MPSI
PCSI
PTSI

INFORMATIQUE

1 DEVOIR CORRIGÉ PAR SEMAINE

avec corrigés commentés

Olivier Arrigoni

ellipses

MPSI
PCSI
PTSI

INFORMATIQUE

1 DEVOIR CORRIGÉ PAR SEMAINE

Olivier ARRIGONI
Professeur en ECG1
au lycée Camille Vernet (Valence)

ellipses

Conception graphique couverture : Nathalie FOULLOY

ISBN 9782340-115583

Dépôt légal : juin 2026

www.editions-ellipses.fr

Avant-propos

Aux concours des écoles d'ingénieurs (E3A-E4A, CCINP, Centrale, Mines-Ponts, X-ENS) l'informatique est généralement évaluée dans une épreuve à part entière sur le programme des trois semestres prévus au programme de l'informatique pour tous (IPT) des CPGE scientifiques (deux en première année, un seul en deuxième année). Ce programme est commun aux filières MPSI, PCSI, PTSI, pour les premières années et MP, PC, PSI, PT en deuxième année.

Il est également possible qu'il fasse l'objet de questions ou même d'exercices entiers dans les épreuves de mathématiques, physique, SII ou encore chimie. Même si la nature de l'interrogation lors d'une de ces épreuves citées est différente d'une épreuve d'informatique IPT pure, la compréhension de la programmation, de la syntaxe et même de la démarche est incontournable. Ainsi, l'informatique est désormais omniprésente dans l'évaluation des concours. De plus, ce type de questions est souvent valorisé dans les épreuves et elles permettent alors de compenser une éventuelle faiblesse sur un des points des programmes des matières fondamentales.

Cet ouvrage ne s'intéresse qu'au programme de la première année, qui constitue tout de même les deux tiers du programme complet d'IPT. On constate dans tous les concours une part importante de ce programme qui est directement évaluée, sans faire nécessairement appel au programme du troisième semestre (deuxième année). On s'intéresse à la maitrise du programme de première année et aux méthodes qui font des écrits réussis.

Citons également des épreuves orales (comme au concours Centrale) qui mêlent mathématiques et évaluation des compétences de mise en œuvre directe sur ordinateur, dépassant encore la syntaxe pure évaluée aux écrits.

Dans ce sens, on propose quelquefois des applications numériques à mettre en œuvre sur l'ordinateur.

Pour mettre en avant la régularité, on présente donc 30 problèmes d'écrits pour que chacun occupe environ une semaine de la première année de CPGE scientifique, qui est une année bien remplie ! On peut considérer que deux problèmes constituent une bonne épreuve de deux heures de seconde année.

On en vient à l'organisation concrète de l'ouvrage.

- On propose tout d'abord un chapitre d'introduction sur les méthodes de travail et comment *apprendre à apprendre*. En particulier, on insiste sur le travail à réaliser tout au long de l'année avec le travail de problèmes d'écrit.
- La partie suivante présente le programme officiel de l'IPT pour la première année des CPGE scientifiques avec une correspondance entre chaque notion de ce programme et un ou plusieurs problème(s) de cet ouvrage.
- Les énoncés de chaque problème sont présentés avec un titre, un thème général, un ensemble de mots clé à la fin et un niveau de difficulté qui se décline en trois catégories :

  aucune étoile : **Facile** $\quad\star$: **Moyen** $\quad\star\star$: **Difficile**

- Les corrigés sont soigneusement rédigés, aux détails près et sont parsemés de conseils, méthodes, remarques en encadrés pour que son utilisation aille bien au-delà de la simple correction dudit problème.

Le travail régulier et approfondi, tant sur les automatismes que sur une bonne rédaction d'une copie, est un travail qui paye vraiment. Un candidat qui aura réellement fait cet effort se démarquera des autres à coup sûr.

Bon travail à tous !

Introduction aux méthodes de travail

On propose dans cette introduction des méthodes pour mener à bien un travail efficace en CPGE. On introduit ensuite quelques conseils pour bien utiliser l'ouvrage de problèmes qui suit.

1) Apprendre à apprendre

a) Pourquoi apprendre à apprendre ?

L'entrée en classe préparatoire représente une transition majeure : la quantité et la technicité des notions à maîtriser s'accroissent et l'autonomie devient indispensable. De plus, les étudiants arrivent à la rentrée avec des niveaux très hétérogènes, notamment à cause de la diversité des parcours en Première et Terminale avec les choix de spécialités. L'enseignant, s'il en tient bien sûr compte, doit pour autant mener un programme dense avec un temps limité.

Réussir dans ces classes ne consiste pas à suivre le rythme, mais à se l'approprier. Travailler efficacement, ce n'est pas travailler plus (même si le temps passé augmentera !), c'est mieux comprendre ce qu'on fait, pourquoi on le fait, et comment on le fait.

