Comment apprendre ce monde qui devient numérique ?

*présentation Thierry Viéville et al**

U. Été de St Flour. Août 2008

(*) *Grâce aux documents et dialogues de **Gilles Dowek** sur le sujet, aux éléments fournis par **Gérard Berry** et aux échanges et corrections de **Antoine Petit**.*

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

La question:

"Quelle part de la science informatique pourrait être incluse dans les programmes de mathématiques au niveau du lycée?"

quelques éléments de réflexion ...

Plan de l'exposé

- Introduction:
  Informatique et programmes scolaires

- L'informatique en 4 étapes et 10 leçons
  Des données structurées
  aux structures algorithmiques

Il est indispensable d'apprendre l' « algorithmique » (et les fondements théoriques qui y conduisent), c'est le levier pour pouvoir comprendre et maîtriser, voir adapter les logiciels et pas uniquement les subir.

- L'informatique en perspective

Informatique et programmes scolaires

L'informatique impacte plusieurs matières

* **Mathématiques:**
  Méthodes numériques et Symboliques. Algorithmique.
* **Physique-Chimie**
  Mesure et traitement des données. Modèles et simulation.
* **Philosophie**
  Logique et épistémologie. Informatique et mécanismes mentaux.
* **Histoire et Géographie ; Sciences économiques**
  Impact socio-économiques de l'informatique
  (Ex: mondialisation, Nouveaux modèles économiques, Green IT)
* **Technologie**
  Architectures des machines. Automatismes et mécanismes robotique.
* **Autres Sciences, Humanités et Matières Artistiques**
  Utilisation omniprésente de l'informatique (méthodes formelles, outils logiciels).

+++ The obvious fact English is widely used at very concrete level

L'informatique au delà des mathématiques

L'exemple de la philosophie:

Le sujet:

- Ma pensée est elle purement mécaniste ?

La culture

- La révolution numérique est elle du même ordre que celle de l'imprimerie ?

La raison et le réel

- La simulation numérique est elle une preuve expérimentale ?
- Y a t'il des barrières au savoir scientifique (ex: incomplétude) ?

La politique

- La mondialisation pouvait-elle se faire sans l'informatique ?

La morale

- Pirater une production intellectuelle constitue t'elle un vol ?

L'informatique . . avec les mathématiques ?

L'informatique est une matière à part entière . .

. . qui peut être ``céant´´ associée à l'enseignement des maths.

À condition:

- Que les profs soient bien formés en science informatique.

Ex: Une vraie épreuve d'informatique théorique au CAPES

- Que le volume horaire en maths soit significativement augmenté.

- Que les maths elle-mêmes de deviennent pas des maths-info

Bien distinguer l'info pour bien préserver les maths

Militer *ensemble* pour *augmenter* les contenus *fondamentaux*

Apprendre l'informatique en quatre étapes et dix leçons

L'informatique en quatre étapes

* [primaire-collège] **De l'usage des outils:**
  apprendre à utiliser les outils: édition, communication, etc...
  et apprendre à utiliser . . un clavier !
  profiter des ressources de l'internet, etc.

* [collège-lycée] **De l'apprentissage des méthodes:**
  apprentissage des algorithmes et de la programmation

* [lycée-etc..] **De la formation scientifique:**
  théorie de l'information, de la compilation,
  traitement des données, etc.

sans oublier

* [tous niveaux] **De la réflexion et de la culture** :
  d'où vient l'informatique, que change t'elle dans la société, etc....

Apprendre l'informatique en ``dix´´ leçons:

(quelques pistes de réflexions au niveau lycée)

L'informatique en dix leçons

et deux volets.

1/ Les données structurées: la chaîne de caractères.

Savoirs:

- encodage (caratères, chaîne), internationalisation;
- opérations sur les chaînes:

concaténer, comparer, transformer, trouver un motif.

Usages:

Monoïde libre, etc..

- charger / sauver une chaîne,
- détecter un caractère, normaliser les majuscules, ..

comprendre le codage d'un symbole et la notion élémentaire de séquence;
découvrir que l'on peut ``computer´´ sur des objets non-numériques;

L'informatique en dix leçons

et deux volets.

