

# Introduction à la programmation

420-W10-SF

# Objectifs

- Comprendre ce qu'est un algorithme
- Comprendre ce qu'est un langage de programmation
- Avoir quelques notions d'histoire pour comprendre comment vous sommes arrivés aux machines programmable
- À partir de là, comprendre la structure du cours et sa place dans le programme

# Algorithmique

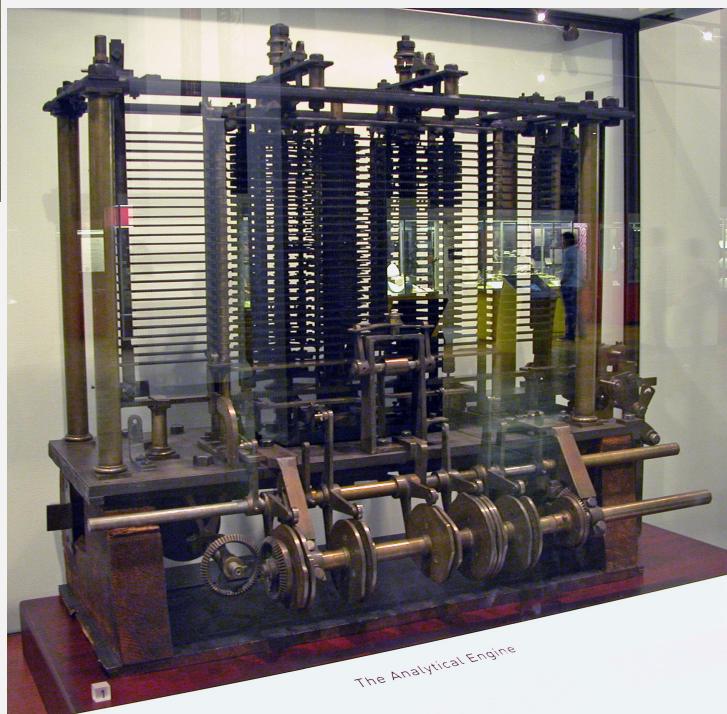
- L'algorithmique est présente partout :
  - Exemple, processus de fabrication :
    - Description :
      - des matériaux en entrée
      - une séquence d'instructions
      - un résultat
    - Exemple :
      - Recette de cuisine : ingrédients, instructions simples (frire, flamber, rissoler, etc.), le plat préparé
      - Tissage : fils, actions répétées, tissu (ex. métier à tisser de Jacquard)

# De la Pascaline à aujourd’hui



**Blaise Pascal**

Pascaline : idée 1642, prototype 1645



**Charles Babbage & Ada Lovelace**

Machine analytique : description 1837,  
prototype 1910 partiel

Utilisation de cartes perforées basées sur le  
métier à tisser de Jacquard



# De la Pascaline à aujourd’hui

Diagram for the computation by the Engine of the Numbers of Bernoulli. See Note G. (page 722 et seq.)

