CM - Découverte de la programmation

Professeur : François Bodin, bodin@irisa.fr

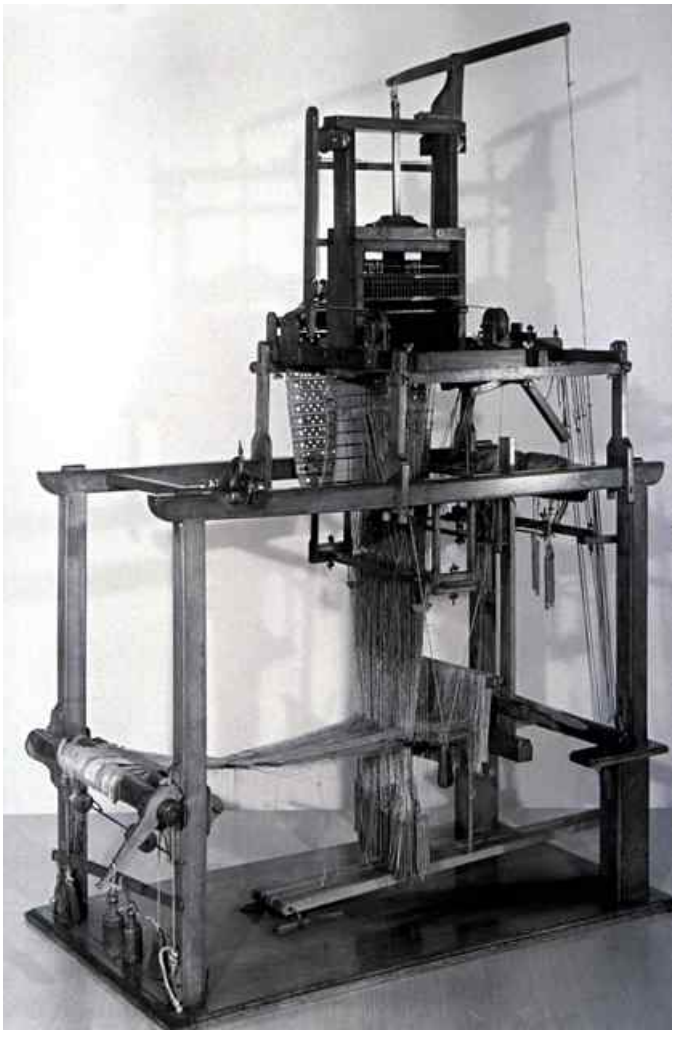
Introduction

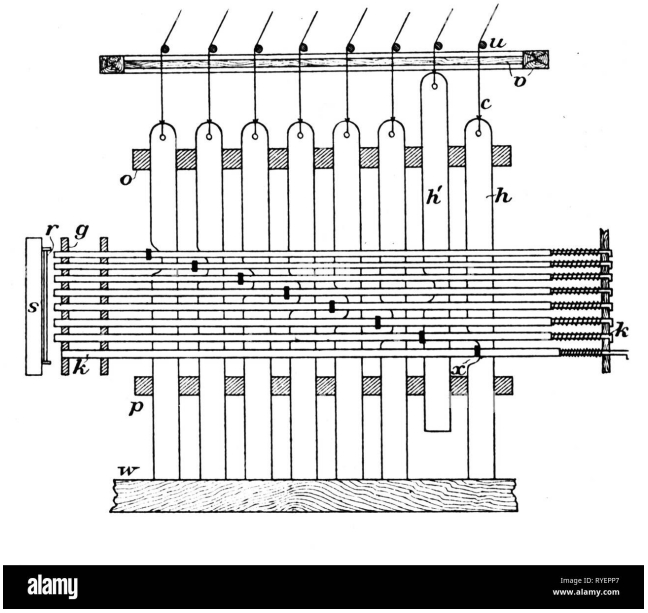
Informatique : pas un domaine nouveau --> *Computer Lib* (1974) par Ted Nelson (inventeur du lien hypertexte)
=> n'importe qui devrait être capable de comprendre un ordinateur (si ce n'est pas le cas : pas assez de vulgarisation)

