

Algorithmique et Programmation

Cours A112

Christel DARTIGUES-PALLEZ

dartigue@unice.fr



Déroulement du module

- ◆ 6 semaines de 8h de cours/TD

- ◆ Objectifs du module

- Base de la programmation, variables, constantes, conditionnelles, itérations, tableaux, pointeurs, gestion des erreurs, tests, compilation simple
- Apprentissage du langage C
- Logiciel utilisé: CodeBlocks

- ◆ Notation

- Interrogations sur les notions vues en cours, QCM, exercices sur machine notés, participation aux TD, examen final sur papier

- ◆ Appel à projet.....

Introduction

◆ Informatique

- Contraction des mots **information** et **automatique**

◆ Ordinateur

- Traduction du terme Computer, qui signifie calculateur
- Signifie Automate programmable
 - ◆ Machine qui accomplit, dans un certain ordre, les opérations pour lesquelles elle a été conçue

Introduction

◆ Ordinateur

- Machine bête
 - ◆ Ne fait qu'obéir à des ordres
- **Mais** machine puissante et rapide
 - ◆ Grâce aux programmes
 - On donne à l'avance la séquence des ordres à effectuer
- Ne comprend que les ordres codés en binaires (langage machine)

Introduction

◆ Ordinateur

- Utilise des données contenues dans des fichiers
- Utilise des programmes
 - ◆ Qui s'exécutent avec des logiciels
 - ◆ Qui sont écrits dans des langages spécifiques

◆ Algorithme

- Suites d'instructions qui conduisent à un résultat donné

◆ Langage de programmation

- Convention pour donner des ordres à un ordinateur
 - ◆ Correspond à une traduction de l'algorithme

Introduction

◆ Algorithme

- Analogue à des situations courantes
 - ◆ Instructions d'une recette de cuisine
 - ◆ Indications pour donner un chemin
 - ◆ Instructions d'un mode d'emploi
 - ◆ Instructions d'une notice de montage d'un meuble

◆ Langage de programmation

- Convention d'écriture des instructions d'un algorithme pour que l'ordinateur les comprenne

