U2 – Mathématiques pour l'informatique

Le programme est conçu avec l'intention de permettre l'acquisition des bases mathématiques nécessaires à la compréhension et la maîtrise des finalités spécifiques du BTS Services informatiques aux organisations et de démarches mathématiques et algorithmiques permettant d'en appréhender la pertinence et l'efficacité pour évoluer dans un environnement numérique. On distingue trois objectifs principaux :

- comprendre et résoudre les problèmes mathématiques élémentaires auxquels une personne informaticienne est couramment confrontée (calcul binaire, masque de réseau, opérateurs logiques…) ;
- comprendre et manipuler les objets mathématiques fréquemment utilisés en programmation, de manière à pouvoir exploiter informatiquement une solution mathématique préalablement construite ;
- résoudre des problèmes numériques nécessitant la mise en oeuvre d'*algorithmes* qu'il s'agit de construire, de mettre en forme et dont on comparera éventuellement les performances.

D'une manière générale, *la recherche et la mise en oeuvre d'algorithmes* en utilisant les *moyens informatiques* propres à la section sont au centre de cette formation.

De par la spécificité des compétences visées, mais aussi afin d'atténuer les difficultés inhérentes au passage à l'abstraction mathématique, les nouveaux concepts gagnent à être introduits après une observation de phénomènes rendue possible par la pratique de la programmation. Toute démarche mise en oeuvre à l'aide d'un programme informatique ou d'une calculatrice permettant de faciliter la compréhension d'un concept ou d'une méthode en l'illustrant graphiquement, numériquement ou dans un contexte lié à la spécialité doit être privilégiée. L'utilisation large des moyens informatiques (calculatrice et ordinateur), repose sur la connaissance d'éléments de syntaxe et de fonctions spécifiques à l'outil utilisé mais sans exigence de virtuosité.

Pour la programmation, un langage de haut niveau doit être effectivement et régulièrement utilisé par le professeur comme par les élèves. Le langage Python, langage de programmation simple d'utilisation, libre d'installation et largement enseigné est recommandé sans être imposé. Dans le cas où un autre langage est choisi, il convient de s'assurer qu'il est accepté par le centre d'examen. Dans certaines situations où leur utilisation semble plus pertinente, on peut notamment utiliser, un tableur, une calculatrice, un logiciel de calcul formel.

Le programme est constitué des modules suivants décrits par le programme de mathématiques des brevets de technicien supérieur (arrêté du 4 juin 2013) :

- arithmétique ;
- calcul matriciel ;
- algèbres de Boole ;
- savoirs de la théorie des ensembles ;
- graphes et ordonnancement ;
- algorithmique appliquée, à l'exception de l'item « récursivité » et de l'item « analyse d'algorithme ».

Annexe III.A – Grille horaire de formation sous statut scolaire

Première année	Premier semestre (15 semaines)					Deuxième semestre (15 semaines)				
Enseignements	Horaire hebdomadaire				*Volume semestriel (à titre indicatif)*	Horaire hebdomadaire				*Volume semestriel (à titre indicatif)*
	Total étud.	div. [3]	½ div. [4]	Lab. [5]		Total étud.	div. [3]	½ div. [4]	Lab. [5]	
Mathématiques pour l'informatique	3	2	1		*45*	3	2	1		*45*
Enseignements facultatifs										
Mathématiques approfondies	2	2			*30*	2	2			*30*

Deuxième année (24 semaines)					
Enseignements	Horaire hebdomadaire				*Volume annuel (à titre indicatif)*
	Total étud.	div. [3]	½ div. [4]	Lab. [5]	
Mathématiques pour l'informatique	3	2	1		*72*
Enseignements facultatifs					
Mathématiques approfondies	2	2			*48*

[3] Lors des séances en classe entière d'enseignement professionnel, les étudiants doivent pouvoir accéder en tant que de besoin à un environnement informatique.

[4] Lors des séances en demi-division, les étudiants doivent pouvoir accéder en tant que de besoin à un environnement informatique adapté.

