

Cours Algorithme(INFO_160)

Mr RAZONARISOA Nirina

Objectif et plan du cours

▶ **Objectif:**

- Apprendre les concepts de base de l'algorithmique et de la programmation
- Etre capable de mettre en œuvre ces concepts pour analyser des problèmes simples et écrire les programmes correspondants

▶ **Plan:**

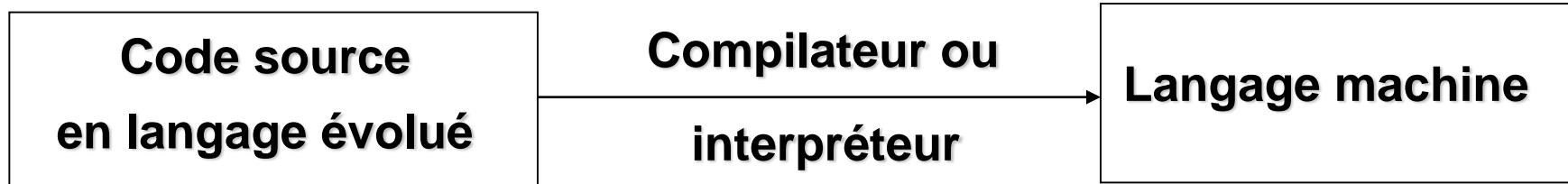
- Généralités (matériel d'un ordinateur, systèmes d'exploitation, langages de programmation, ...)
- Algorithmique (affectation, instructions conditionnelles, instructions itératives, fonctions, procédures, ...)

Langages informatiques

- ▶ Un langage informatique est un outil permettant de donner des ordres (**instructions**) à la machine
 - A chaque instruction correspond une action du processeur
- ▶ Intérêt : écrire des **programmes** (suite consécutive d'instructions) destinés à effectuer une tâche donnée
 - Exemple: un programme de gestion de comptes bancaires
- ▶ Contrainte: être compréhensible par la machine

Langages haut niveau

- ▶ Intérêts multiples pour le haut niveau:
 - proche du langage humain «anglais» (compréhensible)
 - permet une plus grande portabilité (indépendant du matériel)
 - Manipulation de données et d'expressions complexes (réels, objets, a^*b/c , ...)
- ▶ Nécessité d'un traducteur (compilateur/interpréteur), exécution plus ou moins lente selon le traducteur



Compilateur/interpréteur

- ▶ Compilateur: traduire le programme entier une fois pour toutes



- + plus rapide à l'exécution
- + sécurité du code source
- - il faut recompiler à chaque modification

- ▶ Interpréteur: traduire au fur et à mesure les instructions du programme à chaque exécution

