LES BASES DE L'ALGORITHMIQUE

DEVELOPPEMENT DIGITAL

Partie 1: L'algorithmique
Partie 2: Langage Python

IDRISSI KHAMLICHI SAFAA

NTIC 2023-2024

LES BASES DE L'ALGORITHMIQUE

Partie 1:
L'algorithmique

IDRISSI KHAMLICHI SAFAA

NTIC 2023-2024

INTRODUCTION

Le rôle essentiel de l'ordinateur est de traiter de l'information. D'où le terme informatique généralement utilisé pour désigner l'ensemble des techniques permettant d'automatiser un tel traitement. L'informatique, en tant que discipline scientifique, se révèle de nos jours un outil indispensable à la résolution de problème et s'applique à des domaines aussi variés que la gestion, les calculs scientifiques, l'ingénierie, le diagnostic médical, etc.

Informatiser un problème, résoudre une équation du deuxième degré, par exemple, c'est faire réaliser par ordinateur, une tâche qui était réalisée par l'Homme.

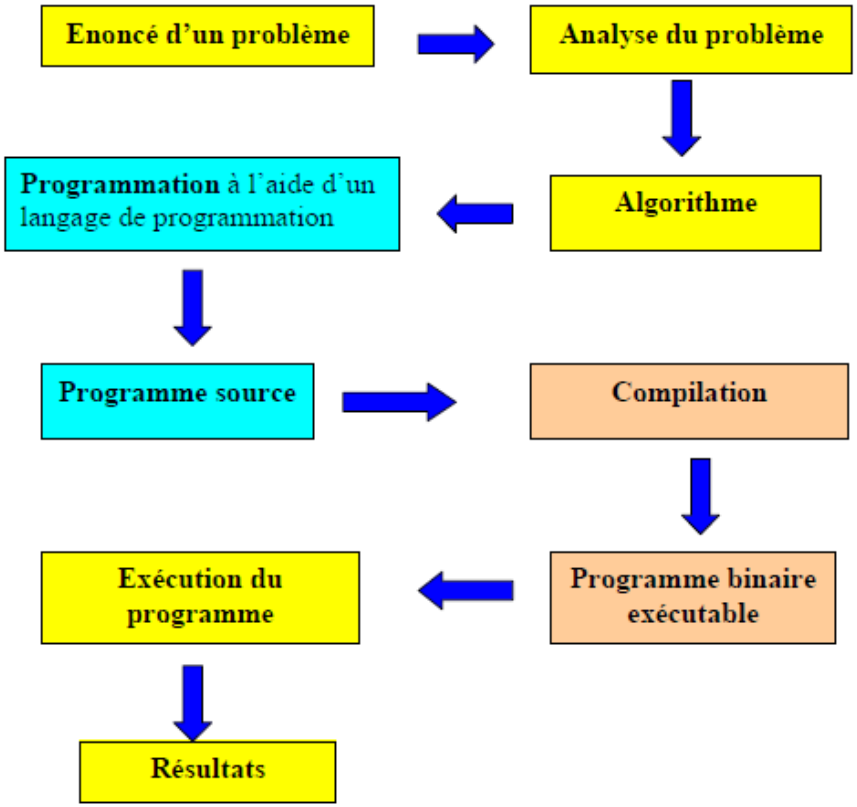
INTRODUCTION

Pour faire exécuter une tâche par ordinateur, il faut tout d'abord, détailler suffisamment les étapes de résolution du problème, pour qu'elle soit exécutable par l'homme. Ensuite, transférer la résolution en une suite d'étapes si élémentaire et simple à exécuter, pouvant être codée en un programme dans un langage compréhensible par ordinateur.

Toute suite d'étapes si élémentaire et simple à exécuter s'appelle un ALGORITHME.

UN PROGRAMME c'est un algorithme codé dans un langage compréhensible par ordinateur à l'aide d'un compilateur (traducteur).