[5] Les temps de formation en laboratoire informatique doivent permettre l'accès de chaque étudiante ou étudiant à l'équipement nécessaire à la réalisation des travaux informatiques, individuels et collectifs.

CALCUL MATRICIEL

Ce module consiste en une initiation au langage matriciel, s'appuyant sur l'observation de phénomènes issus de la vie courante ou d'exemples concrets. On cherche principalement à introduire un mode de représentation facilitant l'étude de tels phénomènes.

On introduit le calcul matriciel sur des matrices d'ordre 2. Les calculs sur des matrices d'ordre 3 ou plus sont effectués à l'aide d'une calculatrice ou d'un logiciel.

CONTENUS	CAPACITÉS ATTENDUES	COMMENTAIRES
Matrices Égalité de deux matrices. Matrice nulle, matrice identité.		Une matrice est introduite comme un tableau de nombres réels permettant de représenter une situation comportant plusieurs « entrées » et « sorties ».
Calcul matriciel élémentaire : – addition ; – multiplication par un nombre réel ; – multiplication.	• Effectuer des calculs matriciels à l'aide d'une calculatrice ou d'un logiciel, y compris le calcul d'une puissance d'une matrice.	Le choix de la définition de chaque opération portant sur les matrices s'appuie sur l'observation de la signification du tableau de nombres ainsi obtenu. On signale le caractère associatif mais non commutatif de la multiplication.
	• Représenter puis traiter une situation simple à l'aide d'une écriture matricielle.	On peut notamment étudier des exemples de processus discrets, déterministes ou stochastiques, à l'aide de suites de matrices.
Inverse d'une matrice Définition, existence éventuelle, unicité en cas d'existence.		
Commutativité d'une matrice inversible et de son inverse.	• Montrer qu'une matrice est l'inverse d'une autre.	
	• Déterminer à l'aide d'une calculatrice ou d'un logiciel l'inverse d'une matrice inversible.	La notion de déterminant n'est pas au programme. Aucune condition d'inversibilité d'une matrice n'est à connaître.
	• Résoudre un système linéaire de n équations à n inconnues à l'aide d'une inversion de matrice.	On ne considère que le cas où le système est de Cramer, sans qu'aucune justification ne soit requise. ⇆ Gestion d'un réseau, matrice d'inertie et changement de base en mécanique, processus aléatoires.

ARITHMÉTIQUE

Le programme concerne les notions les plus utiles à l'informatique. La numération est indispensable aux langages de bas niveau. L'arithmétique modulaire est utile à la cryptographie, aux corrections d'erreurs et plus généralement à de nombreux algorithmes.

CONTENUS	CAPACITÉS ATTENDUES	COMMENTAIRES
Systèmes de numération		
Numération en bases 10, 2 et 16 des entiers et des réels. Conversions entre ces bases.	• Passer de l'écriture d'un nombre dans une base à une autre.	Les nombres négatifs sont précédés du signe moins (–), quelle que soit la base utilisée. On fait le lien entre le calcul binaire et le calcul booléen : les booléens sont alors 1 et 0, interprétés comme signifiant « il y en a, ou pas ».
Notions d'arrondi et de précision.	• Arrondir un entier ou un réel (par défaut, par excès, au plus près…). • Se conformer à un nombre de chiffres significatifs.	On se limite à des cas simples en base 10 et en base 2. On ne fait aucune théorie sur les calculs d'incertitude.
Addition, soustraction, multiplication et division des entiers naturels.	• Calculer à la main : – des additions en bases 2 et 16 ; – des multiplications et des divisions par une puissance de deux, en base 2.	
Arithmétique modulaire		On évite tout excès de technicité en s'efforçant d'utiliser des présentations concrètes.
Division euclidienne : quotient, reste, existence, unicité.		On se limite aux entiers naturels.
Nombres premiers, décomposition en produit de facteurs premiers, entiers premiers entre eux, PGCD de deux entiers.	• Décomposer un entier naturel en produit de facteurs premiers et déterminer tous ses diviseurs. • Mettre en œuvre un algorithme : – de recherche de nombres premiers ; – de décomposition en produit de facteurs premiers.	Aucune technique n'est censée être connue.

