



ALGORITHMIE

Jérémy PERROUAULT



INTRODUCTION

Le langage
Son utilisation

INTRODUCTION

Répondre à une problématique

- Problème mathématique
- ...
- Besoin client

INTRODUCTION

Pour répondre à un besoin complexe

- Subdiviser en problèmes plus petits
 - Subdiviser en problèmes plus petits
 - ...
 - Jusqu'à ce que le problème soit assez simple à résoudre
- L'algorithme sera composé de plusieurs instructions, qui s'exécuteront les unes à la suite des autres
- La résolution successive des différents sous-problèmes répondra au problème principal
 - On appelle chaque problème un sous-programme
 - Chaque sous-programme peut attendre des informations (paramètres)
 - Chaque sous-programme peut retourner une valeur (valeur de retour)

INTRODUCTION

Comment choisir de subdiviser ?

- Prendre connaissance des outils à disposition
 - Que sait faire la machine ?
 - Que me propose mon langage de développement ?
 - Quels Frameworks (boîtes à outils) j'ai à ma disposition ?
 - Quelles données je peux utiliser ?
 - ...

Une recette de cuisine est un algorithme !

- Prendre la casserole
- La remplir d'eau
- Y ajouter du sel
- ...

INTRODUCTION

Pour chaque problème, il existe plusieurs façons d'arriver au résultat

- A vous de déterminer le chemin le plus optimisé, le plus court et/ou le plus simple à utiliser

Dans des conditions et avec des données identiques

- Un algorithme donnera toujours le même résultat

INTRODUCTION

Le langage algorithmique est un langage pédagogique, plus « naturel »

Il est détaché de tout environnement de développement

- Un même algorithme peut être adapté pour du C, du C++, du C#, du JAVA, du Python, du PHP, ...



CONCEVOIR UN ALGORITHME

Les types de variables
Les opérations

LES OPÉRATIONS

Une machine est bête, et ne sait pas faire grand-chose

- Mais c'est une idiote qui va très vite, et qui ne rechigne pas aux tâches récurrentes

Une machine sait

- Additionner
- Soustraire
- Multiplier
- Diviser
- Vérifier une condition

LES OPÉRATIONS

Nos algorithmes se baseront sur cette liste d'opérations pour être décrit

Opérations arithmétiques	Opérations de comparaison	Opérations logiques	Conditions	Boucles
+	>	ET	Si	TantQue
-	<	OU	Sinon Si	Faire..TantQue
*	>=	NON	Sinon	Pour
/	<=			
%	==			
	!=			

LES VARIABLES

Type de variable	Définition	Exemple
Vide	Rien	
Entier	Nombre entier	42
Caractère	Caractère alphanumérique	'a'
Réel	Nombre à virgule	12.8764
Booléen	Vrai ou Faux	Faux – ou 0
Tableau		

LES VARIABLES

Une variable désigne un remplacement de la mémoire vive (RAM) dans lequel est stocké sa valeur

- Elle est définie par un type, un nom et une valeur (initiale ou modifiée en cours d'algorithme)

Créons une variable 'a' et affectons-lui la valeur 42

- 'a' fera référence à l'emplacement mémoire dans lequel sa valeur est
- Dans l'exemple, son adresse mémoire est @12

@1	@2	@3	@4	@5	@6
@7	@8	@9	@10	@11	
@12					

Adresse mémoire	Variable	Valeur
@1	??	??
@2	??	??
@3	??	??
...
@12	a	42

LES VARIABLES

Pour créer une variable, il faut la « déclarer »

- **Type** nom La variable n'aura pas de valeur initiale
- **Type** nom = valeur La variable aura une valeur initiale
- **Entier** a a n'a pas de valeur initiale
- **Entier** a = 10 a vaut 10

/!\ Toute variable doit être déclarée /!

LES VARIABLES

Ci-dessous, les autres adresses mémoires sont indisponibles pour nous

- Un autre programme ?
- Un autre sous-programme ?
- Une définition hors-champs ?

