Cours d'introduction à l'informatique

Partie 1 : Pourquoi apprendre l'informatique ? Quels outils pour ce cours ?

Pourquoi étudier l'informatique ?

Pour devenir informaticien(ne)

~forte demande (>40000 emplois/an d'informaticiens non pourvus)

Pour occuper tout emploi de niveau technicien supérieur ou ingénieur

« outil numérique » central dans les métiers d'aujourd'hui, nécessaire de le dompter pour rester « dans le coup »

En quoi un mathématicien/physicien/chimiste/et autre doit-il se former en informatique

Numérisation massive du monde qui nous entoure (y compris ce qu'on observe), besoin de manipuler des données pour comprendre le monde « moderne »

Informatique : une définition

> **L'informatique est le domaine d'activité scientifique, technique et industriel concernant le traitement automatique de l'information par des machines telles que : calculateur, système embarqué, ordinateur, console de jeux vidéo, robot, automate, etc.**
>
> **[Wikipedia]**

- «Information is Information, no Matter, nor Energy» Wiener 1968

- On travaille sur l'Information (range, ordonne, transforme, transmet,…)

> **« La science informatique n'est pas plus la science des ordinateurs que l'astronomie n'est celle des télescopes. » — Edsger Dijkstra**

Pourquoi de l'informatique ?

- Outil scientifique incontournable sur la scène interna-tionale

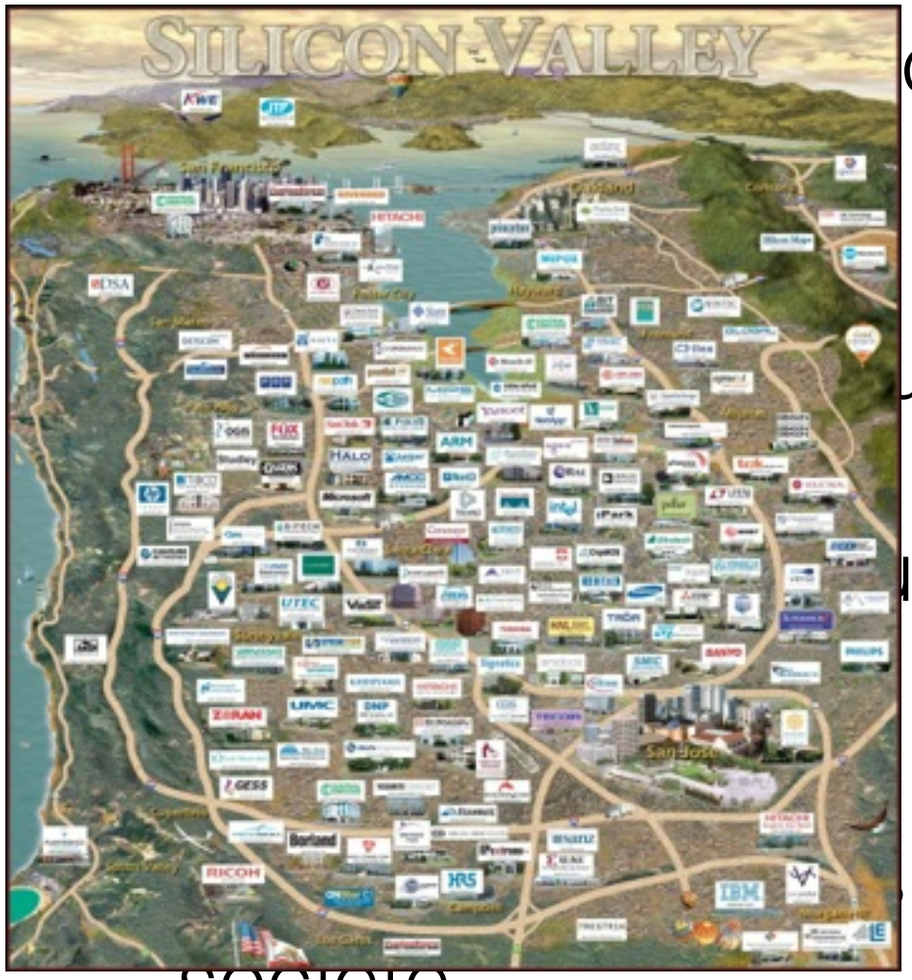
- Domaine scientifique omni-présent

- Science de communication, et source d'échanges scientifi-ques avec les autres disciplines

- Vecteur de progrès technologiques, science de société

- Une diversité thématique insoupçonnée

Pourquoi de l'informatique ?