Congruences. Compatibilité avec l'addition et la multiplication. Propriété : modulo n, les multiples de a sont les multiples de PGCD(a, n).	• Mener un calcul de congruence modulo n. • Parcourir une liste circulaire par sauts d'amplitude constante	On montre l'efficacité du langage des congruences. On note que le parcours n'est exhaustif que quand la longueur du saut et la taille de la liste sont des entiers premiers entre eux.

ALGÈBRES DE BOOLE

1. CALCUL DES PROPOSITIONS ET DES PRÉDICATS

L'objectif est d'introduire quelques éléments de logique en liaison avec l'enseignement de l'informatique. Il s'agit d'une brève étude destinée à familiariser les étudiants à une pratique élémentaire du calcul portant sur des énoncés.

CONTENUS	CAPACITÉS ATTENDUES	COMMENTAIRES
Calcul propositionnel Proposition, valeur de vérité. Connecteurs logiques : - négation (non P , $\neg P$, $\overline{P}$) ; - conjonction (P et Q , $P \wedge Q$) ; - disjonction (P ou Q , $P \vee Q$) ; - implication ; - équivalence.	• Traiter un exemple simple de calcul portant sur un énoncé. • Utiliser des connecteurs logiques pour exprimer une condition.	On dégage les propriétés fondamentales des opérations introduites, de manière à déboucher ensuite sur un exemple d'algèbre de Boole. En situation, on aborde les lois de Morgan. On se limite au cas où l'utilisation d'une table de vérité ou de propriétés élémentaires du calcul propositionnel permet de conclure sans excès de technicité. Cette capacité est également mise en œuvre en algorithmique.
Calcul des prédicats Variable, constante. Quantificateurs $\forall$, $\exists$. Négation de $\forall x$, *p(x)* ; négation de $\exists x$, *p(x)*.	• Passer du langage courant au langage mathématique et inversement. • Exprimer, dans un cas simple, la négation d'un prédicat.	On se limite à des cas simples de prédicats portant sur une, deux ou trois variables. On met en valeur l'importance de l'ordre dans lequel deux quantificateurs interviennent.

2. LANGAGE ENSEMBLISTE

Sans développer une théorie générale des ensembles, l'objectif est de consolider et de prolonger les acquis des étudiants sur les ensembles en liaison avec l'enseignement de l'informatique.

CONTENUS	CAPACITÉS ATTENDUES	COMMENTAIRES
Langage ensembliste Ensemble, appartenance, inclusion, ensemble vide. Ensemble $P(E)$ des parties d'un ensemble E.		
Complémentaire d'une partie, intersection et réunion de deux parties.	• Traiter un exemple simple de calcul portant sur des ensembles finis.	On dégage les propriétés fondamentales des opérations ainsi introduites, de manière à déboucher ensuite sur un exemple d'algèbre de Boole. En situation, on aborde les lois de Morgan.
Ensemble des éléments x d'un ensemble E satisfaisant à une proposition $p(x)$.		On interprète en termes ensemblistes l'implication, la conjonction et la disjonction de deux propositions, ainsi que la négation d'une proposition.

3. CALCUL BOOLÉEN

Cette brève étude est à mener en coordination étroite avec l'enseignement de l'informatique. Il convient d'introduire la notion d'algèbre de Boole à partir des deux exemples précédents. Il s'agit essentiellement d'effectuer des calculs permettant de simplifier des expressions booléennes.

CONTENUS	CAPACITÉS ATTENDUES	COMMENTAIRES
Calcul booléen		On adopte les notations usuelles $\overline{a}$, $a + b$ et ab.
Algèbre de Boole : – définition ; – propriétés des opérations, lois de Morgan.	• Mener des calculs portant sur des variables booléennes. • Simplifier une expression booléenne en utilisant : – un tableau de Karnaugh ; – les règles de calcul booléen. • Passer d'une situation donnée à une expression booléenne correspondante et inversement.	On se limite à des cas simples, comportant au plus trois variables booléennes, pour lesquels on peut conclure sans excès de technicité. On signale l'intérêt des connecteurs non-ou (nor) et non-et (nand), ou exclusif oux (xor).

