Programme d'Informatique en Tronc Commun

Objectifs généraux et lignes directrices

L'informatique en classes préparatoires a pour principaux objectifs d'offrir dans le tronc commun :
- Une familiarisation avec l'utilisation d'outils informatiques évolués (logiciel de calcul formel et numérique, logiciels d'acquisition et de traitement de données, logiciels de modélisation, logiciels de simulation...) en vue de permettre l'approfondissement des disciplines scientifiques et techniques ;
- Une introduction à l'informatique en tant que discipline, par une initiation élémentaire au traitement automatique de l'information, à l'algorithmique et à la programmation structurée (illustrée à l'aide du langage du logiciel de calcul formel retenu).

Utilisation d'outils informatiques

On habituera les élèves à se servir de logiciels, qui fournissent un support au raisonnement par la confrontation rapide et commode des hypothèses et résultats, et permettent :
- D'enrichir la compréhension des phénomènes mathématiques et des modèles physiques par la simulation de leurs comportements en fonction de divers paramètres ;
- De mieux cerner la notion de domaine de validité d'une hypothèse ou d'une méthode par l'étude de cas limites ;
- D'étudier certains problèmes par la mise en œuvre de modèles dont la résolution numérique manuelle serait trop lourde ou trop complexe ;
- D'alléger la part de calcul systématique au profit de l'intuition mathématique ou du sens physique.

Programme du cours d'informatique

Utilisation d'un logiciel de calcul formel

Présentation du logiciel aux élèves

On décrira sommairement la structure et le fonctionnement des ordinateurs (aucune connaissance à ce sujet n'est exigible). On situera le logiciel de calcul formel parmi les outils informatiques. On pourra ainsi expliquer brièvement son fonctionnement (interpréteur, langage de programmation, bibliothèques). On donnera ici quelques idées sur la spécificité du calcul formel et ses différences avec le calcul numérique. L'utilisation du logiciel peut s'envisager, soit de façon interactive, par exécution de commandes directes, soit au moyen de l'écriture de programmes enchaînant des commandes, les deux points de vue étant très liés.

Sous-ensemble du langage à connaître

Variables

- entier, rationnel, flottant, complexe ;
- chaîne de caractères ;
- tableau à une ou plusieurs dimensions d'indice entier ;
- ensemble, liste, intervalle ;
- expressions algébriques.

On expliquera, de façon succincte, la représentation arborescente des expressions manipulées par le logiciel de calcul formel.

Opérateurs de comparaisons, opérateurs logiques et états logiques

- $=, \neq, <, >, \leqslant, \geqslant$, et, ou, non, vrai, faux

On soulignera la différence entre le test d'égalité et l'affectation.

Structures de contrôle
- structures conditionnelles : si. . . alors. . . sinon. . . ;
- structures itératives : boucles conditionnelles ou non conditionnelles.

Fonctions
- arguments ;
- retour de résultats.

Les fonctions peuvent être éventuellement récursives (récursivité simple).
La récursivité est abordée comme moyen d'expression de la récurrence en mathématiques.

On soulignera la différence entre le test d'égalité et l'affectation. Les élèves doivent connaître la distinction qui existe entre les variables *globales* et les variables *locales*.
L'usage de variables locales dans les fonctions est à préférer.
On insistera sur la nécessité d'une programmation très modulaire, reposant sur l'écriture de petits modules.
Le seul mode exigible de passage des arguments sera le passage par valeurs.

Fonctionnalités

Calculs usuels de type arithmétique ou flottant
- calculs exacts dans $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, et sur les expressions ;
- calculs approchés dans $\mathbb{R}$ et $\mathbb{C}$;
- utilisation des opérateurs, fonctions et constantes mathématiques usuels.

Manipulations de polynômes et fractions rationnelles
- développement et factorisation.

Manipulations d'expressions trigonométriques

Commandes mathématiques
- dérivation des fonctions ;
- développements limités et asymptotiques ;
- calcul de limites ;
- suites et séries ;
- calcul matriciel élémentaire ;
- résolution formelle ou numérique de systèmes d'équations ;
- intégration des fonctions ;
- résolution de systèmes d'équations différentielles ;
- analyse vectorielle : gradient, rotationnel, divergence.

les limitations du système seront présentées de manière succincte

Commandes graphiques 2D et 3D
- représentation de courbes et surfaces en coordonnées cartésiennes, paramétriques, polaires, cylindriques et sphériques ;
- courbes et surfaces implicites.

