Proposition de curriculum pour les filières d'économie, de gestion et d'administration

Yannick Brilhault

PROPOSITION DE CURRICULUM POUR LES FILIÈRES D'ÉCONOMIE, DE GESTION ET D'ADMINISTRATION.

Yannick BRILHAULT

INTRODUCTION

Si l'apprentissage d'un SGBD ou d'un outil intégré dans le curriculum des étudiants de **filières économiques de gestion et d'administration** recueille un large consensus, par contre la question de savoir si l'on doit leur apprendre les concepts de base de la programmation reste ouverte. Nous allons examiner trois aspects de l'apprentissage des concepts de base de la programmation (la liste de ces concepts est donnée en 5.1) qui nous semblent déterminants pour ces filières :

- la motivation de cette population ;
- les caractéristiques de cette population ;
- la spécificité de l'informatique.

Nous proposerons et nous évaluerons ensuite un curriculum qui tient compte de ces trois aspects et qui maintient l'apprentissage de la programmation dans ces filières.

Enfin nous examinerons dans le détail un point particulier de ce curriculum à savoir l'ordre d'apprentissage entre les fichiers et les tableaux. Nous indiquerons les avantages que peut procurer l'inversion de cet ordre d'apprentissage pour des filières scientifiques.

1. MOTIVATION POUR L'APPRENTISSAGE DES BASES DE LA PROGRAMMATION

Nous avions remarqué, après plusieurs années d'enseignement, le désintérêt et l'opposition des étudiants de filières économiques et de gestion pour l'apprentissage des bases de la programmation. Cette constatation est confirmée pour des élèves de sections économiques en Belgique par (Michaux 90), ainsi que par (Palombo 88) en France pour des étudiants en aménagement et urbanisme. De même, la sous-représentation des élèves de terminales B (économie) et G (gestion et économie) dans l'option informatique (6% de B et 3.5 % de G) confirme ce désintérêt.

Enfin ce désintérêt est la deuxième raison invoquée par les lycéens pour l'abandon de l'option d'informatique (Baron 89), la première étant le surcroît de travail.

L'importance déterminante de la motivation sur les résultats d'un apprentissage, nous a amené à consacrer une partie de l'introduction au cours d'informatique en **A.E.S.** (**Administration Économique et Sociale**), à expliquer les raisons qui font que cet apprentissage est nécessaire.

Dans cette introduction, nous avions développé les points suivants :

- les bases de la programmation sont indispensables pour comprendre les concepts de base de l'informatique. Le but est moins de savoir programmer dans un langage particulier, que d'avoir un support qui permette d'illustrer et de mettre en oeuvre les concepts de base ;
- les bases de la programmation facilitent l'apprentissage et/ou l'utilisation des SGBD (logiciels parmi les plus utilisés par les gestionnaires) ;
- les bases de la programmation leur permettront d'être des interlocuteurs plus avisés et plus pertinents, des professionnels de l'informatique au cours de leur carrière professionnelle. (« on ne leur racontera pas n'importe quoi ») ;
- les bases de la programmation procurent un surcroît d'autonomie, d'efficacité et de compétence dans l'utilisation de tous les logiciels qui comportent une « partie programmable ».

Afin de vérifier l'assimilation de ces différents points, nous avions posé la question suivante en devoir surveillé : « Quelles sont les raisons qui justifient l'apprentissage des bases de la programmation dans votre formation en A.E.S. ? »

L'analyse des réponses a montré que le message n'était pas passé puisque 65% des étudiants n'ont pas pu donner une seule des raisons développées en cours, les autres n'ont donné pour la plupart qu'une seule raison. De plus, une discussion après la correction nous a montré qu'il y avait pour beaucoup d'entre eux une opposition nette à cet apprentissage. Nous pensons que les raisons de cette opposition viennent en partie du préjugé sur la programmation qui est perçue comme :

- difficile (enseignée par des mathématiciens...),

- trop rigoureuse (syntaxe...),

- inutile (« il y a des logiciels tout faits, on n'apprend pas l'électronique pour se servir d'un téléphone, on n'est pas destiné à programmer...»).

