

## L'informatique

- **Information automatique**
- **pour l'Académie Française :**
  - c'est la science du traitement rationnel, notamment par machines automatiques, de l'information considérée comme le support des connaissances et des communications, dans les domaines technique, économique et social*
- **acception courante : ensemble des sciences et techniques en relation avec le traitement de l'information**

- **informatics : science de l'information**  
étude des systèmes de traitement de l'information
- **computer science : science de l'ordinateur (calculateur), informatique théorique**  
Etude des algorithmes (indirectement logiciels, ordinateurs)
- **computer engineering : génie informatique**  
: science de l'ordinateur (calculateur)  
ce qui concerne la fabrication et l'utilisation des ordinateurs
- **software engineering : génie logiciel**  
ce qui concerne la modélisation et le développement de logiciels

## Ordinateur

- **machine capable d'effectuer automatiquement des opérations arithmétiques et logiques (à des fins scientifiques, administratives, comptables, . . . ) à partir de programmes définissant la séquence de ces opérations**
- **exécuter des séquences définies de calcul**

## Quelques dates

<http://www.grassouille.org/docs/cours-ii-html/node5.html>

### *Outils de calcul*

- les bouliers chinois
- [1620](#) : règle à calcul
- [1642](#) : la machine à additionner de Pascal

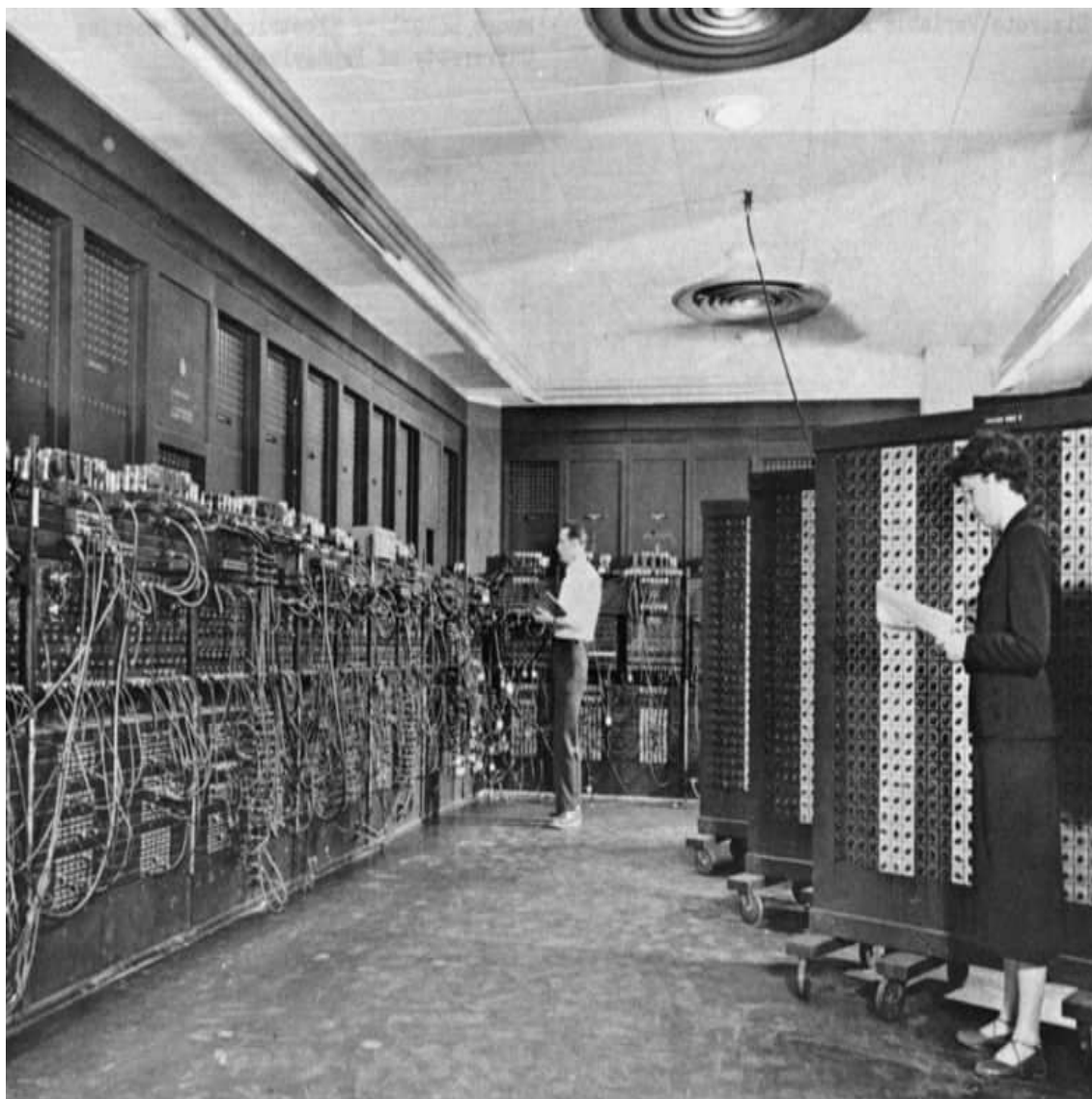


## *Automatisation des calculs*

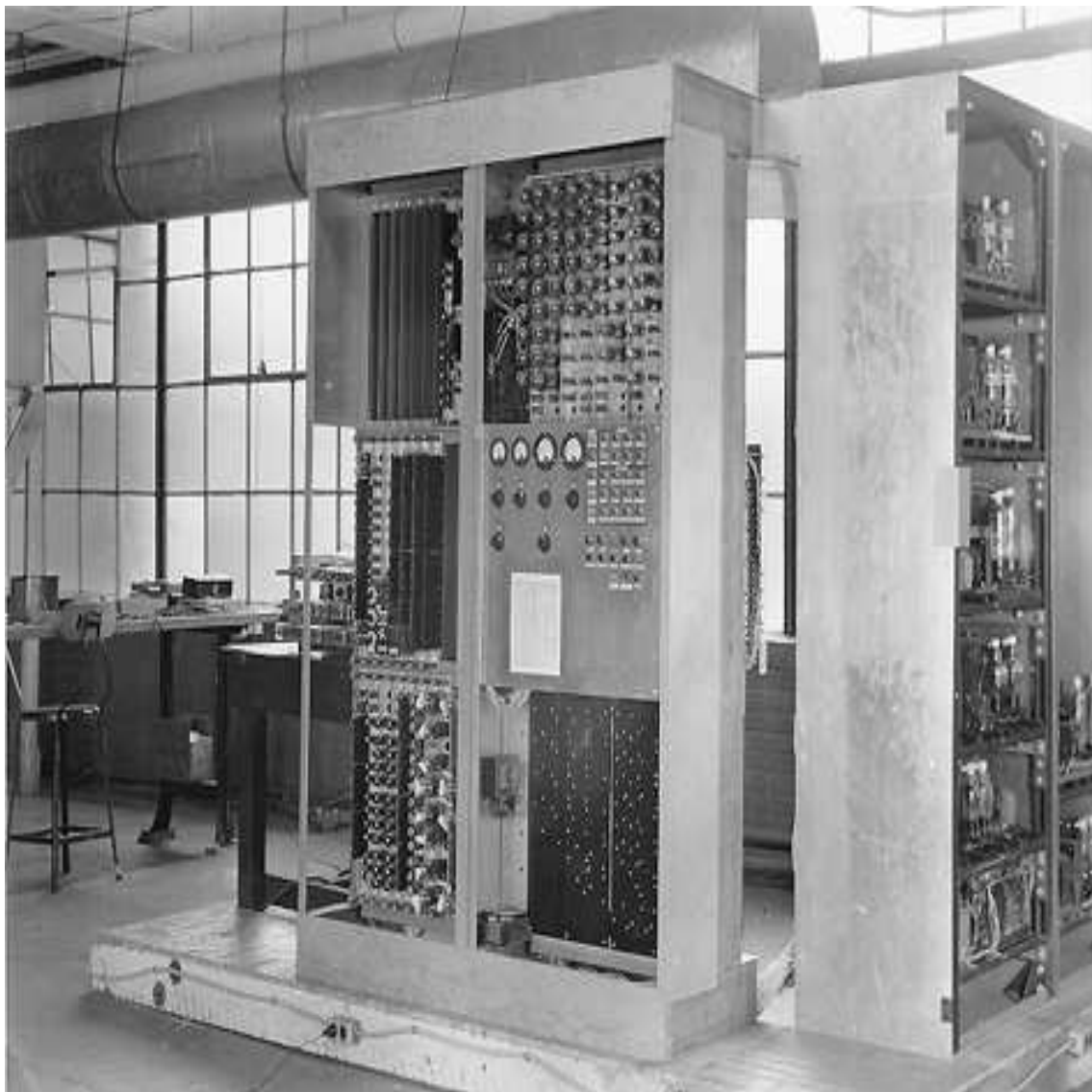
- **18ème** : commande de machines à tisser (planches de bois trouées puis cartons perforés - Jacquart)
- **1834** : Babbage définit une architecture de machine proche des machines électroniques actuelles :
  - programmation de calcul
  - notion de processeur, mémoire, entrée-sortie
  - trop complexe pour la technologie de l'époque