INTRODUCTION



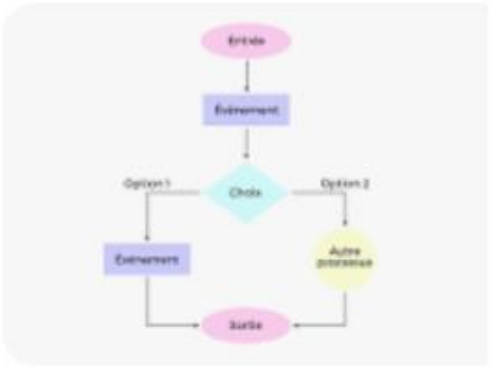
LA REPRÉSENTATION DES ALGORITHMES

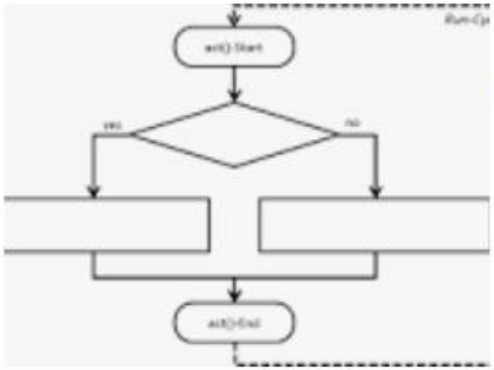
Pour décrire un algorithme comme une démarche de résolution d'un problème, on utilise ce qu'on appelle un ordinogramme, sorte de représentation graphique qui permet de mieux visualiser cette démarche.

Un algorithme peut également être représenté par un pseudo-langage constitué de mots et de règles syntaxiques conduisant à une forme standardisée appelée pseudo-code.

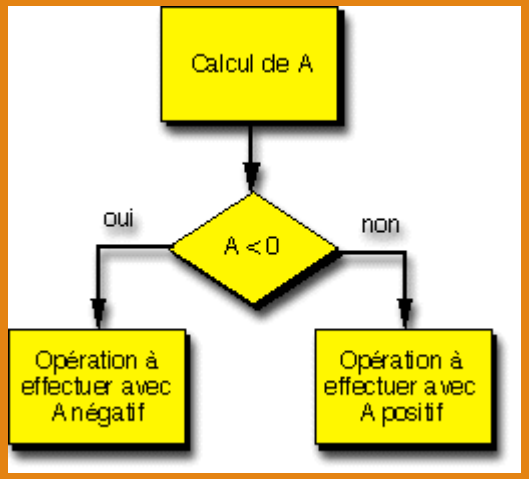
Représentation hiérarchique : Ordinogramme

L'ordinogramme est une sorte de représentation graphique qui permet de mieux visualiser un problème. Il est construit à partir d'un formalisme comprenant des symboles simples.





Représentation hiérarchique : Ordinogramme

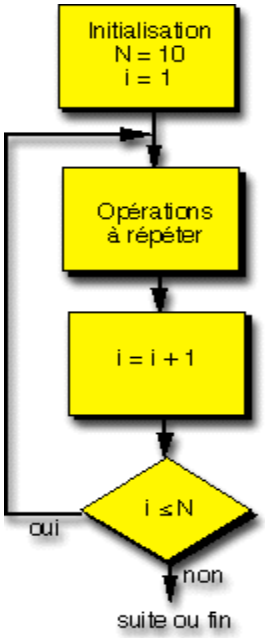


la boîte de traitement

la boîte de décision

Représentation hiérarchique : Ordinogramme

Répétition



EXERCICE :

Exercice 1:

Donner un ordinogramme qui demande une valeur et afficher s'elle est positive, négative ou nulle.

