4

BILAN DES COORDINATEURS DE L'ÉPREUVE
ORALE D'INFORMATIQUE

Galatée Hémery Vaglica et Samy Jaziri

L'épreuve orale consiste en la résolution d'exercices au tableau avec un temps d'appropriation du sujet de 15 minutes avant l'épreuve. Ce temps d'appropriation doit avant tout être utilisé par les candidats pour prendre connaissance des annexes et de l'ensemble du sujet. Les sujets étaient composés de deux exercices sur des parties différentes du programme de *première* et de *deuxième* année. Le premier exercice propose souvent une application directe du cours et le second la résolution guidée d'un problème. Les différents sujets sont de difficulté et de longueur variable et la notation est ajustée selon la difficulté des exercices.

En grande majorité, les candidats étaient bien préparés à l'épreuve orale et ont su réagir aux remarques du jury et entrer en discussion avec lui. Le niveau moyen des candidats reste bon dans la filière MPI au Concours Mines-Télécom. Sans s'avancer outre mesure, le jury se demande tout de même s'il n'y aurait pas une dynamique à la baisse sur ces trois dernières années.

Statistiques 2025

FILIÈRE	NB CANDIDATS	MOYENNE	ECART-TYPE
MPI	414	11,05	3,779

Déroulement de l'épreuve

Le candidat se voit remettre le sujet et ses annexes 15 minutes avant le début de l'épreuve, dans une salle dédiée. Ce temps doit être mis à profit pour s'approprier les annexes, puis le sujet. Il est *fondamental* de lire l'intégralité des annexes et du sujet avant de commencer une phase optionnelle de réflexion sur les questions. À noter que les candidats ne sont pas autorisés à prendre de notes durant ce moment d'appropriation.

À l'issue de ces 15 minutes, les candidats sont conduits jusqu'à leur salle d'interrogation. Le jury s'attend, à l'entrée dans la salle, à ce que le candidat soit familier avec l'ensemble du sujet, annexes comprises. Contrairement à ce que certains ont pensé, il n'est en revanche pas attendu des candidats qu'ils aient déjà résolu certaines questions.

L'interrogation orale dure 30 minutes, incluant un temps incompressible de déplacements et formalités administratives.

En entrant dans la salle d'interrogation, le candidat remet à l'examinateur une pièce d'identité et la feuille d'émargement des examinateurs. Il est souhaitable que ces documents soient prêts à l'avance, tout temps passé à rechercher l'un d'entre eux au fond d'un sac va raccourcir le temps de l'interrogation. C'est l'examinateur qui guide l'oral et, par défaut, il est attendu que le candidat aborde les différentes questions du sujet dans l'ordre. S'il est possible d'échanger avec le jury quant aux questions sur lesquelles le candidat se sent le plus à l'aise, seul l'examinateur peut autoriser le candidat à passer une ou plusieurs questions. L'examinateur peut, tout en informant clairement le candidat, considérer que le candidat n'a pas été en capacité de résoudre une ou plusieurs questions s'il insiste pour passer à la suite.

Le jury possède une copie du sujet avec lui, il n'est pas nécessaire de relire l'intégralité de l'énoncé ou de présenter le sujet devant le jury si ce dernier a été bien compris lors de la phase d'appropriation. Si toutefois le candidat a un doute sur sa compréhension d'une partie de l'énoncé, il lui est conseillé d'en faire part au jury. S'il ne s'agit pas d'un manque de vocabulaire ou de connaissance lié au cours, le candidat ne sera pas pénalisé par cette démarche. Elle sera toujours préférable à la découverte, plus tard dans l'oral, d'un problème de compréhension.

L'interrogation orale dure 30 minutes, incluant un temps incompressible de déplacements et formalités administratives.

Pendant l'oral, les candidats ont à leur disposition un tableau entier. Il est fortement conseillé aux candidats de profiter de cet espace et de bien l'organiser. Les schémas, graphes, arbres de preuves, etc. doivent être lisibles et pour cela les candidats ne doivent pas hésiter à prendre de la place. Même si la majorité des candidats en sont conscients, le jury tient à rappeler qu'il ne faut rien effacer au tableau sans demander au préalable son aval. Enfin, il est important pour les candidats de trouver un bon équilibre entre ce qui doit être écrit au tableau et ce qu'il suffit de présenter à l'oral. Les preuves en particulier nécessitent une part de formalisation écrite et le jury se contentera rarement d'une simple description orale des grandes lignes. Ainsi, lorsque l'examinateur le demande au candidat, celui-ci doit écrire intégralement sa preuve et seul l'examinateur pourra décider de l'écourter, de se contenter de la moitié d'un dessin ou de décrire informellement à l'oral un invariant.

L'oral se termine quand le jury l'annonce au candidat. Il est alors inutile, voire malvenu, de continuer à proposer des solutions, de discuter du déroulement de l'épreuve ou de tout autre sujet personnel. Le candidat rend le sujet à l'examinateur immédiatement, efface le tableau et quitte la salle avec sa feuille d'émargement, sa carte d'identité et ses affaires.

Notation

Lors de l'épreuve orale sont évaluées, non seulement les connaissances en informatique, mais aussi le dynamisme et la capacité à interagir avec le jury, à l'écouter et à rebondir sur ses remarques et indications. Le jury déconseille fortement aux candidats de se murer dans le silence trop longtemps face à une question difficile. Tant qu'il ne fait pas d'erreur de cours ou de raisonnement grossière, le candidat ne peut que gagner à partager ses idées et ses pistes de résolution avec le jury lorsqu'il entre dans une phase de réflexion. Sur une question de cours dont le candidat aurait oublié la réponse, il est cette fois-ci préférable de répondre prudemment et ne pas essayer d'inventer la réponse, quitte à admettre son oubli. C'est au jury de déterminer la suite à donner à une telle réponse. Le jury rappelle qu'en plus des connaissances et des capacités de résolution, la prestation orale est une part non négligeable de la note finale de l'examen. Rentre en compte, en particulier, l'attitude, le vocabulaire et l'élocution. Une attitude désinvolte ou désintéressée, un vocabulaire familier ou un manque de clarté dans l'explication sont autant de facteurs qui peuvent, s'ils sont répétés ou trop marqués, pénaliser le candidat. Il en va de même si le candidat tourne systématiquement le dos au jury ou regarde son sujet pendant la majorité de l'examen oral. En particulier, un candidat qui reste dos à l'examinateur face à son tableau en cachant ce qu'il écrit est pénalisé.

À noter enfin que les sujets peuvent être de longueur et difficulté variables. Certains ne peuvent pas être terminés en 30 minutes, et ce même pour les meilleurs candidats. Il est donc tout à fait possible d'obtenir une excellente note, voire la note maximale, sans avoir traité l'intégralité des questions. À l'inverse, aller jusqu'à la dernière question du sujet ne présume rien quant à la qualité de l'oral, le jury se réservant le droit de ne pas relever certaines erreurs.

REMARQUES ET CONSEILS

Le jury tient tout d'abord à rappeler que l'épreuve orale d'informatique balaye tout le programme de MP2I et MPI. Il peut être demandé au candidat, d'une manière adaptée au format oral de l'épreuve, de présenter des algorithmes, d'étudier et d'écrire du code en C ou en OCaml, ou encore des requêtes en SQL. L'écriture de code en C et Ocaml s'en tient généralement à des questions courtes et raisonnables à traiter au tableau. Les candidats semblent avoir plus de difficultés sur les notions et algorithmes de première année. Le jury insiste sur le fait que de nombreux sujets portent, parfois exclusivement, sur le programme de MP2I.

Trop de candidats butent encore sur des termes précis introduits dans le programme, bien que la notion sous-jacente soit connue, parfois sous un autre nom. Ainsi, le jury s'attend à être compris lorsqu'il parle de "langage non contextuel", de "retour sur trace" ou de "correction" d'un système de preuve. Le jury conseille aux futurs candidats de s'appliquer à faire des démonstrations précises et rigoureuses. Les candidats doivent savoir distinguer une récurrence d'une induction et les poser clairement. Les hypothèses doivent être formulées et vérifiées dans le cas initial et pour l'hérédité. Beaucoup de candidats ont encore du mal avec les preuves et définitions par induction.

Le jury s'étonne du nombre de candidats demandant s'il était vraiment nécessaire de répondre à certaines questions. Voici quelques exemples de phrases que le jury conseille d'éviter lors de l'oral : « *On le fait vraiment, ou…*», « *L'autre branche c'est pareil, je suis obligé de la faire ?*» ou encore « *Et on continue la déterminisation comme ça, je passe à la suite ?* ».

Trop de candidats, bien que souvent vifs par ailleurs, ne sont pas capables de restituer avec exactitude un algorithme ou un théorème de cours, ni de l'appliquer ou de l'utiliser correctement. Le jury rappelle aux candidats que les preuves des théorèmes sont aussi à connaître. Entre autres exemples :

- Le lemme de l'étoile est toujours rarement restitué ou utilisé correctement.
- Il faut savoir donner la description formelle d'un automate ou d'une grammaire non contextuelle.
- Il faut savoir expliquer ce qu'est un algorithme glouton ou être capable de justifier en quoi un algorithme est glouton.

La connaissance du cours prenant une part importante dans la note, le jury encourage vivement les candidats à retravailler les algorithmes au programme et les démonstrations classiques pour cet oral.

Questions de cours

Le jury fournit une liste non exhaustive de questions de cours qui sont apparues cette année dans les exercices pour faciliter le travail de révisions des futurs candidats :

1. Définir NP-complet.
2. Donner informellement l'implémentation des opérations des tableaux associatifs et refaire les calculs de complexité.
3. Construire un arbre de Huffman pour la compression d'une chaîne de caractère, par exemple.
4. Compresser une chaîne de caractère, par exemple, avec l'algorithme de Lempel-Ziv-Welch.
5. Appliquer l'algorithme de Dijkstra sur un exemple.
6. Rappeler ce qu'est la forme normale conjonctive d'une formule logique.
7. Expliquer ce qu'est une pile et une file.
8. Rappeler la définition de mutex et de sémaphore.
9. Rappeler la définition d'un tas binaire max.
10. Appliquer l'algorithme de Kosaraju.
11. Déterminiser un automate.
12. Montrer l'indécidabilité du problème de l'arrêt.
13. Décrire un algorithme de tri efficace. Donner et justifier sa complexité temporelle.
14. Montrer que pour tout langage régulier, il existe une grammaire hors contexte qui l'engendre.
15. Comment construire un automate reconnaissant l'intersection des langages reconnus par deux automates donnés ?
16. À quel besoin répond l'algorithme ID3 ? Décrire brièvement le principe.
17. Énoncer et démontrer le lemme de l'étoile.
18. Rappeler le principe de l'algorithme des k plus proches voisins.
19. Construire le graphe d'un jeu d'accessibilité à deux joueurs.
20. Déterminer la signature d'une fonction OCaml.