En somme, pour avoir un apprentissage efficace en CPGE, il faut aller au bout de son cours, travailler l'articulation de celui-ci, la suite logique de son exposition. Pour ce qui est de l'informatique, la pratique est une pièce maitresse de la compréhension et il faut aussi faire ce travail sur les travaux pratiques (TP). Tout ce qui y sera porté n'est pas superflu, même si certaines parties sont plus importantes que d'autres. Un apprentissage qui se limiterait aux seules formules, aux programmes standards en Python ou à la seule répétition d'exercices types, même si c'est un bon début, ne peut

pas permettre de bien réussir.

b) Les piliers d'un travail efficace en informatique

Il faut comprendre plutôt que de mémoriser. La tentation est grande de vouloir apprendre les cours par cœur. Mais en informatique, la compréhension profonde des notions est la meilleure alliée : un principe de programmation compris est plus facile à retrouver, un programme type bien saisi, pour telle ou telle tâche, se retrouve presque de lui-même. A ce titre, on insiste sur le fait que les programmes type faits en classe présentent un intérêt pédagogique majeur, dans le sens où ils peuvent constituer des modèles sur d'autres programmations. Il ne faut d'ailleurs pas oublier son bon sens, même sur des programmes compliqués.
Un conseil lors de ses révisions est de réexpliquer à haute voix une notion ou un programme comme si l'on était son propre professeur. Ce test simple révèle vite ce qui est vraiment acquis et ce qui mérite une attention plus particulière.
On ne le répètera jamais assez, il faut de la régularité dans son travail, au sens de la répétition des concepts. "Enseigner, c'est répéter", disait-on autrefois. C'est toujours d'actualité ! Un apprentissage *en spirale* est celui qui est à terme le plus efficace, c'est-à-dire celui qui revient le plus régulièrement possible sur les notions et les programmes.
Il faut avoir en tête que le travail en informatique est cumulatif. Il repose sur des bases solides, qui se consolident en avançant et en réinvestissant toujours les apprentissages passés. Travailler régulièrement, c'est éviter l'oubli, mais aussi gagner en profondeur à chaque retour sur une notion.
On ne peut que conseiller de prévoir chaque semaine des temps de reprise : relire ses erreurs, revoir un cours précédent, refaire une partie de TP. Il ne faut surtout pas laisser une séance de côté en se disant qu'on y reviendra plus tard sans avoir fait au moins un retour.

c) L'actif contre le passif : on apprend en faisant

En premier lieu, c'est le cours que l'on doit travailler activement. Lorsqu'on le révise, on doit avoir un crayon à la main pour ajouter une remarque, noter une question à poser, terminer un exemple laissé au lecteur. On se doit par la suite d'en faire une synthèse, sous la forme qui convient à soi-même (fiche, carte mentale...).
De la même façon, lire un corrigé, ce n'est pas faire un exercice. Le cerveau n'apprend que lorsqu'il tente, échoue, ajuste, réussit. Il y a en général peu de chance que l'on soit capable de reproduire ce que l'on n'a pas fait soi-même.

On conseille fortement de toujours commencer par chercher sans regarder la solution. Même un échec total est formateur s'il est analysé ensuite, notamment à l'aide de la correction. Il faut alors voir ce qui n'est manifestement pas connu et mettre les moyens pour le retravailler efficacement.
Si la peur de la "feuille blanche" est trop intense, on peut alors s'autoriser un peu d'aide. Il est tout à fait possible d'aller voir une partie du corrigé d'un problème pour se redonner les moyens de poursuivre.
Enfin, il faut appréhender ses erreurs. L'erreur n'est pas un obstacle, c'est en fait la motivation de l'apprentissage. Une erreur analysée devient une ressource précieuse. Ce qui importe, ce n'est pas de "ne pas se tromper", mais de savoir pourquoi on s'est trompé. C'est d'ailleurs la meilleure façon de ne pas reproduire une telle erreur. Il faut bien avoir en tête que d'autres erreurs guetteront au fur et à mesure que l'année avance, alors faire en sorte que les premières ne reviennent pas est un enjeu majeur. Il n'y a rien de pire que de voir s'accumuler des difficultés, qui mèneraient inéluctablement au découragement.
On peut imaginer de tenir un carnet des erreurs. Une erreur notée, comprise et classée est une erreur qui ne revient plus.
On rappelle qu'un devoir en CPGE ne présente pas d'enjeu de réussite impérative comme lors d'un examen. Chaque énoncé doit être vu comme un outil de formation, quelle que soit la réussite qui en découle à la première tentative.

