Projet de programme d'informatique en BCPST

1 / Objectifs de formation

1.1 / Généralités

L'enseignement de l'informatique en classes préparatoires de la filière BCPST a pour objectif d'introduire puis de consolider les concepts de base de l'informatique, à savoir l'analyse et la conception de processus de raisonnement automatisé, c'est-à-dire des algorithmes, et la question de la représentation des données. Aussi souvent que possible, on favorisera une contextualisation des thèmes informatiques étudiés en s'appuyant sur les autres disciplines scientifiques : biologie, géologie, chimie, physique ou mathématiques.

1.2 / Compétences visées

Cet enseignement doit permettre de développer les compétences suivantes :

analyser et modéliser	un problème, une situation, en lien avec les autres disciplines scientifiques
imaginer	une solution modulaire, utilisant des méthodes de programmation, des structures de données appropriées pour le problème étudié
traduire	un algorithme dans un langage de programmation
spécifier	rigoureusement les modules ou fonctions
évaluer, contrôler, valider	des algorithmes et des programmes
communiquer	à l'écrit ou à l'oral, une problématique, une solution ou un algorithme, une documentation.

L'étude et la maîtrise de quelques algorithmes fondamentaux, l'utilisation de structures de données adaptées et l'apprentissage de la syntaxe du langage de programmation choisi permettent de développer des méthodes (ou paradigmes) de programmation fiables et efficaces : programmation impérative, approche descendante, programmation structurée, utilisation de bibliothèques logicielles, notions élémentaires de complexité en temps ou en mémoire, documentation.

La pratique régulière de la résolution de problèmes de nature algorithmique et des activités de programmation qui en résultent est un aspect essentiel de l'apprentissage de l'informatique. Il est souhaitable que les exemples choisis ainsi que certains exercices d'application soient inspirés par les enseignements de biologie et géologie, de physique et chimie, ou de mathématiques.

Le travail sur la documentation est également important, combinant la documentation des programmes lors de leur conception, en vue de leur réutilisation et possibles modifications ultérieures, avec la pratique raisonnée de la recherche d'informations pertinentes dans les documentations en ligne décrivant les différents composants logiciels que les étudiants auront à manipuler.

Enfin, les compétences acquises en informatique ont vocation à participer pleinement à l'élaboration des travaux d'initiative personnelle encadrée (T.I.P.E.) et à être réutilisées au sein des autres enseignements scientifiques.

2 / Programme de première année (BCPST1)

2.1 / Organisation de cet enseignement

L'ordre de présentation des notions et situations présentées dans les parties 2.3 et 2.4 ci-dessous n'est pas imposé. Il est d'ailleurs recommandé de créer de nombreux liens entre algorithmique et programmation, tout en distinguant soigneusement ces deux domaines.

Un temps introductif sera prévu :

- pour présenter et analyser les relations entre les principaux composants d'une machine numérique telle que l'ordinateur personnel ou un appareil photo numérique : sources d'énergie, mémoire vive, mémoire de masse, processeur, périphériques d'entrée-sortie, ports de communication avec d'autres composants numériques (aucune connaissance particulière des composants cités n'est exigible) ;
- pour présenter et faire manipuler un système d'exploitation (essentiellement : arborescence de fichiers, droits d'accès et de modification de ces derniers) ;
- et pour présenter et faire manipuler un environnement de développement.

2.2 / Outils employés

L'enseignement se fonde sur un environnement de programmation (langage et bibliothèques) basé sur un langage interprété largement répandu et à source libre. Au moment de la conception de ce programme, l'environnement choisi est Python. Des textes réglementaires ultérieurs pourront mettre à jour ce choix en fonction des évolutions et des besoins.

Les travaux pratiques conduiront à éditer et manipuler fréquemment des codes sources et des fichiers ; c'est pourquoi un environnement de développement efficace doit être choisi et utilisé. Les étudiants doivent être familiarisés avec les tâches de création d'un fichier source, d'édition d'un programme, de gestion des fichiers, d'exécution et d'interruption d'un programme.

L'étude approfondie de ces divers outils et environnements n'est pas une fin en soi et n'est pas un attendu du programme.

2.3 / Programmation

On insistera sur une organisation modulaire des programmes ainsi que sur la nécessité d'une programmation structurée et parfaitement documentée.

