Programme de Terminale NSI

Surligné en jaune : ce qui **n'est pas** au programme pour l'épreuve finale du mois de Mars

1- Histoire de l'informatique :

Contenus	Capacités attendues	Commentaires
Événements clés de l'histoire de l'informatique	Situer dans le temps les principaux événements de l'histoire de l'informatique et leurs protagonistes. Identifier l'évolution des rôles relatifs des logiciels et des matériels.	Ces repères viennent compléter ceux qui ont été introduits en première. Ces repères historiques sont construits au fur et à mesure de la présentation des concepts et techniques.

2- Structures de données :

Contenus	Capacités attendues	Commentaires
Structures de données interface et implémentation.	Spécifier une structure de données par son interface. Distinguer interface et implémentation. Écrire plusieurs implémentations d'une même structure de données.	L'abstraction des structures de données est introduite après plusieurs implémentations d'une structure simple comme la file (avec un tableau ou avec deux piles).
Vocabulaire de la programmation objet : classes, attributs, méthodes, objets.	Écrire la définition d'une classe. Accéder aux attributs et méthodes d'une classe.	On n'aborde pas ici tous les aspects de la programmation objet comme le polymorphisme et l'héritage
Listes piles files : structures linéaires. Dictionnaires index et clé.	Distinguer des structures par le jeu des méthodes qui les caractérisent. Choisir une structure de données adaptée à la situation à modéliser. Distinguer la recherche d'une valeur dans une liste et dans un dictionnaire.	On distingue les modes FIFO (first in first out) et LIFO (last in first out) des piles et des files.
Arbres : structures hiérarchiques. Arbres binaires : nœuds racines feuilles sous-arbres gauches sous-arbres droits.	Identifier des situations nécessitant une structure de données arborescente. Évaluer quelques mesures des arbres binaires (taille encadrement de la hauteur etc.).	On fait le lien avec la rubrique « algorithmique ».
Graphes : structures relationnelles. Sommets arcs arêtes graphes orientés ou non orientés.	Modéliser des situations sous forme de graphes. Écrire les implémentations correspondantes d'un graphe : matrice d'adjacence liste de successeurs/de prédécesseurs. Passer d'une représentation à une autre.	On s'appuie sur des exemples comme le réseau routier le réseau électrique Internet les réseaux sociaux. Le choix de la représentation dépend du traitement qu'on veut mettre en place : on fait le lien avec la rubrique « algorithmique ».

3- Bases de données :

Contenus	Capacités attendues	Commentaires
Modèle relationnel : relation, attribut, domaine, clef primaire, clef étrangère, schéma relationnel.	Identifier les concepts définissant le modèle relationnel.	Ces concepts permettent d'exprimer les contraintes d'intégrité (domaine, relation et référence).
Base de données relationnelle.	Savoir distinguer la structure d'une base de données de son contenu. Repérer des anomalies dans le schéma d'une base de données.	La structure est un ensemble de schémas relationnels qui respecte les contraintes du modèle relationnel. Les anomalies peuvent être des redondances de données ou des anomalies d'insertion, de suppression, de mise à jour. On privilégie la manipulation de données nombreuses et réalistes.
Système de gestion de bases de données relationnelles.	Identifier les services rendus par un système de gestion de bases de données relationnelles : persistance des données, gestion des accès concurrents, efficacité de traitement des requêtes, sécurisation des accès.	Il s'agit de comprendre le rôle et les enjeux des différents services sans en détailler le fonctionnement
Langage SQL : requêtes d'interrogation et de mise à jour d'une base de données.	Identifier les composants d'une requête. Construire des requêtes d'interrogation à l'aide des clauses du langage SQL : SELECT, FROM, WHERE, JOIN. Construire des requêtes d'insertion et de mise à jour à l'aide de : UPDATE, INSERT, DELETE.	On peut utiliser DISTINCT, ORDER BY ou les fonctions d'agrégation sans utiliser les clauses GROUP BY et HAVING.

