Mestrado Próprio

Computação e Linguagens

tech universidade tecnológica

tech universidade tecnológica

Mestrado Próprio

Computação e Linguagens

- Modalidade: **online**
- Duração: **12 meses**
- Certificação: **TECH Universidade Tecnológica**
- Créditos: **60 ECTS**
- Tempo dedicado: **16 horas/semana**
- Horário: **Ao seu próprio ritmo**
- Exames: **online**

Acesso ao site: www.techtitute.com/pt/informatica/mestrado-proprio/mestrado-proprio-computacao-linguagens

Índice

01

Apresentação

Os profissionais de informática precisam de manter as suas competências totalmente atualizadas para poderem continuar a atuar na sua área da melhor forma sem perderem nenhum dos desenvolvimentos que estão a ser introduzidos nesta área a um ritmo vertiginoso. Esta capacitação visa dotar os alunos de um conhecimento completo e aprofundado dos conhecimentos essenciais e das novidades mais interessantes na conceção de algoritmos no desenvolvimento de projetos informáticos utilizando os métodos mais inovadores e eficazes do setor.

Adquiria os conhecimentos fundamentais sobre Computação e a forma de os aplicar com sucesso no desenvolvimento de projetos informáticos num Mestrado Próprio de alta competência"

Desta forma, este Mestrado Próprio centra-se nos fundamentos da programação e estrutura de dados, algoritmos e complexidade, bem como na conceção avançada de algoritmos, programação avançada, processadores de linguagem e computação gráfica, entre outros aspetos relacionados com este contexto da informática.

Este Mestrado Próprio fornece ao aluno ferramentas e competências específicas para desenvolver com êxito a sua atividade profissional no vasto ambiente da Computação e Linguagens. Trabalhar em competências fundamentais como o conhecimento da realidade e da prática diária em diferentes áreas Informáticas e desenvolver responsabilidades no controlo e supervisão do seu trabalho, bem como competências específicas dentro deste campo.

Para além disso, tratando-se de um Mestrado Próprio 100% online, o aluno não estará condicionado por horários fixos nem pela necessidade de se deslocar para outro local físico, podendo aceder aos conteúdos em qualquer altura do dia, equilibrando o seu trabalho ou vida pessoal com a sua vida académica.

A equipa docente deste Mestrado Próprio em Computação e Linguagens fez uma seleção cuidadosa de cada uma das disciplinas desta capacitação de forma a oferecer ao aluno a oportunidade de estudo mais completa possível e sempre atual.

Este **Mestrado Próprio em Computação e Linguagens** conta com o conteúdo educacional mais completo e atualizado do mercado. As suas principais características são:

- O desenvolvimento de casos práticos apresentados por especialistas em Computação e Linguagens
- O conteúdo gráfico, esquemático e eminentemente prático fornece informações científicas e práticas sobre as disciplinas que são essenciais para a prática profissional
- Os exercícios práticos em que o processo de autoavaliação pode ser utilizado para melhorar a aprendizagem
- O seu foco especial nas metodologias inovadoras em Computação e Linguagens
- As lições teóricas, perguntas a especialistas, fóruns de discussão sobre questões controversas e atividades de reflexão individual
- A disponibilidade de acesso aos conteúdos a partir de qualquer dispositivo fixo ou portátil com ligação à Internet

Uma oportunidade excecional para aprender de uma forma cómoda e simples os processos e conhecimentos matemáticos e básicos necessários para realizar uma programação informática de qualidade"

Um Mestrado Próprio que baseia a sua eficácia na tecnologia educativa mais valorizada do mercado com sistemas audiovisuais e de estudo que lhe permitirão aprender mais rápida e comodamente"

Colocamos ao seu serviço material didático amplo e claro, que incorpora todos os tópicos de interesse atuais, para que possa continuar a progredir em Computação e Linguagens.

Um estudo com um elevado impacto educativo que lhe permitirá adaptar o esforço às suas necessidades, combinando flexibilidade e intensidade.

O seu conteúdo multimédia, desenvolvido com a mais recente tecnologia educacional, permitirá ao profissional uma aprendizagem situada e contextual, ou seja, um ambiente simulado que proporcionará uma atualização imersiva programada para treinar em situações reais.

A estrutura deste Mestrado Próprio centra-se na Aprendizagem Baseada em Problemas, na qual o profissional deve tentar resolver as diferentes situações de prática profissional que surgem durante o seu decorrer. Para tal, o profissional será auxiliado por um sistema inovador de vídeos interativos criados por especialistas reconhecidos com vasta experiência em Computação e Linguagens.

02

Objetivos

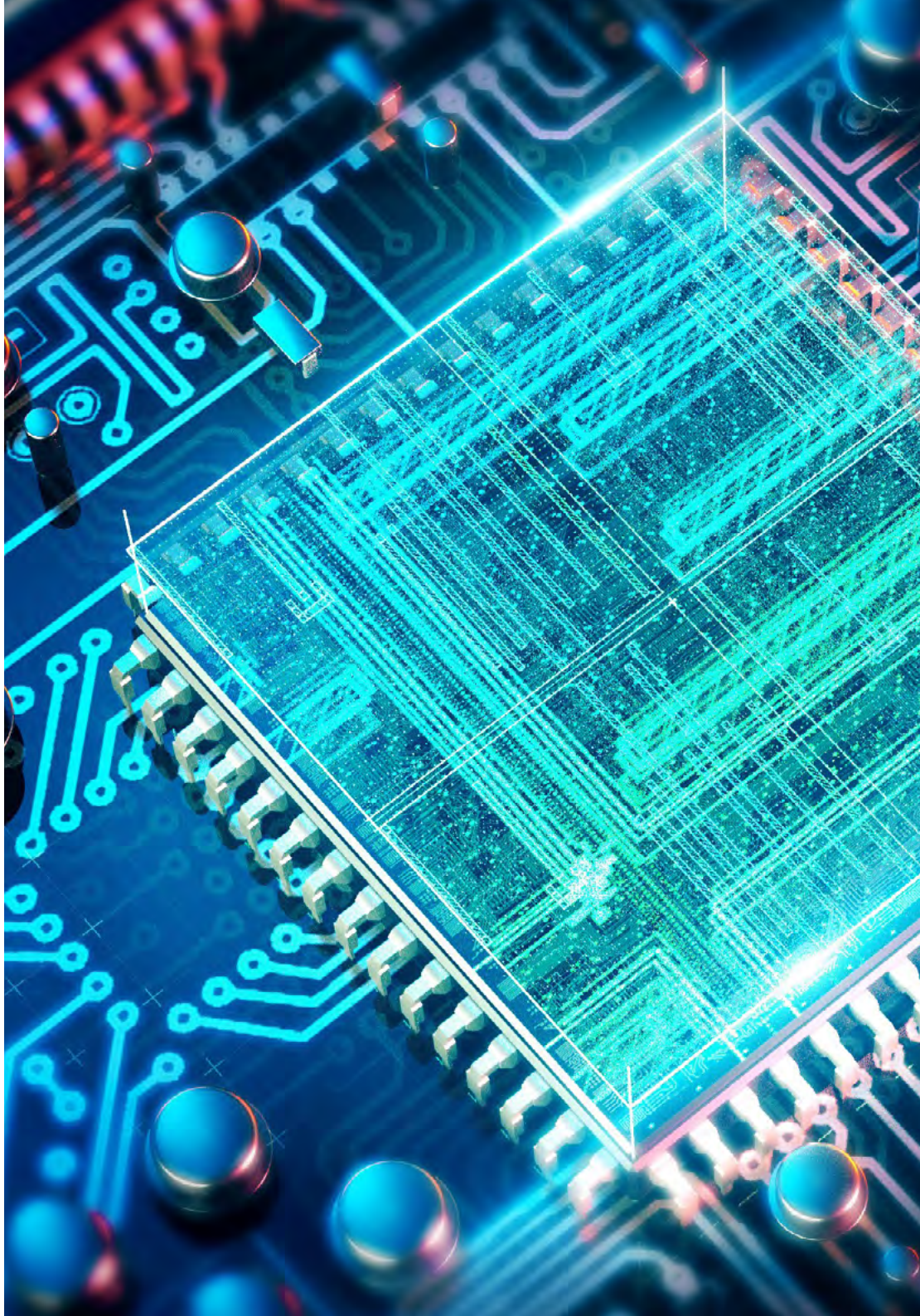
O Mestrado Próprio em Computação e Linguagens foi criado especificamente para o profissional que procura progredir neste domínio rapidamente e com verdadeira qualidade, organizando-o com base em objetivos realistas e de elevado valor que o impulsionarão para outro nível de trabalho neste domínio.