2/ Les données structurées: la valeur numérique.

Savoirs:

- nombres exacts et approchés
  valeurs minimales, maximales, par défaut, précision machine/applicative;
- opérations et calculs sur les nombres:
  expressions algébriques et logiques.

Usages:

- coder vrai/faux avec 0/1 et faire des ``calculs logiques´´
- découvrir les leviers qu'offrent les tirages aléatoires

comprendre les limites du calcul flottant et son coût en temps
distinguer ce qui est calcul symbolique et numérique

L'informatique en dix leçons

et deux volets.

3/ Les données structurées: la structure logique.

Savoirs:

- structure paramétrée et tables de valeurs
  spécification {paramètre :: type = valeur, ..}, validation de données
- notion de données typées
  vérification lexicale, tests de valeurs, propriétés structurelles

Usages:

- coder un carnet d'adresses ou un catalogue de ressources
- comprendre ce qu'est un identifiant universel de ressource

comprendre la notion élémentaire de typage
s'approprier l'idée de spécifier des données structurées

L'informatique en dix leçons

et deux volets.

4/ Les données structurées: un panorama général.

Savoirs:

- codage d'un vecteur et d'une matrice
  paramètres de la structure, indexation; exemple du son et de l'image
- codage d'un hyper-texte structuré
  notion de lien, de structures emboitées; exemple du dessin vectoriel

Usages:

- manipuler une image pixelique et un dessin vectoriel
- se familiariser avec un langage HTML ``jouet´´

bien comprendre la séparation forme – contenu d'un document
découvrir la notion de formats et de standards de données

L'informatique en dix leçons

et deux volets.

5/ Les structures algorithmiques: l'instruction conditionnelle

Savoirs:

- variables et séquences d'instruction
  variable et affectation, suite d'instructions
- la construction if(condition, valeur1, valeur0)
  mécanisme procédural, vue fonctionnelle

Usages:

- écrire les fonctions booléennes avec des if()
- découvrir les règles de simplification des if()

$x or $y → if($x, 1, $z)

if(1, $v, $f) → $v
if($c, $v, $v) → $f

x = 0 or z / x > 0

comprendre la différence entre une expression et une instruction
découvrir le code informatique comme un objet ``calculable´´

L'informatique en dix leçons

et deux volets.

6/ Les structures algorithmiques: l'encapsulation en routine

Savoirs:

- définition de fonctions informatiques
  déclaration et signature, corps de la fonction, composition de fonctions
- comprendre quand le code est récursif
  utilisation algorithmique d'appels récursifs, macros versus fonctions

Usages:

- écrire un algorithme de recherche dichotomique
- factoriser un code existant avec des routines

commencer à apprendre à structurer un code proprement
comprendre pourquoi un programme peut boucler indéfiniment

L'informatique en dix leçons

et deux volets.

7/ Les structures algorithmiques: boucles et itérateurs

Savoirs:

- la construction for($i = 0; $i < n; $i++)
  définition, usages; notion de récursivité primitive
- la construction foreach($structure => $element)
  définition, usages; notion d'itérateurs sur une structure logique

Usages:

- écrire un algorithme de recherche dichotomique
- corriger un algorithme de traitement pixélique d'une image

se familiariser avec la programmation procédurale
apprendre à lire un code existant et à le re-travailler

L'informatique en dix leçons

et deux volets.

8/ Les structures algorithmiques: la notion d'objet

Savoirs:

- ajouter des méthodes aux données structurées
  encapsulation des variables, attacher des routines à un objet
- hériter d'un autre objet et définir des interfaces
  héritage simple, utilisation d'un objet comme variable

Usages:

- écrire la documentation d'un code existant
- encapsuler des routines connues sous forme d'objets

apprendre à séparer interface et implémentation
comprendre comment gérer la complexification d'un développement logiciel

L'informatique en dix leçons

et deux volets.