Les concepts autour de l'informatique existent depuis assez longtemps (on parlait déjà de 3D en 1974)

Avant l'ordinateur moderne

- Machine Jacquard (cartes perforées pour guider le processus de tissage) : XIXè, pour limiter l'intervention humaine





- Machine de Babbage (1834) : machine de calcul qui utilise les cartes du métier Jacquard pour donner des instructions et des données

1840 : notion d'algorithme (Ada Lovelace) pour programmer une machine

Premiers ordinateurs (XXè)

Avec des interrupteurs (fermé/ouvert = 0/1)
L'Eniac qui permet de calculer des tables de tir (avec 17000 tubes à vide)

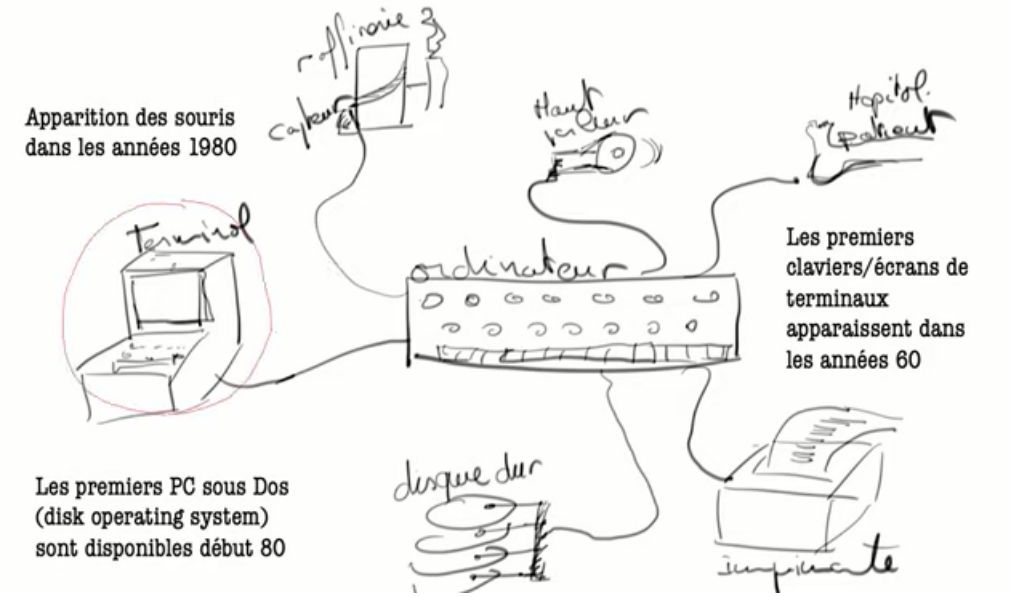
Utilisation des tubes => couts énergétiques importants

Nouveauté : mémoires en lecture et en écriture

- 64 mots de 22 bits de mémoire vive pour le Z1 de Konrad Zuse
- 2 ko de mémoire vive pour l'Eniac

