

Master's Degree

Computing and Programming Languages



Master's Degree Computing and Programming Languages

- » Modality: online
- » Duration: 12 months
- » Certificate: TECH Euromed University
- » Schedule: at your own pace
- » Exams: online

Website: www.techtitute.com/us/information-technology/master-degree/master-computing-programming-languages

Index

01

Introduction

p. 4

02

Objectives

p. 8

03

Skills

p. 14

04

Course Management

p. 18

05

Structure and Content

p. 22

06

Study Methodology

p. 34

07

Certificate

p. 44

01

Introduction

The IT professional needs to keep their skills fully up to date to be able to continue working in their field of competence in the most effective way, without missing out on any of the advances that are being incorporated into this field at a dizzying rate. This update is designed to provide students with complete and in-depth knowledge of the essential knowledge and the most interesting innovations in the design of algorithms for the development of computing projects, using the most innovative and efficient methods in the sector.





“

Acquire the fundamental know-how on Computing and Programming Languages and how to apply it successfully in the development of IT projects, in a highly competitive Master's Degree"

This Master's Degree focuses on the fundamentals of programming and data structure, algorithms and complexity, as well as advanced algorithm design, advanced programming, language processors and computer graphics, among other topics related to this area of computer science.

This Master's Degree provides students with specific tools and skills to successfully develop their professional career in the broad environment of Computing and Programming Languages. It works on key competencies such as knowledge of the day-to-day reality and work in different IT areas and develops responsibility for the monitoring and supervision of work, as well as specific skills within this field.

Additionally, as it is a 100% online Master's Degree students are not constrained by fixed timetables or the need to commute to another physical location, rather, they can access the contents at any time of the day, balancing their professional or personal life with their studies.

The teaching team of this Master's Degree in Computing and Programming Languages has carefully selected each of the topics of this course to offer the student a study opportunity as complete as possible and always linked to current events.

This **Master's Degree in Computing and Programming Languages** contains the most complete and up-to-date program on the market. The most important features include:

- ◆ Case studies presented by experts in Computing and Language
- ◆ The graphic, schematic, and practical contents which they contain, provide scientific and practical information on the disciplines that are essential for professional practice
- ◆ Practical exercises where a self-assessment process can be undertaken to improve learning
- ◆ Special emphasis on innovative methodologies in Computing and Language
- ◆ Theoretical lessons, questions to the expert, debate forums on controversial topics, and individual reflection assignments
- ◆ Content that is accessible from any fixed or portable device with an internet connection



An exceptional opportunity to learn, in a comfortable and straightforward way, the mathematical knowledge and processes necessary to carry out excellent computer programming"

“

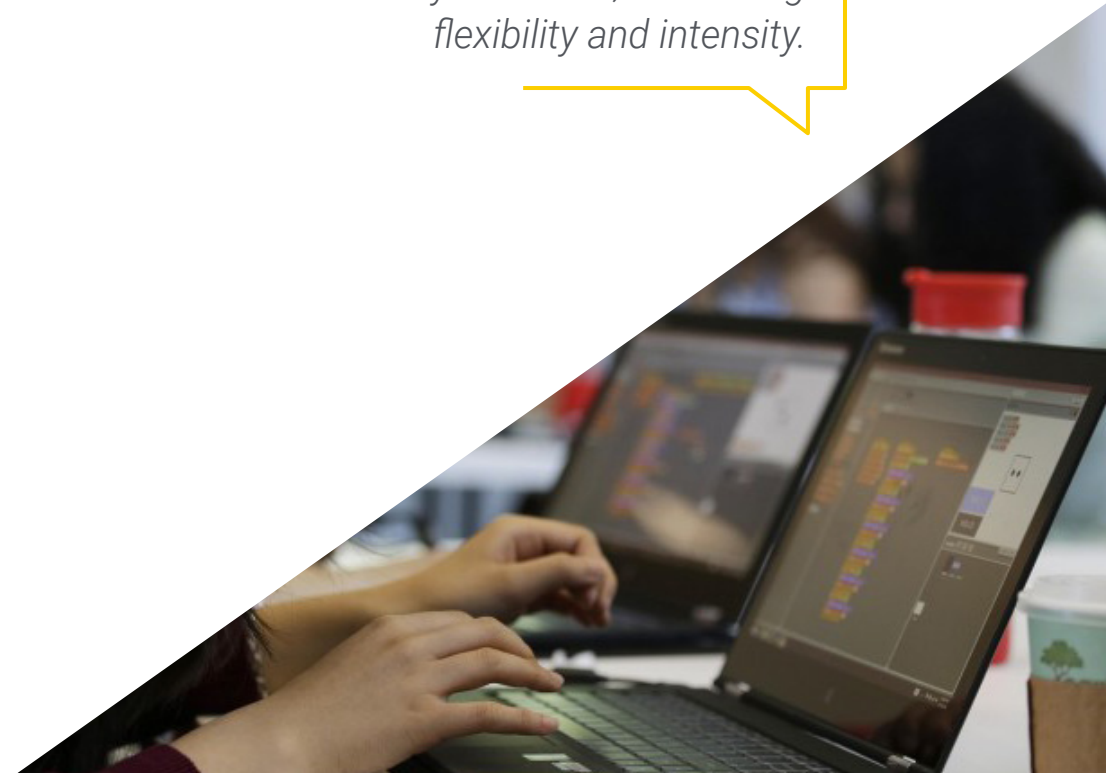
A Master's Degree that owes its effectiveness to the most valued educational technology in the market, with audiovisual and study systems that will allow you to learn in a faster and more comfortable way"

Its multimedia content, developed with the latest educational technology, will allow the professional a situated and contextual academic experience, that is to say, a simulated environment that will provide an immersive refresher designed to prepare the student for real situations.

This program is designed around Problem-Based Learning, whereby the professional must try to resolve the different professional practice situations that arise during the academic year. For this, the professional will have the help of an innovative, interactive video system made by recognized and experienced experts in Computing and Programming Languages.

We place at your disposal extensive and clear didactic materials, which incorporate all the current topics of interest for the professional who wants to advance in Computing and Programming Languages.

A program of high educational impact that will allow you to tailor the workload to your needs, combining flexibility and intensity.



02 Objectives

The Master's Degree in Computing and Programming Languages has been created specifically for the professional who seeks to advance in this area quickly and effectively, and is organized on the basis of realistic and high-value objectives that will propel him or her to another level of performance in this field.



“

Our goal is to provide professionals in the IT field with a high-quality refresher that will allow them to operate with confidence in Computing and Programming Languages”



General Objective

- ◆ Provide scientific and technological development, as well as preparation for professional practice in Computing and Programming Languages, all with a cross-disciplinary and versatile course in line with new technologies and innovations in this field



Make the most of this opportunity and take the next step to get up to date on the latest developments in Computing and Language”



Specific Objectives

Module 1. Programming Fundamentals

- ◆ Understand the basic structure of a computer, software and general-purpose programming languages
- ◆ Learn to design and interpret algorithms, which are the necessary basis for software development
- ◆ Understand the essential elements of a computer program, such as the different data types, operators, expressions, statements, I/O and control statements
- ◆ Understand the different data structures available in general purpose programming languages, both static and dynamic, as well as acquiring essential knowledge for file handling
- ◆ Know the different software testing techniques and the importance of proper documentation together with good source code
- ◆ Learn the basics of the C++ programming language, one of the most widely used programming languages in the world

Module 2. Data Structure

- ◆ Learn the basics of programming in the C++ language, including classes, variables, conditional expressions, and objects
- ◆ Understand abstract data types, linear data structure types, simple and complex hierarchical data structures, and their implementation in C++
- ◆ Understand the operation of advanced data structures that differ from the norm
- ◆ Know about the theory and practice related to the use of priority heaps and queues
- ◆ Learn how hash tables work as abstract data types and functions
- ◆ Understand graph theory, as well as algorithms and advanced graph concepts

Module 3. Algorithm and Complexity

- ◆ Learn the main strategies for algorithm design, as well as the different methods and measures for algorithm computation
- ◆ Know the main sorting algorithms used in software development
- ◆ Understand the operation of different algorithms with trees, Heaps and Graphs
- ◆ Understand the operation of Greedy algorithms, their objective and examples of their use for common problems
- ◆ Also learn about the use of Greedy algorithms on graphs
- ◆ Learn the main strategic concepts of minimum path finding, with an approach to common problems in the field and algorithms for their resolution
- ◆ Understand the Backtracking technique and its main uses, as well as other alternative techniques