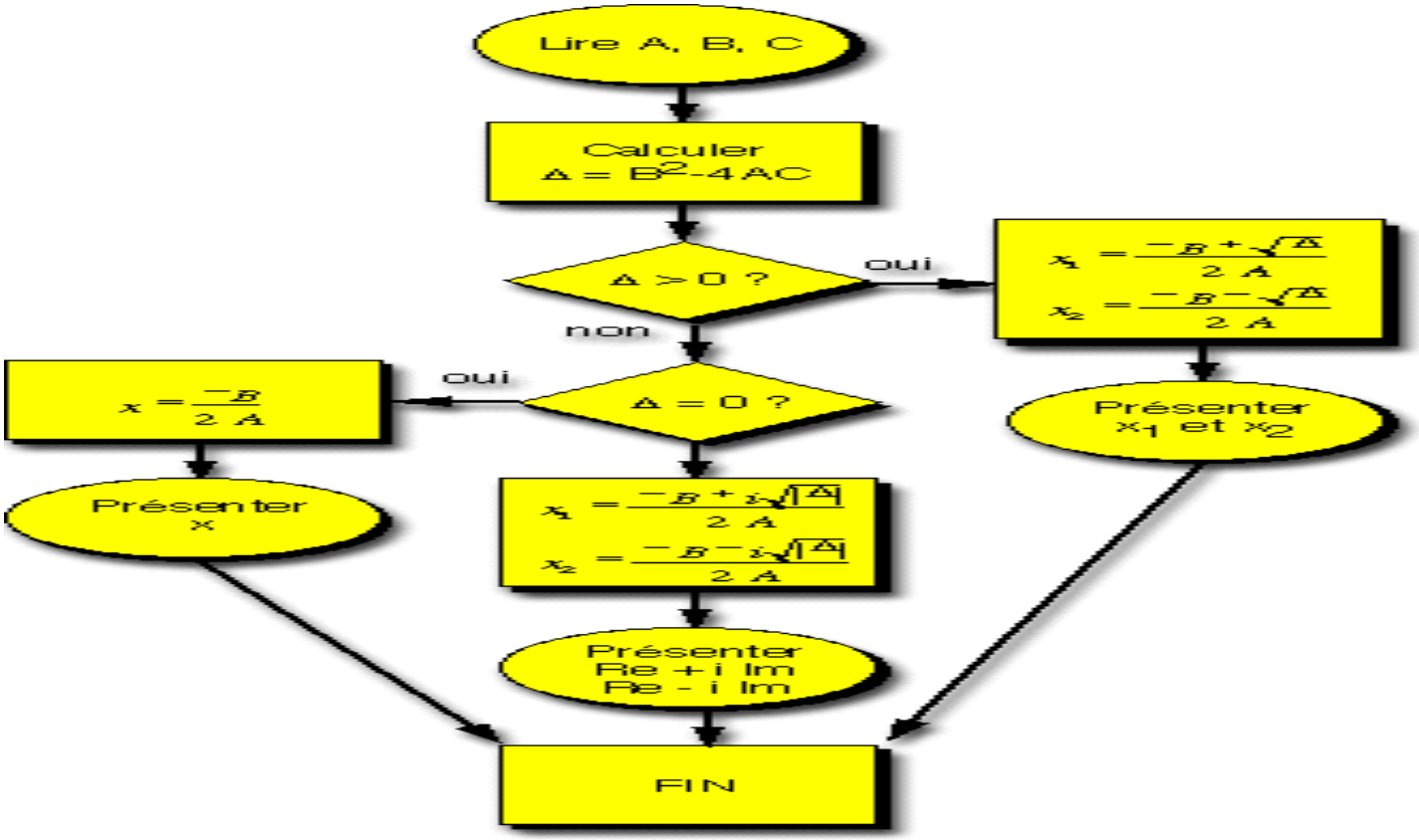
Exercice 2:

Donner un ordinogramme qui test la valeur entrée au clavier s'elle est positive ou négative . On arrêtera la saisie quand le nombre 0 est entré.

Exercice 3:

Donner un ordinogramme qui permet de résoudre une équation du second degré.

EXERCICE : Exemple complet



pseudo-code

Un algorithme doit être lisible et compréhensible par plusieurs personnes. Il doit donc suivre des règles précises, il est composé d'un entête et d'un corps :

l'entête, qui spécifie :

- **le nom de l'algorithme (Nom :)**
- **son utilité (Rôle :)**
- **les données ,c'est-à-dire les éléments qui sont indispensables à son bon fonctionnement (Variables:)**

le corps, qui est composé :

- **du mot clef début.**
- **d'une suite d'instructions.**

Il existe des instructions que l'ordinateur ignore complètement, ce sont les commentaires.