Épreuve orale d'Informatique

Filière MPI, session 2026

Ce document a pour objectif de présenter le cadre de l'**épreuve orale d'Informatique** du **Concours Commun INP de la filière MPI**. Les candidats MPI doivent prendre connaissance des modalités de l'interrogation afin de se préparer au mieux à cette épreuve.

Les éléments principaux de cadrage de la filière MPI du CCINP sont disponibles sur le site du CCINP www.concours-commun-inp.fr spécialement :

- https://www.concours-commun-inp.fr/fr/epreuves/les-epreuves-ecrites.html
- https://www.concours-commun-inp.fr/fr/epreuves/les-epreuves-orales.html
- https://www.concours-commun-inp.fr/fr/epreuves/annales/annales-mpi.html
- https://www.concours-commun-inp.fr/fr/les-ecoles/les-places-dans-les-ecoles/places-mpi.html

L'épreuve orale d'informatique, d'une durée de 1 heure, est constituée de deux exercices et se déroule ainsi :

- Entre 25 et 30 min de *préparation sur table et sur ordinateur* (formalités de début d'épreuve comprises).
- Entre 25 et 30 min d'*interrogation au tableau* (formalités de fin d'épreuve comprises).

Préalablement à la *préparation sur table et sur ordinateur*, le candidat ou la candidate pourra prendre connaissance de l'exercice **type A** dans une salle surveillée durant 15 min (modalité et installation comprises). Ce temps d'analyse préalable est un temps de réflexion sans matériel (hors aménagement spécifique). Il permettra de mieux exploiter le temps de *préparation sur table et sur ordinateur*.

Lors de ce temps d'analyse (seulement) de l'exercice **type A**, aucun brouillon ne sera donc disponible et aucune prise de notes ne sera possible.

L'épreuve est constituée :

- D'un **exercice de type A (noté 8/20)** dont la préparation est sur papier.

L'exercice A ne nécessite pas l'utilisation d'un ordinateur. Lors du passage au tableau, le candidat présente ses résultats en interagissant avec l'examinateur.

Des exemples d'exercices A « zéro » sont disponibles sur le site du CCINP : https://www.concours-commun-inp.fr/fr/epreuves/les-epreuves-orales.html.

- D'un **exercice de type B (noté 12/20)** dont la préparation se fait sur un ordinateur et le cas échéant, sur papier. Quelques questions d'analyse peuvent également y être posées.

Sur l'ordinateur, le candidat dispose d'un fichier source correspondant au problème à résoudre. Le fichier peut être *blanc* (aucune instruction proposée), *partiel* (des instructions sont disponibles, le programme ne peut s'exécuter) ou *complet* (des instructions sont disponibles, le programme peut s'exécuter). Le candidat peut être amené à compléter, corriger ou modifier ce code compagnon.

Une fois le temps de préparation achevé, le fichier source du programme sujet à évaluation est à transférer à l'examinateur par un support physique de type clé USB. Lors du passage au tableau, le fichier est projeté sur support mural, tableau ou écran afin d'accompagner la présentation du candidat et l'interaction entre le candidat et l'examinateur. Plusieurs fichiers peuvent être concernés.

Des exemples d'exercices « zéro » sont disponibles sur le site du CCINP : https://www.concours-commun-inp.fr/fr/epreuves/les-epreuves-orales.html. Les codes sources en correspondance avec les exercices « zéro » sont disponibles dans un fichier compressé distinct.

Par ailleurs, une sélection de sujets des **exercices de type A et B**, commentés et partiellement corrigés, issus des sessions 2023, 2024 et 2025 est proposée dans les ANNALES MPI, https://www.concours-commun-inp.fr/fr/epreuves/annales/annales-mpi.html.

Les éléments suivants complètent les modalités :

- Le candidat peut obtenir une « note correcte » à un exercice donné même s'il ne répond pas à toutes les questions de l'exercice.

- Sur le principe des Oraux du CCINP, le candidat exécute une épreuve orale sur une demi-journée. Ainsi, hors TIPE, les 4 épreuves orales du candidat CCINP MPI sont planifiées sur un schéma de 4 demi-journées contigües, soit 2 jours calendaires.

- Le langage SQL et les concepts de base de données sont susceptibles d'être traités par l'exercice A. L'ordinateur n'est pas exploité.

- Pour un exercice B donné, un seul langage de programmation est requis, *C* ou *OCaml*. Le langage à exploiter par le candidat est précisé dans l'énoncé de l'exercice B. Le candidat se doit de respecter cette consigne. Dans le cas contraire, cela induit un 0 aux questions relatives à l'exploitation technique du langage.

- Lors du passage au tableau, s'établit une interaction entre l'examinateur et le candidat. Le candidat présente ses résultats sur l'exercice A et l'exercice B. Le candidat peut commencer avec l'exercice de son choix. Le candidat prêtera attention à accorder suffisamment de temps pour traiter l'exercice B. Le candidat exploite le tableau pour présenter ses résultats et notifier des instructions supplémentaires. Le code source projeté peut être exécuté ou amendé sur la base de l'interaction examinateur-candidat.

- Le matériel informatique dédié à l'épreuve est fourni par le CCINP. Les données n nécessaires à l'exercice B sont placées dans une clé USB fournie par le CCINP. Un environnement technique est déployé par une configuration système et logiciels. Cet environnement est spécifié dans un document complémentaire *Annexes*, qui est disponible sur https://www.concours-commun-inp.fr/fr/epreuves/les-epreuves-orales.html

- L'utilisation d'un téléphone portable, calculatrice personnelle (programmable ou non) ou tout autre objet non fourni par le Concours est prohibée. Plus largement, il convient de se référer aux consignes générales du CCINP. Une calculatrice est fournie par le Concours, à toute fin utile.

- Lors de la préparation de l'exercice B, toute erreur de manipulation de l'environnement technique (par exemple, une suppression non souhaitée d'instruction ou de fichier) imputable au candidat ne peut engendrer une nouvelle épreuve orale de substitution. Le candidat doit être vigilant quant à l'exploitation de la machine et des logiciels. Le candidat peut s'exercer en téléchargeant l'environnement technique depuis https://www.concours-commun-inp.fr/fr/epreuves/les-epreuves-orales.html

Les connaissances et les compétences générales[1] attendues sont :

- connaître et maîtriser son cours, les concepts associés et les langages de programmation (*OCaml* et *C*) et de requête (*SQL*),

- maîtriser les définitions et les différents algorithmes du cours,

- s'approprier l'information et le contexte d'un problème,

- savoir hiérarchiser ses connaissances des techniques de l'informatique,

- proposer ou mettre en œuvre des méthodes pour la résolution d'un problème,

- mener à son terme la démarche de résolution du problème par les questions posées,

- concevoir une solution, un algorithme ou une structure de données,

- vérifier la pertinence des réponses ou justifier (preuve, validation de code, …),

- pouvoir faire part des directions prises pendant la préparation,

- réagir aux questions de l'examinateur et savoir mettre à profit ses indications,

- exposer clairement, précisément et concisément ses raisonnements et ses résultats.

Les connaissances et les compétences attendues spécifiques à l'exercice A sont :

- modéliser un point de vue avec les objets conceptuels de l'informatique (table, graphe, arbre, automate, …) ou le spécifier avec un formalisme,

- savoir appliquer sur papier un algorithme sur un exemple simple,

- savoir écrire des requêtes en SQL,

- savoir démontrer de manière théorique des énoncés informatiques ; par exemple, la terminaison, la correction (partielle ou non), la complexité d'un algorithme, des propriétés sur des structures de données, etc.

Les connaissances et les compétences attendues spécifiques à l'exercice B sont :

- savoir exploiter un environnement technique (ordinateur et logiciel),

- développer ou amender un algorithme ou une structure de données dans un langage de programmation (C ou OCaml selon la nature de l'exercice),

- pratiquer une programmation sûre par une validation formelle ou expérimentale rigoureuse,

- établir une validation de code et écrire un jeu de test,

- savoir appliquer la programmation défensive et discuter l'invalidité des données d'un programme.

[1] Ces éléments ne se veulent pas exhaustifs. Se référer au programme CPGE *Informatique – MP2I-MPI, 2021*

MPI

CONCOURS COMMUN INP

RAPPORT DE L'ÉPREUVE ORALE D'INFORMATIQUE

PROPOS INTRODUCTIFS

Le sujet de l'oral d'informatique CCINP en filière MPI est constitué de deux exercices :

- un exercice de type A, "théorique", noté sur 8 points et ne nécessitant pas l'utilisation d'un ordinateur. L'exercice A peut évaluer la syntaxe SQL mais ni la syntaxe C ni la syntaxe OCaml.
- un exercice de type B, "pratique", noté sur 12 points et consacré principalement à de la programmation en C ou en OCaml. Quelques questions d'analyse y sont également posées. Cet exercice fournit généralement un code compagnon que le candidat devra compléter, corriger ou modifier. Il est également possible que le candidat doive écrire intégralement le code demandé par l'énoncé.

L'oral, d'une durée totale d'une heure, se décompose en deux temps :

- 30 minutes de préparation du sujet sur feuille et machine, démarches administratives incluses.
- 30 minutes de passage avec examinateur, démarches administratives incluses.

Les rapports des sessions précédentes ainsi qu'une sélection de sujets de l'épreuve orale d'informatique sont publiés sous la rubrique ANNALES, SUJETS ET RAPPORTS DE LA FILIÈRE MPI :
https://www.concours-commun-inp.fr/fr/epreuves/annales/annales-mpi.html
Également, restent disponibles les 12 « exercices 0 » qui furent proposés à la rentrée 2022-23, en amont de la première session Oraux 2023 de la filière MPI, afin que les candidats se préparent au mieux à cette nouvelle épreuve.

Ce rapport rappelle les consignes principales, qui sont susceptibles d'évoluer. Se référer à la notice du Concours pour obtenir l'ensemble de l'information et au site du concours https://www.concours-commun-inp.fr/fr/epreuves/les-epreuves-orales.html rubrique MPI. Vous y trouverez le cadre de l'épreuve (dont l'environnement technique linux sous format fichier .ova).

En complément à ce rapport, une sélection de sujets, commentés et partiellement corrigés, posés lors de la session 2025 sera publiée, ainsi que l'archive correspondante contenant les codes compagnons, les codes corrigés et les fichiers sources LaTeX.