Module 4. Advanced Algorithm Design

- ◆ Deepen your knowledge of advanced algorithm design, analyzing recursive and divide and conquer algorithms, as well as performing amortized analysis
- ◆ Understand dynamic programming concepts and algorithms for NP problems
- ◆ Understand the operation of combinatorial optimization, as well as the different randomization algorithms and parallel algorithms
- ◆ Know and understand the operation of the different local search methods with candidate solutions
- ◆ Learn the mechanisms of formal program verification and iterative program verification, including first-order logic and Hoare's formal system
- ◆ Learn about the operation of some of the main numerical methods such as the bisection method, the Newton Raphson method and the secant method

Module 5. Advanced Programming

- ◆ Build knowledge of programming, especially as it relates to object-oriented programming, and the different types of relationships between existing classes
- ◆ Understand the different design patterns for object-oriented problems
- ◆ Learning about event-driven programming and user interface development with Qt
- ◆ Acquire essential knowledge of concurrent programming, processes and threads
- ◆ Learn how to manage the use of threads and synchronization, as well as the resolution of common problems in concurrent programming
- ◆ Understand the importance of documentation and testing in software development

Module 6. Theoretical Computing

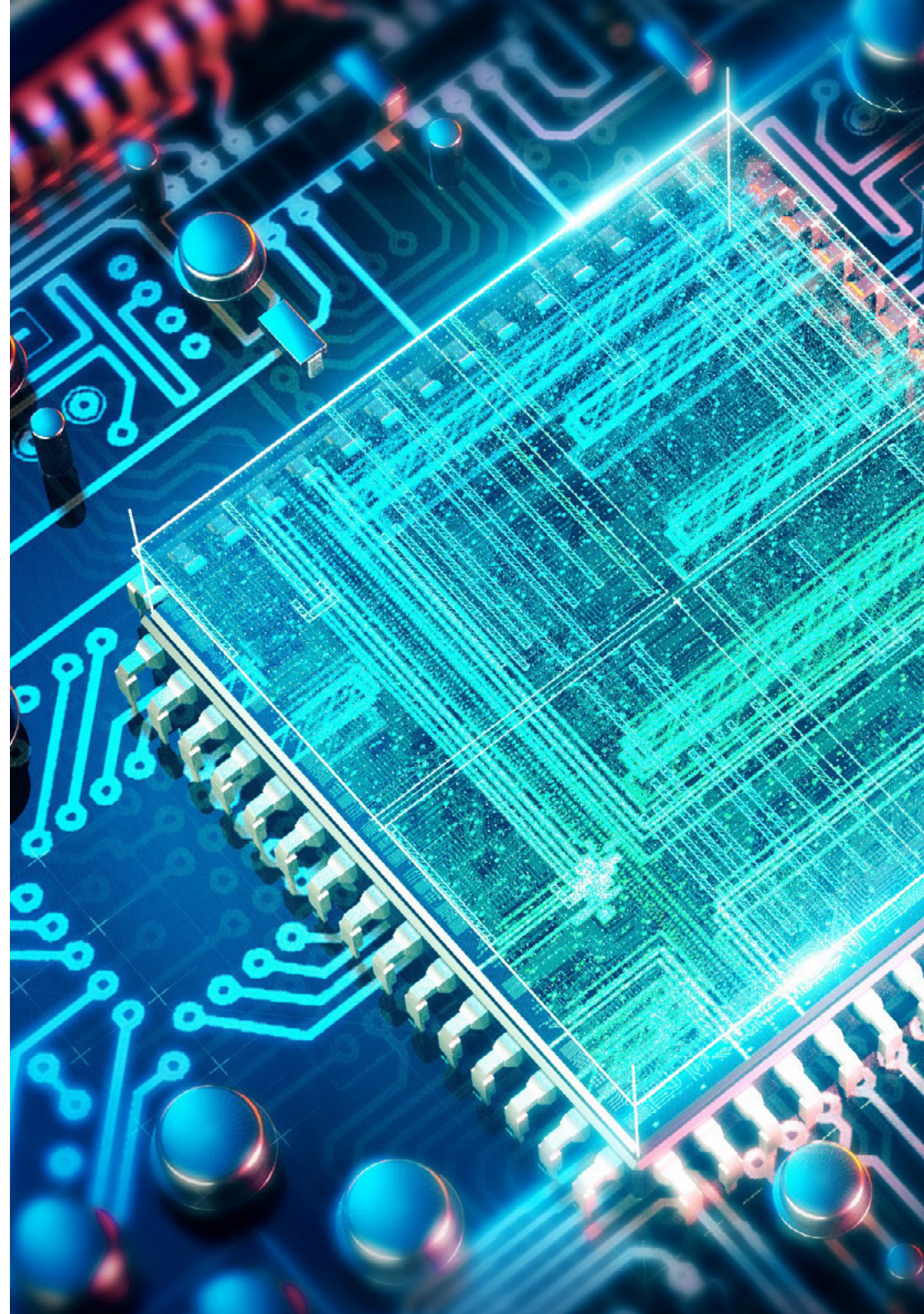
- ◆ Understand the essential theoretical mathematical concepts behind computer science, such as propositional logic, set theory, and numerable and non-numerable sets
- ◆ Understand the concepts of formal languages and grammars, as well as Turing machines in their different versions
- ◆ Learn about the different types of intractable problems, including the different versions of these problems and their approaches
- ◆ Understand the operation of different kinds of randomization-based languages and other classes and grammars
- ◆ Learn about other advanced computing systems such as Membrane Computing, DNA Computing and Quantum Computing

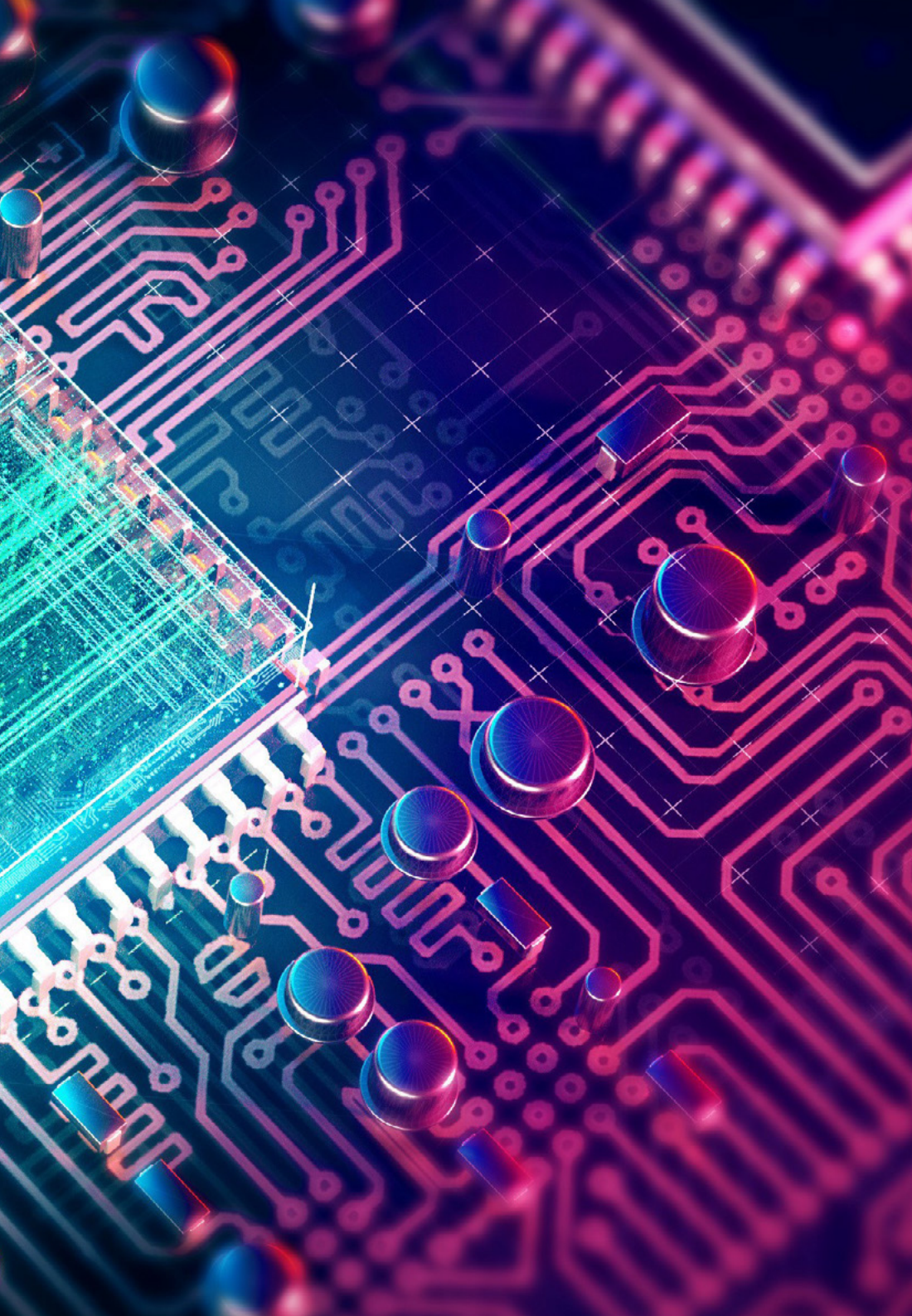
Module 7. Automata Theory and Formal Languages