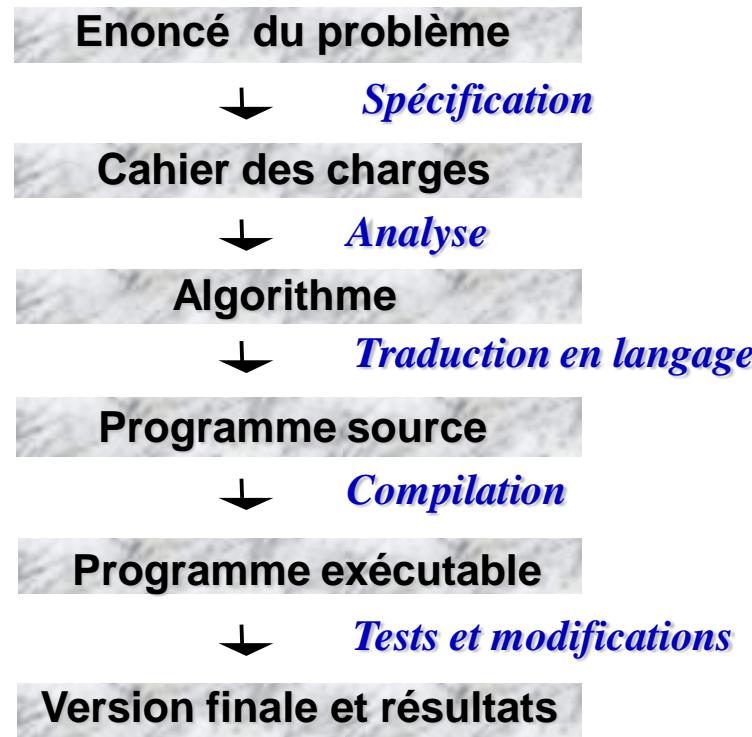


- + exécution instantanée appréciable pour les débutants
- - exécution lente par rapport à la compilation

Langages de programmation:

- ▶ Deux types de langages:
 - Langages procéduraux
 - Langages orientés objets
- ▶ Exemples de langages:
 - **Fortran, Cobol, Pascal, C, ...**
 - **C++, Java, ...**
- ▶ Choix d'un langage?

Etapes de réalisation d'un programme



La réalisation de programmes passe par l'écriture d'algorithmes
⇒ D'où l'intérêt de l'**Algorithmique**

Algorithmique

- ▶ Le terme **algorithme** vient du nom du mathématicien arabe **Al-Khawarizmi** (820 après J.C.)
- ▶ Un algorithme est une description complète et détaillée des actions à effectuer et de leur séquencement pour arriver à un résultat donné
 - Intérêt: séparation analyse/codage (pas de préoccupation de syntaxe)
 - Qualités: **exact** (fournit le résultat souhaité), **efficace** (temps d'exécution, mémoire occupée), **clair** (compréhensible), **général** (traite le plus grand nombre de cas possibles), ...
- ▶ **L'algorithmique** désigne aussi la discipline qui étudie les algorithmes et leurs applications en Informatique
- ▶ Une bonne connaissance de l'algorithmique permet d'écrire des algorithmes exacts et efficaces

Représentation d'un algorithme

Historiquement, deux façons pour représenter un algorithme:

- **L'Organigramme:** représentation graphique avec des symboles (carrés, losanges, etc.)
 - offre une vue d'ensemble de l'algorithme
 - représentation quasiment abandonnée aujourd'hui
- **Le pseudo-code:** représentation textuelle avec une série de conventions ressemblant à un langage de programmation (sans les problèmes de syntaxe)
 - plus pratique pour écrire un algorithme
 - représentation largement utilisée

Notions et instructions de base

Notion de variable

- ▶ Dans les langages de programmation une **variable** sert à stocker la valeur d'une donnée
- ▶ Une variable désigne en fait un emplacement mémoire dont le contenu peut changer au cours d'un programme (d'où le nom variable)
- ▶ Règle : Les variables doivent être **déclarées** avant d'être utilisées, elle doivent être caractérisées par :
 - un nom (**Identificateur**)
 - un **type** (entier, réel, caractère, chaîne de caractères, ...)

Choix des identificateurs (1)

Le choix des noms de variables est soumis à quelques règles qui varient selon le langage, mais en général:

- ▶ Un nom doit commencer par une lettre alphabétique
exemple valide: A1 **exemple invalide: 1A**
 - ▶ doit être constitué uniquement de lettres, de chiffres et du soulignement _
(Eviter les caractères de ponctuation et les espaces) **valides: cdi2016,**
cdi_2016 **invalides: cdi 2016,cdi-2016,cdi;2016**
 - ▶ doit être différent des mots réservés du langage (par exemple en Java: **int, float, else, switch, case, default, for, main, return, ...**)
 - ▶ La longueur du nom doit être inférieure à la taille maximale spécifiée par le langage utilisé

Choix des identificateurs (2)

Conseil: pour la lisibilité du code, choisir des noms significatifs qui décrivent les données manipulées

exemples: **TotalVentes2016, Prix_TTC, Prix_HT**

Remarque: en pseudo-code algorithmique, on va respecter les règles citées, même si on est libre dans la syntaxe