Introduction

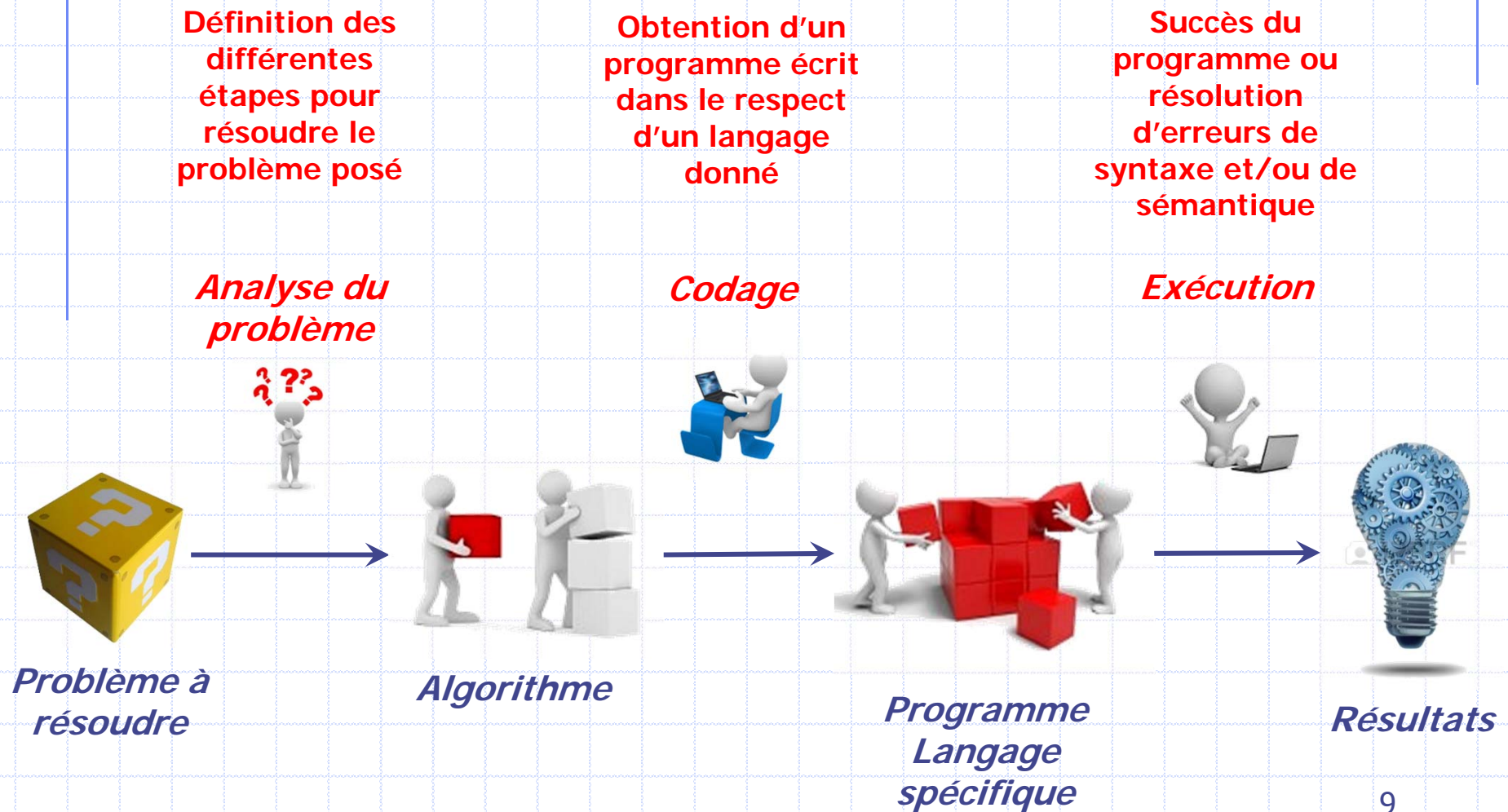
◆ Qualités pour l'algorithmique

- Avoir de l'intuition
 - ◆ Pour trouver les instructions pour résoudre le problème
- Avoir de la rigueur et être méthodique
 - ◆ Se mettre à la place de la machine
 - ◆ Utiliser le crayon et la feuille de papier!!!! (au moins au début...)
 - ◆ Simulation de l'algorithme pour le vérifier (au moins au début)
- Rien à voir avec les maths

Introduction

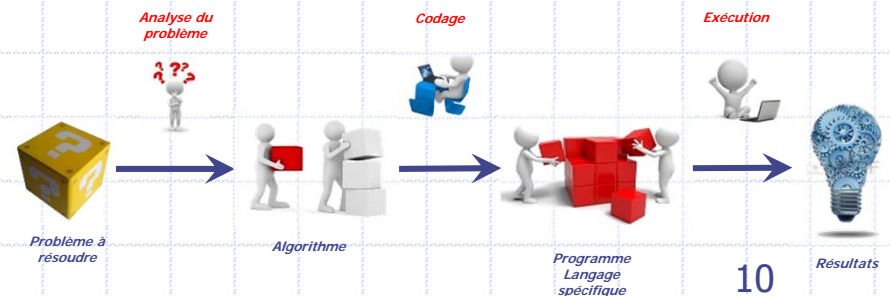
- ◆ Complexité d'un programme \neq de sa longueur
- ◆ Un algorithme est indépendant de tout langage
- ◆ Analogie
 - Algorithmique = plan d'une dissertation (sans rédaction ni orthographe)
 - ◆ On évite les problèmes spécifiques aux langages (syntaxe, types d'instructions, etc.)
 - Programme informatique = dissertation dans la langue voulue

Étapes pour résoudre un problème



Analyse du problème

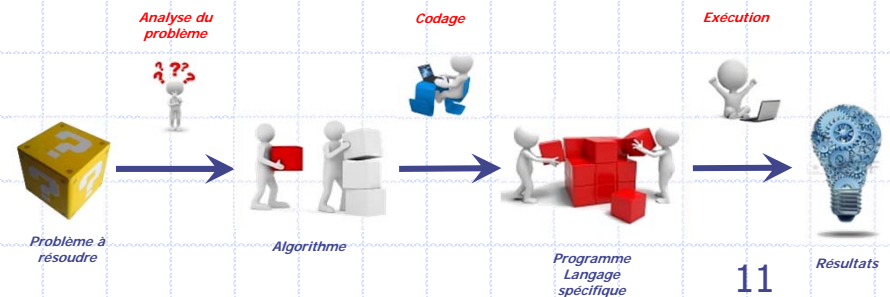
- ◆ Identifier les informations dont on dispose
- ◆ Identifier le(s) résultat(s) qu'il faut fournir
- ◆ Identifier les étapes qui permettent de passer des informations de départ au(x) résultat(s)



Analyse du problème



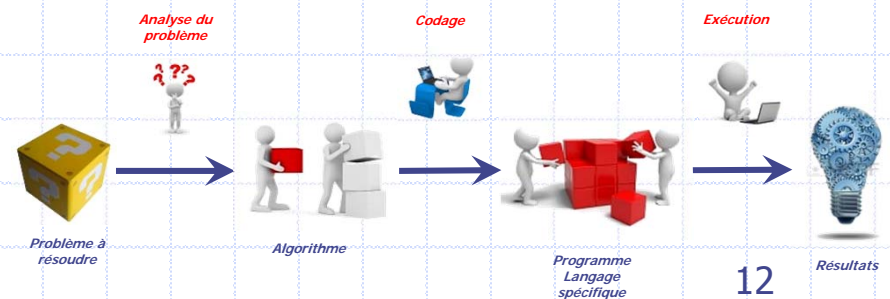
- Se faire livrer une pizza
- ◆ Identifier les informations dont on dispose
- ◆ Identifier le(s) résultat(s) qu'il faut fournir
- ◆ Identifier les étapes qui permettent de passer des informations de départ au(x) résultat(s)



Analyse du problème



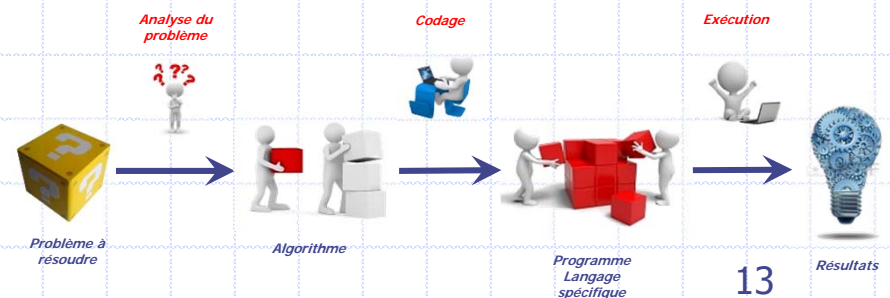
- Se faire livrer une pizza
- ◆ Identifier les informations dont on dispose
 - Combien de personnes ont faim?
- ◆ Identifier le(s) résultat(s) qu'il faut fournir
- ◆ Identifier les étapes qui permettent de passer des informations de départ au(x) résultat(s)