PRÉPARATION DU SUJET

La préparation s'effectue dans une salle dédiée. Plusieurs candidats préparent en même temps, dans des conditions favorisant la concentration, sous la surveillance de vacataires. À l'entrée de la salle de préparation, convocations et pièces d'identité sont vérifiées.

Pour préparer, le candidat dispose :

- d'un ordinateur disposant de l'environnement linux diffusé sur le site du concours à la rubrique MPI[1] et d'un clavier (configuration AZERTY par défaut, sans pavé numérique),
- de feuilles de brouillon,
- des énoncés imprimés des deux exercices (types A et B),
- d'une calculatrice,
- d'une clé USB contenant le code accompagnant l'exercice de type B.

Le candidat doit copier le code compagnon disponible sur la clé sur sa machine pour pouvoir le compléter / modifier, puis copier le code produit dans un répertoire dédié sur sa clé USB en fin de préparation. Nous insistons sur l'importance du respect de ces consignes qui permettent de disposer d'une sauvegarde du travail produit en préparation en cas de mauvaise manipulation de la clé USB. Les consignes quant aux modalités de préparation et notamment la manipulation des clés sont projetées dans la salle de préparation, de même qu'un chronomètre indiquant le temps de préparation restant. Ce dernier est fixé à 27 minutes, les 3 minutes restantes étant réservées aux formalités administratives et au déplacement vers les salles d'interrogation.

En fin de préparation, les candidats sont accompagnés jusqu'aux salles d'interrogation par un vacataire.

PASSAGE DU CANDIDAT

Le candidat rentre dans la salle qui lui est indiquée, fournit à l'examinateur sa feuille de passage et sa pièce d'identité puis signe la feuille d'émargement que lui présente l'examinateur. Le chronomètre est lancé à ce moment, pour un temps de passage effectif de 27 minutes.

Le candidat dispose d'un ordinateur prêt, dans la même configuration que celui qu'il a utilisé dans la salle de préparation. Un vidéoprojecteur permet de présenter son travail à l'examinateur. Le candidat est invité à copier le code présent sur sa clé sur la machine mise à disposition.

Le candidat commence par présenter tout ce qu'il a préparé. Il débute par l'exercice de son choix et peut, à tout moment, changer d'exercice et/ou revenir sur l'exercice précédemment abordé. Il dispose pour sa présentation d'un tableau et de son code source projeté. L'examinateur demandera que le code produit soit compilé / exécuté / interprété. Il peut donner des indications quant au temps restant et, le cas échéant, indiquer au candidat sur quelles questions se concentrer.

En fin d'épreuve, après la fin du temps imparti, le candidat remet à l'examinateur les énoncés des deux exercices, ses brouillons et la clé contenant le code produit. Il ferme, sur l'ordinateur, les fenêtres de son code et efface le tableau avant de récupérer sa pièce d'identité et sa feuille de passage signée par l'examinateur.

CRITÈRES D'ÉVALUATION

Sont prises en compte dans l'évaluation :

- la maîtrise des éléments du programme relatifs aux deux exercices,
- la maîtrise de l'environnement technique et des outils mis à disposition,
- la pertinence de la réflexion,
- l'interactivité lors de la présentation des résultats et des tests (exercice de type B),
- la réactivité aux éventuels conseils et indications de l'examinateur,
- la qualité de la prestation orale.

[1] https://www.concours-commun-inp.fr/fr/epreuves/les-epreuves-orales.html

BILAN

Sur 773 candidats admissibles (dont 120 grands admissibles), 574 se sont présentés aux épreuves orales. L'épreuve Informatique est conçue sur la base d'une banque commune d'exercices. La moyenne des notes obtenues est de 10,15 avec un écart-type de 3,63. Cela répond aux attendus du CCINP et démontre la capacité à classer les candidats.

Les examinateurs ont proposé des exercices couvrant l'ensemble du programme des deux années de MPI. En particulier, les thématiques suivantes ont été abordées : la récursivité, la représentation de types, les structures de données (listes, tableaux, piles, files, tas, etc.), les arbres, les graphes, les algorithmes d'approximation, les algorithmes probabilistes, les stratégies algorithmiques (diviser pour régner, programmation dynamique, retour sur trace, etc.), l'algorithmique des textes, la concurrence, la décidabilité, la NP-complétude, les langages réguliers, les automates finis, les grammaires non contextuelles, les algorithmes d'apprentissage, l'algorithmique des jeux, la déduction naturelle ou encore les bases de données.

Le langage des exercices de type B était imposé : environ 50 % des exercices devaient être traités en OCaml et 50 % en C. La syntaxe SQL a, quant à elle, été évaluée via certains exercices de type A.

Les examinateurs ont pu voir d'excellentes prestations et sont plutôt satisfaits du niveau global des candidats. La majorité des candidats maîtrise les langages OCaml et C de manière satisfaisante. Ils ont dans l'ensemble été bien préparés à cette épreuve aux modalités exigeantes. Cependant, un nombre non négligeable de candidats fait preuve d'une méconnaissance quasi totale des outils mis à disposition par l'environnement machine du concours. Cette méconnaissance a un effet délétère indéniable sur la prestation des candidats concernés et les handicape certainement encore plus en préparation. Le sentiment des examinateurs est que cet état de fait est lié à un manque d'entraînement pendant l'année avec des outils similaires. Ils soulignent l'importance d'exposer les candidats à ce genre d'environnement linux au moins pendant la préparation aux oraux.

CONSEILS AUX CANDIDATS

Nous souhaitons souligner les points suivants, à l'attention des futurs candidats.

Conseils quant à la gestion du temps

De nombreux candidats n'ont pas su gérer leur temps durant le passage, passant par exemple plus de 20 minutes sur l'exercice A et réfléchissant à des questions qu'ils n'ont pas abordées en préparation plutôt que de présenter ce qu'ils ont fait sur l'exercice B d'abord (ou inversement). Plus d'une fois, les examinateurs ont pu constater sur les brouillons des candidats que ces derniers avaient abordé en préparation des questions non présentées pendant l'oral. L'examinateur tente d'aider les candidats dans leur gestion du temps mais leur tâche est rendue difficile par le fait que les candidats n'annoncent que très rarement ce qu'ils ont traité et se lancent dans des questions non traitées avant de présenter ce qui a été fait en préparation. Nous déplorons également des candidats peu efficaces qui, une fois leur présentation achevée, papillonnent sans réellement se concentrer sur les questions qu'ils n'ont pas encore traitées, comme s'il était impossible de répondre aux questions qui n'ont pas été préparées.

Un sujet est composé de deux exercices et il n'est pas possible de faire de transfert de points entre les deux. Un candidat qui réussit brillamment l'exercice A et n'aborde pas l'exercice B ne pourra avoir une note supérieure à 8/20.

En conséquence, nous rappelons que :

- il est conseillé aux candidats de prendre connaissance des deux exercices : certains énoncés sont difficiles à traiter à l'oral sans y avoir un peu réfléchi en amont ;

- l'examinateur connaît le sujet, il n'est donc pas nécessaire pour le candidat de le réintroduire. Il est important de citer le numéro de la question à laquelle on répond mais inutile de lire son énoncé dans son intégralité ;

- il est recommandé aux candidats de commencer par annoncer, en début d'épreuve, les questions des deux exercices qui ont été traitées en préparation. Ceci permet à l'examinateur d'aider les candidats à gérer leur temps et de les interroger sur l'ensemble des questions abordées pendant la phase de préparation ;

- il n'est pas toujours judicieux de commencer par l'exercice A. De plus, il est possible de passer d'un exercice à l'autre pendant le passage, voire de ne pas traiter toutes les questions dans l'ordre (ou même d'en sauter quelques-unes) ;

- le candidat peut, bien entendu, modifier son code pendant l'oral pour corriger ses erreurs ou aborder des questions non traitées en préparation en les programmant devant l'examinateur sur la machine de la salle.

Conseils quant à la manipulation de la machine et de l'environnement

- Savoir éjecter proprement une clé fait partie des compétences attendues d'un élève de MPI. Nous rappelons donc qu'il ne faut pas arracher une clé USB sans précaution, en particulier alors que des documents qui y sont présents sont encore utilisés. Certains candidats n'ont pas respecté les consignes relatives à la manipulation des clés et des fichiers (indiquées devant la salle de préparation, dans la salle de préparation et à l'oral par les vacataires). Ces manquements rendent impossible la récupération de leur travail en cas de problème. Certains candidats ont dû ainsi reprendre l'exercice B depuis le début pendant le passage.

- Durant la session 2025, nous avons pu relever les propos suivants de la part de candidats : "J'ai du mal avec Linux, comment on revient sur la fenêtre du terminal ?", "Vous savez où est le terminal ? Je sais qu'il y a gcc mais on faisait ça avec copilot sur github.", "Je ne trouve pas le bouton pour compiler sous VSCode", "D'habitude on fait ça en ligne (après une demande d'évaluation de code OCaml)".

  Ces propos sont le symptôme d'un manque important de maîtrise de l'environnement machine linux utilisé au CCINP (et similaire aux environnements linux des autres concours). Ne jamais compiler de C via un terminal ou s'entraîner uniquement sur des plateformes en ligne ne permet pas de se préparer correctement à l'épreuve orale d'informatique du CCINP. Nous conseillons aux candidats de prendre en main l'environnement machine du CCINP (voir https://www.concours-commun-inp.fr/fr/epreuves/les-epreuves-orales.html, rubrique MPI) avant les épreuves orales.

- Il est attendu des candidats une maîtrise élémentaire d'un terminal de manière à pouvoir se déplacer dans une arborescence et lancer une commande de compilation de leur code. Sans être nécessaire, l'examinateur rappelle que, dans un terminal, la touche de tabulation permet souvent de compléter des morceaux de la commande en cours et que les flèches directionnelles permettent de naviguer dans l'historique des commandes. Connaître ces raccourcis est un gain de temps non négligeable pour les candidats qui tapent lentement.

- Si aucune connaissance concernant la commande `make` n'est attendue, tous les exercices B utilisant le langage C mettent à disposition des candidats un `Makefile` ainsi que des consignes complètes permettant de l'utiliser, sous réserve d'une familiarité minimale avec un terminal. Le `Makefile` est disponible ici :
  https://www.concours-commun-inp.fr/fr/epreuves/les-epreuves-orales.html (rubrique MPI) et les consignes sont en tête de tous les exercices publiés comportant du C.