« Quel esprit sensé dépensera-t-il de l'énergie pour apprendre quelque chose de difficile et d'inutile ! » Et pourtant, devant l'omniprésence de l'informatique dans la société, les étudiants ont conscience de la nécessité de “maîtriser l'outil”, « apprenez-nous donc les outils ! »

2. CARACTERISTIQUES DE LA POPULATION

Au cours de ces années d'enseignement nous avons souvent été surpris des erreurs de « bon sens » des étudiants. Afin d'illustrer notre propos, nous citons ci-après le texte d'un exercice donné en devoir surveillé en AES, dans le but de donner quelques points aux élèves faibles mais néanmoins capables d'élaborer un programme simple (après 15 HEURES d'informatique).

EXERCICE
Écrire un programme qui calcule les factures d'eau des abonnés. Le prix de l'abonnement est de 120 F par mois.
Les 30 premiers m^3 sont facturés au tarif social de 1,50 F par m^3, les m^3 suivants sont facturés à 2,70 F.

Nous fûmes surpris de constater que 70 % des étudiants s'étaient trompés. On distingue pour cet exercice deux catégories d'erreur :

- les erreurs qui proviennent de l'informatique (erreurs d'informatique) ;
- les autres erreurs.

Une étude détaillée des copies inexactes montre que :

39 % proviennent des erreurs autre que d'informatique,

11 % proviennent d'erreurs d'informatique,

50 % proviennent en partie des deux types d'erreurs.

Ce qui veut dire que 89% des échecs proviennent pour une grande part de ces autres erreurs.

L'interprétation de ces résultats est délicate, ces « autres erreurs » doivent-elle être imputées à :

1 - des aptitudes logico-mathématiques insuffisantes ?

2 - de l'inattention ?

3 - du stress relatif aux conditions de contrôle ?

4 - d'une mauvaise compréhension des énoncés ?

5 - autres ?

Nous avons fait une évaluation avec un exercice équivalent en Sciences économiques, nous avons noté des résultats comparables. Afin d'aider à l'interprétation de ce résultat, nous avons donné l'exercice précédent dans les mêmes conditions d'examen à un groupe de 43 étudiants de filières scientifiques (physique-chimie). Le résultat est significatif, nous n'avons recueilli qu'une seule erreur sur 43 copies. Ce résultat permet d'éliminer les hypothèses (2) et (3).

Nous allons donc supposer que les étudiants des filières d'A.E.S. et d'économie présentent une faiblesse dans leurs aptitudes logico-mathématiques et des difficultés de compréhension des énoncés, ces deux facteurs étant d'ailleurs sûrement liés.

3. SPECIFICITE DE L'INFORMATIQUE

Une évaluation quantitative en A.E.S. montre que le début de l'apprentissage de la programmation est déterminant pour la suite. En effet les étudiants ont un premier contrôle entre 6 et 10 semaines après le début du cours et un deuxième contrôle à la fin du semestre. Nous avons étudié le devenir des étudiants en

situation d'échec au premier contrôle. Sur 72 étudiants que nous avons estimés être en situation d'échec au premier contrôle, 58 étudiants étaient encore en situation d'échec au deuxième contrôle.

La même étude en PC1 (Physique-Chimie 1ère année) et SPI1 (Sciences Physiques Pour l'Ingénieur 1ère année) a été réalisée. Ce qui nous fait dire que 80 % des « lâchés » (93 % pour SPI1 et PC1 confondus) ne rattrapent pas le retard initial.

Considérons donc qu'il y a un barrage dès le début de l'apprentissage de la programmation. Cet aspect particulier de l'informatique a été relevé par d'autres auteurs : (Gueraud et Peyrin 88) ainsi que (Deveau § al. 88).