Analyse du problème



- Se faire livrer une pizza
- ◆ Identifier les informations dont on dispose
 - Combien de personnes ont faim?
- ◆ Identifier le(s) résultat(s) qu'il faut fournir
 - Des pizzas!
- ◆ Identifier les étapes qui permettent de passer des informations de départ au(x) résultat(s)




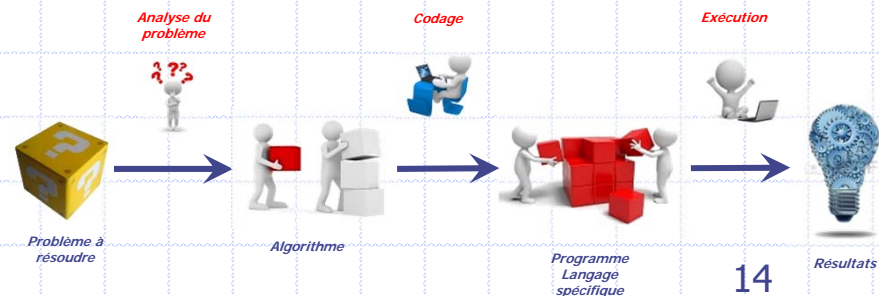
Analyse du problème



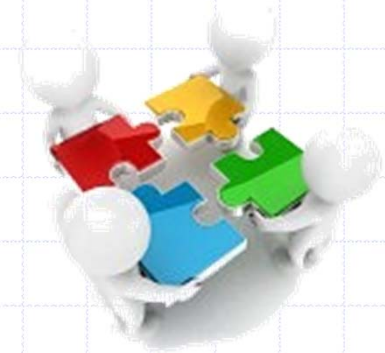
- Se faire livrer une pizza

◆ Identifier les étapes qui permettent de passer des informations de départ au(x) résultat(s)

- Demander à chaque personne combien de pizza elle veut
- Demander à chaque personne ce qu'elle veut sur sa pizza
- Trouver la liste des livreurs de pizza
- Choisir le livreur de pizza
- Appeler le livreur de pizza
- Passer sa commande
- Donner son adresse
- Attendre le livreur.....
- Ouvrir la porte au livreur
- Vérifier que la commande est correcte
- Payer le livreur
- MANGER!

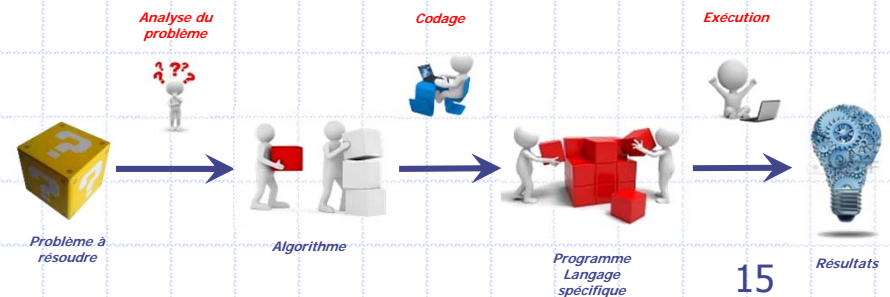


Analyse du problème

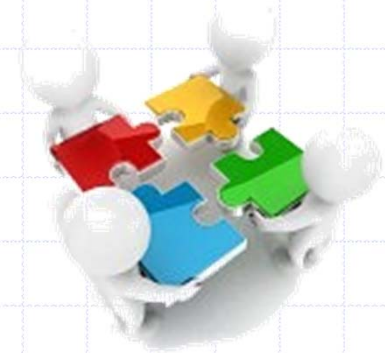


◆ Approche descendante

- Décomposition d'un problème en sous-problèmes plus simples à résoudre
 - ◆ On décompose le problème en sous-problèmes
 - ◆ On résout chacun des sous-problèmes
 - ◆ On combine les solutions des sous-problèmes

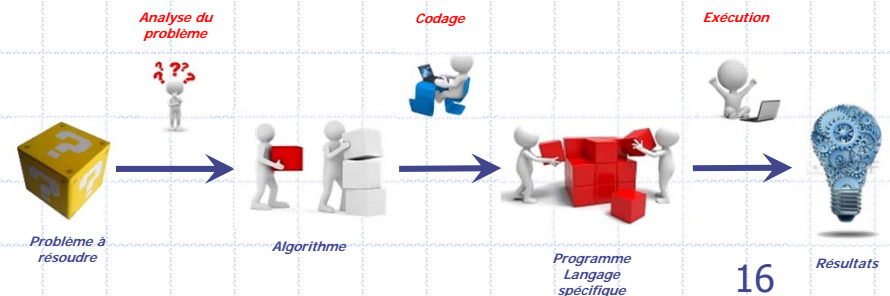


Analyse du problème



◆ Approche descendante

- Décomposition d'un problème en sous-problèmes plus simples à résoudre
 - ◆ Tourner un film
 - ◆ On décompose le problème en sous-problèmes
 - ◆ On résout chacun des sous-problèmes
 - ◆ On combine les solutions des sous-problèmes



