# Algorithmique & Structures de Donnés 1 (ASD-1)

ING TIC-1 / Cours intégré (42h : 21h Cours, 21h TD)

Hajer SALHI

# Plan du cours

Introduction à l'algorithmique

Chapitre 1 : Les Notions de base de l'algorithmique

Chapitre 2 : Quelques algorithmes sur les Tableaux

Chapitre 3 : La récursivité

Chapitre 4 : Les Structures de données dynamiques

Chapitre 5 : Les Structures de données abstraites

# Plan du cours

Introduction à l'algorithmique

1. Historique
2. Définition d'un algorithme
3. Objectifs

Chapitre 1 : Les Notions de base de l'algorithmique

Chapitre 2 : Quelques algorithmes sur les Tableaux

Chapitre 3 : La récursivité

Chapitre 4 : Les Structures de données dynamiques

Chapitre 5 : Les Structures de données abstraites

# Historique

- Algorithme : mot dérivé du nom du mathématicien Al_Khawarizmi qui a vécu au 9ème siècle (vers l'an 820), et qui était membre d'un académie des sciences à Bagdad.
- La notion d'algorithme est donc historiquement liée aux manipulations numériques et aux règles d'arithmétique.
  - s'est progressivement développée pour porter sur des objets de plus en plus complexes,
    - des textes, des images, des formules logiques, des objets physiques, etc.
- La traduction en latin européen du nom Al-Khwarizmi en algorithme au 18ième siècle.
- L'utilisation du mot a évolué pour inclure toutes les procédures définies pour résoudre un problème ou accomplir une tâche.
- L'algorithme au sens informatique apparait avec l'invention des premières machines dotées d'automatismes.

# Algorithmes ?

- Vous exécutez toujours des algorithmes dans votre vie quotidienne sans la savoir :
  - Quand vous montrez un chemin à un touriste
  - Quand vous faites vos courses au marché
  - Quand vous exécutez une recette de cuisine
  - Quand vous rechercher un mot dans un dictionnaire
  - …

**Vous créez et exécutez toujours des algorithmes**

# Définition d'un algorithme

- L'algorithme est le résultat d'une démarche logique de résolution d'un problème donné
- pour la mise en œuvre pratique sur ordinateur et afin d'obtenir des résultats concrets il faut passer par l'intermédiaire d'un langage de propagation.

- Un algorithme prend des données en entrée, exprime un traitement particulier et fournit des données en sortie.
- Programme : série d'instructions pouvant s'exécuter en séquence, ou en parallèle (parallélisme matériel) qui réalise (implémente) un algorithme.

# Définition d'un algorithme

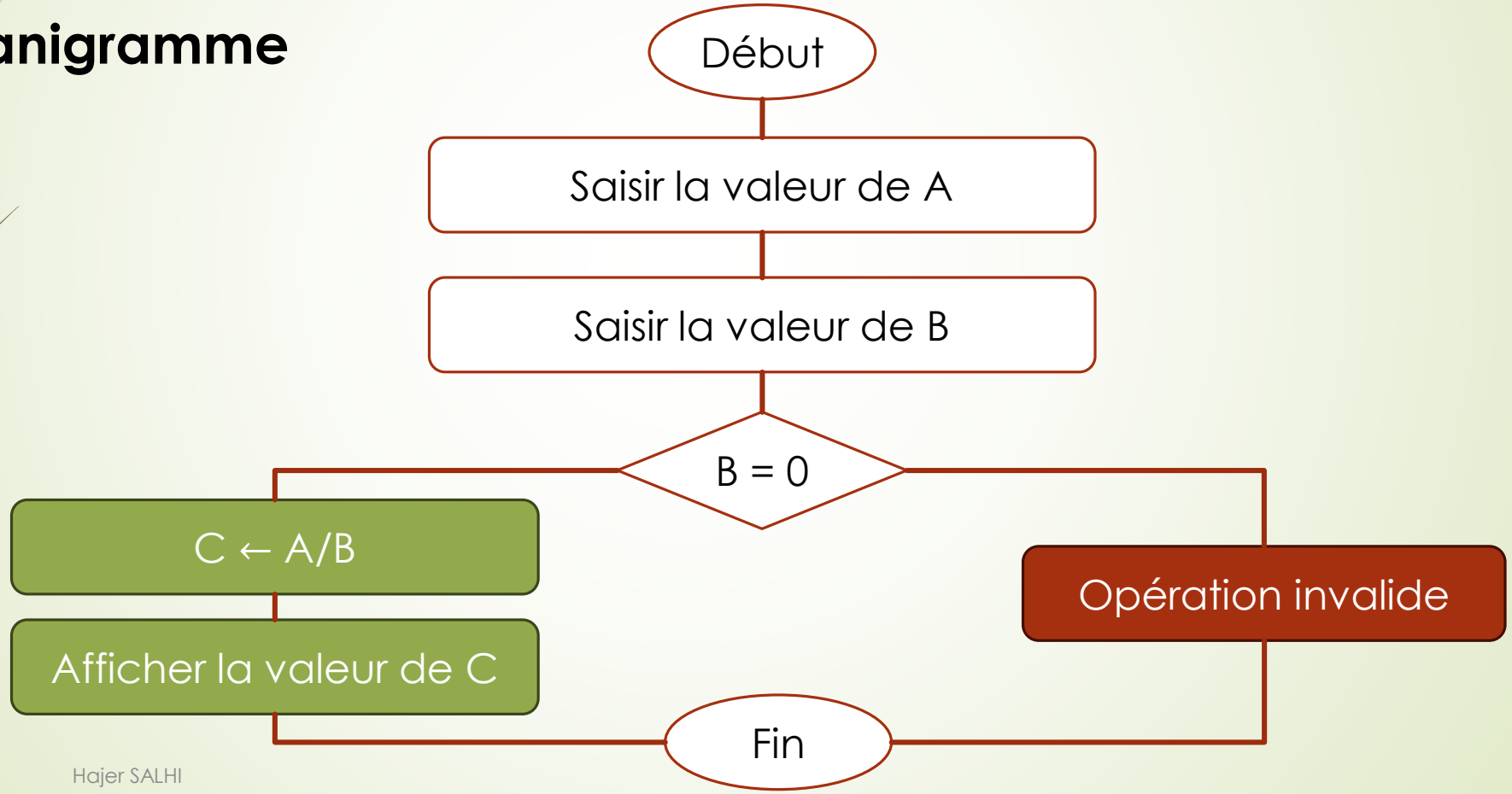
- Le choix du langage dépends de nombreux facteurs :
  - Domaine de la thématique à traiter (Web, Mobile, Réseau, Sécurité, Statistique, IA…)
  - Maniabilité et portabilité du langage de programmation
  - Fiabilité
  - etc
- Chaque langage à ses propre particularité et bien évidement sa propre syntaxe
- Le point commun est la logique de la programmation elle-même ➔ ***Algorithme***
- La première discipline à apprendre pour un futur programmeur ➔ ***Algorithmique***