Contenus	Capacités	Commentaires
Variables notion de type et de valeur d'une variable, types simples.	Choisir un type de données en fonction d'un problème à résoudre.	Les types simples présentés sont les entiers, flottants, booléens et chaînes de caractères.
Expressions et instructions affectation, opérateurs usuels, notion d'expression.		Les expressions considérées ont des valeurs numériques, booléennes ou de type chaîne de caractères.
Instructions conditionnelles expressions booléennes et opérateurs logiques simples, instruction **if**.	Agencer des instructions conditionnelles avec alternatives, éventuellement imbriquées.	L'ordre d'évaluation n'est pas un attendu du programme.
Fonctions notion de fonction (au sens informatique), définition dans le langage utilisé, paramètres (ou arguments) et résultats, portée des variables.	Concevoir l'entête (ou la spécification) d'une fonction, puis la fonction elle-même ; documenter une fonction, un programme plus complexe.	On distingue les variables locales des variables globales, tout en valorisant l'usage de variables locales.
Instructions itératives boucles **for**, boucles conditionnelles **while.**	Organiser une itération, contrôler qu'elle s'achève.	Les sorties de boucle (instruction **break**) peuvent être présentées à l'occasion d'exemples lorsqu'elles contribuent notablement à simplifier la programmation.

Contenus (suite)	Capacités	Commentaires
Manipulation de quelques structures de données chaînes de caractères (création, accès à un caractère, concaténation), listes (création, ajout d'un élément, suppression d'un élément, accès à un élément, extraction d'une partie de liste), tableaux à une ou plusieurs dimensions.	Traduire un algorithme dans un langage de programmation.	On met en évidence le fait que certaines opérations d'apparence simple cachent un important travail pour le processeur On prépare les démarches qui vont suivre en introduisant la structure d'image ponctuelle (ou bitmap) en niveaux de gris, assimilée à un tableau d'entiers à deux dimensions.
Fichiers notion de chemin d'accès, lecture et écriture de données numériques ou de type chaîne de caractères depuis ou vers un fichier.	Gérer efficacement et durablement une série de fichiers, une arborescence.	On encourage l'utilisation de fichiers en tant que supports de données ou de résultats avant divers traitements, notamment graphiques.
Bibliothèques logicielles utilisation de quelques fonctions d'une bibliothèque et de leur documentation en ligne.	Accéder à une bibliothèque logicielle ; rechercher une information au sein d'une documentation en ligne.	On met en évidence l'intérêt de faire appel aux bibliothèques, évitant de devoir réinventer des solutions à des problèmes bien connus. La recherche des spécifications des bibliothèques joue un rôle essentiel pour le développement de solutions fiables aux problèmes posés.

2.4 / Algorithmique

Cette partie rassemble un petit nombre d'algorithmes classiques et d'usage universel ; les attendus du programme se limitent en général à la compréhension et à l'usage de ces algorithmes (éventuellement par appel à une fonction d'une bibliothèque), sauf pour ceux dont la programmation effective doit être étudiée (signalés par un symbole ♦).

Contenus	Commentaires
♦ Recherche dans une liste, ♦ recherche du maximum dans une liste de nombres, ♦ calcul de la moyenne, ♦ de la médiane.	
♦ Recherche d'un mot dans une chaîne de caractères.	On se limite à l'algorithme « naïf », en estimant sa complexité.
Algorithmes de tri d'un tableau à une dimension de valeurs numériques : tri à bulles, ♦ tri par insertion.	Le tri rapide sera évoquée en seconde année.
Exemples d'algorithmes simples opérant sur une image ponctuelle en niveaux de gris.	Les algorithmes présentés sont du type « à balayage » et restent très simples : éclaircissement, accentuation du contraste, flou, accentuation de contours. On met en évidence l'importance de tels algorithmes en les appliquant sur des images issues de la biologie et des géosciences.
♦ Simulation d'une variable aléatoire prenant un nombre fini de valeurs.	On se sert du générateur de nombres pseudo-aléatoires fourni par le langage.

Capacités intervenant dans cette partie : expliquer ce que fait un algorithme donné ;
modifier un algorithme existant pour obtenir un résultat différent ;
concevoir un algorithme répondant à un problème précisément posé.

3 / Programme de seconde année (BCPST2)

En seconde année, l'enseignement d'informatique est orienté vers la pratique et la consolidation des compétences fondamentales. Les trois volets indiqués ci-dessous concourent à enrichir la culture des étudiants par un apport modeste de nouvelles méthodes et la réalisation d'un projet.