Analyse du problème



◆ Approche descendante

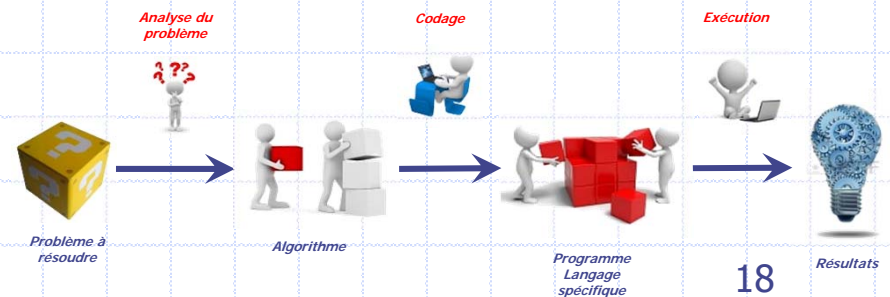
- Décomposition d'un problème en sous-problèmes plus simples à résoudre
 - ◆ Tourner un film
 - ◆ On décompose le problème en sous-problèmes
 - Trouver l'histoire
 - Trouver les financements
 - Trouver les participants
 - Acteurs
 - Techniciens
 - Tourner le film
 - Faire la post-production
 - Distribuer et promouvoir le film
 - ◆ On résout chacun des sous-problèmes
 - ◆ On combine les solutions des sous-problèmes



Analyse du problème

◆ Approche ascendante

- Faire émerger des modèles à partir des données
- Beaucoup plus complexe



Quelques exercices...

- ◆ Écrire un algorithme qui permet de lire le nom, le prénom et l'âge d'une personne et qui les affiche à l'écran
 - Identifier les informations dont on dispose
 - Identifier le(s) résultat(s) qu'il faut fournir

Quelques exercices...

- ◆ Écrire un algorithme qui permet de lire le nom, le prénom et l'année de naissance d'une personne ainsi que l'année courante et affiche à l'écran l'âge de cette personne
 - Identifier les informations dont on dispose
 - Identifier le(s) résultat(s) qu'il faut fournir

Quelques exercices...

- ◆ Écrire un algorithme qui permet de calculer la somme des n premiers entiers
 - Identifier les informations dont on dispose
 - Identifier le(s) résultat(s) qu'il faut fournir

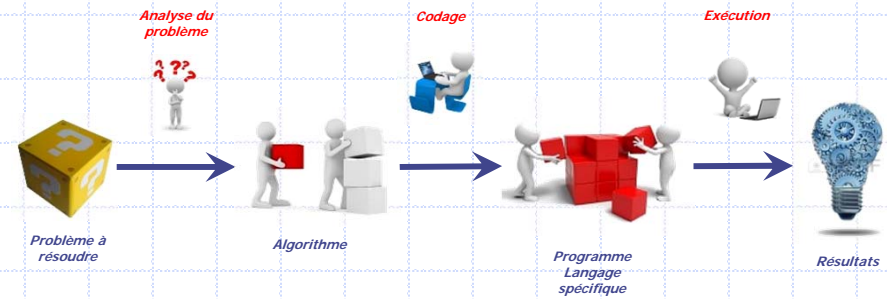
Quelques exercices...

- ◆ Écrire un algorithme qui détermine si un nombre est premier ou non
 - Identifier les informations dont on dispose
 - Identifier le(s) résultat(s) qu'il faut fournir

Quelques exercices...

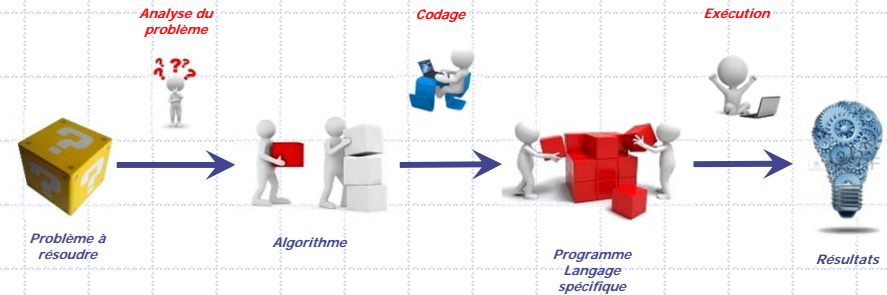
- ◆ Écrire un algorithme qui permet d'afficher à l'écran un rectangle composé d'étoiles de largeur larg et de longueur long
 - Identifier les informations dont on dispose
 - Identifier le(s) résultat(s) qu'il faut fournir

Codage



- ◆ Traduction de l'algorithme
- ◆ Utilisation d'un langage particulier
 - Respect strict de la syntaxe du langage
 - Prise en compte des spécificités du langage

Exécution



◆ 3 cas possibles

- Tout fonctionne comme prévu
- Le programme ne se lance pas
 - ◆ Erreur de syntaxe
 - ◆ Assez facile à résoudre
- Le programme a un comportement « bizarre »
 - ◆ Erreur sémantique: la suite d'instruction n'est pas correcte
 - ◆ Revoir l'algorithme
 - ◆ Beaucoup plus difficile à résoudre

