COURS ALGORITHME : INTRODUCTION

CHARLES 'ARYS' YAICHE

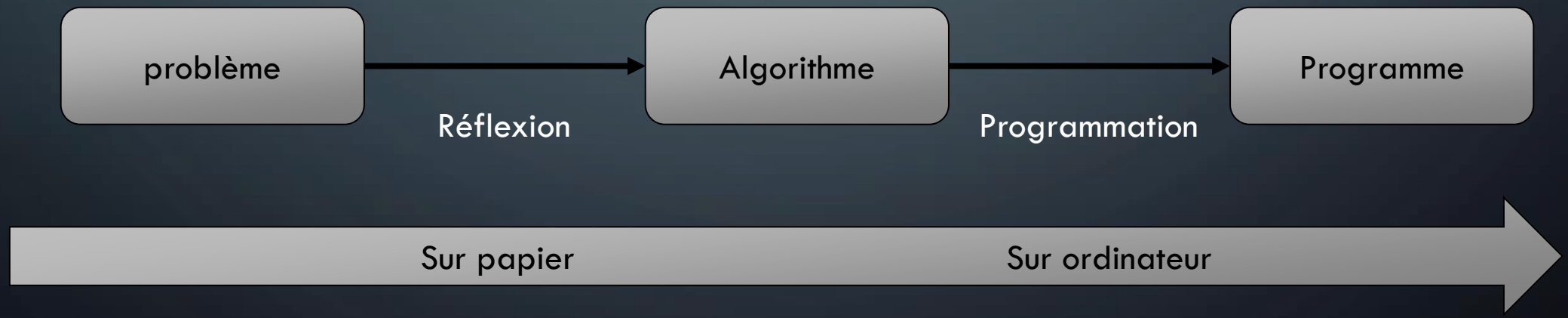
QU'EST-CE QU'UN ALGORITHME

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 GÉNÉRAL

```
algorithme identifiant_algo

    <partie déclarations>

debut

    <partie instructions>

fin algorithme identifiant_algo
```

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.

avec *ident_var* :

- un identifiant de variable
- une référence à un élément d'un tableau
- une référence à un champ d'enregistrement
- un objet pointé

```
ident_var ⟵ <expression>
```

LES STRUCTURES ITÉRATIVES (LES BOUCLES)

Une **itération** ou **structure itérative** est une séquence d'instructions destinée à être exécutée plusieurs fois. C'est aussi l'action d'exécuter cette instruction. Vous entendrez parler parfois de **structures répétitives**, c'est la même chose dite autrement.

TANT QUE (WHILE)

- En python, nous utilisons le mot clef while(booleen)
- Vous pouvez écrire une **boucle infinie** avec comme **condition** (true)
- Vous pouvez imbriquer les **boucles**

BREAK & CONTINUE

- Faites toujours bien attention à ce que votre **boucle** dispose d'une **condition de sortie.** En effet rien ne vous empêche de faire des boucles infinies. Dans le cas d'une structure itérative « **while** », il suffit que la condition soit toujours VRAIE, par exemple :
  - While True:
- Pour sortir d'une **boucle** on utilisera le mot clef **break**
- Pour aller directement à **l'itération** suivante, utiliser le mot clef **continue**
- **Gérer une boucle infinie avec un break est une MAUVAISE idée.**

POUR (FOR)

- En python, nous utilisons le mot clef for (booleen):
- une boucle à l'usage quasi exclusif des compteurs. À chaque passage dans la boucle, un compteur est incrémenté ou décrémenté, selon le cas. On dit alors qu'il s'agit d'une **structure incrémentale.**
- Ne modifiez pas une boucle for
- Utilisation de range

QUELLE BOUCLE CHOISIR

- On emploie une **structure** « Pour » lorsqu'on connaît à l'avance le nombre **d'itérations** nécessaires au traitement. Ce nombre peut être fixe ou calculé par avance avant la **boucle**, peu importe. La **boucle "Pour"** est **déterministe** : son nombre **d'itérations** est fixé une fois pour toute et est en principe **invariable** (bien qu'il reste possible de tricher).
- On emploie les structures « while » lorsqu'on ne connaît pas forcément à l'avance le nombre d'itérations qui seront nécessaires à l'obtention du résultat souhaité. L'exemple le plus concret est représenté par la saisie des notes : on peut en saisir une, 200, aucune, mais on ne le sait pas à l'avance. Mais ça peut être aussi la lecture des lignes d'un fichier (on n'en connaît pas le nombre par avance), d'enregistrements dans une base de données, un calcul complexe devant retourner une précision particulière, un nombre de tours dans un jeu (le premier qui gagne sort de la boucle), etc.

LES FONCTIONS ET PROCÉDURES

- Lorsqu'un programme est très long, il n'est pas réaliste de le coder d'un seul tenant. Le programme est décomposé en de plus petites unités ou parties réutilisables qui sont ensuite appelées le moment opportun par le programme principal. Un sous-programme évite la répétition inutile de code et permet de clarifier le programme. Une fois tous les sous-programmes réalisés, il est possible de les enregistrer dans des bibliothèques pour les réutiliser dans d'autres programmes, quels qu'ils soient, simplifiant ainsi fortement l'écriture du code, et permettant aussi d'aller beaucoup plus vite.

ENREGISTREMENT

- Un enregistrement est une structure qui regroupe dans une même entité logique un ensemble de données de types différents.

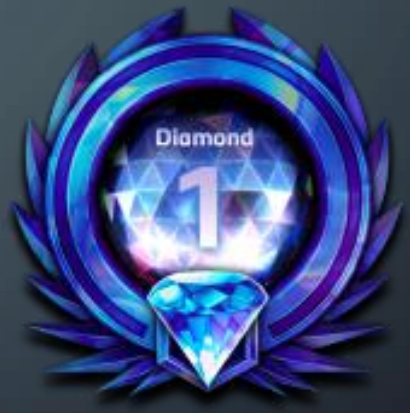
```
types
     ident_enreg = enregistrement
                   ident_type1     ident_champ11, ident_champ12, ...
                   ident_type2     ident_champ21, ident_champ22, ...
                   ...
     fin enregistrement ident_enreg
```

ACCES ENREGISTREMENT

- Pour accéder au contenu d'un enregistrement, on utilise la notation à point : on donne le nom de l'enregistrement (l'identifiant de la variable), suivi d'un point (.) et du nom du champ auquel on désire accéder.

```
ident_var.ident_champ
```

DIFFICULTÉ DES EXO



- Facile
- Moyen
- difficile