## *Avancées théoriques*

- **1854** Boole: algèbre de Boole, logique symbolique
- **1936** Turing : machines de Turing
- **1938** Shannon : liens entre nombres binaires, algèbre de Boole et signaux électriques
- **1944** : l'ENIAC (Electronical Numerical Integrator And Calculator) , premier prototype de calculateur électronique (5000 additions par seconde, une multiplication en 3 millisecondes, quelque chose comme 18000 tubes électroniques, consommant autant que plusieurs rames de métro, 30 tonnes sur 1000  $m^2$ , fréquence d'horloge de 100kHz, ....



- **1947** : successeur de l'ENIAC, l'EDVAC (Mark I) intègre la totalité d'un programme (écrit en binaire sur un clavier) dans la mémoire de la machine (architecture de Von Neuman)



- **1950** : le Z4 fait des branchements conditionnels
- **1952** : l'EDVAC est achevé (1Mhz)
- **1958** : 1-er ordinateur à transistors [MIT] et premier circuit intégré chez *Texas Instrument*
- **1963** : invention de la souris (D. Engelbart)

- **1965 : Moore énonce sa loi:**

*les CPU doubleront de complexité tous les 18 mois*

- **à partir de 1970 : apparition des micro-processeurs, des miniordinateurs et des microordinateurs**

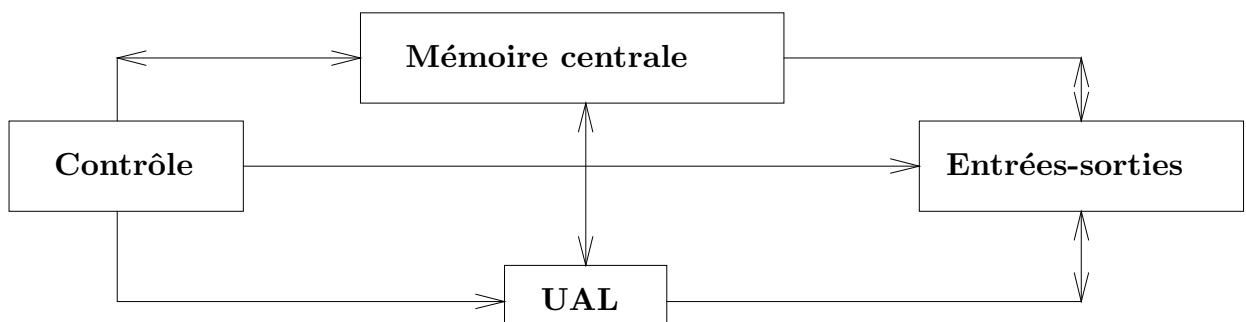
**Génération successive de processeurs (Intel, Motorola et PPC) 8, 16, 32, 64 et 128 bits**