Difficulté des projets informatiques

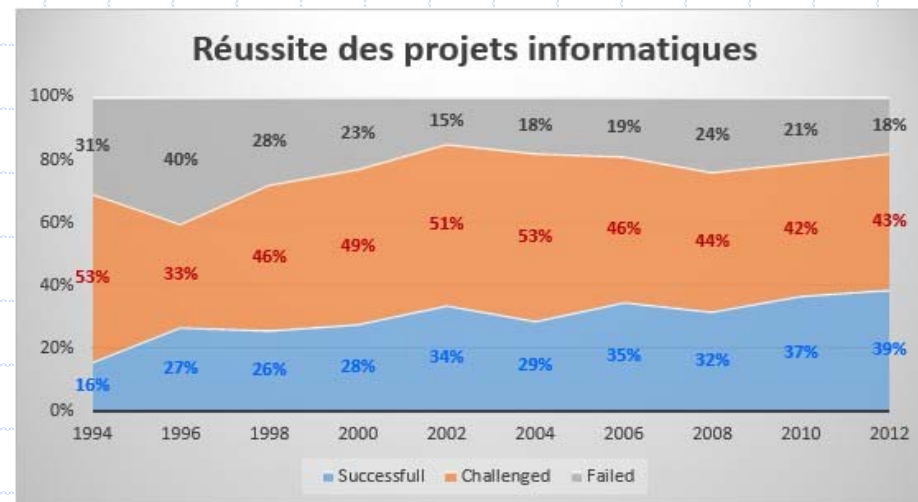
◆ Petits projets

- 3/4 en succès, 4 % en échec,

◆ Grands projets

- 1 fois sur dix en succès, arrêté avant son terme deux fois sur cinq ...

◆ Résultat similaire si l'on considère la taille de l'équipe



Source : <http://blog.sodifrance.fr/it-success-failure-story-check-point-2014/>

Les données manipulées

◆ Besoin de manipuler des objets dans des programmes

- Pour faire des calculs
- Pour écrire des mots
- Pour afficher des graphismes
- ...

◆ Définition

- Littéral: valeur donnée explicitement dans le code source d'un programme
 - ◆ 12, 5.66, "voici du texte", `)` , ...

Les variables

◆ Pourquoi des variables?

- Pour stocker provisoirement des valeurs lors de l'exécution d'un programme informatique

◆ Différentes données

- Issues du disque dur
- Saisies par l'utilisateur au clavier
- Résultats obtenus par le programme
 - ◆ Intermédiaires
 - ◆ Définitifs

Les variables

◆ Différents types de données

- Nombres
- Texte
- Etc.

◆ Variable \approx boîte

- Contenu de la boîte = valeur de la variable
- Étiquette pour avoir accès au contenu de la boîte = nom de la variable

◆ Pour l'ordinateur

- Emplacement en mémoire repéré par une adresse binaire (10011001...)
 - ◆ Numéro de la boîte, du tiroir, du sac, ...



Les variables

◆ Déclaration de variables

- ≈ Créer une boîte et lui associer une étiquette (un nom)
- ≈ prendre un sac à dos et lui donner un nom
- Nom de variable
 - ◆ Contient des lettres et des chiffres (en général commence par une lettre)
 - ◆ Ne contient en général pas de signes de ponctuation et **jamais d'espace**



- Contenu de la boîte
 - ◆ Détermine quel type de données on veut mettre dans la variable
 - La taille de l'emplacement mémoire dépend du type de la variable : Caractère, Entier, Décimal, Chaîne de caractère, ...
 - ◆ Détermine l'ensemble des valeurs que peut prendre la variable
 - ◆ Détermine les opérations qu'on peut effectuer sur la variable
- Le type d'une variable ne peut pas être changé

Variables vs constantes



◆ Constantes

- Données dont la valeur ne peut pas être modifiée

◆ Variables

- Données dont la valeur peut être modifiée

◆ Sont stockées toutes les 2 dans la mémoire de l'ordinateur

Les types de variable

- ◆ Détermine la taille de la boîte et le type de codage utilisé
- ◆ Types simples
 - Entier
 - Réel
 - Caractère
 - Booléen
- ◆ Types composés
 - Structure
 - Tableau
 - ...

Les variables numériques

- ◆ Les types numériques en programmation
 - Différents types numériques existent
 - Permettent de représenter des nombres plus ou moins grands
- ◆ Le type de codage détermine
 - Les valeurs minimales et maximales des nombres qui peuvent être stockés
 - La précision de ces nombres (pour les nombres décimaux)
 - Varie d'un langage à un autre

Type Numérique	Plage	Exemple
Byte (octet)	0 à 255	1 nombre codé avec 1 octet: $2^8 = 256$ valeurs différentes De 1 à 256, ou de 0 à 255, ou de -127 à +128, ...
Entier simple	-32 768 à 32 767	1 nombre codé avec 2 octets: $2^{16} = 65\,536$ valeurs différentes, ...
Entier long	-2 147 483 648 à 2 147 483 647	
Réel simple	-3,40x10 ³⁸ à -1,40x10 ⁴⁵ pour les valeurs négatives 1,40x10 ⁻⁴⁵ à 3,40x10 ³⁸ pour les valeurs positives	Source : http://pise.info/algo/codage.htm
Réel double	1,79x10 ³⁰⁸ à -4,94x10 ⁻³²⁴ pour les valeurs négatives 4,94x10 ⁻³²⁴ à 1,79x10 ³⁰⁸ pour les valeurs positives	

Les variables numériques

- ◆ Attention aux calculs avec des nombres décimaux!
 - La comparaison de 0,1 et 1/10 renvoie que les 2 valeurs sont différentes...
 - Problème du au mode de représentation des nombres flottants

Les variables

- ◆ Autres types numériques utilisés en programmation
 - Type monétaire
 - Type date (jour/mois/année)
- ◆ Les types numériques
 - Toujours choisir le type de données le plus approprié
 - Éviter de gaspiller les ressources de la machine!
 - ◆ Ralentissement de l'ordinateur
 - ◆ Plantage du programme
- ◆ En algorithmique
 - Moins exhaustif: **entier, réel**

Les variables alphanumériques

◆ Un caractère contient

- Une lettre
- Un chiffre
- Un signe de ponctuation
- Un espace
- Un caractère spécial

◆ Toujours noté entre simples cotes

- 'a', '5', '?', ' _ ', '#', ...
- Attention à la confusion entre 1 et '1'

Les variables alphanumériques

◆ Une chaîne de caractères contient

- Des lettres
- Des chiffres
- Des espaces
- Des signes de ponctuation

◆ Représentation

- "Toto", "Toto et Titi", "12345", "4f:er45", ...

