# Mastère Spécialisé
## Informatique et Langages

tech université technologique

tech université technologique

# Mastère Spécialisé
## Informatique et Langages

- Modalité: **en ligne**
- Durée: **12 mois**
- Qualification: **TECH Université Technologique**
- Intensité: **16h/semaine**
- Horaire: **à votre rythme**
- Examens: **en ligne**

Accès au site web: www.techtitute.com/fr/informatique/master/master-informatique-langages

# Sommaire

01

# Présentation

Les professionnels de l'informatique doivent maintenir leurs compétences à jour afin de pouvoir continuer à opérer dans leur domaine de compétence de manière optimale, sans passer à côté des avancées qui sont intégrées dans ce domaine à un rythme effréné. Cette mise à jour est conçue pour fournir aux étudiants une connaissance complète et approfondie des connaissances essentielles et des innovations les plus intéressantes en matière de conception d'algorithmes dans le développement de projets informatiques, en utilisant les méthodes les plus innovantes et efficaces du secteur.

*Acquérir les connaissances fondamentales de l'informatique et savoir comment les appliquer avec succès dans le développement de projets informatiques, dans le cadre d'un Mastère Spécialisé hautement compétent"*

Ce Mastère Spécialisé se concentre sur les principes fondamentaux de la programmation et de la structure des données, des algorithmes et de la complexité, ainsi que sur la conception avancée d'algorithmes, la programmation avancée, les processeurs de langage et l'infographie, entre autres aspects liés à ce domaine de l'informatique.

Ce Mastère Spécialisé fournit aux étudiants des outils et des compétences spécifiques pour développer avec succès leur activité professionnelle dans le vaste environnement de l'informatique et des Langages. Il travaille sur des compétences clés telles que la connaissance de la réalité et de la pratique quotidienne dans différents domaines informatiques et développe la responsabilité dans le suivi et la supervision de leur travail, ainsi que des compétences spécifiques dans ce domaine.

De plus, comme il s'agit d'un Mastère Spécialisé 100% en ligne, l'étudiant n'est pas conditionné par des horaires fixes ou la nécessité de se déplacer vers un autre lieu physique, mais peut accéder aux contenus à tout moment de la journée, en équilibrant sa vie professionnelle ou personnelle avec sa vie académique.

L'équipe enseignante de ce Mastère Spécialisé en informatique et langues a sélectionné avec soin chacun des thèmes de cette mise à jour afin d'offrir à l'étudiant une opportunité d'étude la plus complète possible et toujours liée à l'actualité.

Ce **Mastère Spécialisé en Informatique et Langages** contient le programme le plus complet et le plus à jour du marché. Les principales caractéristiques sont les suivantes:

- Le développement d'études de cas présentées par des experts en informatique et en langages
- Le contenu graphique, schématique et éminemment pratique du programme fournit des informations scientifiques et pratiques sur les disciplines essentielles à la pratique professionnelle
- Exercices pratiques permettant de réaliser le processus d'auto-évaluation afin d'améliorer l'apprentissage
- Elle met l'accent sur les méthodologies innovantes en informatique et en langages.
- Des cours théoriques, des questions posées à l'expert, des forums de discussion sur des sujets controversés et un travail de réflexion individuel vous seront proposés
- La possibilité d'accéder aux contenus depuis n'importe quel appareil fixe ou portable doté d'une connexion internet

*Une occasion exceptionnelle d'apprendre de manière simple et confortable les processus et connaissances mathématiques et de base nécessaires pour réaliser une programmation informatique de qualité"*

*Un Mastère Spécialisé qui fonde son efficacité sur la technologie éducative la plus appréciée du marché, avec des systèmes audiovisuels et d'étude qui vous permettront d'apprendre plus rapidement et plus confortablement"*

Son contenu multimédia, développé avec les dernières technologies éducatives, permettra au professionnel un apprentissage concret et contextuel, c'est-à-dire un environnement simulé qui fournira une actualisation immersive programmée pour s'entraîner dans des situations réelles.

La conception de ce programme est basée sur l'Apprentissage par Problèmes. Ainsi l'étudiant devra essayer de résoudre les différentes situations de pratique professionnelle qui se présentent à lui tout au long du cursus. À cette fin, le professionnel sera assisté par un système vidéo interactif innovant, développé par des experts en informatique et en langage renommés et expérimentés.

*Nous mettons à votre service un matériel didactique large et clair, qui incorpore tous les sujets actuels d'intérêt pour le professionnel qui veut progresser en informatique et en langages.*

*Une étude à fort impact pédagogique qui vous permettra d'adapter l'effort à vos besoins, en alliant flexibilité et intensité.*

02

# Objectifs

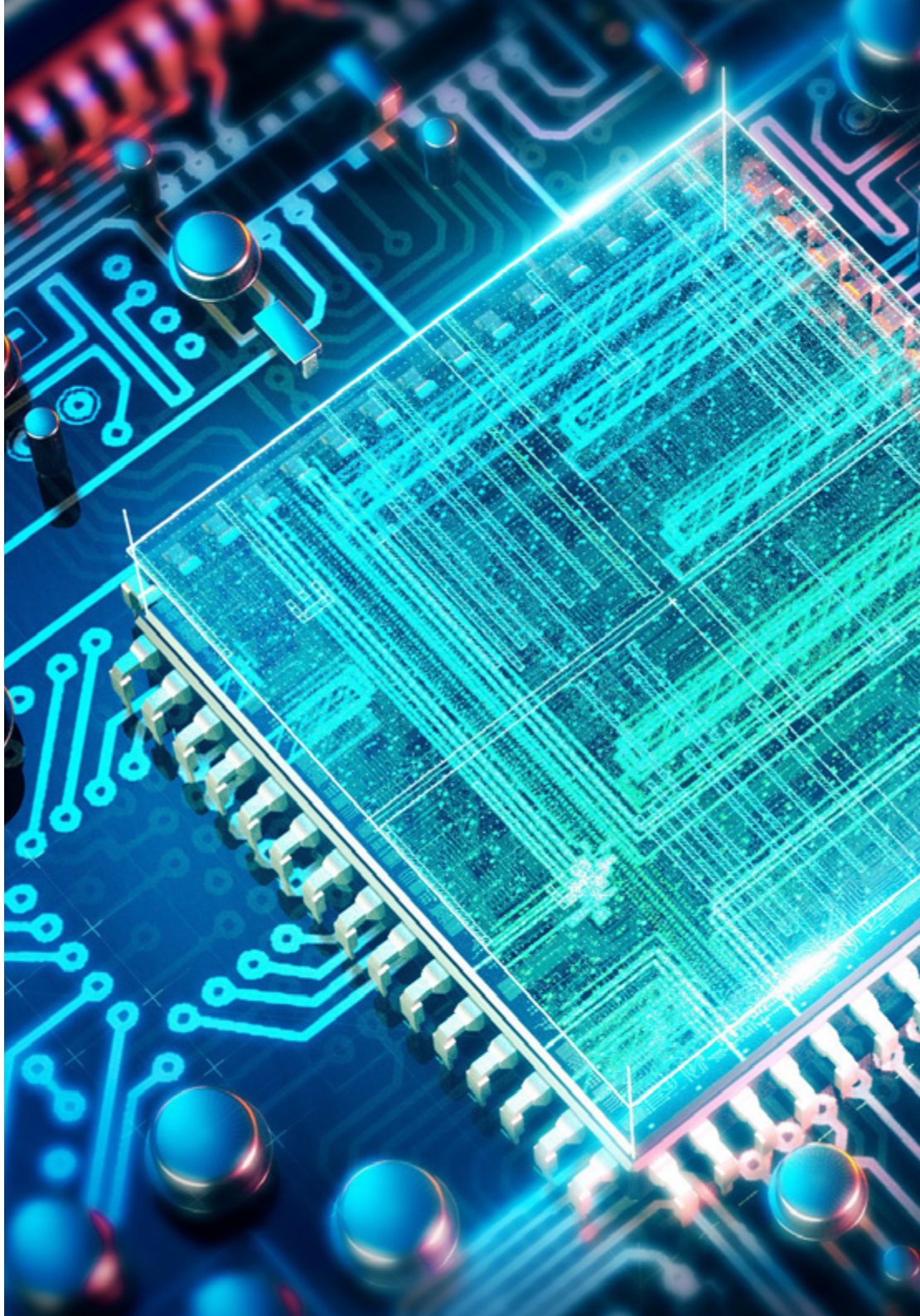
Le Mastère Spécialisé en Informatique et Langages a été créé spécifiquement pour le professionnel qui cherche à progresser dans ce domaine rapidement et avec une réelle qualité, en l'organisant sur la base d'objectifs réalistes et de grande valeur qui le propulseront à un autre niveau de travail dans ce domaine.