ÉLÉMENTS DE LA THÉORIE DES ENSEMBLES

Ce module vient compléter, concernant les ensembles, celui relatif aux algèbres de Boole. Il développe les notions de produit cartésien, de relation et d'application en liaison avec les nombreuses utilisations qui en sont faites en informatique (codage, tri, compression…).

CONTENUS	CAPACITÉS ATTENDUES	COMMENTAIRES
Éléments de la théorie des ensembles		Les exemples utilisés sont choisis principalement en liaison avec l'enseignement de l'informatique.
Produit cartésien de deux ensembles : – définition ; – cardinal de $E\times F$ dans le cas où E et F sont finis.	• Déterminer et dénombrer les éléments du produit cartésien de deux ensembles finis.	On généralise au cas du produit cartésien de n ensembles finis.
Relations binaires : – définition ; – propriétés ; – relations d'équivalence, relations d'ordre.	• Traiter un exemple où les contraintes se traduisent en termes de relation d'ordre ou d'équivalence.	On évite un trop grand formalisme. On ne s'intéresse qu'aux utilisations en informatique.
Application f d'un ensemble E dans un ensemble F : – définition ; – image d'une partie A de E ; – image réciproque d'une partie B de F.	• Déterminer l'image ou l'image réciproque d'une partie finie par une application.	
Injection, surjection, bijection.	• Traiter un exemple où les contraintes se traduisent en termes d'injection, de surjection ou de bijection.	On attache plus d'importance à une caractérisation textuelle qu'à l'énoncé de prédicats. On souligne l'importance de la notion d'injection pour coder des informations.
Composition d'applications.	• Écrire une application sous forme de composée. • Traiter un exemple de composition d'applications toutes deux soit injectives, soit surjectives, soit bijectives.	On souligne le fait que la composition d'applications n'est pas une opération commutative. On privilégie les situations issues des autres enseignements.

GRAPHES ET ORDONNANCEMENT

1. GRAPHES

L'objectif est d'introduire et de mettre en œuvre, dans des situations concrètes très élémentaires et sans théorie générale, des algorithmes permettant de résoudre les problèmes figurant dans la colonne « Capacités attendues ».

CONTENUS	CAPACITÉS ATTENDUES	COMMENTAIRES
Graphes		
Modes de représentation d'un graphe fini simple orienté : représentation géométrique, tableau des successeurs ou des prédécesseurs, matrice d'adjacence booléenne.	• Passer d'un mode de représentation à un autre, pour un graphe donné.	La définition d'un graphe fini simple orienté est limitée à la donnée d'un ensemble de sommets et d'un ensemble d'arcs.
Chemin d'un graphe : définition , longueur, circuit, boucle, chemin hamiltonien.		
Puissances entières et booléennes de la matrice d'adjacence.	• Obtenir et interpréter, pour une matrice d'adjacence M donnée, les coefficients : – d'une puissance entière de M ; – d'une puissance booléenne de M.	On considère uniquement le cas d'un graphe non valué (non pondéré). À partir d'exemples très élémentaires et sans introduire une théorie générale, on montre l'intérêt des méthodes matricielles mettant en œuvre l'addition et la multiplication booléennes des matrices d'adjacence.
	• Mettre en œuvre un algorithme permettant d'obtenir les chemins de longueur p d'un graphe.	
Fermeture transitive d'un graphe.	• Mettre en œuvre un algorithme permettant d'obtenir la fermeture transitive d'un graphe.	
Pour un graphe sans circuit : niveau d'un sommet, niveaux du graphe.	• Mettre en œuvre un algorithme permettant d'obtenir les niveaux dans un graphe sans circuit.	Il convient de savoir déterminer les niveaux, sans qu'aucune méthode ne soit imposée.
	• Représenter géométriquement un graphe en l'ordonnant par niveaux.	
Arborescence.		La notion de connexité étant hors programme, on se limite à la présentation d'exemples simples d'arborescences à partir de leur représentation géométrique, sans recherche d'une caractérisation générale.