9/ Les structures algorithmiques: aspects déclaratifs

Savoirs:

- définir un code de manière déclarative
  éléments de programmation par règles fonctionnelles
- spécifier une requête de recherche
  éléments de définition dans un langage de requête

Usages:

- écrire un simplificateur formel élémentaire
- apprendre à utiliser une base de donnée ``jouet´´

se familiariser avec quelques éléments de programmation déclarative
découvrir l'existence de différentes formes de programmation

L'informatique en dix leçons

et deux volets.

10/ Les structures algorithmiques: codes réactifs

Savoirs:

- notion de signal et de mécanisme temporel
  émission/attente sur un signal, parallélisme et causalité
- implémentation sous forme d'automate
  compilation du code, vérifications avant son exécution

Usages:

- créer un code réactif d'un petit avatar numérique
- faire l'analyse d'un mécanisme d'ascenceur

se sensibiliser aux aspects temps-réel de la programmation
découvrir l'univers de l'informatique enfouie

L'informatique en dix leçons

et deux volets.

+/ Mise en musique:l'environnement de programmation

Savoirs:

- notion de code source et code objet
  compilation et édition de lien, code compilé et interprété
- versions de code et distribution
  méthodes de suivi et de déploiement d'un code, différentes licences

Usages:

- mettre un code en ligne sur un site et prévoir son suivi
- utiliser un système de gestion de version

se sensibiliser aux facettes du génie logiciel
s'ouvrir aux modèles économiques émergeants dans le monde numérique

Apprendre l'informatique : mise en perspective

L'informatique en perspective

Perspectives *Pédagogiques.*

**** L'informatique se prête à une pédagogie participative.***

Avec un enseignement par mini-projets, orienté vers le travail en groupe.

Apprendre à programmer un petit logiciel, c'est donner à l'élève des clés, mais aussi la liberté de s'approprier ces clés et de les mettre en pratique de manière diverse (il y a plusieurs possibles dans la manière de mettre en oeuvre la solution).

**** L'informatique favorise l'apprentissage par l'utilisation.***

Ce qui correspond bien à l'esprit humain.

Ex: découvrir un algorithme avant d'en abstraire la notion sous-jacente.

L'informatique en perspective

Perspectives *Pédagogiques.*

**** L'informatique conduit aussi à un apprentissage de la rigueur*.**

Par un mécanisme spécifique : celui des essais-erreurs
avec une machine « neutre »

- qui ne donnera un résultat que si tout est correct,

- mais qui donnera indéfiniment une chance de corriger,
de reprendre, de re-tester

La machine est un outil qui permet d'apprendre de manière incrémentale, sans jamais porter de jugement de valeur.

L'informatique en perspective

**** L'informatique fait entrevoir l'intérêt des sciences théoriques.***

On peut « toucher » (opérer avec, visualiser...) des objets abstraits.

Si l'informatique est mathématique,
alors il s'agit de mathématiques « incarnées ».

Perspectives *Intellectuelles.*

**** L'informatique est un levier pour les sciences.***

Car elle permet de mieux comprendre des notions universelles (par exemple la notion d'information) ou fondamentales (par exemple le calcul « mécaniste » par opposition à d'autres formes de raisonnement).

**** L'informatique offre la découverte de notions nouvelles.***

Exemples: suites aléatoires, fonctions récursives, ...

L'informatique en perspective

Perspectives *Sociétales.*

* ***C'est en apprenant l'informatique le plus tôt possible*** qu'on tirera le meilleur profit de son rôle transversal à la quasi-totalité des autres disciplines universitaires et socio-économiques.

* ***C'est en donnant aux citoyens les clés du monde numérique,*** qu'on se donnera les moyens, ensemble de:

- découvrir comment les nouvelles technologies aideront à relever les grands défis de notre planète et de l'humanité.

- débattre des enjeux sociétaux liés à l'avènement du numérique

Ex : droit logiciel, GreenIT - technologies de l'information « vertes »...

Apprendre l'informatique: qq idées reçues

L'informatique: quelques idées reçues

L'informatique n'est pas une science, c'est de la technologie.

L'informatique est de piètre intérêt pour la formation de l'esprit.

Ce qui fait « marcher les programmes » est bien de la théorie : logique et algorithmique ou mathématiques combinatoires, etc..
Ce sont des logiciens et des physiciens théoriques du siècle dernier qui sont à l'origine de l'informatique.

