Épreuve de NSI au bac en Terminale

Programme officiel

Fichier à télécharger...

Nature de l'épreuve de NSI au bac en Terminale

D'après le Bulletin Officiel n°36 du 30 septembre 2022 :

- **Durée** : 3 heures 30 + 1 heure
- **Coefficient** : 16
- **Format** : L'épreuve terminale obligatoire de spécialité est composée de deux parties :
  - une partie écrite, comptant pour 12 points sur 20,
  - une partie pratique comptant pour 8 points sur 20.

Partie écrite de l'épreuve de NSI au bac en terminale

- **Durée** : 3 heures 30
- **Modalités** : La partie écrite consiste en la résolution de trois exercices permettant d'évaluer les connaissances et les capacités attendues conformément aux programmes de première et de terminale de la spécialité.

  Chaque exercice est noté sur 4 points.

  Le sujet comporte trois exercices indépendants les uns des autres, qui permettent d'évaluer les connaissances et compétences des candidats.

Points du programme évaluables lors de l'épreuve écrite

Référence : Bulletin officiel n°36 du 30 septembre 2022

- Thème 2 – Structures de données
  - Structures de données, interface et implémentation.
  - Vocabulaire de la programmation objet : classes, attributs, méthodes, objets.
  - Listes, piles, files : structures linéaires. Dictionnaires, index et clé.
  - Arbres : structures hiérarchiques. Arbres binaires : nœuds, racines, feuilles, sous-arbres gauches, sous-arbres droits.
- Thème 3 – Bases de données
  - Modèle relationnel : relation, attribut, domaine, clef primaire, clef étrangère, schéma relationnel.
  - Base de données relationnelle.
  - Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données.
- Thème 4 – Architectures matérielles, systèmes d'exploitation et réseaux

  - Gestion des processus et des ressources par un système d'exploitation.
  - Protocoles de routage.

- Thème 5 – Langages et programmation

  - Récursivité.
  - Modularité.
  - Mise au point des programmes. Gestion des bugs.

- Thème 6 – Algorithmique

  - Algorithmes sur les arbres binaires et sur les arbres binaires de recherche.
  - Méthode « diviser pour régner »

Points du programme non évalués à l'écrit

- Thème 1 – Histoire de l'informatique
- Thème 2 – Structures de données

  - Graphes : structures relationnelles. Sommets, arcs, arêtes, graphes orientés ou non orientés.

- Thème 3 – Bases de données

  - Système de gestion de bases de données relationnelles.

- Thème 4 – Architectures matérielles, systèmes d'exploitation et réseaux

  - Composants intégrés d'un système sur puce.
  - Sécurisation des communications.

- Thème 5 – Langages et programmation

  - Notion de programme en tant que donnée. Calculabilité, décidabilité.
  - Paradigmes de programmation

- Thème 6 – Algorithmique

  - Algorithmes sur les graphes.
  - Programmation dynamique.
  - Recherche textuelle.

Partie pratique de l'épreuve de NSI au bac en terminale

- **Durée** : 1 heure
- **Modalités** : La partie pratique consiste en la résolution de deux exercices sur ordinateur, chacun étant noté sur 4 points.

  Le candidat est évalué sur la base d'un dialogue avec un professeur-examinateur.

  Un examinateur évalue au maximum quatre élèves. L'examinateur ne peut pas évaluer un élève qu'il a eu en classe durant l'année en cours.

  L'évaluation de cette partie se déroule au cours du deuxième trimestre pendant la période de l'épreuve écrite de spécialité.

- **Premier exercice**

  Le premier exercice consiste à programmer un algorithme figurant explicitement au programme, ne présentant pas de difficulté particulière, dont on fournit une spécification.

  Il s'agit donc de restituer un algorithme rencontré et travaillé à plusieurs reprises en cours de formation.

  Le sujet peut proposer un jeu de test avec les réponses attendues pour permettre au candidat de vérifier son travail.

- **Deuxième exercice**

  Pour le second exercice, un programme est fourni au candidat.

  Cet exercice ne demande pas l'écriture complète d'un programme, mais permet de valider des compétences de programmation suivant des modalités variées : le candidat doit, par exemple, compléter un programme « à trous » afin de répondre à une spécification donnée, ou encore compléter un programme pour le documenter, ou encore compléter un programme en ajoutant des assertions, etc.