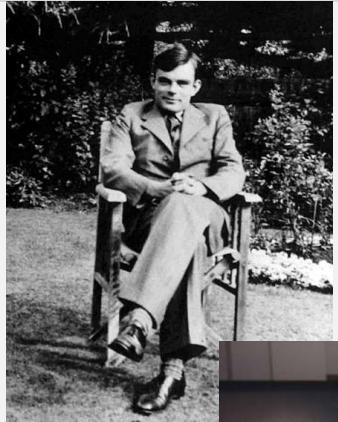
Number of Operation.	Nature of Operation.	Variables acted upon.	Variables receiving results.	Indication of change in the value on any Variable.	Statement of Results.	Data.			Working Variables.									Result Variables.													
						IV <sub>1</sub>	IV <sub>2</sub>	IV <sub>3</sub>	0V <sub>4</sub>	0V <sub>5</sub>	0V <sub>6</sub>	0V <sub>7</sub>	0V <sub>8</sub>	0V <sub>9</sub>	0V <sub>10</sub>	0V <sub>11</sub>	0V <sub>12</sub>	0V <sub>13</sub>	0V <sub>14</sub>	0V <sub>15</sub>	0V <sub>16</sub>	0V <sub>17</sub>	0V <sub>18</sub>	0V <sub>19</sub>	0V <sub>20</sub>	IV <sub>21</sub>	IV <sub>22</sub>	IV <sub>23</sub>	0V <sub>24</sub>		
1	×	IV <sub>2</sub> × IV <sub>3</sub>	IV <sub>4</sub> , IV <sub>5</sub> , IV <sub>6</sub>	$\begin{cases} IV_2 = IV_2 \\ IV_3 = IV_3 \end{cases}$	= 2 n .....	...	2	n	2 n	2 n	2 n															B <sub>1</sub>					
2	-	IV <sub>4</sub> - IV <sub>1</sub>	IV <sub>4</sub>	$\begin{cases} IV_4 = 2V_4 \\ IV_1 = IV_1 \end{cases}$	= 2 n - 1 .....	1	...	...	2 n - 1																		B <sub>2</sub>				
3	+	IV <sub>5</sub> + IV <sub>1</sub>	IV <sub>5</sub>	$\begin{cases} IV_5 = 2V_5 \\ IV_1 = IV_1 \end{cases}$	= 2 n + 1 .....	1	...	...	...	2 n + 1																	B <sub>3</sub>				
4	+	2V <sub>6</sub> + 2V <sub>4</sub>	IV <sub>11</sub>	$\begin{cases} 2V_6 = 0V_6 \\ 2V_4 = 0V_4 \end{cases}$	= $\frac{2n-1}{2n+1}$ .....	...	...	...	0	0	...	...	...	...	...	...	...														
5	÷	IV <sub>11</sub> ÷ IV <sub>2</sub>	IV <sub>11</sub>	$\begin{cases} IV_{11} = 2V_{11} \\ IV_2 = IV_2 \end{cases}$	= $\frac{1}{2} \cdot \frac{2n-1}{2n+1}$ .....	...	2	...	...	...	...	...	...	...	...	...															
6	-	0V <sub>13</sub> - 2V <sub>11</sub>	IV <sub>13</sub>	$\begin{cases} 0V_{13} = 0V_{11} \\ IV_{13} = IV_{13} \end{cases}$	= $-\frac{1}{2} \cdot \frac{2n-1}{2n+1} = A_0$ .....	...	...	...	...	...	...	...	...	...	...	...	0	...													
7	-	IV <sub>3</sub> - IV <sub>1</sub>	IV <sub>10</sub>	$\begin{cases} IV_3 = IV_3 \\ IV_1 = IV_1 \end{cases}$	= n - 1 (= 3) .....	1	...	n	...	...	...	...	...	...	...	...	n - 1														
8	+	IV <sub>2</sub> + 0V <sub>7</sub>	IV <sub>7</sub>	$\begin{cases} IV_2 = IV_2 \\ 0V_7 = IV_7 \end{cases}$	= 2 + 0 = 2 .....	...	2	...	...	...	...	...	...	...	...	2															
9	÷	IV <sub>6</sub> + IV <sub>7</sub>	IV <sub>11</sub>	$\begin{cases} IV_6 = IV_6 \\ IV_7 = IV_11 \end{cases}$	= $\frac{2n}{2} = A_1$ .....	...	...	...	...	2 n	2	...	...	...	...	...		$\frac{2n}{2} = A_1$													
10	×	IV <sub>21</sub> × 3V <sub>11</sub>	IV <sub>12</sub>	$\begin{cases} IV_{21} = 3V_{11} \\ IV_{12} = 3V_{11} \end{cases}$	= B <sub>1</sub> · $\frac{2n}{2} = B_1 A_1$ .....	...	...	...	...	...	...	...	...	...	...		$\frac{2n}{2} = A_1$	B <sub>1</sub> · $\frac{2n}{2} = B_1 A_1$	.....								B <sub>1</sub>				
11	+	IV <sub>12</sub> + IV <sub>13</sub>	IV <sub>13</sub>	$\begin{cases} IV_{12} = 0V_{12} \\ IV_{13} = 2V_{13} \end{cases}$	= $-\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2}$ .....	...	...	...	...	...	...	...	...	...	...	...		0		$\left\{ -\frac{1}{2} \cdot \frac{2n-1}{2n+1} + B_1 \cdot \frac{2n}{2} \right\}$											