contournable sur la scène interna-

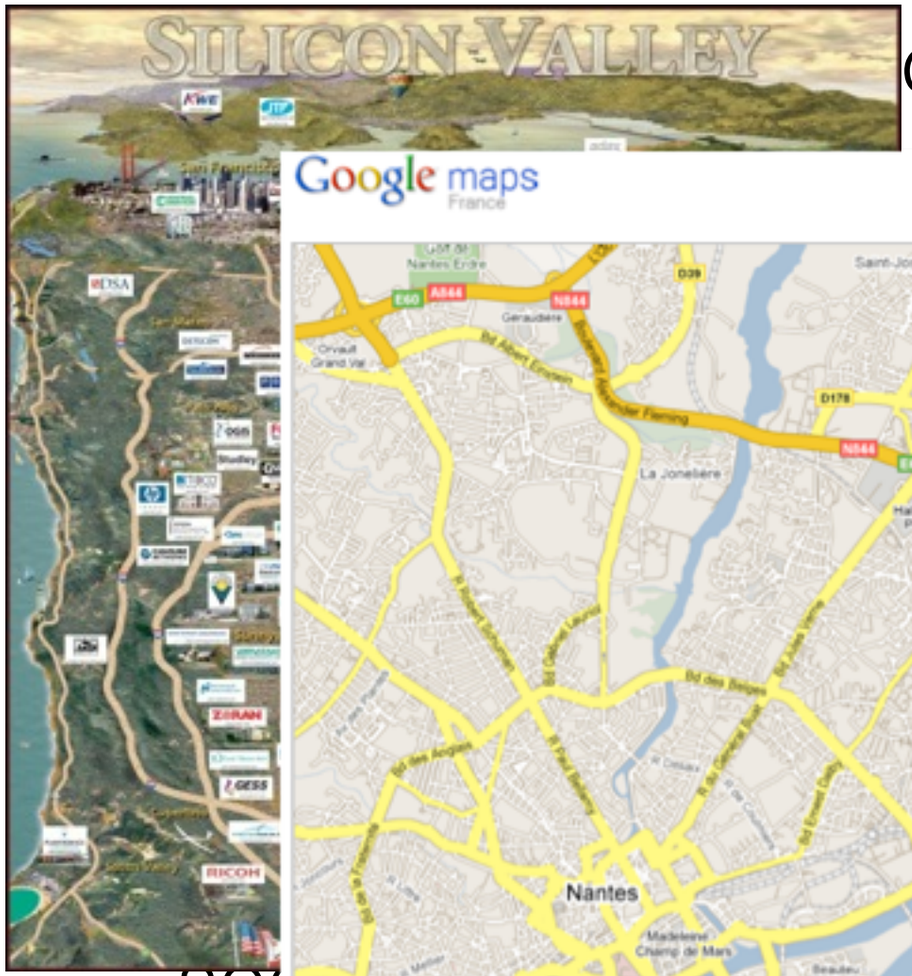
ue omni-présent

nication, et source d'échanges
les autres disciplines

technologiques, science de
société

- Une diversité thématique insoupçonnée

Pourquoi de l'informatique ?



contournable sur la scène interna-

ésent

t source d'échanges
disciplines

iques, science de

soc

- Un upçonnée

Pourquoi de l'informatique ?

ène interna-

echanges

iques, science de

soc

- Un upçonnée

Pourquoi de l'informatique ?

ène interna-

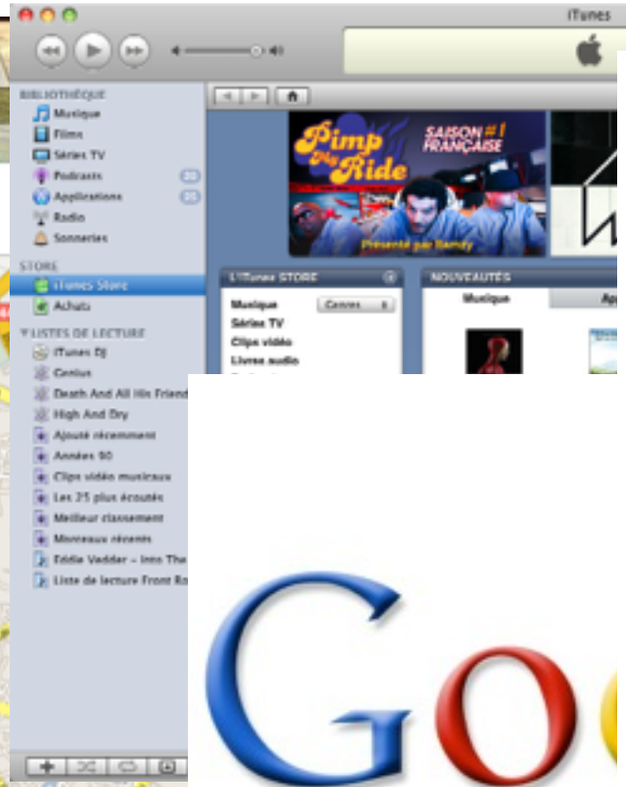
echanges

ence de

soc

- Un upçonnée

Pourquoi de l'informatique ?



erna-

es

soc

- Un upçonnée

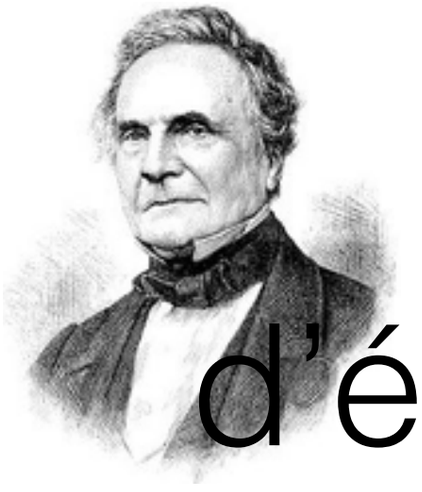
Aspects généraux

- En quoi un scientifique aujourd'hui doit-il maîtriser l'informatique ?

