Table des matières

A. Objectif : apprendre à programmer

L'objectif est d'apprendre à programmer ce qui signifie maitriser la conception algorithmique avec au minimum un langage de programmation. Nous avons choisi les langages C et C++ du fait de leur efficacité et de leur complémentarité mais aussi de leur généricité. En effet, le but est plus largement d'acquérir un savoir faire fondamental en programmation informatique qui va permettre ensuite de circuler relativement facilement dans la plupart des autres techniques et langages de programmation.

B. Contenu du cours

Il s'agit d'un parcours des structures de données fondamentales qui réunit par la pratique l'algorithmique à la programmation. Les structures de données sont envisagées du point de vue d'une évolution en trois étapes majeures :

- Les fondamentaux natifs en langage C mis en pratique avec l'écriture et la recherche d'algorithmes.
- L'exploration de modèles de données listes (arbres binaires compris), non natives en langage C, avec l'expérimentation d'un répertoire d'algorithmes pour les manipuler.
- La découverte des notions de classes et d'objets et l'essentiel des concepts associés en langage C++.

C. Publics visés

Actuellement ce cours est donné dans sa totalité en premier cycle d'informatique. Il couvre les trois premiers semestre de la licence. Les deux premiers semestres sont consacrés à la maitrise des fondamentaux inclus dans le langage C (chapitres 1,2, 3, 4). Le troisième semestre est consacré à l'expérimentation des listes (chapitres 5 et 6) et à la découverte des notions de classes et d'objets en langage C++ (chapitre 7).

Ces trois grands modules peuvent être dissociés et adaptés selon les besoins d'autres publics. Par exemple la partie qui C est utilisée en écoles d'ingénieurs pour des modules de programmation de niveaux initiation ou avancé. Elle a également été utilisé en master de multimédia pour une formation à l'algorithmique avec une réflexion sur l'écriture et la création informatique. La partie C++ est aussi utilisée à part, dans un module de mise à niveau en dernière année de licence ou en première année de master d'informatique. L'expérimentation des listes apparaît surtout comme un module d'approfondissement, un bagage de culture informatique incontournable, utile lorsqu'il est corrélé avec des projets C ou C++.

D. Comment apprendre à programmer ?

Du point de vue pédagogique, la réponse sur laquelle nous insistons est : par la pratique. Dans ce domaine le savoir théorique est indissociable de l'expérimentation répétée à travers des réalisations diverses et variées et de différents niveaux. Avec le temps la pratique seule permet d'acquérir une intuition spécifique qui permet plus facilement de repérer ou de poser des problèmes et de trouver des solutions ensuite.

1. Comprendre n'est pas savoir faire

Pour les débutants confrontés à un langage de programmation, en général comprendre ne suffit pas. Il faut aussi savoir faire et comprendre n'est pas savoir faire. Il faut certes savoir, c'est à dire connaître les outils disponibles et les concepts fondamentaux associés, mais il faut également être capable de les mettre en pratique afin de faire un programme. Faire un programme nécessite une bonne maitrise des outils, mais aussi de savoir faire des hypothèses de réalisation, c'est à dire de pouvoir identifier et poser les problèmes sous-jacents afin d'être en mesure d'inventer des solutions pour les résoudre. Il s'agit d'un processus d'élaboration et de création qui nécessite de la recherche et de la créativité. Les solutions, hormis ce qui est donné dans des librairies de classes ou de fonctions, ne préexistent pas. Trouver une solution c'est souvent créer une solution.

Pour prendre une image, c'est un peu la même situation pour apprendre à jouer d'un instrument de musique. Si quelqu'un vous explique comment jouer de la guitare vous pouvez comprendre les explications mais vous n'êtes pas devenu guitariste ni compositeur de musique. Devenir guitariste suppose de nombreuses heures de travail sur son instrument. Devenir compositeur et avoir des idées de morceaux de musique ou d'arrangements musicaux supposent également beaucoup de questionnements et de travail. C'est la même chose en programmation et conception informatique.