◆ Attention aux confusions

- 12345 est un nombre, "12345" est une chaîne de caractères
- "12345" est le contenu d'une variable, pas le nom de cette variable

Les variables

◆ Les booléens

- Contient uniquement les valeurs **logiques** VRAI et FAUX
- Autres notations
 - ◆ TRUE et FALSE
 - ◆ 1 et 0
- Prend très peu de place en mémoire
- Pas indispensable MAIS améliore énormément la **lisibilité** des programmes

Les variables - déclaration

◆ Booléen estCorrect



◆ Caractère c

◆ Entier A



◆ Réel x

◆ Chaîne de caractères ch



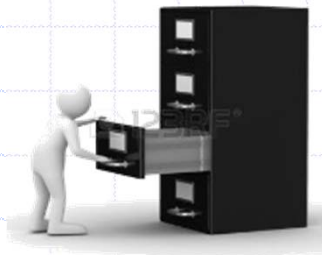
- Essayer de se fixer une convention de nommage

Les briques de base

- ◆ 4 ordres/instructions compréhensibles par les ordinateurs
 - L'affectation de variables
 - La lecture / écriture
 - Les tests
 - Les répétitions (boucles)
- ◆ Programme = combinaison de ces 4 briques de base

L'affectation

- ◆ Seule action possible avec une variable: lui affecter une valeur
- ◆ Structure d'une affectation
 - `NomVariable ← expression ;`
 - ◆ Dans certains langages de programmation :
`NomVariable = expression`
- ◆ \approx On remplit le sac à dos
- ◆ \approx On range quelque chose dans la boîte



L'affectation

◆ L'affectation

■ Exemple 1

- ◆ $\text{VarA} \leftarrow 24 ;$
- ◆ On attribue la valeur 24 à la variable VarA
- ◆ Implique que VarA soit une variable de type numérique

■ Exemple 2

- ◆ $\text{VarB} \leftarrow \text{VarA} ;$
- ◆ La valeur de VarB est maintenant celle de VarA
- ◆ La valeur de VarA n'est absolument **pas** modifiée

L'affectation

◆ L'affectation

■ Exemple 3

- ◆ $\text{VarB} \leftarrow \text{VarA} + 4 ;$
- ◆ Si VarA contenait la valeur 10, alors VarB contient 14
 - La valeur de VarA n'est absolument **pas modifiée**

■ Exemple 4

- ◆ $\text{VarB} \leftarrow \text{VarB} + 2 \times \text{VarA} ;$
- ◆ Si VarB valait 5 et VarA valait 2, alors VarB vaut maintenant 9
- ◆ La valeur de VarB est **modifiée**

L'affectation

```
VarA ← "VarB" ;  
VarC ← "VarA" ;
```

- Valeur de VarA
 - ◆ VarB
- Valeur de VarC
 - ◆ VarA

```
VarA ← "VarB" ;  
VarC ← VarA ;
```

- Valeur de VarA
 - ◆ VarB
- Valeur de VarC
 - ◆ VarB

- Ordre des instructions

Début

A ← 34 ;

A ← 12 ;

Fin

Début

A ← 12 ;

A ← 34 ;

Fin

- L'ordre est essentiel
- Attention à la distinction variable/chaîne de caractères

L'affectation

◆ Left value ← right value

■ Lvalue

- ◆ Signifie « le contenu de ... »
- ◆ Une variable de type Lvalue est définie comme étant le contenu d'un emplacement mémoire
- ◆ Est associé à une adresse mémoire, un type, un nom et une valeur

■ Rvalue

- ◆ Signifie « la valeur de ... »

L'affectation

◆ Left value ← right value

■ Sont des Lvalues

- ◆ Les variables simples
 - De type numérique, pointeurs, ...
- ◆ Les variables de type enregistrement

■ Ne sont pas des Lvalues

- ◆ Les constantes
- ◆ Les littéraux
- ◆ Les fonctions
- ◆ Les expressions
 - Ex: $(A + X) * 2$

=> Ce sont des Rvalues

L'affectation

◆ Attention

- \approx On ne peut pas faire rentrer dans un sac quelque chose de la taille d'une valise...

◆ On ne peut affecter une valeur à une variable que si les valeurs sont de **même type** ou de **type équivalent**

Quelques exercices...

- ◆ Écrire les instructions qui :
- ◆ Définissent 2 variables entières nommées A et B
- ◆ Définissent 2 variables caractères nommées c1 et c2
- ◆ Affectent 2 et 10 à A et B
- ◆ Affectent la valeur A à c1 et la valeur de A à c2

Quelques exercices...

