pourquoi apprendre l'algorithmique pour apprendre à programmer ?

Parce que l'algorithmique exprime les instructions résolvant un problème donné indépendamment des particularités de tel ou tel langage. Apprendre l'algorithmique, c'est apprendre à manier la structure logique d'un programme informatique. Cette dimension est présente quelle que soit le langage de programmation ; mais lorsqu'on programme dans un langage on doit en plus se colleter les problèmes de syntaxe, ou de types d'instructions, propres à ce langage. Apprendre l'algorithmique de manière séparée, c'est donc sérier les difficultés pour mieux les vaincre.

Notion d'algorithmique

Définitions

- **Algorithme :** est un terme d'origine arabe, composé d'une suite de raisonnement et des instructions élémentaires,qui en une fois exécutée correctement conduit à une résultats de certains problèmes.
- **Représentation graphique ou Organigramme :** permet une lecture aisée des algorithmes. Les opérations dans un organigramme sont représentés par les symboles dont les formes sont normalisées. Ces symboles sont reliées entre eux par des ligne flèchées qui indiquent les chemins, la figure suivante représente ces symboles.

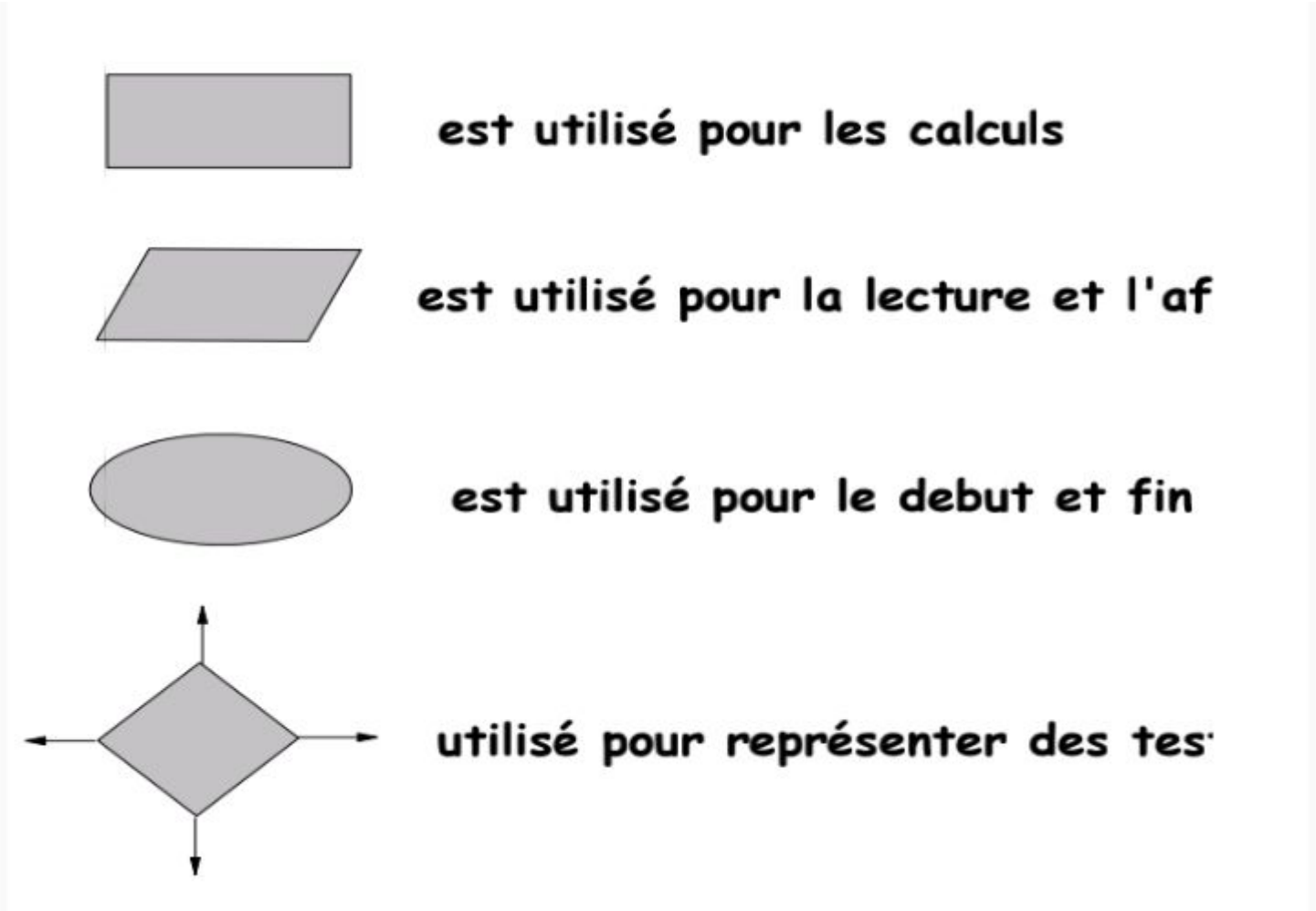


Figure: Les symboles utilisés dans l'organigramme

Notion d'algorithmique

Définitions

- **Programme :** est un assemblage et enchaînement d'instruction élémentaire écrits dans un langage de programmation et exécuté par un ordinateur afin de traiter les données d'un problème et revoyer des résultats.
- **Langage de programmation :** fournit un ensemble de mots-clés et de régles de syntaxe qui peuvent s'exécuter sans soucis, sur une machine. Il existe plusieur langage de programmation tel que : C,C++, Python, Java, Fortran... etc.
  (Ps : le langage Fortran est utilisé dans ce cours).
- **Fortran :** est un langage de programmation développé par IBM vers 1955 est destiné à fournir aus scientifiques un moyen simple pour passer de formules mathématiques jusqu'un programme.
  - son nom est un abbreviation de FORmula TRANslator.
  - Fortran fait objet de plusieurs normalisation : Fortran 77, 90, et 95 et plus récemment Fortran 2003.
  - Le fichier Fortran doit posséder un suffixe ou bien extension *nom.f90*.

Format générale d'un algorithme :

Algorithme Nom_Algorithme;
Variables
Constantes
Début
.
.
Instructions;
.
.
Fin

Format générale d'un programme :

Program Nom_Program
implicit none
Déclaration des variables
.
.
Instructions
.
.
End

Chaque algorithme et un programme doit avoir un nom qui permet son identification.