12	-	IV <sub>10</sub> - IV <sub>1</sub>	IV <sub>10</sub>	$\begin{cases} IV_{10} = 2V_{10} \\ IV_1 = IV_1 \end{cases}$	= n - 2 (= 2) .....	1	...	...	...	...	...	...	...	...	...	...	n - 2														
13	-	IV <sub>6</sub> - IV <sub>1</sub>	IV <sub>6</sub>	$\begin{cases} IV_6 = 2V_6 \\ IV_1 = IV_1 \end{cases}$	= 2 n - 1 .....	1	...	...	...	...	2 n - 1																				
14	+	IV <sub>1</sub> + IV <sub>7</sub>	IV <sub>7</sub>	$\begin{cases} IV_1 = IV_1 \\ IV_7 = 2V_7 \end{cases}$	= 2 + 1 = 3 .....	1	...	...	...	...	...	3																			
15	+	2V <sub>6</sub> + 2V <sub>7</sub>	IV <sub>8</sub>	$\begin{cases} 2V_6 = 2V_6 \\ 2V_7 = 2V_7 \end{cases}$	= $\frac{2n-1}{3}$ .....	...	...	...	...	2 n - 1	3	$\frac{2n-1}{3}$																			
16	×	IV <sub>8</sub> × 3V <sub>11</sub>	IV <sub>11</sub>	$\begin{cases} IV_8 = 0V_8 \\ 3V_{11} = IV_{11} \end{cases}$	= $\frac{2n}{2} \cdot \frac{2n-1}{3}$ .....	...	...	...	...	...	...	0	...	...			$\frac{2n}{2} \cdot \frac{2n-1}{3}$														
17	-	2V <sub>6</sub> - IV <sub>1</sub>	IV <sub>6</sub>	$\begin{cases} 2V_6 = 3V_6 \\ IV_1 = IV_1 \end{cases}$	= 2 n - 2 .....	1	...	...	...	2 n - 2																					
18	+	IV <sub>1</sub> + 2V <sub>7</sub>	IV <sub>7</sub>	$\begin{cases} IV_1 = IV_1 \\ 2V_7 = 3V_7 \end{cases}$	= 3 + 1 = 4 .....	1	...	...	...	...	4																				
19	÷	3V <sub>6</sub> + 3V <sub>7</sub>	IV <sub>9</sub>	$\begin{cases} 3V_6 = 3V_6 \\ 3V_7 = 3V_7 \end{cases}$	= $\frac{2n-2}{4}$ .....	...	...	...	2 n - 2	4	$\frac{2n-2}{4}$								$\left\{ \frac{2n}{2}, \frac{2n-1}{3}, \frac{2n-2}{3} \right\}$												
20	×	IV <sub>9</sub> × 4V <sub>11</sub>	IV <sub>11</sub>	$\begin{cases} IV_9 = 0V_9 \\ 4V_{11} = 5V_{11} \end{cases}$	= $\frac{2n}{2} \cdot \frac{3}{4} \cdot \frac{2n-2}{4} = A_3$ .....	...	...	...	...	...	0																				
21	×	IV <sub>22</sub> × 5V <sub>11</sub>	IV <sub>12</sub>	$\begin{cases} IV_{22} = 2V_{22} \\ 5V_{11} = 0V_{12} \end{cases}$	= B <sub>3</sub> · $\frac{2n}{2} \cdot \frac{2n-1}{3} \cdot \frac{2n-2}{4} = B_3 A_3$ .....	...	...	...	...	...	0																				
22	+	2V <sub>12</sub> + 2V <sub>13</sub>	IV <sub>12</sub>	$\begin{cases} 2V_{12} = 0V_{12} \\ 2V_{13} = 3V_{13} \end{cases}$	= A <sub>0</sub> + B <sub>1</sub> A <sub>1</sub> + B <sub>3</sub> A <sub>3</sub> .....	...	...	...	...	...	...	...	...	...	...		0		$\{ A_0 + B_1 A_1 + B_3 A_3 \}$												
23	-	2V <sub>10</sub> - IV <sub>1</sub>	IV <sub>10</sub>	$\begin{cases} 2V_{10} = 3V_{10} \\ IV_1 = IV_1 \end{cases}$	= n - 3 (= 1) .....	1	...	...	...	...	...	...	...	...	...	...	n - 3														
24	+	IV <sub>13</sub> + 0V <sub>24</sub>	IV <sub>24</sub>	$\begin{cases} 4V_{13} = 0V_{13} \\ 0V_{24} = IV_{24} \end{cases}$	= B <sub>7</sub> .....	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	B <sub>7</sub>		
25	+	IV <sub>1</sub> + IV <sub>3</sub>	IV <sub>3</sub>	$\begin{cases} IV_1 = IV_1 \\ IV_3 = IV_3 \end{cases}$ by a Variable-card. $\begin{cases} 4V_6 = 0V_6 \\ 0V_7 = IV_7 \end{cases}$ by a Variable card.	= n + 1 = 4 + 1 = 5 .....	1	...	n + 1	...	0	0																				