- ◆ Que valent les variables à la fin des instructions? Ces instructions génèrent-elles des erreurs? (si oui pourquoi)

Entier A, B, C

Caractères D

A ← 10

B ← 15

C ← A x 3 - B x 2

C ← C x A x B

D ← "ABC"

A ← D

Quelques exercices ...

◆ Déterminez les instructions qui sont correctes et celles qui ne le sont pas

Entier A, B

Réel x, y

Réel constante $PI \leftarrow 3,14$

Caractère $c1, c2$

Chaîne de caractères ch

$A \leftarrow 2$

$B \leftarrow PI$

$x \leftarrow A$

$PI \leftarrow 3,1418$

$y \leftarrow PI$

$(A+B) \times 2 \leftarrow y$

$'coucou' \leftarrow c1$

$ch \leftarrow 'A'$

$ch \leftarrow "coucou"$

Les variables et leurs opérateurs

Les opérateurs dépendent des types des variables/littéraux considérés

◆ Entiers

■ Opérateurs utilisables

- ♦ opérations : +, -, *, div, mod, *, ++ --
- ♦ comparaisons : ==, !=, <, <=, >, >=

◆ Réels

■ Opérateurs utilisables

- ♦ opérations : +, -, *, /
- ♦ comparaisons : ==, !=, <, <=, >, >=
- ♦ Attention au == entre 2 nombres à virgule!!

◆ Caractères

■ Opérateurs possibles

- ♦ opérations : +
- ♦ comparaisons : ==, !=, <, <=, >, >=

◆ Booléens

■ Opérateurs possibles

- ♦ Et, ou, non

Les opérateurs

◆ Les opérateurs numériques

- + : addition
- - : soustraction
- * : multiplication
- / : division
- mod : reste de la division ($5 \bmod 2 = 1$)
- div : quotient de la division entière ($5 \text{ div } 2 = 2$)

◆ Les opérateurs alphanumériques

- Opérateur de concaténation: +
- Exemple

```
A ← "Bon" ;  
B ← "jour" ;  
C ← A + B ;
```

```
// C = Bonjour
```

Les opérateurs numériques

◆ Écriture compacte

- $++$, $--$, $+=$, $-=$, $*=$, $/=$
 - ◆ $X++$ équivalent à $X = X + 1$
 - ◆ $X--$ équivalent à $X = X - 1$
 - ◆ $++X$ équivalent à $X = X + 1$
 - ◆ $--X$ équivalent à $X = X - 1$
 - ◆ $X += 2$ équivalent à $X = X + 2$
- Attention à ce qu'on écrit!

$Y = 2$
 $X = Y++$

n'est pas équivalent à

$Y = 2$
 $X = ++Y$

Les opérateurs

◆ Les opérateurs de comparaison

- = : égal à
 - ◆ Ou == dans certains langages de programmation
- != : différent de
 - ◆ Ou <>
- < : strictement inférieur à
- ≤ : inférieur ou égal à
- > : strictement supérieur à
- ≥ : supérieur ou égal à

Les opérateurs

◆ Les opérateurs booléens

- ET (&&, \wedge , and, ...)
- OU (||, \vee , or, ...)
- NON (!, \neg , not, ...)

A	B	A && B
VRAI	VRAI	VRAI
VRAI	FAUX	FAUX
FAUX	VRAI	FAUX
FAUX	FAUX	FAUX

A	B	A B
VRAI	VRAI	VRAI
VRAI	FAUX	VRAI
FAUX	VRAI	VRAI
FAUX	FAUX	FAUX

A	! A
VRAI	FAUX
FAUX	VRAI

Les opérateurs

◆ Involution

- $!!a = a$

◆ Loi de Morgan

- $!(a \parallel b) = !a \ \&\& \ !b$

- $!(a \ \&\& \ b) = !a \parallel !b$

◆ Distributivité des opérateurs

- $a \ \&\& \ (b \parallel c) = (a \ \&\& \ b) \parallel (a \ \&\& \ c)$

- $a \parallel (b \ \&\& \ c) = (a \parallel b) \ \&\& \ (a \parallel c)$

Priorité des opérateurs

opérateur	type des opérandes	type du résultat	opération
-	1 entier ou réel	idem opérande	négation
+	2 entiers ou 2 réels ou 1 entier et 1 réel	entier si les 2 opérandes le sont, réel sinon	addition
-		réel	soustraction
×			multiplication
/			division réelle
div mod	2 entiers	entier	quotient entier reste de la division
< ≤ > ≥ = ≠	2 nombres, 2 caractères ou 2 chaînes	booléen	inférieur inférieur ou égal supérieur supérieur ou égal égal différent
non	1 booléen		négation logique
et ou	2 booléens		

Plus forte classe 1 non – unaire
 classe 2 et × / div mod
 classe 3 ou + -
 Plus faible classe 4 = ≠ < ≤ > ≥

TAB. 3 – Priorité des opérateurs

Source: <http://christophe.deleuze.free.fr/esisar05/CS110/fiche.pdf>

Quelques exercices ...

- ◆ Pour chacun des énoncés suivants, identifiez par un nom les proposition élémentaires et transformez l'énoncé en forme logique
 - Ou bien je n'ai pas compris la blague ou bien il est très mauvais pour raconter des blagues
 - Propositions élémentaires
 - ◆ A: j'ai compris la blague
 - ◆ B: il est mauvais pour raconter des blagues
 - Forme logique de l'énoncé
 - ◆ $\neg A \vee B$
 - Variante possible
 - ◆ A: j'ai compris la blague
 - ◆ B: il est très bon pour raconter des blagues
 - ◆ $\neg A \vee \neg B$
 - ◆ $\neg(A \wedge B)$

Quelques exercices ...