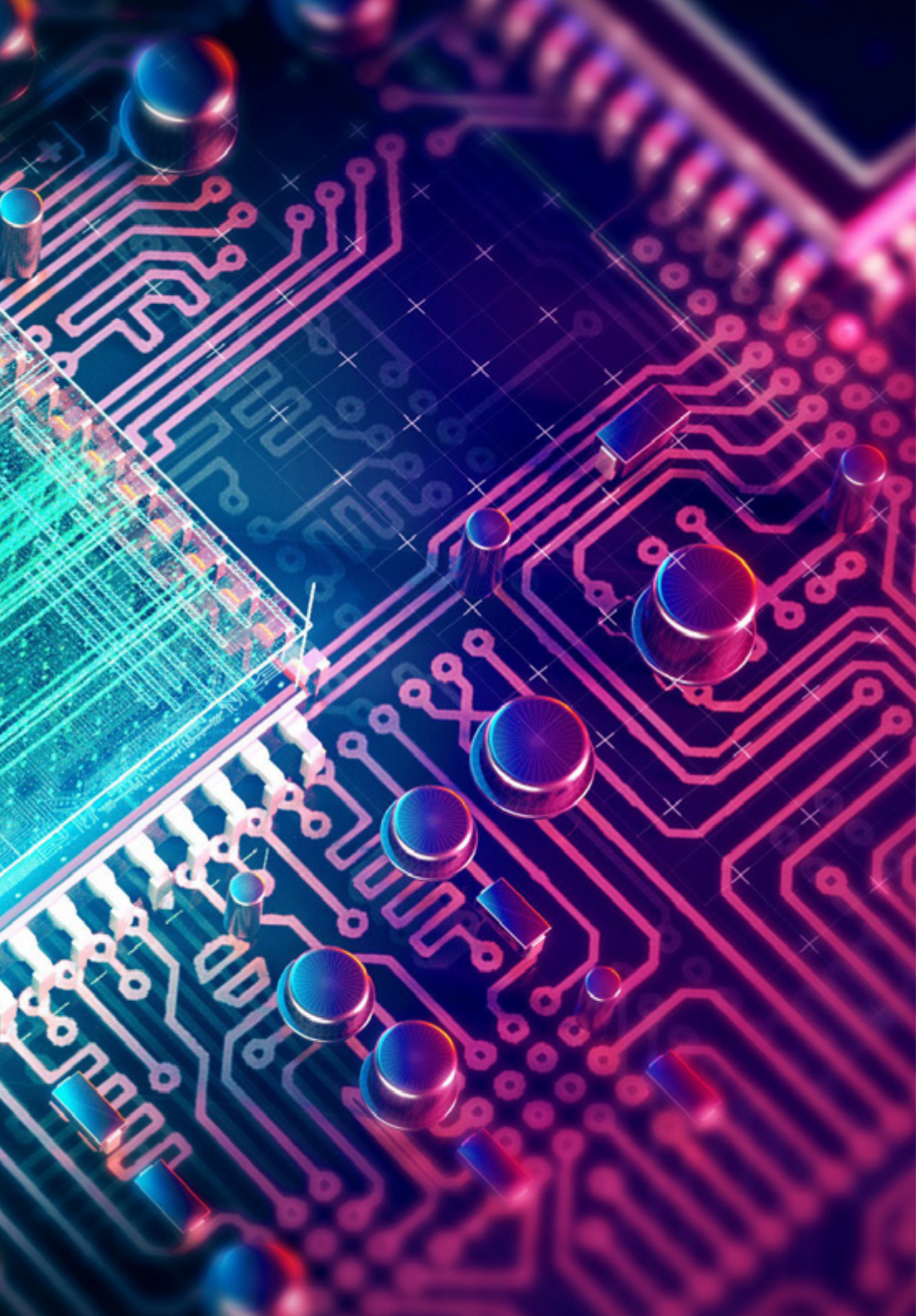
*Notre objectif est de fournir aux professionnels du domaine de l'informatique une mise à jour de haute qualité qui leur permettra d'intervenir avec compétence en Informatique et Langages"*

## Objectif général

- Former scientifiquement et technologiquement, ainsi que préparer à la pratique professionnelle de l'informatique et des langues, le tout avec une formation transversale et polyvalente adaptée aux nouvelles technologies et aux innovations dans ce domaine

*Saisissez l'occasion et faites le pas pour vous mettre à jour sur les derniers développements en informatique et en langages"*







## Objectifs spécifiques

### Module 1. Principes fondamentaux de la programmation

- Comprendre la structure de base d'un ordinateur, les logiciels et les langages de programmation à usage général
- Apprenez à concevoir et à interpréter des algorithmes, qui constituent la base nécessaire au développement de logiciels
- Comprendre les éléments essentiels d'un programme informatique, tels que les différents types de données, les opérateurs, les expressions, les instructions, les entrées/sorties et les instructions de contrôle
- Comprendre les différentes structures de données disponibles dans les langages de programmation polyvalents, tant statiques que dynamiques, et acquérir les compétences essentielles en matière de manipulation de fichiers
- Comprendre les différentes techniques de test des logiciels et l'importance de générer une bonne documentation en même temps qu'un bon code source
- Apprenez les bases du langage de programmation C++, l'un des langages de programmation les plus utilisés dans le monde

### Module 2. Structure des données

- Apprenez les bases de la programmation C++, notamment les classes, les variables, les expressions conditionnelles et les objets
- Comprendre les types de données abstraits, les types de structures de données linéaires, les structures de données hiérarchiques simples et complexes et leur mise en œuvre en C++
- Comprendre le fonctionnement des structures de données avancées autres que les structures habituelles
- Comprendre la théorie et la pratique liées à l'utilisation des monticules de priorité et des files d'attente de priorité

- Apprenez comment les tables de Hash, en tant que types de données et fonctions abstraites
- Comprendre la théorie des graphes, ainsi que les algorithmes et concepts avancés des graphes

### Module 3. Algorithme et complexité

- Apprenez les principales stratégies de conception d'algorithmes, ainsi que les différentes méthodes et mesures de calcul d'algorithmes
- Connaître les principaux algorithmes de tri utilisés dans le développement de logiciels
- Comprendre comment différents algorithmes fonctionnent avec des arbres, *Heaps* et des graphes
- Comprendre le fonctionnement des algorithmes *Greedy*, leur stratégie et des exemples de leur utilisation dans les principaux problèmes connus
- Connaître également l'utilisation des algorithmes *Greedy* sur les graphes
- Nous apprendrons les principales stratégies de recherche du chemin minimal, avec l'approche des problèmes essentiels du domaine et des algorithmes pour leur résolution
- Comprendre la technique du *Backtracking* et ses principales utilisations, ainsi que les techniques alternatives

### Module 4. Conception d'algorithmes avancés

- Approfondir la conception d'algorithmes avancés, en analysant les algorithmes récursifs et de type diviser pour régner, ainsi qu'en effectuant des analyses amorties
- Comprendre les concepts de la programmation dynamique et les algorithmes pour les problèmes NP
- Comprendre le fonctionnement de l'optimisation combinatoire, ainsi que les différents algorithmes de randomisation et les algorithmes parallèles
- Connaître et comprendre le fonctionnement des différentes méthodes de recherche locale et de recherche de candidats
- Apprendre les mécanismes de la vérification formelle des programmes et de la vérification itérative des programmes, y compris la logique du premier ordre et le système formel de Hoare
- Apprenez le fonctionnement de certaines des principales méthodes numériques telles que la méthode de bissection, la méthode de Newton Raphson et la méthode de la sécante

### Module 5. Programmation avancée

- Approfondir les connaissances en programmation, notamment en ce qui concerne la programmation orientée objet, et les différents types de relations entre les classes existantes
- Connaître les différents modèles de conception pour les problèmes orientés objet
- Découvrez la programmation événementielle et le développement d'interfaces utilisateur avec Qt
- Acquérir les connaissances essentielles de la programmation concurrente, des processus et des threads
- Apprenez à gérer l'utilisation des threads et de la synchronisation, ainsi que la résolution des problèmes courants dans le cadre de la programmation concurrente
- Comprendre l'importance de la documentation et des tests dans le développement de logiciels

### Module 6. Informatique théorique

- Comprendre les concepts mathématiques théoriques essentiels à l'informatique, tels que la logique propositionnelle, la théorie des ensembles et les ensembles numériques et non numériques
- Comprendre les concepts de langages formels et de grammaires, ainsi que les machines de Turing dans leurs différentes variantes
- Découvrez les différents types de problèmes indécidables et insolubles, y compris les différentes variantes de ceux-ci et leurs approches

- Comprendre le fonctionnement de différents types de langages basés sur la randomisation et d'autres types de classes et de grammaires
- En savoir plus sur d'autres systèmes informatiques avancés tels que l'informatique membranaire, l'informatique ADN et l'informatique quantique

### Module 7. Théorie des automates et langages formels

- Comprendre la théorie des automates et des langages formels, en apprenant les concepts d'alphabets, de chaînes de caractères et de langages, ainsi que la manière d'effectuer des démonstrations formelles
- Approfondir la compréhension des différents types d'automates finis, qu'ils soient déterministes ou non déterministes
- Apprenez les concepts de base et avancés liés aux langages réguliers et aux expressions régulières, ainsi que l'application du lemme de pompage et la fermeture des langages réguliers
- Comprendre les grammaires indépendantes du contexte, ainsi que le fonctionnement des automates à pile
- Pour approfondir les formes normales, le lemme de pompage des grammaires indépendantes du contexte et les propriétés des langues indépendantes du contexte

### Module 8. Processeurs de langue

- Introduire les concepts liés au processus de compilation et les différents types d'analyse: lexicale, syntaxique et sémantique
- Connaître le fonctionnement d'un analyseur lexical, sa mise en œuvre et la récupération des erreurs
- Approfondir la connaissance de l'analyse syntaxique, à la fois descendante et ascendante, mais en mettant l'accent sur les différents types d'analyseurs ascendants
- Comprendre le fonctionnement des analyseurs sémantiques, la tradition syntaxique, la table des symboles et les différents types
- Apprenez les différents mécanismes de génération de code, tant dans les environnements d'exécution que pour la génération de code intermédiaire
- Poser les bases de l'optimisation du code, notamment la réorganisation des expressions et l'optimisation des boucles

