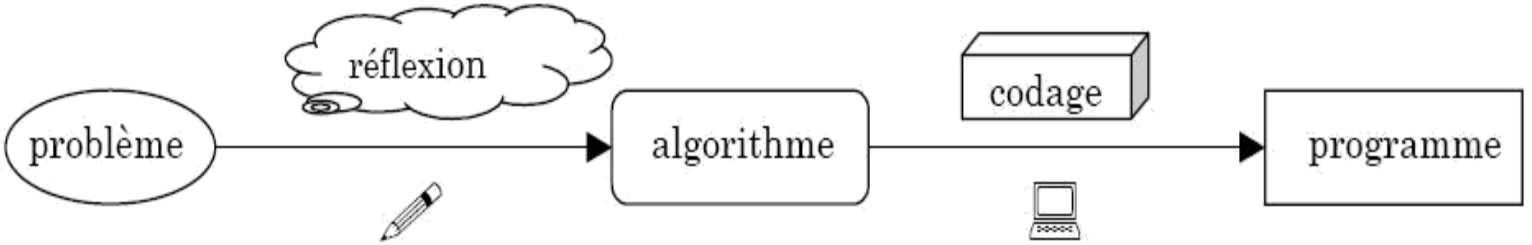
Algorithmique

Définition

- *Définition* : Un algorithme est un **ensemble d'opérations** de calcul élémentaires, **organisé** selon des **règles** précises dans le but de **résoudre** un **problème** donné. Pour chaque donnée du problème, l'algorithme retourne une réponse après un nombre **fini d'opérations** (+, -,/,<,>,...).
- Un algorithme est une **procédure systématique** permettant de résoudre une **classe** de **problèmes**. À partir d'une entrée (représentant une **instance** du problème), un algorithme suit un ensemble **déterminé** de règles et, en un nombre **fini** d'étapes, produit une sortie (représentant une réponse à **l'instance** donnée).
- L'algorithmique est donc l'étude des algorithmes, de leur conception et de leurs **propriétés**.

Méthodologie

- Pour résoudre un problème, il est vivement conseillé de réfléchir d'abord à l'algorithme avant de programmer.



- La résolution d'un problème est caractérisé par 4 étapes:
  - Comprendre la nature du problème posé
  - Préciser les données fournies (Entrées)
  - Préciser les résultats que l'on désire obtenir (Sorties)
  - Déterminer le processus de transformation des données en résultats.

Les instructions fondamentales

Les ordinateurs ne comprennent que **quatre catégories** d'instructions :

1. **l'affectation de variables** ;
2. **la lecture/écriture** ;
3. **les tests** (les structures conditionnelles) ;
4. **les boucles** (les structures itératives).

Important

- Un algorithme informatique se ramène **toujours** à la **combinaison** de ces quatre types **d'instruction**.

Un premier algorithme

```
Algorithme  AuCarre
   {Cet algorithme calcule le carré du nombre que lui fournit l'utilisateur}
  variables unNombre, sonCarre: entiers
 début
  {saisie d'un nombre}
  afficher("Quel nombre voulez-vous elever au carre?")
  saisir(unNombre)
 {calcul du carré du nombre saisi}
  sonCarre ← unNombre X unNombre
 {affichage du résultat}
 afficher("Le carre de ",unNombre)
 afficher("c'est ",sonCarre)
fin
```

Algorithme **AuCarre**

```
variables unNombre, sonCarre: entiers
```

Bloc de Déclaration

début

```
afficher("Quel nombre voulez-vous elever au carre?")
saisir(unNombre)
sonCarre ← unNombre X unNombre
afficher("Le carre de ",unNombre)
afficher("c'est ",sonCarre)
```

Bloc de Traitement Ou Bloc actions

fin

Les variables

- Une variable est un objet dont la valeur peut changer

- Une variable peut être représentée par une **case mémoire**, qui contient la valeur d'une donnée.

- Chaque variable possède un nom unique appelé **identificateur** par lequel on peut accéder à son contenu.

Les variables(suite)

- Une variable possède 3 attributs :
  - Une valeur
  - Un nom (invariable) qui sert à la désigner
  - Un type (invariable) qui décrit l'utilisation possible de la variable
- **Une valeur**
  - la valeur d'une variable (**contenu**) **peut varier** au cours du programme. **L'ancienne** valeur est tout simplement **écrasée** et **remplacée** par la **nouvelle**.