4. ECHEC RELATIF DE L'APPRENTISSAGE DE LA PROGRAMMATION POUR CETTE POPULATION

Après plusieurs évaluations annuelles en A.E.S. et en Sciences Economiques nous avons essayé de tirer un bilan quantitatif de cet enseignement. Nous le résumons de la façon suivante :

A peu près 10% des étudiants ont assimilé les concepts de base de la programmation (ce chiffre est assez stable).

Entre 10 et 30 % des étudiants ont assimilé quelques concepts (pour ceux-là le résultat est globalement positif).

Entre 60 et 80 % des étudiants sont incapables de concevoir un programme simple (comme par exemple la question 1 du devoir donné au § 7). Pour ceux-là, le résultat est vraiment négatif dans la mesure où ils sortent dégoûtés de la matière.

Ces mauvais résultats sont vraisemblablement une des raisons principales qui conduisent à une diminution et souvent à la disparition de l'apprentissage de la programmation dans le curriculum des filières économie, gestion, et administration.

A l'examen cet échec nous semble néanmoins logique, car nous avons la conjonction d'une matière dont le début de l'apprentissage est difficile avec des étudiants :

- **qui ont des aptitudes logico-mathématiques en retrait par rapport à des étudiants de filières scientifiques,**
- **qui ont des difficultés de compréhension des énoncés,**
- qui sont peu motivés.

Nous allons donc proposer un curriculum qui tient compte de ces trois facteurs et qui maintient l'apprentissage des bases de la programmation.

5. PRISE EN COMPTE DE LA POPULATION DANS L'ELABORATION DU CURRICULUM

Nous allons d'abord décrire le curriculum et nous expliquerons ensuite comment nous avons pris en compte les trois facteurs cités au paragraphe précédent dans l'élaboration du curriculum.

5.1 Définition du curriculum

PARTIE 1 *: (10 h cours + 8 h TD)*

Présentation des concepts fondamentaux de la programmation (variables, constantes, séquentialité, affectation, E/S simples, alternative, répétition, types simples (entier, réel, chaîne), accumulateur, compteur, dialogue homme-machine) à l'aide d'exemples et d'exercices simples ***porteurs de sens (voir 5.2).***

PARTIE 2 *(8 h Cours + 9 h TD)*

*Présentation des enregistrements et des fichiers. Le renforcement des concepts de la première partie est effectué à l'aide d'exercices simples de manipulation de fichiers : création, consultation, mise à jour, traitements simples (comme par exemple le*s traitements demandés dans le sujet de l'examen au §7).

PARTIE 3 *(7 h cours + 12 TD)*

Analyse et Conception des Systèmes d'information.

Les procédures.

En Travaux dirigés, on assistera les étudiants dans l'étude et la réalisation d'une petite application de gestion (par exemple la gestion des adhérents et des livres d'une bibliothèque de prêt. Deux séances de T.D sont réservées à l'étude préliminaire dans laquelle l'enseignant joue le rôle du responsable de la bibliothèque qui n'a aucune connaissance en informatique et qui devant les limites de son organisation actuelle envisage une informatisation).

Les tableaux.

Il faut ajouter à ce curriculum 13 h de cours d'informatique générale que nous ne détaillerons pas ici. Ces heures se trouvent disséminées surtout dans la partie 1 et concernent entre autre la structure et le fonctionnement des ordinateurs, le codage de l'information, les langages, les logiciels systèmes et d'application, les fichiers.

5.2 Prise en compte des caractéristiques de la population

- *Capacités logico-mathématiques*

La prise de conscience de leurs aptitudes logico-mathématiques nous a obligé à réexaminer le choix de nombreux exercices dans les séquences

d'apprentissage. En effet si l'on distingue comme (Hoc 82) deux situations d'apprentissage :

- **le sujet dispose déjà d'une procédure exécutable : en programmant il ne met en oeuvre qu'une activité d'expression,**
- **le sujet ne dispose pas encore d'une procédure exécutable : il lui faut donc l'élaborer et l'exprimer.**