### Module 9. Infographie et visualisation

- Introduire les concepts essentiels de l'infographie et de la visualisation par ordinateur, tels que la théorie des couleurs et ses modèles et les propriétés de la lumière
- Comprendre le fonctionnement des primitives de sortie et leurs algorithmes, tant pour le dessin de lignes que pour le dessin de cercles et de remplissages
- Étude approfondie des différentes transformations 2D et 3D, de leurs systèmes de coordonnées et de la visualisation par ordinateur
- Apprenez à faire des projections et des coupes en 3D, ainsi que l'élimination des surfaces cachées
- Apprenez la théorie liée à l'interpolation et aux courbes paramétriques, ainsi que les courbes de Bézier et les B-splines

### Module 10. Informatique bio-inspirée

- Introduire le concept de calcul bio-inspiré, ainsi que comprendre le fonctionnement de différents types d'algorithmes d'adaptation sociale et d'algorithmes génétiques
- Approfondir l'étude des différents modèles d'informatique évolutive, en connaissant leurs stratégies, leur programmation, leurs algorithmes et les modèles basés sur l'estimation des distributions
- Comprendre les principales stratégies d'exploration-exploitation de l'espace pour les algorithmes génétiques
- Comprendre le fonctionnement de la programmation évolutionnaire appliquée aux problèmes d'apprentissage et aux problèmes multi-objectifs
- Apprenez les concepts essentiels liés aux réseaux neuronaux et comprenez comment ils fonctionnent dans des cas d'utilisation réels appliqués à des domaines aussi divers que la recherche médicale, l'économie et la vision par ordinateur

03

# Compétences

Après avoir passé les évaluations du Mastère Spécialisé en Informatique et Langages, le professionnel aura acquis les compétences nécessaires pour connaître les principes fondamentaux de l'informatique avec la capacité de travailler avec des langages de programmation et des données.

*Acquiert la capacité de réaliser de nouveaux développements informatiques à partir de la compréhension et de la maîtrise des différents langages et algorithmes et de leur application pratique"*

## Compétence générale

- Exécuter correctement les tâches liées aux ordinateurs et au langage informatique

*Améliorez vos compétences pour participer à divers projets technologiques"*

## Compétences spécifiques

- Concevoir des algorithmes pour développer des programmes informatiques et appliquer le langage de programmation
- Comprendre et utiliser la structure des données informatiques
- Utiliser les algorithmes nécessaires pour résoudre les problèmes informatiques.
- Connaissance approfondie de la conception d'algorithmes avancés et des méthodes de recherche
- Effectuer des tâches de programmation informatique
- Comprendre et appliquer la théorie qui sous-tend l'informatique, comme les mathématiques
- Connaître la théorie des automates et appliquer le langage informatique
- Connaître les fondements théoriques des langages de programmation et les techniques de traitement lexical, syntaxique et sémantique associées
- Comprendre les concepts de base des mathématiques et de la complexité informatique afin de les appliquer à la résolution de problèmes informatiques
- Connaître et appliquer les principes fondamentaux de l'informatique pour réaliser de nouveaux développements informatiques

04

# Structure et contenu

La structure des contenus a été créée de manière à ce que les connaissances soient assimilées progressivement, en réalisant un parcours de croissance qui vous mènera à l'excellence dans votre profession.

*Tous les domaines d'intérêt que vous devez maîtriser afin de travailler en toute sécurité et avec succès dans le domaine de l'informatique et des langages, compilés dans un syllabus de qualité supérieure"*

## **Module 1.** Principes fondamentaux de la programmation

1.1. Introduction à la programmation
- 1.1.1. Structure de base d'un ordinateur
- 1.1.2. Software
- 1.1.3. Langage de programmation
- 1.1.4. Cycle de vie d'une application logicielle

1.2. Conception d'algorithmes
- 1.2.1. La résolution de problèmes
- 1.2.2. Techniques descriptives
- 1.2.3. Éléments et structure d'un algorithme

1.3. Éléments d'un programme
- 1.3.1. Origine et caractéristiques du langage C++
- 1.3.2. L'environnement de développement
- 1.3.3. Concept de programme
- 1.3.4. Types de données fondamentales
- 1.3.5. Opérateurs
- 1.3.6. Expressions
- 1.3.7. Phrases
- 1.3.8. Entrée et sortie de données

1.4. Phrases de contrôle
- 1.4.1. Phrases
- 1.4.2. Bifurcations
- 1.4.3. Loops

1.5. Abstraction et modularité: fonctions
- 1.5.1. Conception modulaire
- 1.5.2. Concept de fonction et d'utilité
- 1.5.3. Définition d'une fonction
- 1.5.4. Flux d'exécution dans un appel de fonction
- 1.5.5. Prototype d'une fonction
- 1.5.6. Retour des résultats
- 1.5.7. Appel d'une fonction: paramètres
- 1.5.8. Passage de paramètres par référence et par valeur
- 1.5.9. Identification de la zone

1.6. Structures de données statiques
- 1.6.1. *Arrays*
- 1.6.2. Matrices. Polyèdres
- 1.6.3. Recherche et tri
- 1.6.4. Chaînes Fonctions entrées/sorties pour les chaînes
- 1.6.5. Structures Unions
- 1.6.6. Nouveaux types de données

1.7. Structures de données dynamiques: pointeurs
- 1.7.1. Concept. Définition de pointeur
- 1.7.2. Opérateurs et opérations avec des pointeurs
- 1.7.3. *Arrays* de pointeurs
- 1.7.4. Pointeurs et *Arrays*
- 1.7.5. Pointeurs vers des chaînes
- 1.7.6. Pointeurs vers des structures
- 1.7.7. Indirectivité multiple
- 1.7.8. Pointeurs vers les fonctions
- 1.7.9. Passage de fonctions, de structures et de tableaux comme paramètres de fonction

1.8. Fichiers
- 1.8.1. Concepts de base
- 1.8.2. Opérations sur fichiers
- 1.8.3. Types de fichiers
- 1.8.4. Organisation de fichiers
- 1.8.5. Introduction aux fichiers C++
- 1.8.6. Traitement des fichiers

1.9. Récursivité
- 1.9.1. Définition de la récursion
- 1.9.2. Types de récursion
- 1.9.3. Avantages et inconvénients
- 1.9.4. Considérations
- 1.9.5. Conversion récursive-itérative
- 1.9.6. La pile de récursion



1.10. Preuves et documentation
- 1.10.1. Test de programmes
- 1.10.2. Test de la boîte blanche
- 1.10.3. Test de la boîte noire
- 1.10.4. Outils de tests
- 1.10.5. Documentation du programme

## **Module 2.** Structure des données

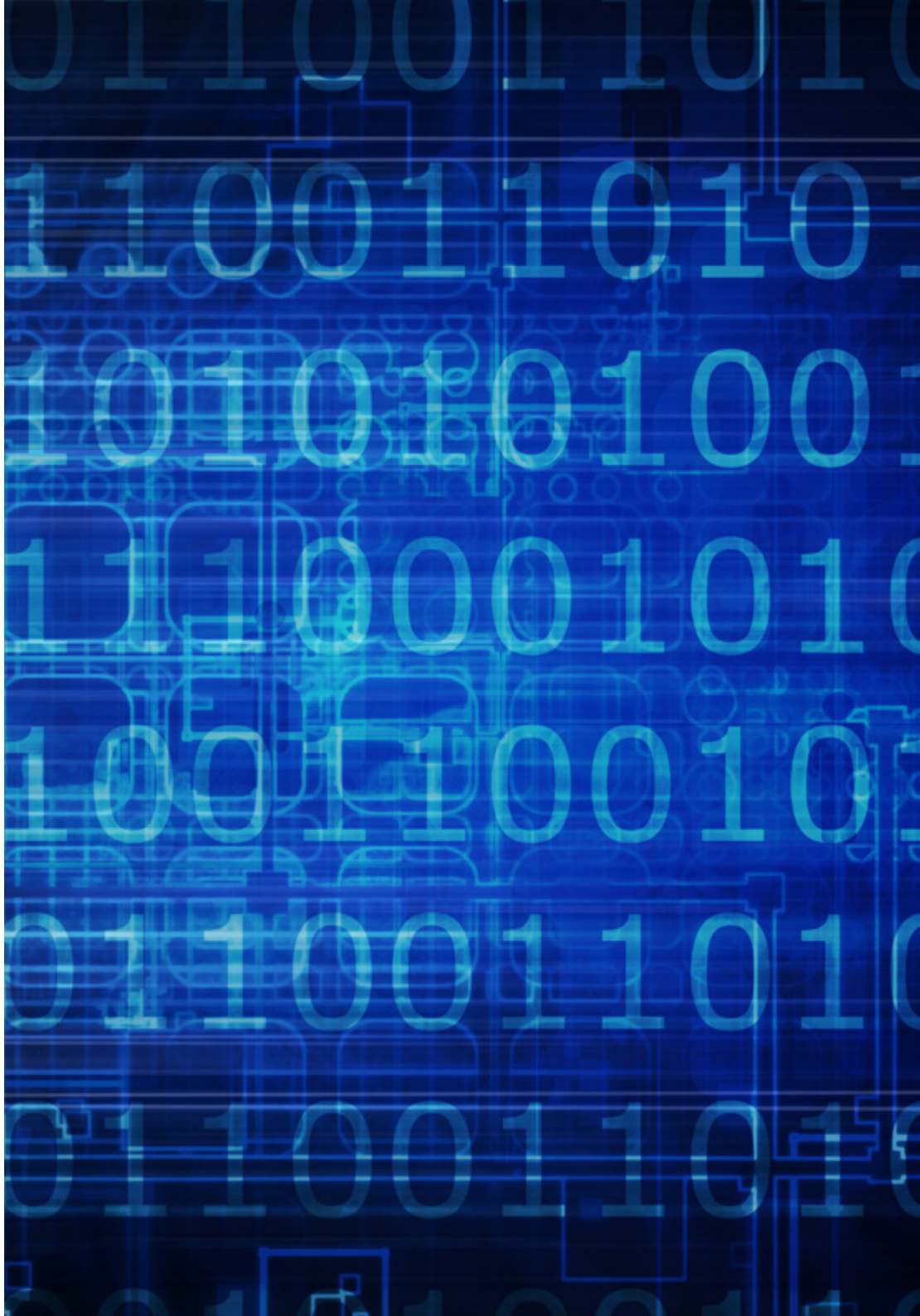
2.1. Introduction à la programmation C++
- 2.1.1. Classes, constructeurs, méthodes et attributs
- 2.1.2. Variables
- 2.1.3. Expressions conditionnelles et boucles
- 2.1.4. Objets