4- Architecture matérielle, systèmes d'exploitation et réseaux

Contenus	Capacités attendues	Commentaires
Composants intégrés d'un système sur puce	Identifier les principaux composants sur un schéma de circuit et les avantages de leur intégration en termes de vitesse et de consommation.	Le circuit d'un téléphone peut être pris comme un exemple : microprocesseurs, mémoires locales, interfaces radio et filaires, gestion d'énergie, contrôleurs vidéo, accélérateur graphique, réseaux sur puce, etc.
Gestion des processus et des ressources par un système d'exploitation	Décrire la création d'un processus, l'ordonnancement de plusieurs processus par le système. Mettre en évidence le risque de l'interblocage (deadlock).	À l'aide d'outils standard, il s'agit d'observer les processus actifs ou en attente sur une machine. Une présentation débranchée de l'interblocage peut être proposée.
Protocoles de routage.	Identifier, suivant le protocole de routage utilisé, la route	En mode débranché, les tables de routage étant données, on se réfère au nombre de

	empruntée par un paquet.	sauts (protocole RIP) ou au coût des routes (protocole OSPF). Le lien avec les algorithmes de recherche de chemin sur un graphe est mis en évidence.
Sécurisation des communications.	Décrire les principes de chiffrement symétrique (clef partagée) et asymétrique (avec clef privée/clef publique). Décrire l'échange d'une clef symétrique en utilisant un protocole asymétrique pour sécuriser une communication HTTPS.	Les protocoles symétriques et asymétriques peuvent être illustrés en mode débranché, éventuellement avec description d'un chiffrement particulier. La négociation de la méthode chiffrement du protocole SSL (Secure Sockets Layer) n'est pas abordée

5- Langages et programmation :

Contenus	Capacités attendues	Commentaires
Notion de programme en tant que donnée. Calculabilité, décidabilité	Comprendre que tout programme est aussi une donnée. Comprendre que la calculabilité ne dépend pas du langage de programmation utilisé. Montrer, sans formalisme théorique, que le problème de l'arrêt est indécidable.	L'utilisation d'un interpréteur ou d'un compilateur, le téléchargement de logiciel, le fonctionnement des systèmes d'exploitation permettent de comprendre un programme comme donnée d'un autre programme.
Récursivité.	Écrire un programme récursif. Analyser le fonctionnement d'un programme récursif.	Des exemples relevant de domaines variés sont à privilégier.
Modularité.	Utiliser des API (Application Programming Interface) ou des bibliothèques. Exploiter leur documentation. Créer des modules simples et les documenter.	
Paradigmes de programmation.	Distinguer sur des exemples les paradigmes impératif, fonctionnel et objet. Choisir le paradigme de programmation selon le champ d'application d'un programme.	Avec un même langage de programmation, on peut utiliser des paradigmes différents. Dans un même programme, on peut utiliser des paradigmes différents.
Mise au point des programmes. Gestion des bugs. Modularité.	Dans la pratique de la programmation, savoir répondre aux causes typiques de bugs : problèmes liés au typage, effets de bord non désirés, débordements dans les tableaux, instruction conditionnelle non exhaustive, choix des inégalités, comparaisons et calculs entre flottants, mauvais nommage des variables, etc.	On prolonge le travail entrepris en classe de première sur l'utilisation de la spécification, des assertions, de la documentation des programmes et de la construction de jeux de tests. Les élèves apprennent progressivement à anticiper leurs erreurs

6- Algorithmique :

Contenus	Capacités attendues	Commentaires
Algorithmes sur les arbres binaires.	Calculer la taille et la hauteur d'un arbre. Parcourir un arbre de différentes façons (ordres infixe préfixe ou suffixe ; ordre en largeur d'abord).	Une structure de données récursive adaptée est utilisée. L'exemple des arbres permet d'illustrer la programmation par classe.
Algorithmes sur les graphes.	Parcourir un graphe en profondeur d'abord, en largeur d'abord. Repérer la présence d'un cycle dans un graphe. Chercher un chemin dans un graphe.	Le parcours d'un labyrinthe et le routage dans Internet sont des exemples d'algorithme sur les graphes. L'exemple des graphes permet d'illustrer l'utilisation des classes en programmation
Méthode "diviser pour régner"	Écrire un algorithme utilisant la méthode « diviser pour régner ».	La rotation d'une image bitmap d'un quart de tour avec un coût en mémoire constant est un bon exemple. L'exemple du tri fusion permet également d'exploiter la récursivité et d'exhiber un algorithme de coût en $n.\log_2 n$ dans les pires des cas.
Programmation dynamique.	Utiliser la programmation dynamique pour écrire un algorithme.	Les exemples de l'alignement de séquences ou du rendu de monnaie peuvent être présentés. La discussion sur le coût en mémoire peut être développée.
Recherche textuelle.	Étudier l'algorithme de Boyer-Moore pour la recherche d'un motif dans un texte.	L'intérêt du prétraitement du motif est mis en avant. L'étude du coût, difficile, ne peut être exigée.