Dans la perspective de savoir-faire nous avons tenu non seulement à exposer le fonctionnement des outils fondamentaux du langage mais également à fournir les moyens de les mettre en pratique. Ainsi chaque section est suivie d'une section "mise en pratique" dédiée. Ce sont des exercices que nous avons essayés de beaucoup diversifier. Chaque exercice est une invitation. Il ne s'agit pas nécessairement de répondre à tout. L'important est de se poser question afin de se mettre en marche. Si une idée de programme à faire vous vient pendant la partie théorique ou à l'occasion d'un exercice, surtout faites le !

2. Trois niveaux de difficulté

Lors de l'apprentissage trois niveaux de difficulté apparaissent : la maîtrise des outils, la résolution de problèmes, la conception de programmes. Pour chaque niveau des objectifs spécifiques sont atteints avec des exercices variés, et répétés pour les points plus difficiles.

a. Maitriser les outils

Ce que nous appelons les outils ce sont les concepts fondamentaux de la programmation et de l'algorithmique tels qu'ils sont implémentés dans le langage C. Il ne sont pas nombreux et sans entrer dans les détails nous en dénombrons neuf en tout. Il y a les variables simples, les opérations, les sauts (if, else), les branchements (switch) les boucles (while, do-while, for), les fonctions, les variables ensembles (structures et tableaux), les variables pointeur et c'est tout. Avec ça il est possible d'écrire tout ce que l'on veut, y compris un système d'exploitation comme unix (qui est à l'origine du langage C).

A ce niveau des outils de base nous proposons de :

- comprendre ces outils,
- connaître leur fonctionnement et être capable de les faire tourner dans sa tête afin de pouvoir anticiper leurs comportements dans un programme,
- savoir lire et interpréter du code où ils apparaissent,
- savoir écrire du code pour les mettre soi-même en œuvre.

A ce premier niveau il n'y a pas véritablement de conception. Il s'agit par exemple de faire une boucle dans un programme de test pour voir comment elle marche et comprendre son fonctionnement.

b. Résoudre un problème

Au niveau suivant il s'agit d'enter dans une dynamique de résolution de problème et de conception algorithmique. Il ne suffit plus de comprendre les outils il faut encore être capable de les utiliser pour faire quelque chose même de simple. Maintenant que vous avez un marteau, un tourne vis, une scie, des clous, des vis, un crayon, une règle et différentes planches, il faut faire un meuble, par exemple une armoire.

A ce niveau d'entrée dans la conception nous proposons :

- d'acquérir la capacité de lire et de comprendre un algorithme
- d'être capable de repérer une structure de donnée adéquate
- d'être capable d'inventer et d'écrire un algorithme

Tout ce qui s'écrit et qui est exécutable par la machine passe par l'élaboration d'un algorithme qui est une suite finie d'opérations en vue de l'accomplissement d'une tâche. A savoir une suite réfléchie d'instructions pour faire faire quelque chose à la machine. Notons que l'algorithmique donne lieu à de véritables découvertes et références scientifiques. Il y a un patrimoine d'algorithmes et certains portent le nom de leurs auteurs, par exemple l'algorithme de Bresenham pour le tracé de droite. C'est dire qu'à ce niveau la difficulté peut être très grande. Mais heureusement il y a aussi des choses très accessibles et, selon notre philosophie, chacun fera ce qu'il pourra avec les défis qu'il se fixera et des exercices choisis.

c. Concevoir un programme

Le dernier niveau que nous envisageons est celui de la composition d'un programme complet faisant plusieurs milliers de ligne de code et réunissant de nombreuses résolutions d'énigmes, de nombreux algorithmes. Ce peut être un jeux vidéo classique où tout autre réalisation. Il s'agit de réaliser un projet conséquent qui permettra de :