- ◆ Understand the theory of automata and formal languages, learning the concepts of alphabets, strings and languages, as well as how to perform formal demonstrations
- ◆ Delve into the different types of finite automata, whether deterministic or non-deterministic
- ◆ Learn the basic and advanced concepts related to regular languages and regular expressions, as well as the application of pumping lemma and the clauses of regular languages
- ◆ Understand context-independent grammars and the operation of stack automata
- ◆ Gain knowledge on normal forms, the pumping lemma of context-independent grammars and properties of context-independent languages

Module 8. Language Processors

- ◆ Introduce the concepts related to the compilation process and the different types of analysis: lexical, syntactic and semantic
- ◆ Know how a lexical analyzer works, its implementation and error recovery
- ◆ Deepen knowledge of syntactic analysis, both top-down and bottom-up, but with special emphasis on the different types of bottom-up syntactic parsers
- ◆ Understand the operation of semantic parsers, syntax-driven tradition, the symbol table and the various types
- ◆ Learn the different mechanisms for code generation, both in execution environments and for intermediate code generation
- ◆ Lay the groundwork for code optimization, including expression reordering and loop optimization





Module 9. Computer Graphics and Visualization

- ◆ Introduce the essential concepts of computer graphics and computer visualization, such as color theory and its models and the properties of light
- ◆ Understand the operation of the output primitives and their algorithms, both for line drawing and for drawing circles and fills
- ◆ Conduct an in-depth study of the different transformations, both 2D and 3D, and their coordinate systems and computer visualization
- ◆ Learn how to make 3D projections and cuts, as well as how to remove hidden surfaces
- ◆ Learn the theory related to interpolation and parametric curves, as well as Bézier Curves and B-splines

Module 10. Bio-inspired Computing

- ◆ Introduce the concept of bio-inspired computing, as well as understanding how different types of social adaptation algorithms and genetic algorithms work
- ◆ Conduct an in-depth study of the different models of evolutionary computing, including its strategies, programming, algorithms and models based on distribution estimation
- ◆ Understand the main space exploration-exploitation strategies for genetic algorithms
- ◆ Understand the operation of evolutionary programming applied to learning problems and multi-objective problems
- ◆ Learn the essential concepts related to neural networks and understand the operation of real-life cases applied to areas as diverse as medical research, economics and computer vision

03 Skills

After passing the assessments of the Master's Degree in Computing and Programming Languages, the professional will have acquired the necessary skills and know the fundamental principles of computing with the ability to work with programming languages and data.



“

Gain the ability to carry out new computer developments based on the understanding and control of the different languages and algorithms and their practical application”



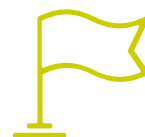
General Skills

- ◆ Successfully perform the tasks related to Computing and Programming Languages

“

Improve your skills to work on various technological projects"





Specific Skills

- ◆ Design algorithms to develop computer programs and apply programming language
- ◆ Understand and use IT data structure
- ◆ Use the necessary algorithms to solve computer problems
- ◆ Gain in-depth knowledge of advanced algorithm design and search methods
- ◆ Carry out computer programming tasks
- ◆ Understand and apply the theory behind computer science, such as mathematics
- ◆ Know automata theory and how to apply computing language
- ◆ Know the theoretical foundations of programming languages and the associated lexical, syntactic and semantic processing techniques
- ◆ Understand the basic concepts of mathematics and computational complexity in order to apply them to the resolution of IT problems
- ◆ Know and apply the fundamental principles of computer science to carry out new computer developments

04

Course Management

This academic program includes the most specialized teaching staff in the current educational market. They are specialists selected by TECH Euromed to develop the whole syllabus. In this way, starting from their own existence and the latest evidence, they have designed the most up-to-date content that provides a guarantee of quality in such a relevant subject.



“

TECH Euromed offers the most specialized teaching staff in the field of study. Enroll now and enjoy the quality you deserve”

International Guest Director

Dr. Jeremy Gibbons is considered an international eminence for his contributions in the field of Programming Methodology and its applications in Software Engineering. For more than two decades, this expert, associated with the Department of Computer Science at the University of Oxford, has driven different development projects whose most tangible results are applied by computer scientists from different parts of the world.

His work covers areas such as generic programming, formal methods, computational biology, bioinformatics and algorithm design with Haskell. This last topic he developed extensively in conjunction with his mentor, Dr. Richard Bird.

In his role as Director of the Algebra of Programming Research Group, Gibbons has led advances in Functional Programming Languages and Pattern Theory in Programming. At the same time, the applications of his innovations have been linked to the healthcare framework, as evidenced by his collaboration with CancerGrid and Datatype-Generic Programming. These and other initiatives reflect his interest in solving practical problems in cancer research and clinical informatics.

Gibbons has also made a significant mark as Editor-in-Chief of scholarly publications in The Journal of Functional Programming and The Programming Journal: The Art, Science, and Engineering of Programming. Through these responsibilities he has carried out intensive outreach and dissemination of knowledge. In addition, he has led several study chairs linked to renowned institutions such as Oxford Brookes University and the University of Auckland, New Zealand.

In addition, this specialist is a member of the Working Group 2.1 on Algorithmic Languages and Computation of the International Federation for Information Processing (IFIP). With this organization, he provides maintenance for the ALGOL 60 and ALGOL 68 programming languages.



Dr. Gibbons, Jeremy

- Director, Software Engineering Program, University of Oxford, UK
- Deputy Head of the Informatics Laboratory and Department of Computer Science, University of Oxford, UK
- Professor at Kellogg College, Oxford Brookes University and the University of Auckland, New Zealand
- Director of the Algebra of Programming Research Group
- Editor-in-Chief of The Art, Science, and Engineering of Programming and the Journal of Functional Programming
- Doctorate in Computer Science from Oxford University
- B.Sc. in Computer Science from the University of Edinburgh
- Member of: Working Group 2.1 on Algorithmic Languages and Computation of the International Federation for Information Processing (IFIP)



*Thanks to TECH Euromed,
you will be able to learn
with the best professionals
in the world"*

05

Structure and Content

The structure of the contents has been created in such a way that the knowledge is assimilated in a progressive way, achieving a growth trajectory that will lead you to excellence in your profession.



“

All the areas of interest that you need to master in order to work confidently and successfully in Computing and Programming Languages, gathered together in a top-quality syllabus"