  Lors du passage à l'oral, l'examinateur peut demander aux candidats de compiler leur code avec un *sanitizer* pour évaluer certains aspects de la gestion de la mémoire. Les commandes de compilation permettant de compiler avec ou sans *sanitizer* sont disponibles dans les consignes, mais il est souvent plus rapide d'utiliser `make` ou `make safe`.

- Il a été constaté chez certains candidats un manque de familiarité avec l'usage d'un clavier. Cela se manifeste par une vitesse de frappe très lente ou des difficultés à taper certains caractères (guillemets, chiffres, …). Cela n'est bien entendu pas pénalisé, mais ces candidats se sont retrouvés mécaniquement désavantagés. Il est à noter que les claviers des machines ne disposent pas de pavé numérique et sont par défaut en configuration AZERTY et que les candidats doivent y être habitués ou bien savoir en changer.

- Certains candidats ont demandé s'il était possible d'utiliser - par exemple - VSCode pour présenter leur code OCaml. Les examinateurs soulignent que les candidats sont libres d'utiliser les éditeurs de leur choix, dans la limite des outils disponibles dans l'environnement du CCINP.

- Selon l'éditeur choisi, le passage à la ligne automatique (word wrap) n'est pas activé par défaut. Cela empêche l'examinateur de lire l'intégralité du code. Il sera alors demandé au candidat d'activer cette option ou de passer à la ligne manuellement.

Conseils quant à la programmation

- Trop nombreux sont les candidats qui arrivent au passage sans avoir jamais évalué leur code en préparation. Cela se voit dès la première question de code. Ils passent alors un temps laborieux et inconfortable à corriger la question 1 pendant le passage et se retrouvent parfois à ne même pas pouvoir présenter le code des questions suivantes. L'épreuve est certes courte, mais coder trop rapidement aux dépens de l'évaluation et des tests n'est pas une bonne stratégie pour réussir l'exercice B. L'immense majorité des candidats qui ne compile / n'évalue / ne teste pas son code en préparation n'atteint pas la moyenne à cette épreuve.

- Les tests des fonctions demandées par l'exercice B font partie de l'évaluation et l'examinateur demande à voir le résultat de ces tests pendant le passage. Dans le langage OCaml, comme dans le langage C, il est possible et souhaitable d'écrire les tests à l'avance plutôt que de devoir les réécrire pendant le passage.

- Même si elle n'est pas explicitement au programme, la fonction Printf.printf en OCaml a un fonctionnement très similaire à celui de la fonction printf en C et est particulièrement pratique pour un oral, que ce soit comme outil simple de débogage ou pour valider simplement un programme par un affichage d'exemples. Le cas échéant, l'énoncé en rappelle les modalités d'usage et les codes de formatage.

- Les fonctionnalités des langages au programme ainsi que tout ce qui se trouve dans la rubrique "éléments de syntaxe devant être reconnus et utilisables après rappels" peuvent être librement utilisés par les candidats. Si un candidat utilise une construction des langages en dehors de ce cadre, il pourra lui être demandé d'expliquer comment répondre à la question avec uniquement les

outils du programme. L'examinateur n'interdit pas, mais ne souhaite en aucun cas encourager, l'utilisation d'éléments hors programme : tous les sujets peuvent être résolus avec le fragment au programme des langages et utiliser des constructions alternatives expose le candidat à des questions qui peuvent lui faire perdre du temps sans que sa réponse ne soit davantage valorisée.

- La manipulation élémentaire de la mémoire en C pose des problèmes importants à de nombreux candidats. L'usage élémentaire de `malloc` ou de `free` est trop souvent mal maîtrisé, en particulier en ce qui concerne la nécessité d'initialiser les variables contenant des pointeurs et la valeur du contenu de la mémoire allouée dynamiquement (non, il n'y pas des 0 par défaut lors de l'allocation dynamique d'un tableau).

- Nous conseillons vivement aux candidats de s'entraîner à lire et comprendre les messages d'erreurs dans les deux langages. En particulier pour le langage C, le *sanitizer* et l'option `-g` (dont l'usage est indiqué dans les commandes de compilation et implémenté dans le `Makefile`) permettent d'obtenir des messages d'erreur plus explicites et localisés qui aident à la correction de ces erreurs, pour peu qu'on sache les interpréter.

- On observe encore de trop nombreuses confusions entre listes et tableaux en OCaml avec des tentatives non pertinentes d'accéder à un élément d'une liste via son indice.

Conseils quant à la maîtrise du programme

- La syntaxe SQL est généralement maîtrisée mais de très nombreux candidats ont eu de grandes difficultés avec la modélisation de problèmes et le modèle entité-association. Même si aucun formalisme n'est spécifié par le programme, des diagrammes entités-association simples font partie des attendus que les candidats doivent savoir mettre en œuvre. Il en va de même pour les schémas relationnels et leur lien avec le modèle entité-association.

- La structure d'arbre binaire de recherche est mal connue et parfois confondue avec celle des tas binaires. La notion d'arbres binaires de recherche équilibrés a posé de nombreux problèmes aux candidats.

- Un algorithme aussi essentiel et central que le parcours de graphe n'est souvent pas maîtrisé. Les difficultés liées au manque de maîtrise de cet algorithme s'étendent à ses variantes, en particulier Dijkstra et A*.

- Les graphes de flot de contrôle sont au programme.

- L'existence d'un invariant est souvent confondue avec une preuve de correction : nous rappelons aux candidats qu'une fois l'invariant prouvé, il reste à l'exploiter, ce qui n'est pas toujours immédiat.

- Il est important de savoir ce qu'est un ordre bien fondé.

- Le chapitre IA du programme de MPI n'est souvent pas bien maîtrisé. Les candidats confondent les noms des algorithmes et peinent à expliquer leur principe.

Divers

Dans certaines salles de passage, le tableau sert aussi d'écran de projection. Il est attendu des candidats qu'ils n'écrivent pas sur la partie du tableau servant à la projection sans l'accord de l'examinateur, sous peine de rendre la projection, ainsi que ce qui est écrit, illisible.

Modalités de l'épreuve orale

d'informatique pour la filière MPI

Ponts ParisTech, ISAE-SUPAERO, ENSTA Paris, TELECOM Paris, MINES Paris,

MINES Saint Étienne, MINES Nancy, IMT Atlantique, ENSAE Paris, CHIMIE ParisTech - PSL

Modalités de l'épreuve orale d'informatique
Filière MPI

1 Généralités sur l'épreuve

L'épreuve orale d'informatique du concours évalue les compétences en informatique et peut porter sur tous les aspects des programmes étudiés en MP2I et en MPI. L'épreuve s'appuie principalement sur le programme d'informatique, mais elle peut aussi mobiliser des connaissances apprises dans les autres matières. L'épreuve est toujours une épreuve sur machine, chaque oral faisant intervenir de la programmation dans un ou plusieurs des langages étudiés en MPI (C, OCaml et SQL).

L'épreuve est notée en utilisant le rendu du candidat, mais aussi les interactions entre celui-ci et le jury. Le rendu est composé d'un compte-rendu papier qui doit être complété au fil de l'oral ainsi que du code écrit sur la machine. La clarté des programmes écrits est prise en compte.

On rappelle que si l'interaction avec le jury est une part importante de l'oral, le jury est là pour évaluer la performance des candidats et non les aider dans les exercices. Il peut arriver qu'un jury débloque un candidat en difficulté et ces aides peuvent être prises en compte ou non au moment de l'évaluation. Par exemple, la reformulation d'une question ou une documentation mal écrite n'est pas prise en compte tandis qu'un candidat bloqué, car ignorant un point de cours, peut se voir pénalisé.

2 Format d'un exercice

Les 3h30 de l'oral peuvent être reparties en un ou plusieurs exercices, chaque exercice étant composé d'une ou plusieurs questions autour d'un même thème avec en général une progression de difficulté. Plusieurs exercices peuvent partager un même thème et l'attaquer sous des angles différents (par exemple une approche différente ou des langages différents). Le passage d'un exercice à l'autre est entièrement à la discrétion du jury selon des critères d'avancement du candidat et de durée prévue pour l'exercice. Les candidats doivent ainsi s'attendre à devoir terminer abruptement un exercice, car le temps imparti pour l'exercice est fini.

Le format de chaque exercice est très libre. Le jury peut fournir toutes les questions d'un exercice dès le début de celui-ci sous la forme d'un document papier ou sur l'ordinateur, mais il peut aussi proposer des exercices très "ouverts" où le candidat est face à un problème général. L'exercice se construit dans un dialogue avec le jury, les candidats pouvant faire des propositions sur la manière de procéder que le jury peut accepter ou non — que la proposition soit pertinente ou non — selon ce qu'il veut tester. Enfin, le jury peut aussi proposer des versions intermédiaires à ce qui est décrit plus haut : par exemple commencer par une question ouverte, puis distribuer un bloc de questions puis de nouveau une question ouverte, etc.

3 Fichiers disponibles pour un exercice

Pendant l'oral, le jury met à disposition des candidats un ensemble de fichiers soit directement sur leur ordinateur soit par l'intermédiaire d'une interface web. Des fichiers peuvent être ajoutés à tout moment : avant le début de l'oral, au début d'un exercice ou même pendant l'exercice.

Si le jury rajoute des fichiers pendant l'examen sur l'ordinateur des candidats, il prendra soin de ne pas écraser des fichiers et de prévenir les candidats de cet ajout. Les fichiers mis sur les machines des candidats peuvent comprendre :

- des *fichiers sources* en C, en OCaml ou en SQL ;
- des *fichiers de données* comme des images, des graphes, des textes, etc. Ces données peuvent être dans des formats standards (comme du CSV) ou non. Il n'est pas attendu de connaître ces formats, il sera toujours fourni soit du code qui lit les données, soit des indications précises sur le format qui permettront de lire facilement les données avec les primitives de base du langage dont la documentation sera fournie ;
- des *scripts* bash ou des Makefile simplifiant la compilation ou faisant des tests sur les programmes. Il n'est pas attendu de savoir lire ou écrire ces scripts. Si un candidat doit lancer un script sans l'aide du jury, il aura à disposition une documentation décrivant la commande à entrer dans le *shell* ;
- des documents décrivant soit l'objectif d'un exercice, soit des questions, soit une explication des fichiers fournis, soit de la documentation.

