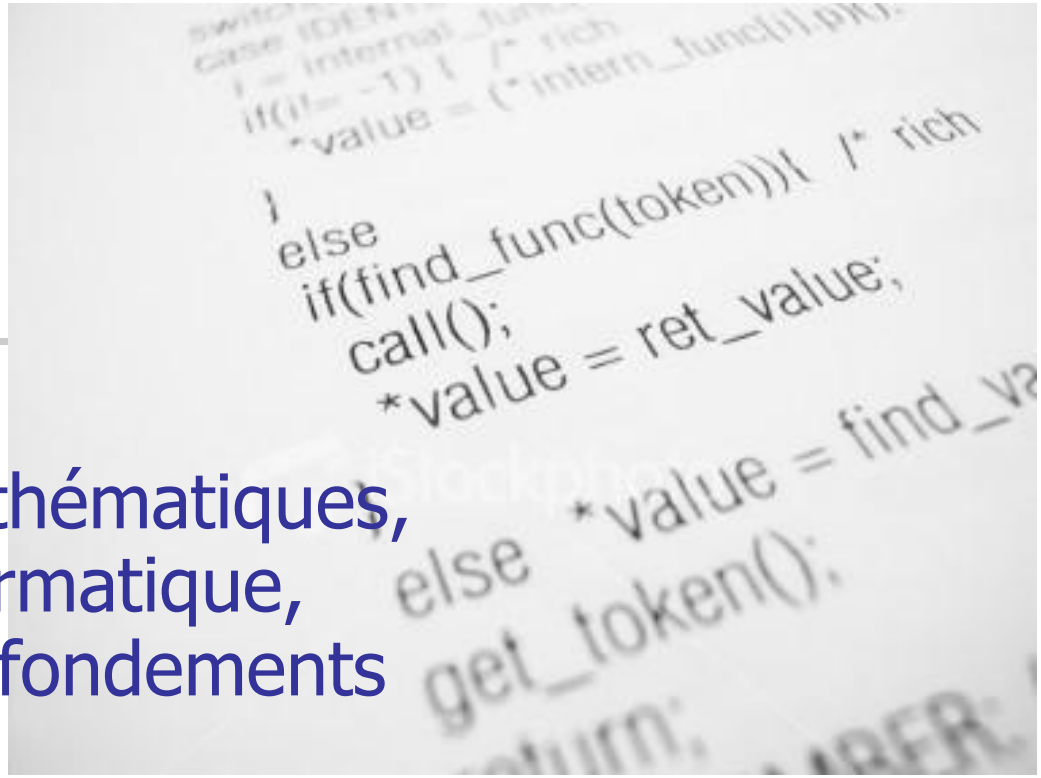


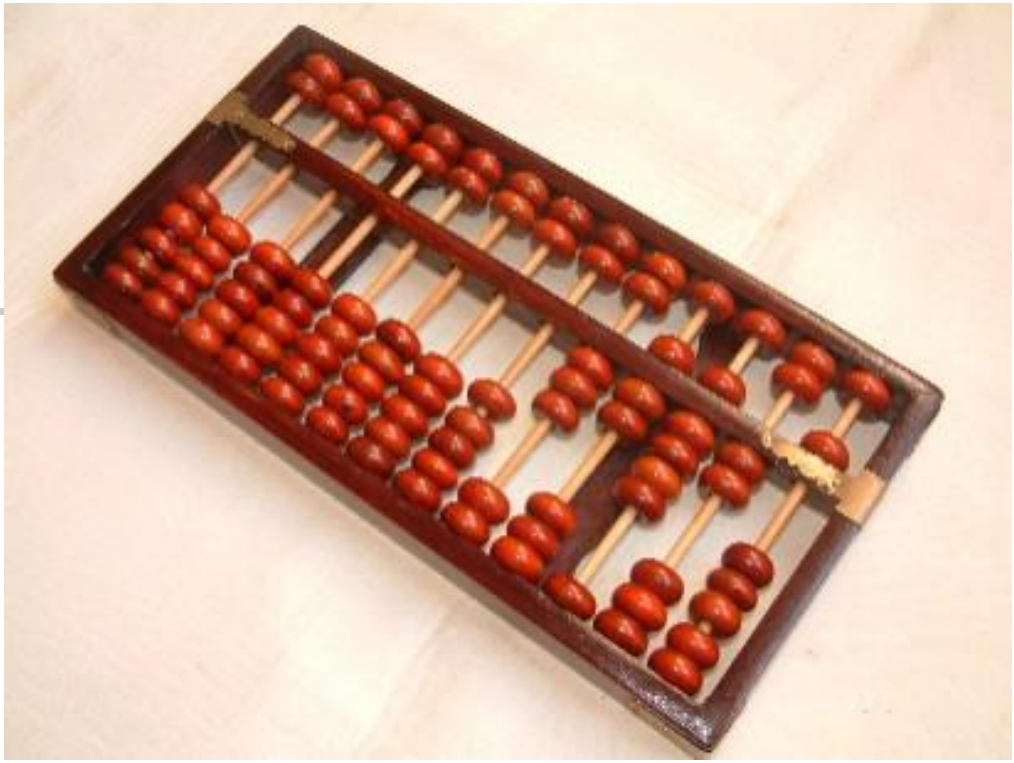
Fondements des mathématiques, fondements de l'informatique, la grande quête des fondements

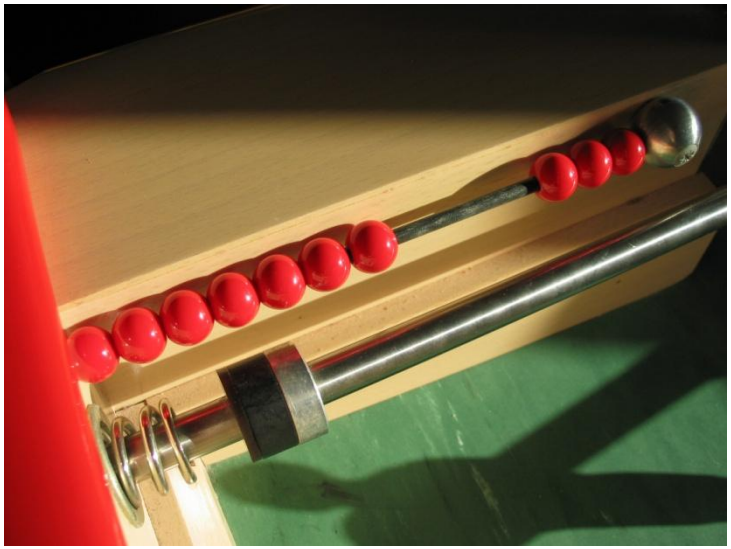
Luc Bougé, ENS Cachan/Bretagne

avec l'aide de David Pichardie, INRIA

L’origine du calcul ou la quête des fondements

Un peu d’histoire…



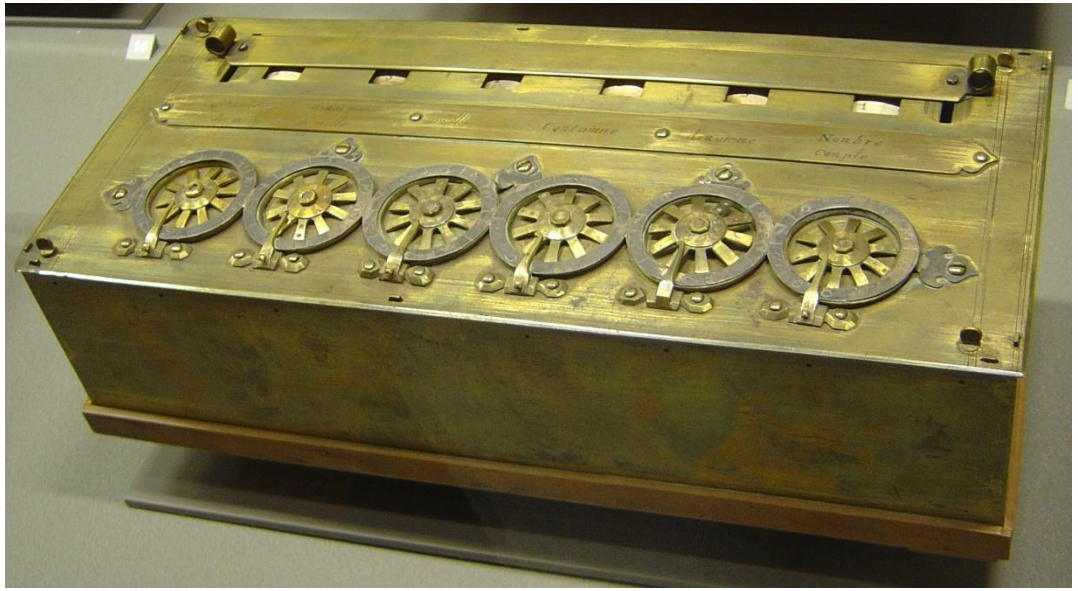


Computer?

- COMPUTER, verbe.
- *Rare* **A.–** *Emploi intrans.* Déterminer une date; calculer, supputer un temps : Quant aux ères, ici on compte par l'année de la création, là par olympiade (...). On **compute** encore par les ères julienne, grégorienne, ibérienne et actienne.
  Chateaubriand, *Génie du christianisme,* t. 1, 1803, p. 127.
- **B.–** *Emploi trans., littér.* [L'obj. désigne un objet quantifiable] Calculer, évaluer. *Si l'on computait tout ce qui est gagné par tous les avocats d'une grande ville* (Say, *Traité d'écon. pol.,* 1832, p. 366). *Une mécanique de bois et de métal qui (...) peut computer les tables astronomiques et nautiques jusqu'à n'importe quel point donné* (Baudelaire, *Nouv. Histoires extraordinaires,* trad. d'E. Poë, 1857, p. 383). *L'épouvantable catastrophe pécuniaire qu'il computait* (J. de La Varende, *Indulgence plénière,* 1951, p. 228).

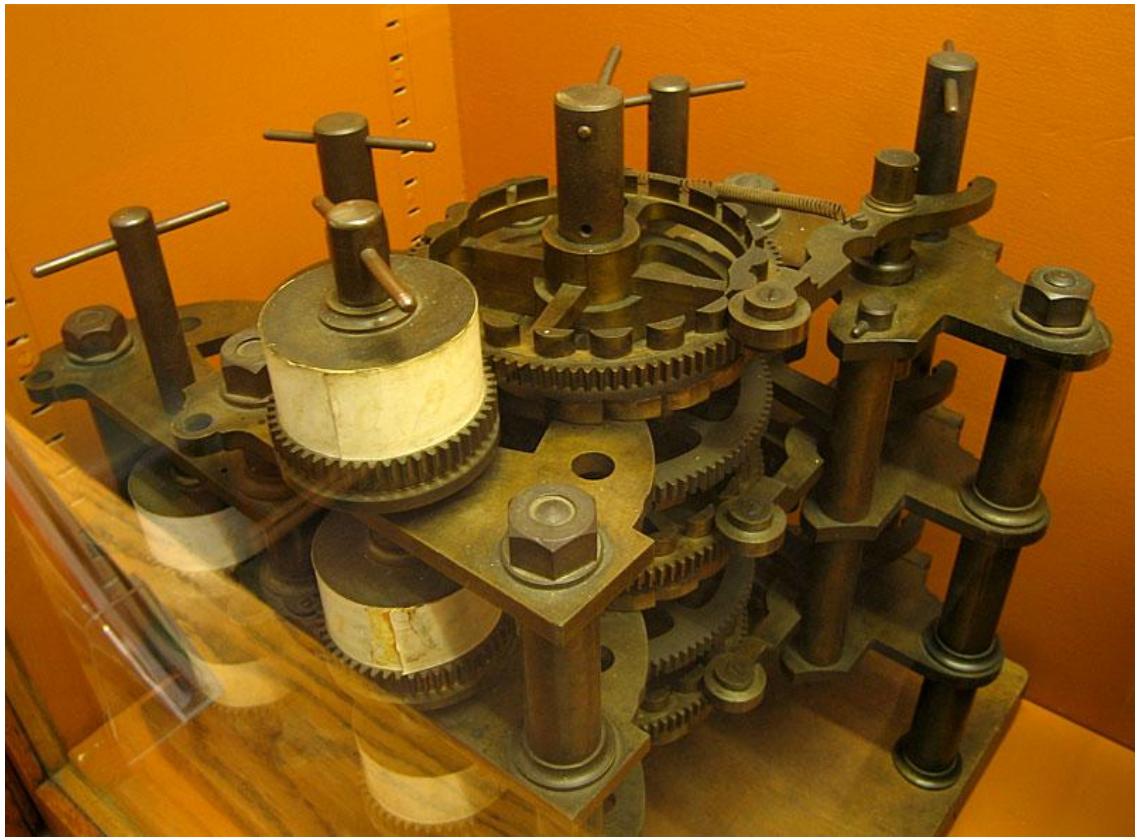
Les grands ancêtres: machines à computer

- Depuis toujours ou presque, le boulier
- Renaissance: calendriers rotatifs, cercles astronomiques, instruments de navigation
- 1649: Pascal invente sa machine à calculer et obtient l'exclusivité de la production de machines à calculer en France



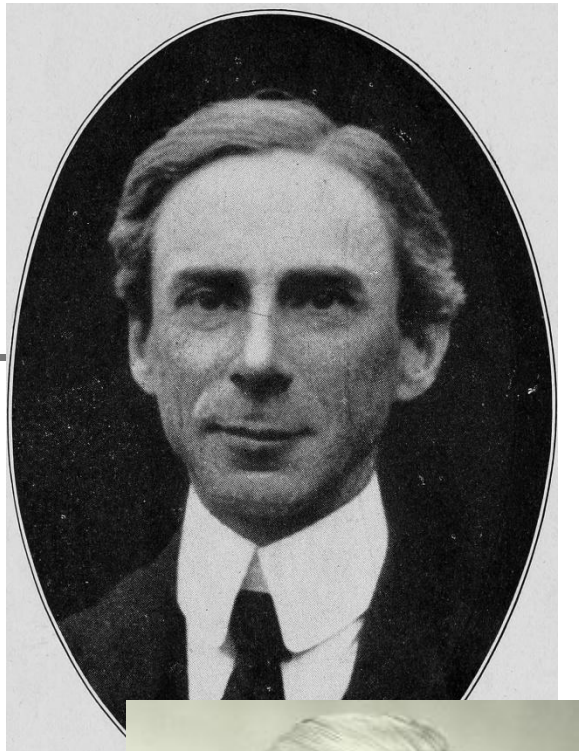
La première machine moderne

- 1791: Charles Babbage, premier ordinateur mécanique: machine à différences.
- Disciple: Ada de Lovelace
- Calcul des polynômes par une méthode différentielle
- Pas de notion de programme



Les grands fondateurs: la quête des fondements

- 1872-1970: Bertrand Russell
  - Philosophe et logicien anglais
- 1903-1995: Alonzo Church
  - Logicien américain
- Logiciens russes
- David Hilbert