Module 1. Programming Fundamentals

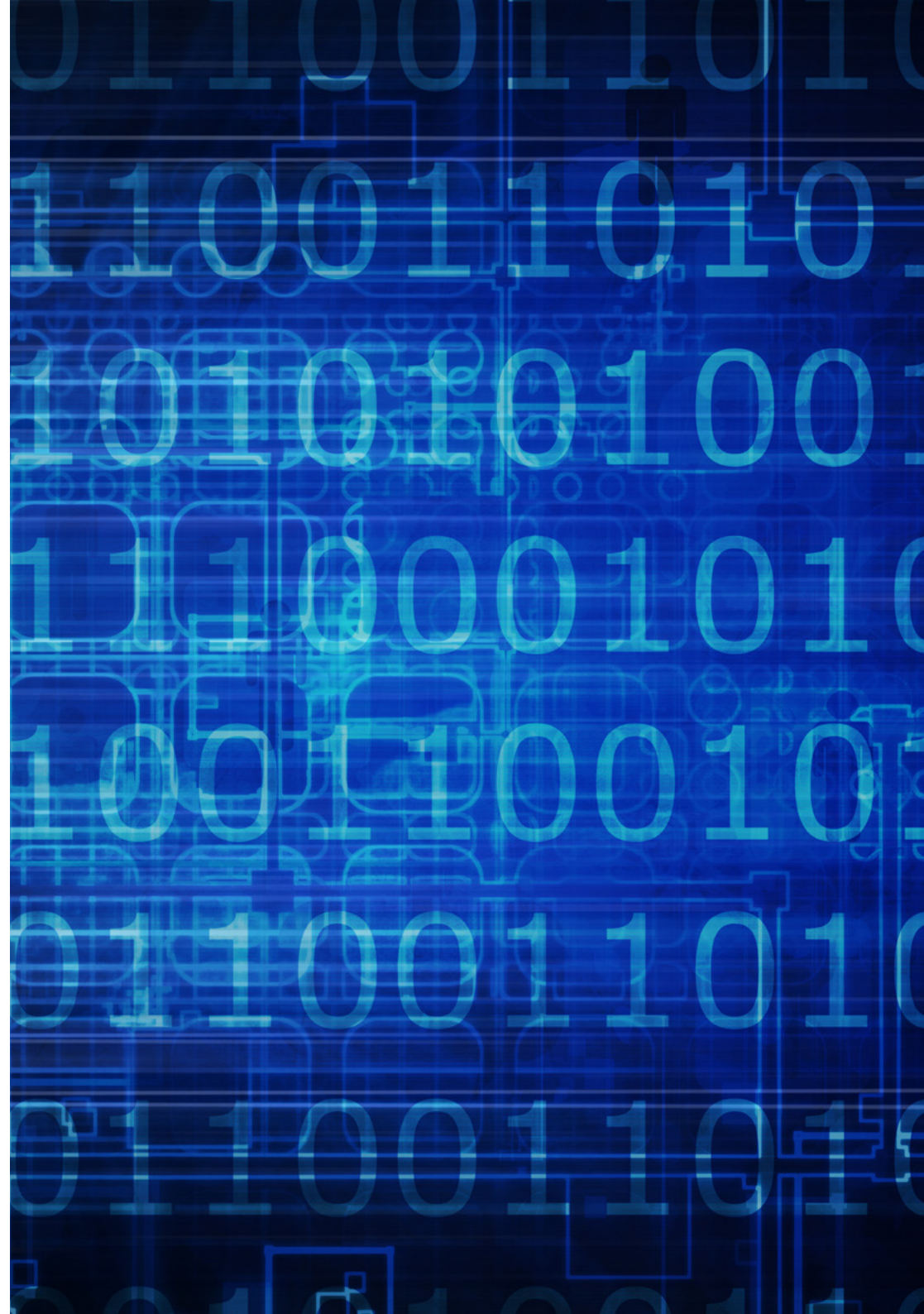
- 1.1. Introduction to Programming
 - 1.1.1. Basic Computer Structure
 - 1.1.2. Software
 - 1.1.3. Programming Languages
 - 1.1.4. Computer Application Life Cycle
- 1.2. Algorithm Design
 - 1.2.1. Problem Solving
 - 1.2.2. Descriptive Techniques
 - 1.2.3. Algorithm Elements and Structure
- 1.3. Program Elements
 - 1.3.1. C++ Origin and Features
 - 1.3.2. Development Environment
 - 1.3.3. Concept of Program
 - 1.3.4. Types of Fundamental Data
 - 1.3.5. Operators
 - 1.3.6. Expressions
 - 1.3.7. Statements
 - 1.3.8. Data Input and Output
- 1.4. Control Statements
 - 1.4.1. Statements
 - 1.4.2. Branches
 - 1.4.3. Loops
- 1.5. Abstraction and Modularity: Functions
 - 1.5.1. Modular Design
 - 1.5.2. Concept of Function and Utility
 - 1.5.3. Definition of a Function
 - 1.5.4. Execution Flow in the Call of a Function
 - 1.5.5. Function Prototypes
 - 1.5.6. Results Return
 - 1.5.7. Calling Functions: Parameters
 - 1.5.8. Parameter Passing According to Reference and Value
 - 1.5.9. Scope Identifier
- 1.6. Statistical Data Structures
 - 1.6.1. Arrays
 - 1.6.2. Matrices Polyhedra
 - 1.6.3. Searching and Sorting
 - 1.6.4. Chaining: I/O Functions for Chains
 - 1.6.5. Structures: Unions
 - 1.6.6. New Types of Data
- 1.7. Dynamic Data Structures: Pointers
 - 1.7.1. Concept. Definition of Pointer
 - 1.7.2. Pointer Operators and Operations
 - 1.7.3. Pointer Arrays
 - 1.7.4. Pointers and Arrays
 - 1.7.5. Chain Pointers
 - 1.7.6. Structure Pointers
 - 1.7.7. Multiple Indirection
 - 1.7.8. Function Pointers
 - 1.7.9. Function, Structure and Array Passing as Function Parameters
- 1.8. Files
 - 1.8.1. Basic Concepts
 - 1.8.2. File Operations
 - 1.8.3. Types of Files
 - 1.8.4. File Organization
 - 1.8.5. Introduction to C++ Files
 - 1.8.6. Managing Files
- 1.9. Recursion
 - 1.9.1. Definition of Recursion
 - 1.9.2. Types of Recursion
 - 1.9.3. Advantages and Disadvantages
 - 1.9.4. Considerations
 - 1.9.5. Recursive-Iterative Conversion
 - 1.9.6. Recursion Stack

- 1.10. Testing and Documentation
 - 1.10.1. Program Testing
 - 1.10.2. White Box Testing
 - 1.10.3. Black Box Testing
 - 1.10.4. Testing Tools
 - 1.10.5. Program Documentation

Module 2. Data Structure

- 2.1. Introduction to C++ Programming
 - 2.1.1. Classes, Constructors, Methods and Attributes
 - 2.1.2. Variables:
 - 2.1.3. Conditional Expressions and Loops
 - 2.1.4. Objects
- 2.2. Abstract Data Types (ADT)
 - 2.2.1. Types of Data
 - 2.2.2. Basic Structures and ADTs
 - 2.2.3. Vectors and Arrays
- 2.3. Lineal Data Structures
 - 2.3.1. ADT List Definition
 - 2.3.2. Linked and Doubly Linked Lists
 - 2.3.3. Ordered Lists
 - 2.3.4. Lists in C++
 - 2.3.5. ADT Stack
 - 2.3.6. ADT Queue
 - 2.3.7. Stack and Queue in C++
- 2.4. Hierarchical Data Structures
 - 2.4.1. ADT Tree
 - 2.4.2. Routes
 - 2.4.3. N-ary Trees
 - 2.4.4. Binary Trees
 - 2.4.5. Binary Search Trees

- 2.5. Hierarchical Data Structures: Complex Trees
 - 2.5.1. Perfectly Balanced Trees or Minimum Height Trees
 - 2.5.2. Multi-path Trees
 - 2.5.3. Bibliographical References
- 2.6. Priority Heaps and Queues
 - 2.6.1. ADT Heaps
 - 2.6.2. ADT Priority Queues
- 2.7. Hash Tables
 - 2.7.1. ADT Hash Table
 - 2.7.2. Hash Functions
 - 2.7.3. Hash Function in Hash Tables
 - 2.7.4. Redistribution
 - 2.7.5. Open Hash Tables
- 2.8. Graphs
 - 2.8.1. ADT Graph
 - 2.8.2. Types of Graph
 - 2.8.3. Graphical Representation and Basic Operations
 - 2.8.4. Graph Design
- 2.9. Algorithms and Advanced Concepts on Graphs
 - 2.9.1. Problems with Graphs
 - 2.9.2. Path Algorithms
 - 2.9.3. Search or Path Algorithms
 - 2.9.4. Other Algorithms
- 2.10. Other Data Structures
 - 2.10.1. Sets
 - 2.10.2. Parallel Arrays
 - 2.10.3. Table of Symbols
 - 2.10.4. Tries



Module 3. Algorithm and Complexity

- 3.1. Introduction to Algorithm Design Strategies
 - 3.1.1. Recursion
 - 3.1.2. Divide and Conquer
 - 3.1.3. Other Strategies
- 3.2. Algorithm Efficiency and Analysis
 - 3.2.1. Efficiency Measures
 - 3.2.2. Measuring Entry Size
 - 3.2.3. Measuring Execution Time
 - 3.2.4. Worst, Best and Average Case
 - 3.2.5. Asymptotic Notation
 - 3.2.6. Mathematical Analysis Criteria for Non-Recursive Algorithms
 - 3.2.7. Mathematical Analysis for Recursive Algorithms
 - 3.2.8. Empirical Analysis for Algorithms
- 3.3. Sorting Algorithms
 - 3.3.1. Concept of Sorting
 - 3.3.2. Bubble Sorting
 - 3.3.3. Selection Sorting
 - 3.3.4. Insertion Sorting
 - 3.3.5. Merge Sorting
 - 3.3.6. Quick Sorting
- 3.4. Tree Algorithms
 - 3.4.1. Concept of Tree
 - 3.4.2. Binary Trees
 - 3.4.3. Tree Paths
 - 3.4.4. Representing Expressions
 - 3.4.5. Sorted Binary Trees
 - 3.4.6. Balanced Binary Trees
- 3.5. Algorithms Using Heaps
 - 3.5.1. Heaps
 - 3.5.2. The HeapSort Algorithm
 - 3.5.3. Priority Queues
- 3.6. Graph Algorithms
 - 3.6.1. Representation
 - 3.6.2. Width Traversal
 - 3.6.3. Depth Traversal
 - 3.6.4. Topological Sorting
- 3.7. Greedy Algorithms
 - 3.7.1. Greedy Strategy
 - 3.7.2. Greedy Strategy Elements
 - 3.7.3. Currency Exchange
 - 3.7.4. Traveling Salesman Problem
 - 3.7.5. Knapsack Problem
- 3.8. Minimal Pathways Search
 - 3.8.1. Shortest Path Problem
 - 3.8.2. Cycles and Negative Arcs
 - 3.8.3. Dijkstra's Algorithm
- 3.9. Greedy Algorithms for Graphs
 - 3.9.1. Minimum Spanning Tree
 - 3.9.2. Prim's Algorithm
 - 3.9.3. Kruskal's Algorithm
 - 3.9.4. Complexity Analysis
- 3.10. Backtracking
 - 3.10.1. Backtracking
 - 3.10.2. Alternative Techniques