4 Compétences évaluées

Cette section présente maintenant plusieurs types de questions auxquels les candidats peuvent être confrontés. Cet inventaire est là pour aider la préparation des candidats en présentant des compétences exigées soit dans le dialogue avec le jury soit dans les questions, mais il ne s'agit en aucun cas d'un inventaire exhaustif des questions possibles. Des questions dont le format ne respecte pas les types suivants pourront être posées aux candidats. Voici différentes grandes catégories de questions qui peuvent être posées aux candidats (voir sujet zéro pour des exemples) :

- *Architecture du code.* Dans ce type de questions, le sujet décrit un problème et le candidat doit décrire comment il décomposerait le problème en fonctions, classes, modules ou fichiers, quelles seraient les structures de données appropriées pour représenter les données manipulées, etc. Pour ce type de questions, il n'est pas attendu de notions avancées de génie logiciel (qui ne sont pas au programme), mais simplement la capacité à décomposer un gros problème en plusieurs sous-problèmes plus simples et indépendants ainsi qu'à choisir les bonnes structures de données.
- *Réflexion algorithmique.* Dans ce type de questions, le sujet décrit un problème et le candidat doit proposer un algorithme pour s'attaquer au problème. En général, on s'intéressera à trouver un algorithme suffisamment efficace qui donne la bonne réponse, mais,

dans certains cas, le jury pourra aussi poser des questions plus ouvertes. Par exemple, on pourra demander aux candidats qu'ils proposent des heuristiques sur un problème NP-complet ou utiliser un algorithme A*.

- *Lecture de programme.* Le jury peut donner aux candidats des fragments de programmes ou des programmes entiers et leur demander d'expliquer ce que font ces programmes. Ces questions peuvent être posées pour permettre aux candidats de se familiariser avec un code qu'ils devront ensuite éditer, compléter ou utiliser comme source d'inspiration pour leur propre production.
- *Programmation défensive, tests, débogage et commentaires.* Dans ces questions, le candidat dispose d'un code (correct ou non) et il doit rajouter soit des commentaires, soit des tests, soit spécifier les cas où le programme peut échouer, soit corriger le programme pour qu'il n'échoue plus, soit écrire des tests pour tester le code. Noter que le programme à éditer ou corriger peut être un programme écrit pendant l'oral et donc le jury invite fortement les candidats à appliquer des principes de la programmation défensive (expliciter les pré- ou post-conditions des fonctions, tester avec assert, etc.).
- *Écriture ou édition de programme.* Pendant l'oral les candidats auront à écrire des programmes qu'ils devront ensuite exécuter. Il est possible que le jury fournisse une base de code à éditer ou compléter, mais il est aussi possible que les candidats doivent écrire un programme sans avoir de base. Dans les deux cas, le langage de programmation dans lequel les candidats doivent écrire peut être imposé par l'exercice.
- *Preuves et calculs.* Bien que l'épreuve soit une épreuve d'informatique pratique, le jury peut poser des questions plus théoriques comme, par exemple, des preuves de correction, des calculs de complexité, des raisonnements logiques.
- *Retours critiques.* À la fin d'un exercice, on pourra demander aux candidats un retour critique sur ce qu'ils ont fait. Par exemple, le jury pourra demander une comparaison des divers algorithmes qui ont été utilisés sur un même problème.

Certaines de ces catégories peuvent ne pas être mobilisées pendant un oral et, à l'inverse, des questions sortant de ces catégories peuvent être posées.

5 Écriture du compte-rendu

Les candidats devront écrire un compte-rendu d'environ une page au fil de l'épreuve orale. Le contenu précis de ce qui est attendu dans le compte-rendu sera précisé soit par écrit dans un sujet soit dans les entretiens avec le jury. Cependant, pour que les candidats puissent mieux se préparer, voici à titre indicatif ce que le jury attend.

Tout d'abord il faudra penser à noter brièvement toutes les questions des examinateurs pour pouvoir relire correctement le compte-rendu. Ensuite, pour les questions qui demandent du code, on précisera le chemin et/ou la portion du fichier écrit ou modifié. On précisera aussi si des tests ont été effectués sur le code. Pour les questions n'attendant pas du code, on cherchera à donner un résumé de ce qui a été expliqué au jury et on pourra utiliser des schémas. Il n'est pas attendu des candidats d'écrire par eux-mêmes une conclusion au compte-rendu, mais le jury peut le demander.

6 Environnement de développement

Les épreuves se déroulent dans des salles équipées d'ordinateurs avec au moins un ordinateur par candidat. Un sujet zéro et une image virtuelle très similaire à l'environnement disponible pendant le concours seront disponibles sur le site du concours.

Il n'est pas attendu des candidats qu'ils soient très familiers de l'environnement du concours, mais un candidat qui n'aurait jamais utilisé un OS similaire à celui fourni ou qui n'aurait utilisé aucun des éditeurs présents dans l'environnement risquerait de perdre du temps pendant l'épreuve. De la même façon, comme les candidats auront à exécuter des commandes, il est recommandé qu'ils sachent naviguer dans le système de fichiers à l'aide des commandes `cd`, `ls` et `pwd`.

Les candidats auront accès, dans l'environnement de développement, à la documentation officielle des langages étudiés, à une petite documentation de l'environnement de développement, ainsi qu'à une sélection d'éditeurs de texte. L'environnement de développement ne comprendra pas tous les éditeurs existants, en particulier les éditeurs trop exotiques, non libres ou non supportés sur l'OS choisi ne seront pas inclus.

8 Épreuve d'Informatique – Filière MPI

Ce chapitre présente le compte rendu de la session 2025 de l'épreuve de TP d'informatique du CCMP. L'objectif de ce document est de présenter les modalités de l'épreuve, de présenter quelques statistiques sur celui-ci et des conseils pour que les candidats soient au mieux préparés les prochaines années.
Un des objectifs de ce document étant de présenter comment les candidats pourraient mieux se préparer, il va lister beaucoup de « défauts » mais le jury précise la qualité de la prestation des candidats. Tout comme en 2024, le jury a trouvé que les candidats de la session 2025 étaient bien préparés même si certains travers pourraient être corrigés.

8.1 Déroulement de l'épreuve.

Accueil des candidats.

Chaque session de l'épreuve du TP d'informatique commençait par un accueil des candidats où les examinateurs rappelaient les consignes générales :

- les candidats peuvent manger et boire dans les salles (mais en faisant très attention aux ordinateurs !) ; ils peuvent aller aux toilettes (en demandant avant !) ;
- pendant l'oral, les candidats peuvent n'avoir avec eux que pièce d'identité, convocation, stylos et éventuellement nourriture et boisson. S'ils ont d'autres affaires (comme des sacs, des téléphones ou des montres), ils peuvent les poser éteints dans un coin de la salle. Du brouillon et un compte rendu vierge leur sont fournis ;
- les candidats peuvent poser toutes leurs questions pendant l'oral et cela n'affecte pas la notation (mais le jury se réserve le droit de ne pas répondre) ;
- il est demandé aux candidats de tester leurs codes ;
- les candidats doivent rédiger un compte rendu (voir la section détaillée, plus bas) ;
- enfin les épreuves ayant lieu sur machine, il est possible d'avoir des pannes. Dans ce cas, les candidats doivent prévenir au plus vite les examinateurs qui résoudront le problème en compensant la perte de temps par du temps supplémentaire à la fin ou dans la notation. Il est attendu des candidats qu'ils enregistrent régulièrement.

Installation dans les salles.

Après l'introduction générale, les candidats sont ensuite répartis dans les diverses salles. Cette année, il y avait encore environ 4 candidats par salle (parfois moins, exceptionnellement plus) . Les candidats posent leurs sacs dans un coin de la salle puis s'installent à un des postes disponibles. Une nouveauté de cette année était que les candidats ont eu le droit à quelques minutes de familiarisation avec la machine.

Épreuve principale.

Les examinateurs donnent le top départ à une heure précise et à ce moment les candidats doivent charger une page web où le sujet est mis.
Même si le sujet est souvent fourni en intégralité dès le début de l'épreuve, le jury déconseille de le lire en entièreté car ce serait une perte de temps, en revanche ce n'est pas une mauvaise idée de lire quelques questions à l'avance (surtout au début de l'épreuve) pour mieux comprendre ce qui est attendu.
Au cours des 3h30, les examinateurs passent régulièrement voir les candidats. Les candidats qui ont une question peuvent appeler les examinateurs sans attendre que ceux-ci ne passent.
Certains candidats tentent de sauter certaines parties sans le demander à l'examinateur. Cette technique est à proscrire lors d'un oral, elle exhibe de toute façon les lacunes du candidat (peur de la technicité, impasse, ou autre).

Fin de l'épreuve.

L'épreuve est prévue pour durer 3h30. Quelques minutes avant la fin les examinateurs rappellent aux candidats l'imminence de la fin de l'épreuve pour que ceux-ci s'assurent que le compte rendu est bien finalisé et les examinateurs passent une dernière fois pour noter où en sont les candidats. Il n'est pas demandé aux candidats de mettre leurs fichiers à un endroit spécifique à la fin de l'épreuve.

Précisions quant à l'écriture du compte rendu.

Dans l'épreuve de TP informatique le compte rendu sert principalement à suivre la progression des candidats durant l'épreuve ainsi qu'à résumer ce qui a été fait.
Quand les examinateurs passent voir les candidats, ils commencent généralement par lire le compte rendu, voient ce qui a changé depuis leur dernier passage puis, posent éventuellement des questions ou regardent ce que le candidat est en train de faire. Ceci permet d'économiser les interactions orales qui feraient perdre du temps au candidat mais aussi pourraient gêner les autres candidats dans la salle. Le compte rendu doit donc être plus détaillé qu'une succession de « Q 8 : faite » mais ce n'est pas non plus la peine de détailler les réponses autant que dans une copie écrite. En général, même les questions les plus compliquées ne requièrent pas d'écrire plus de 5 lignes.
Savoir remplir le compte rendu n'est pas une compétence attendue, le jury ne retire pas de points sur la façon dont les candidats remplissaient les compte rendus, en revanche il peut signaler à des candidats qu'ils peuvent être plus concis dans leurs réponses (pour ne pas perdre de temps) ou, au contraire, qu'il faut détailler plus leurs réponses quand ils trouve celles-ci insuffisantes. Les réponses fausses (mauvais calcul de complexité ou algorithme faux par exemple) sont prises en compte dans la notation. Voici quelques précisions sur la façon de remplir le compte rendu :

- pour toute question posée dans le sujet ou à l'oral, le jury attend une réponse écrite sur le compte rendu ;
- les sujets sont majoritairement donnés en format PDF avec des questions numérotés. Ce n'est donc pas la peine de recopier les questions, il suffit de donner le numéro de la question répondue ;

- certaines questions sont vraiment très simples (comme écrire une fonction qui somme deux vecteurs 2D) dans ce cas le candidat peut se contenter d'un « Q 1 : faite » écrit sur le compte rendu ;
- pour les questions qui n'attendent pas du code, on demande une réponse brève sur le compte rendu et de cette façon l'examinateur peut lire la réponse sans déranger le candidat (ou les autres candidats).
- enfin, pour les questions d'algorithmique ou de code non triviales, le jury attend une description générale de l'algorithme, une complexité (sans justifier) et les tests effectués. Par exemple, dans le cas où l'algorithme est un parcours de graphe, il convient d'expliquer si le parcours est en largeur ou en profondeur, dans quel graphe quand le graphe est implicite, etc. puis d'indiquer brièvement quels graphes ont été testés et que la complexité est $O(n+m)$ avec n le nombre de noeud et m le nombre d'arêtes. Le compte-rendu peut servir de "brouillon" pour préparer les tests qui sont parfois plus lisibles sous forme de dessin (par exemple pour un graphe) que sous forme de code.