Machine dite "**Turing complet**" => peut calculer tout ce qui est calculable

Un ordinateur en 1970



--> ressemblent à nos ordinateurs actuels

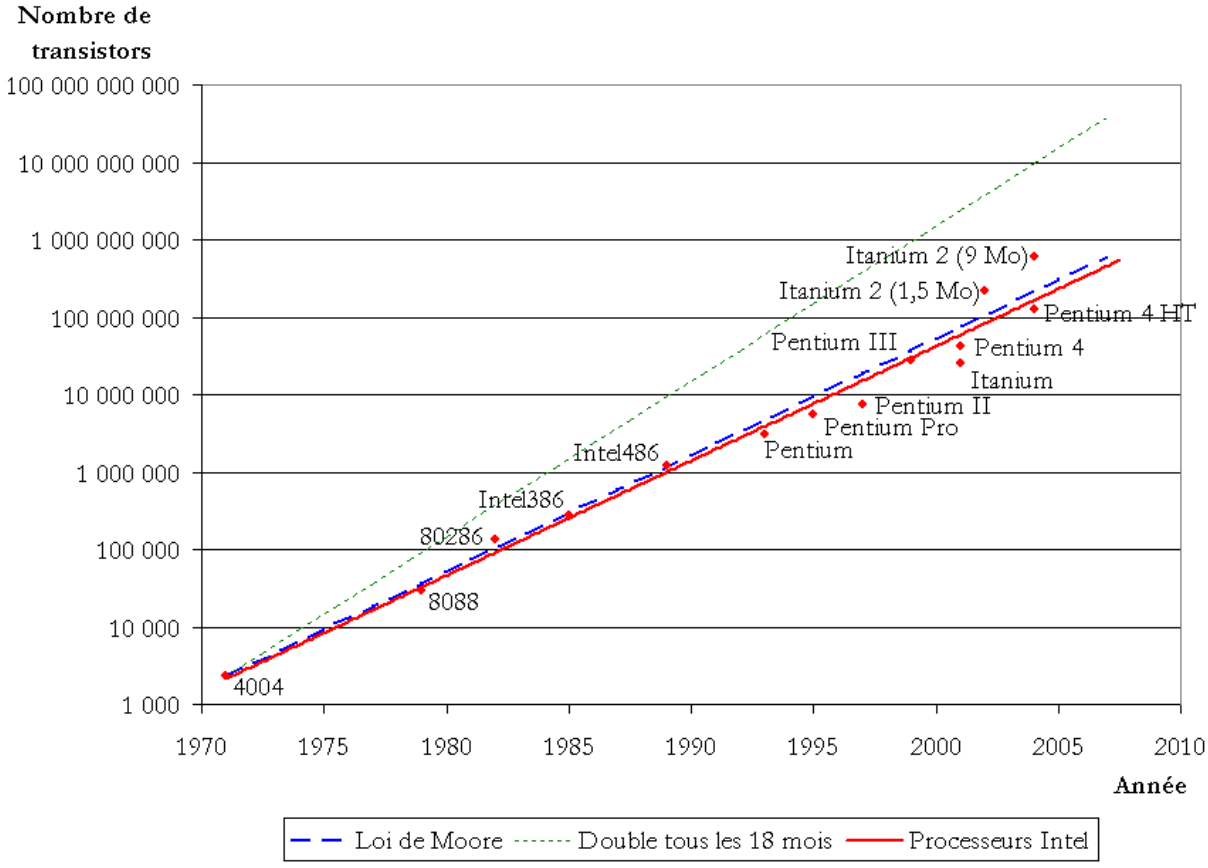
Ordres de grandeur :

- 1938 < 1 Flops (= opérations flottantes par secondes)
- 1945 : 1000 Flops
- 1996 : 1000 000 000 000 000 Flops
- 2023 : 1000 000 000 000 000 000 Flops (*Exascale*)

Les machines qui permettent de faire autant d'opérations par secondes sont énormes (équivalentes à des data centers)

Raisons du succès des ordinateurs

- Le prix
  Aujourd'hui : fin de la **loi de Moore** (tous les 18 mois, on double les transistors mais le prix de change pas)



- la vitesse (limité par la vitesse de la lumière : problème de latence)
- la fréquence

Donc : **augmentation de la puissance, de la mémoire** à prix ≈ constant

Représenter les valeurs dans l'ordinateur

Un codage permet la représentation et la manipulation des nombres/caractères dans l'ordinateur

- Texte/Caractères : au début : ASCII puis Unicode (16 bits)

USASCII code chart

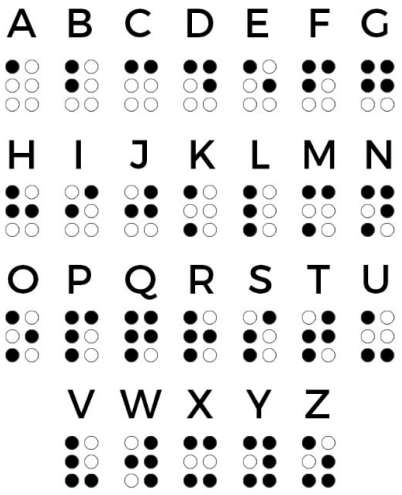
Bits b7 b6 b5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b4 b3 b2 b1 ↓ Row / Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