- Le nom ne doit pas contient un espace.
- Le nom ne doit pas contient des symboles (&#, %...).
- Le nom ne doit pas contient des mot connais dans un programme comme : nom de fonction, nom de boucle...

Exercice : Citer les nom non valid pour un algorithme et un program :

fonction_Log, Volume Pyramide, SurfaceCercle, TempératureConvert, Suite#, NomPrenom.

Variable : désigne un emplacement mémoire dont le contenue peut changer ou cours d'un programme.

- Variable doit être déclarée avant d'être utilisé.
- doit être caractérisée par un nom. **Identificateur** Suivit par le type.

variables

ident : Type;

Exemple:

Variables

A, B : Entier;
val, H : Réel;
test : Booléen;

Constantes : comme une variables, il a un type mais la valeur d'une constante ne varie pas.

Constantes

ident = valeur;

Exemple:

Constantes

A = 55;

pi = 3.14;

Exercice :

Quelle sont les identificateur non valide pamis cette list.

G20, 13C, H, Ener_photon, hbar, pi, TTC-2023, while, read,log, com, mod, hauteur, h#.

Les variables et constantes peuvent avoir les types suivant :

déclaration	signification	ex. d'affectation
`integer :: n,i`	entier	`n=10`
`real :: x,y`	réel	`x=1.0e-5`
		`y=1.0`
`complex :: z`	complexe	`z=cmplx(1.0,2.5e3)`
`character :: c`	caractère	`c='d'`
`logical :: b`	booléen	`b=.true.`

Figure: Les types des données

Les instruction

Un algorithme possède à quatre types d'instruction considérer comme des briques de base pour résoudre un problème :

- L'affection des variables.
- La lecture et l'écriture.
- Les structures conditionnelles (tests).
- Les structures itératives (Les boucles).

Affectation :

Est une opération qui consiste à stocker une valeur dans un variables. Cette opération se fait à l'aide de la syntaxe suivant :

$$nom_variable \leftarrow valeur;$$

Cette syntaxe signifie à mettre la valeur dans la case identifier par nom_variable.

Exemple :

$A \leftarrow 5;$
$X \leftarrow A * B * *2;$
$Montion \leftarrow$'Tres Bien';

Structure de lecture (Entrée) :

L'entrée ou la lecture des données correspond à l'opération qui permet de saisir des valeur pour qu'elles soient utiliser par le program "Lire identificateur".

Exemple :

Lire(h, n);
Lire(nom);

Structure de l'écriture (Sortie) :

La sortie ou l'écriture des données permet l'affichage des valeurs après traitemen. Cette instruction est noté "afficher identificateur".

Exemple :

Écrire('h');
Écrire('h=', h);
Écrire('Bounjour tout le monde');

Les instructions conditionnelle ne permettant d'exécuter des instructions que sous certaines conditions.

Condion : est une expression logique ou conditionnelle.