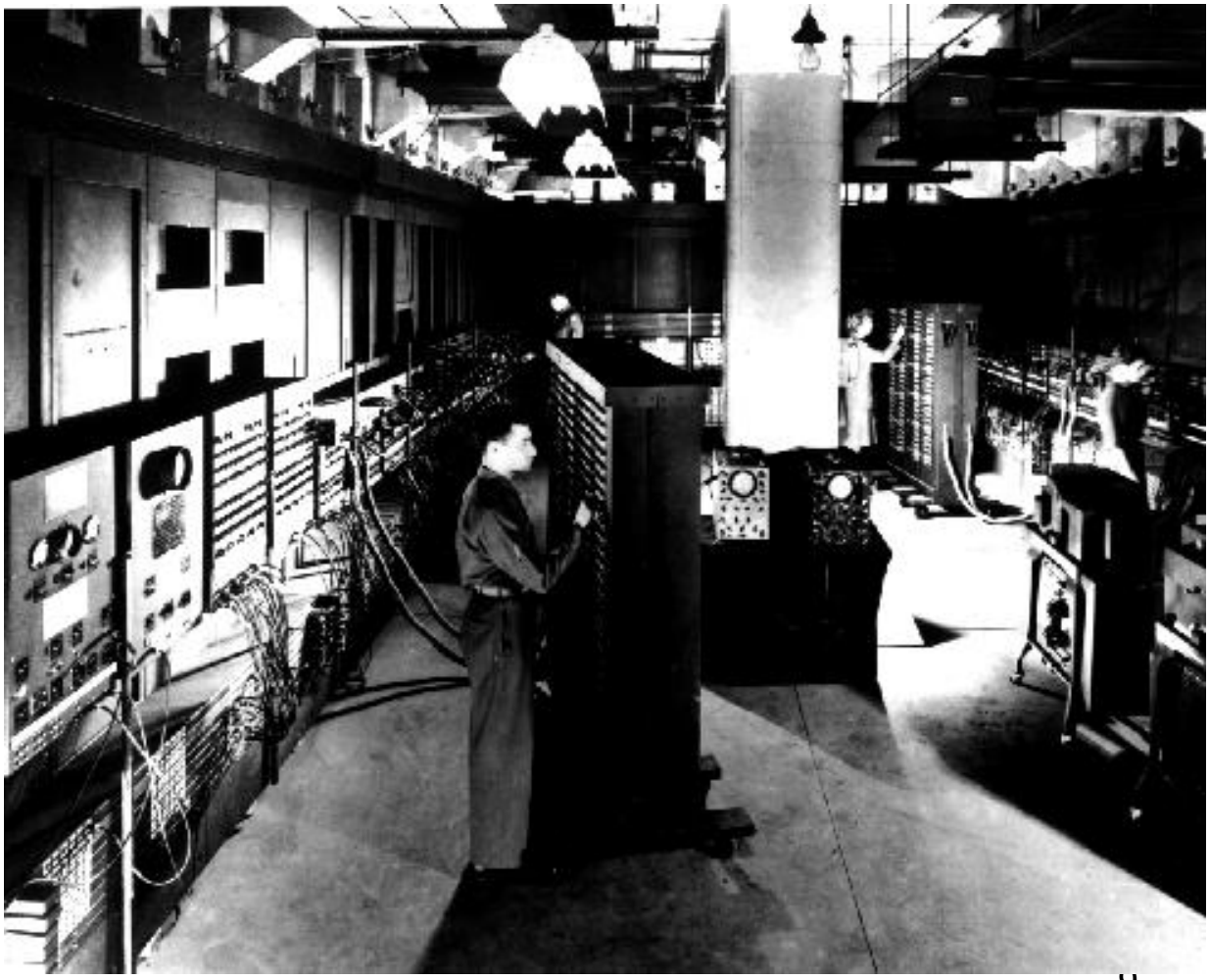


Les premiers ordinateurs

- 1912-1954: Alan Turing
  - Mathématicien, logicien anglais
  - cryptanalyste
- 1910-1995: Konrad Zuse
  - Ingénieur allemand, Z3 en 1941, (➔ Siemens)
- 1903-1957: John von Neumann
  - Mathématicien hongrois-américain
  - Manhattan Project, 1942-1946
- 1906-1978: Kurt Gödel
  - Mathématicien et logicien Autrichien-américain
  - Théorème d'incomplétude: notion de programme comme objet mathématique

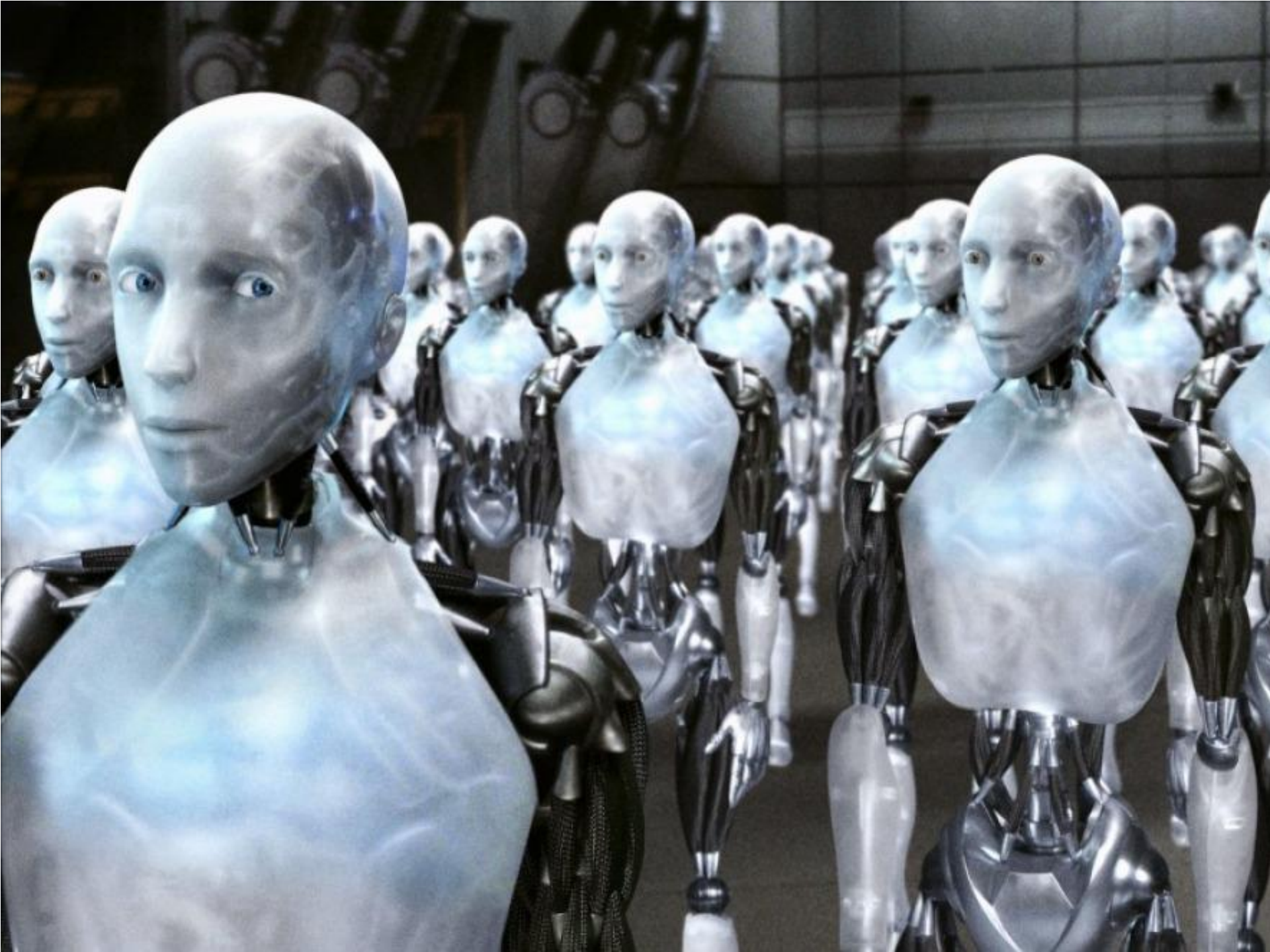
L'un des premiers ordinateurs: ENIAC

- Février 1946
- 18.000 tubes, 30 tonnes, 70 m², 150 kW
- Programmation par câblage
- 2.000 tubes remplacés chaque mois par 6 techniciens

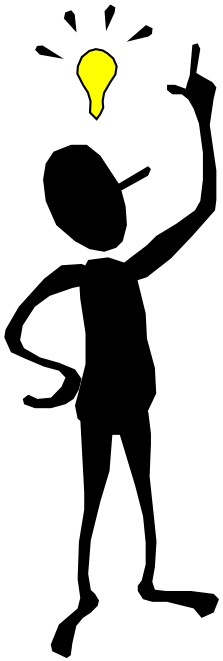


...et aussi le premier “bug”!

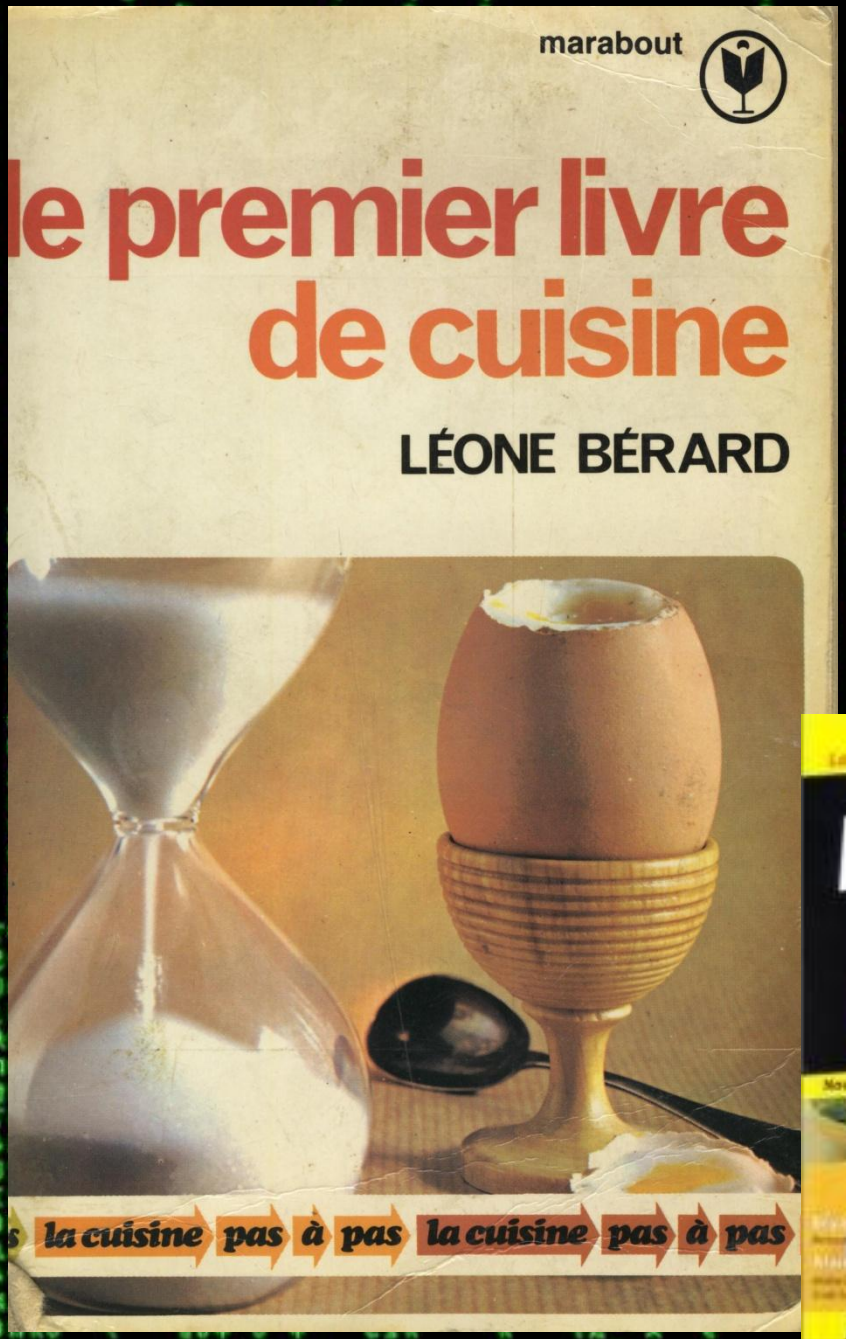
“Tandis que nous travaillions dans un immeuble non climatisé par une chaude et humide journée d'été, l'ordinateur s'arrêta. Nous finîmes par découvrir à l'intérieur d'un gros relais à signal un papillon de nuit grillé. Depuis ce jour, lorsqu'un officier entrait et nous demandait où en étaient les travaux, nous lui répondions que nous 'déboguions' le calculateur”

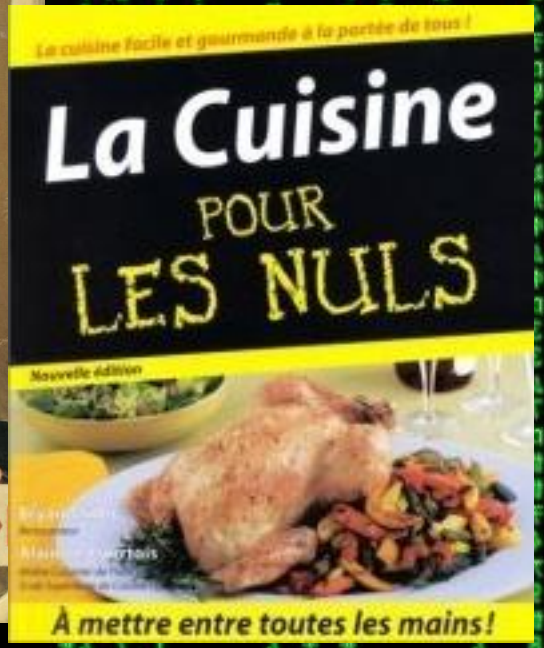


Mais

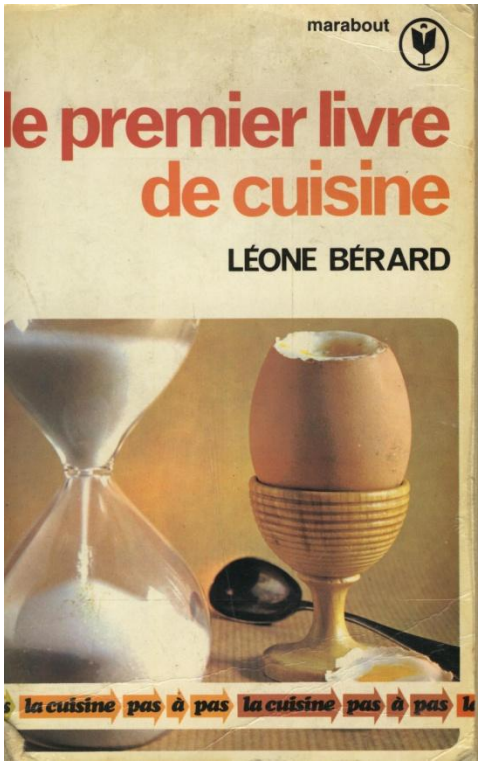


Qu'est-ce que calculer pour une machine?
Notion de programme


marabout
le premier livre de cuisine
LÉONE BÉRARD
la cuisine pas à pas la cuisine pas à pas


La cuisine facile et gourmande à la portée de tous !
La Cuisine
POUR
LES NULS
Nouvelle édition
À mettre entre toutes les mains !