Chemin optimal en longueur. Graphe valué (pondéré) : – définition ; – chemin optimal en valeur.	• Mettre en œuvre un algorithme permettant d'obtenir une optimisation d'un graphe : – en longueur ; – en valeur (graphe valué).	On observe l'importance du résultat : tout sous-chemin d'un chemin optimal est optimal. On fait une simple présentation des graphes valués, sans théorie particulière.

2. ORDONNANCEMENT

L'objectif est double : sensibiliser l'étudiant aux problèmes d'ordonnancement et traiter manuellement un algorithme. Aucune justification théorique des algorithmes utilisés n'est au programme. On abordera MPM ou PERT. On s'attachera surtout à la compréhension des mécanismes. Et, les cas traités resteront suffisamment modestes pour que la rapidité ne soit pas un critère d'évaluation fondamental.

CONTENUS	CAPACITÉS ATTENDUES	COMMENTAIRES
Ordonnancement Ordonnancement : – méthode MPM ou méthode PERT, principe de représentation ; – dates au plus tôt, au plus tard ; – tâches et chemins critiques ; – marge totale, libre, certaine.	• Résoudre un problème d'ordonnancement en mettant en œuvre la méthode des potentiels métra (MPM) ou la méthode PERT, et interpréter les résultats obtenus à travers les notions abordées. • Reconnaître une contrainte non incluse dans la modélisation et en tenir compte lors de l'interprétation.	On présente quelques cas concrets simplifiés et on les interprète. Aucune autre compétence théorique n'est requise. On se limite à des cas très simples où l'interprétation ne soulève aucune difficulté théorique.

ALGORITHMIQUE APPLIQUÉE

L'objectif de ce module est de construire des algorithmes, et de les programmer dans un langage informatique, dans le but de résoudre des problèmes de niveau raisonnable.
Les thématiques abordées lors de l'étude de ce module sont très ouvertes, mais l'objectif visé, à l'intérieur du processus de conception, est ciblé. On veille à ce que les situations proposées soient mathématiquement achevées. Alors qu'une solution, peut être décrite de manière très libre, textuelle ou graphique, par formules ou symboles, par l'exemple ou de manière inachevée, on s'attache ici à les exprimer en utilisant les outils algorithmiques usuels, pour les rendre directement convertibles et exécutables sur machine.
Afin de faciliter la compréhension des mécanismes et la détection d'éventuelles erreurs, il est impératif de concrétiser les algorithmes par l'emploi d'un langage de programmation et de conduire l'étudiant à réaliser des tests. La tâche inverse, consistant à comprendre un algorithme, présente également un grand intérêt pour l'assimilation des mécanismes et lors d'opérations de maintenance.
Les sujets empruntés à la vie courante peuvent être utilisés à chaque fois qu'ils permettent d'illustrer un mécanisme simple avec pertinence. Sinon, on préfère utiliser des sujets dérivés directement d'autres modules mathématiques et, avec un certain équilibre, des sujets associés à des thématiques informatiques (par exemple : codage, cryptage et décryptage, redondance de sécurité, tri itératif et tri récursif, parcours de graphes). Ces sujets peuvent être abordés afin d'illustrer des concepts fondamentaux d'algorithmique sans qu'aucune connaissance spécifique ne soit exigible de l'étudiant dans ces derniers domaines.

Ce module vise à développer les compétences spécifiques suivantes :

– comprendre un algorithme et expliquer ce qu'il fait ;
– modifier un algorithme existant pour obtenir un résultat différent ;
– concevoir une procédure, un algorithme simple ;
– transcrire un algorithme dans un langage informatique ;
– s'interroger sur l'efficacité d'un algorithme.

Les contrôles d'exécution constituent le cœur des mécanismes algorithmiques de base. À ce titre, on attache un soin tout particulier à leur étude progressive mais détaillée. Leur maîtrise pratique est essentielle et les évaluations doivent être centrées sur eux.