@1	@2	@3	@4	@5	@6
@7	@8	@9	@10	@11	
@12					

Adresse mémoire	Variable	Valeur
@1	??	??
@2	??	??
@3	??	??
...
@12	a	42

LES VARIABLES

La déclaration de chaque variable a son importance

- L'emplacement déterminera par qui elle sera accessible, c'est sa portée
- Une variable déclarée dans un bloc d'instructions ne sera accessible que
 - Par ce bloc d'instructions
 - Par les sous-blocs et sous-sous-blocs d'instructions

LES VARIABLES

Dans un sous-programme, il existe des variables qu'on n'appelle pas « variable »

- Un sous-programme est appelé par un autre sous-programme qui peut lui donner des informations
- Dans le sous-programme appelé, on appelle ces informations des paramètres
- On les retrouve dans la signature du sous-programme
 - Un paramètre est déclaré de la même façon qu'une variable : un type et un nom
 - Sa valeur initiale sera défini par le sous-programme appelant
- En transférant un paramètre, une variable est créée, et un nouvel adressage mémoire est fait
 - On dit qu'on transmet le paramètre par « valeur »

LES VARIABLES — TABLEAU

Un tableau est une variable qui contient un ensemble de variables de même type

- Tableau d'Entier
- Tableau de Caractères

On utilise les crochets (`[]`) pour déclarer et manipuler un tableau

Un tableau est donc constitué de n cases, accessibles par un indice

- L'indice commence par 0 (0 est la première case)

-51	24	54	42	678
-----	----	----	----	-----

Case	1	2	3	4	5
Indice	0	1	2	3	4
Valeur	-51	24	54	42	678

LES VARIABLES — TABLEAU

Un tableau a une taille fixée lors de sa création

- Sa taille ne peut pas évoluer

LES VARIABLES — TABLEAU

Type de variables	Déclaration	Initialisation (pour 5 éléments)
Tableau d'entiers	Entier[]	Entier[5]
Tableau de caractères	Caractère[]	Caractères[5]
Tableau de booléens	Booléen[]	Booléen[5]

Pour accéder à l'élément n°2 du tableau « monTab »

- monTab[1] (parce que l'indice commence à 0 : $2 - 1 = 1$)

Attention

- Dans les langages bas-niveau, en dépassant du tableau vous continuez de lire/écrire la mémoire à une adresse qui ne vous appartient probablement plus !
- Dans les langages haut-niveau, une sécurité vous interdira d'aller au-delà du tableau, et vous enverra une erreur « Index out of bounds », ou « indice hors limites »

LES VARIABLES — TABLEAU

-51	24	54	42	678
-----	----	----	----	-----

@1	@2	@3	@4	@5	@6
@7	@8	@9	@10	@11	
@12					

Adresse mémoire	Variable	Valeur
@1	tab[0]	-51
@2	tab[1]	24
@3	tab[2]	54
@4	tab[3]	42
@5	tab[4]	678

LES VARIABLES — CHAÎNE DE CARACTÈRES

Une chaîne de caractères n'existe pas en tant que telle, mais on sait que

- C'est une suite de caractères
- Le type Caractère existe
- On peut regrouper un ensemble de variables de même type dans un tableau

Une chaîne de caractères, c'est un tableau de caractères !

- « Super texte »

Case	1	2	3	4	5	6	7	8	9	10	11
Valeur	'S'	'u'	'p'	'e'	'r'	' '	't'	'e'	'x'	't'	'e'

CONCEVOIR UN ALGORITHME

```
Programme multiplier(Entier a, Entier b) en Entier
Variables:
  Entier resultat <- 0
  Entier i <- 0

Début:
  Si (a = 0 OU b = 0) Alors
    Retour 0
  FinSi

  TantQue (i < b) Faire
    resultat <- a + a
    i <- i + 1
  FinTantQue

  Retour resultat
Fin
```

Le nom de notre sous-programme est "multiplier"

Il attend 2 paramètres a et b, de type Entier

Il retourne une valeur de type Entier

La première ligne correspond à la signature

Les instructions se trouvent entre la première et dernière ligne

- Un bloc d'instructions

Dans « Si » et dans « TantQue », on retrouve des instructions

- Ce sont aussi des blocs d'instructions, limités au « Si » et au « TantQue »