- De l'informatique pour scientifiques :
  - Traitement de mesures
  - Analyse de données
  - Simulations
  - Gestion de données

Début du cours d'algorithmique

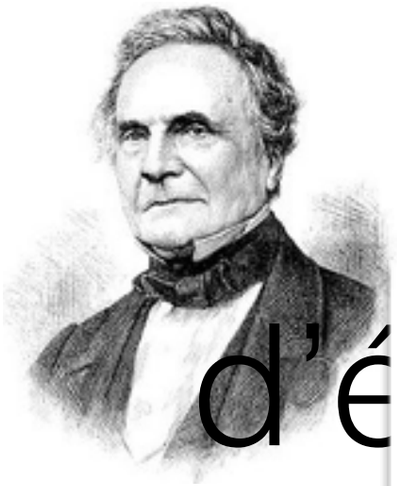
- Le but est d'apprendre à écrire des algorithmes (simples)
- un algorithme = l'écriture dans un langage non ambigu d'une méthode de résolution d'un problème.



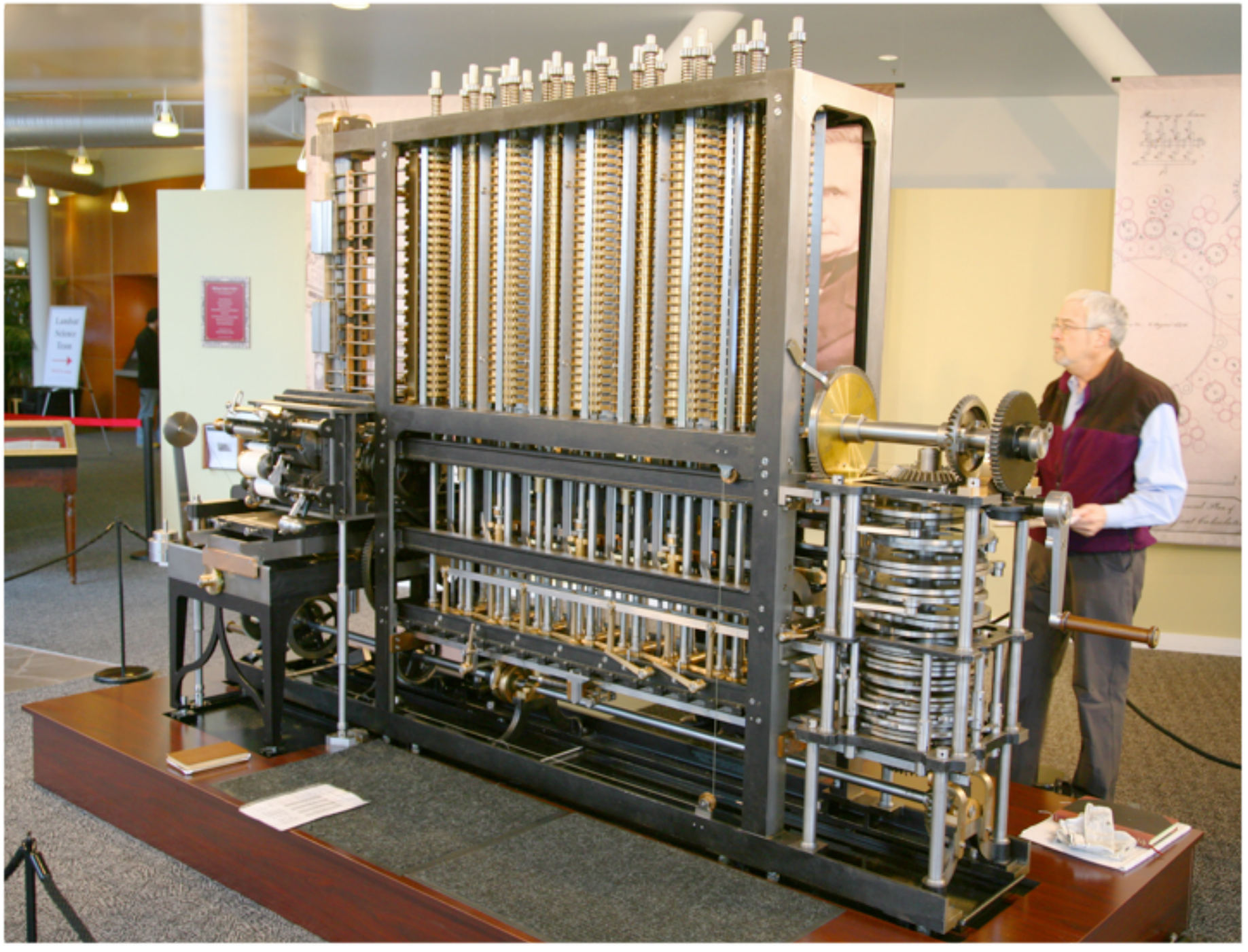
Pourquoi a-t-on besoin d'écrire de bons algorithmes

- Il y a très longtemps: Machine à calculer de Babbage (~1834)
- Instructions stockées sur des cartes perforées.
- Une instruction lue par tour de manivelle.
- Enjeu: diminuer le nombre de tours de manivelles (énergie) et le nombre de pièces nécessaires à construire la machine.

Notion de complexité en temps (nb de tours de manivelle=énergie) et en espace (nb de pièces).