Machine + Programme



17

CREPES ET BEIGNETS

On sert les crêpes et les beignets en desserts, traditionnellement pour le mardi gras et pour la mi-carême et, plus généralement, pendant les mois d'hiver. Ce sont, en effet, des préparations riches en farine et en corps gras et qui possèdent une grande valeur énergétique. Mais on utilise aussi crêpes et beignets en plats salés. Les crêpes fourrées de divers hachis sont alors gratinées au four et nommées, souvent, *pannequets*. Quant aux beignets, ils s'adressent pratiquement à tous les genres d'aliments (viandes, abats, poissons, légumes : voir p. 162).

Crêpes et beignets ne sont pas spécialement faciles à digérer. Mais, en général, tout se passe bien si on n'en abuse pas : une fois par semaine, deux au maximum, et si on leur associe dans les menus des plats et des aliments sans problèmes.

Les crêpes

LA PATE A CREPES

Travail personnel : 15 minutes. A préparer au moins 1 heure à l'avance.
Matériel : 1 terrine — 1 bol — 1 fouet — 1 torchon.

Denrées pour 4 personnes (12 crêpes de taille moyenne)

Farine : 125 g
Œufs : 2
Lait : 1 verre
Eau : 1 verre
Huile : 1 cuill. à soupe
Sel
Parfum : fleur d'oranger
rhum
vanille en poudre
zeste d'orange ou de citron, etc.

Mettre la farine dans la terrine. Ajouter une pincée de sel. Mélanger. Faire une fontaine.

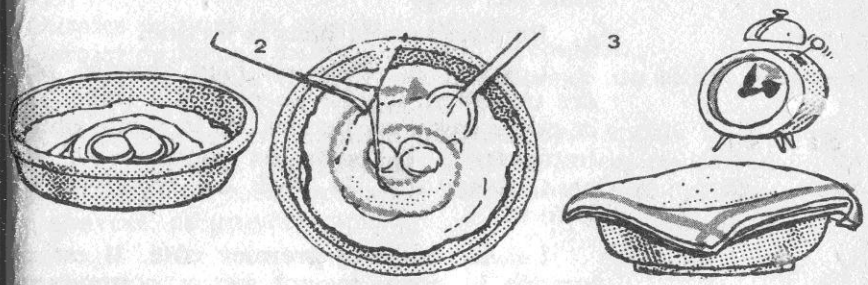
Casser les œufs un à un et les ajouter dans la fontaine.[1] Ajouter l'huile.

Commencer à mélanger en partant du centre vers les bords (la farine tombe peu à peu dans la partie un peu plus molle).

Verser peu à peu le liquide[2] (eau et lait, successivement ou après les avoir mélangés) toujours au centre.

Mélanger soigneusement jusqu'à ce que la pâte soit bien lisse. Ajouter le parfum choisi.

Laisser reposer 1 heure ou 2 après avoir recouvert la terrine d'un torchon propre.[3]



A CUISSON DES CREPES

Travail personnel : 25 minutes
Cuisson : 25 minutes environ
Matériel : 1 poêle épaisse à fond bien lisse — 1 louche — de quoi graisser la poêle — 1 palette — 1 plat.

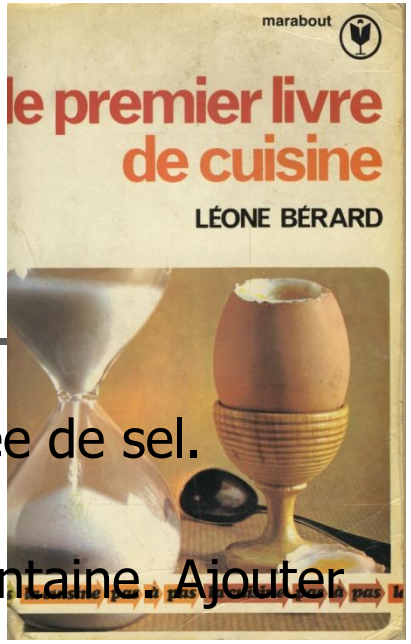
Comment graisser la poêle

La poêle doit être graissée entre deux crêpes, de manière à ce que celles-ci n'attachent pas (même lorsqu'on utilise une poêle à revêtement anti-adhésif, il vaut mieux graisser légèrement, tous les deux ou trois crêpes, par exemple). On peut utiliser l'un des trois systèmes suivants :

Ajouter une noisette de beurre et de margarine entre deux crêpes. Remarque : ce n'est pas une solution très pratique. Par ailleurs, c'est le procédé le plus riche en corps gras et ce corps gras, fragile, est surchauffé, ce qui le rend indigeste ;

Frotter la poêle entre deux crêpes avec un morceau de lard

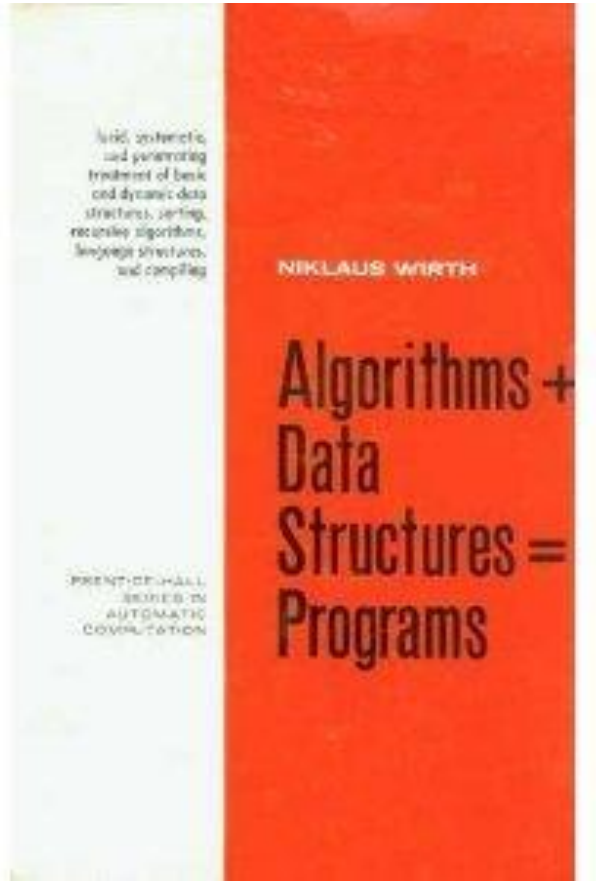
Instructions

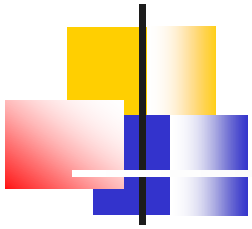


- Mettre la farine dans la terrine Ajouter une pincée de sel. Mélanger. Faire une fontaine
- Casser les œufs un à un et les ajouter dans la fontaine. Ajouter l'huile.
- Commencer à mélanger en partant du centre vers les bords (la farine tombe peu à peu dans la partie un peu plus molle).
- Verser peu à peu le liquide (eau et lait, successivement ou après les avoir mélangés) toujours au centre.
- Mélanger soigneusement jusqu'à ce que la pâte soit bien lisse. Ajouter le parfum choisi.
- Laisser reposer 1 heure ou deux après avoir recouvert la terrine d'un torchon propre.

Algorithmes +
Structures de données
= Programmes

Niklaus Wirth





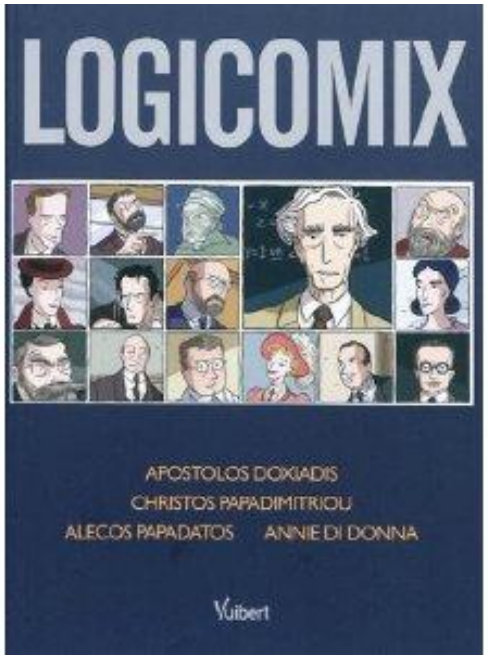
Un peu de syntaxe: instructions TOY

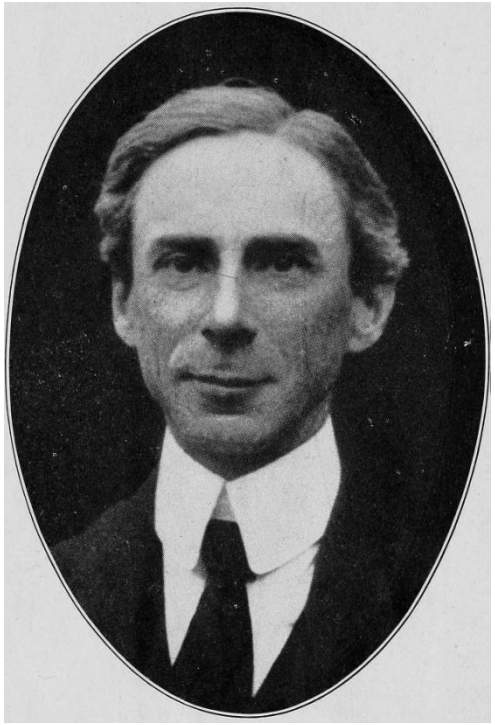
Ne rien faire	skip
Instruction simple	variable := expression
Séquence	(P; Q; R; …)
Choix simple	if b then P
Choix binaire	if b then P else Q
Itération tant que	while b do P
Itération jusqu'à	do P until b

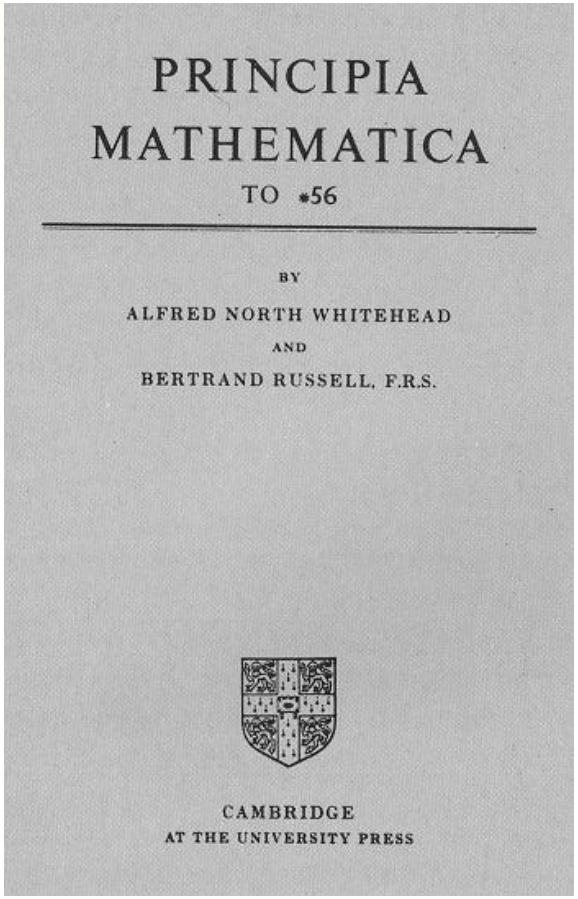
De quoi n'avons-nous pas parlé?

- Aspect temporel
  - Laisser reposer une heure ou deux
- Aspect non-déterministe
  - Parfum: fleur d'oranger ou rhum
- Aspect réactif
  - Arrêter quand le compte-minute sonne
- Aspect concurrent
  - Pendant que la pâte repose, faites chauffer la poêle

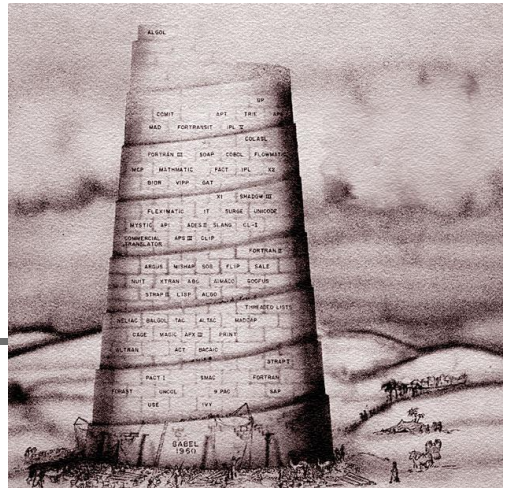
La question de l'expressivité: que peut-on calculer?







Bertrand Russell

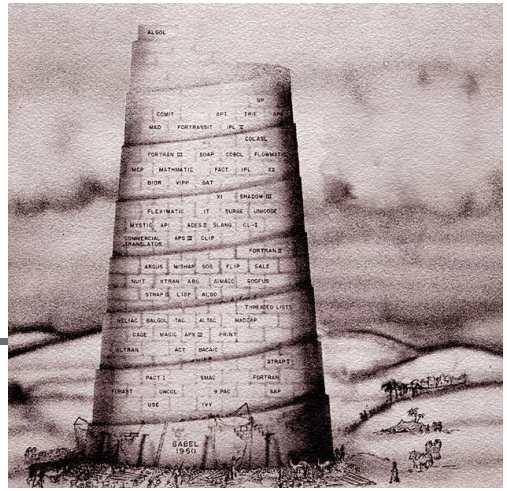


Expressivité

- Instruction élémentaire
  - skip
  - x := e
- Séquence
- Choix
- Itération tant que

- Question: est-ce que ça suffit pour exprimer tous les algorithmes?

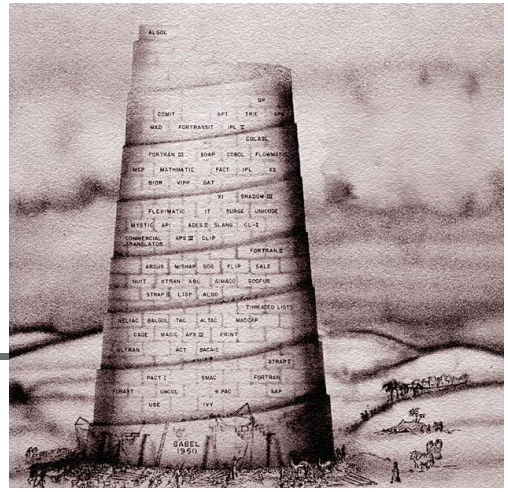
Définitions dérivées



- do P until b
  - P; while not b do P
- if b then P
  - if b then P else skip
- if b then P else Q
  - tmp = b; if tmp then P else (if not tmp then Q)

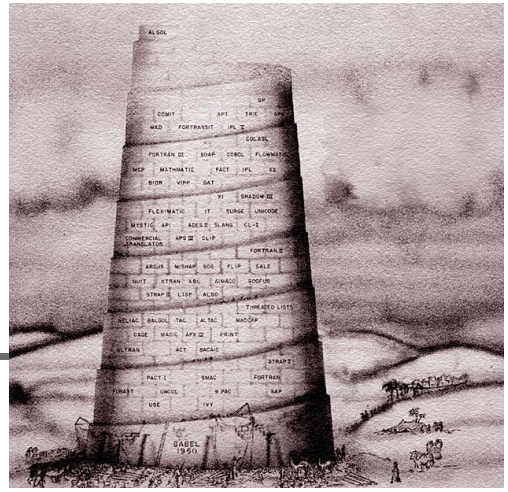
Notion de variable fraîche

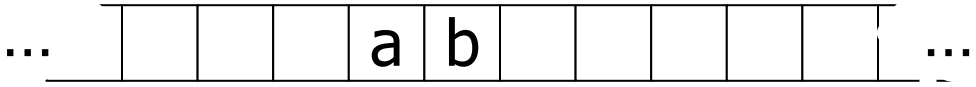
Qu’est-ce que calculer… pour un élève de primaire?

