Máster Título Propio
Computación y Lenguajes



tech universidad tecnológica



tech universidad tecnológica

Máster Título Propio

Computación y Lenguajes

Modalidad: **Online**
Duración: **12 meses**
Titulación: **TECH Universidad Tecnológica**
Horas lectivas: **1.500 h.**
Acceso web: **www.techtitute.com/informatica/master/master-computacion-lenguajes**

Índice

01
Presentación
pág. 4

02
Objetivos
pág. 8

03
Dirección del curso
pág. 14

04
Estructura y contenido
pág. 18

05
Metodología
pág. 30

06
Titulación
pág. 38

01

Presentación

El profesional de la informática necesita mantener sus capacidades totalmente actualizadas para poder seguir actuando en su campo de competencia de manera óptima, sin perder ninguno de los avances que, de manera vertiginosa se van incorporando a este territorio. Esta actualización está diseñada para permitir a los alumnos, conocer de forma completa y profunda los conocimientos esenciales y las novedades más interesantes en el diseño de algoritmos en el desarrollo de proyectos informáticos, con los métodos más innovadores y eficientes del sector.





Adquiere los conocimientos fundamentales sobre Computación y la manera de aplicarlos de forma exitosa en el desarrollo de proyectos informáticos, en un Máster Título Propio de alta competencia"

Este programa se centra en los fundamentos de programación y la estructura de datos, la algoritmia y complejidad, así como el diseño avanzado de algoritmos, la programación avanzada, o los procesadores de lenguajes y la informática gráfica, entre otros aspectos relacionados con este ámbito de la informática.

Este programa proporciona al alumno herramientas y habilidades específicas para que desarrolle con éxito su actividad profesional en el amplio entorno de la Computación y Lenguajes. Trabaja competencias claves como el conocimiento de la realidad y práctica diaria en distintas áreas informáticas y desarrolla la responsabilidad en el seguimiento y supervisión de su trabajo, así como habilidades específicas dentro de este campo.

Además, al tratarse de un programa 100 % online, el alumno no está condicionado por horarios fijos ni necesidad de trasladarse a otro lugar físico, sino que puede acceder a los contenidos en cualquier momento del día, equilibrando su vida laboral o personal con la académica.

El equipo docente de este programa en Computación y Lenguajes ha realizado una cuidadosa selección de cada uno de los temas de esta actualización para ofrecer al alumno una oportunidad de estudio lo más completa posible y ligada siempre con la actualidad.

Este **Máster Título Propio en Computación y Lenguajes** contiene el programa más completo y actualizado del mercado. Sus características más destacadas son:

- El desarrollo de casos prácticos presentados por expertos en Computación y Lenguajes
- Los contenidos gráficos, esquemáticos y eminentemente prácticos con los que están concebidos recogen una información científica y práctica sobre aquellas disciplinas indispensables para el ejercicio profesional
- Los ejercicios prácticos donde realizar el proceso de autoevaluación para mejorar el aprendizaje
- Su especial hincapié en metodologías innovadoras en Computación y Lenguajes
- Las lecciones teóricas, preguntas al experto, foros de discusión de temas controvertidos y trabajos de reflexión individual
- La disponibilidad de acceso a los contenidos desde cualquier dispositivo fijo o portátil con conexión a internet

Una oportunidad excepcional de aprender de forma cómoda y sencilla los procesos y conocimientos matemáticos y de base necesarios para realizar programaciones informáticas de calidad”

Un Máster Título Propio que sustenta su eficacia en la tecnología educativa más valorada del mercado, con sistemas audiovisuales y de estudio que te permitirán aprender de forma más rápida y cómoda"

Ponemos a tu servicio un material didáctico amplio y claro, que incorpora todos los temas de interés en la actualidad, para el profesional que quiere avanzar en Computación y Lenguajes.

Un estudio de alta incidencia educativa que te permitirá adaptar el esfuerzo a tus necesidades, compaginando flexibilidad e intensidad.

Su contenido multimedia, elaborado con la última tecnología educativa, permitirá al profesional un aprendizaje situado y contextual, es decir, un entorno simulado que proporcionará una actualización inmersiva programada para entrenarse ante situaciones reales.