Algorithmique et programmation

Contexte

L'enseignement de la programmation ne constitue pas une fin en soi et est limité à un petit nombre de concepts permettant de décrire un enchaînement d'opérations de base.
Les algorithmes à mettre en œuvre sont de type formel ou numérique. L'objectif principal est d'entraîner les élèves à combiner, sur des exemples simples, un petit nombre de commandes dont la fonction est clairement indiquée, en vue de résoudre un problème pratique donné. Aucune connaissance n'est exigible sur la complexité des algorithmes et sur les techniques de preuve de programmes.

La mise en œuvre de la programmation n'est pas séparée de l'utilisation du logiciel de calcul formel en tant qu'outil et s'effectue à l'occasion des séances de travaux pratiques, appliquées à la résolution de problèmes de mathématiques, de physique, de chimie, de mécanique et automatique.

Utilisation interactive du logiciel de calcul formel

Le logiciel est utilisé comme une aide au calcul et la représentation de résultats. Les élèves doivent être familiers des menus, et des opérations simples d'entrée-sortie. Ils doivent savoir éditer et exécuter des commandes, simples ou enchaînées, pour résoudre un exercice de mathématiques ou de sciences physiques et parvenir à l'obtention d'un résultat formel, numérique ou graphique. L'écriture, syntaxiquement correcte, et la commande de l'évaluation d'expressions doivent être maîtrisées.

L'outil informatique n'est pas une fin en soi mais un moyen efficace pour faire des mathématiques, des sciences physiques ou des sciences industrielles. La connaissance de la liste exhaustive des fonctions prédéfinies et des bibliothèques ne peut être exigée des élèves. Toutefois, ils doivent savoir utiliser l'aide en ligne ou la documentation du logiciel pour retrouver des informations qui leur sont utiles.

Mise en œuvre

Horaires et intégration pédagogique des outils informatiques

L'initiation au logiciel de calcul formel et à l'algorithmique est faite dans la première période de première année (classe supérieure) en quinze heures de cours théoriques assurés par les professeurs de mathématiques et de sciences physiques. Ces cours théoriques sont illustrés au moyen d'un ordinateur, connecté à un dispositif de projection approprié. Cette formation est complétée, en première et deuxième année, par des séances de deux heures tous les quinze jours, par groupes de 12 élèves. Elles ont pour objet non seulement la pratique du logiciel de calcul formel mais aussi l'initiation des élèves à l'utilisation de logiciels plus spécialisés en modélisation, simulation, acquisition et traitement de données, mécanique et automatique. Ces séances sont à répartir entre les enseignements de mathématiques, de sciences physiques et de sciences industrielles. Il est très souhaitable qu'elles soient encadrées, au moins partiellement, par les professeurs scientifiques de la classe. Dans tous les cas, ces professeurs restent responsables de la définition des contenus des activités
Bien que l'apprentissage de l'outil informatique soit limité aux quinze heures de formation théorique et aux séances de travaux pratiques pour les élèves, l'emploi d'un tel outil par les enseignants dans le cours magistral des diverses disciplines scientifiques est fortement encouragé en vue de sa bonne intégration pédagogique. À cette fin, le matériel nécessaire (ordinateur connecté à un dispositif de projection approprié) est disponible et facilement accessible dans les salles de cours, de travaux dirigés et de travaux pratiques.