- comprendre un programme complet
- savoir concevoir un programme complet

A ce niveau, les programmes peuvent réunir de nombreux fichiers sources et faire appel à de nombreuses librairies. Des problématiques de méthodologie se posent pour l'écriture du code et pour la gestion d'un projet réalisé éventuellement en équipe. Un éventail d'étapes de travail apparaît. Elles relient l'idée initiale au programme réel qui marche plus ou moins sans bogue et fait plus ou moins ce que l'on attend de lui à la plus ou moins grande satisfaction du client. C'est la conduite et la gestion de projet informatique. Notre objectif ici n'est pas d'approfondir ces questions mais, nous l'espérons, de préparer à en comprendre les fondements et les enjeux.

3. Un apprentissage non linéaire

D'une façon général l'apprentissage n'est pas réductible à une mécanique simple. La clarté se fait progressivement un peu comme un brouillard qui se lève. Au départ on ne distingue rien et petit à petit, au fur et à mesure que le brouillard se dissipe, des formes apparaissent de plus en plus précises et claires. Il arrive que certains se sentent perdus au départ comme devant un mur lisse sur lequel ils n'ont aucune prise. Peut-être un peu la même impression dans un pays étranger en entendant une langue que l'on ne maitrise pas bien et que l'on s'efforce de comprendre. De ce fait l'apprentissage n'est pas frontal, direct, facile à planifier. Avec le temps et des exercices des pôles de clarté, de compréhension et de savoir faire s'affirment et se relient doucement entre eux. Et même si des zones de flou perdurent, la persévérance finit toujours par en venir à bout. Les étapes de l'acquisition de la compétence de concevoir et réaliser un programme informatique dépendent des personnes et du travail consenti. Le talent ici n'est pas d'être brillant c'est plutôt d'aimer programmer parce que celui ou celle qui aime programmer y passe plus de temps que les autres et il devient meilleur pour cette raison.

En cas de découragement penser que c'est un univers fini. Le nombre des outils fondamentaux est limité. Il ne va pas augmenter. C'est un point rassurant. De plus le langage C fournit un passeport efficace pour tous les langages ensuite et cet apprentissage est un bon placement.

E. Organisation du livre

1. Chapitres

La partie qui concerne l'apprentissage du langage C et les premières armes en algorithmique est la plus développée. Elle regroupe les quatre premiers chapitres.

Le premier chapitre permet de maitriser l'écriture et l'organisation d'instructions simples à partir de variables basiques et d'opérations effectuées dessus. A ce niveau, beaucoup de détails nécessaires et peu de conception possible.

Le second chapitre introduit la notion de bloc d'instructions à savoir un niveau supérieur d'organisation des instructions avec les possibilités fondamentales de saut, branchements, boucles et écriture de fonctions. A ce niveau des problèmes algorithmiques peuvent être posés et se pose également la question du style pour l'écriture et la mise en page d'un programme.

Le troisième chapitre expose des types de données beaucoup plus puissants : les structures et les tableaux. Ces nouveaux types de variables permettent d'accéder à des niveaux de conception beaucoup plus élaborés. Il devient possible d'écrire un vrai programme de plusieurs milliers de lignes de code. A ce niveau la difficulté porte sur des questions de modélisation, d'idée générale, de recherche et de cheminement à conduire pour réaliser un projet. Dans le cadre d'un enseignement de licence d'informatique, ce chapitre termine le premier semestre de travail et s'accompagne de la réalisation d'un projet. La plupart du temps il s'agit de réaliser un jeu en mode console. Pour ce faire deux librairies de développement de jeux en mode console sous windows sont développées et accessibles sur internet. La première a été développée par un étudiant particulièrement avancé, M. Julien Batonnet et la seconde fait en quelque sorte partie du cours, afin d'assurer le développement de projets intéressants. Cette librairie est présentée en annexe.