El diseño de este programa se centra en el Aprendizaje Basado en Problemas, mediante el cual el profesional deberá tratar de resolver las distintas situaciones de práctica profesional que se le planteen a lo largo del curso académico. Para ello, el profesional contará con la ayuda de un novedoso sistema de vídeo interactivo realizado por reconocidos expertos en Computación y Lenguajes, y con gran experiencia.



02

Objetivos

El programa en Computación y Lenguajes se ha creado de manera específica para el profesional que busca avanzar en este campo de forma rápida y con calidad real, organizándolo con base en objetivos realistas y de alto valor que le impulsarán a otro nivel de trabajo en este campo.



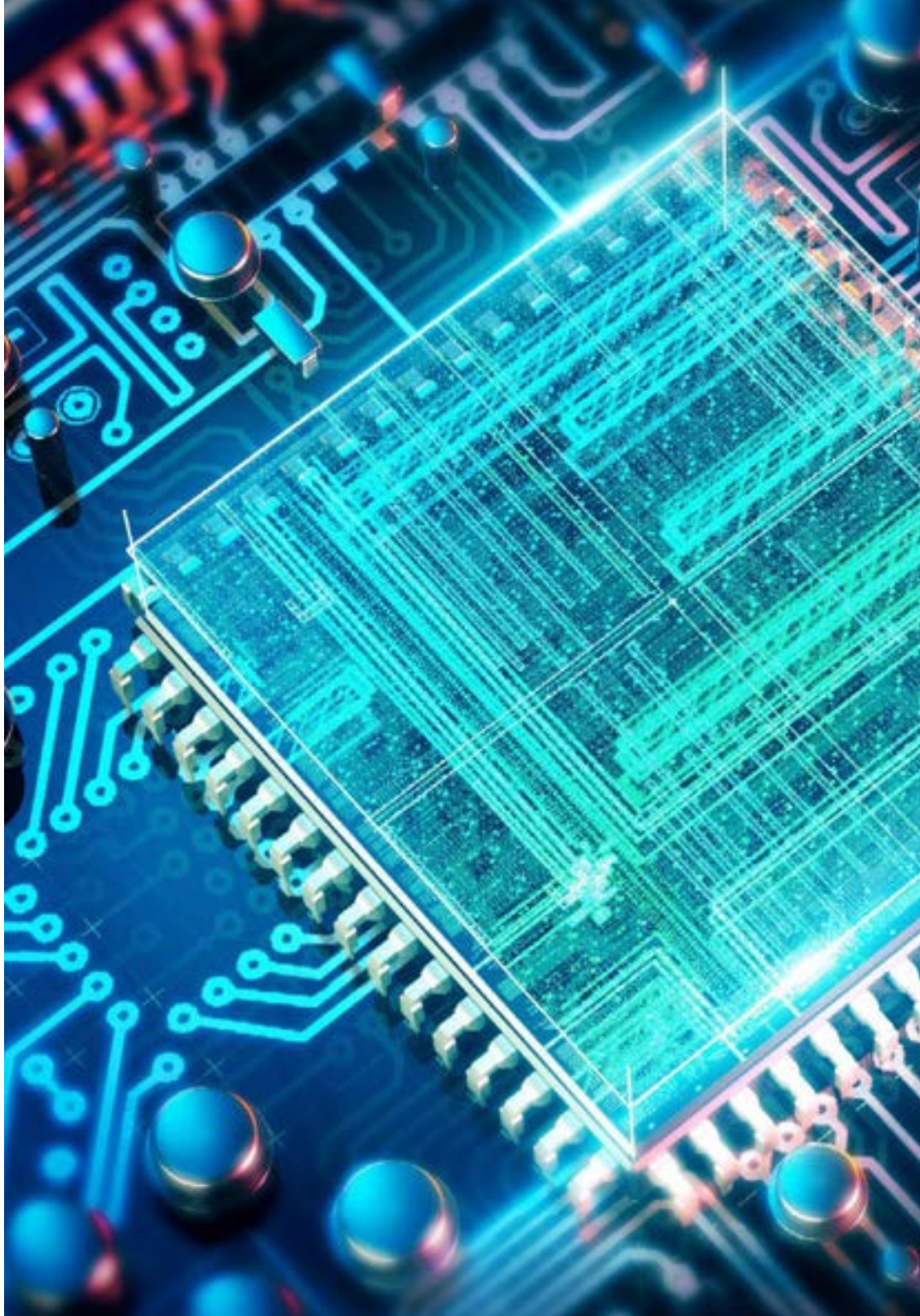


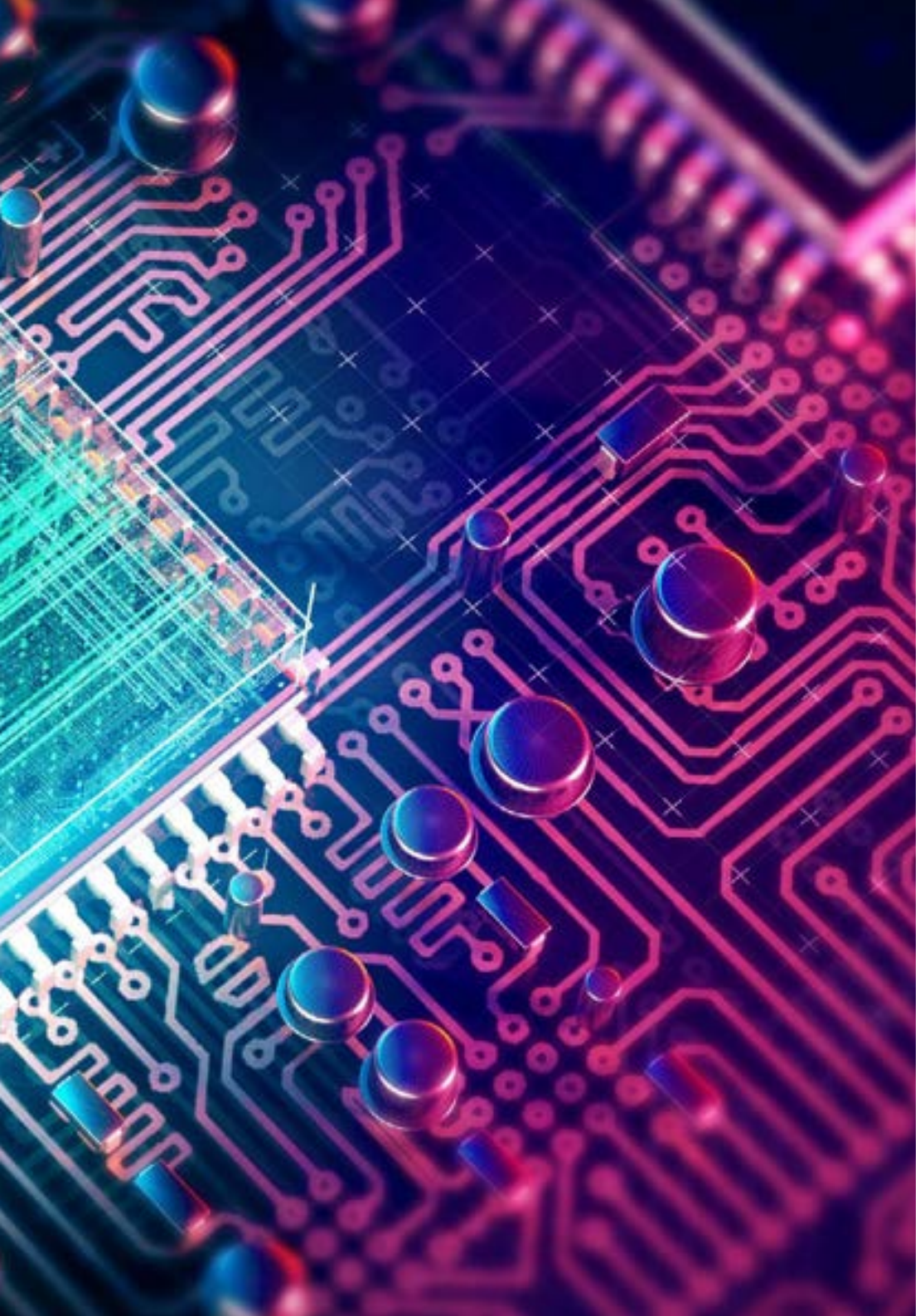
Nuestro objetivo es poner al servicio de los profesionales del campo de la informática, una actualización de alta calidad que les permita intervenir con solvencia en Computación y Lenguajes"