Environnement

Les spécifications minimales du matériel à utiliser sont publiées par circulaire réactualisée. Le système d'exploitation doit intégrer une interface graphique (environnement multi-fenêtres avec souris ou dispositif équivalent). Une liste limitée de logiciels de calcul formel adaptés au travail en classe préparatoire scientifique est définie par circulaire. L'équipe pédagogique scientifique de l'établissement choisit l'un de ces logiciels, qui servira d'outil informatique en mathématiques, sciences physiques, mécanique et automatique. Par ailleurs le langage de

programmation attaché au logiciel choisi servira de support à la pratique de l'algorithmique. Une liste de logiciels spécialisés à utiliser en mathématiques, sciences physiques, mécanique et automatique est publiée par circulaire et régulièrement actualisée. Ces logiciels peuvent être utilisés à l'oral des concours, ou dans des épreuves de travaux pratiques, mais aucune connaissance spécifique ne peut être exigée des candidats. Les cours et travaux dirigés de mathématiques, de sciences physiques et de mécanique et automatique ont lieu dans des salles équipées d'un micro-ordinateur connecté à un dispositif de projection.

Concours

Il est souhaitable que les concours prennent en compte d'une manière significative les activités informatiques, dans le cadre du programme. C'est le cas à l'écrit, dans les épreuves de mathématiques pratiques ou de mathématiques appliquées. À l'oral, certaines épreuves scientifiques peuvent faire appel à l'utilisation d'un ordinateur équipé, entre autres, des logiciels de calcul formel retenus. En revanche, pour ce qui concerne le tronc commun informatique, aucune épreuve spécifique de programmation ne peut être inscrite à un concours à l'écrit comme à l'oral. Les épreuves orales «assistées par ordinateur» ne doivent évidemment pas sanctionner la capacité du candidat à utiliser le matériel disponible et l'interface du logiciel choisi mais demeurent des épreuves de spécialité. L'examinateur donne toute indication utile pour le bon emploi du matériel et de l'interface du logiciel associé. L'utilisation éventuelle de bibliothèques ne peut être envisagée qu'avec l'appui d'une documentation concise et explicite et l'aide de l'examinateur.

Champs d'application

L'application des outils informatiques dans les disciplines scientifiques n'a pas pour objectif de faire apprendre aux élèves un catalogue de solutions. Il s'agit tout au contraire de développer chez eux la capacité d'utiliser ces outils à bon escient. Dans le cas du logiciel de calcul formel, il importe de conserver toute sa généralité et sa pluridisciplinarité à l'outil. Pour cette raison, on ne recommande pas ici de thèmes précis de travaux pratiques ou d'illustration de cours dans les différentes disciplines. Les activités se prêtant à l'emploi du logiciel, ou à des activités algorithmiques sont signalées dans les programmes de chaque matière

Programme de l'option mathématiques et informatique en classe Supérieure MP

Préambule

L'informatique sera considérée ici comme une science opérant sur des représentations rigoureuses de concepts bien définis. Le programme doit donc permettre de présenter les principes de la programmation ainsi que les bases de l'algorithmique, de la théorie des automates et de la logique. Le programme qui suit se veut à la fois ambitieux et cohérent, tout en évitant d'aborder les concepts trop difficiles ou trop techniques, qui relèvent des études ultérieures en École.
Par ailleurs, un enseignement d'informatique doit être confronté à un «principe de réalité» : les élèves doivent donc avoir à mettre en œuvre les outils conceptuels étudiés, en programmant dans un langage de programmation, sous la forme de programmes clairs, courts et précis. La liste des langages autorisés sera publiée et actualisée par une circulaire. Les candidats au concours ne devront pas être évalués sur leur virtuosité à manipuler tel ou tel langage et il est souhaitable que les épreuves de concours ne favorisent pas les utilisateurs d'un langage particulier.

Méthodes de programmation

On dégagera la méthode d'analyse descendante (c'est-à-dire par raffinements successifs). Même si l'on ne prouve pas systématiquement tous les algorithmes, on dégagera l'idée qu'un algorithme doit se prouver, à l'instar d'un théorème de mathématiques. On étudiera systématiquement la complexité des algorithmes du programme et, sur certains exemples, le lien entre cette complexité et les structures de données. On s'attachera à obtenir des étudiants une documentation aussi complète que possible de leurs algorithmes (conditions d'entrée dans un module, conditions de sortie, invariants dans les boucles ou les appels récursifs). Toutes ces notions seront dégagées à partir des algorithmes au programme, sans aucune théorie sur les prédicats ou les invariants de boucles.

Itération

Boucles conditionnelles et boucles inconditionnelles.