Il est certain que nous devions souvent croire que la majorité des étudiants se trouvait dans la première situation alors qu'en fait elle se trouvait dans la seconde. Nous avons donc soigneusement choisi les exemples et les exercices de façon à ce qu'ils mobilisent le moins possible leurs capacités logico-mathématiques. En particulier, nous avons essayé de graduer les difficultés car pendant l'apprentissage de la programmation, « les étudiants sont assaillis pendant une courte période par une quantité d'informations fondamentales et de techniques nouvelles et il leur est demandé en plus de réfléchir et de comprendre » (Deveau § al 88)

- *Compréhension des énoncés*

Nous avons essayé de rédiger les énoncés de façon à ce qu'aucune ambiguïté ne soit possible malgré les lourdeurs et les redondances que cela entraîne souvent.

- *Gestion de la motivation*

Etant donné le peu d'effet de l'exhortation présentée au paragraphe 1, nous allons indiquer comment nous avons essayé de susciter cette motivation au cours des trois parties du curriculum.

Partie 1

Dans cette partie nous avons choisi des **exercices simples** de manière à redonner confiance à ces étudiants qui ont pour la plupart une expérience plutôt négative des mathématiques (« car comme les mathématiques l'informatique est perçue comme une matière qui requiert un esprit logique, et puisqu'ils n'ont pas été très bons en mathématiques ils doutent de leur capacités logiques »).Dans cette optique nous avons prévu du temps supplémentaire, afin que l'étudiant, après avoir réussi un exercice, puisse l'utiliser avec différents jeux de données, le peaufiner, l'améliorer... Les satisfactions (ou gratifications) ainsi recueillies serviront à augmenter sa motivation et sa confiance.

Nous avons choisi des **exercices porteurs de sens**. Afin d'illustrer notre propos, examinons un exercice très formateur que l'on trouve fréquemment dans les scénarios d'apprentissage sous cette forme (ou sous une forme voisine) :
Ecrire un programme qui calcule le maximum de 3 nombres.

Pour l'avoir proposé et en avoir discuté avec les étudiants, ils se demandent quel sens peut avoir cet exercice « Je rentre 3 nombres et l'ordinateur me dit quel est le plus grand !!! » Ce manque de sens les trouble, et est un frein à la résolution de l'exercice. Or cet exercice peut être transformé de manière à ce qu'il ait un sens immédiatement perceptible par l'étudiant tout en gardant sa qualité formative :

Ecrire un programme qui affiche par ordre décroissant les pourcentages de voix obtenues par chacun des 3 candidats à une élection (les noms des candidats et les nombres de voix obtenues seront entrés au clavier).

En outre, le sens contenu dans les exercices permet de cadrer la tâche à exécuter, et cela à un stade de son apprentissage ou l'apprenant manque sérieusement de repères.

Partie 2

On remarquera que la partie 1 est très courte, ce qui nous permet de présenter rapidement les enregistrements et les fichiers.

Cette introduction très précoce des fichiers permet de traiter des petits problèmes de gestion qui se trouvent dans leur domaine d'étude, et ils imaginent eux même le bénéfice qu'ils pourront en retirer dans leur vie professionnelle (par exemple dans la question 3 du sujet donné au §7, l'étudiant se met en situation de décideur).

Dans une gestion de stock ou dans le sujet de l'examen, l'intérêt est immédiatement perceptible par l'étudiant.

De plus, l'utilisation précoce des fichiers fournit une grande variété de problèmes significatifs de complexité variable, cela est très utile dans l'enseignement des méthodes, car il est facile de trouver des problèmes qui s'énoncent simplement et pour lesquels l'absence de méthode empêche l'apprenant de trouver la solution.

Partie 3

La dynamique et la motivation induites par le projet dans un apprentissage nous semble suffisamment établies, en particulier par (Not 90). Nous en avons eu confirmation par l'assiduité manifestée devant les micro-ordinateurs pendant les heures de libre-service.