Types des variables

Le type d'une variable détermine l'ensemble des valeurs qu'elle peut prendre, les types offerts par la plupart des langages sont:

▶ Type numérique (entier ou réel)

- **Byte** (codé sur 1 octet): de 0 à 255
- **Entier court** (codé sur 2 octets) : -32 768 à 32 767
- **Entier long** (codé sur 4 ou 8 octets)
- **Réel simple précision** (codé sur 4 octets)
- **Réel double précision** (codé sur 8 octets)

▶ Type logique ou booléen: deux valeurs VRAI ou FAUX

▶ Type caractère: lettres majuscules, minuscules, chiffres, symboles, ...

exemples: 'A', 'a', '1', '?', ...

▶ Type chaîne de caractère: toutes suites de caractères,

exemples: " Nom, Prénom", "code postal: 1000", ...

Déclaration des variables

- ▶ Rappel: toutes variables utilisées dans un programme doit avoir fait l'objet d'une déclaration préalable
- ▶ En pseudo-code, on va adopter la forme suivante pour la déclaration de variables

Variables liste d'identificateurs : type

- ▶ Exemple:

Variables i, j,k : entier

x, y : réel

OK: booléen

ch1, ch2 : chaîne de caractères

- ▶ Remarque: pour le type numérique on va se limiter aux entiers et réels sans considérer les sous types

L'instruction d'affectation

- ▶ **l'affectation** consiste à attribuer une valeur à une variable
(ça consiste en fait à remplir où à modifier le contenu d'une zone mémoire)
- ▶ En pseudo-code, l'affectation se note avec le signe \leftarrow
 $\text{Var} \leftarrow \text{e}$: attribue la valeur de e à la variable Var

- e peut être une valeur, une autre variable ou une expression
- Var et e doivent être de même type ou de types compatibles
- l'affectation ne modifie que ce qui est à gauche de la flèche

- ▶ **Ex valides:** $i \leftarrow 1$ $j \leftarrow i$ $k \leftarrow i+j$
 $x \leftarrow 10.3$ $OK \leftarrow \text{FAUX}$ $ch1 \leftarrow "SMI"$
 $ch2 \leftarrow ch1$ $x \leftarrow 4$ $x \leftarrow j$

(voir la déclaration des variables dans le transparent précédent)

- ▶ **non valides:** $i \leftarrow 10.3$ $OK \leftarrow "SMI"$ $j \leftarrow x$

Quelques remarques

- ▶ Beaucoup de langages de programmation (C/C++, Java, ...) utilisent le signe égal = pour l'affectation \leftarrow . Attention aux confusions:
 - l'affectation n'est pas commutative : $A=B$ est différente de $B=A$
 - l'affectation est différente d'une équation mathématique :
 - $A=A+1$ a un sens en langage de programmation
 - $A+1=2$ n'est pas possible en langage de programmation et n'est pas équivalente à $A=1$
- ▶ Certains langages donnent des valeurs par défaut aux variables déclarées. Pour éviter tout problème il est préférable **d'initialiser les variables** déclarées

Exercices simples sur l'affectation (1)

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

Variables A, B, C: Entier

Début

A ← 3

B ← 7

A ← B

B ← A + 5

C ← A + B

C ← B – A

Fin

Exercices simples sur l'affectation (2)

Donnez les valeurs des variables A et B après exécution des instructions suivantes ?

Variables A, B : Entier

Début

A ← 1

B ← 2

A ← B

B ← A

Fin

Les deux dernières instructions permettent-elles d'échanger les valeurs de A et B ?