Here follows a repetition of Operations thirteen to twenty-three.

[https://fr.wikipedia.org/wiki/Ada\\_Lovelace](https://fr.wikipedia.org/wiki/Ada_Lovelace)

**Ada Lovelace**  
Premier programme informatique  
1843

# De la Pascaline à aujourd’hui



**Alan Turing**

Bases de la science informatique 1930

Machine de Turing, décryptage d'Énigma, IA



**Konrad Zuse**

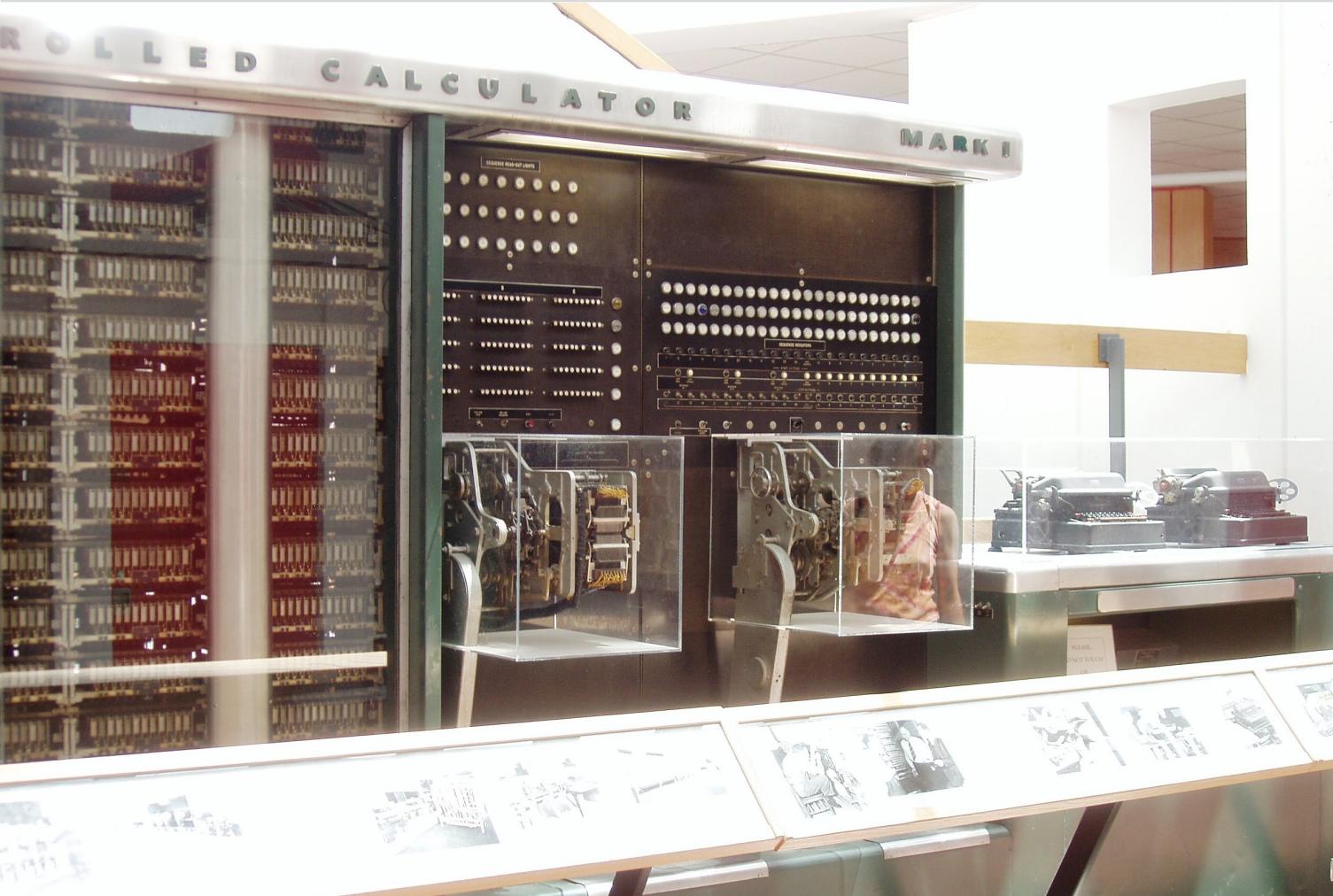
Z1, Z2, Z3, Z4 : 1938 -> 1950

Z3 : premier calculateur électromécanique programmable binaire

[https://fr.wikipedia.org/wiki/Alan\\_Turing](https://fr.wikipedia.org/wiki/Alan_Turing)

[https://fr.wikipedia.org/wiki/Konrad\\_Zuse](https://fr.wikipedia.org/wiki/Konrad_Zuse)

# De la Pascaline à aujourd’hui



## **l'ASCC d'IBM : 1944**

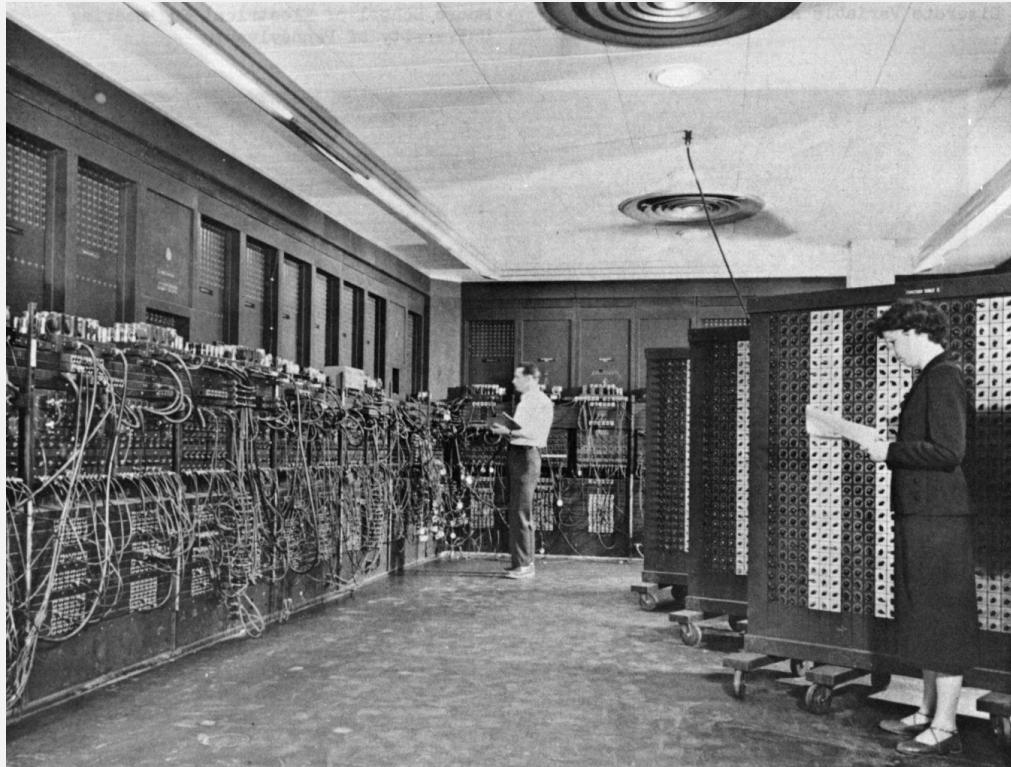
- 4,5 tonnes
- 2,4 x 0,5 x 16 m

## **Capacité de calcul :**

- 72 nombres avec 23 chiffres
- 3 additions / s
- 1 multiplication : 6s
- 1 division : 15,3 s



# De la Pascaline à aujourd’hui



## ENIAC : 1946

*Calculateur de trajectoires balistiques*

Premier ordinateur entièrement numérique à lampes :

- $2,4 \times 0,9 \times 30,5$  m occupant une surface de  $167\text{ m}^2$
- 30 tonnes
- 150 kW

## Capacité de calcul :

- 20 nombres à 10 chiffres
- 5 000 additions / s
- 357 multiplications / s
- 38 divisions / s

# De la Pascaline à aujourd'hui

- Premiers compilateurs : traducteur d'un langage à un autre (machine à l'époque)
  - 1952 – A-O : Grace Hopper
  - 1957 – Fortran : Équipe de John Backus
  - 1960 – COBOL : fonctionne sur plusieurs architecture avec le même code