Impossible, pour une nation, de maîtriser l'informatique et la faire progresser sans maîtriser collectivement ses savoirs.

Les « bugs » constituent un exemple édifiant ! Ils sont le plus souvent le fait de la programmation « programmeurs » non scientifiques.
C'est de l'étude scientifique des logiciels qu'émergent aujourd'hui les logiciels sûrs.

L'informatique: quelques idées reçues

Pas besoin d'apprendre l'informatique, cela s'apprend tout seul.

La preuve : ce sont les enfants qui apprennent l'informatique aux enseignants ! ?

Cette double idée reçue est la conséquence d'une confusion entre l'apprentissage des usages et l'apprentissage des fondements.

Cette idée reçue néglige aussi le fait que, depuis 40 ans, l'informatique s'est stratifiée et complexifiée : on ne peut plus y « bricoler ».

Le mythe de l'auto-apprentissage se brise devant la nécessité d'apprendre au plus grand nombre des savoirs et des pratiques qui doivent être intégrés à l'échelle d'une société entière.

Le risque de mal apprendre (et de devoir passer des heures à se corriger), les risques liés aux mauvaises méthodes (perte de données, logiciels non fiables...) deviennent majeurs : l'enseignement de l'informatique en tant que matière rigoureuse est une nécessité.

L'informatique: quelques idées reçues

L'informatique, ce n'est pas très féminin.

La c . . . . . e non plus, du reste (*Florence Foresti).*

C'est bien la mixité des genres qui aide à créer la mixité et l'ouverture des idées... et l'informatique a quelques grands noms:

Ada Lovelace (1ère notions de programmes et de traitement symboliques)
Emmy Noether (processus algébriques permettant de « mécaniser des calculs »)
Grace Hopper (1er programme de compilation (COBOL), mémoires tanmpons)
Rose Dieng (ontologies informatiques pour le web sémantiques)

féminins mais

la situation reste *discriminatoire* (15-20% effectifs) en France contrairement au pays émergent (65% étudiants/profs en Thaïlande).

L'informatique: quelques idées reçues

L'informatique est déjà omniprésente dans toutes les matières

(mathématiques, technologie, humanités..). Quel est l'intérêt de l'apprendre « en plus » ?

Il est évident qu'un savoir fondamental ne s'apprend que sous la forme d'une matière identifiée.

L'informatique a des bases théoriques indispensables à connaître pour maîtriser le monde numérique,

de même que la physique et la chimie permettent aux futurs ingénieurs de maîtriser les technologies qu'ils seront amenés à utiliser et développer.

Ce sont ces notions fondamentales qu'il faut diffuser à tous les citoyens quel que soit leur futur métier.

L'informatique: quelques idées reçues

On est bloqué par la formation des enseignants !

Comment leur apprendre à apprendre ce qu'ils n'ont pas appris ?

Il faut simplement créer des diplômes d'enseignement (CAPES, Agrégation) en informatique.
Les postes seront pourvus progressivement, lycée par lycée, comme ce fut le cas pour d'autres apprentissages (ex: technologie).

La situation actuelle est particulièrement favorable:
- grand nombre d'étudiants universitaires en informatique
- étudiants de « maths-info » → informatique fondamentale
- la situation démographique va créer une jouvence dans les carrières des enseignants

Conclusion:

L'informatique doit être enseignée . .

. . Vite !

Laisserons nous nos enfants subir ou maîtriser la révolution du numérique ?

« Mon enfant, la révolution numérique est passée. Ton monde ne sera pas pire, pas forcément meilleur, mais il sera en partie à réinventer. Tu es un natif du numérique et ce que nous te léguons ici fait du monde qui est le tien quelque chose qui est "autre".

Viens apprendre à le découvrir et réussir à le comprendre pour pouvoir te l'approprier.»

Liens . . à utiliser sans modération ..

http://interstices.info

contenus de culture scientifique en STIC

http://www-sop.inria.fr/science-participative

sujets de TPE à gogo + support en ligne

http://javascool.gforge.inria.fr

outil pour apprendre à programmer au lycée