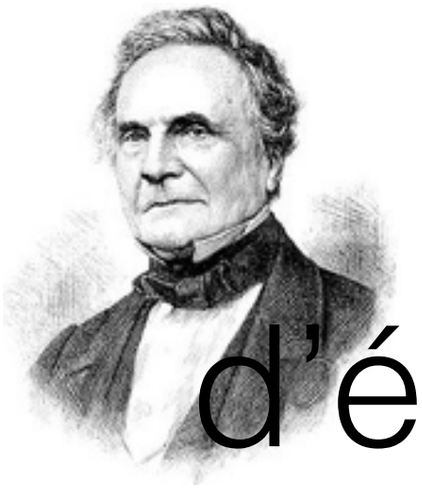


Pourquoi a-t-on besoin d'é…es

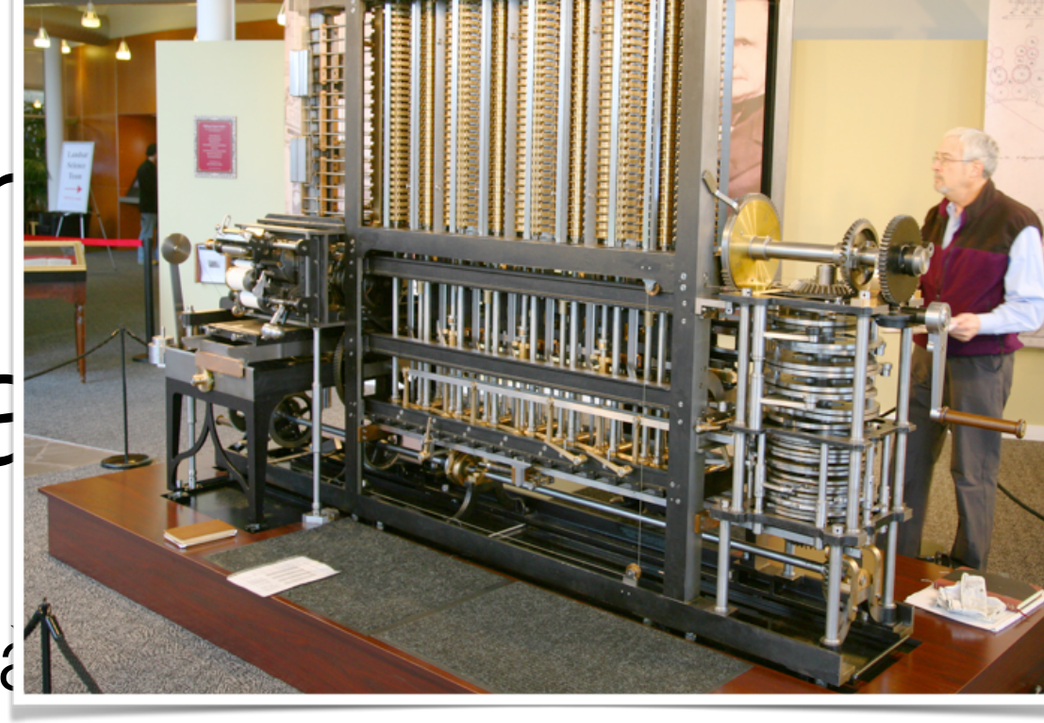


- Il y … Ba…
- Ins…
- Un…
- Enj… …s (ér… cor…

Notion de complexité en temps (nb de tours de manivelle=énergie) et en espace (nb de pièces).



Pourquoi a-t-on d'écrire de bons a



- Il y a très longtemps: Machine a Babbage (~1834)
- Instructions stockées sur des cartes perforées.
- Une instruction lue par tour de manivelle.
- Enjeu: diminuer le nombre de tours de manivelles (énergie) et le nombre de pièces nécessaires à construire la machine.

Notion de complexité en temps (nb de tours de manivelle=énergie) et en espace (nb de pièces).

Pourquoi a-t-on toujours besoin d'écrire de bons algorithmes

- Aujourd'hui encore plus primordial
- Réduire le nombre d'instructions = réduire la consommation = allonger la durée de vie de la batterie
- Retour à des machines avec moins de puissance de calcul
- Problèmes environnementaux -> greenIT

-on t

bons

plus pri

- Réduire le nombre d'instructions = réduire la consommation = allonger la durée de vie de la batterie

- Retour à des machines avec moins de puissance de calcul

- Problèmes environnementaux -> greenIT

Algorithmes

- L'algorithme permet **d'exécuter** et **d'étudier** cette méthode de résolution

- Un exemple « difficile »: calculer une puissance

$$x^{2^k} = \underbrace{x \times \cdots \times x}_{2^k \text{ fois}} = \underbrace{\left(\left(\left(x^2\right)^2\right)^2\right)^{2\cdots}}_{k \text{ fois}}$$

Algorithmes

$$x^{2^k} = \underbrace{x \times \cdots \times x}_{2^k \text{ fois}} = \underbrace{\left(\left(\left(x^2\right)^2\right)^2\right)^{2\cdots}}_{k \text{ fois}}$$

Version 1

puissance ← 1
répéter 2^k fois
    puissance ← *puissance* × *x*

Restituer le résultat

Version 2

puissance ← *x*
répéter *k* fois
    puissance ← *puissance*^{2}

Restituer le résultat

$$x^{2^k} = \underbrace{x \times \cdots \times x}_{2^k \text{ fois}} = \underbrace{\left(\left(\left(x^2\right)^2\right)^2\right)^{2\cdots}}_{k \text{ fois}}$$