<- est l'affectation d'une valeur à une variable

=, < sont des opérations de comparaison

CONCEVOIR UN ALGORITHME

Mais nous n'allons pas utiliser ce langage pour décrire nos algorithmes

- On va se rapprocher d'un langage de développement

```
Programme Entier multiplier(Entier a, Entier b) {  
  Entier resultat = 0;  
  Entier i = 0;  
  
  Si (a == 0 OU b == 0) {  
    Retour 0;  
  }  
  
  TantQue (i < b) {  
    resultat = a + a;  
    i = i + 1;  
  }  
  
  Retour resultat;  
}
```

Les blocs d'instructions sont entre { et }

On met un point-virgule à chaque fin d'instruction

<- (affectation) est remplacé par =

= (comparaison) est remplacé par ==

CONCEVOIR UN ALGORITHME

Déclaration d'un programme sans paramètre, qui ne retourne rien

```
Programme Vide nomDuProgramme() {  
}
```

Déclaration d'un programme sans paramètre, qui retourne un Entier

```
Programme Entier nomDuProgramme() {  
}
```

Déclaration d'un programme avec 1 paramètre Booléen, qui retourne un Entier

```
Programme Entier nomDuProgramme(Booléen nomParametre) {  
}
```

Déclaration d'un programme avec 2 paramètres Booléen et Entier, qui ne retourne rien

```
Programme Vide nomDuProgramme(Booléen nomParametre1, Entier nomParametre2) {  
}
```


CONCEVOIR UN ALGORITHME

Condition Si, Sinon

```
Si (a == 0) {  
}  
  
Sinon {  
}
```

Condition Si, Sinon Si, Sinon

```
Si (a == 0) {  
}  
  
Sinon Si (b == 0) {  
}  
  
Sinon {  
}
```

CONCEVOIR UN ALGORITHME

Boucle TantQue

```
TantQue (a < 10) {  
}
```

Boucle Faire..TantQue

```
Faire {  
} TantQue (a < 10);
```

Boucle Pour

```
Pour (Entier i = 0; i < 10; i = i + 1)  
}
```

CONCEVOIR UN ALGORITHME

Les instructions s'exécutent les unes après les autres

Si vous appelez un autre sous-programme, les instructions de ce dernier s'exécuteront avant la suite de votre bloc d'instructions

```
Programme Entier unSousProgramme(Entier a) {  
    a = a * 2; // #4  
    Retour a; // #5  
}  
  
Entier i = 1; // #1  
Entier j = 0; // #2  
  
TantQue (i == 1) { // #3 et #8  
    j = unSousProgramme(i); // #6 - affectation du résultat à j  
    i = i + j; // #7  
}  
  
// #9
```

CONCEVOIR UN ALGORITHME

Procédez étape-par-étape

N'hésitez pas à prendre une feuille et à faire des schémas pour vous aider à visualiser

Faites attention

- Il n'y aura pas de logiciel pour vous aider à pointer vos erreurs
 - Oublie d'une déclaration de variable
 - Sortir du tableau
 - Oublie d'un paramètre
 - Mauvais type utilisé
 - Pas de valeur de retour
 - ...
- L'algorithme n'est vérifiable que par votre tête (ou celle de votre voisin !)

EXERCICE — ECRIRE DES ALGORITHMES

1. Faire en sorte de pouvoir calculer une puissance n ($2^4 = 2*2*2*2$)
2. Créer un tableau de 5 entiers, tous initialisés à 0
3. Insérer un nouvel entier dans un tableau (pas nécessairement plein) à un indice précis
4. Calculer la somme des entiers contenus dans un tableau reçu en paramètre (dont vous ignorez la taille)
5. Créer une variable capable de stocker la phrase suivante : « Mon premier algo »
6. Compter le nombre de caractères contenus dans un tableau de caractères (idem #4)
7. Compter le nombre d'espaces (caractères espace) contenus dans un tableau de caractères (idem #4)
8. Compter le nombre de mots contenus dans un tableau de caractères (idem #4)
9. Compter le nombre de voyelles contenues dans un tableau de caractères (idem #4)
10. Ranger chaque caractère d'un tableau par ordre alphabétique (idem #4)

EXERCICE — LIRE DES ALGORITHMES

Lire et comprendre les algorithmes de vos collègues