**Architecture RISC**

## Ordinateur de Von Neumann

Cinq composants :

1. unité arithmétique et logique (UAL) ou de calcul
2. unité de contrôle (ou de commande)
3. mémoire centrale ou principale (RAM)
4. unités d'entrée
5. unités de sortie



Processeur = union de 1 et 2

Unité centrale = union du processeur et de la mémoire

## Quelques termes

- **bit** (*binary digit*)  
quantité d'information apportée par un chiffre dans le système binaire.  
Utilisé pour parler d'un chiffre binaire
- **notations** :  
0/1, vrai/faux, bas/haut, ouvert/fermé
- **octet** (*byte*) :  
ensemble de 8 octets
- **mot** :  
composé d'un certain nombre d'octets

- **ordres de grandeur / unités**

---

**kilo** ( $10^3$ ) [**K**], **kibi** ( $2^{10}$ ) [**Ki**]  
**mega** ( $10^6$ ) [**M**], **mébi** ( $2^{20}$ ) [**Mi**]  
**giga** ( $10^9$ ) [**G**], **gibi** ( $2^{30}$ ) [**Gi**]  
**téra** ( $10^{12}$ ) [**T**], **tébi** ( $2^{40}$ ) [**Ti**]  
**péta** ( $10^{15}$ ) [**P**], **pébi** ( $2^{50}$ ) [**Pi**]  
**exa** ( $10^{18}$ ) [**E**], **exbi** ( $2^{60}$ ) [**Ei**]  
**zetta** ( $10^{21}$ ) [**Z**], **zébi** ( $2^{70}$ ) [**Zi**]  
**yotta** ( $10^{24}$ ) [**Y**], **yobi** ( $2^{80}$ ) [**Yi**]

**milli** ( $10^{-3}$ ) [*m*]  
**micro** ( $10^{-6}$ ) [ $\mu$ ]  
**nano** ( $10^{-9}$ ) [*n*]  
**pico** ( $10^{-12}$ ) [*p*]  
**femto** ( $10^{-15}$ ) [*f*]  
**atto** ( $10^{-18}$ ) [*a*]  
**zepto** ( $10^{-21}$ ) [*z*]  
**yocto** ( $10^{-24}$ ) [*y*]

- **les jeux**
- **la bureautique (traitements de textes, tableurs)**
- **la communication (Internet)**
- **son, images et video**
- **les calculs mathématiques (météo, balistique)**
- **bases et banques de données**
- **calculs massifs (machines parallèles et clusters)**
- **graphisme (CAO/PAO, reconnaissances de formes ou de codes)**

- **l'embarqué (automobile, avionique, téléphonie)**
- **la physique et le contrôle (simulation, résistance de matériaux)**
- **la médecine (génomique, imagerie médicale, opérations)**
- **la robotique**

## Les aspects logiciels

Machine nue a priori inutilisable ou très difficilement utilisable, d'où l'intérêt de disposer d'outils spécialisés

- système d'exploitation  : il assure la gestion des ressources physiques (processeur, mémoire, périphériques) ou logiques (fichiers par exemple)
- outils logiciels divers  : ils permettent l'utilisation plus ou moins conviviale, la réalisation de tâches spécifiques, le développement et la maintenance de nouvelles applications, . . . , en s'appuyant sur les services du système d'exploitation.

Par exemple : outils de bureautique (tableurs ou traitement de textes), compilateurs de langages, débogueurs, SGBD, . . .

## Système d'exploitation

- Un système d'exploitation assure la gestion des ressources matérielles et logicielles sur une machine.
  