Les variables(suite)

- Nom de la variable
  - C'est une **suite** de lettres et de chiffres **commençant** nécessairement **par** une **lettre**
  - Le nombre maximal de caractères imposé **varie** selon les langages
  - La lisibilité des programmes dépend de l'habilité à **choisir** des noms **représentatifs**
  - Le nom de la variable doit être le plus **représentatif** possible du contenu de celle-ci pour **faciliter** la **lecture** de l'algorithme. En revanche, il ne doit pas non plus être **trop long** pour ne pas nuire à la **lisibilité** de l'ensemble.

Les variables(suite)

- Type de lavariable
  - **Entier** : il s'agit des variables destinées à contenir un nombre entier positif ou négatif.
  - **Réel** :il s'agit des variables numériques qui ne sont pas des entiers, c'est à dire qui comportent des décimales
  - **Caractère** :Les variables de type caractère contiennent des caractères alphabétiques ou numériques seul(ex: 'c')
  - **Booléen** :Les variables qui prennent les valeurs (vrai ou faux) ou les valeurs (oui ou non).

Les variables(suite)

- Type de la variable
  - **Chaîne de caractères :** représentant un texte, contenant un ou plusieurs caractères(ex: "Bonjour tout le monde")
  - Tous les traducteurs de langages prennent en compte cette notion de type par des instructions de déclaration de type
  - Exemple:

Variable Moyenne : **réel**; **{Moyenne en numérique}**
Variable NbreEtudiant : **entier; {NbreEtudiant en numérique}**
Variable c1, lettre, z : **caractère**;

Instruction d'affectation

- **Exemple :**

1. affecter une valeur à une variable

   - X ← 5, On charge la variable X avec la valeur 5

2. Affecter le contenu d'une variable à une autre variable

   - X ← Y, On charge X avec le contenu de Y

   - Y représente :
     - Constante ou **Nom d'une variable** ou **Expression logique**

   - X et Y doivent être de **même type**

Les instructions(suite)

Les instructions d'Entrée/Sortie

- Un programme est amené à :
  - **Prendre des données** par le périphérique(clavier) : rôle de l'instruction de lecture
  - **Communiquer** des **résultats** par l'intermédiaire du périphérique(écran) : rôle de l'instruction de l'écriture
- Instruction de lecture
  - Rôle : fournir des données au programme
  - Syntaxe : **saisir**(variable)
  - Exemple : **saisir**(X) on saisie une valeur pour la stocker après dans la variable X
- Instruction d'écriture
  - Rôle : fournir des résultats directement compréhensibles
  - Syntaxe : **afficher**(variable), **afficher**(''chaine de caractères'')
  - Exemple : **afficher**(X), **afficher**(''Bonjour'')

Permutation

```
1  Algorithme CaFaitQuoi?
2  {Cet algorithme .....}
3  variables valA, valB: réels
4  début
5  {saisie de deux valeurs }
6  afficher("Donnez-moi deux valeurs :")
7  saisir (valA, valB)
8  afficher("Vous m'avez donne ", valA, "et ", valB)
9  {traitement :??? }
10 valA←valB
11 valB←valA
12 {présentation du résultat}
13 afficher("Maintenant, mes donnees sont : ", valA, "et ", valB)
14 Fin
```

Est-ce-que la permutation est faite ??

Ce qu'il fallait faire

- Déclarer une variable supplémentaire

variables valA, valB, valTemp : entiers

- Utiliser cette variable pour stocker provisoirement une des valeurs :

saisir(valA,

valB)

valTemp←valA

valA←valB

valB←valTemp

L'instruction conditionnelle

Syntaxe :

```
si   <expression logique>   alors
     <bloc instruction 1>
     [sinon <bloc instruction 2> ]
fsi
```

Fonction :

Si l'expression logique (la condition) prend la valeur **vrai, le premier bloc d'instructions est** exécuté; si elle prend la valeur **faux, le second** bloc est exécuté (s'il est présent).

les conditionnelles imbriquées

Problème :

Afficher "Reçu avec mention" si une note est supérieure ou égale à 12, "Passable" si elle est supérieure à 10 et inferieure a 12, et "Insuffisant" dans tous les autres cas.

Solution :

```
1  si note >= 12    alors
2          afficher( "Recu avec mention" )
3    sinon si note >= 10 alors
4              afficher( "Passable" )
5          sinon
6              afficher( "Insuffisant" )
7        fsi
8  fsi
```

Exercices

- **Exercice 1:**

  Calculer le montant de la facture d'un client ayant commandé une quantité d'un produit avec un prix unitaire hors taxe

  - Le taux de T.V.A est : 20%
  - Les frais de transport sont 0.8 DH de Km , Le client est disposé du frais de transport si le montant est supérieur 4500 DH

- **Exercice 2:**

  Écrire un algorithme qui calcule et affiche le maximum de trois nombre A,B et C ?