# Représentation d'un algorithme

- Un algorithme peut être représenté sous forme de :
  - Langage naturel
  - Organigramme
  - Pseudo-code
- Le pseudo-code est plus utilisé car il est plus proche de la structure d'un vrai programme

# Représentation d'un algorithme
# Exemple : Division (A/B)

**Organigramme**



# Représentation d'un algorithme Exemple : Division (A/B)

**Pseudo-code**

```
Algorithme Division

Var
        A,B : Entier
        C : Réél

Début
        Ecrire ('Entrer la valeur de A :')
        Lire (A)
        Ecrire ('Entrer la valeur de B :')
        Lire (B)
        Si B= alors
                Ecrire('Opération invalide')
        Sinon
                C <- A/B
                Ecrire('Le résultat de la division est',C)
        FinSi
Fin
```

# Une approche pour bien programmer

- Soit un Problème P donné :
  - expliquer à la «machine» comment elle doit s'y prendre ?

- Besoins :
  - savoir expliciter son raisonnement
  - savoir formaliser son raisonnement
  - concevoir (et écrire) des algorithmes
    - séquence d'instructions qui décrit comment résoudre un problème particulier.

# Une approche pour bien programmer

- Pour écrire un algorithme
  - Savoir expliquer comment faire un travail sans la moindre ambiguïté
- Algorithme (démarche logique) :
  - suite finie d'actions (des instructions, élémentaires ou non élémentaires) respectant une chronologie imposée.
- Un algorithme ne dépend pas du langage dans lequel il est implémenté,
  - ni de la machine qui exécutera le programme correspondant.

# Une approche pour bien programmer

- Il n'existe pas de règles absolues pour construire un « bon » programme.
- Il existe seulement des propriétés générales qu'un bon programme doit avoir :
  - Il doit bien effectuer ses fonctionnalités.
  - Il doit être efficace de point de vue coût, c'est-à-dire il doit utiliser le minimum de ressources informatiques en terme de mémoire et de temps d'exécution.
- La difficulté pour écrire et comprendre de gros programmes réside dans la différence entre le langage naturel et le langage de programmation.
  - il faut se doter d'une approche ou méthodologie pour programmer juste.

# Une approche pour bien programmer

- On pourrait exposer cette approche **en trois étapes** :
- ***Etape 1 : La spécification des données et résultats***
  - Une bonne compréhension de l'énoncé du problème. En effet, il est impératif de bien lire la spécification du problème pour en extraire les données.
  - **Formuler le problème**, c'est à dire :
    - définir ses données, les caractériser par leurs natures, leurs propriétés et leurs domaines.
    - définir les résultats,
    - spécifier toutes les relations liant les résultats aux données et éventuellement les résultats entre eux.

# Une approche pour bien programmer

- ***Etape 2 : La résolution du problème :***
  - autrement dit la définition d'un algorithme réalisant le traitement,
  - Résoudre le problème à travers **un algorithme** réalisant le traitement décrit par la relation qui existe entre données et résultats.