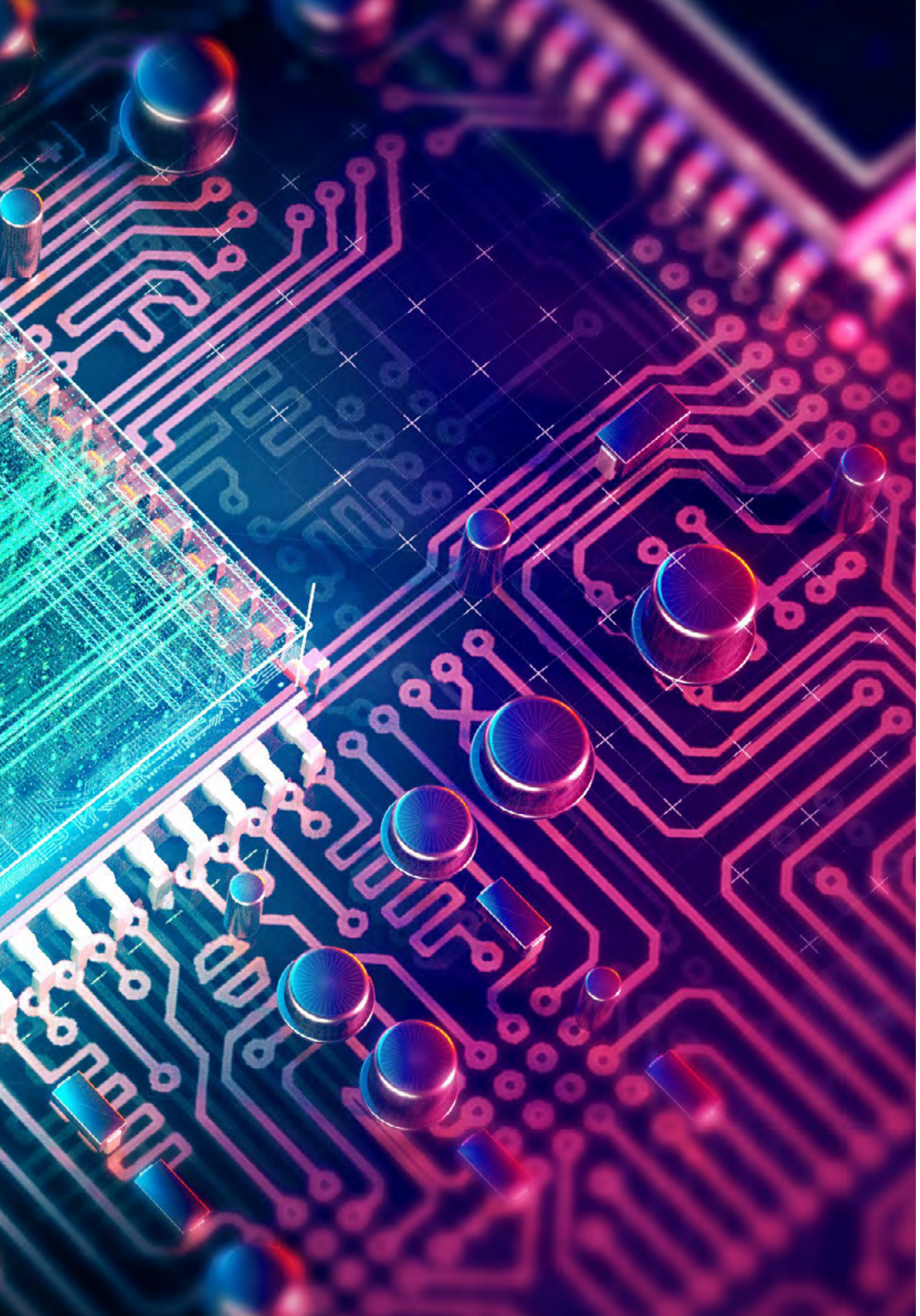
O nosso objetivo é proporcionar aos profissionais da área da informática uma atualização de alta qualidade que lhes permita intervir com competência em Computação e Linguagens"

Objetivo geral

- Capacitar os profissionais científica e tecnologicamente, assim como prepará-los para a prática profissional da Computação e Linguagens, tudo isto com um Mestrado Próprio transversal e versátil adaptado às novas tecnologias e inovações neste campo

Aproveite a oportunidade e avance para se atualizar com os últimos desenvolvimentos em Computação e Linguagens"





Objetivos específicos

Módulo 1. Fundamentos de programação

- Compreender a estrutura básica de um computador, software e linguagens de programação de uso geral
- Aprender a conceber e interpretar algoritmos, que são a base necessária para o desenvolvimento de programas informáticos
- Compreender os elementos essenciais de um programa informático, tal como os diferentes tipos de dados, operadores, expressões, sentenças, I/O e sentenças de controlo
- Compreender as diferentes estruturas de dados disponíveis em linguagens de programação de uso geral, tanto estáticas como dinâmicas, e adquirir conhecimentos essenciais para a gestão de ficheiros
- Compreender as diferentes técnicas de teste nos programas informáticos e a importância de gerar uma boa documentação juntamente com um bom código fonte
- Aprenda os conceitos básicos da linguagem de programação C++, uma das linguagens de programação mais utilizadas em todo o mundo

Módulo 2. Estrutura de dados

- Aprender os fundamentos de programação na linguagem C++, incluindo aulas, variáveis, expressões condicionais e objetos.
- Compreender os tipos de dados abstratos, tipos de estruturas de dados lineares, estruturas de dados hierárquicos simples e complexas e a sua implementação em C++
- Compreender o funcionamento de estruturas de dados avançadas para além das habituais
- Compreender a teoria e a prática relacionadas com a utilização de montículos e filas de espera prioritárias
- Aprender o funcionamento das tabelas Hash, como tipos abstratos de dados e funções
- Compreender a teoria dos grafos, bem como os algoritmos e conceitos avançados sobre grafos

Módulo 3. Algoritmia e complexidade

- Aprender as principais estratégias para a conceção de algoritmos, bem como os diferentes métodos e medidas para o cálculo de algoritmos
- Conhecer os principais algoritmos de ordenação utilizados no desenvolvimento de software
- Compreender o funcionamento dos diferentes algoritmos com árvores, *Heaps* e Grafos
- Compreender o funcionamento dos algoritmos *Greedy*, a sua estratégia e exemplos da sua utilização nos principais problemas conhecidos
- Conhecer também a utilização de algoritmos *Greedy* sobre Grafos
- Aprender as principais estratégias de procura de caminhos mínimos com a abordagem de problemas essenciais do âmbito e algoritmos para a sua resolução
- Entender a técnica de *Backtracking* e as suas principais utilizações, bem como outras técnicas alternativas

Módulo 4. Desenho avançado de algoritmos

- Aprofundar na conceção avançada de algoritmos, analisando algoritmos recursivos e de divisão e conquista, bem como realizar análise amortizada
- Compreender conceitos de programação dinâmica para e os algoritmos para problemas NP
- Entender o funcionamento da optimização combinatória, bem como os diferentes algoritmos de aleatorização e algoritmos paralelos
- Conhecer e compreender o funcionamento dos diferentes métodos de pesquisa locais e com candidatos
- Aprender os mecanismos de verificação formal de programas e de programas iterativos, incluindo a lógica de primeira ordem e o sistema formal de Hoare
- Aprender o funcionamento de alguns dos principais métodos numéricos como o método de bissecção, o método de Newton-Raphson e o método da Secante

Módulo 5. Programação avançada

- Aprofundar conhecimentos em programação, especialmente em relação à programação orientada a objetos, e nos diferentes tipos de relações entre as classes existentes
- Conhecer os diferentes padrões de conceção para problemas orientados a objetos
- Aprender sobre programação orientada a eventos e o desenvolvimento de interfaces de utilizadores com Qt
- Adquirir os conhecimentos essenciais de programação concorrente, os processos e os tópicos
- Aprender a gerir a utilização dos fios a e sincronização, bem como a resolução de problemas comuns no âmbito da programação concorrente
- Compreender a importância da documentação e das provas no desenvolvimento de software

Módulo 6. Informática teórica

- Compreender os conceitos matemáticos teóricos essenciais por detrás da informática, tais como a lógica proposicional, a teoria de conjuntos e os conjuntos numeráveis e não numeráveis
- Compreender os conceitos de línguas e gramáticas formais, bem como as máquinas Turing nas suas diferentes variantes
- Aprender sobre os diferentes tipos de problemas indecidíveis e intratáveis, incluindo as diferentes variantes destes e as suas abordagens
- Compreender o funcionamento de diferentes classes de linguagens baseadas na aleatorização e outros tipos de classes e gramáticas
- Conhecer outros sistemas de computação avançados, tais como a computação com membranas, a computação com ADN e a computação quântica

Módulo 7. Teoria dos autómatos e linguagens formais

- Compreender a teoria dos autómatos e das linguagens formais, aprendendo os conceitos de alfabetos, cadeias e linguagens, bem como a forma de realizar demonstrações formais
- Aprofundar a compreensão dos diferentes tipos de autómatos finitos, sejam eles deterministas ou não deterministas
- Aprender os conceitos básicos e avançados relacionados com as linguagens regulares e as expressões regulares, bem como a aplicação do lema de bombagem e o encerramento das linguagens regulares
- Compreender as gramáticas independentes de contexto, bem como o funcionamento dos autómatos empilhados
- Aprofundar os conhecimentos nas formas normais, o lema de bombeamento das gramáticas independentes do contexto e as propriedades das linguagens independentes do contexto

Módulo 8. Processadores de linguagens

- Introduzir os conceitos relacionados com o processo de compilação e os diferentes tipos de análise: léxica, sintática e semântica
- Conhecer o funcionamento de um analisador lexical, a sua implementação e recuperação de erros
- Aprofundar o conhecimento da análise sintática, tanto descendente como ascendente, mas com especial ênfase nos diferentes tipos de analisadores sintáticos de ascendentes
- Compreender como funcionam os analisadores semânticos, a tradição orientada pela sintaxe, a tabela de símbolos e os diferentes tipos
- Aprender os diferentes mecanismos de geração de códigos, tanto em ambientes de tempo de execução como para a geração de códigos intermédios
- Lançar as bases da optimização do código, incluindo a reordenação da expressão e a optimização de loops

Módulo 9. Informática gráfica e visualização

- Introduzir os conceitos essenciais da informática gráfica e da visualização por computador, tais como a teoria da cor e os seus modelos e as propriedades da luz
- Compreender o funcionamento dos primitivos de saída e os seus algoritmos, tanto para desenhar linhas como para desenhar círculos e preenchimentos
- Aprofundar o estudo das diferentes transformações 2D e 3D e dos seus sistemas de coordenadas e visualização por computador
- Aprenda a fazer projecções e cortes em 3D, bem como a eliminação de superfícies ocultas
- Aprenda a teoria relacionada com a interpolação e curvas paramétricas, bem como relacionada com as Curvas de Bézier e os B-Splines