2.2. Types de données abstraites (ADT)
- 2.2.1. Types de données
- 2.2.2. Structures de base et TAD
- 2.2.3. Vecteurs et Arrays

2.3. Structures de données linéaires
- 2.3.1. TAD liste. Définition
- 2.3.2. Listes liées et listes doublement liées
- 2.3.3. Listes ordonnées
- 2.3.4. Listes en C++
- 2.3.5. TAD Pile
- 2.3.6. TAD Queue
- 2.3.7. Pile et Queue en C++

2.4. Structures de données hiérarchiques
- 2.4.1. TAD Arbre
- 2.4.2. Sentiers
- 2.4.3. Arbres N-aires
- 2.4.4. Arbres binaires
- 2.4.5. Arbres de recherche binaires

2.5. Structures de données hiérarchiques : arbres complexes
  2.5.1. Arbres parfaitement équilibrés ou arbres de hauteur minimale
  2.5.2. Arbres à trajets multiples
  2.5.3. Références bibliographiques
2.6. Monticules et queue de priorité
  2.6.1. Monticules TAD
  2.6.2. TAD File d'attente prioritaire
2.7. Tables Hash
  2.7.1. TAD Table *Hash*
  2.7.2. Fonctions *Hash*
  2.7.3. Fonction *Hash* en tables *Hash*
  2.7.4. Redispersion
  2.7.5. Tables *Hash* ouvertes
2.8. Networks
  2.8.1. TAD Network
  2.8.2. Type de Network
  2.8.3. Représentation graphique et opérations de base
  2.8.4. Design de Networks
2.9. Algorithmes et concepts graphiques avancés
  2.9.1. Problèmes sur les graphiques
  2.9.2. Algorithmes sur les routes
  2.9.3. Algorithmes sur les routes
  2.9.4. Autres algorithmes
2.10. Autres structures de données
  2.10.1. Sets
  2.10.2. Arrays parallèles
  2.10.3. Tableaux des symboles
  2.10.4. *Tries*



## **Module 3.** Algorithme et complexité

3.1. Introduction aux stratégies de conception d'algorithmes
- 3.1.1. Récursivité
- 3.1.2. Diviser pour mieux régner
- 3.1.3. Autres stratégies

3.2. Efficacité et analyse des algorithmes
- 3.2.1. Mesures d'efficacité
- 3.2.2. Mesurer la taille de l'entrée
- 3.2.3. Mesure du temps d'exécution
- 3.2.4. Pire cas, meilleur cas et cas moyen
- 3.2.5. Notation asymptotique
- 3.2.6. Critères d'analyse mathématique pour les algorithmes non récursifs
- 3.2.7. Analyse mathématique des algorithmes récursifs
- 3.2.8. Analyse empirique des algorithmes

3.3. Algorithmes de tri
- 3.3.1. Concept de tri
- 3.3.2. Tri de la bulle
- 3.3.3. Tri par sélection
- 3.3.4. Ordre d'insertion
- 3.3.5. Fusionner le tri (Merge Sort)
- 3.3.6. Tri rapide (QuickSort)

3.4. Algorithmes avec arbres
- 3.4.1. Concept d'arbre
- 3.4.2. Arbres binaires
- 3.4.3. Parcours d'arbres
- 3.4.4. Représentation des expressions
- 3.4.5. Arbres binaires ordonnés
- 3.4.6. Arbres binaires balancés

3.5. Algorithmes avec *Heaps*
- 3.5.1. Les *Heaps*
- 3.5.2. L'algorithme HeapSort
- 3.5.3. Files d'attente prioritaires

3.6. Algorithmes avec graphes
- 3.6.1. Représentation
- 3.6.2. Parcours en largeur
- 3.6.3. Visite en profondeur
- 3.6.4. Ordre topologique

3.7. Algorithmes *Greedy*
- 3.7.1. La stratégie *Greedy*
- 3.7.2. Éléments de La stratégie *Greedy*
- 3.7.3. Change de devises
- 3.7.4. Le problème du voyageur
- 3.7.5. Problème de sac à dos

3.8. Recherche de chemins minimaux
- 3.8.1. Le problème du chemin minimal
- 3.8.2. Arcs et cycles négatifs
- 3.8.3. Algorithme de Dijkstra

3.9. Algorithmes Greedy sobre Grafos
- 3.9.1. L'arbre à chevauchement minimal
- 3.9.2. L'algorithme de Prim
- 3.9.3. L'algorithme de L'algorithme de Kruskal
- 3.9.4. Analyse de complexité

3.10. *Backtracking*
- 3.10.1. Le *Backtracking*
- 3.10.2. Techniques alternatives

## **Module 4.** Conception d'algorithmes avancés

4.1. Analyse des algorithmes récursifs et des algorithmes de type "diviser pour régner"
- 4.1.1. Poser et résoudre des équations de récurrence homogènes et non-homogènes
- 4.1.2. Aperçu de la stratégie "diviser pour régner"

4.2. Analyse amortie
- 4.2.1. Analyse des agrégats
- 4.2.2. La méthode de comptabilisation
- 4.2.3. La méthode du potentiel

4.3. Programmation dynamique et algorithmes pour les problèmes NP
- 4.3.1. Caractéristiques de la programmation dynamique
- 4.3.2. Backtracking: retour en arrière
- 4.3.3. Branchements et élagage

4.4. Optimisation combinatoire
- 4.4.1. Représentation des problèmes
- 4.4.2. Optimisation 1D

4.5. Algorithmes de randomisation
- 4.5.1. Exemples d'algorithmes de randomisation
- 4.5.2. Le théorème de Buffon
- 4.5.3. Algorithme de Monte Carlo
- 4.5.4. Algorithme de Las Vegas

4.6. Recherche locale et recherche de candidats
- 4.6.1. *Garcient Ascent*
- 4.6.2. *Hill Climbing*
- 4.6.3. *Simulated Annealing*
- 4.6.4. *Tabu Search*
- 4.6.5. Recherche de candidats

4.7. Vérification formelle des programmes
- 4.7.1. Spécification d'abstractions fonctionnelles
- 4.7.2. Le langage de la logique du premier ordre
- 4.7.3. Le système formel de Hoare

4.8. Vérification des programmes itératifs
- 4.8.1. Les règles système formel de Hoare
- 4.8.2. Concept d'itérations invariantes

4.9. Méthodes numériques
- 4.9.1. La méthode de bissection
- 4.9.2. Méthode de Newton Raphson
- 4.9.3. La méthode sécante

4.10. Algorithmes parallèles
- 4.10.1. Opérations binaires parallèles
- 4.10.2. Opérations parallèles avec les réseaux
- 4.10.3. Le parallélisme dans le principe "diviser pour régner"
- 4.10.4. Le parallélisme dans la programmation dynamique

## **Module 5.** Programmation avancée

5.1. Introduction à la programmation orientée objet
- 5.1.1. Introduction à la programmation orientée objet
- 5.1.2. Conception de classe
- 5.1.3. Introduction à UML pour la modélisation des problèmes

5.2. Relations entre les classes
- 5.2.1. Abstraction et transmission
- 5.2.2. Concepts avancés de succession
- 5.2.3. Polymorphisme
- 5.2.4. Composition et agrégation

5.3. Introduction aux patrons de conception pour les problèmes orientés obje
- 5.3.1. Que sont les modèles de conception?
- 5.3.2. Modèle *Factory*
- 5.3.3. Modèle *Singleton*
- 5.3.4. Modèle *Observer*
- 5.3.5. Modèle *Composite*

5.4. Exceptions
- 5.4.1. Quelles sont les exceptions?
- 5.4.2. Capture et traitement des exceptions
- 5.4.3. Lancement des exceptions
- 5.4.4. Création d'exceptions

5.5. Interfaces utilisateur
- 5.5.1. Introduction à Qt
- 5.5.2. Positionnement
- 5.5.3. Que sont les événements?
- 5.5.4. Événements: définition et capture
- 5.5.5. Développement de l'interface utilisateur