/* ceci est un commentaire */

du mot clef fin.

Nom:
Rôle:

Variable :

Debut

Instructions

Fin

Les instructions de base d'un algorithme

LES VARIABLES

LES OPERATEURS

LIRE ET ÉCRIRE

Les variables

La déclaration d'une variable se fait par la donnée du nom de la variable et du type de la variable.

Noms de variables

Le nom de la variable (l'étiquette) obéit à des règles :

- **Le nom de variable peut comporter des lettres et des chiffres,**
- **On exclut la plupart des signes de ponctuation, en particulier les espaces.**
- **Un nom de variable doit commencer par une lettre.**
- **Ne pas utiliser les mots clés du langage de programmation.**

Les variables

Types de variables

Lorsqu'on déclare une variable, il ne suffit pas de réserver un emplacement mémoire ; il faut préciser ce que l'on voudra mettre dedans, car cela dépend de la taille de l'emplacement mémoire et le type de codage utilisé

(entier ➔ 2 octets , réel ➔ 4 octets ,caractère ➔ 1 octet ,chaîne de caractère ,…)

Les variables

Affectation

L'affectation est l'action élémentaire dont l'effet est de donner une valeur à une variable

L'affectation est réalisée au moyen de l'opérateur ←

Exemple :

X ← 3

Donner à la variable X la valeur 3

X ← Y ← Z+2 ;

ceci revient à affecter à Y le résultat de l'évaluation de Z+2, puis à X le résultat de Y

Les operateurs

Opérateurs numériques :

+ addition

- soustraction

*** multiplication**

/ division

Mentionnons également le ^ qui signifie “puissance".

Opérateurs de comparaison

= != < > =< >=

Opérateurs logiques

ET, OU, et NON.(AND , OR , NOT)

Les operateurs

Les opérateurs d'incrémentation et de décrémentation

Les opérateurs d'incrémentation « ++ » et de décrémentation « -- » permettent de remplacer les deux instructions suivantes :

i = i + 1; <==> i++;

j = j - 1; <==> j--;

Lire et écrire

lire(A) signifie : mettre dans la variable A, la valeur donnée par le clavier(organe d'entrée de la machine).

écrire(A) signifie : mettre sur l'écran (organe de sortie de la machine) le contenu de la variable A .

Remarque:

Avant de lire une variable, il est conseillé d'écrire des messages sur l'écran, afin de prévenir l'utilisateur de ce qu'il doit frapper :

```
écrire("Entrez votre nom : ")

lire(NomFamille)
```

Exemple :

Écrire un algorithme qui demande deux entiers et qui calcule et affiche leur produit.

```
Variables A, B, C : entiers
Début
    écrire(" entrer la valeur de A : ")
    lire(A)
    écrire(" entrer la valeur de B : ")
    lire(B)
    C ← A * B
    écrire(" le produit de ",A," et ",B," est : ",C)
Fin
```

Remarque:

```
    écrire(" le produit de ",A," et ",B," est : ",C)
```

Cette ligne peut être remplacer par

```
    écrire(" le produit est : ",C)
```

Exercices

Les TESTS

Les TESTS

Les tests vont permettre à la machine de "choisir" les exécutions suivant les valeurs des données.

syntaxe

```
SI condition ALORS
    instructions1
Finsi
```

A pour effet de faire exécuter instructions1 si et seulement si condition est satisfaite.

Les TESTS

<u>Une condition</u> est une comparaison:

une valeur (peuvent être de n'importe quel type : numériques, caractères…).

- **un opérateur de comparaison.**
- **une autre valeur.**

Les opérateurs de comparaison sont : = != < > =< >=

<u>Conditions composées</u>

opérateurs logiques : ET, OU, et NON.(AND , OR , NOT)

Les TESTS

Syntaxe d'un test complet

SI condition ALORS

instructions1

SINON

instructions2

Finsi

A pour effet de faire exécuter instructions1 si condition 1 est satisfaite ou bien instructions2 dans le cas contraire.