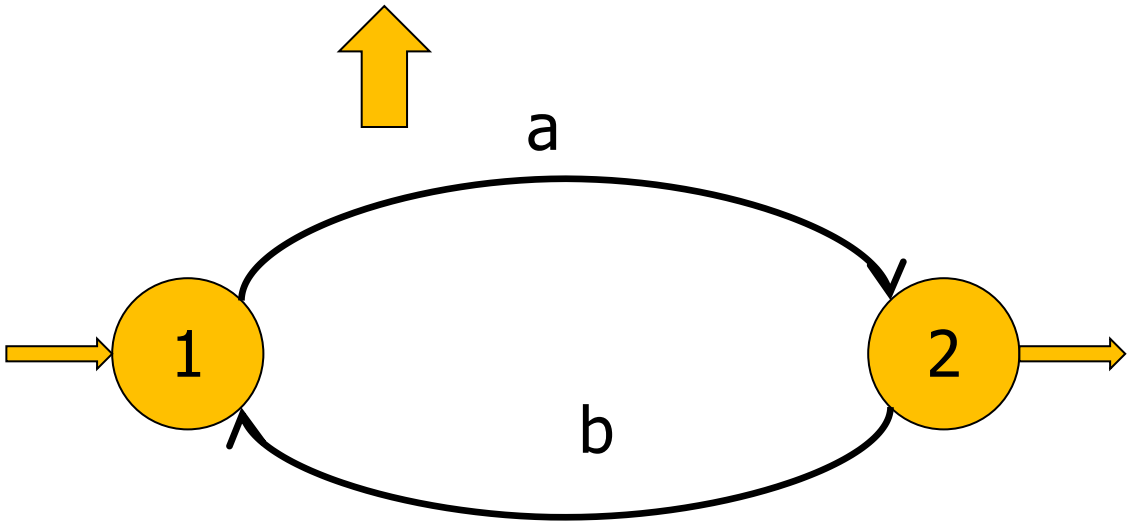


	9	2	9
+		8	3
1	0	1	2

Le calcul vu comme un objet mathématique: notion d'automate

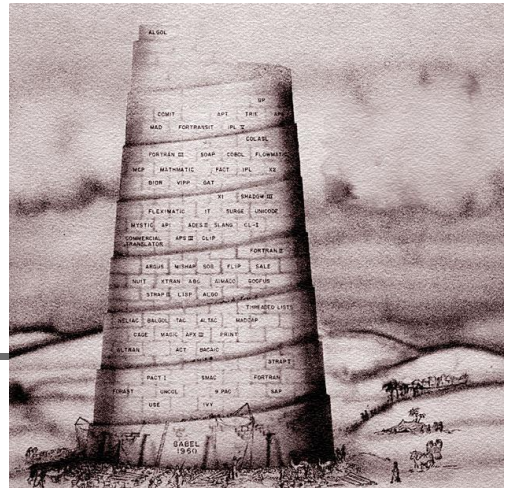


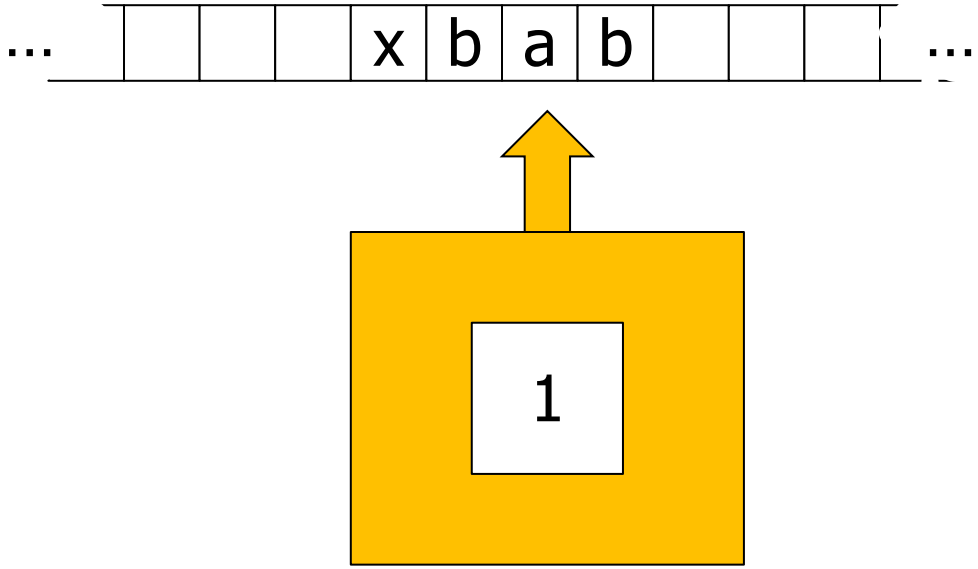




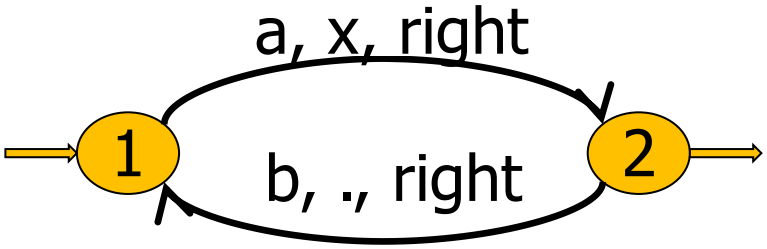
abababab

Machine de Turing

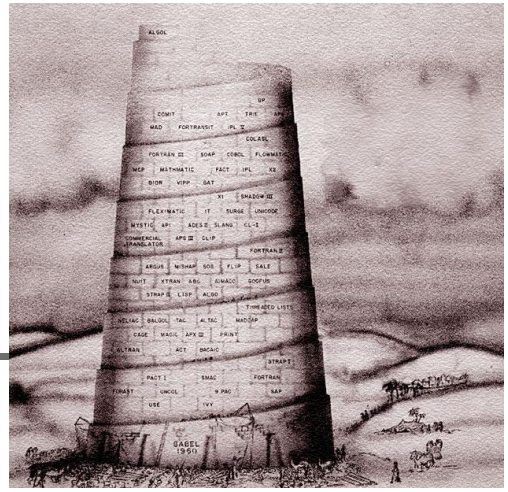


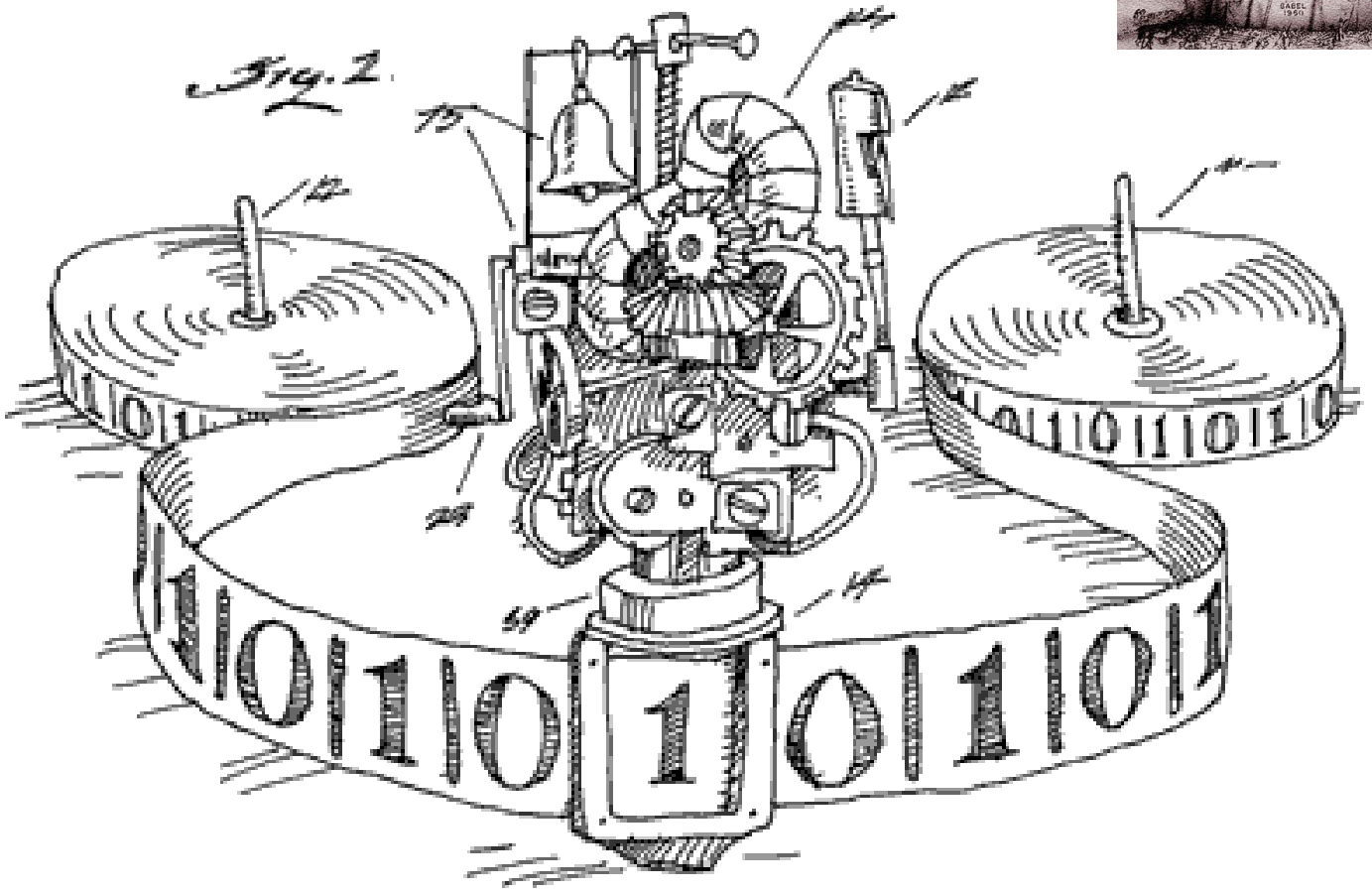


(1, a) ➔ (2, x, right)
(2, b) ➔ (1, ., left)

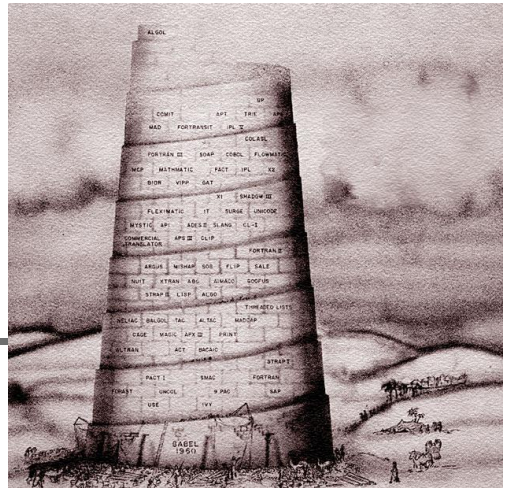


Machine de Turing, pour les élèves de lycée





La machine de Turing pour les mathématiciens ☺



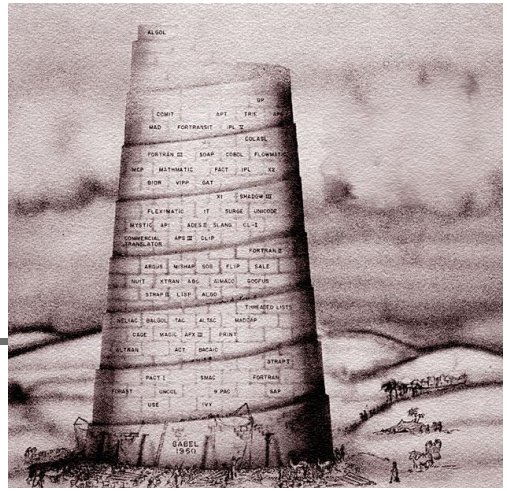
Une machine de Turing est un septuplet $(Q,\Gamma,B,\Sigma,q_0,\delta,F)$ où

- Q est un ensemble fini d'*états* ;
- Γ est l'*alphabet de travail* des symboles de la bande;
- $B \in \Gamma$ est un symbole particulier (dit *blanc*) ;
- Σ est l'*alphabet* des symboles en entrée ($\Sigma \subseteq \Gamma \setminus \{B\}$) ;
- $q_0 \in Q$ est l'*état initial* ;
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \rightarrow\}$ est la *fonction de transition* ;
- $F \subseteq Q$ est l'ensemble des *états acceptants* (ou finaux, terminaux).

Les flèches dans la définition de δ représentent les deux déplacements possibles de la tête de lecture, à savoir le déplacement à gauche et le déplacement à droite. La signification de cette fonction de transition peut être expliquée sur l'exemple suivant : $\delta(q_1, x) = (q_2, y, \leftarrow)$ signifie que si la machine de Turing est dans l'état q_1 et qu'elle lit le symbole x, elle écrit y à la place de x, va dans l'état q_2, et déplace sa tête de lecture vers la gauche.

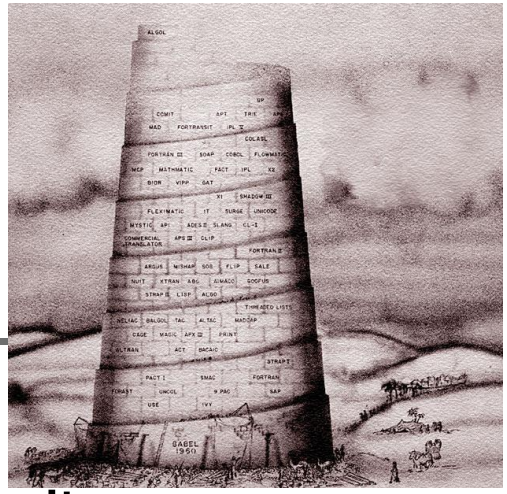
Le fonctionnement de la machine de Turing est alors le suivant. À chaque étape de son calcul, la machine évolue en fonction de l'état dans lequel elle se trouve, et du symbole inscrit dans la case du ruban où se trouve la tête de lecture. Ces deux informations permettent la mise à jour de l'état de la machine grâce à la fonction de transition. À l'instant initial, la machine se trouve dans l'état q_0, et le mot inscrit sur le ruban est l'entrée du programme. La machine s'arrête lorsqu'elle rentre dans un état terminal. Le résultat du calcul est alors le mot inscrit sur le ruban.