Exercices simples sur l'affectation (3)

Ecrire un algorithme permettant d'échanger les valeurs de deux variables A et B

Expressions et opérateurs

- ▶ Une **expression** peut être une valeur, une variable ou une opération constituée de variables reliées par des **opérateurs**
exemples: **1, b, a*2, a+ 3*b-c, ...**
- ▶ L'évaluation de l'expression fournit une valeur unique qui est le résultat de l'opération
- ▶ Les **opérateurs** dépendent du type de l'opération, ils peuvent être :
 - **des opérateurs arithmétiques:** +, -, *, /, % (modulo), ^ (puissance)
 - **des opérateurs logiques:** NON, OU, ET
 - **des opérateurs relationnels:** =, ≠, <, >, <=, >=
 - **des opérateurs sur les chaînes:** & (concaténation)
- ▶ Une expression est évaluée de gauche à droite mais en tenant compte de **priorités**

Priorité des opérateurs

- ▶ Pour les opérateurs arithmétiques donnés ci-dessus, l'ordre de priorité est le suivant (du plus prioritaire au moins prioritaire) :
 - ^ élévation à la puissance
 - * , / multiplication, division
 - % modulo
 - + , - addition, soustraction
- ▶ En cas de besoin (ou de doute), on utilise les parenthèses pour indiquer les opérations à effectuer en priorité
 - exemple:** **$2 + 3 * 7$ vaut 23**
 - exemple:** **$(2 + 3) * 7$ vaut 35**

Les instructions d'entrées-sorties: lecture et écriture (1)

- ▶ Les instructions de lecture et d'écriture permettent à la machine de communiquer avec l'utilisateur
- ▶ La **lecture** permet d'entrer des données à partir du clavier
 - En pseudo-code, on note: **lire (var)**
la machine met la valeur entrée au clavier dans la zone mémoire nommée var
 - Remarque: Le programme s'arrête lorsqu'il rencontre une instruction Lire et ne se poursuit qu'après la frappe d'une valeur au clavier et de la touche Entrée

Les instructions d'entrées-sorties: lecture et écriture (2)

- ▶ L'**écriture** permet d'afficher des résultats à l'écran (ou de les écrire dans un fichier)
 - En pseudo-code, on note: **écrire (var)**
la machine affiche le contenu de la zone mémoire var
 - Conseil: Avant de lire une variable, il est fortement conseillé d'écrire des messages à l'écran, afin de prévenir l'utilisateur de ce qu'il doit frapper

Exemple (lecture et écriture)

Ecrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui calcule et affiche le double de ce nombre

Algorithme Calcul_double

variables A, B : entier

Début

écrire("entrer le nombre ")

lire(A)

B \leftarrow 2*A

écrire("le double de ", A, "est :", B)

Fin

Exercice (lecture et écriture)

Ecrire un algorithme qui vous demande de saisir votre nom puis votre prénom et qui affiche ensuite votre nom complet

Algorithme AffichageNomComplet

variables Nom, Prenom, Nom_Complet : chaîne de caractères

Début

écrire("entrez votre nom")

lire(Nom)

écrire("entrez votre prénom")

lire(Prenom)

 Nom_Complet ← Nom & Prenom

écrire("Votre nom complet est : ", Nom_Complet)

Fin

Tests: instructions conditionnelles (1)

Tests: instructions conditionnelles (2)

- ▶ La partie Sinon n'est pas obligatoire, quand elle n'existe pas et que la condition est fausse, aucun traitement n'est réalisé
 - On utilisera dans ce cas la forme simplifiée suivante:

Si condition **alors**

instruction ou suite d'instructions1

Finsi

Exemple (Si...Alors...Sinon)

Algorithme AffichageValeurAbsolue (version1)

Variable x : réel

Début

Ecrire (" Entrez un réel : ")

Lire (x)

Si (x < 0) **alors**

Ecrire ("la valeur absolue de ", x, "est:", -x)

Sinon

Ecrire ("la valeur absolue de ", x, "est:", x)

Finsi

Fin

Exemple (Si...Alors)

Algorithme AffichageValeurAbsolue (version2)

Variable x,y : réel

Début

Ecrire (" Entrez un réel : ")

Lire (x)

 y \leftarrow x

Si (x < 0) **alors**

 y \leftarrow -x

Finsi

Ecrire ("la valeur absolue de ", x, "est:",y)