Les TESTS

Exemple

Étant donnés deux nombres entiers positifs, identifier le plus grand des deux nombres.

```
variables A, B : entiers
début
    écrire("Programme permettant de déterminer le plus grand de deux entiers positifs")
    écrire("Entrer le premier nombre : ")
    lire(A)
    écrire ("Entrer le second nombre : ")
    lire(B)
    si (A>B) alors
        écrire("Le nombre le plus grand est : ",A)
    sinon
        écrire("Le nombre le plus grand est : ", B)
    finsi
fin
```

Les TESTS

Remarque : *Tests imbriqués*

```
Variable Temp:Entier
Début
   écrire("Entrez la température de l'eau :" )
   lire(Temp)
   si Temp > 0 Alors
         si Temp < 100 Alors
               écrire("C'est du liquide")
         sinon
               écrire("C'est de la vapeur")
         finsi
   sinon
        écrire("C'est de la glace")
   finsi
Fin
```

Les TESTS

Remarque : pour plusieurs tests imbriqués il est préférable d'utiliser cette instruction ➔Selon …

```
Selon (variable)
Choix1:
        instructions1
Choix2:
        instructions2
Choix3:
        instructions3
…
ChoixN:
        instructionsN
Sinon
        instructions
Finselon
```

Exercices

Les BOUCLES

Les BOUCLES

La notion d'itération (boucle) est une des notions fondamentales de l'algorithmique. On l'utilise souvent quand on doit exercer plusieurs fois le même traitement sur un même objet, ou plusieurs objets de même nature.

Il existe trois façons d'exprimer algorithmiquement l'itération :

- **TantQue**
- **Répéter … jusqu'à …**
- **Pour … jusqu'à …**

Les BOUCLES

La boucle TantQue

TantQue (conditions)

…

Instructions

…

FinTantQue

Si la valeur de la condition est VRAI, le programme exécute les instructions qui suivent, jusqu'à ce qu'il rencontre la ligne FinTantQue(condition est FAUX).

Les BOUCLES

Exemple :

```
Variable a entier
Debut
   a ← -1
   TantQue (a<0)
      écrire("Entrer une valeur positive")
      lire(a)
   FinTantQue
Fin
```

```
Variable a entier
Debut
      écrire("Entrer une valeur positive")
      lire(a)
   TantQue (a<0)
      écrire("Entrer une valeur positive")
      lire(a)
   FinTantQue
Fin
```

Les BOUCLES

La boucle Répéter … jusqu'à …

Répéter

…

Instructions

…

jusqu'à (conditions)

Le principe est simple : toutes les instructions écrites entre Répéter et jusqu'à sont exécutées au moins une fois et leur exécution est répétée jusqu'à ce que la condition d'arrêt est satisfaite.

Les BOUCLES

Exemple:

```
Variable a entier
Debut
   Répéter
      écrire("Entrer une valeur positive")
      lire(a)
   Jusqu'à (a<0)

Fin
```

Les BOUCLES

La boucle Pour … jusqu'à …

Cette boucle est utile surtout quand on connaît le nombre d'itérations à effectuer.

```
Pour i allant de début jusqu'à fin
    …
    Instructions
    …
FinPour
```

Le principe est simple :

- **on initialise i par début**
- **on test si on a pas dépassé fin**
- **on exécute les instructions**
- **on incrémente i ← (i + 1)**
- **on test si on a pas dépassé fin**
- **etc.**

Les BOUCLES

Exemple:

```
Variable i entier
Debut
   Pour i allant de début jusqu'à 10
      écrire("Bonjour !!!")

   finPour
Fin
```

Les BOUCLES

Remarque :

Si on désire utiliser la boucle TantQue ou Répeter au lieu de la boucle Pour, il faut initialiser et incrémenter le i.

Exemple :

```
variable n, i entières
Debut
    écrire("entrer le nombre n :")
    lire(n)
    i ←1
    TantQue(i<=n) faire
            écrire("Bonjour à tous la ",i," fois")
            i ← i + 1
    FinTantQue
Fin
```

Exercices