Sommaire

1. Objectifs du cours 1
2. Rappels : algorithme & programme 1
3. Les langages de programmation: typologies 2
4. Pourquoi la POO? 5

1. Objectifs du cours

Les objectifs du cours de la POO peuvent se résumer en trois objectifs de base:

1) Comprendre les fondements et la philosophie derrière la POO
2) Apprendre une démarche pour le développement orientée objet
3) Apprendre des langages de programmation orientée objet: C++ & Java.

2. Rappels : algorithme & programme

1. Définition d'un algorithme: c'est la méthode de résolution d'un problème. Plus particulièrement c'est une suite d'étapes (on dit aussi instruction) à exécuter séquentiellement (éventuellement par une machine) l'une après l'autre pour résoudre un certain problème. En effet, de cette définition, l'algorithme n'est pas nécessairement exécuté par une machine, mais il peut être même exécuté par un humain qui comprend le langage dans lequel cet algorithme est exprimé.

2. Définition d'un programme: Un programme est un texte écrit dans un langage particulier, dit langage de programmation. Ce texte représente la traduction de la méthode de résolution (par exemple un algorithme, dans le cas d'un **programme impératif**) déjà existante vers un langage compréhensible par la machine.

Cette définition du concept "programme" est générique et d'autre définition plus formelles ont été proposées. Parmi ces définitions, on trouve la définition proposée par Niklaus Worth (le concepteur du langage PASCAL et ALGOL), qui consiste en:

Programme=Algorithme+Structures de Données

Cette définition est très classique (depuis les années 1970) et concerne plutôt la définition d'un programme impératif (un concept qui sera introduit dans la section 3 de cette introduction). Le concept de programme reste donc un concept plus générale qui dépasse cette définition restreinte de Wirth.

3. Les langages de programmation: typologies

Il existe plusieurs classification possibles des langages de programmation selon différents critères. Les critères les plus adoptés sont comme suit:

1. Classification historique: Ici, on peut trouver tout d'abord les langages de bas niveau (binaire, et assembleurs) avec lesquels l'informatique a connu sa naissance dans les années 40, ensuite l'apparition des langages dits de "haut niveau" (dans les années 1960). Les langages de haut niveau sont dits abstraits (car ils sont définis indépendamment de la machine, de son architecture, de ses détails physique). La majorité des langages qu'on utilise actuellement trouvent leurs racines dans ces langages apparus dans les années 60. D'autres concepts ont été introduit avec le temps, comme le concept de Type, le concept de module, de types abstrait de données, d'objet, d'aspect, d'agent, etc.

2. Classification selon l'usage: Les langages peuvent aussi être classifiés selon leurs applications et usages:
   - Langages pédagogique: utilisés pour des raisons pédagogique, enseignement de la programmation à l'université ou ailleurs. Quelques célèbres langages sont: **Basic** (Beginner's All-purpose Symbolic Instruction Code) apparu en 1964, **ALGOL** (apparu en 1985, ensuite développé jusqu'aux années 68), **PASCAL** apparu 1971,
   - Langages d'applications commerciales: souvent orienté vers la gestion de fichiers, la gestion de base de données, la gestion des transactions, la comptabilité. Ici, il y'a aussi une variété de langages qui étaient développés: l'ancien COBOL (COmmon Business Oriented Language) apparu en 1959, les langages SQL (Structured Query Language) apparu en 1974, et enfin le fameux PHPMyadmin 1998, etc
   - Langages systèmes: Ce sont des langages orientés pour la programmation systèmes (tout ce qui est lié à la gestion des ressources physiques et logique de la machine). Ces langages sont plutôt proche du niveau physique de la machine en permettant au programmeur de manipuler ses détails architecturaux (mémoires, régistres, unité de contrôles, unité de calcul, etc). Le langage C (1972) est le plus représentant de ce genre de langages avec plusieurs de ses variantes (C++, C#, Embedded C). Le langage C est la base de l'écriture du système UNIX, LINUX, Windows et plusieurs autres systèmes d'exploitation qu'on utilise de nos jours.

Figure1. Ken Thompson (à gauche) et Dennis Ritchie (à droite).
auteurs du langage C au laboratoire Bell, 1972

- Langages de programmation scientifiques: ces langages sont utilisés souvent pour le calcul scientifique exigeant une précision et une vitesse de calcul. Même si des langages comme C, C++, Java (inventé en 1990) sont proposés aussi pour faire du calcul scientifique, de nos jours le monde s'oriente plutôt pour des langages plus flexible et plus puissant comme le langage Python (apparu en 1991 pour la première fois).

Figure2. Guido van Rossum, créateur de Python, à la OSCON 2006.