- Images : pixels (mais au début, traçage avec des faisceaux verts/noirs) en RGB
- Son : échantillonnage d'une courbe (fréquence d'échantillonnage doit être au moins 2 fois supérieure à la fréquence maximale représentée)

Le **binaire** s'impose au **niveau hardware** par sa facilité de mise en œuvre

Un exemple de codage : le braille

Une grille de 6 points peut encoder chaque lettre et chaque chiffre



Les programmes

> **Programme**
>
> Ensemble d'instructions destinées à être exécutées par un ordinateur
> -> s'exprime dans un langage (compréhensible par l'ordinateur de préférence)```
>
> Plusieurs sortes de programmes :
>
> - programme source écrit par un informaticien
> - programme binaire : traduction automatique du programme binaire
> - programme assembleur : forme lisible du programme binaire

RISC (*Reduced instruction set computer*) : pas de microprogramme associé aux instructions
CISC (*Complex instruction set computer*) : boucles ou séquences d'instructions possibles

→ n'est pas propre à l'informatique

Langages de programmation

Notation permettant de décrire des séquences de calculs, ils permettent de représenter des concepts
Il en existe des milliers (Java, Python, C, C++, Fortran...) : recherche d'un compromis d'utilisation et puissance expressive

Les langages font le lien entre la pensée et la machine

Tous les langages sont sujets à standardisation (pas un·e concepteur·ice) --> les langages évoluent (tous les ans/tous les 2 ans)

Un des premiers langages : Fortran II

= mathematical FORmula TRANslating system

Exemple de programme

```
program degrad
    ! ================================================
    ! Imprime une table de conversion degrés -> radians
    ! avec 32 chiffres significatifs (réels 128 bits)
    ! ================================================
    use, intrinsic :: iso_fortran_env, only: wp=>real128
    implicit none
    ! Déclaration des variables
    integer  :: deg
    real(wp) :: radians(0:90)
    ! Déclaration des constantes
    real(wp), parameter :: pi = 4.0_wp * atan(1.0_wp)
    real(wp), parameter :: coeff = (2.0_wp * pi) / 360.0_wp
    character(*), parameter :: ligne_horizontale = "(49('-'))"
    ! Remplissage du tableau :
    radians = [ (coeff * deg, deg=0,90) ]
    ! En-tête de la table
    write (*, ligne_horizontale)
    write (*, "('| Degrés |', 15(' '), 'Radians', 16(' '), '|')")
    write (*, ligne_horizontale)
    ! Affichage formaté
    do deg = 0, 90
        write (*, "('|   ', I2,'   |  ', F34.32,'  |')") deg, radians(deg)
    end do
    ! Fin de la table
    write (*, ligne_horizontale)
end program degrad
```

Est toujours utilisé dans les milieux scientifiques (de moins en moins) pour faire des modèles numériques (1 ligne de code pour un langage ≈ 10-100€ donc pour ne pas tout réécrire, on continue à utiliser ce langage)

Cartes perforées

Chaque colonne correspond à 1 octet

> **Les cartes perforées**
>
> Les cartes perforées sont parmi les premiers systèmes d'entrée-sortie et les premières mémoires de masse utilisés dans les débuts de l'informatique au XXe siècle.
>
> Source

Programmation pour le projet Apollo

Au départ : beaucoup de femmes dans l'informatique (*Exemples : [Margaret Hamilton*] (https://www.wikiwand.com/fr/Margaret_Hamilton_(scientifique)) et de plus en plus masculin avec l'arrivée des ordinateurs

Le code source

C'est le texte du programme qui sera produit lorsque je programme, une idée est transformée en une suite d'instructions
Saisie à l'aide d'un éditeur de texte (de type Bloc-Note et non Word)