3.1 / Compléments d'algorithmique

Les algorithmes décrits ci-dessous sont présentés puis mis en œuvre soit par programmation soit en faisant appel à une fonction de bibliothèque.

Contenus	Commentaires
Algorithmes de tri d'un tableau à une dimension de valeurs numériques : tri rapide (ou quicksort).	La présentation du tri rapide peut servir à introduire l'idée de la récursivité sans entrer dans les détails de gestion de la mémoire.
Algorithme de Dijkstra de recherche de plus court chemin dans un graphe pondéré à poids positifs.	Le graphe est représenté par la matrice d'adjacence.
Simulation d'une variable aléatoire à densité suivant une loi uniforme, exponentielle ou normale.	On se sert du générateur de nombres pseudo-aléatoires fourni par le langage, et, pour la loi normale, d'une fonction de bibliothèque.

Capacités intervenant dans cette partie : expliquer ce que fait un algorithme donné ;
modifier un algorithme existant pour obtenir un résultat différent ;
concevoir un algorithme répondant à un problème précisément posé.

3.2 / Méthodes numériques

L'utilisation des bibliothèques de calcul numérique ou matriciel, de visualisation de données ou de traitement d'images, ou encore de bioinformatique, permet d'introduire les méthodes numériques classiques de résolution de problèmes issus des autres disciplines : résolution approchée d'équations différentielles, résolution de systèmes linéaires, statistiques, simulation, traitement et représentation de données expérimentales ou de mesures directement prélevées sur des montages expérimentaux, etc.

Un exemple important de traitement numérique est fourni par les images ponctuelles (ou bitmap) en niveaux de gris ou en couleurs, qui apparaissent fréquemment dans de nombreux contextes expérimentaux ou appliqués (radiologie, échographie etc.). Les algorithmes de transformation ou d'extraction permettent d'analyser des structures, distributions, comportements et peuvent participer à des démarches de diagnostic.

L'objectif pédagogique est double : d'une part, savoir repérer et utiliser correctement les fonctions utiles d'une bibliothèque logicielle en se servant de la documentation en ligne, et d'autre part prendre conscience des questions posées par le calcul sur les nombres flottants (arrondis, précision du calcul, différence entre un nombre très petit et un nombre nul).

La connaissance détaillée de ces bibliothèques n'est pas un attendu du programme, lequel se limite à quelques exemples contextualisés d'utilisation d'éléments de bibliothèques.

Capacités intervenant dans cette partie : accéder à une bibliothèque ;
rechercher une information dans une documentation en ligne ;
documenter un programme réalisé en s'appuyant sur une ou plusieurs bibliothèques ;
identifier ou construire un modèle ;
confronter un modèle au réel ;
développer un regard critique sur les résultats obtenus.

3.3 / Réalisation d'un projet

L'acquisition durable de compétences, même modestes, en informatique repose sur une régularité d'exercices pratiques et s'accorde au mieux avec le développement de projets. Il est donc recommandé de faire réaliser aux étudiants un projet mettant en valeur les compétences acquises, dès que ces compétences commencent à être effectives. Pour la réalisation de ce projet, les étudiants peuvent travailler en groupe de taille

réduite (4 au maximum). Le temps passé sur les projets doit rester modeste afin de ne pas empiéter sur les autres tâches et disciplines.

Les thèmes des projets doivent être choisis de manière à représenter la diversité des applications possibles, notamment en biologie et géologie. Un renouvellement fréquent des thèmes des projets est indispensable afin d'éviter la reproduction sans enjeu d'activités stéréotypées et de développer l'esprit d'innovation chez les étudiants.

Ces projets doivent pouvoir être présentés (sous forme écrite et orale) par les étudiants en mettant en valeur :

- la nature et l'intérêt du problème scientifique étudié
- l'approche choisie pour résoudre le problème
- l'organisation choisie pour la conduite du projet (répartition des tâches, échéancier)
- la structuration de la solution (découpage en diverses tâches et modules)
- l'adéquation de la solution par rapport au problème initialement posé.

Capacités intervenant dans cette partie : recueillir des informations et mobiliser des ressources ;
initier des perspectives nouvelles ;
organiser un travail impliquant un développement logiciel ;
collaborer au sein d'une équipe pour réaliser une tâche ;
développer un regard critique sur les résultats obtenus ;
présenter une solution à l'écrit, à l'oral.