Module 4. Advanced Algorithm Design

- 4.1. Analysis of Recursive and Divide and Conquer Algorithms
 - 4.1.1. Posing and Solving Homogeneous and Non-Homogeneous Recurrence Equations
 - 4.1.2. Divide and Conquer Strategy Overview
- 4.2. Amortized Analysis
 - 4.2.1. Aggregated Analysis
 - 4.2.2. The Accounting Method
 - 4.2.3. The Potential Method
- 4.3. Dynamic Programming and Algorithms for NP Problems
 - 4.3.1. Dynamic Programming Features
 - 4.3.2. Backtracking
 - 4.3.3. Branching and Pruning
- 4.4. Combinatorial Optimization
 - 4.4.1. Representation of Problems
 - 4.4.2. Optimization in 1D
- 4.5. Randomization Algorithms
 - 4.5.1. Examples of Randomization Algorithms
 - 4.5.2. The Buffon Theorem
 - 4.5.3. The Monte Carlo Algorithm
 - 4.5.4. The Las Vegas Algorithm
- 4.6. Local Search and with Candidates
 - 4.6.1. Gradient Ascent
 - 4.6.2. Hill Climbing
 - 4.6.3. Simulated Annealing
 - 4.6.4. Tabu Search
 - 4.6.5. Search with Candidates
- 4.7. Formal Program Verification
 - 4.7.1. Specification of Functional Abstractions
 - 4.7.2. The Language of First Order Logic
 - 4.7.3. Hoare's Formal System

- 4.8. Iterative Program Verification
 - 4.8.1. Rules of Hoare's Formal System
 - 4.8.2. Concept of Invariant Iterations
- 4.9. Numeric Methods
 - 4.9.1. The Bisection Method
 - 4.9.2. The Newton Raphson Method
 - 4.9.3. The Secant Method
- 4.10. Parallel Algorithms
 - 4.10.1. Parallel Binary Operations
 - 4.10.2. Parallel Operations with Graphs
 - 4.10.3. Parallelism in Divide and Conquer
 - 4.10.4. Parallelism in Dynamic Programming

Module 5. Advanced Programming

- 5.1. Introduction to Object Oriented Programming
 - 5.1.1. Introduction to Object Oriented Programming
 - 5.1.2. Class Design
 - 5.1.3. Introduction to Unified Modeling Language (UML) for Problem Modeling
- 5.2. Class Relations
 - 5.2.1. Abstractions and Heritage
 - 5.2.2. Advanced Concepts of Heritage
 - 5.2.3. Polymorphism
 - 5.2.4. Composition and Aggregation
- 5.3. Introduction to Design Patterns for Object Oriented problems
 - 5.3.1. What Are Design Patterns?
 - 5.3.2. Factory Pattern
 - 5.3.3. Singleton Pattern
 - 5.3.4. Observer Pattern
 - 5.3.5. Composite Pattern
- 5.4. Exceptions
 - 5.4.1. What Are the Exceptions?
 - 5.4.2. Catching and Handling Exceptions
 - 5.4.3. Launching Exceptions
 - 5.4.4. Creation Exceptions

- 5.5. User Interface
 - 5.5.1. Introduction to Qt
 - 5.5.2. Positioning
 - 5.5.3. What Are Events?
 - 5.5.4. Events: Definition and Capture
 - 5.5.5. User Interface Development
- 5.6. Introduction to Concurrent Programming
 - 5.6.1. Introduction to Concurrent Programming
 - 5.6.2. Concept of Process and Thread
 - 5.6.3. Process and Thread Interaction
 - 5.6.4. C++ Threads
 - 5.6.5. Advantages and Disadvantages of Concurrent Programming
- 5.7. Thread Management and Synchronization
 - 5.7.1. Thread Life Cycle
 - 5.7.2. Thread Class
 - 5.7.3. Thread Planning
 - 5.7.4. Thread Groups
 - 5.7.5. Daemon Threads
 - 5.7.6. Synchronization
 - 5.7.7. Locking Mechanisms
 - 5.7.8. Communication Mechanisms
 - 5.7.9. Monitors
- 5.8. Common Problems in Concurrent Programming
 - 5.8.1. Producer-Consumer Problem
 - 5.8.2. Readers-Writers Problem
 - 5.8.3. Dining Philosophers Problem
- 5.9. Software Testing and Documentation
 - 5.9.1. Why Is It Important to Document Software?
 - 5.9.2. Design Documentation
 - 5.9.3. Documentation Tool Use

- 5.10. Software Tests
 - 5.10.1. Introduction to Software Tests
 - 5.10.2. Types of Tests
 - 5.10.3. Unit Test
 - 5.10.4. Integration Tests
 - 5.10.5. Validation Test
 - 5.10.6. System Tests

Module 6. Theoretical Computing

- 6.1. Mathematical Concepts Used
 - 6.1.1. Introduction to Propositional Logic
 - 6.1.2. Relationship Theory
 - 6.1.3. Numerable and Non-Numerable Sets
- 6.2. Formal Languages and Grammar and Introduction to Turing Machines
 - 6.2.1. Formal Languages and Grammar
 - 6.2.2. Decision Problem
 - 6.2.3. The Turing Machine
- 6.3. Extensions for Turing Machines, Constrained Turing Machines and Computers
 - 6.3.1. Programming Techniques for Turing Machines
 - 6.3.2. Programming Techniques for Turing Machines
 - 6.3.3. Restricted Turing Machines
 - 6.3.4. Turing Machines and Computers
- 6.4. Undecidable Problems
 - 6.4.1. Non-Recursively Enumerable Language
 - 6.4.2. An Undecidable Recursively Enumerable Problem
- 6.5. Other Undecidable Problems
 - 6.5.1. Programming Techniques for Turing Machines
 - 6.5.2. Post Correspondence Problem (PCP)
- 6.6. Intractable Problems
 - 6.6.1. P and NP Classes
 - 6.6.2. A Complete NP Problem
 - 6.6.3. Restricted Satisfiability Problem
 - 6.6.4. Other Complete NP Problems

- 6.7. Co-NP and PS Problems
 - 6.7.1. Complements to NP languages
 - 6.7.2. Solvable Problems in Polynomial Space
 - 6.7.3. Complete PS Problems
- 6.8. Classes of Randomization-Based Languages
 - 6.8.1. MT Model with Randomization
 - 6.8.2. RP and ZPP Classes
 - 6.8.3. Primality Test
 - 6.8.4. Complexity of the Primality Test
- 6.9. Other Classes and Grammars
 - 6.9.1. Probabilistic Finite Automata
 - 6.9.2. Cellular Automata
 - 6.9.3. McCulloch and Pitts Cells
 - 6.9.4. Lindenmayer Grammar
- 6.10. Advanced Computing Systems
 - 6.10.1. Membrane Computing: Systems
 - 6.10.2. Computing with DNA
 - 6.10.3. Quantum Computing

Module 7. Automata Theory and Formal Languages

- 7.1. Introduction to Automata Theory
 - 7.1.1. Why Study Automata Theory?
 - 7.1.2. Introduction to Formal Demonstrations
 - 7.1.3. Other Types of Demonstration
 - 7.1.4. Mathematical Induction
 - 7.1.5. Alphabets, Strings and Languages
- 7.2. Deterministic Finite Automata
 - 7.2.1. Introduction to Finite Automata
 - 7.2.2. Deterministic Finite Automata
- 7.3. Non-Deterministic Finite Automata
 - 7.3.1. Non-Deterministic Finite Automata
 - 7.3.2. Equivalency between AFD and AFN
 - 7.3.3. Finite Automata with Transitions