Algorithmes

Version 1

puissance ← 1
répéter 2^k fois
    puissance ← *puissance* × *x*

Restituer le résultat

Version 2

puissance ← *x*
répéter *k* fois
    puissance ← *puissance*^{2}

Restituer le résultat

Un exemple d'exécution pour x=3 et k=4 pour calculer 3^{16}

$$x^{2^k} = \underbrace{x \times \cdots \times x}_{2^k \text{ fois}} = \underbrace{\Big(\big((x^2)^2\big)^2\Big)^{2\cdots}}_{k \text{ fois}}$$

Algorithmes

Version 1

puissance ← 1
répéter 2^k fois
    puissance ← *puissance* × *x*

Restituer le résultat

Version 2

puissance ← *x*
répéter *k* fois
    puissance ← *puissance*^{2}

Restituer le résultat

Un exemple d'exécution pour calculer 3^{16}

Version 1

puissance ← 1

puissance	←	*puissance*	×	3	☞	*puissance vaut 3*
puissance	←	*puissance*	×	3	☞	*puissance vaut 9*
puissance	←	*puissance*	×	3	☞	*puissance vaut 27*
puissance	←	*puissance*	×	3	☞	*puissance vaut 81*
puissance	←	*puissance*	×	3	☞	*puissance vaut 243*
puissance	←	*puissance*	×	3	☞	*puissance vaut 729*
puissance	←	*puissance*	×	3	☞	*puissance vaut 2187*
puissance	←	*puissance*	×	3	☞	*puissance vaut 6561*
puissance	←	*puissance*	×	3	☞	*puissance vaut 19683*
puissance	←	*puissance*	×	3	☞	*puissance vaut 59049*
puissance	←	*puissance*	×	3	☞	*puissance vaut 177147*

$$x^{2^k} = \underbrace{x \times \cdots \times x}_{2^k \text{ fois}} = \underbrace{\left(\left(\left(x^2\right)^2\right)^2\right)^{2\cdots}}_{k \text{ fois}}$$

Algorithmes

Version 1

puissance ← 1
répéter 2^k fois
    puissance ← *puissance* × *x*

Restituer le résultat

Version 2

puissance ← *x*
répéter *k* fois
    puissance ← *puissance*^{2}

Restituer le résultat

Version 1

puissance ← 1
    puissance ← *puissance* × 3 ☞ *puissance vaut 3*
    puissance ← *puissance* × 3 ☞ *puissance vaut 9*
    puissance ← *puissance* × 3 ☞ *puissance vaut 27*
    puissance ← *puissance* × 3 ☞ *puissance vaut 81*
    puissance ← *puissance* × 3 ☞ *puissance vaut 243*
    puissance ← *puissance* × 3 ☞ *puissance vaut 729*
    puissance ← *puissance* × 3 ☞ *puissance vaut 2187*
    puissance ← *puissance* × 3 ☞ *puissance vaut 6561*
    puissance ← *puissance* × 3 ☞ *puissance vaut 19683*
    puissance ← *puissance* × 3 ☞ *puissance vaut 59049*
    puissance ← *puissance* × 3 ☞ *puissance vaut 177147*
    puissance ← *puissance* × 3 ☞ *puissance vaut 531441*
    puissance ← *puissance* × 3 ☞ *puissance vaut 1594323*
    puissance ← *puissance* × 3 ☞ *puissance vaut 4782969*
    puissance ← *puissance* × 3 ☞ *puissance vaut 14348907*
    puissance ← *puissance* × 3 ☞ *puissance vaut 43046721*

Restituer le résultat = 43046721

…our x=3 et k=4 pour calculer 3^{16}

$$x^{2^k} = \underbrace{x \times \cdots \times x}_{2^k \text{ fois}} = \underbrace{\left(\left(\left(x^2\right)^2\right)^2\right)^{2\cdots}}_{k \text{ fois}}$$

Algorithmes

Version 1

puissance ← 1
répéter 2^k fois
    puissance ← *puissance* × *x*

Restituer le résultat

Version 2

puissance ← *x*
répéter *k* fois
    puissance ← *puissance*^{2}

Restituer le résultat

Version 1

puissance ← 1
    puissance ← *puissance* × 3 ☞ *puissance vaut 3*
    puissance ← *puissance* × 3 ☞ *puissance vaut 9*
    puissance ← *puissance* × 3 ☞ *puissance vaut 27*
    puissance ← *puissance* × 3 ☞ *puissance vaut 81*
    puissance ← *puissance* × 3 ☞ *puissance vaut 243*
    puissance ← *puissance* × 3 ☞ *puissance vaut 729*
    puissance ← *puissance* × 3 ☞ *puissance vaut 218*
    puissance ← *puissance* × 3 ☞ *puissance vaut 65(*
    puissance ← *puissance* × 3 ☞ *puissance vaut 19(*
    puissance ← *puissance* × 3 ☞ *puissance vaut 59(*
    puissance ← *puissance* × 3 ☞ *puissance vaut 17?*
    puissance ← *puissance* × 3 ☞ *puissance vaut 53(*
    puissance ← *puissance* × 3 ☞ *puissance vaut 159*
    puissance ← *puissance* × 3 ☞ *puissance vaut 4782969*
    puissance ← *puissance* × 3 ☞ *puissance vaut 14348907*
    puissance ← *puissance* × 3 ☞ *puissance vaut 43046721*