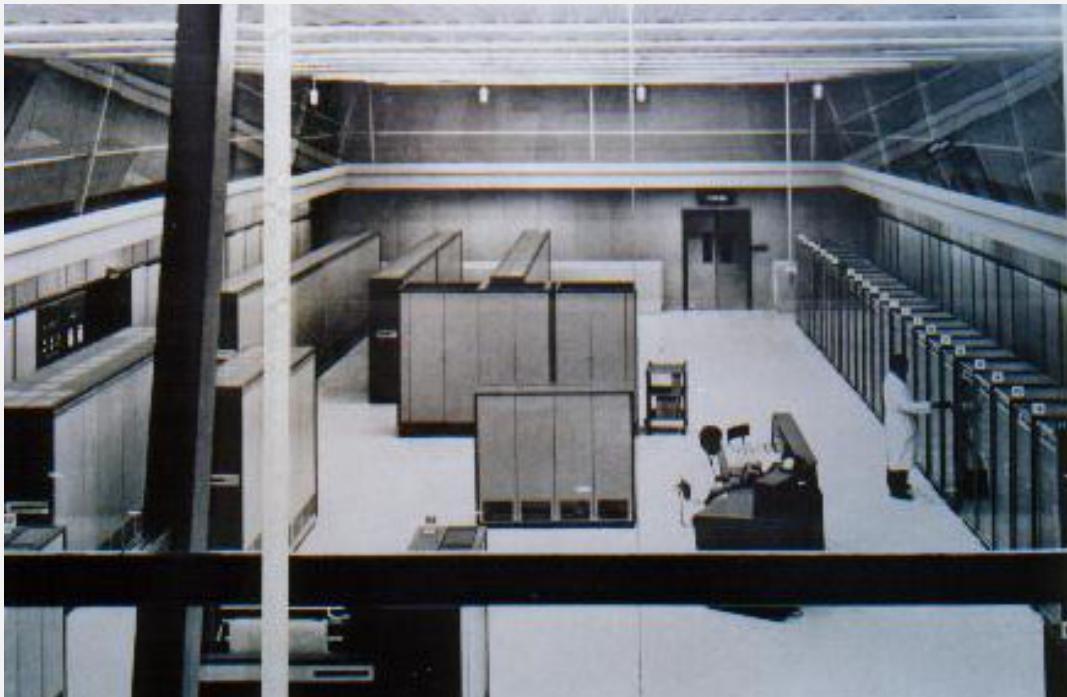
0800 Anton started  
1000 " stopped - anton ✓ { 1.2700  
1300 PRO 2  
033 CONC  
Relays 6-2 in 033 failed special s  
in Relays " 10.00 hrs  
Relays changed  
1100 Started Cosine Tape (Sine check)  
1525 Started Multi+ Adder Test.  
1545 Relay #70 P  
(Moth) in relay.  
First actual case of bug being found.