Récursivité

Lien avec le principe de récurrence, exemples tirés des mathématiques (factorielle, puissances, dérivées d'ordre n). Récursivité simple (la fonction s'appelle elle-même), récursivité croisée (deux fonctions s'appellent l'une l'autre).
Lien avec les relations d'ordre ; exemples de récursions fondés sur des relations d'ordre sur des parties de $\mathbb{N}\times\mathbb{N}$.

Comparaisons sur quelques exemples d'algorithmes récursifs et itératifs (factorielle, puissances entières, suites récurrentes, PGGD, insertion et suppression dans un arbre binaire).

Exemples de structures récursives.

On se limitera à une présentation pratique de la récursivité.

On insistera sur l'importance de la relation d'ordre sur l'ensemble permettant de garantir la terminaison de l'algorithme.
On mentionnera certains problèmes posés par la gestion de la récursion au niveau de la machine (occupation mémoire et temps d'exécution) : sauvegarde et restauration du contexte. Toute théorie générale de la dérécursification est hors programme.
On cherchera à dégager sur quelques exemples le lien entre les structures de données et les méthodes de programmation.

Diviser pour régner

Principe général de la méthode.
Exemples d'application : multiplication des entiers (par dichotomie), des polynômes, des matrices, tri rapide, tri par fusion.

L'objectif poursuivi ici est de parvenir à ce que les élèves puissent par eux mêmes, dans une situation donnée, mettre en œuvre la stratégie «diviser pour régner».

Éléments de complexité des algorithmes

Notion de taille de données, évaluation du nombre d'opérations (calculs, comparaisons...) nécessaires à l'exécution et de l'encombrement mémoire.
Notion et exemples de complexité logarithmique, polynomiale, exponentielle (comparaison de temps d'exécution).

On se limitera à dénombrer les opérations nécessaires pour évaluer le temps d'exécution. Tout autre considération est exclue du programme.
On pourra consacrer une séance de travaux pratiques à une évaluation expérimentale de la complexité de divers algorithmes pour résoudre un même problème.

Structures de données et algorithmes

Il s'agit de montrer l'influence des structures de données sur les algorithmes et les méthodes de programmation. Notamment, on mettra en parallèle les structures récursives des types de données et des programmes qui les manipulent. Les algorithmes seront présentés au tableau, en étudiant, dans la mesure du possible, leur complexité (dans le cas le pire et/ou en moyenne). Certains de ces algorithmes feront l'objet d'une programmation effective : les programmes correspondants devront rester clairs, courts et précis.

Listes et Piles

Définition récursive du type liste.

On peut utiliser la notation :

$$\text{Liste} = \text{nil} + \text{Élément} \times \text{Liste}.$$

Fonctions Tête et Queue, parcours correspondant. Calculs récursifs de la longueur, du maximum, du n-ième élément, de la concaténation de deux listes et de l'image miroir. Insertion et suppression d'un élément. Insertion dans une liste triée.

Mêmes programmes avec une programmation itérative. Comparaison des deux méthodes, en montrant la simplicité de l'écriture récursive. Exemples de représentation d'objets mathématiques par des listes.

Piles. Évaluation d'une expression arithmétique postfixée à l'aide d'une pile.

Seule la définition des piles est au programme. La dualité entre récursivité et itération avec piles ne l'est pas.

Arbres

Ce paragraphe n'est pas abordé en MP supérieure.

Automates finis

Ce chapitre n'est pas abordé en MP supérieure.

Notions de logique

Le but de cette partie est de familiariser progressivement les élèves avec la différence entre syntaxe et sémantique, ceci à travers l'étude des expressions logiques et arithmétiques. L'étude du calcul des prédicats et les théorèmes généraux de la logique du premier ordre ne sont pas au programme.

Éléments de base du calcul propositionnel

Variables propositionnelles, connecteurs logiques (NOT, AND, OR, XOR, NAND, $\Rightarrow$, $\Longleftrightarrow$), formules logiques, représentation d'une formule logique par un arbre, évaluation des formules logiques, tables de vérité, tautologie et satisfiabilité, difficulté de tester une tautologie : solutions de caractère exponentiel.