Ma première machine… de Turing



- Entrée: **a**aaaaa#aaabaa
- Idée:
  - Se positionner sur la première lettre à gauche
  - marquer cette lettre, puis se déplacer à droite du signe # pour trouver la première lettre non marquée
  - Si cette lettre est différente de la dernière lettre marquée, on a terminé
  - Sinon, on marque cette lettre et on se déplace à gauche du signe # pour trouver la dernière lettre non marquée

Exécution du programme



- **a**aaaaa#abaaaa
  - On marque et on se déplace à droite en mémorisant “a”
- aaaaaa#**a**baaaa
  - C’est un “a” → on continue
  - On marque et on se déplace à gauche
- a**a**aaaa#abaaaa
  - On se déplace à droite en mémorisant “a”
- aaaaaa#**ab**aaaa
  - C’est un “b” → on s’arrête dans l’état **“Echec”**
- Si on épuise les lettres, **“Réussite”**

Le programme de la machine de Turing

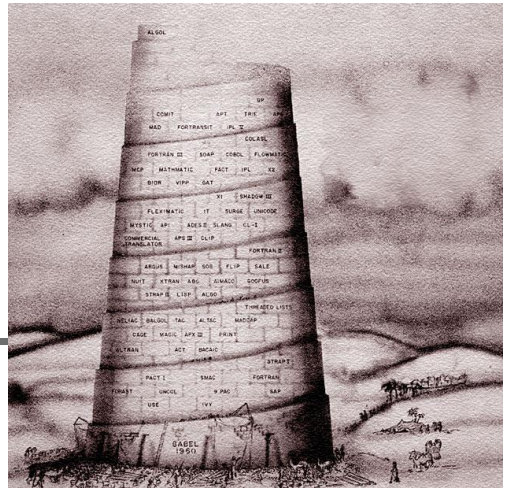
```
Titre Test d'égalité
// u#v sur la bande
// teste si u = v
// curseur à gauche
de u

Init . 0
aaaaaa#aaaaaa

0 . 0 . D
```

```
// mémoriser le
symbole
// et le marquer
0 a 10 A D
0 b 11 B D
0 # 90 - D

// aller vers v
10 - 10 - D
10 # 20 - D

11 - 11 - D
11 # 21 - D
```

```
// trouver le premier
symbole non
marqué
20 A 20 - D
20 B 20 - D
20 a 30 A G
20 - 99 - =  // échec

21 A 21 - D
21 B 21 - D
21 b 30 B G
21 - 99 - =  // échec
```

```
// retourner vers u
30 - 30 - G
30 # 40 - G

40 A 0 - D
40 B 0 - D
40 - 40 - G

// fin de u
90 a 99 - =
90 b 99 - =
90 . 100 - =
90 - 90 - D
```

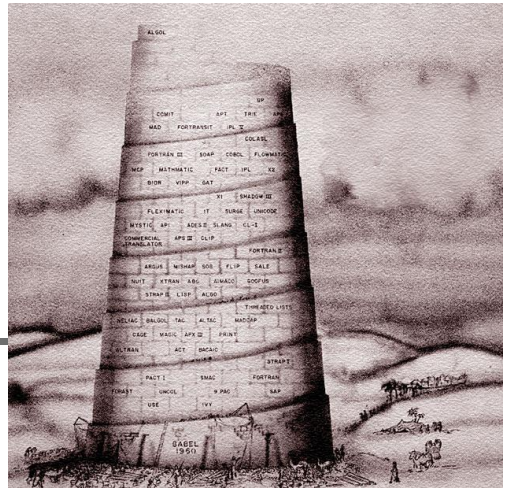
Et maintenant, ça marche pour de vrai! (Si, si!)



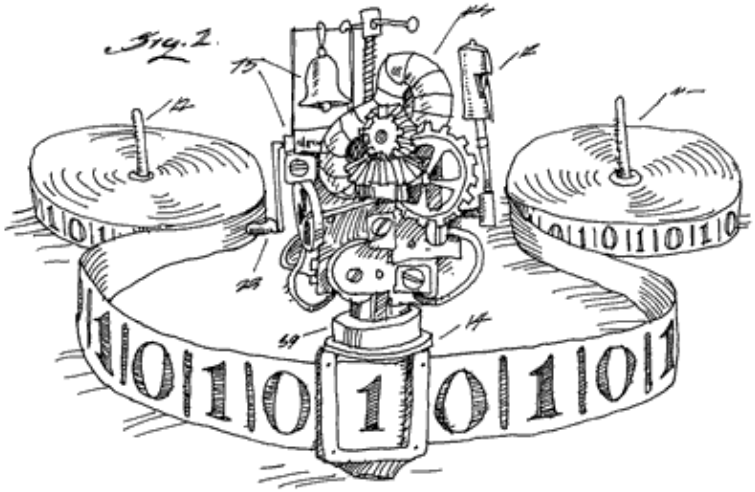
http://www.labri.fr/perso/betrema/MC/TuringJNLP.html

LaBRI:
Laboratoire bordelais d'informatique

Quelques machines de Turing à expérimenter soi-même

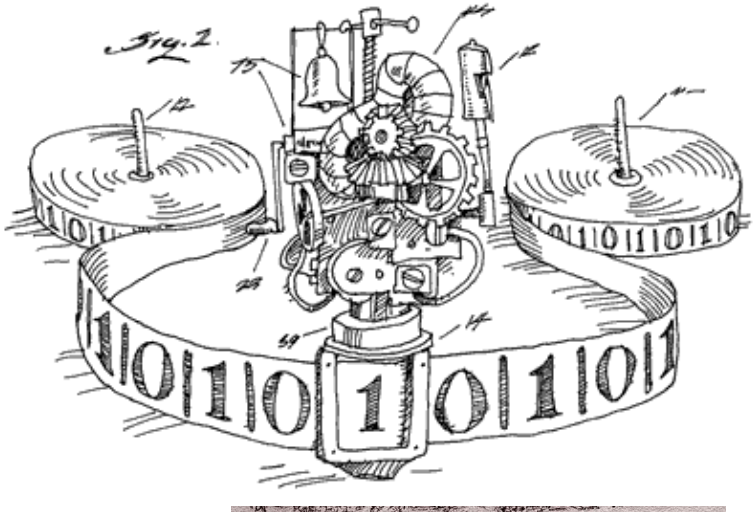


- Le langage TEMPL
  - Jean-Baptiste Mélès, ENS Paris
  - http://baptiste.meles.free.fr/?Le-langage-TMPL
- Analogies visuelles et animations
  - Warthmann associates, Technical Writers, Palo Alto
  - http://www.warthman.com/ex-turing.htm
- Interstices, le site de médiation du CNRS et de l'INRIA
  - Une autre applet Java
  - http://interstices.info/jcms/c_43049/machine-de-turing



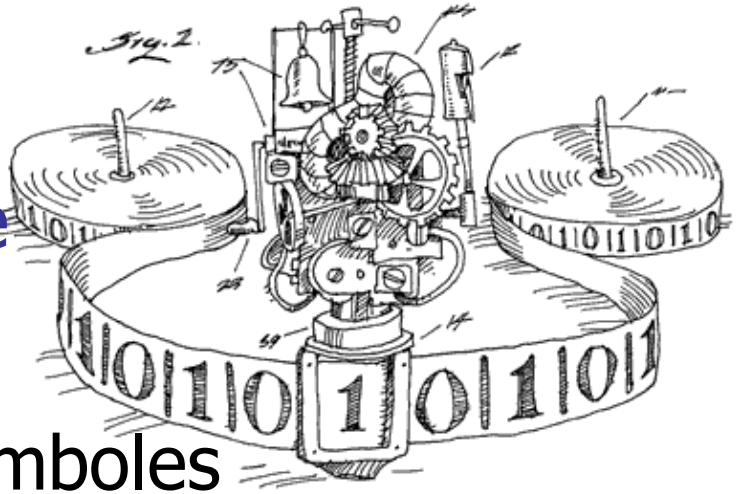
La puissance expressive de la machine de Turing

Question 1: Est-ce que le modèle est stable?



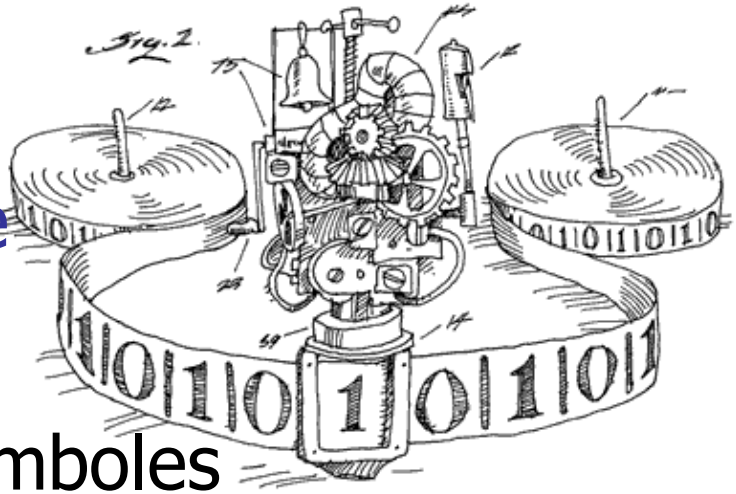
- Automate simple
- Automate avec une pile
- Automate avec un ruban infini en lecture/écriture
- Un demi-ruban infini
- Plusieurs têtes de lecture
- Plusieurs têtes de L/E et plusieurs rubans
- Un ruban multi-dimensionnel

Est-ce qu'il est nécessaire d'avoir un ruban?

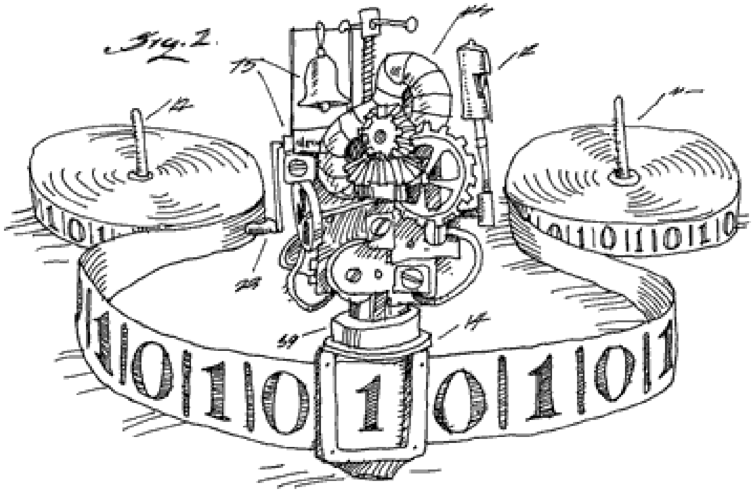


- Ruban = suite finie de symboles
  - Chiffres dans une base B
- On peut recoder les chiffres en base 2
  - 3 symboles: 0, 1, #
- On coupe le ruban au niveau de la tête de L/E
  - 2 demi-rubans
- Ils ne sont accédés que par leurs extrémités
  - push, pop
- Machine à 2 piles

Est-ce qu'il est nécessaire d'avoir un ruban?



- Ruban = suite finie de symboles
  - Chiffres dans une base B
- Etat du ruban + position = 2 nombres écrits en base B
  - 543210/6789
  - Mouvements = opérations arithmétiques
- Aller à gauche
  - gauche = (gauche/10)*10 + droite%10
  - droite = droite/10
- Machine à compteurs

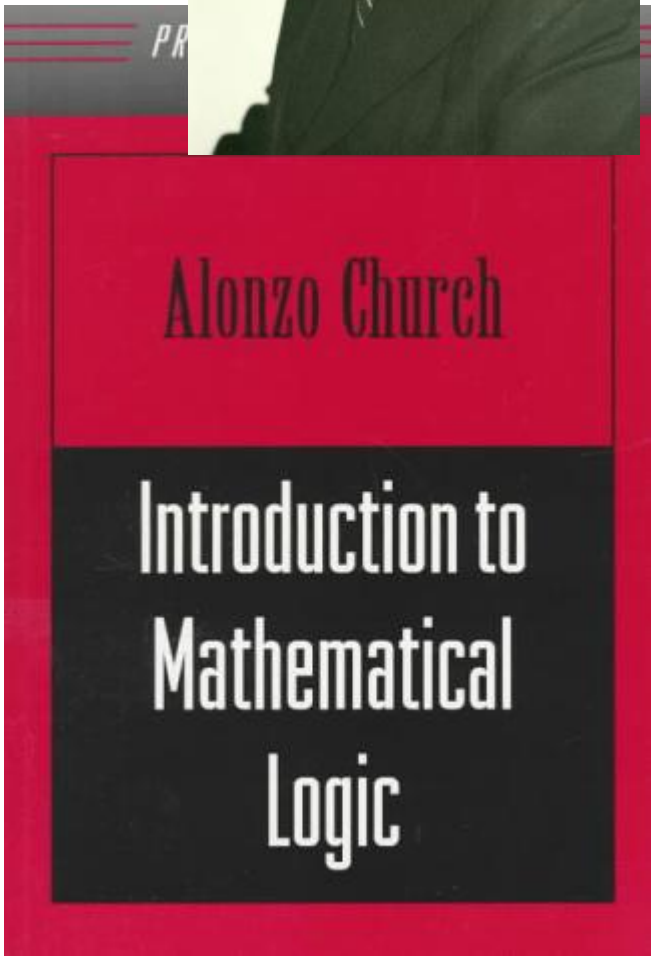


Conclusion sur la stabilité

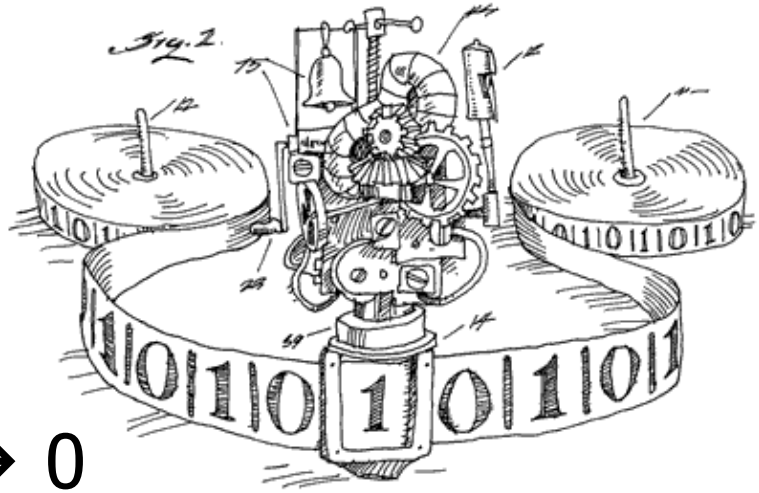
- Modèle intuitif
  - Interprétation spatio-temporelle intuitive
- Modèle extrêmement stable
  - Rubans, têtes, symboles, états
  - Réductions *simples* (polynomiales)
- Modèle de la même famille que les modèles habituels
  - Machine à 2 piles
  - Machine à 3(2) compteurs

Question 2: Est-ce que le modèle est expressif?