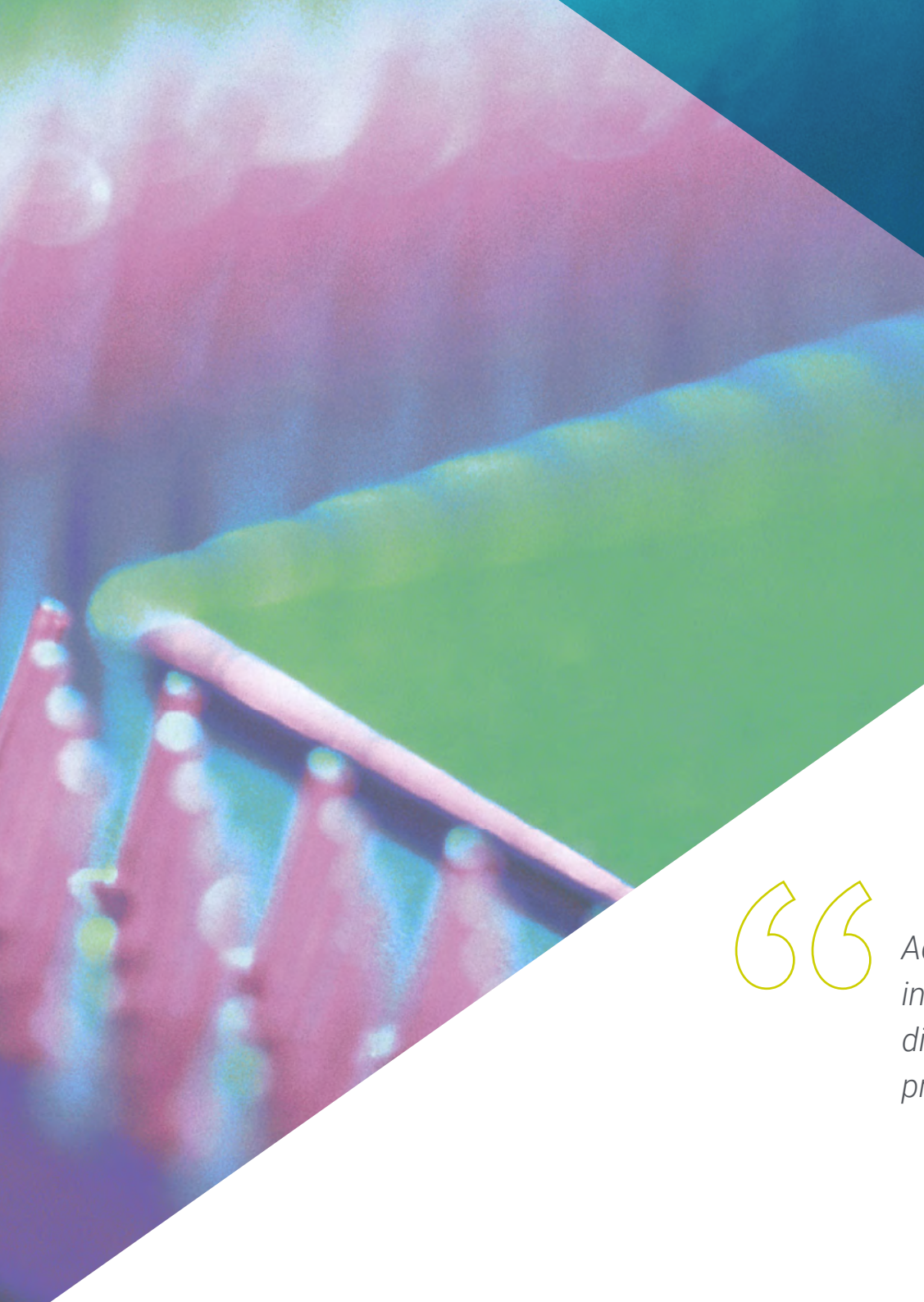
Módulo 10. Computação bioinspirada

- Introduzir o conceito de computação bioinspirada, bem como compreender o funcionamento dos diferentes tipos de algoritmos de adaptação social e de algoritmos genéticos
- Aprofundar o estudo dos diferentes modelos de computação evolutiva, conhecendo as suas estratégias, programação, algoritmos e modelos baseados na estimativa de distribuições
- Compreender as principais estratégias de exploração do espaço para algoritmos genéticos
- Compreender o funcionamento da programação evolutiva aplicada a problemas de aprendizagem e dos problemas multiobjetivos
- Aprender os conceitos essenciais relacionados com redes neurais e compreender como funcionam em casos de uso real aplicados a áreas tão diversas como a investigação médica, a economia e a visão artificial

03

Competências

Após ser aprovado nas avaliações do Mestrado Próprio em Computação e Linguagens, o profissional terá adquirido as competências necessárias para conhecer os princípios fundamentais da computação com a capacidade de trabalhar com linguagens de programação e dados.



Adquira a capacidade de realizar novos desenvolvimentos informáticos a partir da compreensão e do controlo das diferentes linguagens e algoritmos e da sua aplicação prática"

Competências gerais

- Realize corretamente as tarefas relacionadas com a computação e a linguagem informática

Melhore as suas competências para participar em vários projetos tecnológicos"

Competências específicas

- Conceber algoritmos para desenvolver programas informáticos e aplicar a linguagem de programação
- Compreender e utilizar a estrutura dos dados informáticos
- Utilizar os algoritmos necessários para resolver problemas informáticos
- Conhecer profundamente o desenho avançado de algoritmos assim como os métodos de pesquisa
- Levar a cabo tarefas de programação informática
- Compreender e aplicar a teoria que existe por detrás da informática, como é o caso da matemática
- Conhecer a teoria de autómatos e aplicar a linguagem informática
- Conhecer os fundamentos teóricos das linguagens de programação e as técnicas de processamento léxico, sintático e semântico associadas
- Compreender os conceitos básicos da matemática e da complexidade computacional, a fim de os aplicar na resolução de problemas informáticos
- Conhecer e aplicar os princípios fundamentais da computação para levar a cabo novos desenvolvimentos informáticos

04

Direção do curso

Este programa académico conta com o corpo docente mais especializado do mercado educativo atual. São especialistas seleccionados pela TECH para desenvolver todo o itinerário. Desta forma, com base na sua própria experiência e nas mais recentes evidências, conceberam os conteúdos mais actuais que oferecem uma garantia de qualidade numa matéria tão relevante.

A TECH oferece-lhe o corpo docente mais especializado na área de estudo. Inscreva-se já e desfrute da qualidade que merece".

Diretor Internacional Convidado

O Dr. Jeremy Gibbons é considerado uma **eminência internacional** pelas suas contribuições no campo da **Metodologia da Programação** e as suas aplicações na **Engenharia de Software**. Durante mais de duas décadas, este especialista associado ao Departamento de Ciências da Computação da Universidade de Oxford impulsionou **diferentes projetos de desenvolvimento** cujos resultados mais palpáveis são aplicados por informáticos de diversas partes do mundo.

O seu trabalho abrange áreas como a **programação genérica**, os métodos formais, a biologia computacional, a bioinformática e a conceção de algoritmos com Haskell. Este último foi desenvolvido extensivamente em conjunto com o seu mentor, o Dr. Richard Bird.

Na sua função de **Diretor** do Grupo de Investigação em Álgebra de Programação, Gibbons propiciou avanços relacionados ás **Linguagens de Programação Funcional** e à **Teoria de Padrões em Programação**. Ao mesmo tempo, as aplicações das suas inovações têm sido ligadas ao setor de saúde, como demonstra a sua colaboração com o **CancerGrid** e o **Datatype-Generic Programming**. Estas e outras iniciativas refletem o seu interesse em resolver problemas práticos na **investigação do cancro** e na **informática clínica**.

Além disso, Gibbons também deixou uma marca significativa como **Editor Chefe** de publicações académicas no **The Journal of Functional Programming** e no **The Programming Journal**: **The Art**, **Science**, **and Engineering of Programming**. Através dessas responsabilidades, realizou uma intensa divulgação e disseminação do conhecimento. Além disso, liderou várias cátedras de estudo vinculadas a instituições de renome, como a **Universidade Oxford Brookes** e a **Universidade de Auckland**, **Nova Zelândia**.

É também membro do **Grupo de Trabalho 2.1 sobre Linguagens Algorítmicas e Cálculos da Federação Internacional para o Processamento da Informação (IFIP)**. Com esta organização, assegura a manutenção das linguagens de programação **ALGOL 60** e **ALGOL 68**.

Dr. Gibbons, Jeremy

- Diretor do Programa de Engenharia de Software da Universidade de Oxford, Reino Unido
- Subdiretor do Laboratório de Informática e do Departamento de Ciências da Computação, Universidade de Oxford
- Professor no Kellogg College, na Oxford Brookes University e na Universidade de Auckland, Nova Zelândia
- Diretor do Grupo de Investigação em Álgebra de Programação
- Editor-chefe das revistas The Art, Science, and Engineering of Programming e Journal of Functional Programming
- Doutoramento em Ciências da Computação pela Universidade de Oxford
- Licenciatura em Ciências da Computação, Universidade de Edimburgo
- Membro de:
  - Grupo de Trabalho 2.1 sobre Linguagens Algorítmicas e Computação da Federação Internacional de Processamento de Informação (IFIP)

Graças à TECH, poderá aprender com os melhores profissionais do mundo"

05

Estrutura e conteúdo

A estrutura dos conteúdos foi criada de forma a que os conhecimentos sejam assimilados progressivamente, conseguindo uma trajetória de crescimento que o levará à excelência na sua profissão.