On évitera de parler de preuves et surtout de systèmes logiques. Le principe est d'insister sur l'interprétation d'une formule logique et sur les manipulations logiques élémentaires.

Fonctions booléennes

Fonction booléenne associée à une formule, formules équivalentes : lois de Morgan et du tiers exclu, contraposition, expression de l'implication en fonction de la négation et de la disjonction, formes normales disjonctives et conjonctives.

Circuits élémentaires

Circuits réalisés par des portes logiques (AND, OR, NOT, NAND, NOR, XOR), formules logiques et circuits, additionneur 1-bit et n-bit.

Il s'agit de montrer comment la construction de circuits digitaux suit la formulation logique en calcul des propositions : on donne quelques exemples pour montrer le côté algorithmique. Il ne s'agit pas de faire un cours sur les circuits électroniques ni sur l'architecture des ordinateurs.

Exemples de manipulation formelle de termes et de formules sans quantificateur

Ce paragraphe n'est pas abordé en MP supérieure.

Programme de l'option mathématiques et informatique en classe Spéciale MP

Préambule

L'informatique sera considérée ici comme une science opérant sur des représentations rigoureuses de concepts bien définis. Le programme doit donc permettre de présenter les principes de la programmation ainsi que les bases de l'algorithmique, de la théorie des automates et de la logique. Le programme qui suit se veut à la fois ambitieux et cohérent, tout en évitant d'aborder les concepts trop difficiles ou trop techniques, qui relèvent des études ultérieures en École.
Par ailleurs, un enseignement d'informatique doit être confronté à un «principe de réalité» : les élèves doivent donc avoir à mettre en œuvre les outils conceptuels étudiés, en programmant dans un langage de programmation, sous la forme de programmes clairs, courts et précis. La liste des langages autorisés sera publiée et actualisée par une circulaire. Les candidats au concours ne devront pas être évalués sur leur virtuosité à manipuler tel ou tel langage et il est souhaitable que les épreuves de concours ne favorisent pas les utilisateurs d'un langage particulier.

Méthodes de programmation

On dégagera la méthode d'analyse descendante (c'est-à-dire par raffinements successifs). Même si l'on ne prouve pas systématiquement tous les algorithmes, on dégagera l'idée qu'un algorithme doit se prouver, à l'instar d'un théorème de mathématiques. On étudiera systématiquement la complexité des algorithmes du programme et, sur certains exemples, le lien entre cette complexité et les structures de données. On s'attachera à obtenir des étudiants une documentation aussi complète que possible de leurs algorithmes (conditions d'entrée dans un module, conditions de sortie, invariants dans les boucles ou les appels récursifs). Toutes ces notions seront dégagées à partir des algorithmes au programme, sans aucune théorie sur les prédicats ou les invariants de boucles.

Itération

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Récursivité

On donnera de nouveaux exemples de structures récursives en liaison avec les notions introduites aux paragraphes suivants.

On insistera sur l'équivalence des différentes formes d'une expression algébrique (arbre, notation préfixée, expression parenthésée).

Diviser pour régner

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Éléments de complexité des algorithmes

Étude des récurrences usuelles :

$T(n) = T(n-1) + a.n$	par exemple pour le tri par insertion.
$T(n) = a.T(n/2) + b$	par exemple pour la recherche dichotomique.
$T(n) = 2.T(n/2) + f(n)$	notamment pour la méthode «diviser pour régner»

Utilisation de la notation O.

Notion de taille de données, évaluation du nombre d'opérations (calculs, comparaisons...) nécessaires à l'exécution et de l'encombrement mémoire.
Notion et exemples de complexité logarithmique, polynomiale, exponentielle (comparaison de temps d'exécution).
Exemples de calculs de complexité dans le cas le plus défavorable, dans lé cas moyen (tris, recherches dans un tableau ou dans un arbre, calcul de PGCD).

Comparaisons de complexités de divers algorithmes pour résoudre un même problème : calcul des puissances, évaluation de polynômes, suites récurrentes (méthode récursive simple, méthode récursive avec stockage des résultats intermédiaires, méthode itérative), recherche dans un tableau ordonné (recherche linéaire, recherche dichotomique).