8.2 Commentaires généraux sur la méthode de programmation.

Le jury a constaté que les défauts relevés dans les rapports des concours 2023 et 2024 étaient bien moins présents en 2025, mais certains défauts persistent.

Architecture du code.

Beaucoup de candidats voient chaque question comme un bloc unitaire et semblent avoir du mal à voir la combinaison de plusieurs sous-problèmes qui peuvent être implémentés dans plusieurs fonctions avec des tests.
De manière plus générale, le jury trouve que, même quand les candidats ont la bonne idée, ils ont du mal à décrire et donc à réfléchir sur leur solution avant de se lancer dans le code.
Cela pénalise souvent les candidats qui perdent du temps à ne pas tester chaque partie indépendamment quand il y a des bugs, qui écrivent souvent des bouts de code inutiles qu'ils doivent ensuite retravailler et parfois qui se rendent compte qu'au bout de 15 minutes que leur solution ne marche pas. Pour toutes les questions non triviales, le jury conseille de passer quelques minutes à élaborer rapidement la solution (par exemple sous forme de pseudo code ou juste en donnant les grandes étapes) puis à réfléchir à comment simplifier la solution ou la décomposer en plusieurs sous-problèmes.
Tout ceci fait que les candidats sont en majorité bien plus à l'aide sur les sujets très guidés avec beaucoup de questions intermédiaires, et sont parfois déroutés sur les sujets plus ouverts.

Tests.

Pour les questions d'algorithmique non triviales, il était attendu des candidats qu'ils testent leurs programmes. Cette consigne était donnée avant l'oral et les examinateurs le rappelaient régulièrement aux candidats. Il n'est pas attendu de faire des tests complets qui pourraient presque garantir une absence de bugs mais simplement vérifier qu'il n'y a pas d'erreur manifeste. Cette demande de tester les programmes remplit plusieurs objectifs qui permettent :

- de valider la compétence qui figure au programme de la filière MPI ;
- aux candidats d'avoir confiance dans ce qu'ils écrivent. Certains candidats semblent penser que tester un programme est une perte de temps voire un aveu de faiblesse. Le jury préfère toujours un candidat prudent qui teste son code à un candidat trop sûr de lui qui ne le teste pas, même quand le code est correct. Un candidat qui refuse de tester un code que le jury sait faux, fait une mauvaise impression.

Il est évident que la quantité de tests à effectuer dépend de la complexité du code de la difficulté d'écrire des tests et de la précision des tests. Une fonction qui renvoie une information booléenne (si un graphe est connexe ou non par exemple) a plus de chances de renvoyer la mauvaise réponse au hasard qu'une fonction qui renvoie quelque chose de précis (par exemple si cela affiche pour chaque composante la liste des nœuds). De la même manière une fonction très simple (par exemple la somme de deux vecteurs 2D) a moins de chances d'être fausse que l'implémentation d'un algorithme compliqué (p.ex. une file à priorité).
Voici plusieurs conseils pour les candidats :

- il est important de calculer à l'avance le résultat des tests. À plusieurs reprises, le jury a vu des candidats écrire un unique test compliqué et prendre le résultat de leur fonction comme étant le standard pour leur test alors que la fonction renvoyait un résultat faux ;
- des candidats utilisent un éditeur pour leur code et testent avec un shell (comme `utop`) et donc leurs tests "disparaissent" et ils perdent beaucoup de temps à les réécrire pour chaque petit changement de leur fonction, ce qui pousse les candidats à n'écrire que peu de tests. Le jury ne déconseille pas d'utiliser `utop` mais recommande d'écrire les tests dans le code principal ;
- il est parfois possible de tester plusieurs fonctions en même temps quand les questions sont données à l'avance. S'il est demandé, par exemple, de faire des fonctions somme, produit puis évaluation pour des polynômes, on peut tester très rapidement chaque fonction puis faire des tests qui combinent ces trois fonctions en testant que $((P+Q)*R)(42) = (P(42)+Q(42))*R(42)$ pour plusieurs polynômes P, Q et R.

Gestion des bugs.

Les outils adaptés au débuggage (comme `ocamldebug` ou `trace` en OCaml et `gdb` en C) sont peut-être un peu compliqués pour les candidats mais voici quelques conseils faciles à mettre en œuvre pour s'attaquer méthodiquement au débuggage :

- parfois un premier problème engendre un second plus visible (par exemple un calcul renvoie le mauvais résultat et à cause de cela on fait un accès hors des cases du tableau). Il faut bien penser à détecter le moment où le premier problème apparaît. Pour cela la programmation défensive et l'utilisation d'assert permet de gagner du temps. De la même manière, il est souvent intéressant de chercher à simplifier l'exemple où l'algorithme buggue, car il est plus simple de suivre le déroulé de l'algorithme ;

- les candidats pensent souvent à faire des printf mais pas toujours à l'utilisation des assert. Même si le assert est moins informatif et donc qu'il est parfois plus utile de débugguer en utilisant aussi printf, lire des dizaines de lignes prend plus de temps qu'un simple assert... il faut savoir combiner les deux ;
- quand on manipule une structure de données un peu compliquée (comme un tas ou un arbre rouge-noir) il peut être intéressant de faire une fonction qui vérifie que cette structure a le format attendu (pour un tas ou un arbre rouge-noir, par exemple, la fonction de vérification est assez simple comparé au reste de l'algorithme). Lors de la phase de test on peut par exemple vérifier la structure après chaque modification (ce qui renforce la confiance en la correction de l'algorithme car les tests testent bien mieux). Il peut être utile d'utiliser une variable `debug` pour déclencher ou non ces tests ;
- quand le candidat écrit une fonction qui attend un argument qui a une forme précise, il ne faut pas hésiter à mettre un assert pour le vérifier ;
- quand un programme a plusieurs fonctions et sous-fonctions on peut tester chaque fonction indépendamment ;
- enfin, il est presque toujours recommandé d'utiliser les warnings du compilateur et les options de compilation recommandées en C sont `-Wall -Wextra` et `-fsanitize=address` (mais un candidat qui n'a jamais testé ces options risque d'être décontenancé par les warnings).

Environnement de développement.

Nous encourageons vivement les candidats à tester la machine virtuelle proposée sur le site du concours, ne serait-ce que 10 min, pour ne pas être déstabilisé face à l'environnement qu'ils rencontreront le jour du concours. Pour la session 2026, la principale évolution prévue est la mise à jour pour la nouvelle version stable de debian (Trixie) avec comme principal changement le passage d'Ocaml en version 5.3.
Juste avant le début de l'épreuve, les candidats avaient quelques minutes pour se familiariser avec la machine. Certains candidats, pris par le stress commencent à taper des bouts de code en C ou en OCaml, le jury n'est pas persuadé que cela les aide beaucoup d'autant que les candidats ne savent à ce moment pas quel sera le langage ni si un patron de code est fourni.
Par exemple, lorsque rappelée, l'utilisation de `ocamlopt` ou de `ocamlc` peut se révéler laborieuse et le manque d'aisance fait perdre du temps aux candidats. À l'inverse, certains candidats ne savent utiliser OCaml qu'en compilant. Dans certains sujets, utiliser un shell (comme `utop` ou même `ocaml` utilisé en mode interactif) est un grand atout : on peut tester rapidement des fonctions, quand le code manipule des types sommes récursifs, on peut les afficher sans faire de fonction d'affichage ad-hoc, etc. Au passage, rappellons l'existence de `#use "fichier.ml"` qui permet de renvoyer à l'évaluation l'intégralité du fichier là où un copier-coller peut être assez lent.
Pour finir rappellons que les candidats vont passer un certain temps dans le terminal. Sans que le jury n'évalue directement la familiarité avec l'outil, certains candidats perdent beaucoup de temps en ne connaissant pas les raccourcis de base comme flèche du haut pour remonter dans l'historique, ctrl+R pour rechercher, ctrl+C pour arrêter un processus, etc.

Nom des variables, fonctions et commentaires.

Les examinateurs examinant le code des candidats, il est attendu des candidats qu'ils produisent des codes lisibles. Le jury n'attend pas des candidats des choses élaborées ou le suivi de conventions particulières mais simplement que les candidats utilisent des noms de fonctions ou de variables pertinents et des commentaires aux endroits qui le nécessitent.
Quand le nom de la fonction et de ses arguments ne suffisent pas à comprendre son objectif, ou si la fonction est longue avec plusieurs blocs qui accomplissent plusieurs choses différentes, le jury attend aussi un commentaire rapide qui décrit ce que fait le bout de code considéré.
Pour des variables locales à une fonction, les candidats peuvent se contenter de noms simples de variables mais pour les variables globales, pour les noms de fonction et éventuellement pour les variables nommant les arguments de ces fonctions, il est demandé de donner des noms qui précisent ce qu'elles représentent. Le jury n'a pas à chercher quels sont les rôles respectifs de `aux`, `aux1`, `aux2` et `aux1bis` ni ce que signifient les arguments `a`, `b` et `c` tous de type `int`.

Documentation.

Le jury fournissait une documentation accessible via le navigateur. La documentation de référence en C et OCaml étaient disponibles ainsi qu'une "cheatsheet" de sqlite3.
Les noms et prototypes des fonctions de la bibliothèque standard qui étaient nécessaires dans certains sujets (pour ouvrir des fichiers ou prendre des mutex) étaient généralement directement rappelés dans les sujets soit par de la documentation soit par des exemples mais il ne faut pas que ça empêche les candidats d'aller lire la documentation qui contient souvent des exemples d'utilisation.
Pour l'édition 2025 du concours, les principales documentations fournies pour OCaml et C étaient des copies locales de
`https://ocaml.org/manual/4.13/index.html` et `https://en.cppreference.com/w/c.html`.