- principales évolutions :
  - ◇ traitement séquentiel des tâches (lecture d'un paquet de cartes perforées et exécution du programme lu)
  - ◇ multiprogrammation : lecture anticipée en mémoire (et donc cohabitation de plusieurs programmes en mémoire), toujours avec une exécution séquentielle

- ◇ **parallélisation des phases de calcul et d'entrées-sorties** grâce au mécanisme d'**interruption** : les phases inertes du CPU pendant les E/S peuvent être récupérées pour une autre tâche
- ◇ **partage du temps** entre différentes tâches par ajout d'une horloge externe envoyant périodiquement une interruption
- ◇ **mémoire virtuelle** permettant d'utiliser plus de mémoire que n'en possède physiquement l'ordinateur (extension de la mémoire vive par une zone dite de **swap**)
- ◇ utilisation en mode **multi-utilisateurs**
- ◇ répartition des services (systèmes réseaux et systèmes répartis, clusters)

- **fonctions du système**

- ◇ **pilotes de périphériques**
- ◇ **ordonnancement des tâches**
- ◇ **gestion de la mémoire**
- ◇ **gestion de fichiers**

- **systèmes propriétaires/ systèmes libres**
  - ◇  **systèmes propriétaires des constructeurs longtemps prédominants**
  - ◇  **systèmes Windows de Microsoft sur PC**
  - ◇  **le phénomène Unix**
    - \*  **ce que c'est**
    - \*  **la normalisation POSIX**
    - \*  **les Unix propriétaires**
    - \*  **les logiciels libres (Linux, FreeBSD)**
  
- **systèmes temps-réel**
  
- **systèmes embarqués**

**Interfaces utilisateur visant à permettre à un utilisateur de réaliser des actions (finalement exécuter des programmes)**

- **aspects matériels :**

- ◇ clavier/écran/souris
- ◇ périphériques (disques, imprimantes, . . . )

- **aspects logiciels :**

- ◇ interfaces textuelles rudimentaires (shells Unix, interpréteur de commandes DOS)
- ◇ interfaces graphiques : interaction via la souris avec des boutons, des menus, . . .

**Du point de vue d'un programmeur, possibilité de réutiliser des morceaux de code existant pour développer de nouvelles applications (bibliothèques, paquetages)**

## Langages de programmation

- ils permettent l'expression de programmes (l'équivalent des phrases pour une langue naturelle) compréhensibles par les ordinateurs, c'est-à-dire permettant l'expression d'algorithmes (suite d'opérations/actions/traitements/. . . ) réalisables par un ordinateur
- les programmes écrits dans un langage doivent être non ambigus
- les langages définissent une syntaxe que doivent scrupuleusement respecter les programmes prétendument écrits dans ce langage

## Niveaux de langages de programmation

- plus bas niveau (initialement pas d'autres) : le langage binaire (langage machine) permet l'écriture de programmes directement exécutables par l'ordinateur au travers de ses microprogrammes : programmation fastidieuse et programmes non portables

tout programme écrit dans un autre langage ne peut être exécuté directement : il nécessite soit sa traduction en un programme écrit en langage machine, soit dans un langage intermédiaire interprétable par un programme particulier exécutable par l'ordinateur

- les langages d'assemblage (1950) : ils définissent une notation symbolique des opérations du processeur. L'assemblage est l'opération de traduction correspondante
- les langages évolués : indépendants des machines, plus faciles à comprendre par l'humain. Les opérations de compilation et/ou d'interprétation en permettent l'exécution par un ordinateur.

## Différentes approches sont possibles :

- compilation pure :

un programme binaire directement directement exécutable par le processeur est généré à partir d'un programme écrit dans le langage source.

*Par exemple, un programme écrit en C est traduit par un compilateur, (par exemple le compilateur GNU gcc) en un binaire directement exécutable par le processeur*

```
--> cat Bonjour.c
#include <stdio.h>
#include <stdlib.h>
main() {
    int x = 3, y = 5;
    printf("Bonjour\n");
    printf("%d\n", x+y); }
--> ls -l Bonjour.c
ls -l Bonjour.c
-rwxrwxrwx  1 jmr  Enseigna  113 Sep 20 16:16 Bonjour.c
--> file Bonjour.c
Bonjour.c: ASCII C program text
--> gcc Bonjour.c -o Bonjour
--> file Bonjour
Bonjour: ... executable i386
--> ls -l Bonjour
-rwxrwxrwx  1 jmr  Enseigna  13368 Sep 24 14:43 Bonjour
--> ./Bonjour
Bonjour
8
-->
```

- **compilation dans un langage intermédiaire** :

un programme d'un langage intermédiaire est construit qui sera exécutable (interprétable) au travers d'un programme particulier jouant le rôle d'une machine virtuelle.