2) Comment travailler avec cet ouvrage ?

a) Construction de l'ouvrage

Ce livre est pensé comme un compagnon de progression. Le fait qu'il prévoit de traiter un problème par semaine montre une volonté d'accompagner et consolider les connaissances tout au long du parcours.
Même si la progression qui est prévue n'est pas celle que l'on trouvera dans sa classe avec son enseignant, les problèmes sont classés par ordre croissant d'outils informatiques nécessaires pour être traités. On commence donc par des problèmes de programmation simple, sur des sujets souvent mathématiques connus de tous, qui ne nécessitent que des prérequis très élémentaires de programmation en langage Python pour être abordés. Ensuite, on rentre de plus en plus dans le programme d'informatique pour tous en CPGE scientifiques, notamment dans les programmes de référence et les grands thèmes à ce programme.
La division selon les deux semestres est déjà un point de repère. Les 18 premiers problèmes sont très axés sur un travail de premier semestre. On

notera que certains d'entre eux présentent déjà quelques notions de second semestre, comme la complexité et la terminaison des algorithmes, au moins du point de vue expérimental. Ces deux thèmes sont transversaux sur toute étude en informatique. Plus précisément, ils sont en fait l'une des motivations principales de la discipline : savoir qu'un algorithme est bien correct et surtout quel "temps" il prend à être exécuté.

b) Sur l'utilisation de chaque problème

Chaque problème a sa difficulté propre. Mais ce que l'on cherche à avoir en commun, c'est un enchainement des questions qui le différencie d'un simple exercice de TD ou TP. Certains problèmes peuvent même comporter plusieurs parties plus ou moins indépendantes.
Étant donné les longueurs choisies pour chaque, on peut considérer que deux problèmes parmi tous ceux proposés constituent une épreuve classique de 2 heures au concours. Le rythme étant supposé s'intensifier sur les deux années, on peut considérer qu'une heure et demi est une bonne durée pour la rédaction d'un problème en classe de première année. Au début, le rythme sera certainement difficile à tenir, mais l'entrainement régulier permettra d'aller plus vite. On rappelle aussi que si la quantité traitée lors du concours est bien sûr importante, mieux vaut avoir traité moins de parties mais de meilleure qualité ; ce sera alors valorisé. Avec la correction et toutes les aides qui y sont données, on vise justement une bonne qualité de rédaction et de programmation.
Lorsque l'on traite un problème, il est fortement recommandé de lire l'intégralité du sujet (sur 5 à 10 minutes par exemple). Lors de cette lecture, on doit repérer le mieux possible les notions du cours qui sont mobilisées, les programmes type, mais aussi les questions de cours qui seraient "déguisées" par des demandes de rappels. Même lors d'une épreuve chronométrée, ce n'est pas du temps de perdu.

c) Il faut aimer chercher...

Le goût de l'effort intellectuel est un plaisir qui se cultive. Il ne va pas toujours de soi, et il est vrai qu'on peut vite être découragé car on ne réussit pas aussi bien qu'on le pense, notamment vis-à-vis de l'investissement que l'on y met. Mais le moment où l'on trouve enfin la clé d'un problème est une récompense profonde. Ce plaisir est au cœur de la réussite dans toute discipline où l'on résout des problèmes.

Il ne faut jamais oublier que l'élève qui réussit, ce n'est pas toujours celui qui comprend tout du premier coup. C'est celui qui n'abandonne pas la recherche et qui analyse ses démarches. On ne répètera jamais assez

l'adage suivant : quand on trouve sans chercher, c'est qu'on aura longtemps cherché sans trouver.

Sommaire

Devoirs

Corrigés

Index thématique du programme

Enseignement d'informatique du 1er semestre

Programme officiel	Problèmes correspondants
Recherche séquentielle dans un tableau (liste et dictionnaire). Recherche du maximum, du second maximum. Comptage des éléments.	1 2 3 4 5 12
Algorithmes opérant sur une structure séquentielle par boucles imbriquées.	6 13
Utilisation de modules et bibliothèques.	**Répartie dans tous les problèmes**
Algorithmes dichotomiques : recherche dans un tableau trié. Exponentiation rapide.	7 8 9
Fonctions récursives.	9 10 11 15
Algorithmes gloutons : rendu de monnaie, allocation de salle, sélection d'activité...	16 17 18
Matrices de pixels et images.	6
Tris : par insertion, par sélection (quadratiques). Tri par partition-fusion. Tri rapide. Tri par comptage.	12 13 14 15

Enseignement d'informatique du 2nd semestre

Programme officiel	Problèmes correspondants
Instruction, expression. Effet de bord. Spécifications. Annotations. Assertions. Justification des choix de programmation.	**Répartis dans tous les problèmes.**
Terminaison. Correction partielle et correction totale. Variant et invariant (notamment pour les boucles). Jeu de tests	**23** **24** **25** **26**
Complexité en temps (calcul) et espace. Meilleur des cas et pire des cas.	**23** **24** **25** **26**
Représentations des entiers positifs et signés sur des mots de taille fixe. Complément à deux. Entiers multi-précision.	**19** **20**
Distinction réel, décimaux et flottants. Représentation sur des mots de taille fixe. Notion de mantisse et d'exposants. Précision des calculs en flottants.	**21** **22**
Vocabulaires des graphes : orienté ou non, sommets (nœuds), arcs (arêtes), boucle, degré, cycle, chemin. Notations. Liste, dictionnaire et matrice d'adjacence. Pondération d'un graphe.	**27**
Connexité dans les graphes non orientés.	**28**
Parcours des graphes. Recherche d'un plus court chemin dans un graphe pondéré avec des poids positifs.	**29** **30**

DEVOIRS