

Développement Web (R112)

MMI, IUT1, UGA

Gwen Salaün

L'équipe enseignante

- Gwen Salaün (CM, TD, TP)
- Alix Goguey (TD, TP)
- Pierre-Alain Jacquot (TD, TP)



Objectif (réf. de formation)

Initier les étudiants aux bases de l'algorithmique et de la programmation en les illustrant avec la génération de pages HTML

Mise en œuvre

- CM : $6 \times 1h$
- TD : $3 \times 2h = 6h$
- TP : $4 \times 2h = 8h$

=> Les supports de CM, TD, TP seront disponibles sur moodle

Mise en œuvre détaillée

- CM
 - 3h d'algorithmique
 - 1h de PHP
 - 1h de révisions
 - 1h pour corriger l'examen de l'année passée
- TD : exercices sur papier et correction au tableau
- TP : implémentation avec PHP (utilisation de PHPStorm)

Évaluation

Un unique examen écrit : 1h30 à la fin du module
(décembre)

Algorithmique

(cours 1)

Plan

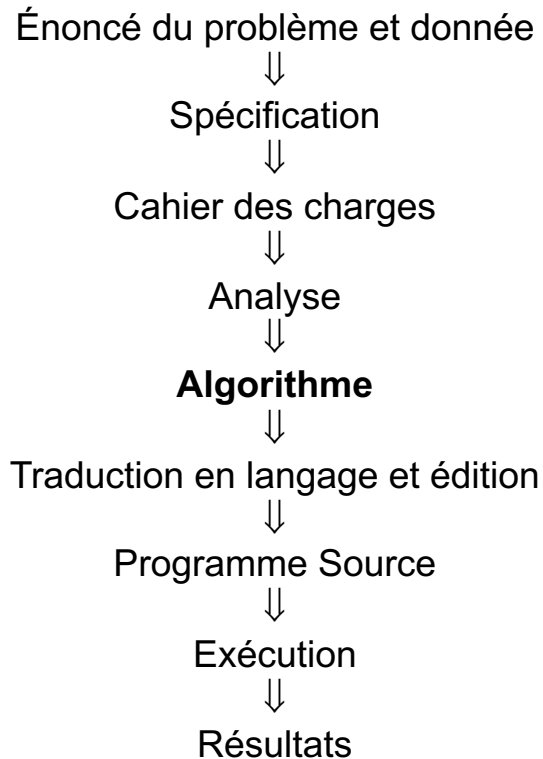
- Programmation
 - Algorithmique
 - Notion de variable
 - L'instruction d'affectation
 - Les instructions de lecture/écriture
 - Le choix
 - Les boucles
 - Factorisation du code
-
- Cours 1
- Cours 2
- Cours 3
- Cours 4

Programmation

- Un microprocesseur ne sait exécuter que du code machine.
- Code peut lisible, difficile à maintenir => utiliser des langages de programmation de haut niveaux
- De nombreux langages de différents types (impératifs, fonctionnels, déclaratifs, objets, concurrents, ...)
 - Pascal, C, ADA
 - Java, C++, Python
 - Caml, Lisp
 - PHP, Javascript
- Mais avec un point commun

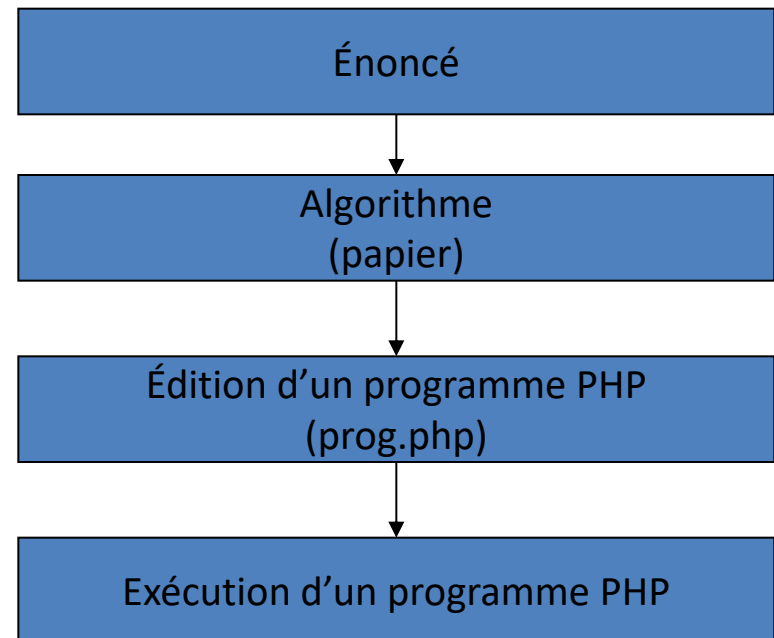
Programmation

Exemple de réalisation d'un programme par analyse descendante avant un langage interprété :