Fin

Exercice (tests)

Ecrire un algorithme qui demande un nombre entier à l'utilisateur, puis qui teste et affiche s'il est divisible par 3

Algorithme Divsible_par3

Variable n : entier

Début

Ecrire " Entrez un entier : "

Lire (n)

Si (n%3=0) alors

Ecrire (n," est divisible par 3")

Sinon

Ecrire (n," n'est pas divisible par 3")

Finsi

Fin

Conditions composées

- ▶ Une condition composée est une condition formée de plusieurs conditions simples reliées par des opérateurs logiques:
ET, OU, OU exclusif (XOR) et NON
- ▶ Exemples :
 - x compris entre 2 et 6 : $(x > 2)$ ET $(x < 6)$
 - n divisible par 3 ou par 2 : $(n \% 3 = 0)$ OU $(n \% 2 = 0)$
 - deux valeurs et deux seulement sont identiques parmi a, b et c :
 $(a = b)$ XOR $(a = c)$ XOR $(b = c)$
- ▶ L'évaluation d'une condition composée se fait selon des règles présentées généralement dans ce qu'on appelle tables de vérité

Tables de vérité

C1	C2	C1 ET C2
VRAI	VRAI	VRAI
VRAI	FAUX	FAUX
FAUX	VRAI	FAUX
FAUX	FAUX	FAUX

C1	C2	C1 OU C2
VRAI	VRAI	VRAI
VRAI	FAUX	VRAI
FAUX	VRAI	VRAI
FAUX	FAUX	FAUX

C1	C2	C1 XOR C2
VRAI	VRAI	FAUX
VRAI	FAUX	VRAI
FAUX	VRAI	VRAI
FAUX	FAUX	FAUX

C1	NON C1
VRAI	FAUX
FAUX	VRAI

Tests imbriqués

- ▶ Les tests peuvent avoir un degré quelconque d'imbrications

Si condition1 alors

Si condition2 alors

instructionsA

Sinon

instructionsB

Finsi

Sinon

Si condition3 alors

instructionsC

Finsi

Finsi

Tests imbriqués: exemple (version 1)

Variable n : entier

Début

Ecrire ("entrez un nombre : ")

Lire (n)

Si (n < 0) alors

Ecrire ("Ce nombre est négatif")

Sinon

Si (n = 0) alors

Ecrire ("Ce nombre est nul")

Sinon

Ecrire ("Ce nombre est positif")

Finsi

Finsi

Fin

Tests imbriqués: exemple (version 2)

Variable n : entier

Début

Ecrire ("entrez un nombre : ")

Lire (n)

Si (n < 0) alors Ecrire ("Ce nombre est négatif")

Finsi

Si (n = 0) alors Ecrire ("Ce nombre est nul")

Finsi

Si (n > 0) alors Ecrire ("Ce nombre est positif")

Finsi

Fin

Remarque : dans la version 2 on fait trois tests systématiquement alors que dans la version 1, si le nombre est négatif on ne fait qu'un seul test

Conseil : utiliser les tests imbriqués pour limiter le nombre de tests et placer d'abord les conditions les plus probables

Tests imbriqués: exercice

Le prix de photocopies dans une reprographie varie selon le nombre demandé:

- 0,5 euros la copie pour un nombre de copies inférieur à 10,
- 0,4 euros pour un nombre compris entre 10 et 20,
- et 0,3 euros au-delà.

Ecrivez un algorithme qui demande à l'utilisateur le nombre de photocopies effectuées, qui calcule et affiche le prix à payer

Tests imbriqués: corrigé de l'exercice

Variables copies : entier
 prix : réel

Début

Ecrire ("Nombre de photocopies : ")

Lire (copies)

Si (copies < 10) **Alors**

 prix ← copies*0.5

Sinon Si (copies < 20) **Alors**

 prix ← copies*0.4

Sinon

 prix ← copies*0.3

Finsi

Finsi

Ecrire ("Le prix à payer est : ", prix)

Fin