Restituer le résultat = 43046721

…our x=3 et k=4 pour calculer 3^{16}

Version 2

puissance ← 3
    puissance ← *puissance*^{2} ☞ *puissance vaut 9*
    puissance ← *puissance*^{2} ☞ *puissance vaut 81*
    puissance ← *puissance*^{2} ☞ *puissance vaut 6561*
    puissance ← *puissance*^{2} ☞ *puissance vaut 43046721*

Restituer le résultat = 43046721

$$x^{2^k} = \underbrace{x \times \cdots \times x}_{2^k \text{ fois}} = \underbrace{\left(\left(\left(x^2\right)^2\right)^2\right)^{2\cdots}}_{k \text{ fois}}$$

Algorithmes

Version 1

puissance ← 1
répéter 2^k fois
    puissance ← *puissance* × *x*

Restituer le résultat

Version 2

puissance ← *x*
répéter *k* fois
    puissance ← *puissance*^{2}

Restituer le résultat

Version 1

puissance ← 1
    puissance ← *puissance* × 3 ☞ *puissance vaut 3*
    puissance ← *puissance* × 3 ☞ *puissance vaut 9*
    puissance ← *puissance* × 3 ☞ *puissance vaut 27*
    puissance ← *puissance* × 3 ☞ *puissance vaut 81*
    puissance ← *puissance* × 3 ☞ *puissance vaut 243*
    puissance ← *puissance* × 3 ☞ *puissance vaut 729*
    puissance ← *puissance* × 3 ☞ *puissance vaut 2187*
    puissance ← *puissance* × 3 ☞ *puissance vaut 6561*
    puissance ← *puissance* × 3 ☞ *puissance vaut 19683*
    puissance ← *puissance* × 3 ☞ *puissance vaut 59049*
    puissance ← *puissance* × 3 ☞ *puissance vaut 177147*
    puissance ← *puissance* × 3 ☞ *puissance vaut 531441*
    puissance ← *puissance* × 3 ☞ *puissance vaut 1594323*
    puissance ← *puissance* × 3 ☞ *puissance vaut 4782969*
    puissance ← *puissance* × 3 ☞ *puissance vaut 14348907*
    puissance ← *puissance* × 3 ☞ *puissance vaut 43046721*

Restituer le résultat = 43046721

our x=

Version 2

puissance ← 3
    puissance ← *puissance*^{2} ☞ *puissance vaut 9*
    puissance ← *puissance*^{2} ☞ *puissance vaut 81*
    puissance ← *puissance*^{2} ☞ *puissance vaut 6561*
    puissance ← *puissance*^{2} ☞ *puissance vaut 43046721*

Restituer le résultat = 43046721

$$x^{2^k} = \underbrace{x \times \cdots \times x}_{2^k \text{ fois}} = \underbrace{\left(\left(\left(x^2\right)^2\right)^2\right)^{2\cdots}}_{k \text{ fois}}$$

Algorithmes

Version 1

puissance ← 1
répéter 2^k fois
puissance ← *puissance* × 3 ☞ *puissance vaut 3*
puissance ← *puissance* × 3 ☞ *puissance vaut 9*
puissance ← *puissance* × 3 ☞ *puissance vaut 27*
puissance ← *puissance* × 3 ☞ *puissance vaut 81*
puissance ← *puissance* × 3 ☞ *puissance vaut 243*
puissance ← *puissance* × 3 ☞ *puissance vaut 729*
puissance ← *puissance* × 3 ☞ *puissance vaut 2187*
puissance ← *puissance* × 3 ☞ *puissance vaut 6561*
puissance ← *puissance* × 3 ☞ *puissance vaut 19683*
puissance ← *puissance* × 3 ☞ *puissance vaut 59049*
puissance ← *puissance* × 3 ☞ *puissance vaut 177147*
puissance ← *puissance* × 3 ☞ *puissance vaut 531441*
puissance ← *puissance* × 3 ☞ *puissance vaut 1594323*
puissance ← *puissance* × 3 ☞ *puissance vaut 4782969*
puissance ← *puissance* × 3 ☞ *puissance vaut 14348907*
puissance ← *puissance* × 3 ☞ *puissance vaut 43046721*

Restituer le résultat = 43046721

Version 2

puissance ← *x*
répéter *k* fois
puiss
puissance ← *puissance* × *puissance* ☞ *puissance vaut 6561*

Etudier l'algorithme, c'est dire par exemple que:

1) les deux versions ont besoin d'une seule « mémoire » (pour stocker la valeur de puissance)
2) La première version a besoin d'effectuer 2^k multiplications pour arriver au résultat
3) La deuxième version n'a besoin que de k multiplications pour arriver au même résultat

…et conclure que la deuxième version est meilleure que la première…

$$x^{2^k} = \underbrace{x \times \cdots \times x}_{2^k \text{ fois}} = \underbrace{\left(\left(\left(x^2\right)^2\right)^2\right)^{2\cdots}}_{k \text{ fois}}$$

Algorithmes

Version 1

puissance ← 1

Etud

1) les

«

2) La

m

3) La

multiplications pour arriver au même résultat

…et conclure que la deuxième version est meilleure que la première…

puissance ← *puissance* × 3 ☞ *puissance vaut 4782969*
puissance ← *puissance* × 3 ☞ *puissance vaut 14348907*
puissance ← *puissance* × 3 ☞ *puissance vaut 43046721*

```
Restituer le résultat = 43046721
```

Version 2

puissance ← *x*

e)

puiss

6721

L'algorithme 2 se généralise pour n'importe quelle puissance x^n.
Il porte le nom d'exponentiation rapide.