- Condition doit évaluer est ce qu'elle VRAI ou FAUX.
  - Si elle VRAI, un traitement est réalisé.
  - Si la condition est fausse, une autre instruction va être exécutée, puis le programme va continuer normalement.

Structure conditionnelle simple :

La syntaxe de la conditionnelle simple écrit dans l'algorithme comme suit :

si<condition> **alors**
  Instructions;
Fin si

Structure alternative :

cette instruction incorpore deux blocs d'instruction dont un et un seule bloc exécuté, de manière alternative en fonction de la condition. La syntaxe de conditionnelle alternative prend cette forme :

si<condition> **alors**
  traitement 1;
Sinon
  traitement 2;
Fin si

En Fortran les structures conditionnelle sont écrits dans les formes suivantes :

if (condition) **then**
  treatment 1
Else
  treatment 2
End if

Structures conditionnelles imbriquées :

Le test imbriquées est une généralisation de la structure de contrôle conditionnelle, lorsque le nombre de traitement différents est plus grand que deux.

L'algorithme actuelle présente la syntaxe de plusieurs conditions imbriquées.

```
si<condition1> alors
   si<condition2> alors
      traitement 1;
   Sinon
      traitement 2;
   Fin si
Sinon
   si<condition3> alors
      traitement 3;
   Fin si
Fin si
```

La structure de choix multiples :

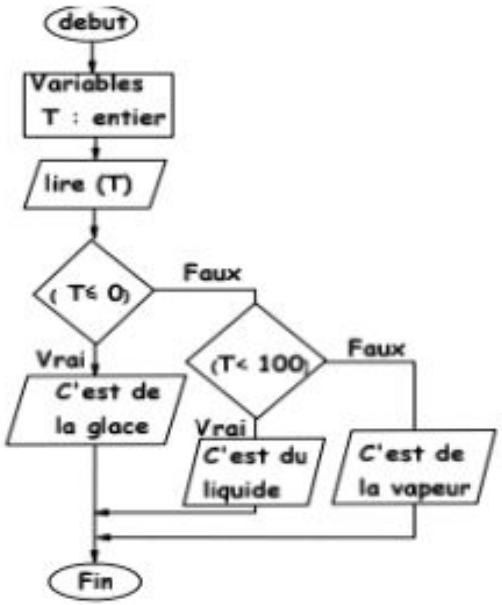
La structure de choix permet (en fonction de plusieurs conditions de type booléen) d'effectuer des actions suivant les valeurs d'une seule variables.
La syntaxe s'est comme suit :

Selon que
<condition 1>:<action 1>;
<condition 2>:<action 2>;
.
.
.
<condition n>:<action n>;
Sinon:<action sinon>;
Finsi

Exercice :

À l'aide d'organigramme et programme Fortran, écrire l'algorithme qui demande à l'utilisateur la température d'eau et affiche son état (solid, liquid, vapeur).

Organigramme :



Programme Fortran :

Programme fortran :

```
program températureH2O
INTEGER T
write(*,*)' entrer la valeur de T '
read(*,*) T
IF (T.LE.0) then
write(*,*)' C"est de la glace '
ELSE
      IF (T.LT.100) then
       write(*,*)' C"est du liquide '
      ELSE
       write(*,*)' C"est de la vapeur '
END IF
END
```

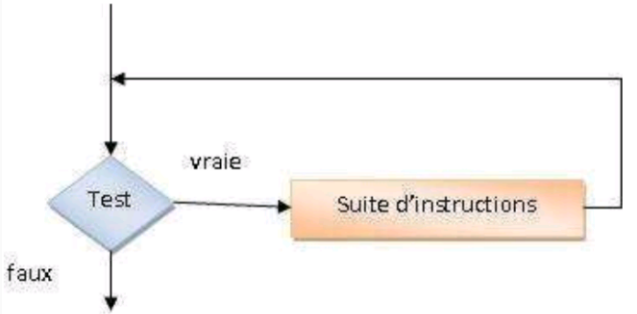
Une boucle permet de répéter une instruction plusieurs fois. Il y a principalement trois types de boucles :

Boucle Tant que :

Le nombre d'itération d'un boucle **Tant que** n'est pas connu au moment d'entrée, il dépend la condition.
Syntaxe de boucle Tant que prend la forme suivante :

Tant que (condition) faire :
    instructions;
Fin Tant que

La représentation graphique illustré dans la figure suivante :



Organigramme de l'instruction tant que

Les instructions itératives
boucle Tant que

En Fortran la boucle Tant que s'écrit de la forme suivante :

dowhile (condition)
    instructions
End do

Exemple :
comment afficher le message 'Bonjour à tous' 5 fois.
Algorithme Affichage;
    Variables
     i : entier;
    Début
     $i \leftarrow 1$;
     Tant que(i $<=$ 5) faire :
      Écrire ('Bonjour à tous ');
      $i \leftarrow i + 1$;
     Fin Tant que
Fin