On se limitera à dénombrer les opérations nécessaires pour évaluer le temps d'exécution.
Tout autre considération est exclue du programme.

Pour le cas moyen, on traitera un ou deux exemples où l'on peut mener simplement un calcul explicite grâce à l'analyse combinatoire et dans certains cas on se contentera de notions intuitives sur la probabilité de répartitions des données.
En particulier, on mettra en évidence sur des exemples le compromis souvent nécessaire entre temps de calcul et occupation de la mémoire. On montrera comment le stockage de résultats intermédiaires peut diminuer la complexité d'un algorithme.

Structures de données et algorithmes

Il s'agit de montrer l'influence des structures de données sur les algorithmes et les méthodes de programmation. Notamment, on mettra en parallèle les structures récursives des types de données et des programmes qui les manipulent. Les algorithmes seront présentés au tableau, en étudiant, dans la mesure du possible, leur complexité (dans le cas le pire et/ou en moyenne). Certains de ces algorithmes feront l'objet d'une programmation effective : les programmes correspondants devront rester clairs, courts et précis.

Listes et Piles

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Arbres

Définition récursive du type arbre binaire.

On peut utiliser la notation :

$$\text{Arbre} = \text{nil} + \text{Arbre} \times \text{Élément} \times \text{Arbre}.$$

Nœuds, feuilles. Arbres n-aires. Hauteur. Arbres équilibrés, et hauteur en $\log n$ dans le cas d'un tel arbre de n feuilles. Relation entre nombre de nœuds et nombre de feuilles dans le cas des arbres binaires.

On ne parlera pas des arbres AVL, 2-3, 2-3-4 ou bicolores.

Calculs récursifs du nombre de nœuds, de la hauteur, du nombre de feuilles. Insertion et suppression dans un arbre. Arbres de recherche.

On insistera sur la simplicité de l'écriture récursive.

Représentation des expressions arithmétiques par des arbres. Parcours préfixes, infixes et postfixes d'arbres. Application à l'impression préfixe, infixe ou postfixe d'expressions arithmétiques.

Cette partie interagit avec le dernier paragraphe du chapitre logique.

Automates finis

Il s'agit ici de présenter un modèle élémentaire de calculateur. On insistera sur cet aspect en présentant les automates comme des systèmes simples réagissant à des événements extérieurs.

Automates finis déterministes et non-déterministes.	On donnera une représentation graphique des automates.
Langage reconnu par un automate. Algorithme de déterminisation.	On présentera à ce propos les définitions de base : alphabet, mot, langage.
Expressions rationnelles. Langage associé.	On précisera bien la différence entre expressions rationnelles et langages.
Propriétés de fermeture des langages reconnus par automate : complémentation, intersection, union, concaténation et itération.	On insistera évidemment sur l'aspect constructif de ces propriétés de fermeture.
Lemme de l'étoile. Existence de langages non reconnus par un automate.	

Notions de logique

Le but de cette partie est de familiariser progressivement les élèves à la différence entre syntaxe et sémantique, ceci à travers l'étude des expressions logiques et arithmétiques. L'étude du calcul des prédicats et les théorèmes généraux de la logique du premier ordre ne sont pas au programme.

Éléments de base du calcul propositionnel

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Fonctions booléennes

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Circuits élémentaires

Ce paragraphe ne fait pas l'objet d'un programme spécifique en classe de spéciales.

Exemples de manipulation formelle de termes et de formules sans quantificateur

Différence entre syntaxe abstraite (ou arborescente) et valeur d'une expression arithmétique. Évaluation dans $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$ ou $\mathbb{C}$ d'une expression arithmétique (avec les opérateurs $+$, $-$, $\times$, $/$). Dérivation formelle. Exemples de formules sans quantificateur écrites avec des symboles de prédicat unaires ou binaires ; interprétation.	Le but de cette partie est de montrer la différence entre syntaxe et sémantique, expression formelle et interprétation. Elle est l'occasion d'une familiarisation avec du calcul formel tout à fait élémentaire et elle fournit des exemples de formules logiques qui peuvent servir dans des preuves de programmes simples (par exemple le tri).