La nécessité de présenter les tableaux a été réalisée en « situation-problème » (Meyrieu 88) : lorsque nous avons proposé un jeu d'essai conséquent qui condamne la recherche séquentielle des enregistrements. Nous avons donc présenté le concept d'« accès direct avec tableau d'index ». Cela lui permettra d'utiliser l'indexation dans un SGBD en sachant ce qui se passe au niveau logiciel sous-jacent.

7. EVALUATION

Nous avons effectué 3 évaluations de cette approche par deux enseignants différents (2 promotions A.E.S., une promotion de Sciences Economiques, ce qui représente 306 étudiants). Nous allons détailler la première qui porte sur une promotion d' A.E.S. de 105 étudiants. Les deux autres évaluations ont été faites suivant le même principe avec des tests isomorphes, les résultats ont été reportés dans le tableau 1. Nous donnions dans l'énoncé le programme de création des

clients d'une société avec pour chaque client un enregistrement comportant le nom, l'adresse, la ville, le nom du représentant qui s'occupe de ce client et le montant des ventes déjà effectuées avec ce client. Nous posions ensuite les questions suivantes :

Question 1 :

Ecrire le programme qui donne la liste des clients d'une ville donnée.

Question 2 :

Ecrire l'algorithme qui donne le total des ventes effectuées par un représentant (un représentant s'occupe en général de plusieurs sociétés).

Question 3 :

Ecrire le programme qui informe tous les clients, dont le total des ventes déjà effectuées dépasse 500 000 Francs, qu'ils auront droit à une remise de 3% sur leurs prochains achats.

Nous avons distingué trois catégories : **TP** (très positif), **N** (négatif) et **GP** (globalement positif).

31 étudiants ont répondu correctement à la question 2 ainsi que à la question 1 et/ou la question 3. On vérifie que dans ce cas, tous les concepts présentés dans les deux premières parties du curriculum sont mobilisés (exceptée la notion de compteur). **TP**

27 étudiants n'ont répondu correctement à aucune question. **N**

Nous avons rangé les 47 étudiants qui restent dans la catégorie intermédiaire. **GP**

T*	AES(1) 40 heures	%	AES(2) 45 heures	%	SC-ECO 40 heures	%	TOTAUX 3 Promotions	%	Résultats antérieurs
TP	31	29	29	18	13	29	73	**24**	environ 10%
GP	47	45	85	54	12	27	144	**47**	entre 10% et 30%
N	27	26	43	27	19	43	89	**29**	entre 60% et 80%

T* : temps consacré à l'apprentissage des bases de la programmation

La comparaison des deux dernières colonnes montre que ces résultats sont très supérieurs aux résultats antérieurs. Cela nous a conduit à persévérer dans cette voie et à tester une partie de ce curriculum dans des filières scientifiques qui ne conduisent pas à une spécialisation en informatique. En effet, dans ces filières scientifiques, on note aussi une proportion importante d'étudiants qui sont très faibles en informatique. Nous pensons que l'inversion de l'ordre d'apprentissage entre les tableaux et les fichiers peut améliorer cette situation.

8. TABLEAUX ... FICHIERS OU BIEN FICHIERS ... TABLEAUX

Dans un curriculum classique d'apprentissage de la programmation on note l'ordre suivant :

C1

exercices mettant

..... itération ..... tableaux .....en oeuvre itérations ..... i .... fichiers

et tableaux

Nous avons noté en T.D la difficulté qu'éprouvent beaucoup d'étudiants de disciplines scientifiques (Physique, Chimie) dans la résolution de ces exercices. Cette difficulté s'explique à notre avis par trois raisons :

- ils se trouvent à un stade de leur apprentissage où ils manquent de repères,
- l'itération est un concept difficile comme le remarque aussi (Rogalski 88) ainsi que (Dagdilesis § al 90),
- la difficulté supplémentaire qui résulte de la désignation des éléments du tableau.