- 7.4. Languages and Regular Expressions (I)
 - 7.4.1. Languages and Regular Expressions
 - 7.4.2. Finite Automata and Regular Expressions
- 7.5. Languages and Regular Expressions (II)
 - 7.5.1. Conversion of Regular Expressions into Automata
 - 7.5.2. Applications of Regular Expressions
 - 7.5.3. Algebra of Regular Expressions
- 7.6. Lemma Pumping and Closure of Regular Languages
 - 7.6.1. Lemma Pumping
 - 7.6.2. Closure Properties of Regular Languages
- 7.7. Equivalence and Minimization of Automata
 - 7.7.1. FA Equivalence
 - 7.7.2. FA Minimization
- 7.8. Context Independent Grammars (CIG)
 - 7.8.1. Context Independent Grammars
 - 7.8.2. Derivation Trees
 - 7.8.3. GIC Applications
 - 7.8.4. Ambiguity in Grammars and Languages
- 7.9. Stack Automata and GIC
 - 7.9.1. Definition of Stack Automata
 - 7.9.2. Languages Accepted by a Stack Automaton
 - 7.9.3. Equivalence between Stack Automata and GICs
 - 7.9.4. Deterministic Stack Automata
- 7.10. Normal Forms, GIC Lemma Pumping and Properties of LICs
 - 7.10.1. Normal Forms of GICs
 - 7.10.2. Lemma Pumping
 - 7.10.3. Closure Properties of Languages
 - 7.10.4. Decision Properties of LICs

Module 8. Language Processors

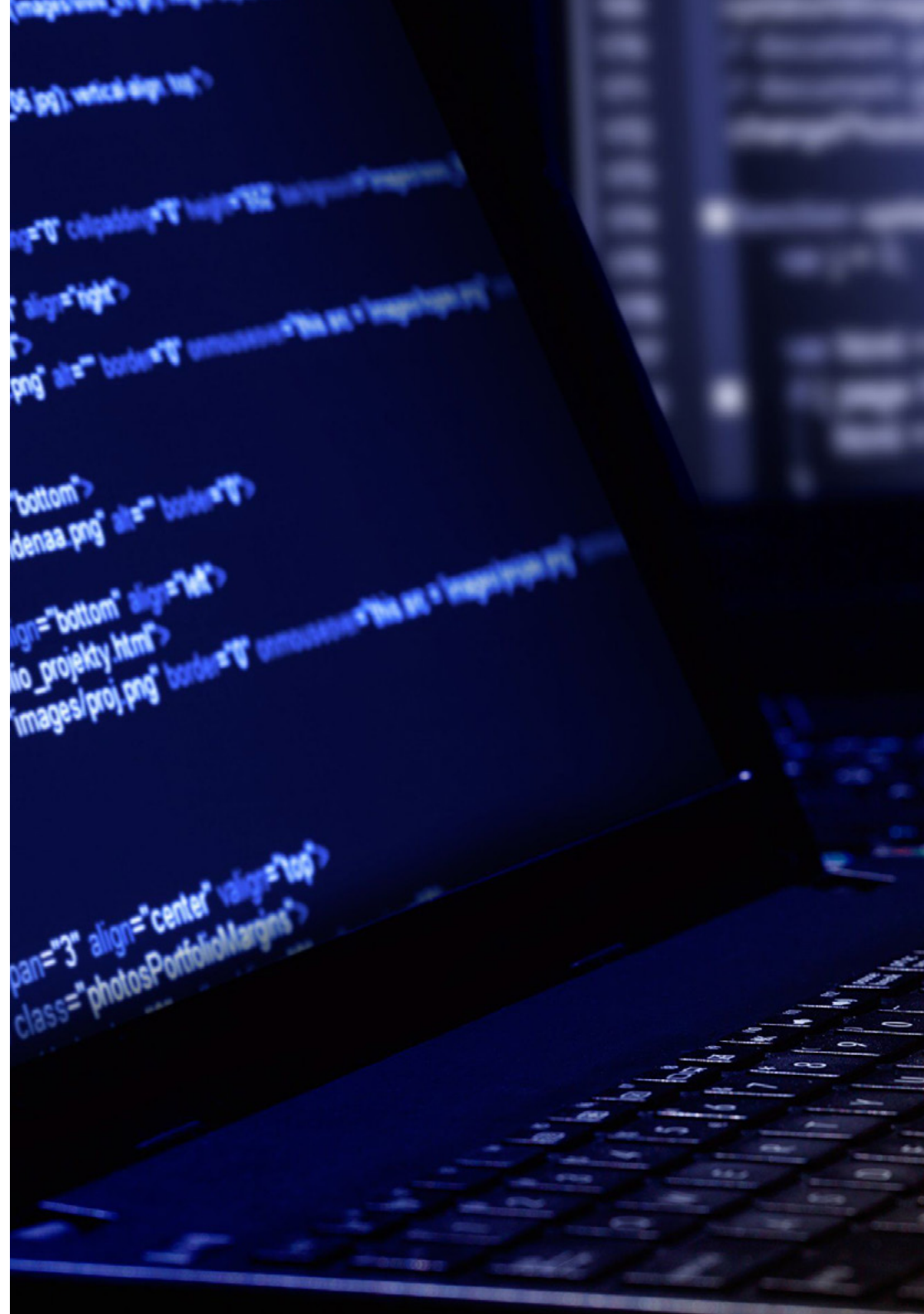
- 8.1. Introduction to the Compilation Process
 - 8.1.1. Compilation and Interpretation
 - 8.1.2. Compiler Execution Environment
 - 8.1.3. Analysis Process
 - 8.1.4. Synthesis Process
- 8.2. Lexical Analyzer
 - 8.2.1. What is a Lexical Analyzer?
 - 8.2.2. Implementation of the Lexical Analyzer
 - 8.2.3. Semantic Actions
 - 8.2.4. Error Recovery
 - 8.2.5. Implementation Issues
- 8.3. Parsing
 - 8.3.1. What is a Syntactic Analyzer?
 - 8.3.2. Previous Concepts
 - 8.3.3. Top-down Analyzers
 - 8.3.4. Bottom-up Analyzers
- 8.4. Top-down Parsing and Bottom-up Parsing
 - 8.4.1. LL Analyzer (1)
 - 8.4.2. LR Analyzer (0)
 - 8.4.3. Analyzer Example
- 8.5. Advanced Bottom-Up Parsing
 - 8.5.1. SLR Analyzers
 - 8.5.2. LR Analyzer (1)
 - 8.5.3. LR Analyzer(K)
 - 8.5.4. LALR Analyzers
- 8.6. Semantic Analysis (I)
 - 8.6.1. Syntax-Driven Translation
 - 8.6.2. Table of Symbols

- 8.7. Semantic Analysis (II)
 - 8.7.1. Type Checking
 - 8.7.2. The Type Subsystem
 - 8.7.3. Type Equivalence and Conversions
- 8.8. Code Generation and Execution Environment
 - 8.8.1. Design Aspects
 - 8.8.2. Execution Environment
 - 8.8.3. Memory Organization
 - 8.8.4. Memory Allocation
- 8.9. Intermediate Code Generation
 - 8.9.1. Synthesis-driven Translation
 - 8.9.2. Intermediate Representations
 - 8.9.3. Examples of Translations
- 8.10. Code Optimization
 - 8.10.1. Assignment of Records
 - 8.10.2. Elimination of Dead Assignments
 - 8.10.3. Compile-time Execution
 - 8.10.4. Reordering of Expressions
 - 8.10.5. Loop Optimization

Module 9. Computer Graphics and Visualization

- 9.1. Color Theory
 - 9.1.1. Properties of Light
 - 9.1.2. Color Models
 - 9.1.3. The CIE Standard
 - 9.1.4. Profiling
- 9.2. Output Primitives
 - 9.2.1. The Video Controller
 - 9.2.2. Line Drawing Algorithms
 - 9.2.3. Circumferences Drawing Algorithms
 - 9.2.4. Backfill Algorithms

- 9.3. 2D Transformations and 2D Coordinate Systems and 2D Clipping
 - 9.3.1. Basic Geometric Transformations
 - 9.3.2. Homogeneous Coordinates
 - 9.3.3. Inverse Transformation
 - 9.3.4. Transformation Composition
 - 9.3.5. Other Transformations
 - 9.3.6. Coordinate Changing
 - 9.3.7. 2D Coordinate Systems
 - 9.3.8. Coordinate Changing
 - 9.3.9. Standardization
 - 9.3.10. Clipping Algorithms
- 9.4. 3D Transformations
 - 9.4.1. Translation
 - 9.4.2. Rotation
 - 9.4.3. Scaling
 - 9.4.4. Reflection
 - 9.4.5. Shearing
- 9.5. Viewing and Changing 3D Coordinates
 - 9.5.1. 3D Coordinate Systems
 - 9.5.2. Visualisation
 - 9.5.3. Coordinate Changing
 - 9.5.4. Projection and Standardization
- 9.6. 3D Projection and Cropping
 - 9.6.1. Orthogonal Projection
 - 9.6.2. Oblique Parallel Projection
 - 9.6.3. Projection Perspective
 - 9.6.4. 3D Clipping Algorithms