Le quatrième chapitre va surtout donner de la puissance à l'écriture via les variables pointeurs qui permettent l'allocation dynamique et l'élaboration de structures de données non natives dans le langage comme les listes chainées. Compte tenu de la difficulté rencontrée en général devant cet apprentissage, le second semestre lui est consacré. Un module complémentaire de programmation graphique avec la librairie ALLEGRO lui est associé afin de pouvoir à l'issue réaliser des programmes en mode graphique. Cette librairie avec sa documentation est accessible sur différents sites internet. Sur le site http://fdrouillon.free.fr vous pouvez trouver également un tutorial en français et de nombreux exemples de code. Le second semestre se termine avec un nouveau projet en mode graphique cette fois.

Dans ces quatre premiers chapitres qui constituent la première partie, la plupart des points importants sont résumés avec des petits programmes commentés nommés "Expérimentations". Les expérimentations peuvent être très pratiques pour condenser le cours afin par exemple de le présenter à un public déjà formé ayant juste besoin de se remettre en mémoire tous les points importants du langage C.

La partie suivante regroupe les chapitre 5 et 6 et concerne essentiellement les listes. Le chapitre 6 présente listes chainées dynamiques et statiques, mais également les piles et de files ainsi que les arbres, essentiellement les arbres binaires. Pour ce faire le chapitre 5 donne les clés de la récursivité, élément incontournable pour l'implémentation des arbres en C et C++. Ces structures de données sont présentées du point de vue de leur implémentations en C et de leurs fonctionnements. Souvent ces sujets sont traités à part dans des ouvrages spécialisés de la rubrique "structures de données et algorithmes". Nous avons choisi de les présenter ici entre le langage C et le langage C++ parce que ces figures attestent d'un type particuliers de structures de données. Des structures de données qui font appellent à des constructions et manipulations algorithmiques. Des

structures de données qui peuvent constituer en elles-mêmes des petits programmes et que l'on va retrouver tout naturellement dans des bibliothèques de classes du monde objet. De ce fait il nous a paru intéressant d'en faire une passerelle entre le monde sans objet du C et et le monde de l'objet C++ tout en affirmant que les deux aspects, savoir utiliser d'une part et savoir implémenter d'autre part, restent d'actualité.

La dernière partie constituée du chapitre 7 présente les fonctionnalités et la dimension objet du langage C++. Tous les points importants sont abordés avec des petits programmes d'illustration. L'idée est de faire un marche-pied conséquent vers le monde de l'objet et d'autres langages objet. En général, à l'issue les étudiant(e)s n'ont aucun problème pour passer à java ou C# par exemple.

2. Solutions des exercices

Les solutions aux exercices ne sont pas dans le livre mais fournies à part sur le site de ENI. Toutes les solutions ne sont pas données compte tenu du nombre très important des exercices proposés. Certains en effet peuvent être des sujets de projets et il serait dommage de devoir s'en priver le cas échéant. Mais en ce qui concerne les solutions, d'une façon générale attention à bien en user. Pourquoi ? Parce que l'essentiel est de commencer par se poser problème. Sans problème il n'y a pas de solution et une solution qui vient sans que l'on se soit posé sérieusement problème ni que l'on ait cherché activement ne sert à rien. C'est pire que pas de solution du tout. L'illusion d'avoir compris peut maintenir dans l'illusion d'avoir la compétence de trouver alors qu'on ne l'a pas. Il faut aborder un problème, chercher une solution et acquérir les compétences requises pour trouver et faire soi-même. Le concepteur finit toujours par obtenir la technique dont il a besoin pour réaliser ce qu'il a décidé de réaliser.

3. Annexes

En annexe se trouvent un tableau de la priorité des opérateurs et surtout une présentation de la librairie CREACO donnée en complément du livre sur le site de ENI ou sur le site http://fdrouillon.free.fr. Cette librairie fournit un certain nombre de fonctions pour faire de jeux en mode console, c'est à dire avec des lettres et seize couleurs. Actuellement la librairie gère la souris, le clavier, permet de redimensionner la fenêtre et fournit la possibilité d'un affichage extrêmement rapide pour faire des animations fluides. Il y a beaucoup à faire pour développer cette librairie qui est open source. Ce qui est fourni ici n'est qu'une amorce de ce qu'il est possible de faire et c'est très intéressant. N'hésitez pas à la compléter le cas échéant.