Les instructions itératives
boucle Pour

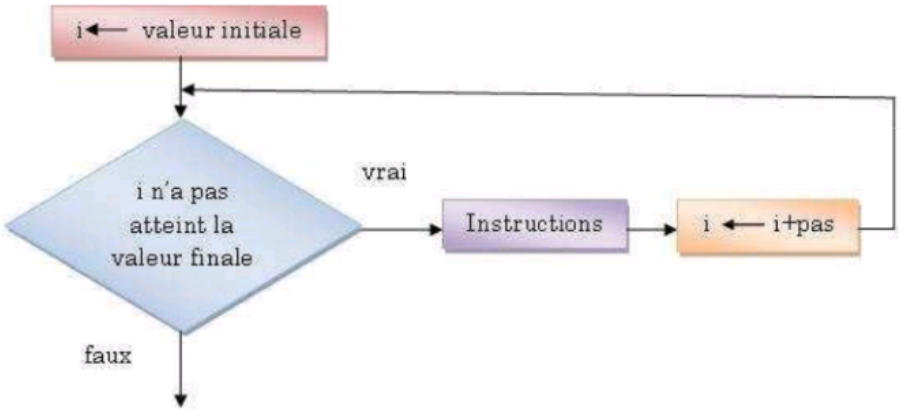
Boucle pour :

permettre de répéter une instruction un nombre donné de fois .

Syntaxe de boucle pour prend la forme suivante :

pour allant de valeur initiale à valeur final par du pas faire :
instructions;
Fin pour

La représentation graphique illustré dans la figure suivante :



La structure itératives
Boucle Pour

En Fortran la boucle pour s'écrit de la forme suivante :

do i = valeur initiale, valeur final, pas
instructions
End do

Exemple :
comment afficher le message 'Bonjour à tous' 5 fois.

Algorithme Affichage;
Variables
i : entier;
Début
pour i allant 1 à 5 par 1 faire :
Écrire ('Bonjour à tous ');
Fin pour
Fin

La structure itératives
Boucle Répéter

La boucle Répéter s'écrit de la forme suivante :

Répéter
    instructions;
Jusqu'à(condition)

Exemple :
comment afficher le message 'Bonjour à tous' 5 fois.

Algorithme Affichage;
  Variables
   i : entier;
  Début
   $i \leftarrow 1$;
   Répéter
    Écrire ('Bonjour à tous ');
    $i \leftarrow i + 1$;
   jusqu'à(i>5)
Fin

Définition :

est une structure de données **T** qui permet de stocker un certain nombre d'éléments $T[i]$ repérer par indice **i** les tableaux vérifient généralement les priorités suivant :

- Tous les éléments ont le même type de base.
- Le nombre d'éléments stockés est fixée.
- Utilisation d'un tableau nous offrons un intérêt gain de temps, rétrécissement du volume d'algorithme.

Déclaration des tableaux :

La déclaration des tableaux prend la forme suivant :

- **real** A(10) $->$ Tableaux réel à un dimension de 10 éléments.
- **Integer** X(0:3) $->$ un tableau entier à 1 dimensions de 4 éléments.
- **Dimension** Z(7,3) $->$ un tableau réel de 2 dimensions de 7 lignes et 3 colonnes.
- **real** C(3,7,6) $->$ un tableau réel à 3 dimensions.

Exercice

Écrire un algorithme et un programme qui déclare et remplir un tableau de 10 éléments mettant tous à zero.

Algorithme Remplissage;
Tableau A(10) : réel;
Variable i : entier;
Début
pour allant 1 à 10
A(i) $\leftarrow$ 0;
fin pour
pour allant 1 à 10
Écrire ('les éléments de tableaux sont', A(i));
fin pour
Fin

```
Program Remplissage;
implicite none
Dimension A(10) : réel;
Integer i
do i = 1, 10
A(i) = 0
end do
do i = 1, 10
write(*,*) "les elements de tableaux sont", A(i)
end do
End
```

Les Tableau
Exercice

Écrire un algorithme qui calcul la moyenne de 30 étudiants.

algorithme Moyenne;

Constante

$N \leftarrow 30$;

Variables

moy : Tableau[1...N] de réel;

Début

pour i de 1 à N faire

Écrire ('donner la moyenne de l'étudiant', i);

Lire(moy[i]);

Fin faire

$S \leftarrow 0$;

pour i de 1 à N faire

$S \leftarrow S + moy[i]$;

Fin faire

$S \leftarrow S/30$;

Écrire ('La moyenne du groupe est', S);

Fin

Questions ?