Todas as áreas de interesse que precisa de dominar para trabalhar com segurança em Computação e Linguagens compiladas num Mestrado Próprio de alta qualidade"

Módulo 1. Fundamentos de programação

1.1. Introdução à programação
- 1.1.1. Estrutura básica de um computador
- 1.1.2. Software
- 1.1.3. Linguagens de programação
- 1.1.4. Ciclo de vida de uma aplicação informática

1.2. Design de algoritmos
- 1.2.1. A resolução de problemas
- 1.2.2. Técnicas descritivas
- 1.2.3. Elementos e estrutura de um algoritmo

1.3. Elementos de um programa
- 1.3.1. Origem e caraterísticas da linguagem C++
- 1.3.2. O ambiente de desenvolvimento
- 1.3.3. Conceito de programa
- 1.3.4. Tipos de dados fundamentais
- 1.3.5. Operadores
- 1.3.6. Expressões
- 1.3.7. Instruções
- 1.3.8. Entrada e saída de dados

1. 4. Sentenças de controlo
- 1.4.1. Instruções
- 1.4.2. Bifurcações
- 1.4.3. Ciclos

1,5. Abstração e modularidade: funções
- 1.5.1. Design modular
- 1.5.2. Conceito de função e utilidade
- 1.5.3. Definição de uma função
- 1.5.4. Fluxo de execução numa chamada de função
- 1.5.5. Protótipo de uma função
- 1.5.6. Devolução de resultados
- 1.5.7. Chamada de uma função: parâmetros
- 1.5.8. Passagem de parâmetros por referência e por valor
- 1.5.9. Âmbito de identificação

1.6. Estruturas de dados estáticas
- 1.6.1. *Matrizes*
- 1.6.2. Matrizes. Poliedros
- 1.6.3. Pesquisa e ordenação
- 1.6.4. Strings. Funções de E/S para strings
- 1.6.5. Estruturas. Uniões
- 1.6.6. Novos tipos de dados

1.7. Estruturas de dados dinâmicas: ponteiros
- 1.7.1. Conceito. Definição de ponteiro
- 1.7.2. Operadores e operações com ponteiros
- 1.7.3. Matrizes de ponteiros
- 1.7.4. Ponteiros e Matrizes
- 1.7.5. Ponteiros para strings
- 1.7.6. Ponteiros para estruturas
- 1.7.7. Indireção múltipla
- 1.7.8. Ponteiros para funções
- 1.7.9. Passagem de funções, estruturas e matrizes como parâmetros de funções

1.8. Ficheiros
- 1.8.1. Conceitos básicos
- 1.8.2. Operações com ficheiros
- 1.8.3. Tipos de ficheiros
- 1.8.4. Organização dos ficheiros
- 1.8.5. Introdução aos ficheiros C++
- 1.8.6. Tratamento de ficheiros

1.9. Recursividade
- 1.9.1. Definição de recursividade
- 1.9.2. Tipos de recursão
- 1.9.3. Vantagens e desvantagens
- 1.9.4. Considerações
- 1.9.5. Conversão recursivo-iterativa
- 1.9.6. A pilha de recursão

1.10. Prova e documentação
- 1.10.1. Teste de programas
- 1.10.2. Teste de caixa branca
- 1.10.3. Teste de caixa negra
- 1.10.4. Ferramentas de teste
- 1.10.5. Documentação de programas

Módulo 2. Estrutura de dados

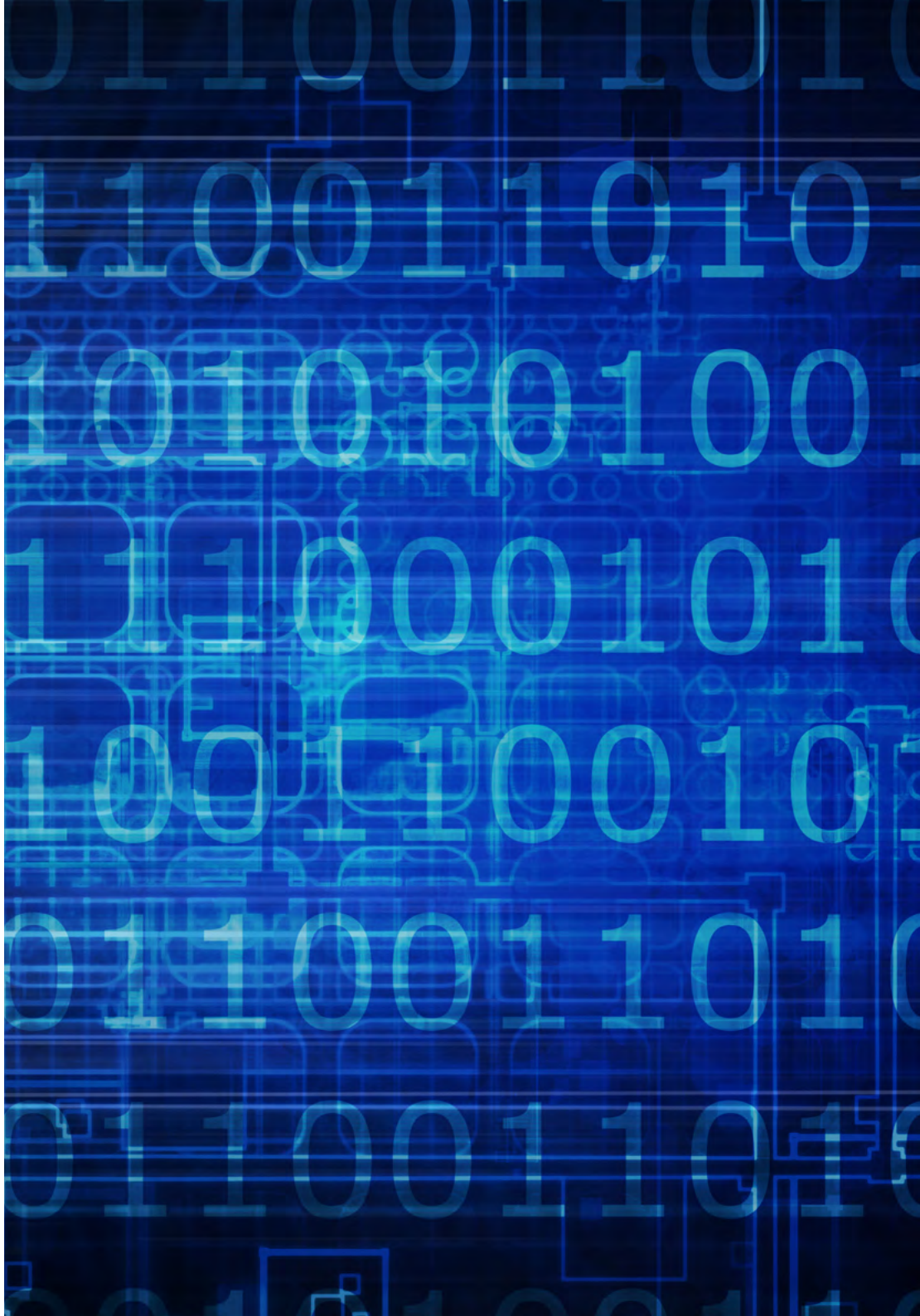
2.1. Introdução à programação em C++
- 2.1.1. Classes, construtores, métodos e atributos
- 2.1.2. Variáveis
- 2.1.3. Expressões condicionais e loops
- 2.1.4. Objetos

2.2. Tipos abstratos de dados (TAD)
- 2.2.1. Tipos de dados
- 2.2.2. Estruturas básicas e TAD
- 2.2.3. Vetores e arrays

2.3. Estruturas de dados lineares
- 2.3.1. TAD Lista. Definição
- 2.3.2. Listas ligadas e duplamente ligadas
- 2.3.3. Listas ordenadas
- 2.3.4. Listas em C++
- 2.3.5. TAD pilha
- 2.3.6. TAD fila
- 2.3.7. Pilha e fila em C++

2.4. Estruturas de dados hierárquicas
- 2.4.1. TAD árvore
- 2.4.2. Caminhos
- 2.4.3. Árvores n-arios
- 2.4.4. Árvores binários
- 2.4.5. Árvores binários de pesquisa

2.5. Estruturas de dados hierárquicas: árvores complexas
  2.5.1. Árvores perfeitamente equilibradas ou de altura mínima
  2.5.2. Árvores multicaminho
  2.5.3. Referências bibliográficas
2.6. Heaps e fila prioritários
  2.6.1. TAD heaps
  2.6.2. TAD fila prioritária
2.7. Tabelas hash
  2.7.1. TAD Tabela *Hash*
  2.7.2. Funções *Hash*
  2.7.3. Função *Hash* em tabelas *Hash*
  2.7.4. Redispersão
  2.7.5. Tabelas *Hash* abertas
2.8. Grafos
  2.8.1. TAD Grafo
  2.8.2. Tipos de grafo
  2.8.3. Representação gráfica e operações básicas
  2.8.4. Desenho de grafos
2.9. Algoritmos e conceitos avançados sobre grafos
  2.9.1. Problemas sobre grafos
  2.9.2. Algoritmos sobre caminhos
  2.9.3. Algoritmos de pesquisa ou caminhos
  2.9.4. Outros algoritmos
2.10. Outras estruturas de dados
  2.10.1. Conjuntos
  2.10.2. Arrays paralelos
  2.10.3. Tabela de símbolos
  2.10.4. *Tries*



Módulo 3. Algoritmia e complexidade

3.1. Introdução às estratégias de desenho do algoritmos
- 3.1.1. Recursividade
- 3.1.2. Divide e conquista
- 3.1.3. Outras estratégias

3.2. Eficiência e análise dos algoritmos
- 3.2.1. Medidas de eficiência
- 3.2.2. Medir o tamanho da entrada
- 3.2.3. Medir o tempo de execução
- 3.2.4. Caso pior, melhor e médio
- 3.2.5. Notação assintótica
- 3.2.6. Critérios de análise matemática de algoritmos não recursivos
- 3.2.7. Análise matemática de algoritmos recursivos
- 3.2.8. Análise empírica de algoritmos