5.6. Introduction à la programmation concurrente
- 5.6.1. Introduction à la programmation concurrente
- 5.6.2. Le concept de processus et de threads
- 5.6.3. Interaction entre processus ou threads
- 5.6.4. Threads en C++
- 5.6.5. Avantages et inconvénients de la programmation concurrente

5.7. Gestion et synchronisation des threads
- 5.7.1. Cycle de vie d'un thread
- 5.7.2. La classe *Thread*
- 5.7.3. Planification de threads
- 5.7.4. Groupes de threads
- 5.7.5. Threads de type démon
- 5.7.6. Synchronisation
- 5.7.7. Mécanismes de verrouillage
- 5.7.8. Mécanismes de Communication
- 5.7.9. Moniteurs

5.8. Problèmes courants de la programmation concurrente
- 5.8.1. Le problème des producteurs-consommateurs
- 5.8.2. Le problème des lecteurs et des écrivains
- 5.8.3. Le problème du dîner des philosophes

5.9. Documentation et test des logiciels
- 5.9.1. Pourquoi est-il important de documenter les logiciels?
- 5.9.2. Documentation de conception
- 5.9.3. Utilisation d'outils pour la documentation

5.10. Tests de logiciels
- 5.10.1. Introduction aux tests logiciels
- 5.10.2. Types de tests
- 5.10.3. Test unitaire
- 5.10.4. Test d'intégration
- 5.10.5. Test de validation
- 5.10.6. Test du système

## **Module 6.** Informatique théorique

6.1. Concepts mathématiques utilisés
- 6.1.1. Introduction à la logique propositionnelle
- 6.1.2. Théorie des relations
- 6.1.3. Ensembles numérotables et non numérotables

6.2. Langages et grammaires formels et introduction aux machines de Turing
- 6.2.1. Langages formels et grammaires
- 6.2.2. Problème de décision
- 6.2.3. La machine de Turing

6.3. Extensions pour les machines de Turing, machines de Turing sous contrainte et ordinateurs
- 6.3.1. Techniques de programmation pour les machines de Turing
- 6.3.2. Extensions pour les machines de Turing
- 6.3.3. Machines de Turing sous contrainte
- 6.3.4. Machines de Turing et ordinateurs

6.4. Indécidabilité
- 6.4.1. Langage non récursivement énumérable
- 6.4.2. Un problème indécidable récursivement énumérable

6.5. Autres problèmes innommables
- 6.5.1. Extensions pour les machines de Turing
- 6.5.2. Le problème de la correspondance postale (PCP)

6.6. Problèmes insolubles
- 6.6.1. Les classes P et NP
- 6.6.2. Un problème NP complet
- 6.6.3. Problème de satisfiabilité restreinte
- 6.6.4. Autres problèmes NP complet

6.7. Problèmes de Co-NP et PS
- 6.7.1. Complémentaire aux langages NP
- 6.7.2. Problèmes résolubles dans un espace polynomial
- 6.7.3. Compléter les problèmes PS

6.8. Classes de langages basés sur la randomisation
- 6.8.1. Modèle TM avec caractère aléatoire
- 6.8.2. Les classes RP et ZPP
- 6.8.3. Test de primauté
- 6.8.4. Complexité du test de primauté

6.9. Autres classes et grammaires
- 6.9.1. Automates finis probabilistes
- 6.9.2. Autómatas celulares
- 6.9.3. Cellules McCullogh et Pitts
- 6.9.4. Grammaires de Lindenmayer

6.10. Systèmes informatiques avancés
- 6.10.1. Informatique membranaire: systèmes P
- 6.10.2. Informatique ADN
- 6.10.3. L'informatique quantique

## **Module 7.** Théorie des automates et langages formels

7.1. Introduction à la théorie des automates
- 7.1.1. Pourquoi étudier la théorie des automates?
- 7.1.2. Introduction aux démonstrations formelles
- 7.1.3. Autres formes de démonstration
- 7.1.4. Induction mathématique
- 7.1.5. Alphabets, chaînes de caractères et langues

7.2. Automates finis déterministes
- 7.2.1. Introduction aux automates finis
- 7.2.2. Automates finis déterministes

7.3. Automates finis non déterministes
- 7.3.1. Automates finis non déterministes
- 7.3.2. Equivalence entre AFD et AFN
- 7.3.3. Automates finis avec transitions

7.4. Langues et expressions régulières (I)
- 7.4.1. Langages et expressions régulières
- 7.4.2. Automates finis et expressions régulières

7.5. Langues et expressions régulières (II)
- 7.5.1. Conversion des expressions régulières en automates
- 7.5.2. Applications des expressions régulières
- 7.5.3. Algèbre des expressions régulières

7.6. Pompage des lems et fermeture des langages réguliers
- 7.6.1. Slogan de pompage
- 7.6.2. Pompage des lems fermeture des langages réguliers

7.7. Équivalence et minimisation des automates
- 7.7.1. Équivalence HF
- 7.7.2. Minimisation de l'HF

7.8. Grammaires indépendantes du contexte (CIG)
- 7.8.1. Grammaires indépendantes du contexte
- 7.8.2. Arbres de dérivation
- 7.8.3. Applications des GIC
- 7.8.4. Ambiguïté dans les grammaires et les languages

7.9. Contrôleurs de pile et GICs
- 7.9.1. Définition des automates à piles
- 7.9.2. Langues supportées par un automate empilé
- 7.9.3. Équivalence entre les automates à pile et les GICs
- 7.9.4. Automate à pile déterministe

7.10. Formes normales, schéma de pompage des CPG et propriétés des CPL
- 7.10.1. Formes normales des CPG
- 7.10.2. Slogan de pompage
- 7.10.3. Pompage des lems fermeture des langages
- 7.10.4. Propriétés décisionnelles des PCD

## **Module 8.** Processeurs de langue

8.1. Introduction au processus de compilation
- 8.1.1. Compilation et interprétation
- 8.1.2. Environnement d'exécution du compilateur
- 8.1.3. Processus d'analyse
- 8.1.4. Processus de synthèse

8.2. Analyseur lexical
- 8.2.1. Qu'est-ce qu'un analyseur lexical?
- 8.2.2. Implémentation de l'analyseur lexical
- 8.2.3. Actions sémantiques
- 8.2.4. Récupération des erreurs
- 8.2.5. Questions de mise en œuvre

8.3. Analyse syntaxique
- 8.3.1. Qu'est-ce qu'un analyseur syntaxiques?
- 8.3.2. Concepts préliminaires
- 8.3.3. Analyseurs descendants
- 8.3.4. Analyseurs ascendants

8.4. Analyse syntaxique descendante et ascendante
- 8.4.1. Analyseur LL(1)
- 8.4.2. Analyseur LR(0)
- 8.4.3. Exemple d'un analyseur

8.5. Analyse ascendante avancée
- 8.5.1. Analyseur SLR
- 8.5.2. Analyseur LR(1)
- 8.5.3. Analyseur LR(k)
- 8.5.4. Analyseur LALR

8.6. Analyse sémantique (I)
- 8.6.1. Traduction axée sur la syntaxe
- 8.6.2. Tableaux des symboles

8.7. Analyse sémantique (II)
- 8.7.1. Vérification du type
- 8.7.2. Le sous-système des types
- 8.7.3. Équivalence de taux et conversions

8.8. Génération de code et environnement d'exécution
- 8.8.1. Aspects de la conception
- 8.8.2. Environnement d'exécution
- 8.8.3. Organisation de la mémoire
- 8.8.4. Allocation de mémoire

8.9. Génération de code intermédiaire
- 8.9.1. Traduction axée sur la synthèse
- 8.9.2. Représentations intermédiaires
- 8.9.3. Exemples de traductions

8.10. Optimisation du code
- 8.10.1. Affectation des registres
- 8.10.2. Élimination des allocations mortes
- 8.10.3. Exécution au moment de la compilation
- 8.10.4. Réorganisation des expressions
- 8.10.5. Optimisation des boucles