Objetivo general

- Capacitar científica y tecnológicamente, así como preparar para el ejercicio profesional de la Computación y los lenguajes, todo ello con una formación transversal y versátil adaptada a las nuevas tecnologías e innovaciones en este campo

Aprovecha la oportunidad y da el paso para ponerte al día en las últimas novedades en Computación y Lenguajes"





Objetivos específicos

Módulo 1. Fundamentos de programación

- Comprender la estructura básica de un ordenador, el Software y de los lenguajes de programación de propósito general
- Aprender a diseñar e interpretar algoritmos, que son la base necesaria para poder desarrollar programas informáticos
- Entender los elementos esenciales de un programa informático, como son los distintos tipos de datos, operadores, expresiones, sentencias, E/S y sentencias de control
- Comprender las distintas estructuras de datos disponibles en los lenguajes de programación de propósito general tanto estáticas como dinámicas, así como adquirir los conocimientos esenciales para el manejo de ficheros
- Conocer las distintas técnicas de pruebas en los programas informáticos y la importancia de generar una buena documentación junto con un buen código fuente
- Aprender los conceptos básicos del lenguaje de programación C++, uno de los más usados a nivel mundial

Módulo 2. Estructura de datos

- Aprender los fundamentos de la programación en el lenguaje C++, incluyendo clases, variables, expresiones condicionales y objetos
- Entender los tipos abstractos de datos, los tipos de estructuras de datos lineales, estructuras de datos jerárquicas simples y complejas, así como su implementación en C++
- Comprender el funcionamiento de estructuras de datos avanzadas distintas de las habituales
- Conocer la teoría y la práctica relacionada con el uso de montículos y colas de prioridad
- Aprender el funcionamiento de las tablas hash, como tipos abstractos de datos y funciones
- Entender la teoría de grafos, así como algoritmos y concepto avanzados sobre grafos

Módulo 3. Algoritmia y complejidad

- Aprender las principales estrategias de diseño de algoritmos, así como los distintos métodos y medidas para de cálculo de los mismos
- Conocer los principales algoritmos de ordenación usados en el desarrollo de Software
- Entender el funcionamiento de los distintos algoritmos con árboles, *Heaps* y Grafos
- Comprender el funcionamiento de los algoritmos *Greedy*, su estrategia y ejemplos de su uso en los principales problemas conocidos
- Conocer también el uso de algoritmos *Greedy* sobre Grafos
- Aprender las principales estrategias de búsqueda de caminos mínimos, con el planteamiento de problemas esenciales del ámbito y algoritmos para su resolución
- Entender la técnica de *Backtracking* y sus principales usos, así como otras técnicas alternativas

Módulo 4. Diseño avanzado de algoritmos

- Profundizar en el diseño avanzado de algoritmos, analizando algoritmos recursivos y tipo divide y conquista, así como realizando análisis amortizado
- Comprender los conceptos de programación dinámica y los algoritmos para problemas NP
- Entender el funcionamiento de la optimización combinatoria, así como los distintos algoritmos de aleatorización y algoritmos paralelos
- Conocer y comprender el funcionamiento de los distintos métodos de búsqueda local y con candidatos
- Aprender los mecanismos de verificación de formal de programas y de programas iterativos, incluyendo la lógica de primer orden y el sistema formal de Hoare
- Aprender el funcionamiento de algunos de los principales métodos numéricos como el método de la bisección, el método de Newton Raphson y el método de la secante

Módulo 5. Programación avanzada

- Profundizar en los conocimientos de programación, especialmente en lo relaciona a la programación orientada a objetos, y los distintos tipos de relaciones entre clases existentes
- Conocer los distintos patrones de diseño para problemas orientados a objetos
- Aprender sobre la programación orientada a eventos y el desarrollo de interfaces de usuario con Qt
- Adquirir los conocimientos esenciales de la programación concurrente, los procesos y los hilos
- Aprender a gestionar el uso de los hilos y la sincronización, así como la resolución de los problemas comunes dentro de la programación concurrente
- Entender la importancia de la documentación y las pruebas en el desarrollo del Software