3.3. Algoritmos de ordenação
- 3.3.1. Conceito de ordenação
- 3.3.2. Ordenação da bolha
- 3.3.3. Ordenação por seleção
- 3.3.4. Ordenação por inserção
- 3.3.5. Ordenação por mistura (Merge Sort)
- 3.3.6. Ordenação rápida (Quicksort)

3.4. Algoritmos com árvores
- 3.4.1. Conceito de árvore
- 3.4.2. Árvores binários
- 3.4.3. Caminhos de árvore
- 3.4.4. Representar expressões
- 3.4.5. Árvores binários ordenadas
- 3.4.6. Árvores binárias equilibradas

3.5. Algoritmos com *Heaps*
- 3.5.1. Os *Heaps*
- 3.5.2. O algoritmo Heapsort
- 3.5.3. As filas de prioridade

3.6. Algoritmos com Grafos
- 3.6.1. Representação
- 3.6.2. Caminho de largura
- 3.6.3. Caminho de profundidade
- 3.6.4. Ordenação topológica

3.7. Algoritmos *Greedy*
- 3.7.1. A estratégia *Greedy*
- 3.7.2. Elementos da estratégia *Greedy*
- 3.7.3. Câmbio de moedas
- 3.7.4. Problema do viajante
- 3.7.5. Problema da mochila

3.8. Pesquisa de caminhos mínimos
- 3.8.1. O problema do caminho mínimo
- 3.8.2. Arcos negativos e ciclos
- 3.8.3. Algoritmo de Dijkstra

3.9. Algoritmos greedy sobre grafos
- 3.9.1. A árvore de extensão mínima
- 3.9.2. O algoritmo de Prim
- 3.9.3. O algoritmo Kruskal
- 3.9.4. Análise de complexidade

3.10. *Backtracking*
- 3.10.1. O *Backtracking*
- 3.10.2. Técnicas alternativas

Módulo 4. Design avançado de algoritmos

4.1. Análise de algoritmos recursivos e de divisão e conquista
- 4.1.1. Posicionar e resolver equações de recorrência homogéneas e não homogéneas
- 4.1.2. Descrição geral da estratégia divisão e conquista

4.2. Análise amortizado
- 4.2.1. A análise agregada
- 4.2.2. O método de contabilidade
- 4.2.3. O método do potencial

4.3. Programação dinâmica e algoritmos para problemas NP
- 4.3.1. Caraterísticas da programação dinâmica
- 4.3.2. Volta atrás: backtracking
- 4.3.3. Ramificação e poda

4.4. Otimização combinatória
- 4.4.1. Representação de problemas
- 4.4.2. Optimizaçáo em 1D

4.5. Algoritmos de aleatorização
- 4.5.1. Exemplos de algoritmos de aleatorização
- 4.5.2. O teorema Buffon
- 4.5.3. Algoritmo de Monte Carlo
- 4.5.4. Algoritmo Las Vegas

4.6. Pesquisa local e com candidatos
- 4.6.1. *Garcient Ascent*
- 4.6.2. *Hill Climbing*
- 4.6.3. *Simulated Annealing*
- 4.6.4. *Tabu Search*
- 4.6.5. Pesquisa com candidatos

4.7. Verificação formal de programas
- 4.7.1. Especificação de abstrações funcionais
- 4.7.2. A linguagem da lógica de primeira ordem
- 4.7.3. O sistema formal de Hoare

4.8. Verificação de programas iterativos
- 4.8.1. Regras do sistema formal de Hoare
- 4.8.2. Conceito de invariante de iterações

4.9. Métodos numéricos
- 4.9.1. O método da bisecção
- 4.9.2. O método de Newton-Raphson
- 4.9.3. O método das secantes

4.10. Algoritmos paralelos
- 4.10.1. Operações binárias paralelas
- 4.10.2. Operações paralelas com grafos
- 4.10.3. Paralelismo em divisão e conquista
- 4.10.4. Paralelismo em programação e dinâmica

Módulo 5. Programação avançada

5.1. Introdução à programação orientada a objetos
- 5.1.1. Introdução à programação orientada a objetos
- 5.1.2. Desenho de classes
- 5.1.3. Introdução à UML para modelação de problemas

5.2. Relações entre classes
- 5.2.1. Abstração e herança
- 5.2.2. Conceitos avançados de herança
- 5.2.3. Poliformismo
- 5.2.4. Composição e agregação

5.3. Introdução aos padrões de de desenho para problemas orientados a objetos
- 5.3.1. O que são padrões de design?
- 5.3.2. Padrão *Factory*
- 5.3.3. Padrão *Singleton*
- 5.3.4. Padrão *Observer*
- 5.3.5. Padrão *Composite*

5.4. Exceções
- 5.4.1. O que são as exceções
- 5.4.2. Captura e gestão de exceções
- 5.4.3. Lançamento de exceções
- 5.4.4. Criação de exceções

5.5. Interfaces de utilizadores
- 5.5.1. Introdução a Qt
- 5.5.2. Posicionamento

5.5.3. O que são os eventos?
5.5.4. Eventos: definição e captura
5.5.5. Desenvolvimento de interfaces de utilizador
5.6. Introdução à programação concorrente
5.6.1. Introdução à programação concorrente
5.6.2. O conceito de processo e thread
5.6.3. Interação entre processos ou threads
5.6.4. Os threads em C++
5.6.5. Vantagens e desvantagens da programação concorrente
5.7. Gestão de threads e sincronização
5.7.1. Ciclo de vida um thread
5.7.2. A classe *Thread*
5.7.3. Planificação de threads
5.7.4. Grupos threads
5.7.5. Threads de tipo demónio
5.7.6. Sincronização
5.7.7. Mecanismos de bloqueio
5.7.8. Mecanismos de comunicação
5.7.9. Monitores
5.8. Problemas comuns dentro da programação concorrente
5.8.1. O problema dos produtores consumidores
5.8.2. O problema dos leitores e escritores
5.8.3. O problema do jantar dos filósofos
5.9. Documentação e provas de software
5.9.1. Porque é que é importante documentar o software?
5.9.2. Documentação de design
5.9.3. Uso de ferramentas para a documentação
5.10. Provas de software
5.10.1. Introdução às provas de software
5.10.2. Tipos de provas
5.10.3. Prova de unidade
5.10.4. Prova de integração
5.10.5. Prova de validação
5.10.6. Prova do sistema

Módulo 6. Informática teórica

6.1. Conceitos matemáticos utilizados
6.1.1. Introdução à lógica proposicional
6.1.2. Teoria de relações
6.1.3. Conjuntos numeráveis e não numeráveis
6.2. Línguas e gramáticas formais e introdução às máquinas Turing
6.2.1. Linguagens e gramáticas formais
6.2.2. Problema de decisão
6.2.3. A máquina de Turing
6.3. Extensões para as máquinas de Turing, máquinas de Turing restringidas e computadores
6.3.1. Técnicas de programação para as máquinas de Turing
6.3.2. Extensões para as máquinas de Turing
6.3.3. Máquinas de Turing restringidas
6.3.4. Máquinas de Turing e computadores
6.4. Indecidibilidade
6.4.1. Linguagem não recursivamente enumerável
6.4.2. Um problema indecidível, recursivamente enumerável
6.5. Outros problemas indecidíveis
6.5.1. Problema indecidíveis para as máquinas de Turing
6.5.2. Problemas de correspondência de Post (PCP)
6.6. Problemas intratáveis
6.6.1. As classes P e NP
6.6.2. Um problema NP completo
6.6.3. Problema da satisfação restrita
6.6.4. Outros problemas NP completos
6.7. Problemas co-NP e PS
6.7.1. Complementares das linguages de NP
6.7.2. Problemas resolúveis no espaço polinomial
6.7.3. Problema PS completos
6.8. Classes de linguagens baseadas na aleatorização
6.8.1. Modelo da MT com aleatoriedade
6.8.2. As classes RP e ZPP
6.8.3. Teste de primalidade
6.8.4. Complexidade do teste de primalidade

6.9. Outras classes e gramáticas
- 6.9.1. Autómatos finitos e probabilísticos
- 6.9.2. Autómatos celulares
- 6.9.3. Células de McCulloch e Pitts
- 6.9.4. Gramáticas de Lindenmayer

6.10. Sistemas avançados de computação
- 6.10.1. Computação com membranas: sistemas P
- 6.10.2. Computação com ADN
- 6.10.3. Computação quântica

Módulo 7. Teoria dos autómatos e linguagens formais

7.1. Introdução à teoria da autómatos
- 7.1.1. Porque estudar teoria de autómatos
- 7.1.2. Introdução às demonstrações formais
- 7.1.3. Outras formas de demonstração
- 7.1.4. Indução matemática
- 7.1.5. Alfabetos, cadeias e linguagens

7.2. Autómatos finitos e deterministas
- 7.2.1. Introdução aos autómatos finitos
- 7.2.2. Autómatos finitos e deterministas

7.3. Autómatos finitos e deterministas
- 7.3.1. Autómatos finitos e deterministas
- 7.3.2. Equivalência entre AFD e AFN
- 7.3.3. Autómatos finitos com transições

7.4. Linguagens e expressões regulares (I)
- 7.4.1. Linguagens e expressões regulares
- 7.4.2. Autómatos finitos e expressões regulares

7.5. Linguagens e expressões regulares (II)
- 7.5.1. Conversão de expressões regulares em autómatos
- 7.5.2. Aplicações das expressões regulares
- 7.5.3. Álgebra das expressões regulares

7.6. Lema do bombeamento e encerramento de linguagens regulares
- 7.6.1. Lema do bombeamento
- 7.6.2. Propriedades de fecho das linguagens regulares

7.7. Equivalência e minimização de autómatos
- 7.7.1. Equivalência de AF
- 7.7.2. Minimização de AF

7.8. Gramáticas independentes de contexto (GIC)
- 7.8.1. Gramáticas independentes de contexto
- 7.8.2. Árvores de derivação
- 7.8.3. Aplicações das GIC
- 7.8.4. Ambiguidade nas gramáticas e linguagens