8.3 Commentaires liés au programme.

Il est important de bien maîtriser et de savoir implémenter rapidement tous les algorithmes de base (parcours en largeur et en profondeur, tas, etc.), mais il faut aussi connaître toutes les définitions et tous les algorithmes du programme. Il est très dangereux de faire une impasse complète sur un point du programme.
Le jury a remarqué que certains candidats (même parmi les bons) ont une connaissance limitée voire inexistante de certains traits du langages pourtant explicitement au programme. Un exemple : certains candidats ne savent pas rattraper une exception en Ocaml (soit parce qu'ils ne connaissent pas `try` et `with` soit parce qu'ils ne savent pas qu'il faut que les types du blocs `try` et `with` soient compatibles). Rappellons que de la documentation sur les langages étant fournie et les candidats qui ont oublié certains points de syntaxe peuvent souvent s'en sortir seuls.

8.4 Commentaires liés au langage SQL.

Cette année, nous avons de nouveau eu des candidats qui se déclaraient totalement incompétents en SQL et souhaitaient sauter cette partie. L'impasse en SQL est fortement déconseillée et est pénalisée.

La plupart des candidats ont réussi à résoudre les questions de SQL mais certains le font beaucoup plus rapidement que d'autres. Nous conseillons de s'entraîner à résoudre rapidement des problèmes de SQL.
Cette année encore, tous les sujets qui manipulaient du SQL utilisaient le moteur de requête SQLite3 (dont une brève documentation était fournie) mais le jury se réserve le droit d'utiliser d'autres moteurs de requêtes dans le futur. Il n'est pas attendu de connaissance spécifique aux légères variations qui peuvent exister entre les divers moteurs de requêtes et en particulier il n'est pas attendu des candidats qu'ils connaissent les options de chaque client SQL. Le jury fournissait la commande à lancer et indiquait qu'il était recommandé de taper les commandes `.header on` et `.mode column` pour avoir des résultats de commandes plus lisibles.
Enfin le jury conseille de connaître la construction `COUNT(DISTINCT v)` car elle peut souvent simplifier l'écriture des requêtes (qui pouvaient s'écrire autrement dans tous les sujets posés).

8.5 Commentaires liés au langage C.

Le jury a été favorablement surpris par la bonne maîtrise du langage C. Voici quelques précisions pour une meilleure préparation.

Gestion de la mémoire.

Certains candidats ont du mal avec `malloc`. Dans les erreurs récurrentes que les candidats ont commises : oubli de `sizeof` (et donc mémoire allouée trop petite), tentatives d'appels à `malloc` en dehors de toute fonction (pour des variables globales), quelques candidats qui ne savent pas allouer un tableau 1D, d'autres, plus nombreux, qui ont des problèmes avec les tableaux 2D (que ce soit des tableaux de tableaux ou des tableaux linéarisés).

Compilation.

Le jury recommande fortement aux candidats de compiler avec les options `-Wall -Wextra` et `-fsanitize=address` car cela permet d'attraper diverses erreurs et facilite le débuggage. Quand il n'y a qu'un seul fichier à compiler, le jury déconseille de compiler en deux étapes (fichier objet puis exécutable) car cela fait perdre du temps aux candidats, et ce d'autant plus que certains candidats retapent entièrement chaque commande dans le shell (plutôt que de rechercher avec ctrl+r ou de relancer une commande précédente avec flèche du haut). Attention à l'utilisation de `-fsanitize=address` si cette option permet de détecter des bugs, elle affichera une sortie qui ressemble à une erreur en cas de mémoire non libérée, il est préférable que les candidats aient l'habitude de ces options.

Libération de la mémoire et valeurs de retour des fonctions de la bibliothèque.

Le jury ne demande pas de libérer la mémoire et il ne faut pas que les candidats perdent du temps à le faire (sauf demande explicite dans le sujet ou par l'examinateur). De la même manière la vérification des valeurs de retour des fonctions (comme fopen) n'est pas attendue.
Le jury n'a pas pris en compte la qualité du code (pertinence des noms de variables et fonctions, commentaires, tests et assert, lisibilité) dans la note. Les candidats qui écrivent du code de mauvaise qualité rendent très difficile l'aide de l'examinateur à débugguer du code voire empêche les candidats eux-même de débugguer.

8.6 Commentaires liés au langage OCaml.

La maîtrise du langage OCaml par les candidats est assez bonne mais le jury formule quelques conseils et remarques pour les candidats.

Utilisation simple de la bibliothèque standard.

Peu de candidats utilisent les fonctions "de base" sur la manipulation de listes (comme `filter`, `map`, `iter`, `exists`) qui sont au programme. Le jury ne pénalise pas leur non-usage mais les connaître permet souvent d'écrire du code plus court donc plus rapide à écrire et plus simple à relire. Le jury a aussi vu très peu d'utilisation des tables de hachage (par exemple pour détecter des doublons).

Utilisation avancée de la bibliothèque.

Seule une petite partie de la bibliothèque standard OCaml est au programme et seule cette partie est exigible mais cela ne veut pas dire que les candidats doivent s'interdire toute autre fonction. Les candidats ayant une installation standard d'OCaml avec la documentation complète, le jury souhaite leur laisser accès à ces bibliothèques pour ne pas pénaliser ceux qui en ont l'habitude mais il ne souhaite pas non plus que la connaissance des fonctionnalités avancées de la bibliothèque avantage certains candidats.

Il y a, par exemple, dans la bibliothèque standard les `Set`. Ces `Set` peuvent servir d'arbres binaires équilibrés ou de files de priorité. Si une question demande d'écrire un arbre binaire de recherche équilibré, un candidat peut répondre avec `Set`. Si un candidat veut utiliser de telles fonctions avancées, il convient d'abord demander à l'examinateur.

Le jury a constaté l'utilisation de nombreuses fonctions de la bibliothèque hors programme comme `List.mapi`, `List.assoc`, `Array.of_list` ou même `String.split_on_char` et le jury a considéré toutes ces utilisations acceptables du moment que les candidats pouvaient rapidement les ré-implémenter. Le jury ne conseille pas spécialement l'apprentissage de ces fonctions qui seraient données par le Jury en cas de besoin. Cela étant, connaître, par exemple, la structure de liste associative ou l'idée de décomposer un problème de parsing en un découpage de la chaîne en token pour ensuite traiter chaque token peuvent aider les candidats.

Attention tout de même à l'utilisation de la bibliothèque. De multiples candidats qui ont un style de programmation impératif et utilisent beaucoup, par exemple, `List.nth` sans faire attention à la complexité de cette opération ! Quand les candidats utilisent une fonction de la bibliothèque dans un algorithme, ils doivent être capables de donner la complexité de l'algorithme.

Des sujets peuvent nécessiter l'utilisation de bibliothèques, auxquels cas les candidats auront à lire la documentation. Certains candidats ont utilisé la documentation pour voir ce qui existait. Par exemple, pour un exercice de parsing certains ont regardé la documentation du module `String`. Tant que les candidats n'utilisent pas des fonctions trop avancées, le jury n'a aucun de problème avec cette pratique. Lire la documentation n'est pas forcément évident et il est donc conseillé d'avoir déjà de l'expérience avec la documentation OCaml.

Style de programmation.

Certains candidats connaissent les `ref` mais pensent qu'il ne faut surtout pas les utiliser en vertu d'une programmation fonctionnelle pure. La pureté du programme écrit n'étant pas notée, c'est dommageable pour les candidats qui perdent du temps à cause de cela ; par exemple, en voulant utiliser une boucle `for` mais sans `ref` ou pour faire un calcul simple sur un tableau.
À l'inverse, certains candidats ne programment qu'en style impératif, et c'est parfois plus compliqué. Par exemple, si le sujet demande d'écrire une fonction qui somme deux nombres en base B représentés sous forme de listes et que les candidats utilisent des boucles `for` et `List.nth` alors cela risque d'avoir un impact sur la lisibilité, la concision, voire la complexité du code résultant.
Pour toutes ces raisons, le jury rappelle que le style de programmation est libre mais conseille d'être capable d'un peu de souplesse sur ce style de programmation en s'adaptant au sujet (par exemple en utilisant plutôt des fonctions récursives sur les listes et plutôt des boucles et des `ref` sur les tableaux).

8.7 Évolutions envisagées pour l'édition 2026.

Aucune évolution importante n'est envisagée pour l'édition 2026 mais il est possible que des logiciels aient été mis à jour (passage de Debian Bookworm à Trixie).

20. Travaux pratiques d'informatique

20.1. Introduction

L'épreuve de travaux pratiques d'informatique en MPI est une épreuve d'algorithmique et de programmation, avec utilisation d'un ordinateur, d'une durée de 3 heures. À partir d'un sujet imposé, elle demande de traiter sur machine des questions de programmation, d'effectuer des choix de modélisation et d'aborder certains aspects théoriques de l'informatique, tout en communiquant très fréquemment avec le jury.

L'objectif de l'épreuve est d'évaluer les capacités de programmation, la maîtrise des méthodes classiques du programme, les capacités de modélisation, d'abstraction et d'inventivité, ainsi que l'application de bonnes pratiques que l'on est en droit d'attendre de futurs ingénieures et ingénieurs.

Organisation de l'oral

Des ordinateurs configurés avec un environnement de travail adapté aux sujets demandés sont fournis par le concours pour cette épreuve. Pour la session 2025, l'environnement était une distribution GNU/Linux Ubuntu 22.04. Un passage vers une distribution Debian 13 est envisagé pour les sessions futures, sans que cela n'ait d'impact majeur sur l'environnement proposé.

Avant le début de l'épreuve, le jury laisse une dizaine de minutes aux candidats afin de prendre en mains librement cet environnement de travail, de bien choisir son éditeur, de vérifier qu'il sait interpréter ou compiler en OCaml et en C et de se familiariser avec la documentation qui est proposée avec Zeal pour les langages OCaml/C/SQL. Pendant ce temps, qui n'entre pas en compte dans l'évaluation, il est possible de poser des questions techniques sur l'environnement aux membres du jury, de s'échauffer en écrivant de petits programmes simples ou encore de préparer des fonctions utilitaires générales.

Les sujets sont fournis en version papier ainsi qu'en version numérique au format pdf. Les sujets sont généralement accompagnés de fichiers auxiliaires, comprenant notamment :

- des fichiers sources, complets ou à compléter selon les cas ;
- des fichiers de données à exploiter ;
- des scripts d'aide à la compilation.

Ces fichiers sont mis à disposition dans un dossier de travail spécifique à chaque candidat, dont l'accès est déverrouillé grâce à un code donné en début d'épreuve par le jury. Les candidats peuvent travailler directement dans ce dossier et il ne leur est demandé aucun transfert de fichier.