- ◆ Pour chacun des énoncés suivants, identifiez par un nom les propositions élémentaires et transformez l'énoncé en forme logique
 - Ni Anne ni Cathy n'ont échoué à leur examen

Quelques exercices ...

- ◆ Pour chacun des énoncés suivants, identifiez par un nom les propositions élémentaires et transformez l'énoncé en forme logique
 - Anne et Cathy ne réussiront pas toutes les 2

Quelques exercices ...

- ◆ Pour chacun des énoncés suivants, identifiez par un nom les proposition élémentaires et transformez l'énoncé en forme logique
 - Les personnes qui ont droit à une réduction sont les hommes de plus de 40 ans, les femmes entre 18 ans et 35 ans.
 - ◆ Homme et >40 OU femme et $(>18 \text{ et } <35)$

Quelques exercices ...

- ◆ Pour chacun des énoncés suivants, identifiez par un nom les proposition élémentaires et transformez l'énoncé en forme logique
 - Ralf n'est pas idiot, mais il fait du bruit et il dérange les autres

Quelques exercices ...

- ◆ Pour chacun des énoncés suivants, identifiez par un nom les proposition élémentaires et transformez l'énoncé en forme logique
 - Les cours se passent bien si les étudiants travaillent, ne jouent pas sur leur ordinateur, ne discutent pas et si l'enseignant est intéressant

Quelques exercices ...

- ◆ Pour chacun des énoncés suivants, identifiez par un nom les proposition élémentaires et transformez l'énoncé en forme logique
 - Les étudiants sont concentrés le matin sauf les lendemains de fêtes étudiant

Quelques exercices ...

- ◆ Pour chacun des énoncés suivants, identifiez par un nom les propositions élémentaires et transformez l'énoncé en forme logique
 - Je peux traverser le carrefour si le feu est vert, si le feu est orange et qu'il est trop avancé pour pouvoir s'arrêter, certainement pas si le feu est rouge

Quelques exercices ...

◆ Soient A la proposition "Lucie aime les chiens" et B la proposition "Les chiens aiment Lucie". Pour chacune des formules qui suivent trouvez l'énoncé en français correspondant le plus simple possible:

- $(\neg A) \wedge B$
- $\neg(A \wedge B)$
- $\neg(A \vee B)$
- $A \vee \neg B$
- $\neg(\neg A \wedge B)$

Quelques exercices...

◆ Quel est le type de chacune des expressions suivantes

Entier A, B, C

Réel X, Y, Z

Booléen Bool

X

Y + 3.1444

56

A ← C

A == C

A + 6

Z + 6

A--

X < 2

A > 0 && A < 10

X * (A + 3)

A / 2

(10 > 0) && Bool

A == B && Bool

Lecture - écriture

- ◆ Instruction qui permettent à l'utilisateur d'interagir avec le programme

- ◆ Lecture

- Permet de saisir des valeurs dans des variables
- Les valeurs saisies sont tapées au clavier

- ◆ Écriture

- Permet d'afficher des valeurs dans des variables
- Les valeurs écrites sont affichées à l'écran

Lecture - écriture

- ◆ On peut lire
 - Des variables (Lvalues): nombre, texte, ...
- ◆ On peut écrire
 - Des variables (Lvalues)
 - Des expressions
 - Des données fournies par des fonctions
- ◆ Ces opérations sont typées
 - Il faut formater ce qu'on souhaite lire/écrire
 - Lire un entier n'est pas la même chose que lire un texte

Lecture - écriture

◆ Syntaxe

```
ecrire ("type", laVariable)
```

```
lire ("type", laVariable)
```

◆ Exemple

```
Algorithme ExempleLectureEcriture
```

```
Rôle: ....
```

```
Variables : a, b entier
```

```
Début
```

```
    lire ("entier", a) ;  
    b <- a + 3 ;  
    écrire ("entier", a) ;  
    écrire ("entier", b) ;
```

```
Fin
```

Écrire un algorithme

- ◆ Convention pour écrire un algorithme
 - Utilisation d'un pseudo-code
 - Ressemble à un langage de programmation
 - N'est pas interprétable par un ordinateur

◆ Exemple

Indique l'objectif visé avec cet algorithme

Algorithme ajouterDeuxEntiers

Indique le début d'un algorithme. Il est obligatoire de lui donner un nom

Rôle: additionner deux entiers a et b et mettre le résultat dans c

Indique la section de déclaration des variables utiles pour l'algorithme

Variables :

Début

Fin

c ← a + b ;

Marque le début de l'algorithme

Marque la fin de l'algorithme

Marque la fin d'une instruction

Nom	Rôle	Type
a		entier
b		entier
c		entier

Quelques exercices...

- ◆ Quelles seront les valeurs des variables a et b après exécution des instructions suivantes?

Algorithme Exemple_1

Rôle:

Variables : a, b entier

Début

a ← 1 ;

b ← a + 3 ;

a ← 3

Fin

Nom	Rôle	Type	Valeur finale
a		entier	
b		entier	

Quelques exercices...

- ◆ Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes?

Algorithme Exemple_2

Rôle:

Variables : a, b, c entier

Début

```
a <- 3 ;  
b <- 10 ;  
c <- a + b ;  
b <- a + b ;  
a <- c ;
```

Fin

Nom	Rôle	Type	Valeur finale
A		entier	
B		entier	
C		entier	