*Par exemple, un programme écrit en JAVA est traduit par un compilateur Java (javac) en du bytecode exécutable par la JVM (Java Virtual Machine)*

```
--> cat Bonjour.java
```

```
class Bonjour {  
    public static void main(String[ ] st){  
        int x = 3, y = 5;  
        System.out.println("Bonjour");  
        System.out.println(x + y);  
    }  
}
```

```
--> ls -l Bonjour.java
```

```
-rwxrwxrwx  1 jmr  Enseigna  166 Sep 24 14:52 Bonjour.java
```

```
--> javac Bonjour.java
```

```
--> ls -l Bonjour.class
```

```
-rwxrwxrwx  1 jmr  Enseigna  443 Sep 24 14:54 Bonjour.class
```

```
--> file Bonjour.class
```

```
Bonjour.class: compiled Java class data, version 49.0
```

```
--> java Bonjour
```

```
Bonjour
```

```
8
```

```
-->
```

- **interprétation** :

**un programme interprète à la volée les instructions d'un programme du langage source**

*Un fichier de commandes shell est interprété par un interpréteur du shell correspondant ou un programme Python est interprété par un interpréteur Python*

```
--> cat Bonjour.py
x=3
y=5
print "Bonjour"
print x+y
--> python Bonjour.py
Bonjour
8
-->
```

**Pour un langage donné, il peut exister sur un ordinateur à la fois un (ou plusieurs compilateurs) et un interpréteur.**

## La jungle des langages évolués

- des disciplines de programmation, d'où des langages satisfaisant des slogans
  - ◇ décrire les structures de données
  - ◇ éviter les branchements (instruction goto)
  - ◇ éviter les effets de bord
  - ◇ manipuler les pointeurs avec précaution
  - ◇ écrire des assertions
  - ◇ commenter
  
- des styles de programmation : impérative, fonctionnelle, orientée objet, logique, formelle
  
- des types d'applications particuliers (calcul scientifique, logiciel de gestion, calcul formel, calcul logique, intelligence artificielle, logiciel de base, . . . )

## Des styles de programmation

- la programmation impérative :
  - ◇ langage machine de haut niveau pour une machine virtuelle basé sur l'effet de bord (opération appelée affectation) permettant de remplacer en mémoire une valeur par une autre.
  - ◇ utilisation de noms symboliques pour les adresses
  - ◇ un programme est une suite de traitements exécutés séquentiellement et possibilité
    - \* de réaliser des tests
    - \* itérer des traitements
    - \* modulariser le traitement (sous-programmes, procédures, fonctions)

◇ famille de langages illustrée (entre autres) par

- \* **FORTRAN** (*FORmula TRANslator*) [1957] pour les applications scientifiques (John Backus, IBM)
- \* **ALGOL** (*ALGOrithmic Language*) [1960]
- \* **COBOL** (*COmmon Business Oriented Language*) [1960] pour les applications de gestion
- \* **BASIC** (Beginner's All-purpose Symbolic Instruction Code) [1964] pour les ordinateurs personnels
- \* **PL1** (*Programming Language I*) [1964] (IBM)
- \* **PASCAL** [1968] (N. Wirth)
- \* **C** [1973] (Kernighan et Ritchie, Bell Labs)
- \* **ADA** [1983] (Ichbiah)
- \* **Perl** (*Practical Extraction and Report Language*) [1987]