- **Exercice 3:**

  Écrire un algorithme qui permet de résoudre l'équation du premier degré : $aX+b=0$?

- **Exercice 4:**

  Écrire un algorithme qui permet de résoudre une équation de second degré : $aX^2+bX+c=0$?

- **Exercice 5:**

  Écrire un algorithme qui lie trois nombre A,B et C , puis il détermine si l'un est égal à la somme de 2 autres sinon il affiche un message « pas de solution »?

La sélection sur choix multiples

```
1  si abreviation ="M"
2          alors afficher( "Monsieur" )
3          sinon si abreviation = "Mme"
4                  alors afficher("Madame")
5                  sinon si abreviation = "Mlle"
6                          alors afficher( "Mademoiselle" )
7                          sinon afficher( "Monsieur,Madame ")
8                  fsi
9          fsi
10 fsi
```

```
1  selon abreviation
2          "M" : afficher( "Monsieur" )
3          "Mme" : afficher( "Madame" )
4          "Mlle" : afficher( "Mademoiselle" )
5          autres : afficher( "Monsieur, Madame ")
6  fin selon
```

L'instruction selon

Syntaxe :

```
selon <identificateur>
        (liste de) valeur(s) : instructions
        (liste de) valeur(s) : instructions
        …
        [autres : instructions]
fin selon
```

Fonction :

S'il y a plus de deux choix possibles, l'instruction selon permet une facilite d'ecriture.

- Meme efficacite (meme nombre d'operations)
- Mais pas de tests possibles

Structure répétitive(suite)

- **Tant que**
- **Les répétitives où la condition d'arrêt est placée au début.**
  - **Syntaxe** : :

    Tant que <expression logique> **faire**

    <séquence d'instructions>

    Ftantque

    TantQue condition

    actions

    FinTantQue

  - Ce qui signifie : tant que la condition est vraie, on exécute les actions.

Pour

```
Variable Num1 : Entier
Début
Num1 ← 0
TantQue Num1 < 15 Faire
    Num1 = Num1 + 1
afficher("Passage numéro :",Num1)
FinTantQue
Fin
```

```
Variable Num1 : Entier
Début
Pour Num1 = 1 à 15 Faire
afficher( "Passage numéro :", Num1)
Num1 Suivant
Fin Pour
Fin
```

la structure : Pour est un cas particulier de TantQue : celui où le programmeur peut dénombrer à l'avance le nombre de tours de boucles nécessaires.

Structure répétitive

- **Pour**
  - Les répétitives où le nombre d'itération est fixée une fois pour toute.
  - **Syntaxe :**

```
Pour Compteur = Initial à Final Pas ValeurDuPas
        …
        Instructions
        …
Fin Pour
```

Structure répétitive(suite)

- Les structures **TantQue** sont employées dans les situations où l'on doit procéder à un traitement sur les éléments d'un ensemble dont on ne connaît pas d'avance la quantité, comme par exemple :
  - le contrôle d'une saisie.
  - la gestion des tours d'un jeu (tant que la partie n'est pas finie, on recommence).
- Les structures **Pour** sont employées dans les situations où l'on doit procéder à un traitement sur les éléments d'un ensemble dont on connaît d'avance la quantité.

Structure répétitive(suite)

- **Répéter..Jusqu'à :**
  - la condition d'arrêt est placée à la fin
  - **Syntaxe :**

    Répéter

    <séquence d'instructions>

    Jusqu'à <expression logique>

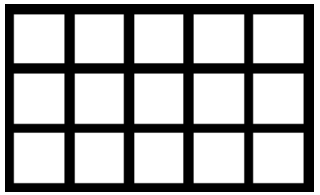
    Frépéter
  - **Exemple :**

```
Num1:=1
Répéter
        Afficher ("Passage numéro : ",Num1)
        Num1 = Num1 + 1
Jusqu'à (Num1 >= 15)
```

Les tableaux

- Un tableau est un ensemble ordonne contenant un nombre constant d‘éléments d'un même type ;
- Exemple
  - un tableau a une dimension de 7 éléments :

  - un tableau a deux dimensions de 3 * 5 = 15 éléments :



  - Un tableau peut avoir n'importe quel nombre de dimensions ; En général, on manipule surtout :
- des tableaux a une dimension (= vecteurs) ;
- des tableaux a deux dimensions (= matrices) ;