C'est certainement un des algorithmes exécuté le plus souvent (à la base de beaucoup de systèmes de protection de données)

Ce qu'on va apprendre dans ce cours sur les algorithmes

- Un langage « simple », commun qui permet de décrire sans ambiguité les opérations « mathématiques/ informatiques » qui permettent d'arriver à résoudre un problème

- Langage « algorithmique », codifié par un **vocabulaire** et une **syntaxe** qu'il convient de respecter pour partager un algorithme.

- Utilisation de ce qui est appris pour résoudre des problèmes dans de nombreuses disciplines de l'informatique

Organisation du cours

Algorithmique
Introduction de quelques bases pour travailler
(variables, types, instructions, structures de contrôle, fonctions et tableaux)

Réseaux sociaux
Facebook et cie: quelle théorie derrière ces outils ?

Bioinformatique
L'algorithmique pour traiter des données de la biologie, un exemple de pluridisciplinarité.

Traitement automatique des langues
Les particularités des données textuelles: traitements algorithmiques

Cryptographie et TI
Comment s'échanger de l'information efficacement et de manière sûre ?

Musique et son
Quels algorithmes pour générer automatiquement de la musique et du son par ordinateur ?

Organisation du cours

Algorithmique
Introduction de quelques bas
(variables, types, instructions, struct ions et tableaux)

Réseaux sociaux
Facebook et ci
d

Bioinformatique
L'algorithmique pour traiter des données de la biologie, un exemple de pluridisciplinarité.

Traitement automatique des langues
Les particularités des données textuelles: traitements algorithmiques

Cryptographie et TI
Comment s'échanger de l'information efficacement et de manière sûre ?

Musique et son
Quels algorithmes pour générer automatiquement de la musique et du son par ordinateur ?

Evaluation des connaissances pour les cours
(Questions de cours)

Algorithmique vs programmation

- L'algorithmique: écriture de méthodes de résolution de problèmes dans une forme compréhensible par un humain.

- La programmation: écriture de méthodes de résolution de problèmes dans une forme interprétable par une machine.

- Il faut aussi un langage codifié pour les programmes….

- Il en existe beaucoup (plus de 1000 dont plus de100 qui sont utilisés couramment)

Algorithmique vs programmation

- L'algorithmique: écriture de méth résolution de problèmes dans une for sible par un humain.

- La progr on: écriture de méthodes de résolution de problèmes dans une forme interprétable par une machine.

- Il faut aussi un langage codifié pour les programmes….

- Il en existe beaucoup (plus de 1000 dont plus de100 qui sont utilisés couramment)

Résolution d'exercices en TD
(au moins deux CC)

Algorithmique vs programmation

- L'algorithmique: écriture de méth… résolution de problèmes dans une for… …sible par un humain.

- La progr… …on: écriture de méthodes de résolution de problèmes dans une forme interprétable par une machine.

- Il faut aussi un langage co… …rogrammes….

- Il en exi… …s de 1000 dont plus de100 qui sont … …couramment)

Résolution d'exercices en TD
(au moins deux CC)

Codage d'algorithmes en TP
(au moins deux TP notés + projet)

Javascript

- Javascript est langage de programmation interprété et défini par un standard. Il est supporté par tous les navigateurs (mise à disposition d'un environnement de programmation qui tourne même sur smartphone et ipad !)

- Le web se développe: «Lars Bak, responsable du moteur Javascript de Google Chrome. *Pour une entreprise qui se lance aujourd'hui, utiliser ce langage est une évidence: il suffit de créer une application une seule fois, et elle fonctionnera partout !*"»

- Javascript est un langage «simple mais puissant» (notamment avec typage dynamique) mais sa syntaxe ressemble beaucoup à celle de la famille C/C++/java (partage d'éléments de syntaxe, bonne base pour la suite).