- **la programmation fonctionnelle** :
  - ◇ le principe y est de n'écrire que des fonctions et des appels de fonctions. On y privilégie la notion de calcul plutôt que celle de modification de l'état de la mémoire comme c'est le cas dans le style impératif
  - ◇ fondé sur la théorie du  $\lambda$ -calcul (branche de la logique mathématique étudiant la notion générale de fonction indépendamment de ses domaines)
  - ◇ notion d'environnement (ensemble de couples [nom,valeur]), dans lequel une paire ne peut pas être modifiée une fois créée

- ◇ **famille de langages illustrée de manière plus ou moins pure par**
  - \* **LISP** (*LISt Processing*) [1959] (J. Mc Carthy, MIT) : typage statique (lors de l'exécution)
  - \* **APL** (*A Programming Language*) [1962] (K. Iverson)
  - \* **Scheme** [1975]
  - \* **ML** (*MetaLanguage*) [1978] (R. Milner) et sa descendance (**Caml** [1984] et **Caml-light** [1990])
  - \* **Miranda** [1985]
  - \* **Haskell** [1990] (calcul retardé au maximum)

- **la programmation orientée objet** :

- ◇ **caractéristiques**

- \* **méthode de programmation et de conception de langage dont les buts sont de rendre les grands projets logiciels plus facile à gérer, améliorer la qualité et réduire le nombre d'échecs de projet.**
- \* **objet = unité de base d'un programme : il contient des données (attributs), et une série d'actions (méthodes) applicables aux données. D'où l'idée de modéliser un ensemble d'éléments d'une partie du monde réel en un ensemble d'entités informatiques (objets). De telles données informatiques regroupent les principales caractéristiques des éléments du monde réel (taille, la couleur, ...)**

## ◇ famille de langages illustrée par

- \* Simula [1967]
- \* SmallTalk [1972] (Xerox)
- \* C++ [1983] (B. Stroustrup)
- \* Eiffel [1985] (B. Meyer)
- \* Python [1990]
- \* Objective C [1983]
- \* Ocaml [1995] (*Objective caml*)
- \* JAVA [1995] (J. Gosling, Sun Microsystems).
- \* C#[2000] (C-sharp) (Microsoft)

- **la programmation logique** :
  - ◇ famille de langages illustrée par **PROLOG** (*PROgrammer en LOGique*) [1975] (Colmerauer) : adapté à l'IA (aux systèmes sur les langues naturels, systèmes experts)
  
- **les métalangages et langages de description** utilisés pour définir des langages et structurer des documents tels que
  - **SGML** *Standard Generalized Markup Language* [1986]
  - **XML** *Extensible Markup Language* [1998] (W3C)
  - **HTML** *Hyper Text Markup Language* [1992]

- les langages de script (les shells, Perl, Python, PHP, JavaScript)
- les langages de requêtes tels que SQL (*Structured Query Language*) pour communiquer avec une base de données
- les langages de calcul formel tels que Mathematica ou Maple

## Des exemples de programme :

On souhaite calculer la somme des nombres entiers d'une suite donnée, c'est-à-dire

$$\sum_{i=1}^n t[i]$$

- en Basic (avec goto) :

```
i=1
s=0
Loop:IF i=n GOTO END
    s=s+t[i]
    i=i+1
    GOTO Loop
END: PRINT s
```

- en APL :

```
+/t
```

- en MAPLE :

```
sum('t[i]', i=1..n) ;
```

- en C :

```
int somme(int t[ ], int n) {
    int i, s = 0;
    for(i=0; i<n; i++) s = s + t[i];
    return s;
}
```

- en **JAVA** :

```
class SommeListe {  
    public static int sommeListe(int [] t, int n){  
        int s = 0;  
        for(int i=0; i<n; i++) s = s + t[i];  
        return s; } }
```

- en style fonctionnel (**Caml**) :

```
let rec sigma l =  
    match l with  
        [ ] -> 0  
        | h::t -> h + sigma t;;
```

*ou encore :*

```
let rec fold f r l =  
    match l with  
        [ ] -> r  
        | h::t -> fold f (f r h) t;;  
let sigma l =  
    fold (+) 0 l;;
```