- 9.7. Elimination of Hidden Surfaces
 - 9.7.1. Back Face Removal
 - 9.7.2. Z-buffer
 - 9.7.3. Painter Algorithm
 - 9.7.4. Warnock Algorithm
 - 9.7.5. Hidden Line Detection
 - 9.8. Interpolation and Parametric Curves
 - 9.8.1. Interpolation and Approximation with Polynomials
 - 9.8.2. Parametric Representation
 - 9.8.3. Lagrange's Polynomial
 - 9.8.4. Natural Cubic Splines
 - 9.8.5. Base Functions
 - 9.8.6. Matrix Representation
 - 9.9. Bézier Curves
 - 9.9.1. Algebraic Construction
 - 9.9.2. Matrix Form
 - 9.9.3. Composition
 - 9.9.4. Geometric Construction
 - 9.9.5. Drawing Algorithm
 - 9.10. B-Splines
 - 9.10.1. The Problem of Local Control
 - 9.10.2. Uniform Cubic B-Splines
 - 9.10.3. Base Functions and Control Points
 - 9.10.4. Derivation of the origin and Multiplicity
 - 9.10.5. Matrix Representation
 - 9.10.6. Non-Uniform B-Splines
- Module 10. Bio-Inspired Computing**
- 10.1. Introduction to Bio-Inspired Computing
 - 10.1.1. Introduction to Bio-Inspired Computing
 - 10.2. Social Adaptation Algorithms
 - 10.2.1. Bio-inspired Computing Based on Ant Colonies
 - 10.2.2. Variants of Ant Colony Algorithms
 - 10.2.3. Particle Cloud Computing
 - 10.3. Genetic Algorithms
 - 10.3.1. General Structure
 - 10.3.2. Implementations of the Main Operators
 - 10.4. Space Exploration-Exploitation Strategies for Genetic Algorithms
 - 10.4.1. CHC Algorithm
 - 10.4.2. Multimodal Problems
 - 10.5. Evolutionary Computing Models (I)
 - 10.5.1. Evolutionary Strategies
 - 10.5.2. Evolutionary Programming
 - 10.5.3. Algorithms Based on Differential Evolution
 - 10.6. Evolutionary Computing Models (II)
 - 10.6.1. Evolution Models based on Estimation of Distributions (EDA)
 - 10.6.2. Genetic Programming
 - 10.7. Evolutionary Programming Applied to Learning Disabilities
 - 10.7.1. Rule-Based Learning
 - 10.7.2. Evolutionary Methods in Instance Selection Problems
 - 10.8. Multi-objective Problems
 - 10.8.1. Concept of Dominance
 - 10.8.2. Application of Evolutionary Algorithms to Multi-objective Problems
 - 10.9. Neural Networks (I)
 - 10.9.1. Introduction to Neural Networks
 - 10.9.2. Case Study with Neural Networks
 - 10.10. Neural Networks (II)
 - 10.10.1. Examples of the Use of Neural Networks in Medical Research
 - 10.10.2. Examples of the Use of Neural Networks in the Economy
 - 10.10.3. Examples of the Use of Neural Networks in Artificial Vision

06

Study Methodology

TECH Euromed University is the first in the world to combine the **case study** methodology with **Relearning**, a 100% online learning system based on guided repetition.

This innovative pedagogical strategy has been conceived to offer professionals the opportunity to update their knowledge and develop their skills in an intensive and rigorous way. A learning model that places students at the center of the educational process giving them the leading role, adapting to their needs and moving away from more conventional methodologies.



“

*TECH Euromed University will
prepare you to face new challenges
in uncertain environments and
achieve success in your career”*

The student: the priority of all TECH Euromed University programs

In TECH Euromed University's study methodology, the student is the main protagonist. The teaching tools for each program have been carefully selected taking into account the demands of time, availability and academic rigor that, today, not only students demand but also the most competitive positions on the labor market.

With TECH Euromed University's asynchronous educational model, students have the flexibility to choose when and how much time they dedicate to studying, as well as how they establish their routines, and all this from the comfort of their preferred electronic device. The student will not have to participate in live classes, which in many cases they will not be able to attend. The learning activities can be completed at a time that is convenient for the student, allowing them to decide when and where they want to study.

“

At TECH Euromed University you will NOT have live classes (which you might not be able to attend)”



The most comprehensive study plans at the international

TECH Euromed University is distinguished by offering the most complete academic itineraries on the higher education landscape. This comprehensiveness is achieved through the creation of syllabi that not only cover the essential knowledge, but also the most recent innovations in each area.

By being constantly up to date, these programs enable students to keep up with market changes and acquire the skills most valued by employers. As a result, those who complete their studies at TECH Euromed University receive a well-rounded education that provides them with a significant competitive edge to advance their careers.

Moreover, they can access their studies from any device-PC, tablet, or smartphone.

“

TECH Euromed University's model is asynchronous, so it allows you to study with your pc, tablet or your smartphone wherever you want, whenever you want and for as long as you want”

Case Studies and Case Method

The case method has been the learning system most used by the world's best business schools. Developed in 1912 so that law students would not only learn the law based on theoretical content, its function was also to present them with real complex situations. In this way, they could make informed decisions and value judgments about how to resolve them. In 1924, Harvard adopted it as a standard teaching method.

With this teaching model, it is students themselves who build their professional competence through strategies such as Learning by Doing or Design Thinking, used by other renowned institutions such as Yale or Stanford.

This action-oriented method will be applied throughout the entire academic itinerary that the student undertakes with TECH Euromed University. Students will be confronted with multiple real-life situations and will have to integrate knowledge, research, discuss and defend their ideas and decisions. All this with the premise of answering the question of how they would act when facing specific events of complexity in their daily work.



Relearning Methodology

At TECH Euromed University, case studies are enhanced with the best 100% online teaching method: Relearning.

This method breaks with traditional teaching techniques to put the student at the center of the equation, providing the best content in different formats. In this way, it manages to review and reiterate the key concepts of each subject and learn to apply them in a real context.

In the same line, and according to multiple scientific researches, reiteration is the best way to learn. For this reason, TECH Euromed University offers between 8 and 16 repetitions of each key concept within the same lesson, presented in a different way, with the objective of ensuring that the knowledge is completely consolidated during the study process.

Relearning will allow you to learn with less effort and better performance, involving you more in your specialization, developing a critical mindset, defending arguments, and contrasting opinions: a direct equation to success.



A 100% online Virtual Campus with the best teaching resources

In order to apply its methodology effectively, TECH Euromed University focuses on providing graduates with teaching materials in different formats: texts, interactive videos, illustrations and knowledge maps, among others. All of them are designed by qualified teachers who focus their work on combining real cases with the resolution of complex situations through simulation, the study of contexts applied to each professional career and learning based on repetition, through audios, presentations, animations, images, etc.

The latest scientific evidence in the field of Neuroscience points to the importance of taking into account the place and context where the content is accessed before starting a new learning process. Being able to adjust these variables in a personalized way helps people to remember and store knowledge in the hippocampus to retain it in the long term. This is a model called Neurocognitive context-dependent e-learning that is consciously applied in this university qualification.

In order to facilitate tutor-student contact as much as possible, you will have a wide range of communication possibilities, both in real time and delayed (internal messaging, telephone answering service, email contact with the technical secretary, chat and videoconferences).

Likewise, this very complete Virtual Campus will allow TECH Euromed University students to organize their study schedules according to their personal availability or work obligations. In this way, they will have global control of the academic content and teaching tools, based on their fast-paced professional update.



The online study mode of this program will allow you to organize your time and learning pace, adapting it to your schedule”

The effectiveness of the method is justified by four fundamental achievements:

1. Students who follow this method not only achieve the assimilation of concepts, but also a development of their mental capacity, through exercises that assess real situations and the application of knowledge.
2. Learning is solidly translated into practical skills that allow the student to better integrate into the real world.
3. Ideas and concepts are understood more efficiently, given that the example situations are based on real-life.
4. Students like to feel that the effort they put into their studies is worthwhile. This then translates into a greater interest in learning and more time dedicated to working on the course.

The university methodology top-rated by its students

The results of this innovative teaching model can be seen in the overall satisfaction levels of TECH Euromed University graduates.

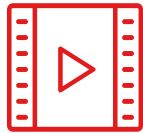
The students' assessment of the teaching quality, the quality of the materials, the structure of the program and its objectives is excellent. Not surprisingly, the institution became the top-rated university by its students according to the global score index, obtaining a 4.9 out of 5.

Access the study contents from any device with an Internet connection (computer, tablet, smartphone) thanks to the fact that TECH Euromed University is at the forefront of technology and teaching.

You will be able to learn with the advantages that come with having access to simulated learning environments and the learning by observation approach, that is, Learning from an expert.