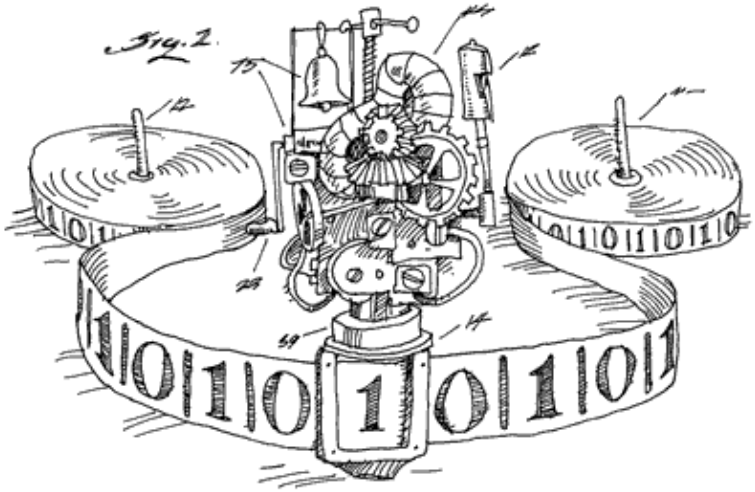
- Quelle est la classe des fonctions qui sont calculables par une MT?
  - Church: les fonctions définies par des termes du lambda-calcul
  - Church, Kleene, Rosser: les fonctions qui peuvent être calculées par des schémas récursifs



Les fonctions récursives primitives sur N



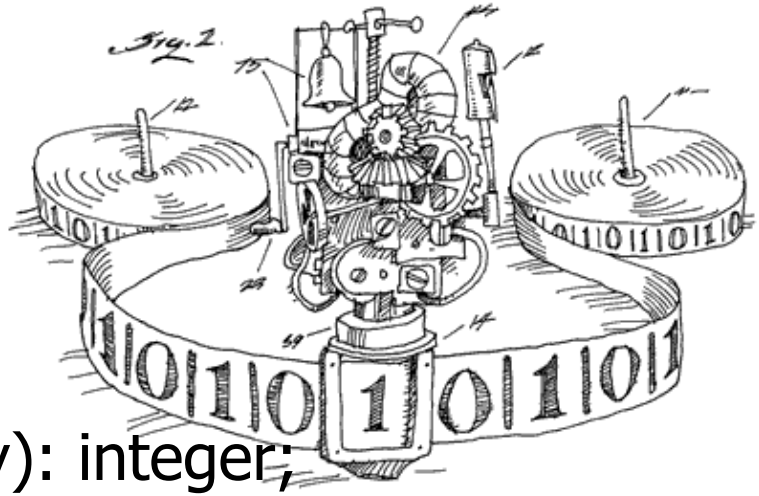
- Fonction nulle: zero(x) → 0
- Successeur: succ(x) → x+1
- Projection: $(x_1, x_2, \ldots, x_n) \rightarrow x_i$
- Composition: $f(x) = h(g_1(x), g_2(x), \ldots, g_n(x))$
- Schéma récursif:
  - $f(0, y_1, y_2, \ldots, y_n) = g(y_1, y_2, \ldots, y_n)$
  - $f(succ(x), y_1, y_2, \ldots, y_n) =$
    $h(x, f(x, y_1, y_2, \ldots, y_n), y_1, y_2, \ldots, y_n)$



Que peut-on exprimer?

- Constante
  - Const(x) = succ(succ(zéro(x)))
- Addition
  - plus(0, y) = y
  - plus(s(x), y) = succ(plus(x, y))
- Multiplication
  - mult(0, y) = 0
  - mult(s(x), y) = plus(mult(x, y), y)

Une vision informatique



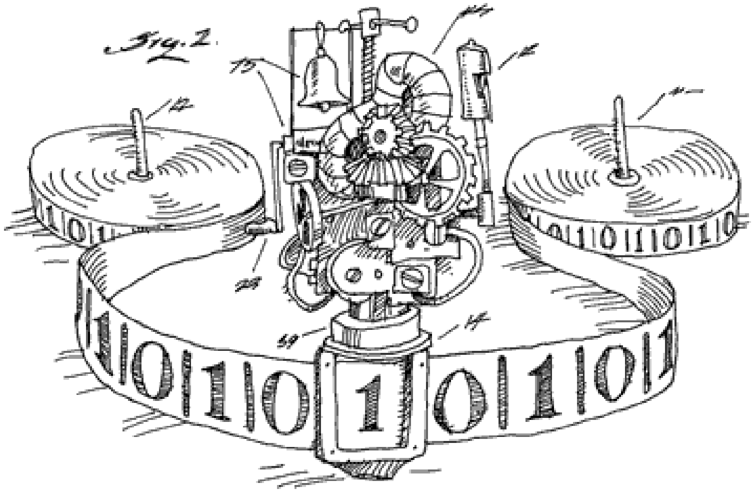
```
function f00(integer x, integer y): integer;
    var acc: integer;
    begin
        acc := g(y);
        for i := 1 to x do acc := h(x, acc, y);
        f00 := acc      (* return z *)
    end
```

Boucle for

$f(0, y) = g(y)$

$f(succ(x), y) = h(x, f(x, y), y)$

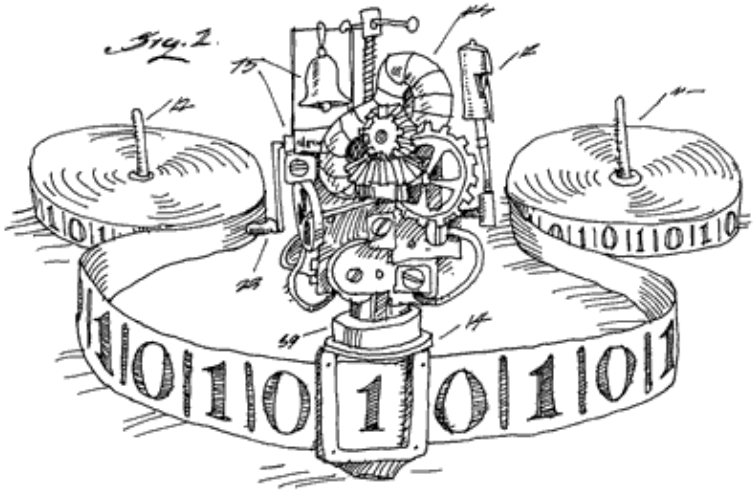
Peut-on tout exprimer?



- Fonction d'Ackermann
  - $Ack(0, p) = succ(p)$
  - $Ack(succ(n), 0) = Ack(n, 1)$
  - $Ack(succ(n), succ(p)) = Ack(n, Ack(succ(n), p))$

- $Ack(0, p) = p+1$
- $Ack(1, p) = p+2$
- $Ack(2, p) = 2p+3$
- $Ack(3, p) = 2\string^(2\string^(2\string^\ldots))-3$ *(p+3 termes)*

Les fonctions récursives générales

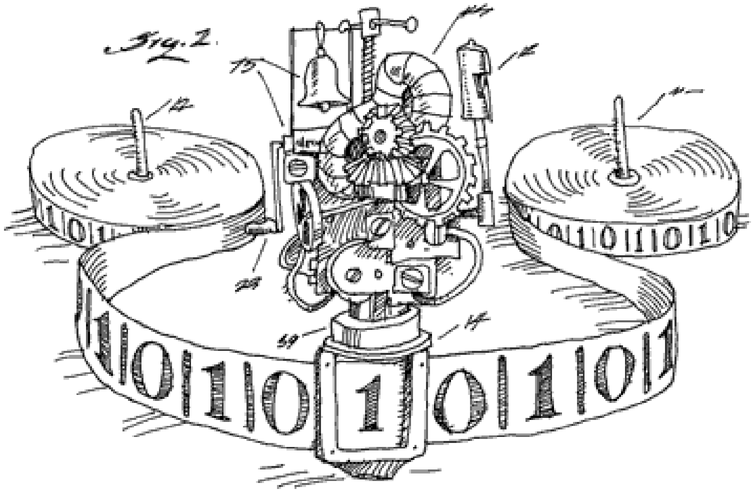


- $\mu_f(x_1,\ldots,x_n) = \inf_y\{f(y,x_1,\ldots,x_n) = 0\}$
  - Si un tel y n'existe pas, μ_f n'est pas définie en $(x_1,\ldots,x_n)$

y := 0; while f(y,x) <> 0 do y := y + 1

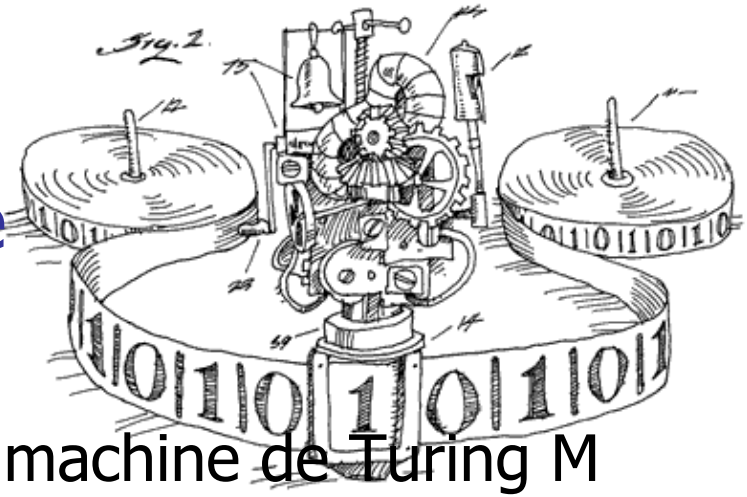
Boucle while

Théorème fondamental (Kleene, etc.)



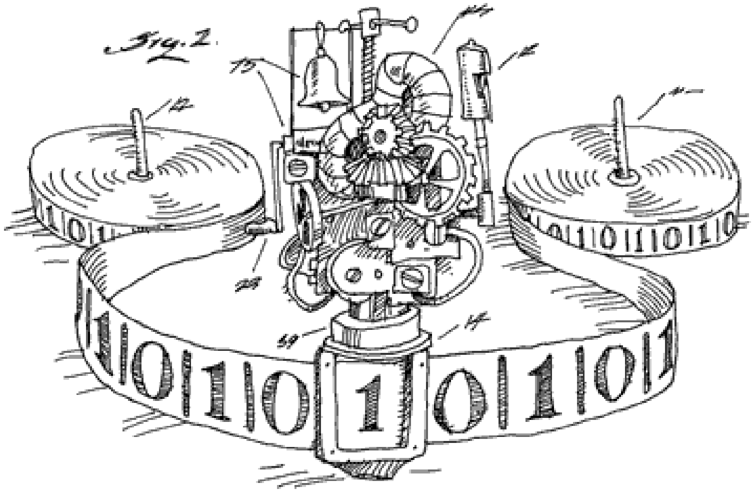
- L’ensemble des fonctions constructibles par le schéma précédent est exactement l’ensemble des fonctions engendrables par une machine de Turing.

Question 3: Est-ce que le modèle est réflexif?



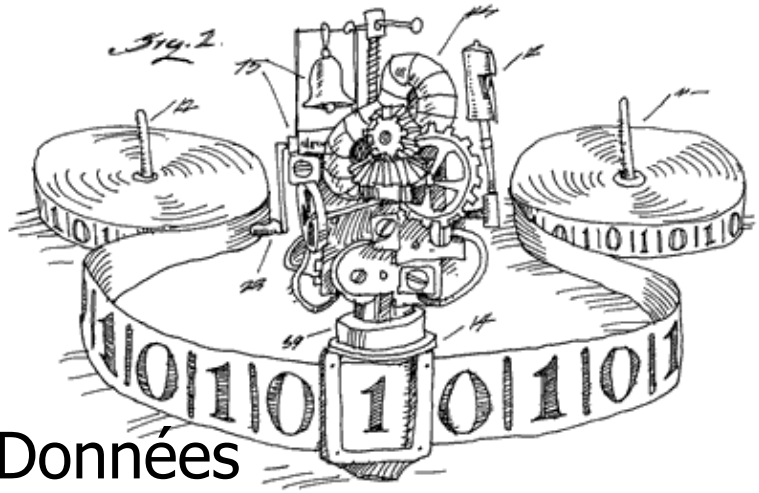
- Il est possible de décrire une machine de Turing M par une suite finie de symboles
  - Ensemble d'états, symboles, règles
  - Description(M)
- Il existe une machine de Turing avec la propriété suivante:
  - Entrée: #Description(M)#chaîne(s)#
  - S'arrête si et seulement si M s'arrête sur l'entrée s, et avec la sortie de M(s) sur sa bande
  - Le temps d'arrêt est une fonction simple du temps d'arrêt de M sur s
- Machine de Turing universelle

Thèse de Church-Turing (vers 1950)

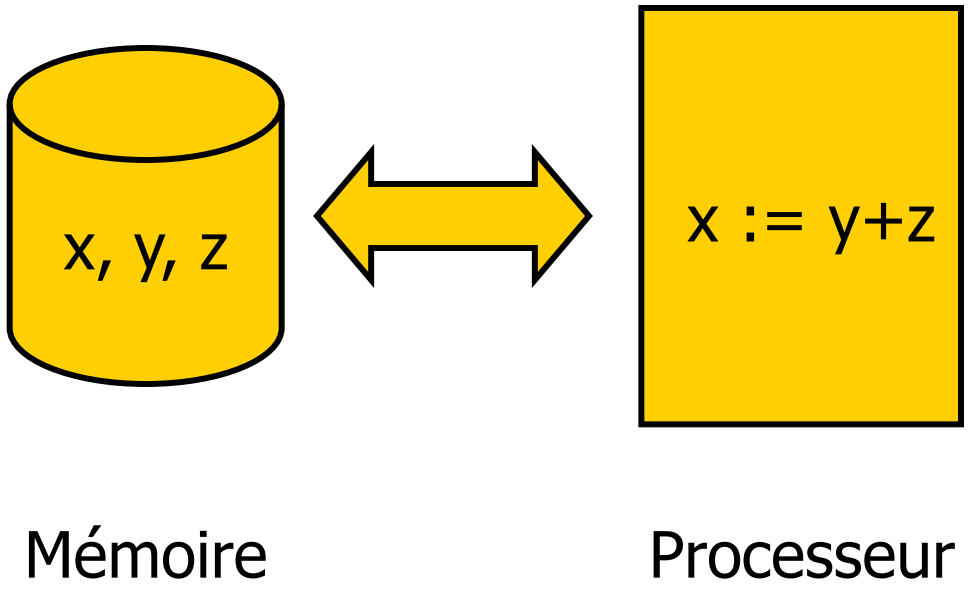


- Tout procédé (humainement) effectif de calcul peut être simulé par une machine de Turing
  - le procédé consiste en un ensemble fini d'instructions simples et précises qui sont décrites avec un nombre limité de symboles ;
  - le procédé doit toujours produire le résultat en un nombre fini d'étapes ;
  - le procédé peut en principe être suivi par un humain avec seulement du papier et un crayon ;
  - la mise en œuvre du procédé ne requiert pas d'intelligence de l'humain sauf celle qui est nécessaire pour comprendre et exécuter les instructions.