## **Module 9.** Infographie et visualisation

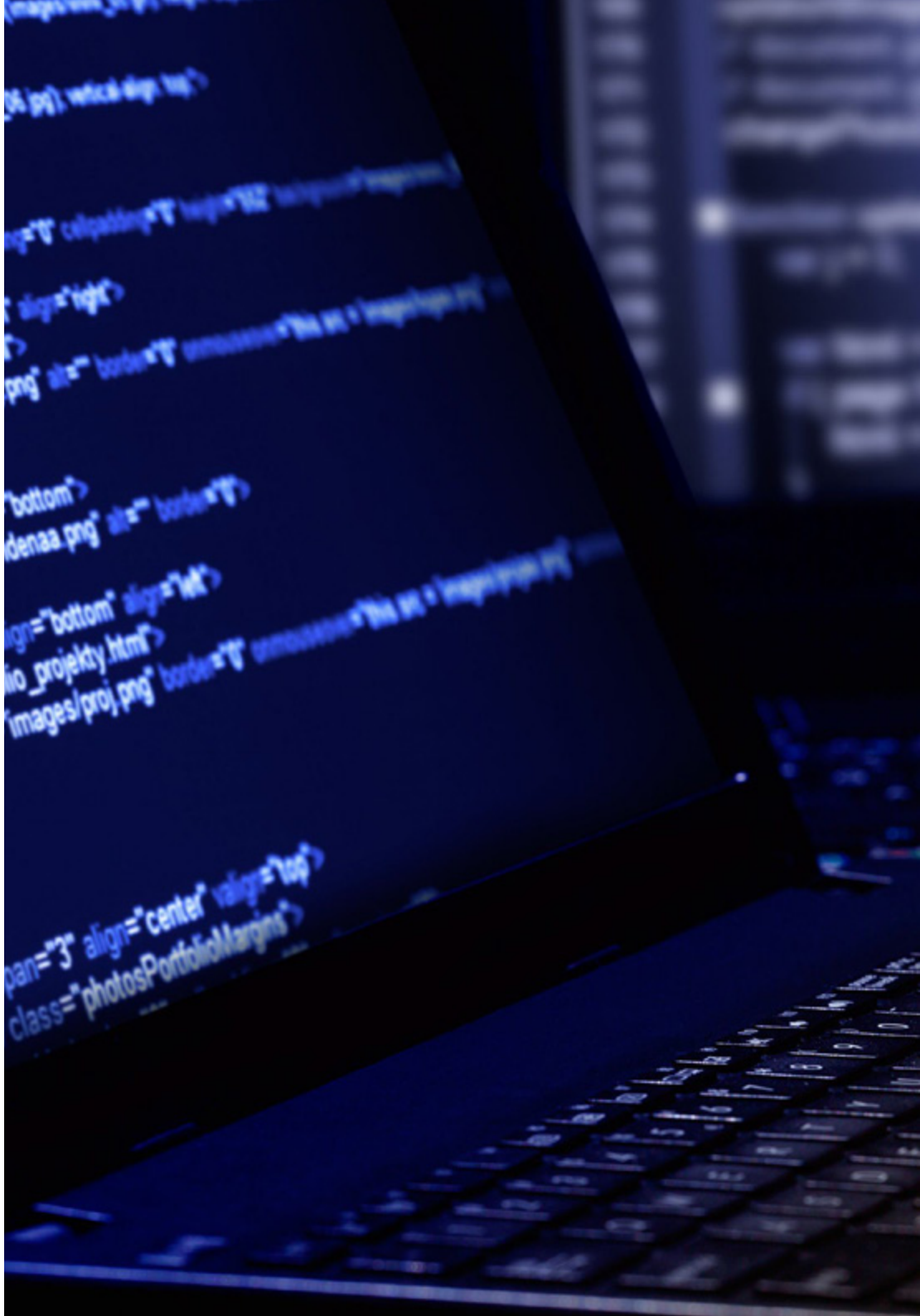
9.1. Théorie des couleurs
- 9.1.1. Propriétés de la lumière
- 9.1.2. Modèles de couleurs
- 9.1.3. La norme CIE
- 9.1.4. *Profiling*

9.2. Primitives de sortie
- 9.2.1. Le contrôleur vidéo
- 9.2.2. Algorithmes de dessin au trait
- 9.2.3. Algorithmes de dessin au trait
- 9.2.4. Algorithmes de remplissage

9.3. Transformations 2D et systèmes de coordonnées 2D et découpage 2D
- 9.3.1. Transformations géométriques de base
- 9.3.2. Coordonnées homogènes
- 9.3.3. Transformation inverse
- 9.3.4. Composition des transformations
- 9.3.5. Autres transformations
- 9.3.6. Coordonner le changement
- 9.3.7. Systèmes de coordonnées 2D
- 9.3.8. Changement de coordonnées
- 9.3.9. Normalisation
- 9.3.10. Algorithmes de découpage

9.4. Transformations 3D
- 9.4.1. Traduction
- 9.4.2. Rotation
- 9.4.3. Scale
- 9.4.4. Réflexion
- 9.4.5. Cisaillement

9.5. Visualisation et modification des coordonnées 3D
- 9.5.1. Systèmes de coordonnées 3D
- 9.5.2. Visualisation
- 9.5.3. Changement de coordonnées
- 9.5.4. Projection et normalisation



9.6. Projection et découpage 3D
- 9.6.1. Projection orthogonale
- 9.6.2. Projection parallèle oblique
- 9.6.3. Projection en perspective
- 9.6.4. Algorithmes de découpage 3D

9.7. Suppression des surfaces cachées
- 9.7.1. *Back face removal*
- 9.7.2. Z-buffer
- 9.7.3. Algorithme du peintre
- 9.7.4. Algorithme de Warnock
- 9.7.5. Détection des lignes cachées

9.8. Interpolation et courbes paramétriques
- 9.8.1. Interpolation et approximation polynomiale
- 9.8.2. Représentation paramétrique
- 9.8.3. polynôme de Lagrange
- 9.8.4. *Splines* cubes naturels
- 9.8.5. Fonctions de base
- 9.8.6. Représentation matricielle

9.9. Courbes de Bézier
- 9.9.1. Construction algébrique
- 9.9.2. Forme matricielle
- 9.9.3. Composition
- 9.9.4. Construction géométrique
- 9.9.5. Algorithme de dessin

9.10. *B-Splines*
- 9.10.1. Le problème du contrôle local
- 9.10.2. B-splines cubiques uniformes
- 9.10.3. Fonctions de base et points de contrôle
- 9.10.4. Dérive vers l'origine et la multiplicité
- 9.10.5. Représentation matricielle
- 9.10.6. *B-splines* non uniformes

## **Module 10.** Informatique bio-inspirée

10.1. Introduction à l'informatique bio-inspirée
- 10.1.1. Introduction à l'informatique bio-inspirée

10.2. Algorithmes d'adaptation sociale
- 10.2.1. Calcul bio-inspiré basé sur les colonies de fourmis
- 10.2.2. Variantes des algorithmes de colonies de fourmis
- 10.2.3. Informatique en nuage de particules

10.3. Algorithmes génétiques
- 10.3.1. Structure générale
- 10.3.2. Mise en œuvre des principaux opérateurs

10.4. Stratégies d'exploration-exploitation de l'espace pour les algorithmes génétiques
- 10.4.1. Algorithme CHC
- 10.4.2. Problèmes multimodaux

10.5. Modèles de calcul évolutif (I)
- 10.5.1. Stratégies évolutives
- 10.5.2. Programmation évolutive
- 10.5.3. Algorithmes basés sur l'évolution différentielle

10.6. Modèles de calcul évolutif (II)
- 10.6.1. Modèles d'évolution basés sur l'estimation des distributions (EDA)
- 10.6.2. Programmation génétique

10.7. La programmation du développement appliquée aux troubles de l'apprentissage
- 10.7.1. Apprentissage basé sur des règles
- 10.7.2. Méthodes évolutionnaires dans les problèmes de sélection d'instances

10.8. Problèmes multi-objectifs
- 10.8.1. Concept de dominance
- 10.8.2. Application des algorithmes évolutionnaires aux problèmes multi-objectifs

10.9. Réseaux neuronaux (I)
- 10.9.1. Introduction aux réseaux neuronaux
- 10.9.2. Exemple pratique avec les réseaux neuronaux

10.10. Réseaux neuronaux (II)
- 10.10.1. Cas d'utilisation des réseaux neuronaux dans la recherche médicale
- 10.10.2. Cas d'utilisation des réseaux neuronaux en économie
- 10.10.3. Cas d'utilisation des réseaux neuronaux dans la vision artificielle

05

# Méthodologie

Ce programme de formation offre une manière différente d'apprendre. Notre méthodologie est développée à travers un mode d'apprentissage cyclique: ***le Relearning***.

Ce système d'enseignement est utilisé, par exemple, dans les écoles de médecine les plus prestigieuses du monde et a été considéré comme l'un des plus efficaces par des publications de premier plan telles que le ***New England Journal of Medicine.***

*Découvrez Relearning, un système qui renonce à l'apprentissage linéaire conventionnel pour vous emmener à travers des systèmes d'enseignement cycliques: une façon d'apprendre qui s'est avérée extrêmement efficace, en particulier dans les matières qui exigent la mémorisation"*

## Étude de Cas pour mettre en contexte tout le contenu

Notre programme offre une méthode révolutionnaire de développement des compétences et des connaissances. Notre objectif est de renforcer les compétences dans un contexte changeant, compétitif et hautement exigeant.

*Avec TECH, vous pouvez expérimenter une manière d'apprendre qui ébranle les fondations des universités traditionnelles du monde entier"*


*Vous bénéficierez d'un système d'apprentissage basé sur la répétition, avec un enseignement naturel et progressif sur l'ensemble du cursus.*

*L'étudiant apprendra, par des activités collaboratives et des cas réels, à résoudre des situations complexes dans des environnements commerciaux réels.*

## Une méthode d'apprentissage innovante et différente

Cette formation TECH est un programme d'enseignement intensif, créé de toutes pièces, qui propose les défis et les décisions les plus exigeants dans ce domaine, tant au niveau national qu'international. Grâce à cette méthodologie, l'épanouissement personnel et professionnel est stimulé, faisant ainsi un pas décisif vers la réussite. La méthode des cas, technique qui constitue la base de ce contenu, permet de suivre la réalité économique, sociale et professionnelle la plus actuelle.

> *Notre programme vous prépare à relever de nouveaux défis dans des environnements incertains et à réussir votre carrière"*

La méthode des cas est le système d'apprentissage le plus largement utilisé dans les meilleures écoles d'informatique du monde depuis qu'elles existent. Développée en 1912 pour que les étudiants en Droit n'apprennent pas seulement le droit sur la base d'un contenu théorique, la méthode des cas consiste à leur présenter des situations réelles complexes afin qu'ils prennent des décisions éclairées et des jugements de valeur sur la manière de les résoudre. En 1924, elle a été établie comme méthode d'enseignement standard à Harvard.