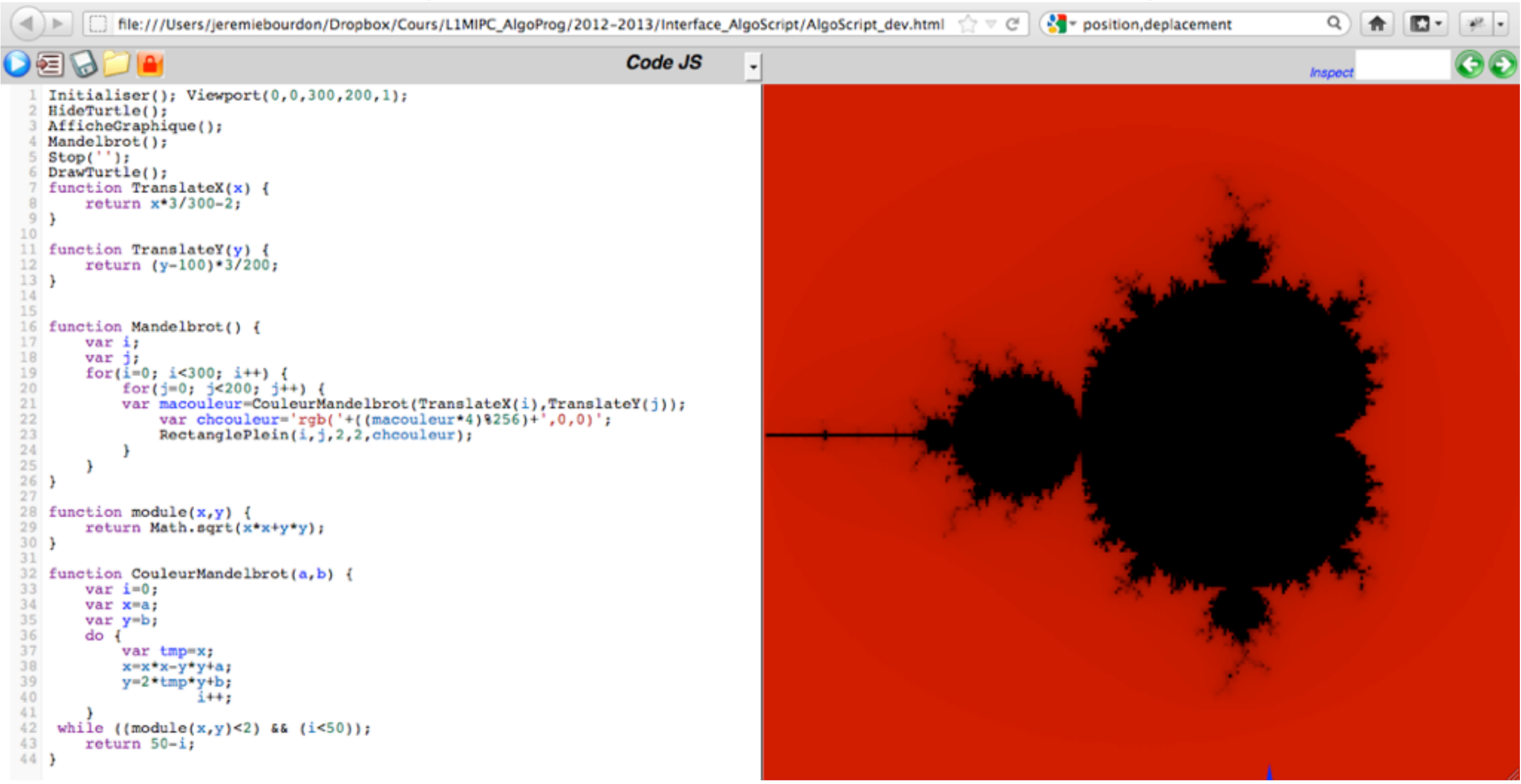
AlgoScript pour palier aux manques de javascript

- AlgoScript = une grosse bibliothèque de fonctions pour gérer les entrées/sorties dans un environnement de programmation intégré

  - texte: Ecrire(ch), Saisie(), getEntreetexte(),...
  - graphique: Point(x,y,couleur), Rectangle(...),...
  - sonore: ChargerSon(url), CreerSon(...),...
  - fichiers: writeFile(..), includeFile(...),...

- Mais la bibliothèque est écrite en javascript « pur »… Le programme final est donc du javascript « pur »

Un environnement de programmation intégré



Démonstrations

- Exemples de code «en direct» (afficher en même temps la sortie texte)

```
Ecrire('Bonjour le monde');
```

```
var message=Saisie();
Ecrire(message);
```

```
var x=Saisie();
Ecrire(5*x+12);
```

- Avec sortie graphique (penser à initialiser l'écran si besoin):

```
Rectangle(10,10,100,200,'red');
RectanglePlein(150,50,200,200,'blue');
Ligne(0,0,500,300,'green');
```

- L-Systems : exemple1.js et exemple2.js
- Ajouter un piano, une guitare, l'effet plasma
- Jouer avec les différentes fenêtres
- Et pourquoi pas, jouer avec la webcam, openstreetmap et autre…

Démonstrations

- Exemples de code «en direct» (afficher en même temps la sortie texte)

```
Ecrire('Bonjour le monde');
```

```
var message=Saisie();
Ecrire(message);
```

```
var x=Saisie();
Ecrire(5*x+12);
```

- Avec sortie graphique (penser à initialiser l'écran si besoin):

```
Rectangle(10,10,100,200,'red');
RectanglePlein(150,50,200,200,'blue');
Ligne(0,0,500,300,'green');
```

- L-Systems : exemple1.js et exemple2.js
- Ajouter un piano, une guitare, l'effet plasma
- **https://github.com/jeremiebourdon/AlgoScript**
- Et pourquoi pas, jouer avec la webcam, openstreetmap et autre…

Résumé

- Algorithmique et programmation: deux langages à maîtriser

- Evaluations des cours, TD et TP (2 CC pour les cours et TD, 2 CC pour les TP et 1 projet).

- Vidéos des cours mise en ligne avec possibilité d'annoter **pour tout le monde** (point à éclaircir, remarque, liens supplémentaires,…).

- Ressources pédagogiques sur Madoc.