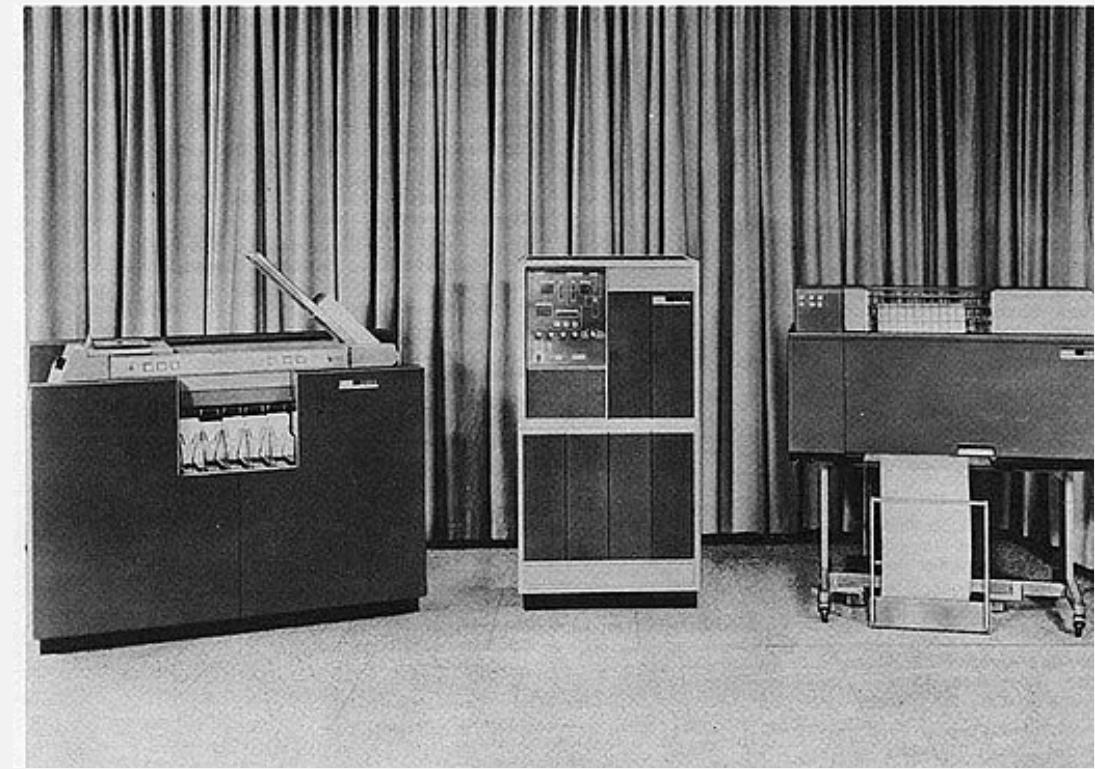
1951 - UNIVAC I

# De la Pascaline à aujourd’hui

Ordinateurs à transistors



**Gamma 60 de Bull - 1958**



**IBM 1401 - 1959**

# De la Pascaline à aujourd’hui

Ordinateurs à transistors



**Margaret Hamilton – Apollo 11 – 1969**  
OS avec priorisation des tâches  
Terme "Software Engineer"

# De la Pascaline à aujourd’hui

Ordinateur personnel



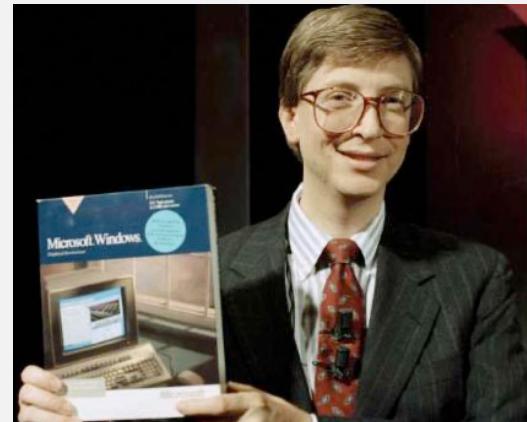
Apple I : 1976



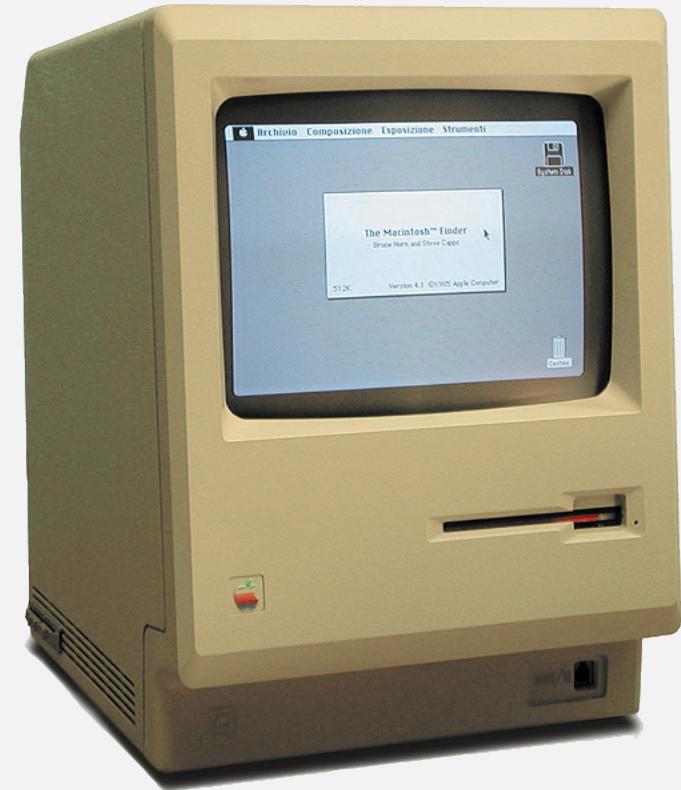
IBM PC - 1981



Apple IIc : 1984



Windows 1.0 : 1985



Macintosh 128K : 1984

# De la Pascaline à aujourd’hui

Mini ordinateur



**IBM AS/400 - 1987**

# De la Pascaline à aujourd’hui



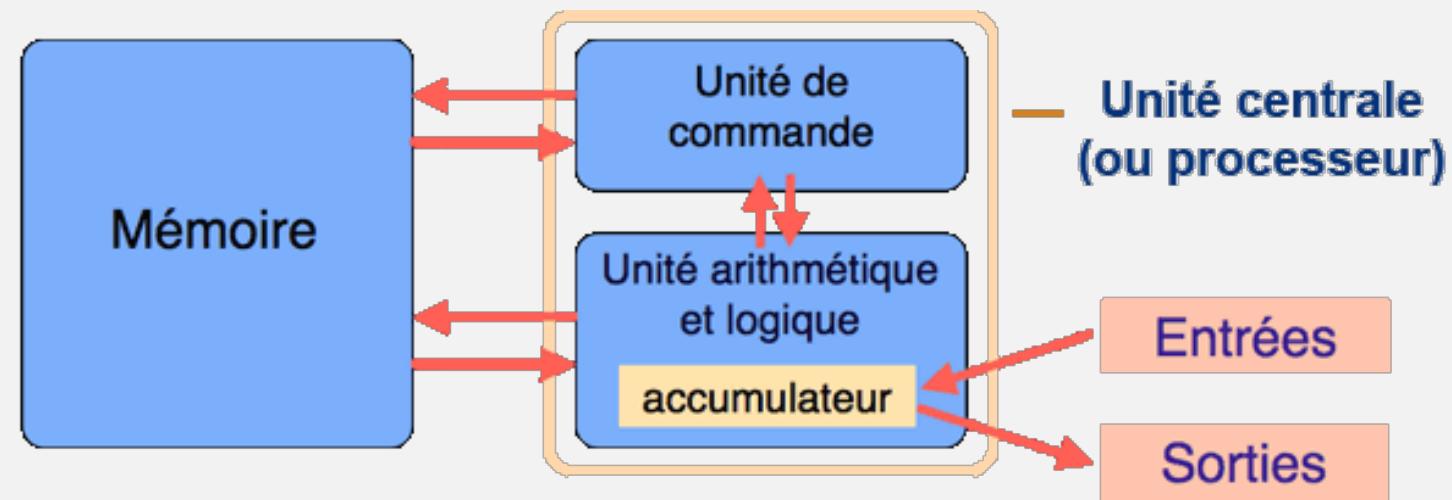
IBM Simon : 1994

Ordiphone (smartphone)