7.9. Autómatos com pilha e GIC
- 7.9.1. Definição dos autómatos com pilha
- 7.9.2. Linguagens aceites por um autómato com pilha
- 7.9.3. Equivalência entre autómatos a pilha e GIC
- 7.9.4. Autómato com pilha determinista

7.10. Formas normais, lema do bombeamento das GIC e propriedades dos LIC
- 7.10.1. Formas normais das GIC
- 7.10.2. Lema do bombeamento
- 7.10.3. Propriedades de fecho das linguagens
- 7.10.4. Propriedades de fecho dos LIC

Módulo 8. Processadores de linguagens

8.1. Introdução ao processo de compilação
- 8.1.1. Compilação e interpretação
- 8.1.2. Ambiente de execução de um compilador
- 8.1.3. Processo de análise
- 8.1.4. Processo de síntese

8.2. Analisador léxico
- 8.2.1. O que é um analisador léxico?
- 8.2.2. Implementação do analisador léxico
- 8.2.3. Ações semânticas
- 8.2.4. Recuperação de erros
- 8.2.5. Questões de implementação

8.3. Análise sintático
- 8.3.1. O que é um analisador sintático?
- 8.3.2. Conceitos prévios
- 8.3.3. Analisadores descendentes
- 8.3.4. Analisadores ascendentes

8.4. Análise sintático de descendente e análise sintático ascendente
- 8.4.1. Analisador LL (1)
- 8.4.2. Analisador LR (0)
- 8.4.3. Exemplo de analisador

8.5. Análise sintático ascendente avançado
- 8.5.1. Analisador SLR
- 8.5.2. Analisador LR (1)
- 8.5.3. Analisador LR (k)
- 8.5.4. Analisador LALR

8.6. Análise semântica (I)
- 8.6.1. Tradução dirigida pela sintaxe
- 8.6.2. Tabela de símbolos

8.7. Análise semântica (II)
- 8.7.1. Comprovação de tipos
- 8.7.2. O subsistema de tipos
- 8.7.3. Equivalência de tipos e conversões

8.8. Geração de código e ambiente de execução
- 8.8.1. Aspetos de desenho
- 8.8.2. Ambiente de execução
- 8.8.3. Organização da memória
- 8.8.4. Atribuição de memória

8.9. Geração de código intermédio
- 8.9.1. Tradução dirigida pela sintaxe
- 8.9.2. Representações intermédias
- 8.9.3. Exemplos de traduções

8.10. Optimização de código
- 8.10.1. Atribuição de registos
- 8.10.2. Eliminação de atribuições mortas
- 8.10.3. Execução em tempo de compilação
- 8.10.4. Reordenação de expressões
- 8.10.5. Optimização de loops

Módulo 9. Informática gráfica e visualização

9.1. Teoria das cores
- 9.1.1. Propriedades da luz
- 9.1.2. Modelos a cor
- 9.1.3. O padrão CIE
- 9.1.4. *Profiling*

9.2. Primitivos de saída
- 9.2.1. O controlador de vídeo
- 9.2.2. Algoritmos de design de linhas
- 9.2.3. Algoritmos de design de circunferências
- 9.2.4. Algoritmos de preenchimento

9.3. Transformações 2D e sistemas de coordenadas 2D e recorte 2D
- 9.3.1. Transformações geométricas básicas
- 9.3.2. Coordenadas homogéneas
- 9.3.3. Transformação inversa
- 9.3.4. Composição de transformações
- 9.3.5. Outras transformações
- 9.3.6. Mudança de coordenada
- 9.3.7. Sistemas de coordenadas 2D
- 9.3.8. Mudança de coordenadas
- 9.3.9. Normalização
- 9.3.10. Algoritmos de recorte

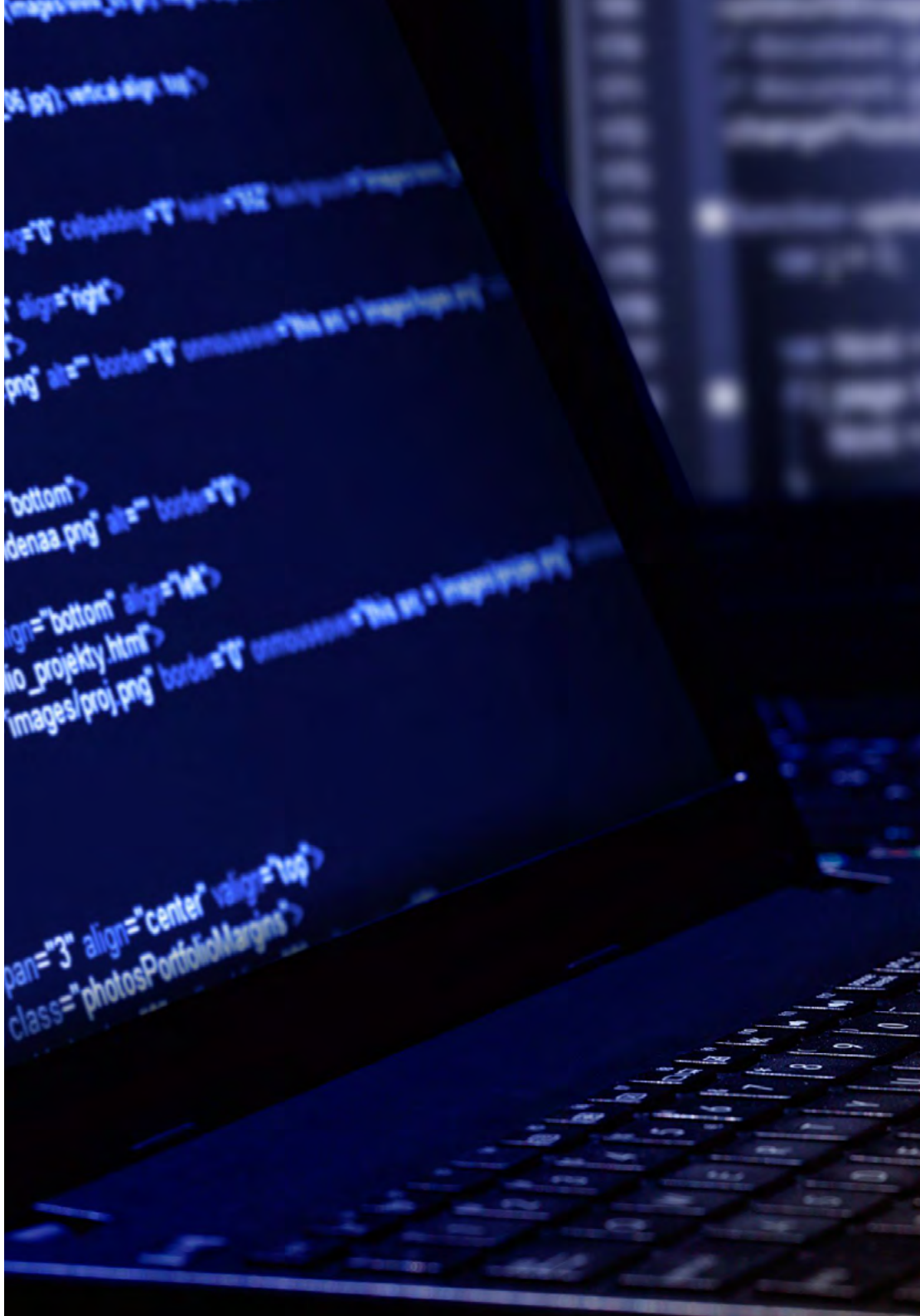
9.4. Transformações 3D
- 9.4.1. Translação
- 9.4.2. Rotação
- 9.4.3. Escalonamento
- 9.4.4. Reflexão
- 9.4.5. Cizalla

9.5. Visualização e mudança de coordenadas 3D
- 9.5.1. Sistemas de coordenadas 3D
- 9.5.2. Visualização
- 9.5.3. Mudança de coordenadas
- 9.5.4. Projeção e normalização

9.6. Projeção e recorte 3D
- 9.6.1. Projeção ortogonal
- 9.6.2. Projeção paralela oblíqua
- 9.6.3. Projeçáo perspetiva
- 9.6.4. Algoritmos de recorte 3D

9.7. Eliminação de superfícies ocultas
- 9.7.1. *Back face removal*
- 9.7.2. Z-buffer
- 9.7.3. Algoritmo do pintor
- 9.7.4. Algoritmo de Warnock
- 9.7.5. Deteção de linhas ocultas

9.8. Interpolação e curvas paramétricas
- 9.8.1. Interpolação e aproximação polinomial
- 9.8.2. Representação paramétrica
- 9.8.3. Polinómio de Lagrange
- 9.8.4. *Splines* cúbicos naturais
- 9.8.5. Funçoes base
- 9.8.6. Representação matricial



9.9. Curvas Bézier
- 9.9.1. Construção algébrica
- 9.9.2. Forma matricial
- 9.9.3. Composição
- 9.9.4. Construção geométrica
- 9.9.5. Algoritmo de desenho

9.10. *B-Splines*
- 9.10.1. O problema do controlo local
- 9.10.2. B-splines cúbicos uniformes
- 9.10.3. Funções base e pontos de controlo
- 9.10.4. Derivação para a origem e multiplicidade
- 9.10.5. Representação matricial
- 9.10.6. *B-splines* não uniformes

Módulo 10. Computação bioinspirada

10.1. Introdução à computação bioinspirada
- 10.1.1. Introdução à computação bioinspirada

10.2. Algoritmos de inspiração social
- 10.2.1. Computação bioinspirada baseada em colónias de formigas
- 10.2.2. Variantes dos algoritmos de colónias de formigas
- 10.2.3. Computação baseada em nuvens de partículas