Pour l'écriture des algorithmes, une représentation textuelle convenablement indentée avec des indicateurs de début et de fin explicites facilite la lecture. Pour aider à la compréhension, il est utile également d'indiquer un en-tête composé d'un nom, d'un rôle, de l'indication des données d'entrée et de sortie. Des commentaires sont ajoutés, notamment pour préciser le rôle des variables ou fournir des indications méthodologiques.

Un algorithme est indépendant de tout langage de programmation ; aussi aucun langage n'est imposé, mais il convient de s'assurer qu'il est accepté par le centre d'examen. On privilégie un langage de programmation simple d'utilisation et libre d'installation. L'existence d'une communauté d'utilisateurs et de bibliothèques facilite le développement.

La présentation linéaire du programme, avec une entrée par les contenus n'indique pas d'ordre dans sa mise en œuvre. Aussi les concepts fondamentaux (algorithme, finitude, modularité, identifiant, type, constante, variable, fonction, procédure, expression numérique, expression conditionnelle et plus généralement booléenne…) sont acquis par l'usage, sans faire appel à des définitions formelles préalables.

CONTENUS	CAPACITÉS ATTENDUES	COMMENTAIRES
Types de données Types simples : entier naturel, entier relatif, réel, booléen. Chaîne de caractères.	• Connaître les modes de codage et gérer les différences entre données mathématiques et informatiques : – domaine de valeurs ; – représentation exacte ou approchée, précision.	Il n'est pas nécessaire de connaître la représentation exacte des données en machine, notamment en ce qui concerne les flottants. On évite de considérer une chaîne comme un tableau de caractères.
Tableaux de données : – de type homogène à une ou deux dimensions ; – à deux dimensions dans lequel, soit les lignes soit les colonnes, peuvent être de types différents. Procédure et fonction : – paramètres d'entrée ; – valeur(s) retournée(s) par une fonction ; – variables globales ou locales.	• Construire un tableau. • Traiter les données d'un tableau : – accéder à ses différents éléments en lecture et en écriture ; – traiter les éléments d'une ligne ou d'une colonne ; – copier un tableau. • Gérer les transferts en entrée seule, en sortie seule, en entrée et sortie. • Utiliser les variables globales et locales.	On adapte la construction et l'exploitation de ces tableaux aux possibilités de l'outil informatique utilisé. Les structures de données et les objets ne sont pas au programme de mathématiques : ils figurent dans ceux des enseignements professionnels. Sans aborder la programmation objet, les concepts de modularité et d'interface doivent être connus.
Instructions élémentaires Lecture, écriture. Affectation, affectation récursive. **Opérateurs** Opérateurs numériques : addition, soustraction, multiplication, division, exponentiation, quotient et reste de la division entière, signe. Fonctions mathématiques usuelles.	• Saisir une donnée depuis le clavier, manipuler les variables et afficher sur l'écran. • Gérer la chronologie des valeurs contenues dans les variables et produire la trace d'un algorithme. • Transformer une expression mathématique en expression numérique en ligne et réciproquement.	Les fichiers ne sont pas au programme de mathématiques. En standard, la mise en forme des affichages est limitée à l'utilisation de séparateurs et de changements de ligne. Sinon, les outils nécessaires sont fournis et décrits. La gestion des pointeurs n'est pas au programme. « Fonctions mathématiques usuelles » doit être entendu au sens informatique et inclure les fonctions d'arrondi, ainsi qu'un générateur de nombres pseudo-aléatoires uniforme dans un intervalle.