As such, the best educational materials, thoroughly prepared, will be available in this program:



Study Material

All teaching material is produced by the specialists who teach the course, specifically for the course, so that the teaching content is highly specific and precise.

This content is then adapted in an audiovisual format that will create our way of working online, with the latest techniques that allow us to offer you high quality in all of the material that we provide you with.



Practicing Skills and Abilities

You will carry out activities to develop specific competencies and skills in each thematic field. Exercises and activities to acquire and develop the skills and abilities that a specialist needs to develop within the framework of the globalization we live in.



Interactive Summaries

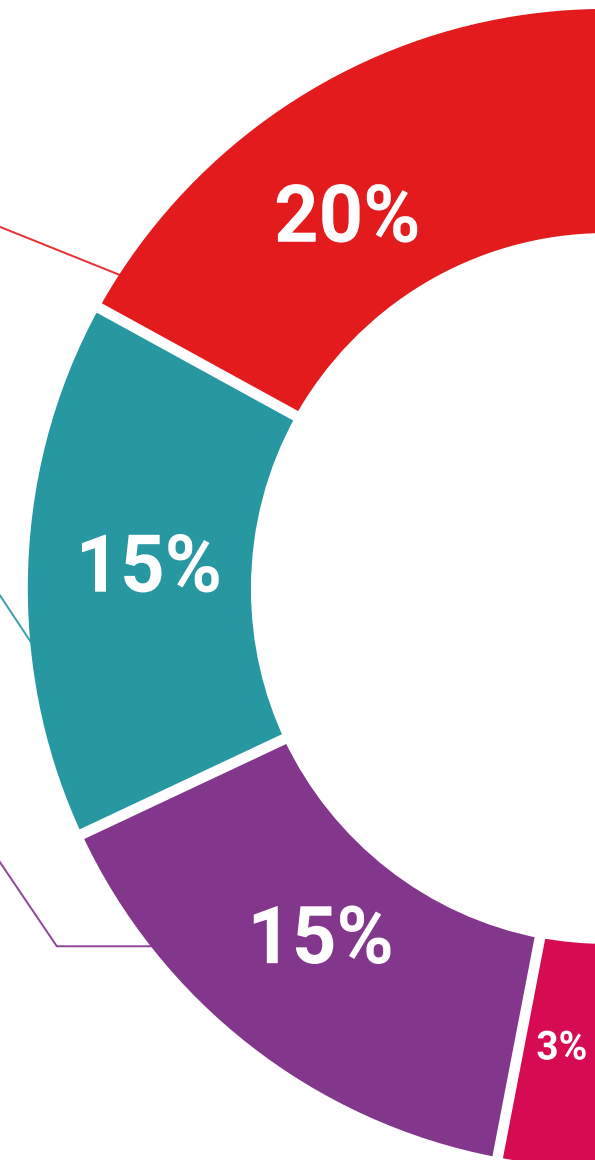
We present the contents attractively and dynamically in multimedia lessons that include audio, videos, images, diagrams, and concept maps in order to reinforce knowledge.

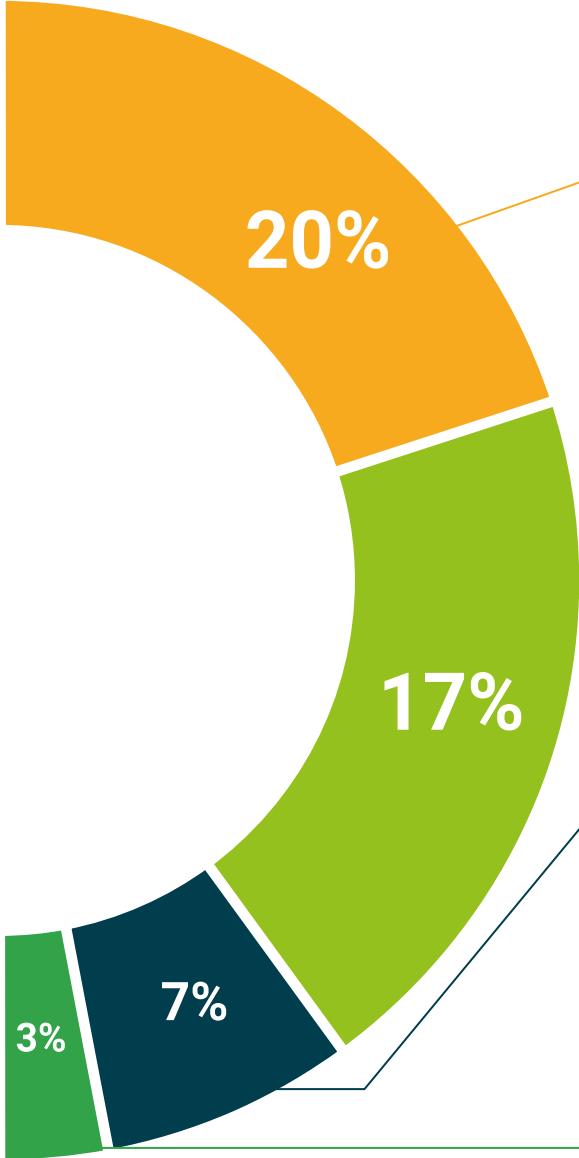
This exclusive educational system for presenting multimedia content was awarded by Microsoft as a "European Success Story".



Additional Reading

Recent articles, consensus documents, international guides... In our virtual library you will have access to everything you need to complete your education.





Case Studies

Students will complete a selection of the best *case studies* in the field. Cases that are presented, analyzed, and supervised by the best specialists in the world.



Testing & Retesting

We periodically assess and re-assess your knowledge throughout the program. We do this on 3 of the 4 levels of Miller's Pyramid.



Classes

There is scientific evidence suggesting that observing third-party experts can be useful.
Learning from an expert strengthens knowledge and memory, and generates confidence for future difficult decisions.



Quick Action Guides

TECH Euromed University offers the most relevant contents of the course in the form of worksheets or quick action guides. A synthetic, practical and effective way to help students progress in their learning.



07

Certificate

The Master's Degree in Computing and Programming Languages guarantees students, in addition to the most rigorous and up-to-date education, access to a Master's Degree diploma issued by TECH Global University and another one from the Euromed University of Fes.



“

Successfully complete this program and receive your university qualification without having to travel or fill out laborious paperwork”

The **Master's Degree in Computing and Programming Languages** is the most complete and up-to-date educational program on the market. Upon graduation, students will receive a university diploma issued by TECH Global University, as well as one from Euromed University of Fes.

These continuing education and professional development programs from TECH Global University and Euromed University of Fes ensure the acquisition of key competencies in the field, granting significant academic value to students who successfully complete the program and pass the evaluations.

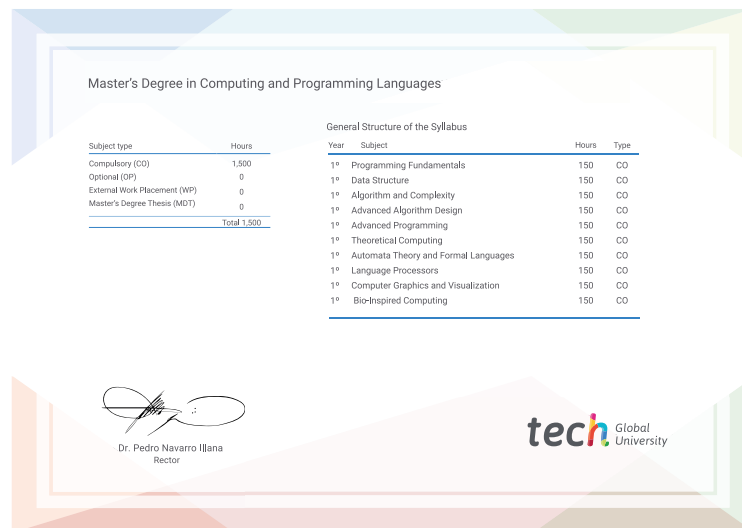
This double certification from two prestigious universities offers dual recognition for a thorough and high-quality education, ensuring that graduates receive a qualification recognized both nationally and internationally. This academic achievement will position them as highly skilled professionals, ready to meet the challenges and demands in their professional field.

Title: **Master's Degree in Computing and Programming Languages**

Modality: **online**

Duration: **12 months**

Accreditation: **60 ECTS**



*Apostille Convention. In the event that the student wishes to have their paper diploma issued with an apostille, TECH Euromed University will make the necessary arrangements to obtain it, at an additional cost.



Master's Degree Computing and Programming Languages

- » Modality: online
- » Duration: 12 months
- » Certificate: TECH Euromed University
- » Schedule: at your own pace
- » Exams: online

Master's Degree

Computing and Programming Languages