iPhone : 2007

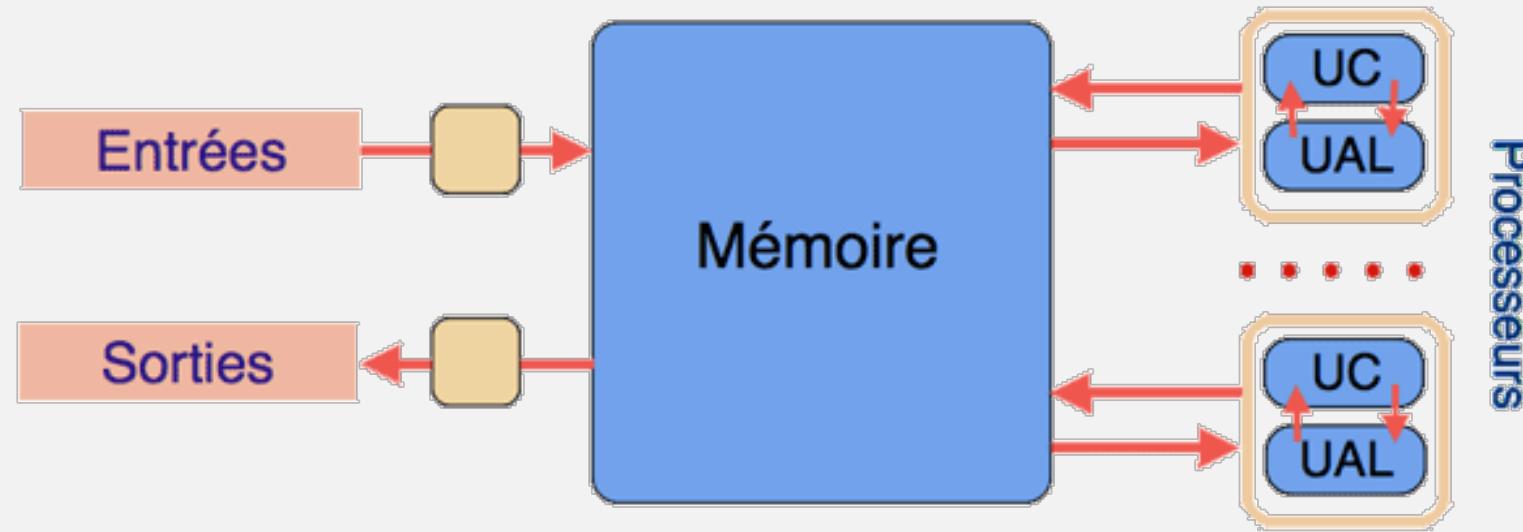
# Architecture de Von Neumann – 1945



Bases de l'ordinateur moderne :

- Séparation de l'unité de commande et de calculs
- Le programme est en mémoire plutôt que sur un support externe
- Données et programmes dans la mémoire
  - Un compteur contient l'emplacement (adresse) de l'instruction en cours
  - Un programme peut être traité comme une donnée

# Architecture de Von Neumann – Aujourd’hui



L'ordinateur moderne :

- Les entrées / sorties sont liées à la mémoire et non plus au processeur et passent par un contrôleur
- Processeurs multiples : processeurs et cœurs
- Mémoire au centre

# Composantes d'un ordinateur

- Mémoire centrale (RAM)
  - Stocke les données de manière temporaire
- Processeur (CPU)
  - Exécute les instructions d'un programme
- Pérophérique
  - Échange d'information avec le CPU et la RAM
  - Ex: Écran, clavier, souris, imprimante, ...

# Programme

- Suite d'indications, d'instructions
  - Comme une recette de cuisine
- Pour atteindre un objectif précis
- Spécifie:
  - Les opérations à effectuer
  - L'ordre de ces opérations
- Rôle de l'ordinateur:
  - Lire et exécuter ces instructions.

# Données

- Informations manipulées par le programme
  - Exemple d'un système bancaire:
    - Nom, Adresse, Numéro de Téléphone
    - Numéro de compte
    - Solde
    - Dossier d'hypothèque, cartes de crédit, etc.

# Données

- L'utilisateur fournit des données en entrée:
  - Directement par la saisie à l'écran
  - Par un fichier, sur l'ordinateur, ou sur internet
- Le programme traite ces données
  - Modifie l'information reçue en entrée
  - Crée de nouvelles données
    - Ex: débiter 100\$ de votre compte pour une facture
- Le programme produit de nouvelles données
  - A l'écran, dans un fichier, en impression, etc.
  - Ex: Afficher le nouveau solde de votre compte

# Mémoire



- Stocke :
  - Les données manipulées par les programmes en cours d'exécution
  - Les programmes eux même
- En binaire : 01010110010100100100...
- Autant les programmes que les données

# Mémoire

- Organisée en cases mémoires
  - Chaque donnée occupe une place précise (une adresse)
  - Cette adresse sert à placer et à récupérer les données

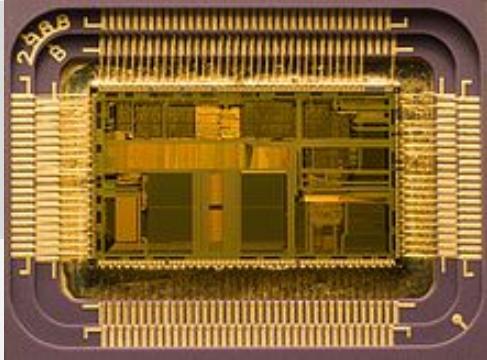
01001001	11001001	00011100	00000000	...
@0CF2	@0CF3	@0CF4	@0CF5	@0CF6

# Mémoire

- Deux opérations :
  - Lire une case mémoire à partir de son adresse
  - Écrire dans une case mémoire

01001001	11001001	00011100	00000000	...
@0CF2	@0CF3	@0CF4	@0CF5	@0CF6

# CPU



"Central Processing Unit"

- Récupérer des données de la mémoire centrale ou d'un périphérique
- Faire le traitement demandé par le programme
- Envoyer le résultat
  - Vers une case mémoire
  - Vers un périphérique