10.3. Algoritmos genéticos
- 10.3.1. Estrutura geral
- 10.3.2. Implementações dos principais operadores

10.4. Estratégias de exploração do espaço para algoritmos genéticos
- 10.4.1. Algoritmo CHC
- 10.4.2. Problemas multimodais

10.5. Modelos de computação evolutiva (I)
- 10.5.1. Estratégias evolutivas
- 10.5.2. Programação evolutiva
- 10.5.3. Algoritmos baseados em evolução diferencial

10.6. Modelos de computação evolutiva (II)
- 10.6.1. Modelos de evolução baseados na estimativa das distribuições (EDA)
- 10.6.2. Programação genética

10.7. Programação evolutiva aplicada a problemas de aprendizagem
- 10.7.1. Aprendizagem baseada em regras
- 10.7.2. Métodos evolutivos em problemas de seleção de instâncias

10.8. Problemas multiobjetivo
- 10.8.1. Conceito de dominância
- 10.8.2. Aplicação de algoritmos evolutivos a problemas multiobjetivos

10.9. Redes neuronais (I)
- 10.9.1. Introdução às redes neuronais
- 10.9.2. Exemplo prático com redes neuronais

10.10. Redes neuronais (II)
- 10.10.1. Casos de utilização de redes neuronais na investigação médica
- 10.10.2. Casos de utilização de redes neuronais na economia
- 10.10.3. Casos de utilização de redes neuronais na visão artificial

06

Metodologia

Este programa de capacitação oferece uma forma diferente de aprendizagem. A nossa metodologia é desenvolvida através de um modo de aprendizagem cíclico: ***o Relearning***. Este sistema de ensino é utilizado, por exemplo, nas escolas médicas mais prestigiadas do mundo e tem sido considerado um dos mais eficazes pelas principais publicações, tais como a ***New England Journal of Medicine.***

Descubra o Relearning, um sistema que abandona a aprendizagem linear convencional para o levar através de sistemas de ensino cíclicos: uma forma de aprendizagem que provou ser extremamente eficaz, especialmente em disciplinas que requerem memorização"

Estudo de Caso para contextualizar todo o conteúdo

O nosso programa oferece um método revolucionário de desenvolvimento de competências e conhecimentos. O nosso objetivo é reforçar as competências num contexto de mudança, competitivo e altamente exigente.

Com a TECH pode experimentar uma forma de aprendizagem que abala as fundações das universidades tradicionais de todo o mundo"

Terá acesso a um sistema de aprendizagem baseado na repetição, com ensino natural e progressivo ao longo de todo o programa de estudos.

O estudante aprenderá, através de atividades de colaboração e casos reais, a resolução de situações complexas em ambientes empresariais reais.

Um método de aprendizagem inovador e diferente

Este programa da TECH é um programa de ensino intensivo, criado de raiz, que propõe os desafios e decisões mais exigentes neste campo, tanto a nível nacional como internacional. Graças a esta metodologia, o crescimento pessoal e profissional é impulsionado, dando um passo decisivo para o sucesso. O método do caso, a técnica que constitui a base deste conteúdo, assegura que a realidade económica, social e profissional mais atual é seguida.

> *"O nosso programa prepara-o para enfrentar novos desafios em ambientes incertos e alcançar o sucesso na sua carreira"*

O método do caso tem sido o sistema de aprendizagem mais amplamente utilizado nas principais escolas de informática do mundo desde que existem. Desenvolvido em 1912 para que os estudantes de direito não só aprendessem o direito com base no conteúdo teórico, o método do caso consistia em apresentar-lhes situações verdadeiramente complexas, a fim de tomarem decisões informadas e valorizarem juízos sobre a forma de as resolver. Em 1924 foi estabelecido como um método de ensino padrão em Harvard.

Numa dada situação, o que deve fazer um profissional? Esta é a questão que enfrentamos no método do caso, um método de aprendizagem orientado para a ação. Ao longo do programa, os estudantes serão confrontados com múltiplos casos da vida real. Terão de integrar todo o seu conhecimento, investigar, argumentar e defender as suas ideias e decisões.

Relearning Methodology

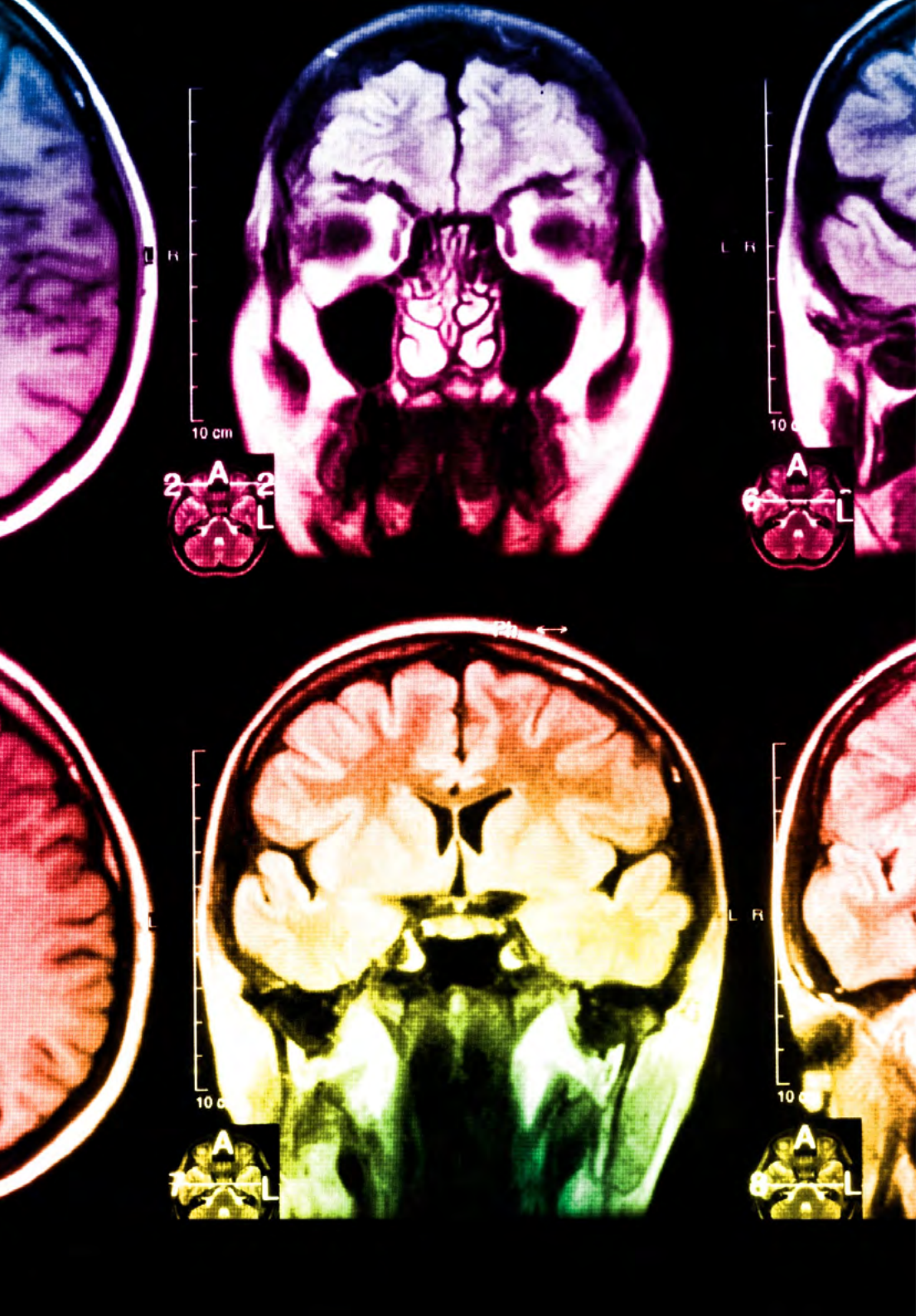
A TECH combina eficazmente a metodologia do Estudo de Caso com um sistema de aprendizagem 100% online baseado na repetição, que combina elementos didáticos diferentes em cada lição.

Potenciamos os Casos Práticos com o melhor método de ensino 100% online: o Relearning.

Em 2019 alcançámos os melhores resultados de aprendizagem de todas as universidades online de língua espanhola do mundo.

Na TECH aprenderá com uma metodologia de vanguarda concebida para formar os gestores do futuro. Este método, pioneiro na pedagogia mundial, chama-se Relearning.

A nossa universidade é a única licenciada para utilizar este método de sucesso. Em 2019, conseguimos melhorar os níveis globais de satisfação dos nossos estudantes (qualidade de ensino, qualidade dos materiais, estrutura dos cursos, objetivos...) no que diz respeito aos indicadores da melhor universidade online em espanhol.



No nosso programa, a aprendizagem não é um processo linear, mas acontece numa espiral (aprender, desaprender, esquecer e reaprender). Por isso, combinamos cada um destes elementos de forma concêntrica. Com esta metodologia formamos mais de 650.000 alunos com um sucesso sem precedentes em áreas tão diversas como Bioquímica, Genética, Cirurgia, Direito Internacional, Competências de Gestão, Ciências Desportivas, Filosofia, Direito, Engenharias, Jornalismo, História ou Mercados e Instrumentos Financeiros. Tudo isto num ambiente altamente exigente, com um corpo estudantil universitário com um elevado perfil socioeconómico e uma idade média de 43,5 anos.

A reaprendizagem permitir-lhe-á aprender com menos esforço e mais desempenho, envolvendo-o mais na sua capacitação, desenvolvendo um espírito crítico, defendendo argumentos e opiniões contrastantes: uma equação direta rumo ao sucesso.

A partir das últimas provas científicas no campo da neurociência, não só sabemos como organizar informação, ideias, imagens e memórias, mas sabemos que o lugar e o contexto em que aprendemos algo é fundamental para a nossa capacidade de o recordar e armazenar no hipocampo, para o reter na nossa memória a longo prazo.