Nous pensons que ces difficultés simultanées sont à l'origine du barrage mis en évidence au paragraphe 3 et quelles sont insurmontables pour beaucoup d'étudiants de filières non scientifiques. C'est pourquoi dans le curriculum (proposé précédemment en 5.1), on remarquera la séquence **C2 :**

C2

.... itération ..... fichiers .....exercices mettant en oeuvre itérations et fichiers ..... i .... .. tableaux

que nous avons préférée, car plus facile et plus adaptée.

Nous avons donc aussi expérimenté **C2** avec un groupe de 79 étudiants de PC1 (Physique-Chimie 1ère année), et nous avons pris un groupe témoin de 94 étudiants de SPI1 (Sciences Physiques Pour l'Ingénieur 1ère année) qui a eu la séquence **C1.**

Nous avons effectué une évaluation dans les deux groupes, au point **i**, avec un problème équivalent (dans un cas les données sont dans un fichier et dans l'autre cas elles sont entrées dans un tableau).Les premiers résultats sont meilleurs dans le groupe expérimental (le test du chi-2 permet d'affirmer ce fait avec un risque d'erreur inférieur à 6%)

9. CONCLUSION

Nous ne résistons pas à l'envie de transposer une citation du pédagogue (A. Giordan 78) « **Cette expérience montre que *les étudiants de filières économie, gestion ou administration* peuvent apprendre les concepts de base de la *programmation*, s'ils sont abordés selon leurs cadres de référence et leurs motivations et si l'enseignant n'a pas un souci de rigueur incompatible avec le niveau de pensée de l'étudiant.** »

A un moment où les orientations de l'étudiant sont déjà « choisies » et pour rester fidèle à ce principe, il est nécessaire de différencier l'enseignement de l'informatique suivant trois grandes catégories :

- filières non scientifiques ;
- filières scientifiques (qui ne débouchent pas naturellement vers l'informatique) ;
- filières informatiques.

Pour ces deux premières populations il nous semble que l'apprentissage précoce des fichiers d'enregistrements permet une progression plus douce qui évite de laisser une proportion trop importante d'étudiants déçus de l'informatique.

Yannick BRILHAULT

LICIAP - Département d'informatique
Université de Pau et des pays de l'Adour
Avenue de l'université
64000 PAU - FRANCE

BIBLIOGRAPHIE

BARON G.L (1989) *L'informatique discipline scolaire ? Le cas des lycées*, PUF.

DAGDILESIS V., BALACHEFF N., CAPONNI B. (1990) « L'apprentissage de l'itération dans deux environnements informatiques », ASTER N°11, *Informatique, regards didactiques*, INRP, p. 45-66.

DEVEAUX D., RAPHALEN M., REVAULT J. (1988) « Spécification d'un interpréteur de mini-langage algorithmique pour l'initiation à la programmation », *Actes du 1er Colloque Francophone sur la didactique de l'informatique*, EPI, Paris, p. 87-101.

GIORDAN A. (1978). Une pédagogie pour les sciences expérimentales PAIDO-GUIDES.

GUERAUD V., PEYRIN J.M. (1988) « Un jeu de rôles pour l'enseignement de la programmation », *Actes du 1er Colloque Francophone sur la didactique de l'informatique*, EPI, Paris, p. 48-59.

HOC J.M. (1982) « La programmation informatique à l'école ? Les exigences de cette activité », *4ème journées sur l'éducation scientifique*, Chamonix, p. 123-133

MEYRIEU P. (1987) *Apprendre oui...mais comment*, ESF ,Paris.

MICHAUX M.P. (1990). « Quelle informatique enseigner dans les diverses sections de l'enseignement général », *Actes du 2éme Colloque Francophone sur la didactique de l'informatique*, Namur

NOT L. (1989) *L'enseignement répondant*, PUF.

PALOMBO N. (1988) "Une expérience d'enseignement de l'informatique en aménagement", *Journées SPECIF 1988*, Besançon (Editeur Michel Lucas),p. 66-70.

ROGALSKY J. (1988) « Enseignement de méthodes de programmation dans l'initiation à l'informatique », *Actes du 1er Colloque Francophone sur la didactique de l'informatique*, EPI, Paris, p. 61-74.