# CPU

- Opérations supportées:
  - Mathématiques
    - Addition, Soustraction, Multiplication, Division, etc.
  - Comparaisons de deux valeurs
    - Égales, différentes, plus grand/petit que, etc.
  - Envoie d'une donnée vers un périphérique
  - Récupération d'une donnée depuis un périph.
  - Lecture/Écriture d'une case mémoire

# Oui mais comment on programme ?

Avant, il faut revenir au but : Automatiser la résolution de problème

**Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre une classe de problèmes**

# Oui mais comment on programme ?

- Processeurs : exécute une suite d'opérations / instructions
- Instruction de processeurs : binaire (0 ou 1)
  - => Difficile à appréhender
  - => Difficile à lire
  - => Difficile à maintenir

Utilisation d'un **langage de programmation** qui lui est plus proche de l'humain

# Langage de programmation - Définition

Un langage de programmation est une **notation conventionnelle** destinée à **formuler des algorithmes** et produire des programmes informatiques qui les appliquent.

# Théorème programme structuré (1966)

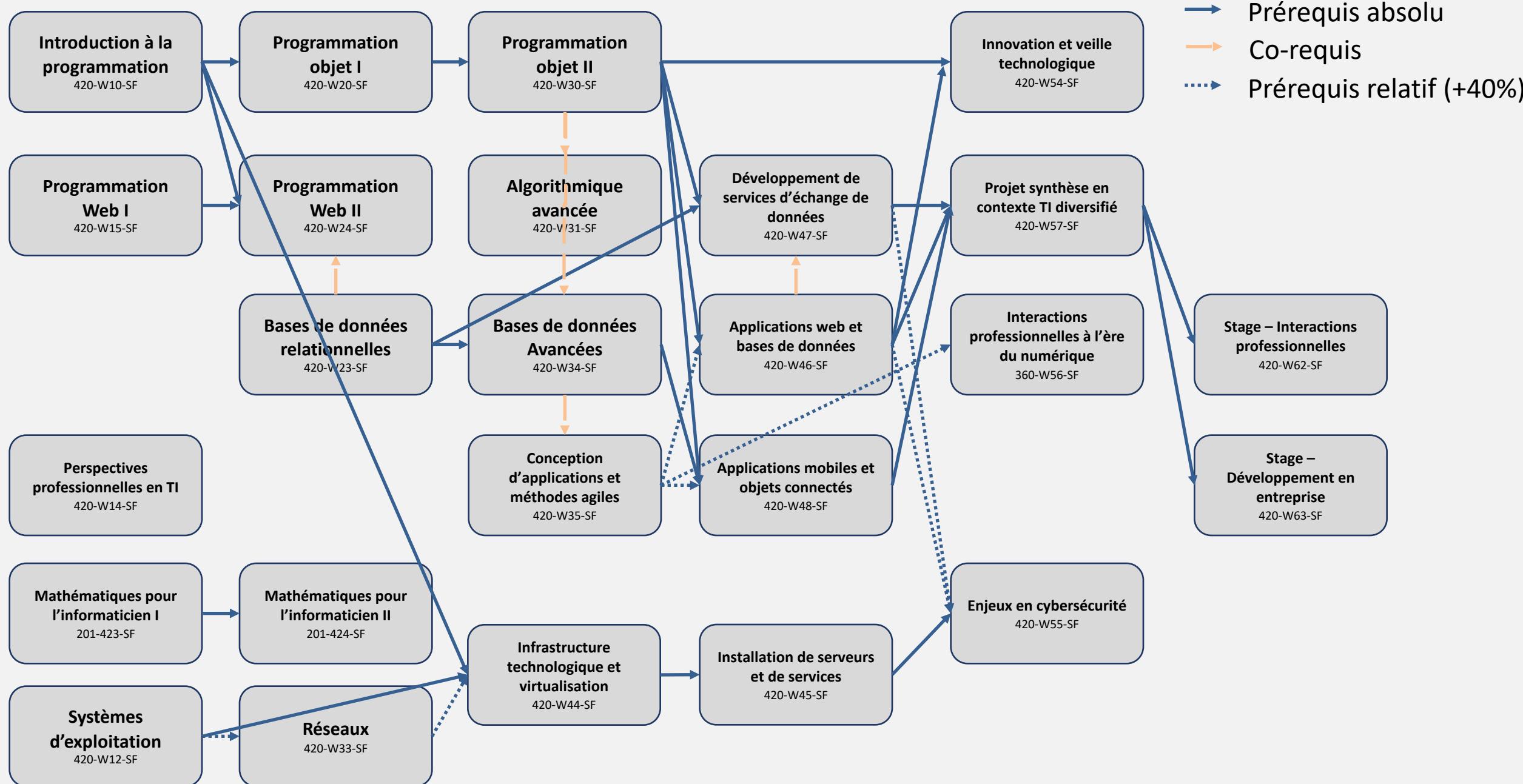
Une fonction calculable peut être représentée par l'assemblage de trois structures :

- Exécuter une **séquence** de sous-programmes
- Exécuter un des deux sous-programmes proposés en le **sélectionnant** suivant qu'une **condition** est vraie ou fausse
- **Répéter** l'exécution d'un sous-programmes tant qu'une condition est vraie

=> C'est ce que nous allons travailler ici !

# Nous... dans tout ça ?

- Bases algorithmique
  - Variables : stocker des données
  - Répétitions : exécuter une suite d'instructions tant que nécessaire
  - Alternatives : prendre des décisions
  - Fonctions : définir de nouvelles fonctionnalités réutilisables, simplifier l'écriture
- Langage : C#
- Environnement : Visual Studio 2019



# En attendant le prochain cours...

- À faire à la maison : <https://studio.code.org/s/express-2019> - leçons # 2 et # 4