COURS ALGORITHME : INTRODUCTION

CHARLES 'ARYS' YAICHE

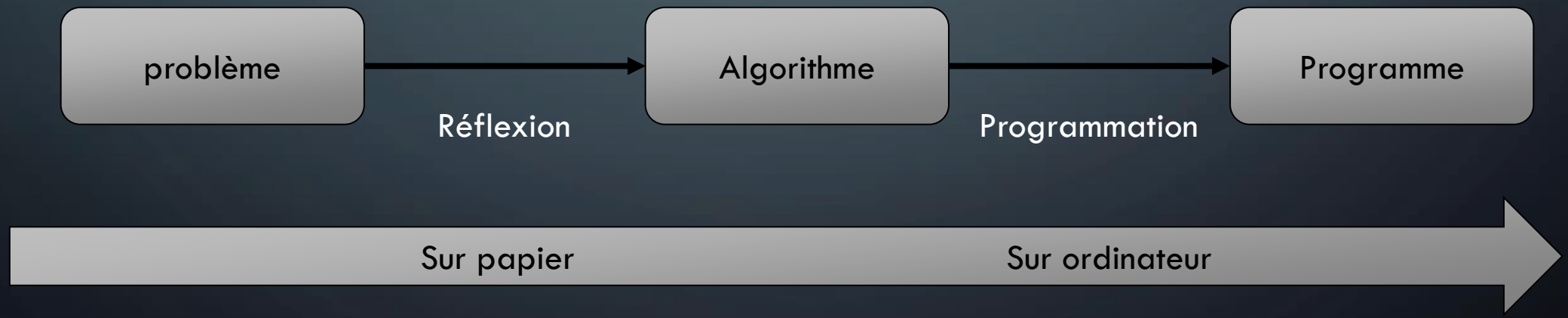
QU'EST-CE QU'UN ALGORITHME

Une suite d'**instructions** qui, quand elles sont exécutées correctement, aboutissent au résultat attendu.

- Énoncé dans un langage clair
- Bien défini
- Ordonné
- Réutilisable (sauf cas précis)
- Adapté au problème

POURQUOI EN UTILISER

L'algorithme décrit **formellement** ce que doit faire l'ordinateur pour arriver à un but bien précis. Ce sont les **instructions** qu'on doit lui donner. Ces **instructions** sont souvent décrites dans un langage clair et compréhensible par l'être humain : faire ceci, faire cela si le résultat a telle valeur, et ainsi de suite.



LES STRUCTURES ALGORITHMIQUES

- Les structure de contrôle
  - Séquences (bloc d'instruction, fonctions, procédures, etc.)
  - Conditionnel (conditions, expressions booléennes, etc.)
  - Boucles (itérations)
- Les structure de données
  - Constantes
  - Variables
  - Tableau
  - Structures récursives
  - …

LANGAGE ALGO : STRUCTURE GÉNÉRALE

```
algorithme identifiant_algo

    <partie déclarations>

debut

    <partie instructions>

fin algorithme identifiant_algo
```

LES COMMENTAIRES

- Les commentaires sont des bouts de texte, ignoré dans notre algorithme, ceux-ci servent uniquement pour le développeur, pour se repérer, ajouter des notes ou des commentaires
- Les commentaires se définisse comme :

// <commentaire sur une ligne>

/* <commentaire sur plusieurs lignes>

<commentaire sur plusieurs lignes>

*/

DÉCLARATIONS

- C'est ici que nous allons déclarer tout ce dont nous avons besoin pour l'algorithme : *constantes*, *types*, *variables* ainsi que les *routines* (*procédures* et *fonctions*).

```
<partie déclarations>:

    <déclarations des constantes>
    <déclarations des types>
    <déclarations des variables>
    <déclarations des routines>
```

TYPE PRÉDÉFINI

- entier	nombres entiers signés	42
- reel	nombres flottants signés	0.154
- octet(mot)	nombres entiers non signés sur un (deux) octet(s)	
- booleen	énumération définissant les données *vrai* et *faux*	
- caractere	caractère ANSI sur un octet	'a'
- chaine	chaîne de caractères	"lapin"

EXPRESSION

- Une expression représente une succession de calculs ; elle peut faire intervenir des constantes, des variables, des fonctions et des opérateurs. Les expressions sont utilisées dans tout l'algorithme : dans les affectations, en paramètre des routines, dans les structures de contrôles...

- une valeur	42
- une variable	x
- une constante	C
- un appel de fonction	cos(x)
- <expression>OpérateurBinaire <expression>	32 + x
- OpérateurUnaire <expression>	non A
- (<expression>)	(a + 15)

VARIABLE & AFFECTATION

- Cette instruction permet d'affecter une valeur à une variable. La valeur peut être n'importe quelle expression de type compatible avec la variable.
- On parle d’expression mutable
  - Un identifiant de variable
  - Une affectation
  - Une expression

```
ident_var ⟵ <expression>
```

CONSTANTE

- Une **constante** se **déclare** et se définis comme une **variable**
- La différence est qu'une **constante** ne peux pas se redéfinir ou se modifier
- On parle d'expression **immutable**

LES OPÉRATEURS

- Opérateur Unaire
  - (opérateur) <opérande>
- Opérateur Binaire
  - <opérande> (opérateur) < opérande >

LES OPÉRATEURS ARITHMÉTIQUES

- Les opérateurs arithmétiques permettent des expressions arithmétiques
- Il existe deux types d'opérateur :
  - Les opérateur Unaire (-)x
  - Les opérateur binaire x + y

- (unaire)	Changement de signe
+	Addition
-	Soustraction
*	Multiplication
/	Division flottante

`div`	Division entière
`mod`	Modulo (reste de la division entière)

LES VARIABLES BOOLEAN

- Les variable boolean sont des variables pouvant n'avoir que deux valeurs, vrai ou faux
- Les variable boolean permettent de créer des expressions de vérité et de test entre plusieurs expressions

Exemple

```
1 > 3 // faux
```

OPÉRATEUR LOGIQUE ET RELATIONNEL

non	négation logique (¬)
et	et logique (∧)
ou	ou logique (∨)
oue	ou exclusif

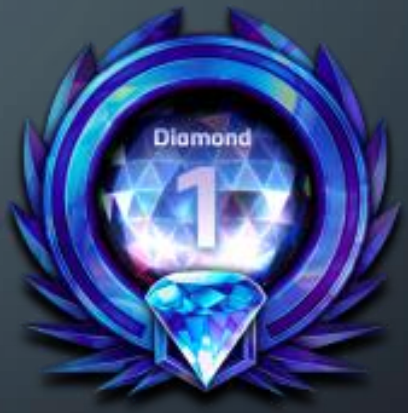
=	égal
<>	différent
<	inféreur à
>	supérieur
<=	inférieur ou égal
>=	supérieur ou égal

STRUCTURE CONDITIONNELLE

- Si la condition (exprimée par l'expression booléenne) est vraie alors seule la suite d'instructions placée après le **alors** sera exécutée. Dans le cas contraire, si la partie **sinon** existe elle sera exécutée, si elle n'existe pas, rien ne se passe.

```
si <expression booleenne> alors
        <instructions>
[sinon
        <instructions> ]
fin si
```

DIFFICULTÉ DES EXOS



- Facile
- Moyen
- difficile