3. Classification selon la structuration: On peut trouver des langages structurés ou non structurés. Par la structuration, on veut dire la manière dont le programme est présenté. Un programme est considéré structuré si il assure une séparation nette entre la partie déclaration de données et la partie traitement et manipulation de ces données. D'où, on considère que la structuration est souvent liée à la définition déjà présentée par Niklaus Wirth. D'où des langages comme **ALGOL**, **PASCAL** où la forme du programme a une élégance particulière sont des langages plutôt structurés. Alors que des langages comme le C où il n'y a de contrainte sur l'arrangement des déclarations et des traitements sont des langages moins structurés ou carrément non structurés.
   Autre que cet aspect de structuration selon les données, on trouve aussi des langages typés (cas du Pascal, Algol, C, etc) qui exige la déclaration d'un type pour les données utilisées, et les langages non typés (cas de Python, **BASIC**, etc).

4. Classification selon le paradigme de programmation:
   Un paradigme est un style, approche ou une philosophie qu'on doit suivre et appliquer qu'on programme. Il existe plusieurs paradigmes de programmation, mais les plus connus sont les suivants:

   4.1. Paradigme impérative: C'est le paradigme le plus connu, le plus répandu surtout dans l'enseignement. Dans ce style, le programme est vu comme une suite d'ordres (d'où, on parle d'impératif) que le programmeur ordonne à la machine pour exécuter. Plusieurs langages sont basé sur ce paradigme, comme: **Fortran 1957 (***Formula Translation***)**, **Algol**, **Pascal**, **C**, ...

   4.2. Paradigme fonctionnel: C'est un paradigme qui considère le programme comme une fonction mathématique. Dans ce cas, il n'y a pas d'instruction que le programmeur ordonne à la machine, mais plutôt la machine doit évaluer une fonction et donne les résultats. Des langages fonctionnel comme: **Lisp (1958)**, **ML (1973)**, **Miranda (1985)**, ...

Exemple de code de la fonction factoriel en langage Lisp:

```
(defun factorial (n)
 "Calcule la factorielle de l'entier n."
 (if (<= n 1)
     1
   (* n (factorial (- n 1)))))
```

4.3. Paradigme déclaratif: à l'opposition du paradigme impératif où le programmeur donne des ordre, ici le programmeur plutôt déclare des situations (écrites en logique) et la machine doit déduire de nouvelles situations (ou de nouvelles connaissance). Ce paradigme utilise, souvent, la logique pour exprimer les situations) programmer, d'où on parle de programmation logique. Le célèbre langages ici est Prolog (1972) avec ses versions.

Exemple de code en langage Prolog (https://en.wikipedia.org/wiki/Prolog):

```
mother_child(trude, sally).

father_child(tom, sally).
father_child(tom, erica).
father_child(mike, tom).

sibling(X, Y)      :- parent_child(Z, X), parent_child(Z, Y).

parent_child(X, Y) :- father_child(X, Y).
parent_child(X, Y) :- mother_child(X, Y).
```

4.4. Paradigme parallèle: Dans ce cas le programme est vu comme un ensemble de sous programmes qui peuvent être exécutés en parallèle, en concurrence et aussi qui peuvent être distribués. Au début, il y'avait des langages spécifiques qui offrent ce style de programmation et des machines particulières qui permettent ce genre de programmation (comme par exemple, la machine dite **transputer** dans les années **1980** avec plusieurs microprocesseurs, voir la figure 3, et son langages **OCCAM**[1] apparu en **1983**), mais de nos jours toutes les machines même les laptops sont équipées par des architectures multi-processeurs et multi-cores permettant d'exécuter des programmes parallèlement.

Figure3. Le transputer: machine multi-processeurs.

[1] OCCAM doit son nom au philosophe Ockham (1287–1347). Plus de détail ici: https://link.springer.com/chapter/10.1007/11423348_5

4.5. Paradigme orienté objet: C'est le cœur de ce cours, et c'est parmi les paradigmes les plus révolutionnaires et qui a marqué l'évolution en informatique. Si l'informatique est elle est ce qu'elle l'est, c'est grâce à la notion d'orienté objet. Un programme orienté objet est donc un ensemble d'objets (des entité indépendantes) qui interagissent et coopèrent pour réaliser une tâche. Ces objets peuvent être exécuté en concurrence, en parallèle ou même distribués sur plusieurs processeurs ou machines. Le reste du cours va détailler ce paradigme.

4. Pourquoi la POO?

En effet, la programmation orienté objet trouve ses motivations dans plusieurs raisons:

- Limites de la programmation impérative (la plus répandu): ces limites concernent le nom réalisme, manque d'abstraction, manque de naturalisme. D'où la POO est venue pour renforcer l'abstraction et donc assure plus de réalisme et de naturalisme pour ce lui qui réalise des programmes. Ceci va lui faciliter la tâche de programmation. Ce qui implique moins d'erreur dans la programmation et plus de fiabilité des produits réalisés.
- La POO renforce aussi la maintenabilité des systèmes en offrant des solutions composées d'entités plus au moins indépendante (on parle d'une haute cohésion et d'un faible couplage), ces deux concepts seront plus détaillés dans les prochaines parties de ce cours.
- La mise en place de méthodes de développement de systèmes orientés objets, des langages de modélisation OO (cas de UML) et aussi la mise en place d'une variété de langages de programmation OO (C++, Java, etc).