Opérateurs de comparaison : =, <> ou !=, <, <=, >, >=. Opérateurs booléens : non, et, ou, oux.	• Traduire la condition (éventuellement composée) d'une itération (tant que / répéter jusqu'à ce que) ou d'une alternative (si) sous forme d'expression booléenne.	On compare soit des valeurs numériques, soit des chaînes de caractères.
Opérateurs booléens bit à bit.	• Appliquer des opérations booléennes sur les bits.	On interprète notamment en termes de masque, de mise à un, de mise à zéro, de changement d'état.
Opérateur de chaînes : concaténation. Fonctions permettant l'extraction en début, milieu ou fin, la recherche d'un motif.	• Construire et manipuler des chaînes et construire les messages affichés à l'écran.	L'usage d'expressions régulières simples est possible, mais l'étude des expressions régulières est hors programme.
Transtypage	• Gérer les différents types de données et effectuer des conversions d'un type vers un autre.	D'autres instructions, fonctions ou procédures peuvent être introduites dans l'écriture d'algorithmes. Les descriptions sémantiques et syntaxiques précises sont alors mises à disposition de l'étudiant.
Structures de contrôle et d'exécution. Exécution séquentielle. Exécution à structure conditionnelle (si-alors-sinon). Exécution à structure itérative (pour) et (tant que / répéter jusqu'à ce que).	• Mettre en œuvre ces structures.	Afin d'en maîtriser le fonctionnement, les structures d'exécution sont elles-mêmes présentées sous forme d'algorithmes.
Construction des structures itératives : raisonnement par récurrence, initialisation, mise à jour itérative, calcul itératif, mise en forme finale.	• Gérer une itération en distinguant la préparation, l'itération elle-même et la mise en forme finale.	Le raisonnement par récurrence n'a pas à être évalué pour des démonstrations. Il est introduit pour servir de base à une construction des itérations. Le calcul itératif est souvent récursif

Symboles $\sum_{i=}$ et $\prod_{i=}$, traduction algorithmique.	• Gérer une somme ou un produit d'un nombre variable d'opérandes dépendant d'un paramètre.	Généralement la variable affectée récursivement est initialisée à l'élément neutre de l'opération utilisée, avant d'entrer dans l'itération.
Structures imbriquées	• Gérer des structures imbriquées.	On traite également des exemples où les éléments de contrôle des structures internes dépendent de ceux des structures externes. Le nombre d'imbrications n'est pas limité, sauf pour les itérations en dépendance, où on se limite à deux. On évite les excès de complexité, ainsi que les constructions ne correspondant pas à un besoin concret.
	• Traiter un produit matriciel.	La formule de calcul des coefficients d'un produit est donnée.
Récursivité. Nécessité d'un test. Nécessité de cas particuliers résolus sans appel à la récursivité. Finitude.	• Mettre en œuvre ou exploiter une fonction ou une procédure récursive simple.	On peut traiter des exemples de récursivité terminale, de conversion en algorithme non récursif, de récursivité non terminale. On ne présente pas de récursivité mutuelle entre plusieurs procédures.
Analyse d'algorithmes		Ce paragraphe se veut être une simple sensibilisation aux notions de complexité, de correction, de recherche d'erreur et de finalité d'un algorithme. En cas d'évaluation portant sur l'un de ces thèmes, on apporte suffisamment d'indications pour limiter les prérequis.
Notions de complexité temporelle et spatiale.	• Calculer un nombre minimal ou maximal d'opérations significatives ou la taille globale de données choisies.	On présente des variantes produisant les mêmes résultats avec des complexités très différentes. Aucune connaissance théorique n'est exigible.
Validation et débogage.	• Procéder à un suivi de variables par la production d'une trace ou l'utilisation de jeux d'essai pour le test d'un algorithme et la recherche d'erreur.	On propose des algorithmes délibérément erronés dont les défauts sont repérés puis débogués. Selon le langage choisi, on peut montrer les fonctions permettant

		le suivi de variables, le débogage pas à pas**.** Afin de mieux sensibiliser à certains risques, on peut présenter des cas d'effets indésirables (effets de bord, évaluation partielle lors de calcul d'expressions booléennes, débordements ou approximations numériques, transtypage, utilisation d'indices hors domaine,…) et leurs conséquences spectaculaires. Aucune théorie n'est au programme.
Interprétation d'algorithmes.	• Savoir reconstituer le rôle d'un algorithme.	L'ajout d'une ou plusieurs procédures ou fonctions à un ensemble interdépendant peut constituer une excellente base. On se limite à des cas simples.