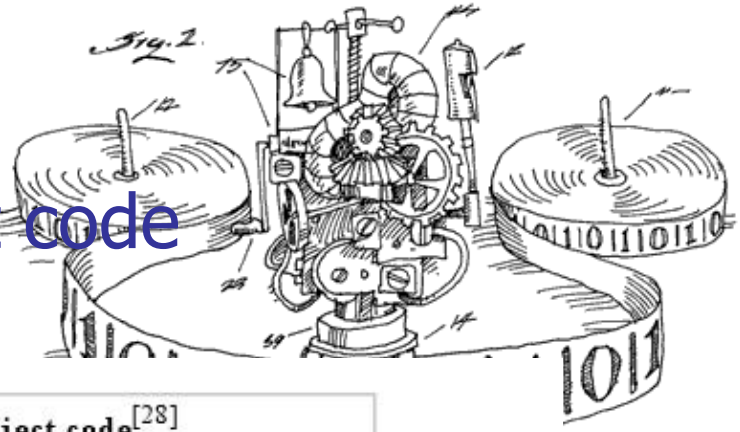
Modèle de von Neumann



- Notion de cycle de calcul: instruction
- Récupérer l'instruction à exécuter
- Récupérer les données dans la mémoire
- Effectuer l'opération demandée par le programme
- Ranger le résultat dans la mémoire



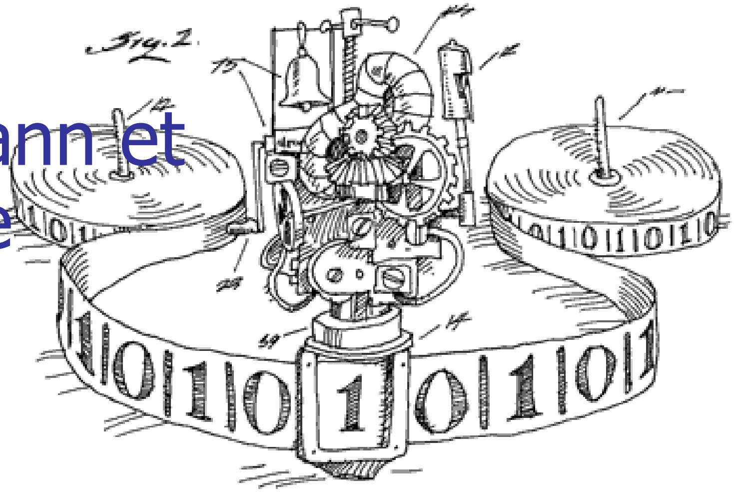
Modèle de Von Neumann et code assembleur



Address	Label	Instruction (AT&T syntax)	Object code[28]
		.begin	
		.org 2048	
	a_start	.equ 3000	
2048		ld length,%	
2064		be done	00000010 10000000 00000000 00000110
2068		addcc %r1,-4,%r1	10000010 10000000 01111111 11111100
2072		addcc %r1,%r2,%r4	10001000 10000000 01000000 00000010
2076		ld %r4,%r5	11001010 00000001 00000000 00000000
2080		ba loop	00010000 10111111 11111111 11111011
2084		addcc %r3,%r5,%r3	10000110 10000000 11000000 00000101
2088	done:	jmpl %r15+4,%r0	10000001 11000011 11100000 00000100
2092	length:	20	00000000 00000000 00000000 00010100
2096	address:	a_start	00000000 00000000 00001011 10111000
		.org a_start	
3000	a:		

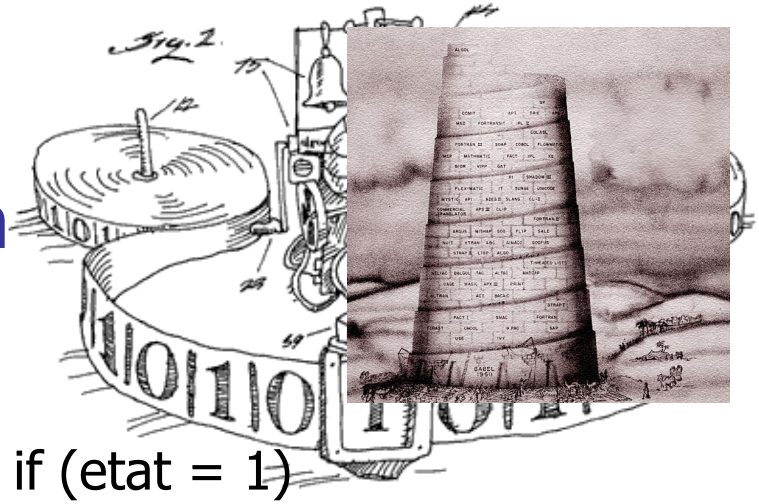
Example of a selection of instructions (for a virtual computer[29]) with the corresponding address in memory where each instruction will be placed. These addresses are not static, see memory management. Accompanying each instruction is the generated (by the assembler) object code that coincides with the virtual computer's architecture (or ISA).

Le modèle de Von Neumann et l'assembleur peuvent être programmés en TOY



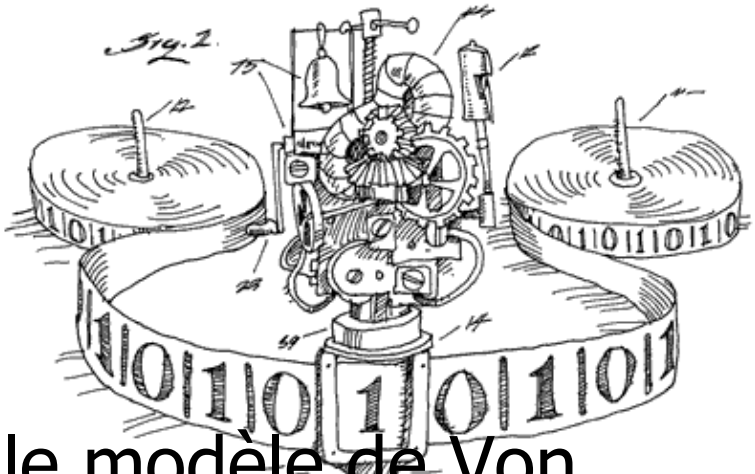
```
PC := 0
Memoire[] := "….."
while (PC != -1)
    Instruction := Memoire[PC]
    if (instruction = ADD)
        operandes := Mémoire[…]
        Mémoire[…] := sum(operandes)
        PC = PC + 1
    if (instruction = GOTO)
        operandes := Mémoire[…]
        PC = operande
    if (instruction = HALT)
        PC := -1
```

On peut simuler une machine de Turing par un programme en TOY



```
etat := 0          // borné
tmp := ' '     // borné
i := 0          // non borné
ruban = "....."    // non borné
while (etat != -1)
do
    if (etat= 0)
      tmp := ruban[i]
      if tmp = 'a'
            ruban[i] := 'b'
            i := i + 1
            PC := 1
    if (etat = 1)
      tmp := ruban[i]
      if tmp := 'a'
            ruban[i] := 'c'
            i := i – 1
            PC := 45

/**********************/

    if (etat = 9999)
      etat := -1
```

The Folk Theorem

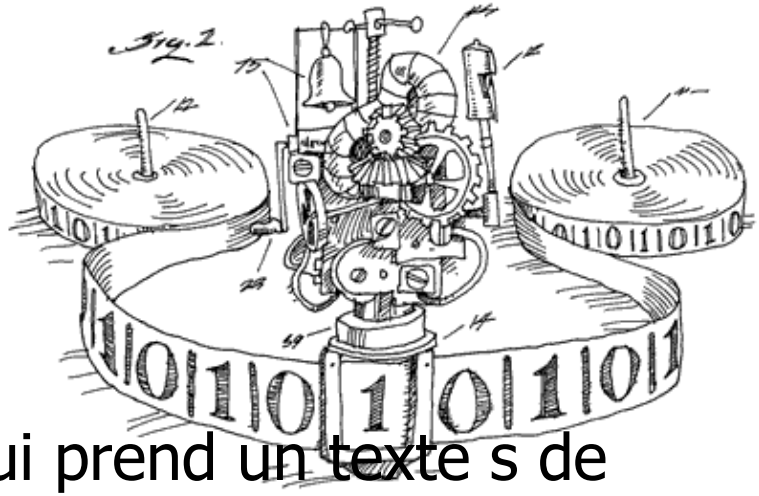


- La machine de Turing et le modèle de Von Neumann sont équivalents
- Ils sont tous les d u langage TOY
- Tout programme peut d'exprimer à l'aide d'une seule boucle while et de deux variables entières non bornées statiquement

Processeur physique

Index mémoire
Contenu mémoire

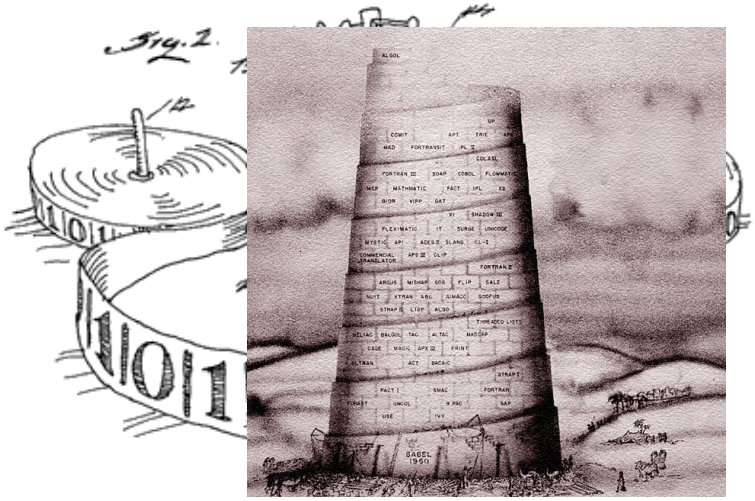
Le théorème de la halte en 4 lignes



- Soit P(s, x) un programme qui prend un texte s de programme à une entrée et une valeur x d'entrée.
- Supposons qu'il termine toujours et qu'il renvoie
  - oui si le programme S de texte s s'arrête sur l'entrée x
  - non si le programme S de texte s ne s'arrête pas sur l'entrée x
- Soit Q(s) le programme construit avec P tel que
  - Q ne s'arrête pas si le programme S de texte s s'arrête sur l'entrée s
  - Q s'arrête si le programme S de texte s en s'arrête pas sur l'entrée s
- Soit q le texte de Q. Est-ce que Q(q) s'arrête ou pas?

En guise de conclusion…

Conséquence: Babel revisitée



C++

Java

C

Javascript

Le langage encore inconnu...

Fortran

x!) du monde

Simula

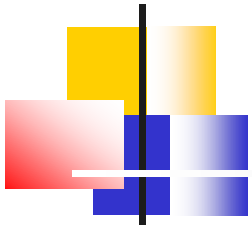
Caml

Assembleur

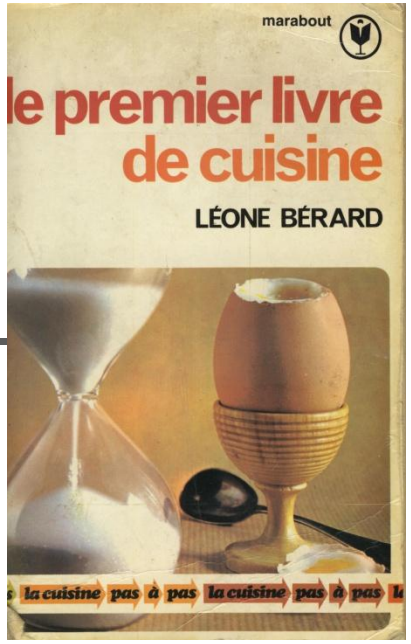
Lisp

Prolog

A vous de jouer!

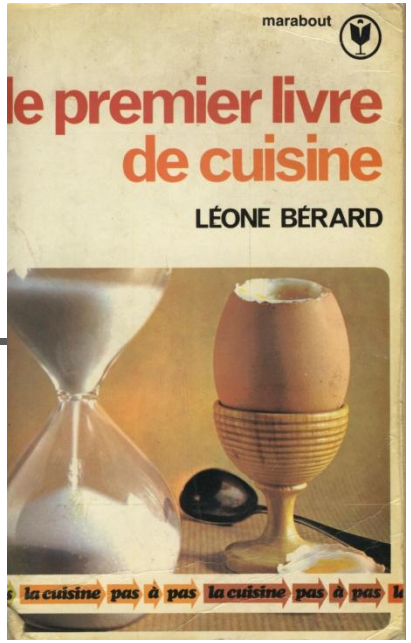


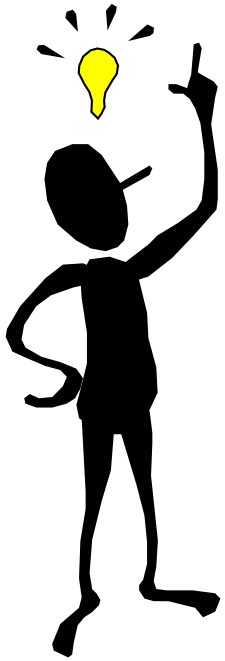
Nom du programme



- La pâte à crêpes

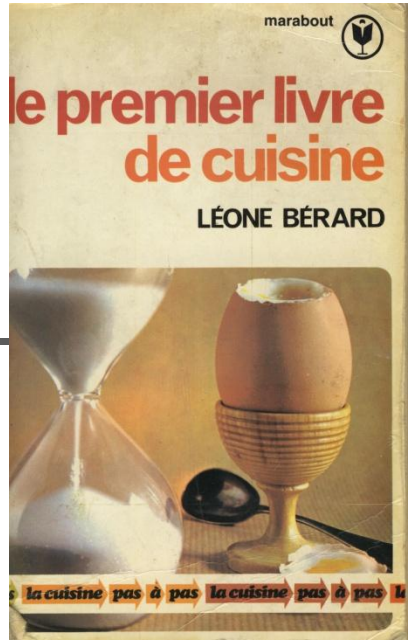
Commentaires et assertions de validité



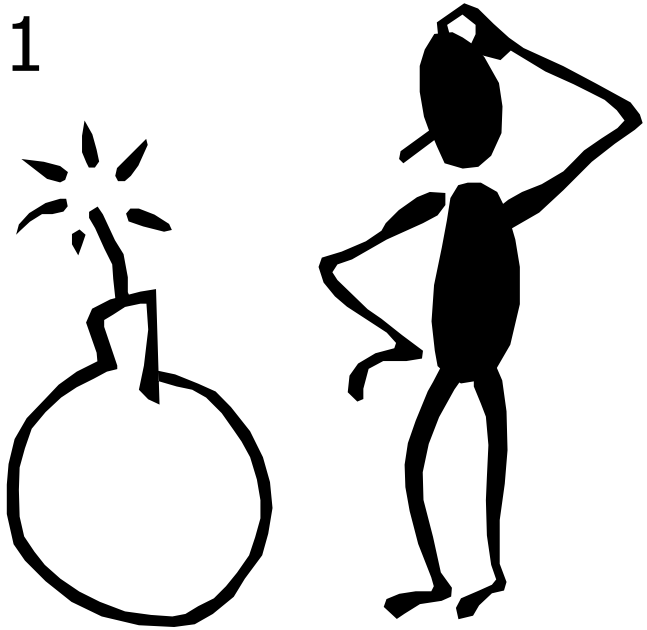


- Travail personnel: 15 minutes. A préparer au moins une heure à l'avance

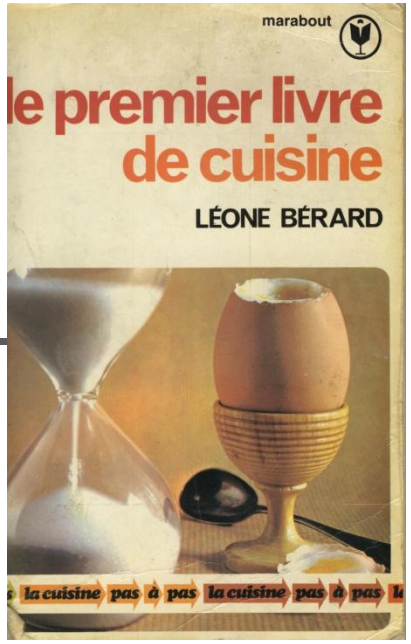
Dispositifs matériels périphériques



- Matériel: 1 terrine, 1 bol, 1 fouet, 1 torchon



Déclarations, définitions de constantes et initialisations



- Denrées pour 4 personnes (12 crêpes de taille moyenne)
  - Farine: 125 g
  - Œufs: 2
  - Lait: 1 verre
  - Eau: 1 verre
  - Huile: 1 cuiller à soupe
  - Sel
  - Parfum: Fleur d'oranger, rhum, vanille en poudre, zeste d'orange ou de citron, etc.