Pour nous cette année :

Traduction algorithmique d'un problème puis traduction en langage PHP, compilation et exécution



Algorithmique

- Définition : Un algorithme est un moyen pour un humain de présenter la résolution par calcul d'un problème à une autre personne physique (un autre humain) ou virtuelle (un calculateur).
- Un algorithme est un énoncé dans un langage bien défini d'une suite d'opérations permettant de résoudre par calcul un problème.
- Si ces opérations s'exécutent en séquence, on parle d'**algorithme séquentiel**. Si les opérations s'exécutent sur plusieurs processeurs en parallèle, on parle d'algorithme parallèle. Si les tâches s'exécutent sur un réseau de processeurs on parle d'algorithme distribué.

Algorithmique

- Utilise une structure logique indépendante du langage
- Deux approches :
 - Organigrammes (lourdeur, manque de structure, passage à l'échelle)
 - Pseudo code ou Pseudo langage

Algorithmique

Algo demo

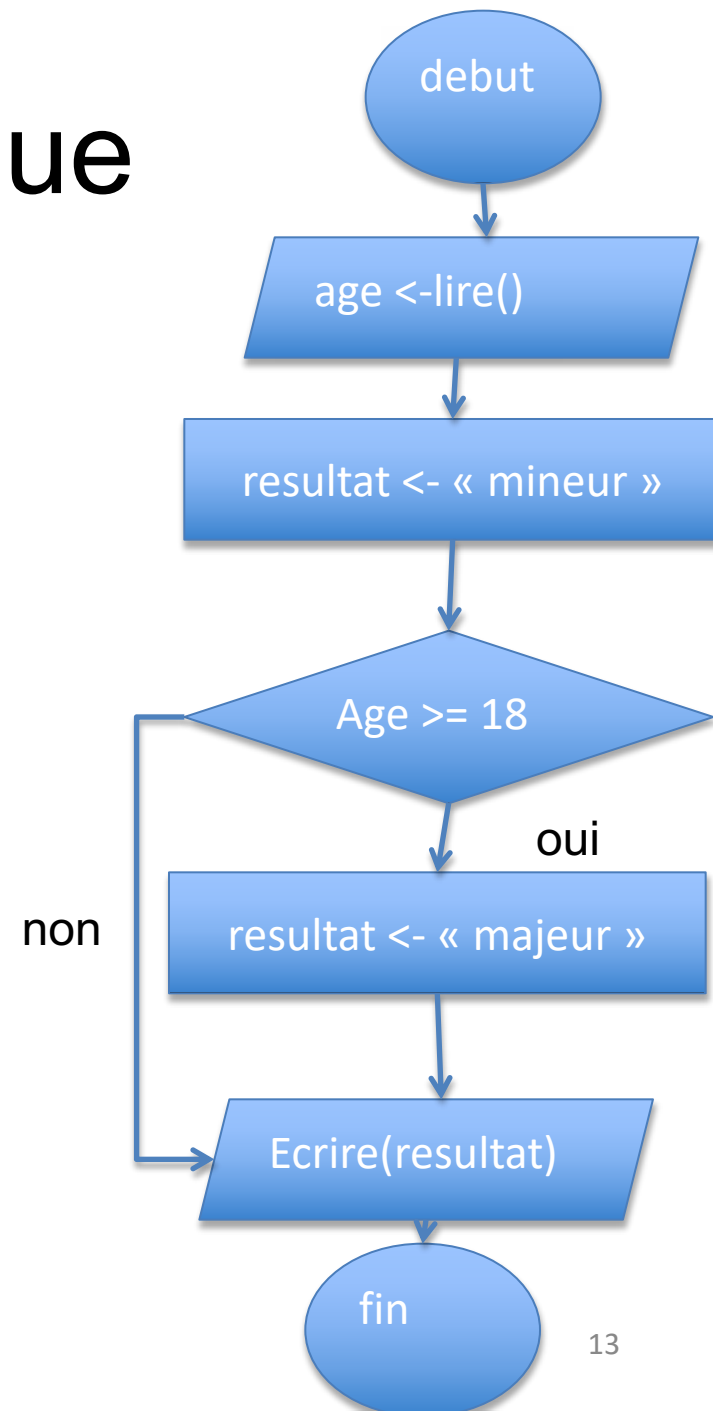
var age : entier
Var resultat : chaine

Début

age <- lire()
resultat <- « mineur »
Si age >= 18 **Alors**
 resultat <- « majeur »