Dans une situation donnée, que doit faire un professionnel? C'est la question à laquelle nous sommes confrontés dans la méthode des cas, une méthode d'apprentissage orientée vers l'action. Tout au long du programme, les étudiants seront confrontés à de multiples cas réels. Ils devront intégrer toutes leurs connaissances, faire des recherches, argumenter et défendre leurs idées et leurs décisions.

## Relearning Methodology

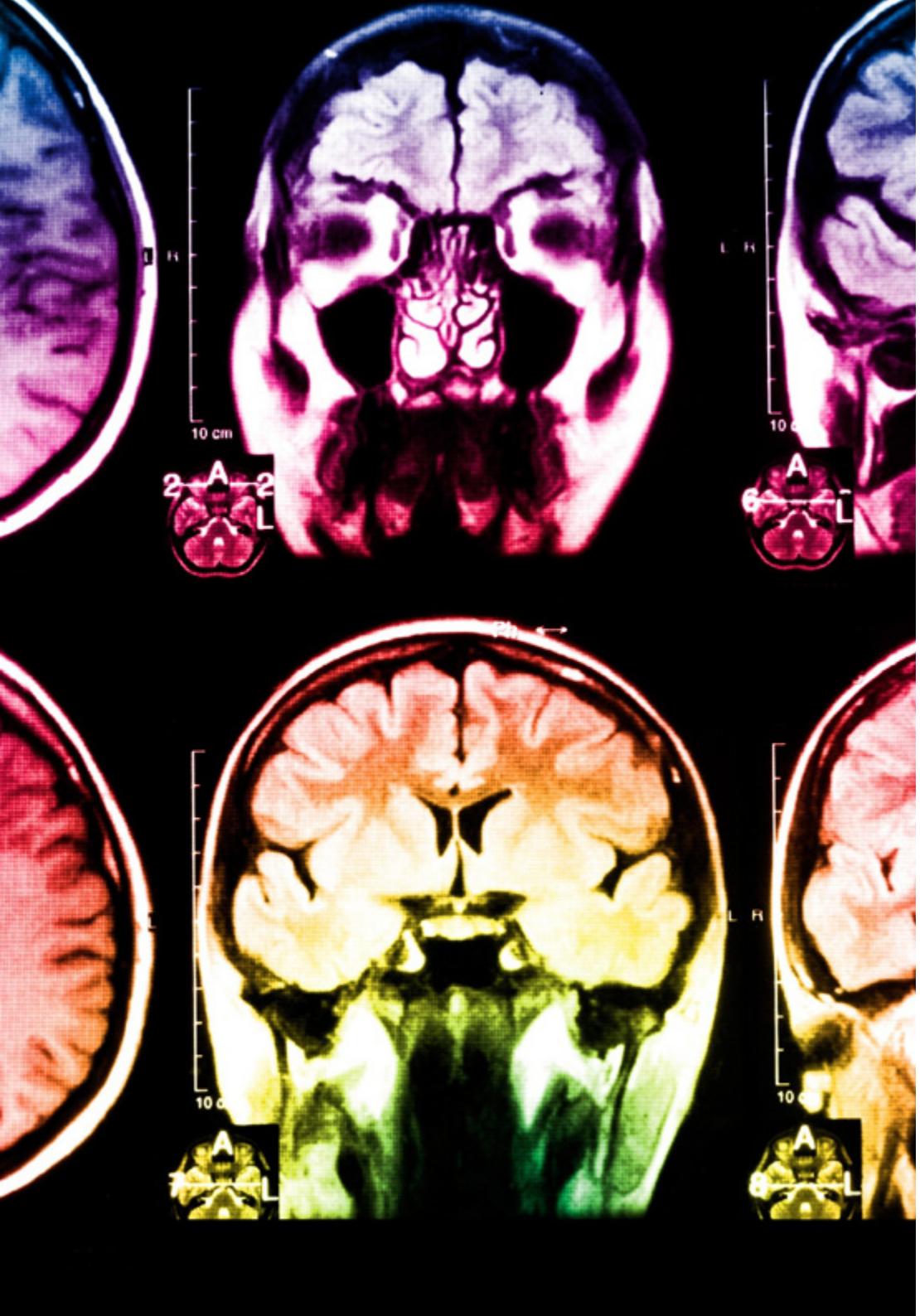
TECH combine efficacement la méthodologie des Études de Cas avec un système d'apprentissage 100% en ligne basé sur la répétition, qui associe différents éléments didactiques dans chaque leçon.

Nous enrichissons l'Étude de Cas avec la meilleure méthode d'enseignement 100% en ligne: le Relearning.

*En 2019, nous avons obtenu les meilleurs résultats d'apprentissage de toutes les universités en ligne du monde.*

À TECH, vous apprendrez avec une méthodologie de pointe conçue pour former les managers du futur. Cette méthode, à la pointe de la pédagogie mondiale, est appelée Relearning.

Notre université est la seule université autorisée à utiliser cette méthode qui a fait ses preuves. En 2019, nous avons réussi à améliorer les niveaux de satisfaction globale de nos étudiants (qualité de l'enseignement, qualité des supports, structure des cours, objectifs...) par rapport aux indicateurs de la meilleure université en ligne.



Dans notre programme, l'apprentissage n'est pas un processus linéaire, mais se déroule en spirale (apprendre, désapprendre, oublier et réapprendre). Par conséquent, chacun de ces éléments est combiné de manière concentrique. Cette méthodologie a permis de former plus de 650.000 diplômés universitaires avec un succès sans précédent dans des domaines aussi divers que la biochimie, la génétique, la chirurgie, le droit international, les compétences en gestion, les sciences du sport, la philosophie, le droit, l'ingénierie, le journalisme, l'histoire, les marchés financiers et les instruments. Tout cela dans un environnement très exigeant, avec un corps étudiant universitaire au profil socio-économique élevé et dont l'âge moyen est de 43,5 ans.

*Le Relearning vous permettra d'apprendre avec moins d'efforts et plus de performance, en vous impliquant davantage dans votre formation, en développant un esprit critique, en défendant des arguments et en contrastant les opinions: une équation directe vers le succès.*

À partir des dernières preuves scientifiques dans le domaine des neurosciences, non seulement nous savons comment organiser les informations, les idées, les images et les souvenirs, mais nous savons aussi que le lieu et le contexte dans lesquels nous avons appris quelque chose sont fondamentaux pour notre capacité à nous en souvenir et à le stocker dans l'hippocampe, pour le conserver dans notre mémoire à long terme.

De cette manière, et dans ce que l'on appelle Neurocognitive context-dependent e-learning, les différents éléments de notre programme sont reliés au contexte dans lequel le participant développe sa pratique professionnelle.

**Ce programme offre le support matériel pédagogique, soigneusement préparé pour les professionnels:**

### Support d'étude

Tous les contenus didactiques sont créés par les spécialistes qui enseigneront le cours, spécifiquement pour le cours, afin que le développement didactique soit vraiment spécifique et concret.

Ces contenus sont ensuite appliqués au format audiovisuel, pour créer la méthode de travail TECH en ligne. Tout cela, avec les dernières techniques qui offrent des pièces de haute qualité dans chacun des matériaux qui sont mis à la disposition de l'étudiant.

### Cours magistraux

Il existe des preuves scientifiques de l'utilité de l'observation par un tiers expert.

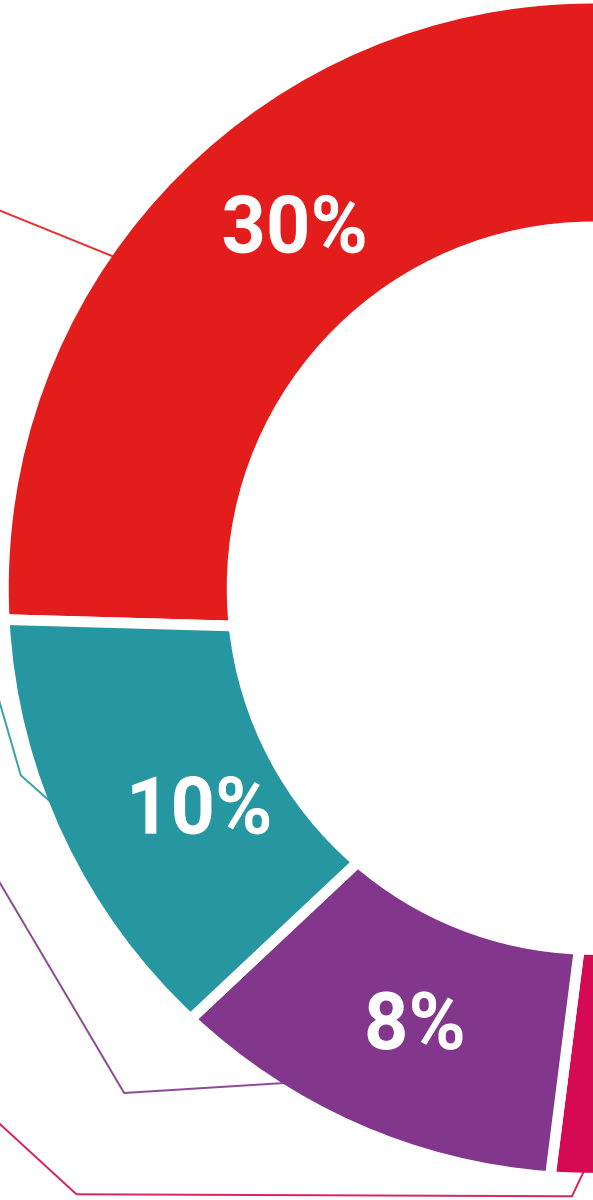
La méthode "Learning from an Expert" renforce les connaissances et la mémoire, et donne confiance dans les futures décisions difficiles.