Les candidats sont responsables des sauvegardes de leur travail, qu'ils doivent effectuer à intervalles réguliers.

Les sujets sont organisés en une suite de questions, à traiter généralement de façon linéaire, sauf mention explicite du contraire. Les questions sont réparties en trois catégories :

- des questions de programmation à réaliser sur la machine ;
- des questions à préparer pour une présentation orale ;
- des résultats à reporter ou des questions de rédaction à réaliser sur un compte rendu.

Le compte rendu sert uniquement de support pendant l'interaction orale et est évalué par le jury directement pendant l'épreuve. On n'attend pas une rédaction détaillée comme on pourrait en trouver sur une copie à l'écrit, d'éventuelles imprécisions pouvant être levées par la discussion orale.

Comme indiqué dans le rapport précédent, ni le compte rendu, ni le code source ne font l'objet d'une évaluation en tant que telle après l'épreuve. Ceux-ci peuvent cependant être consultés a posteriori lorsque de nouveaux éléments y ont été apportés entre la dernière interaction avec le jury et la toute fin de l'épreuve. En effet, le jury effectue un dernier passage auprès des candidats un peu avant la fin de l'épreuve, mais ne peut interroger tous les candidats simultanément. Il arrive donc que certaines questions soient traitées sans que le jury n'ait eu le temps de les évaluer pendant la séance, auquel cas il consulte le compte rendu et les programmes sources pour finaliser son évaluation.

Le concours fournit toujours les feuilles et les brouillons utiles à l'épreuve. Les candidats doivent se munir du matériel d'écriture usuel (stylos, crayons, gomme, règle). Aucun autre matériel n'est autorisé. Pour certaines applications numériques demandées par les sujets, les candidats ont convenablement réussi à utiliser les fonctions de calcul des ordinateurs.

Les sujets proposés suivent tous le format des épreuves en vigueur depuis la session 2023. Ce format permet d'évaluer correctement le niveau des candidats et il n'est pas prévu d'évolution majeure pour la session 2026.

Programme des épreuves orales

Les sujets posés permettent d'évaluer l'ensemble des notions présentes dans les programmes d'informatique des classes de MP2I et de MPI. Chaque sujet porte sur une ou plusieurs parties spécifiques de ces programmes. L'ensemble de tous les sujets couvrait globalement l'intégralité du programme des deux années.

Les langages évalués sont C, OCaml et SQL. Les sujets peuvent utiliser, selon la pertinence par rapport au thème abordé, un seul, deux ou trois langages. Même si aucun sujet n'évalue uniquement le langage SQL, la partie consacrée à ce langage peut constituer une part importante du barème et il est impératif de ne pas faire une impasse totale sur ce langage. La très grande majorité des sujets évalue au minimum deux langages. Le langage à utiliser pour chaque question est quasiment tout le temps imposé, mais certaines parties ont parfois été laissées librement au choix des candidats.

Évaluation des épreuves orales

Les sujets proposés restent volontairement longs. Les candidats les plus efficaces ont pu tout de même aborder la quasi intégralité de leur contenu. Les sujets comportent tous des parties d'application immédiate du cours, assez guidées, destinées à évaluer les compétences de base en programmation et les connaissances de cours, ainsi que des parties d'ouverture favorisant la prise d'initiative et permettant d'évaluer l'autonomie et la capacité de prise de recul des candidats.

L'évaluation de l'épreuve ne consiste cependant pas exclusivement, et loin de là, en une mesure du plus grand nombre de questions traitées. Le jury a valorisé non seulement l'efficacité en termes de programmation, mais également l'intérêt porté à la discipline de programmation (compiler fréquemment, s'assurer que le code fonctionne, proposer spontanément des tests). L'épreuve de travaux pratiques d'informatique doit mener à un programme qui fonctionne effectivement. Le jury préfère des prestations avec un peu moins de questions traitées, mais dans lesquelles les programmes écrits avaient été correctement compilés et soigneusement testés.

20.2. Analyse globale des résultats

Les prestations des candidats sont dans leur très large majorité d'excellente facture, avec encore une hausse du niveau par rapport à la session 2024. Les exigences d'évaluation et de notation ont donc été revues à la hausse pour obtenir une moyenne et un écart-type cibles, de manière à pouvoir correctement classer les candidats. Il ne faut donc pas voir une baisse de niveau dans la diminution notable de la moyenne de l'épreuve entre la session 2025 et la session 2024.

Le niveau observé en programmation est très satisfaisant et la syntaxe des langages est généralement bien maîtrisée. Les connaissances de cours sont globalement complètes, même si le jury constate toujours quelques faiblesses récurrentes qui restent très discriminantes dans l'évaluation. Le jury insiste encore et toujours sur le fait que le contenu du cours de MP2I et de MPI doit être parfaitement connu, au point d'être restitué efficacement. Les sujets continuent de proposer des questions de cours, sans lequel il est vraiment difficile d'aborder correctement les questions de programmation qui suivent.

Les candidats semblent encore mieux préparés à l'épreuve que précédemment, en maîtrisant quasiment toujours les attendus et les modalités de l'épreuve. Dans la très grande majorité des cas, les compétences visées par les programmes de MP2I et MPI sont acquises, les aspects théoriques sont maîtrisés et la technique est manipulée avec efficacité. Le jury adresse, encore une fois, ses félicitations à l'ensemble des candidats et à leurs enseignants.

20.3. Commentaires sur les réponses apportées et conseils aux futurs candidats

Maitrise du cours

Le jury considère que la maîtrise du cours demeure un élément absolument essentiel : il encourage les candidats à bien apprendre le cours afin de traiter rapidement et efficacement les questions qui s'y rapportent. Les questions de cours restent discriminantes et peuvent être bloquantes pour traiter la suite du sujet. Dans de tels cas, le jury est contraint de faire des rappels de cours très importants, ce qui se ressent nécessairement sur la note obtenue.

Le jury observe toujours des confusions : les candidats confondent des algorithmes de recherche de motifs avec des algorithmes de compression, ou bien confondent les méthodes algorithmiques usuelles (diviser pour régner, programmation dynamique, etc.) sans en connaître vraiment le principe.

Le jury invite à faire preuve de précision dans l'utilisation des noms d'algorithmes au programme, dans une optique d'efficacité de la communication. La simple confusion de noms n'a cependant pas été sanctionnée quand il était clair que le candidat maîtrisait l'algorithme demandé et savait le décrire précisément. À contrario, redonner simplement en tant que mot-clé le nom d'un algorithme sans en connaître le principe n'a pas été valorisé.

Préparation technique

Le jury encourage de nouveau toutes les formations à proposer à leurs élèves, durant leur scolarité, un environnement de travail adapté au programme de MP2I et MPI, permettant au minimum d'aborder les points suivants :

– l'accès à un shell dans un terminal permettant d'invoquer les compilateurs mais également d'autres commandes, et de manière générale permettant de se familiariser avec le fonctionnement d'une ligne de commandes ;

– le fonctionnement d'un système POSIX, en particulier quant à la gestion des processus, des fils d'exécution, des flux standard et leur redirection, éventuellement avec des tubes ;
– la familiarisation avec des outils de compilation usuels (make ou dune). Leur usage fera toujours l'objet d'un rappel et aucune compétence spécifique n'est attendue, mais avoir déjà rencontré ces outils permet une plus grande efficacité.

Remarques concernant OCaml

Le jury maîtrise la bibliothèque standard du langage mais il n'exige pas que les candidats utilisent d'eux-mêmes des fonctions de haut niveau, par exemple celles provenant du module `List`. L'évaluation est par exemple indifférente à l'utilisation, qui n'est ni valorisée ni sanctionnée, de `List.fold_left` (et autres fonctions similaires). Le jury est surtout attentif à la clarté du code produit, à sa correction et à la maîtrise de ce code par les candidats.

Il est pertinent de réfléchir quelques instants à la manière de concevoir un code, pour éviter que le résultat soit inutilement compliqué et soit source d'erreurs difficiles à détecter et corriger.

Les candidats doivent savoir proposer un programme OCaml complet, qui compile dans son intégralité et pas uniquement ligne par ligne dans un REPL. Une aide à la compilation a systématiquement été fournie. Le jury accepte si nécessaire le double point-virgule (`;;`) pour séparer les phrases, même si ce dernier ne fait pas partie du langage à proprement parler ; il le propose aux candidats quand cela aide parfois à mieux cerner certaines erreurs de syntaxe difficiles à situer d'après le message du compilateur. Sur ce dernier point, le jury attire l'attention des candidats sur le fait que `;` est un opérateur binaire et non pas un terminateur d'instruction.

Les candidats doivent savoir identifier les erreurs de syntaxe dans leurs programmes, celles-ci se situent souvent bien avant la ligne à laquelle est indiquée l'erreur.

Remarques concernant C

La gestion de la mémoire est un aspect maîtrisé par de nombreux candidats, mais les oublis d'allocations avec `malloc` et surtout les oublis de libération avec `free` restent trop fréquents. Il est toujours pertinent de s'interroger sur la politique d'allocation de la mémoire, en particulier lorsque le sujet invite à programmer une petite API manipulant une structure de données, dont on se sert ensuite. Avoir une politique claire d'allocation *et de libération* est essentiel.

Lors de la préparation des candidats, il peut être utile d'indiquer comment compiler un programme avec `gcc -g -Wall -fsanitize=address` et d'interpréter la sortie en cas d'erreur de segmentation ou de libération oubliée. La compilation séparée a fait l'objet de rappels, mais le jury souligne que celle-ci doit être reconnue par les candidats qui sont donc supposés être un minimum familiers avec ce procédé.

Remarques concernant SQL

Le niveau de maîtrise est très hétérogène parmi les candidats interrogés. Le jury attire de nouveau l'attention sur le fait que plusieurs candidats semblent avoir fait l'impasse sur ce langage.

Les sujets proposent des requêtes de niveaux différents. Le jury s'attend à ce que les requêtes les plus faciles puissent être traitées par tous les candidats, notamment en ce qui concerne :

– l'usage des fonctions d'agrégation avec ou sans `GROUP BY` ;
– l'usage de jointures ;
– l'usage de `LIMIT` et `OFFSET`.

20.4. Conclusion

Le jury est toujours globalement très satisfait par les prestations des candidats. Il attire encore une fois l'attention sur l'importance de la compréhension et de la maîtrise du cours, sur les dangers de faire l'impasse sur le langage SQL, sur la nécessaire prise de recul quant aux notions abordées pendant l'année et sur l'intérêt d'une pratique très régulière sur machine tout au long du cursus.