La librairie originelle est le fait d'un étudiant M. Julien Batonnet. Sa librairie, toujours open source est en réalité plus avancée dans le développement. Elles est également accessible sur internet à http://libconlib.googlecode.com/svn/trunk/

F. Environnements de développement

Tout le code présenté est en théorie portable. Seules l'utilisation des librairies spécialisées comme windows.h peut poser problème. Il faut alors trouver un équivalent chez mac et sous linux. Mais en ce qui concerne les langages C et C++, c'est totalement standard, aux normes.

Le travail a été réalisé avec le compilateurs Mingw32 et l'IDE CodeBlocks. Ce compilateur et cet environnement de développement sont très pratiques et faciles à installer. Vous pouvez vous procurer le tout sur http://www.codeblocks.org/

Le compilateur Microsoft visual C++ est également utilisé, notamment lorsque la nécessité d'un débogueur plus important est utile. Microsoft diffuse des versions gratuites des ces outils.

G. Remerciemments

Avant de conclure je voudrais remercier en particulier toutes les personnes qui m'ont donné la possibilité d'enseigner. Mes remerciements sincères vont ainsi, pour l'Université Paris 8 Vincennes saint Denis à Ghislaine Azémard et Pierre Audibert de l'UFR MITSIC (mathématiques, informatique, technologies, sciences de l'information et de la communication.)

Pour l'école centrale d'électronique à Paris (ECE), je remercie chaleureusement Frédéric Ravaut, Enseignant-chercheur, Responsable des enseignements d'informatique et M. Jean-Pierre Segado Professeur Responsable Adjoint des enseignements en Informatique du cycle L. Pour l'institut supérieur de l'automobile et des transports de Nevers (ISAT) je remercie M. Philippe Brunet de la direction pédagogique et pour l'école supérieure d'ingénierie informatique de Nevers (CS2I-Bourgogne) où j'enseigne actuellement, son ancien Directeur de la pédagogie M. Bernard Buffière ainsi que celui qui lui succède maintenant M. David Aurousseau.

Il va sans dire que je remercie également personnellement chaque étudiant et étudiante que j'ai rencontré et avec l'ensemble desquels j'ai partagé cette joie formidable de créer et d'écrire des programmes.

H. Conclusion : la programmation comme écriture

La programmation est envisagée ici comme une écriture qui permet d'exprimer spécifiquement des idées et des points de vue sur pratiquement tous les sujets.

S'il y a des amateurs de jeux vidéos, du point de vue de la programmation, le jeu c'est de programmer le jeu. La résolution d'énigmes est véritablement passionnante et plus le niveau monte plus c'est difficile et intéressant. L'ordinateur est un robot dont vous êtes le cerveau. Vous devez entrer "dans la peau" de la machine pour lui faire faire ce que vous voulez. Cela suppose :

- De comprendre la machine.
- D'avoir des idées, de la volonté et des objectifs à lui faire atteindre.

- De pouvoir traduire vos souhaits en des termes exécutables par la machine via une réflexion méthodique, la maîtrise d'un langage de programmation, l'écriture du programme qui inclut différentes étapes de conception.

L'informaticien qui écrit des programmes ne se réduit pas aux techniques qu'il maîtrise. Il fait des propositions dans les domaines et les situations qu'il aborde et la technique seule ne suffit pas aux réalisations. Il faut aussi être capable de s'ouvrir à d'autres domaines, d'élaborer des connaissances à la manière d'un consultant ou d'un chercheur, et d'avoir même de l'inspiration, ce qui renvoie nécessairement à une culture ouverte, si possible vivante.