Desta forma, e no que se chama Neurocognitive context-dependent e-learning, os diferentes elementos do nosso programa estão ligados ao contexto em que o participante desenvolve a sua prática profissional.

Este programa oferece o melhor material educacional, cuidadosamente preparado para profissionais:

Material de estudo

Todos os conteúdos didáticos são criados pelos especialistas que irão ministrar o curso, em específico para o mesmo, para que o desenvolvimento didático seja realmente específico e concreto.

Estes conteúdos são então aplicados em formato audiovisual, para criar o método de trabalho online da TECH. Tudo isto, com as mais recentes técnicas que oferecem componentes de alta-qualidade em cada um dos materiais que são colocados à disposição do aluno.

Masterclasses

Existem provas científicas acerca da utilidade da observação por terceiros especialistas.

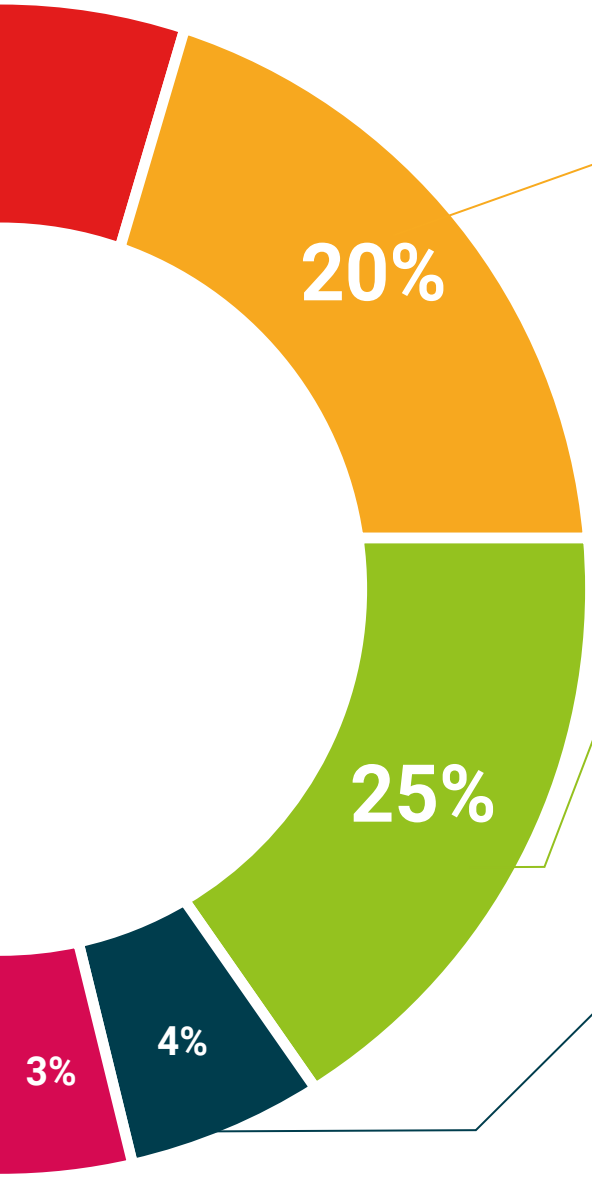
O que se designa de Learning from an Expert fortalece o conhecimento e a recordação, e constrói a confiança em futuras decisões difíceis.

Estágios de aptidões e competências

Exercerão atividades para desenvolver competências e aptidões específicas em cada área temática. Práticas e dinâmicas para adquirir e desenvolver as competências e capacidades que um especialista deve desenvolver no quadro da globalização em que vivemos.

Leituras complementares

Artigos recentes, documentos de consenso e guias internacionais, entre outros. Na biblioteca virtual da TECH, o aluno terá acesso a tudo o que precisa para completar a sua capacitação.



Case studies

Completarão uma seleção dos melhores estudos de caso escolhidos especificamente para esta licenciatura. Casos apresentados, analisados e instruídos pelos melhores especialistas do panorama internacional.

Resumos interativos

A equipa da TECH apresenta os conteúdos de forma atrativa e dinâmica em conteúdos multimédia que incluem áudios, vídeos, imagens, diagramas e mapas conceituais, a fim de reforçar o conhecimento.

Este sistema educativo único para a apresentação de conteúdos multimédia foi premiado pela Microsoft como um "Caso de Sucesso Europeu".

Testing & Retesting

Os conhecimentos do aluno são periodicamente avaliados e reavaliados ao longo do curso, por meio de atividades e exercícios de avaliação e autoavaliação, para que o aluno controle o cumprimento dos seus objetivos.

07

Certificação

O Mestrado Próprio em Computação e Linguagens garante, para além do conteúdo mais rigoroso e atual, o acesso a um grau de Mestre emitido pela TECH Universidade Tecnológica.

Conclua este plano de estudos com sucesso e receba o seu certificado sem sair de casa e sem burocracias"

Este **Mestrado Próprio em Computação e Linguagens** contém o conteúdo mais educativo completo e atual do mercado.

Uma vez aprovadas as avaliações, o aluno receberá por correio* com aviso de receção, o certificado correspondente ao título de **Mestrado Próprio** emitido pela **TECH Universidade Tecnológica**.

O certificado emitido pela **TECH Universidade Tecnológica** expressará a qualificação obtida no Mestrado Próprio, atendendo aos requisitos normalmente exigidos pelas bolsas de emprego, concursos públicos e avaliação de carreiras profissionais.

Certificação: **Mestrado Próprio em Computação e Linguagens**

ECTS: **60**

Carga horária: **1500 horas**


tech universidade tecnológica

Concede o presente

CERTIFICADO

a

Sr./Sra. _____ com o documento de identificação nº_____

Por ter concluído e acreditado com sucesso o

MESTRADO PRÓPRIO

em

Computação e Linguagens

Este é um certificado atribuído por esta Universidade, reconhecido por 60 ECTS e equivalente a 1500 horas, com data de início a dd/mm/aaaaa e data de conclusão a dd/mm/aaaaa.

A TECH é uma Instituição Privada de Ensino Superior reconhecida pelo Ministério da Educação Pública a partir de 28 de junho de 2018.

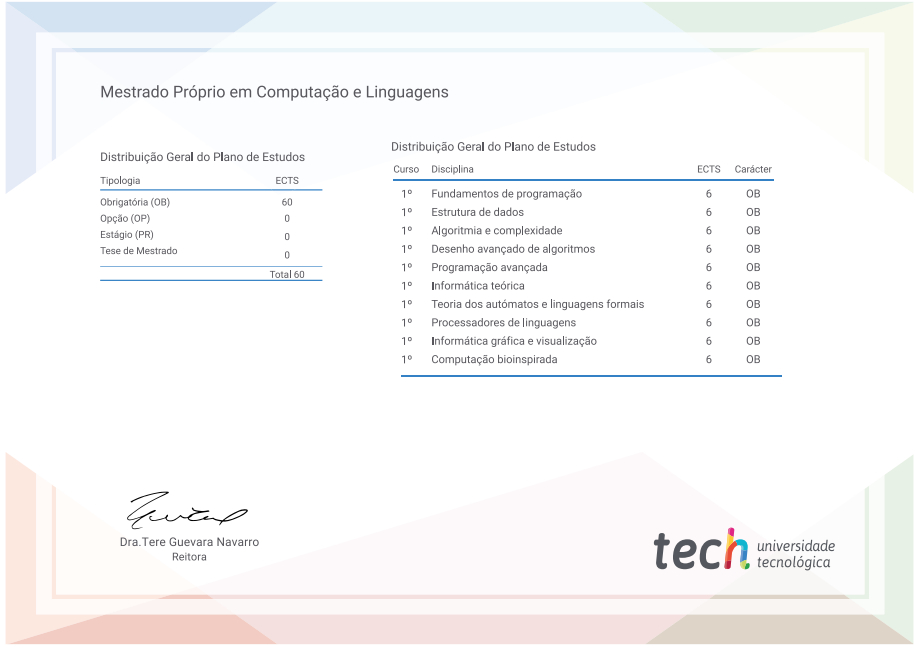
A 17 de junho de 2020

Prof.Tere Guevara Navarro
Reitora

Universidade Oficial Online da NBA

Esta qualificação deve ser sempre acompanhada por um certificado universitário emitido pela autoridade competente para a prática profissional em cada país.

código único TECH: AFWOR23S techtitute.com/titulos


Mestrado Próprio em Computação e Linguagens

Distribuição Geral do Plano de Estudos

Tipologia	ECTS
Obrigatória (OB)	60
Opção (OP)	0
Estágio (PR)	0
Tese de Mestrado	0
	Total 60

Distribuição Geral do Plano de Estudos

Curso	Disciplina	ECTS	Carácter
1º	Fundamentos de programação	6	OB
1º	Estrutura de dados	6	OB
1º	Algoritmia e complexidade	6	OB
1º	Desenho avançado de algoritmos	6	OB
1º	Programação avançada	6	OB
1º	Informática teórica	6	OB
1º	Teoria dos autómatos e linguagens formais	6	OB
1º	Processadores de linguagens	6	OB
1º	Informática gráfica e visualização	6	OB
1º	Computação bioinspirada	6	OB

Dra.Tere Guevara Navarro
Reitora

tech universidade tecnológica

*Apostila de Haia Caso o aluno solicite que o seu certificado seja apostilado, a TECH EDUCATION providenciará a obtenção do mesmo com um custo adicional.

tech universidade tecnológica

Mestrado Próprio
Computação e Linguagens

- Modalidade: **online**
- Duração: **12 meses**
- Certificação: **TECH Universidade Tecnológica**
- Créditos: **60 ECTS**
- Tempo dedicado: **16 horas/semana**
- Horário: **Ao seu próprio ritmo**
- Exames: **online**

Mestrado Próprio
Computação e Linguagens

tech universidade tecnológica