Fin si
Ecrire (resultat)

Fin



Constructions de base

- L'instruction d'affectation de variables
- Les instructions de lecture / écriture
- L'instruction de choix
- Les instructions de boucles
- La factorisation de code avec les fonctions

Syntaxe Pseudo Langage

En pseudo code

Algo nomAlgo

déclarations

Début

instructions

Fin

Syntaxe Pseudo Langage

Bonne Syntaxe

Algo Affect1

var A : entier
var B : entier

Début

A ← 12
B ← A - 10
A ← 2

Fin

Mauvaise Syntaxe

Algo Affect1

var A : entier
var B : entier

A ← 5
B ← A+1
A ← 2

Fin

Fin

Commentaires

- Tout code doit être commenté

// commentaire

// commentaire

Variable

Les variable peuvent être vues comme des boîtes avec des étiquettes (leurs noms) qui ne peuvent contenir qu'un type de donnée.

Il faut avant d'utiliser une variable, créer la boîte et lui coller l'étiquette, c'est une **déclaration**.

(var a: entier)



À un instant donné une variable possède une valeur et une seule, elle ne possède **pas d'historique**.

Une variable qui ne peut changer de valeur est dite **constante**.

Variables – types

Types en Pseudo code

- Entier (1,-1,0, ...)
- Réel (0.5, 1, 1.1)
- Booléen (vrai, faux)
- Caractère ('a','0',' ')
- Chaîne (« il fait beau »)

Les variables – affectation

identificateur ← expression

Une instruction d'affectation ne modifie que la partie gauche de la flèche

Une variable utilisée dans une affectation doit être déclarée

En partie gauche d'une affectation ne peut se trouver qu'une variable (pas une autre expression)

Une variable utilisée en partie droite doit avoir été initialisée

Affectation – expression

- Les expressions sont obtenues en appliquant des opérateurs à des variables, des constantes et des littéraux.
- Une expression est interprétée en une valeur.
- Exemples :
 - $2+3$ vaut 5
 - " il fait " . " beau " vaut "il fait beau"
 - $-4+6$ vaut 2

Affectation – exemples (1)

Algo Affect2

var A : entier

Début

A ← 12

A ← 2

Fin

A vaut 2

Algo Affect3

var A : entier

Début

A ← 2

A ← 12

Fin

A vaut 12

Affectation – exemples (2)

Algo Affect4

var A : entier
var B : entier

Début

A \leftarrow 12
B \leftarrow 2

Fin

A vaut 12
B vaut 2

Algo Affect5

var A : entier
var B : entier

Début

A \leftarrow 2
B \leftarrow A * 2

Fin

A vaut 2
B vaut 4

Affectation– exemples (3)

Algo Affect6

var A : entier
var S : entier

Début

A ← 12
S ← 10
S ← S + A + 1

Fin

A vaut 12
S vaut 23

Algo Affect7

var A : entier
var B : entier
var C : entier

Début

A ← 3
B ← 10
C ← A + B
B ← A + B
A ← C

Fin

A vaut ?
B vaut ?
C vaut ?

Affectation– exemples (3)

Algo Affect6

var A : entier
var S : entier

Début

A ← 12
S ← 10
S ← **S** + A + 1

Fin

A vaut 12
S vaut 23

Algo Affect7

var A : entier
var B : entier
var C : entier

Début

A ← 3
B ← 10
C ← A + B
B ← A + B
A ← C

Fin

A vaut 13
B vaut 13
C vaut 13

Au prochain cours

- Les instructions de lecture/écriture
- Le choix

Pour aller plus loin :

- <http://fr.wikipedia.org/wiki/Algorithmique>
- Cours de Christophe Darmangeat
<http://www.pise.info/algo/codage.htm>