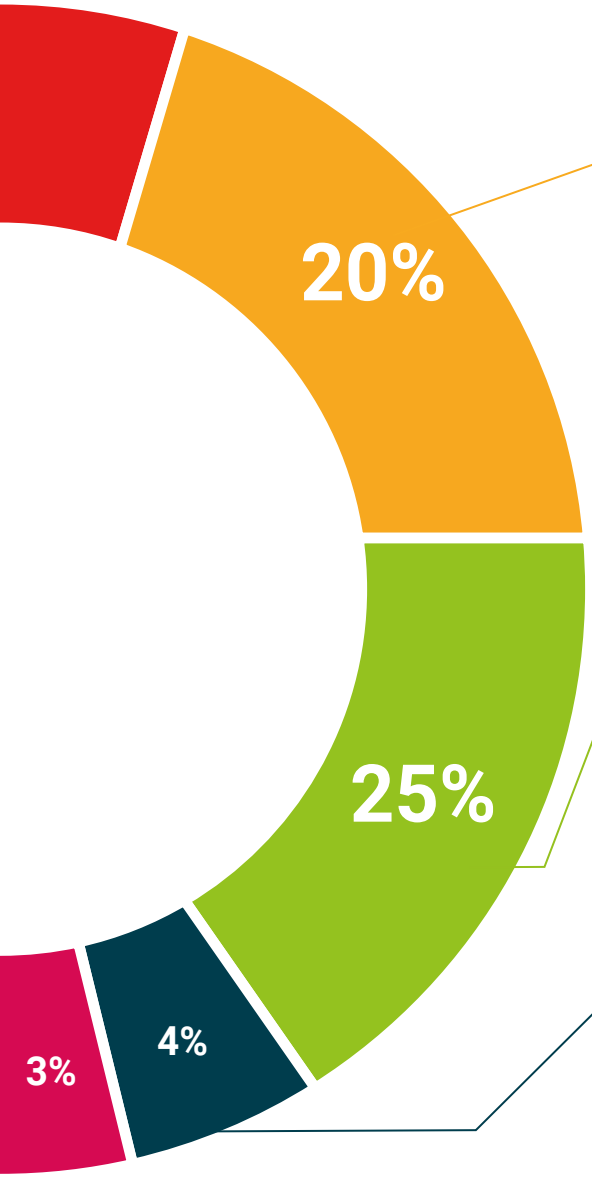
### Pratiques en compétences et aptitudes

Les étudiants réaliseront des activités visant à développer des compétences et des aptitudes spécifiques dans chaque domaine. Des activités pratiques et dynamiques pour acquérir et développer les compétences et aptitudes qu'un spécialiste doit développer dans le cadre de la mondialisation dans laquelle nous vivons.

### Lectures complémentaires

Articles récents, documents de consensus et directives internationales, entre autres. Dans la bibliothèque virtuelle de TECH, l'étudiant aura accès à tout ce dont il a besoin pour compléter sa formation.






### Case studies

Ils réaliseront une sélection des meilleures études de cas choisies spécifiquement pour ce diplôme. Des cas présentés, analysés et tutorés par les meilleurs spécialistes de la scène internationale.

### Résumés interactifs

L'équipe TECH présente les contenus de manière attrayante et dynamique dans des pilules multimédia comprenant des audios, des vidéos, des images, des diagrammes et des cartes conceptuelles afin de renforcer les connaissances.

Ce système éducatif unique pour la présentation de contenu multimédia a été récompensé par Microsoft en tant que "European Success Story".

### Testing & Retesting

Les connaissances de l'étudiant sont périodiquement évaluées et réévaluées tout au long du programme, par le biais d'activités et d'exercices d'évaluation et d'auto-évaluation, afin que l'étudiant puisse vérifier comment il atteint ses objectifs.

06

# Diplôme

Le Mastère Spécialisé en Informatique et Langages vous garantit, en plus de la formation la plus rigoureuse et la plus actuelle, l'accès à un diplôme universitaire de Mastère Spécialisé délivré par TECH Université Technologique.

*Réussissez ce programme et recevez votre Certificat Avancé sans déplacements ni formalités administratives"*

Ce **Mastère Spécialisé en Informatique et Langages** contient le programme le plus complet et le plus actuel du marché.

Après avoir réussi l'évaluation, l'étudiant recevra par courrier postal* avec accusé de réception son correspondant diplôme de **Mastère Spécialisé** délivré par **TECH Université Technologique**.

Le diplôme délivré par **TECH Université Technologique** indiquera la note obtenue lors du Mastère Spécialisé, et répond aux exigences communément demandées par les bourses d'emploi, les concours et les commissions d'évaluation des carrières professionnelles.

Diplôme: **Mastère Spécialisé en Informatique et Langages**

N.º d'heures officielles: **1.500 h.**


tech université technologique

Délivre le présent

DIPLÔME

à

Mme/M. _____, avec nº d'identification _____

Pour avoir finalisé et accrédité avec succès le programme de

MASTÈRE SPÉCIALISÉ

en

Informatique et Langages

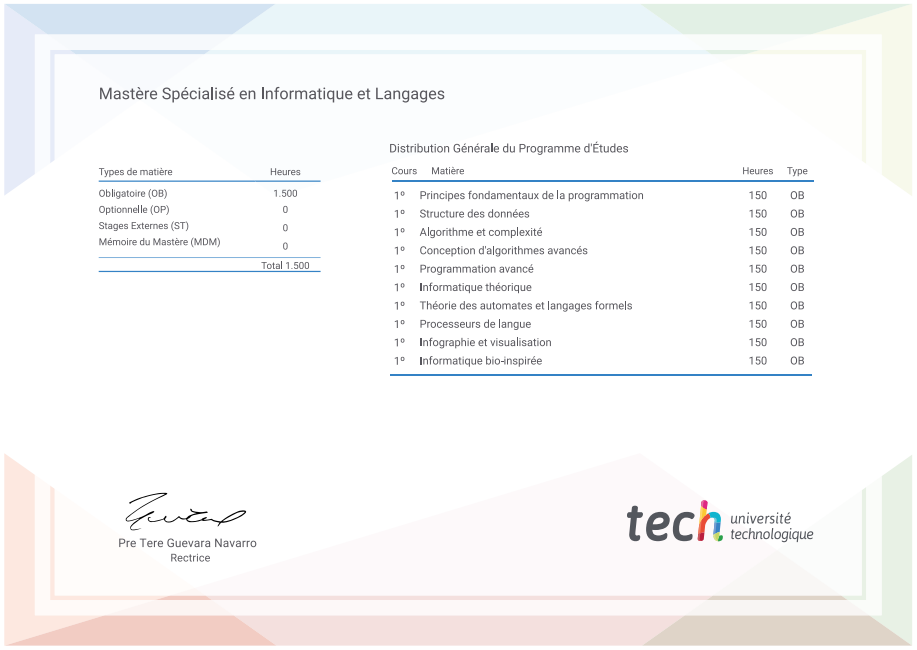
Il s'agit d'un diplôme spécialisé octroyé par cette Université d'une durée de 1.500 heures, débutant le dd/mm/aaaa et finalisant le dd/mm/aaaa.

TECH est une Institution Privée d'Enseignement Supérieur reconnue par le Ministère de l'Enseignement Public depuis le 28 juin 2018.

Fait le 17 juin 2020

Pre Tere Guevara Navarro
Rectrice

Code Unique TECH: AFWOR23S techtitute.com/diplômes


Mastère Spécialisé en Informatique et Langages

| Types de matière | Heures |
|---|---|
| Obligatoire (OB) | 1.500 |
| Optionnelle (OP) | 0 |
| Stages Externes (ST) | 0 |
| Mémoire du Mastère (MDM) | 0 |
| | Total 1.500 |

Distribution Générale du Programme d'Études

| Cours | Matière | Heures | Type |
|---|---|---|---|
| 1º | Principes fondamentaux de la programmation | 150 | OB |
| 1º | Structure des données | 150 | OB |
| 1º | Algorithme et complexité | 150 | OB |
| 1º | Conception d'algorithmes avancés | 150 | OB |
| 1º | Programmation avancé | 150 | OB |
| 1º | Informatique théorique | 150 | OB |
| 1º | Théorie des automates et langages formels | 150 | OB |
| 1º | Processeurs de langue | 150 | OB |
| 1º | Infographie et visualisation | 150 | OB |
| 1º | Informatique bio-inspirée | 150 | OB |

Pre Tere Guevara Navarro
Rectrice

tech université technologique

*Si l'étudiant souhaite que son diplôme version papier possède l'Apostille de La Haye, TECH EDUCATION fera les démarches nécessaires pour son obtention moyennant un coût supplémentaire.

tech université technologique

# Mastère Spécialisé
## Informatique et Langages

- Modalité: **en ligne**
- Durée: **12 mois**
- Qualification: **TECH Université Technologique**
- Intensité: **16h/semaine**
- Horaire: **à votre rythme**
- Examens: **en ligne**

# Mastère Spécialisé
## Informatique et Langages

tech université technologique