# Une approche pour bien programmer

- ***Etape 3 : La codification de la solution dans le langage choisi.***
  - Rédiger, dans **un langage de programmation**, un programme qui codifie la solution proposée dans l'algorithme et l'adapte aux possibilités offertes aussi bien par le langage de programmation choisi que par l'ordinateur.

# Une approche pour bien programmer Exemple

- Calcul de la moyenne de 10 notes :
  - Etape 1 :
    - Les données sont les 10 notes.
    - Le résultat est un nombre à calculer.
    - La relation entre les données et le résultat : le résultat est égale à la moyenne des 10 nombres.

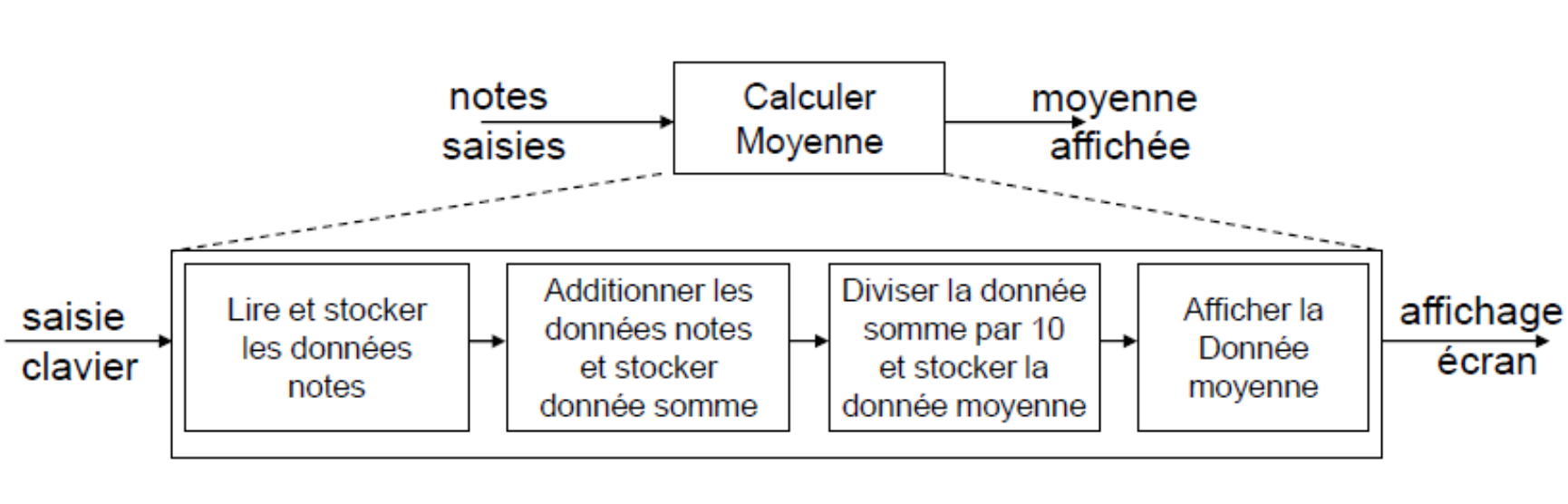
# Une approche pour bien programmer Exemple

- Calcul de la moyenne de 10 notes :
  - Etape 2 : On peut résoudre ce problème en exécutant les actions suivantes dans l'ordre :
    - 1) taper le premier nombre.
    - 2) taper le deuxième nombre.
    - 3) taper le troisième nombre.
    - …
    - 10) taper le dixième nombre
    - 11) calculer la somme
    - 12) calculer la moyenne
    - 13) afficher le résultat

} Saisie des données nécessaires

# Une approche pour bien programmer Exemple

- Calcul de la moyenne de 10 notes :
  - Etape 2 :



# Une approche pour bien programmer Exemple

- Calcul de la moyenne de 10 notes :
  - Etape 2 :

```
Algorithme Somme

Variables
        x, S: Entier
        M : Réél

Début
        S<-0
        Pour i de 1 à 10 faire
                Ecrire ('Entrer la', i, 'ème valeur :')
                Lire (x)
                S <- S+x
        FinPour
        M<-C/10
        Ecrire('La moyenne est égale à ',M)
Fin
```

# Une approche pour bien programmer Exemple

- Calcul de la moyenne de 10 notes :
  - Etape 3 :

Langage C :

```
1
2   int main()
3   {
4       int x, S;
5       float M;
6
7       S = 0;
8       for (i=1; i<=10; i++)
9           printf ('Entrer la %d ème valeur :',x);
10          scanf("%d",&x);
11          S <- S+x
12      FinPour
13      M = C/10;
14      printf ('La moyenne est égale à : %f',M);
15
16  Fin
```

# Objectifs :

- Connaitre et savoir utiliser les différentes structures de données
  - Choisir la bonne structure de donnée selon l'application
- Connaitre les différents algorithmes de Tri
- Savoir partitionner le problème en des sous-problèmes moins complexes
  - Maitriser l'utilisation des procédures et fonctions
- Maitriser la récursivité

→ **Elaborer des algorithmes exacts, performants et efficaces**

# Comment écrire un algorithme