Etapes dans les langages

- 1957 : Fortran
- 1958 : Algol (ALGOrithmic Language)
- 1978 : C
- 1982 : C++
- 1993 : Lua
- 1995 : Java

Beaucoup de langages peuvent être très efficaces mais non popularisés car non utilisés par les programmeurs·euses → donc dépend d'à quel point le·la programmeur·euse "aime" le langage

Lua

- langage interprété (comme Java)
- exécuté directement sans étape préalable de conversion en instruction binaire

Au début : perte de performance mais maintenant le matériel est suffisamment puissant pour que cela ne soit pas visible

Conclusion

Des progrès constants des systèmes informatiques : la machine est suffisamment puissante pour que son fonctionnement soit transparent à l'utilisateur. De nombreux langages de programmation offrent un accès simple à des fonctionnalités complexes.

Comprendre la programmation est au cœur de l'appropriation de l'outil informatique.

Pourquoi des recherches en logiciels ?

- Des nouveaux problèmes - nouveaux algorithmes
  - décodage du génome → algorithmes d'identification de séquences
- Des problèmes théoriques toujours pas résolus
  - la complexité de certains algorithmes n'est pas clairement établie → peut-on faire plus efficace ?
- Nouvelles capacités des ordinateurs, nouvelles possibilités
  - exploitation pétrolière intéressée par *Exascale* pour faire une carte beaucoup plus précise des sous-cols ; simulation des climats
- Des nouveaux usages, des nouveaux besoins
  - systèmes cyberphysiques, Internet des Objets, Big Data
- Complexité croissante des logiciels
  - recherche de nouvelles méthodes de conception, de vérification

---

DIE - Projet

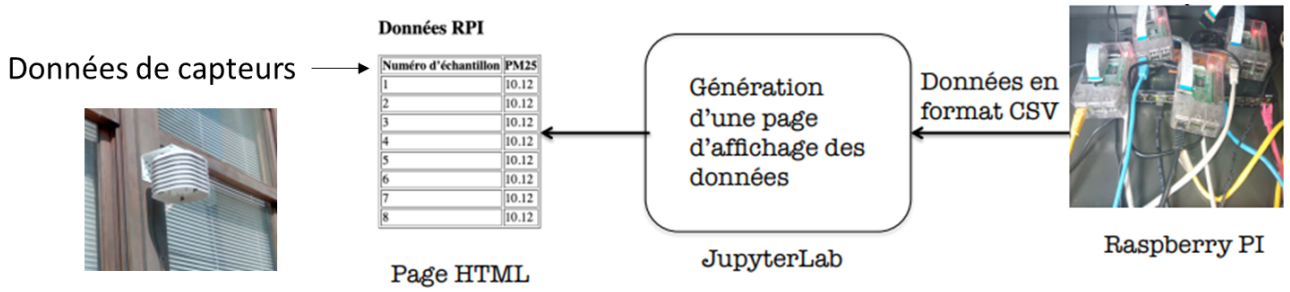
Introduction

--> Surtout orienté vers l'usage des technologies (pratique + que théorique)

Objectif : Introduction à la programmation

- Introduction à la construction d'une application distribuée
- Se familiariser avec les concepts et composants d'un système informatique
- Développer des méthodes de travail efficaces

Objectif : Développement de visualisation de données
Développement de codes d'affichage de données de capteurs dans des pages Web



Méthode de travail

- S'inspire des techniques agiles utilisées dans l'industrie
- Se décompose en étapes (sprint), chacune définie de la manière suivante
  - une tâche à réaliser
  - un rendu de fichier à effectuer

    - un quizz d’auto-évaluation
- Les sprints ont des durées variables suivant leurs difficultés et le volume de travail

Matériel pédagogique :

- [Document de projet](http://prototypel1.irisa.fr/docum entation/documentationprojet.pdf)
- Vidéos de démonstration
- Exemples de codes
- Page moodle avec des sujets d'examens

Tout doit être rendu pour le 15 octobre (mais rendre avant car risque de bug technique)
Voir les modalités ici (binôme pas obligatoire)