Dispositifs matériels

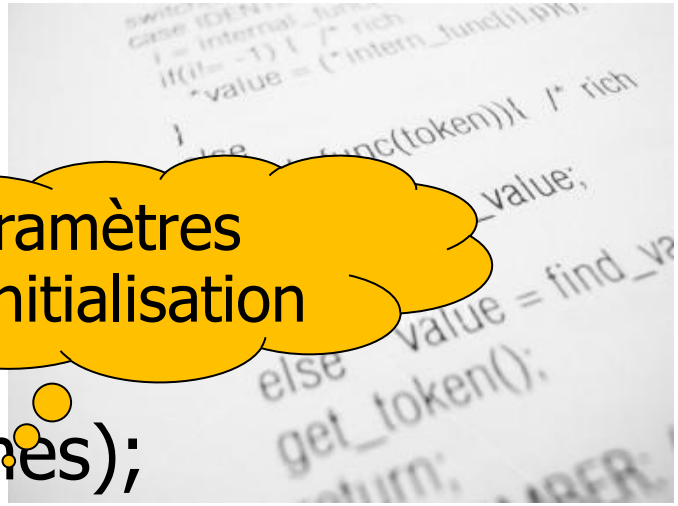
Port terrine;

Port bol;

Port fouet;

Port torchon

open(terrine, bol, fouet, torchon)



Type, classe

Paramètres d'initialisation

Ingredient farine(125 grammes);

Ingredient œuf(2 unites);

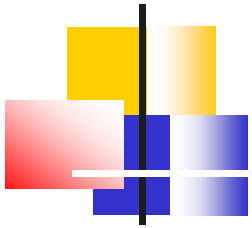
Ingredient sel(1 pincee);

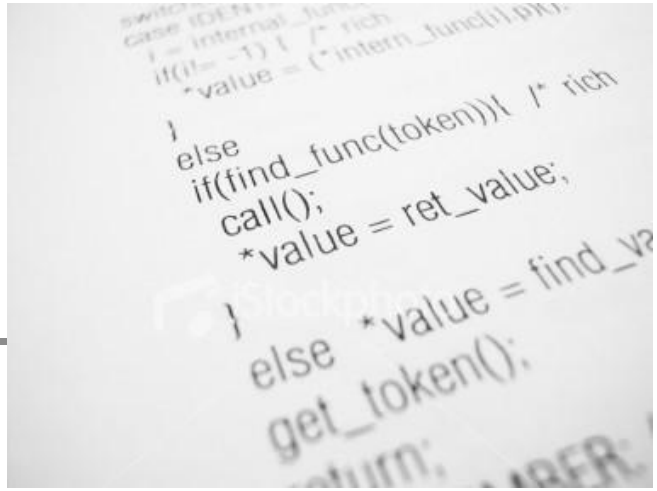
Ingredient lait(1 verre);

Ingredient eau(1 verre);

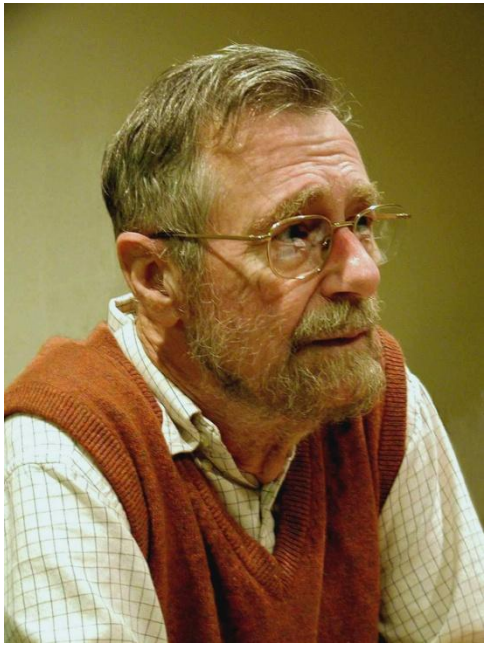
Ingredient huile(1 cuilleree);

Ingredient parfum(…)



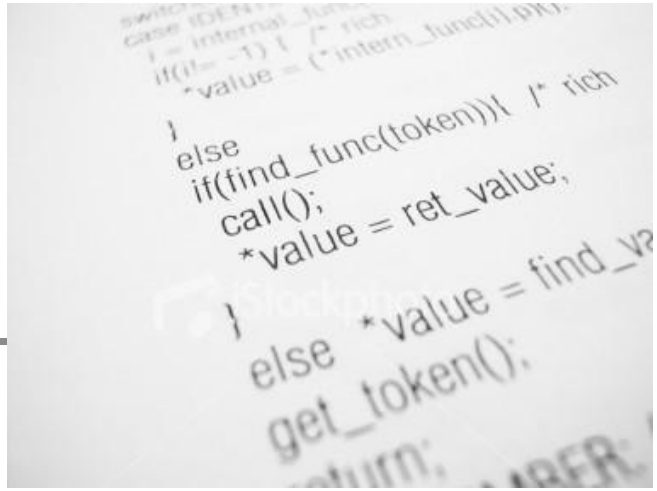


Quel langage pour quels programmes?



E.W.G Dijkstra

Styles de programmation



Pascal
C

- Impératif
  - Enchaînement d'instructions d'affectation de variables conduisant à un état final

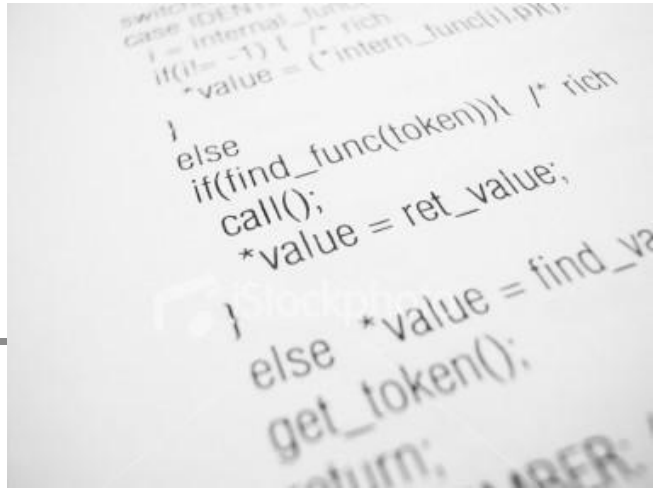
Lisp
Caml

- Fonctionnel
  - Imbrication de fonctions produisant un résultat final

Prolog

- Logique, contraintes
  - Suite de conditions logiques spécifiant les propriétés du résultat

➔ C'est au système de résoudre ce système d'équations



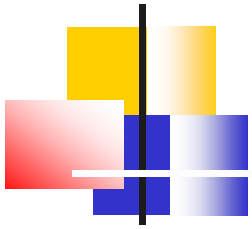
Contrôle vs données

- Centrage sur le contrôle
  - Types de données simples et constructeurs de types
  - Fonctions externes
  - Notion de modularité

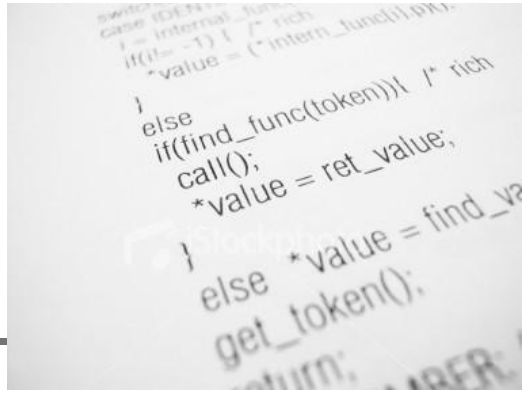
Pascal
C

- Centrage sur les données
  - Notion d'objet = données + procédures = attributs + méthodes
  - Notion de type ➔ Notion de classe
  - Opération de base: appel de méthode
  - Notion de visibilité

C++
Java



Pour cette semaine: Javascool



- Un environnement de programmation spécialisé pour les collèges et lycées
  - Base d'exemples et d'activité pédagogiques
- Java simplifié ➔ style Pascal/C
  - Possibilité de Java/Swing complet
- Très grande portabilité
  - Complètement écrit en Java
  - Distribué sous forme d'une archive JAR
  - Facilement extensible
- Produit INRIA Sophia, Thierry Viéville
  - Engagement de soutien de l'INRIA

JAVA Scool

. . . and java is so cool !

ACCUEIL LANCEMENT (AIDE) ACTIVITÉS RESSOURCES CONTACTS

Rechercher

Activités

- Démarrage

Ressources

- Aide au TPE
- interstices

Contacts

- F.A.Q.
- Bureau d'accueil

Développement

- Manifeste
- Wiki de travail
- Doc Java
- Licence

Une version améliorée de la 3.2 avec *«proglets»* et les *«educlets»* est disponible NEW! !

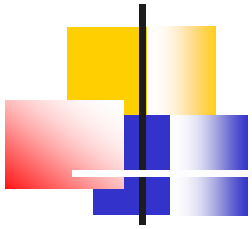
Java's Cool (alias Javascool) est un logiciel conçu pour l'apprentissage des bases de la programmation. Il a été conçu à la demande de professeurs de lycées et de fuscia avec UNICIEL. Il permet de manipuler un Macro-Langage de programmation, basé sur le langage Java. Pour plus de détails . . essayez le ! Créé pour apprendre aux personnes qui savent juste allumer leur ordinateur à programmer, par exemple des jeux (en 2D) ou des minis-logiciels, son interface est simple ainsi que son langage dit de "balisage". Ce programme est écrit en Java, il est donc compatible avec tous les systèmes.

Le poster de présentation

Qu'est ce que c'est ? Un recueil de ressources (documents et logiciels) pour aider les professeur(e)s des lycées et du début du supérieur au niveau de l'enseignement de l'algorithmique, donc de l'informatique en seconde. La démarche est expliquée dans le manifeste et se base sur un cadrage précis.

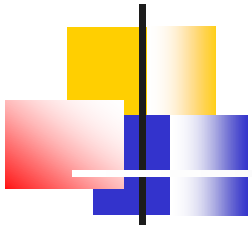
On y trouve des textes de base sur ce qu'est l'information, comment sont codés les objets numériques et sur les ingrédients des algorithmes. Un parcours de formation, pour s'approprier

Mon premier programme Java(scool)




```
void main() {
    println("Hello world!");
}
```

Hello world!



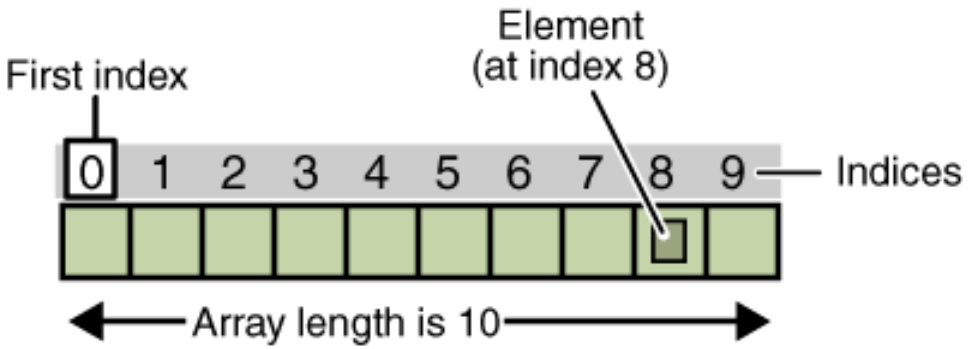
Les types simples

```
void main() {
    int i = 1;                          // 32 bits
    long j = 123456789012345678L;       // 64 bits
    double k = 3.14159;                 // 64 bits
    println(i+j+k);
    char c = 'a';
    println(c);                         // 16 bits Unicode
    boolean b = true;
    println(b);
    }
```

```
1.2345678901234567E18
97
true
```

Les types construits

```
void main () {
    final int N = 10;

    int tab[] = new int[N];

    for (int i = 0; i < N; i++)
        tab[i] = i*i;
    for (int i = 0; i < N; i++)
        println(tab[i]);
}
```




```
0
1
4
9
16
...
```

Les expressions

- Opérations arithmétiques classiques
  - +, -, *, /
  - Modulo: %
- Tests d'égalité
  - ==, !=
- Tests arithmétiques
  - >, >=, <, <=
- Concaténation de chaînes
  - +
- Opérateurs logiques
- &&, ||, !

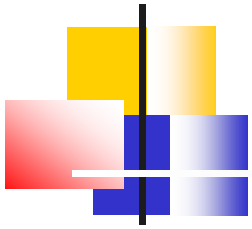
Attention à la division entière!

Attention à "=="!

Evaluation conditionnelle

Les instructions

- Affectation
  - x = 2;
- Choix
  - if (x == 2) y = 3;
  - if (x == 2) y = 3; else y = 4;
- Itération non bornée
  - while (x > 0) x = x – 1;
  - Sortie anticipée: break
- Groupage d'instructions
  - {P; Q; R …}



Affectations avec auto-incrémentation

- Affectation simple
  - x = 2
- Auto-incrémentation
  - x++, x--
- Affectation avec incrémentation
  - x += 2
  - x -= 3
  - x *= 2
- Equivalence
  - x++ ➔ {tmp = x; x = x+1; return tmp;}

Instruction for

- Itération bornée:

  for (init; test; incr) corps

  - init: déclaration et initialisation
  - test: test de continuation
  - incr: instructions d'incrémentation

- {init; while (test) {corps; incr;}}
  - Sortie anticipée: break

```
for (int i = 0; i < N; i++)
    tab[i] = i*i;
```

La bouche for-each

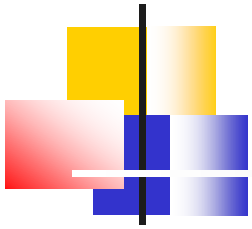
```
void main () {
    final int N = 5;
    int tab[] = new int[N];
    for (int i = 0; i < N; i++)
          tab[i] = i*i;

    for (int n: tab)
          println(n);
}
```

```
0
1
4
9
16
...
```

Enumérations

```
enum Chiffre {ZERO, UN, DEUX, TROIS};

void main () {
    Chiffre n;
    n = Chiffre.DEUX;
    println(n);

    for (Chiffre p: Chiffre.values())
        println(p);
}
```

```
DEUX

ZERO
UN
DEUX
TROIS
```



Chaînes

```
void main() {
    String s = "Bonjour";
    String t = "les gars!";
    println(s + " " + t);

    for (int i = 0; i < s . length(); i ++)
     println(s . charAt(i));

    for(char c: s.toCharArray())
     println(c);
}
```

```
Bonjour les gars!
66
111
110
106
111
117
114
...
```