Módulo 6. Informática teórica

- Comprender los conceptos matemáticos teóricos esenciales tras la informática, como son la lógica proposicional, la teoría de conjuntos y los conjuntos numerables y no numerables
- Entender los conceptos de lenguajes y gramáticas formales, así como el de máquinas de Turing en sus distintas variantes
- Aprender sobre los distintos tipos de problemas indecibles y de problemas intratables, incluyendo las distintas variantes de los mismos y sus aproximaciones
- Comprender el funcionamiento de las distintas clases de lenguajes basados en la aleatorización y otros tipos de clases y gramáticas
- Conocer otros sistemas de avanzados cómputo como son la Computación con membranas, la Computación con ADN y la Computación cuántica

Módulo 7. Teoría de autómatas y lenguajes formales

- Comprender la teoría de autómatas y lenguajes formales, aprendiendo los conceptos de alfabetos, cadenas y lenguajes, así como a realizar demostraciones formales
- Profundizar en los distintos tipos de autómatas finitos, ya sean deterministas o no deterministas
- Aprender los conceptos básicos y avanzados relacionados con los lenguajes y las expresiones regulares, así como la aplicación del lema de bombeo y la clausura de los lenguajes regulares
- Entender las gramáticas independientes de contexto, así como el funcionamiento de los autómatas a pila
- Profundizar en las formas normales, el lema de bombeo de las gramáticas independientes de contexto y propiedades de los lenguajes independientes de contexto

Módulo 8. Procesadores de lenguajes

- Introducir los conceptos relacionados con el proceso de compilación y los distintos tipos de análisis: léxico, sintáctico y semántico
- Conocer el funcionamiento de un analizador léxico, su implementación y recuperación de errores
- Profundizar en el conocimiento del análisis sintáctico, tanto descendente como ascendente, pero profundizando especialmente en los distintos tipos de analizadores sintácticos ascendentes
- Entender el funcionamiento de los analizadores semánticos, la tradición dirigida por la sintaxis, la tabla de símbolos y los distintos tipos
- Aprender los distintos mecanismos de generación de código, tanto en entornos de ejecución como para la generación de código intermedio
- Sentar las bases de la optimización de código, incluyendo la reordenación de expresiones y la optimización de bucles

Módulo 9. Informática gráfica y visualización

- Introducir los conceptos esenciales de la informática gráfica y la visualización por ordenador, como la teoría del color y sus modelos y las propiedades de la luz
- Comprender el funcionamiento de las primitivas de salida y sus algoritmos, tanto de dibujo de líneas, como de dibujo de circunferencias y de relleno
- Profundizar en el estudio de las distintas transformaciones tanto 2D como 3D, y sus sistemas de coordenadas y visualización por ordenador
- Aprender a realizar proyecciones y cortes en 3D, así como la eliminación de superficies ocultas
- Aprender la teoría relacionada con la interpolación y curvas paramétricas, así como lo relacionado con las Curvas Bézier y los B-splines

Módulo 10. Computación bioinspirada

- Introducir el concepto de Computación bioinspirada, así como comprender el funcionamiento de los distintos tipos de algoritmos de adaptación social y de algoritmos genéticos
- Profundizar en el estudio de los distintos modelos de Computación evolutiva, conociendo sus estrategias, programación, algoritmos y modelos basados en estimación de distribuciones
- Entender las principales estrategias de exploración-explotación del espacio para algoritmos genéticos
- Comprender el funcionamiento de la programación evolutiva aplicada a problemas de aprendizaje y de los problemas multiobjetivo
- Aprender los conceptos esenciales relacionados con redes neuronales y entender el funcionamiento de casos de uso reales aplicados a áreas tan dispares como la investigación médica, la economía y la visión artificial

03

Competencias

Después de superar las evaluaciones del programa en Computación y Lenguajes, el profesional habrá adquirido las competencias necesarias para conocer los principios fundamentales de la Computación con la capacidad de trabajar con lenguajes de programación y datos.



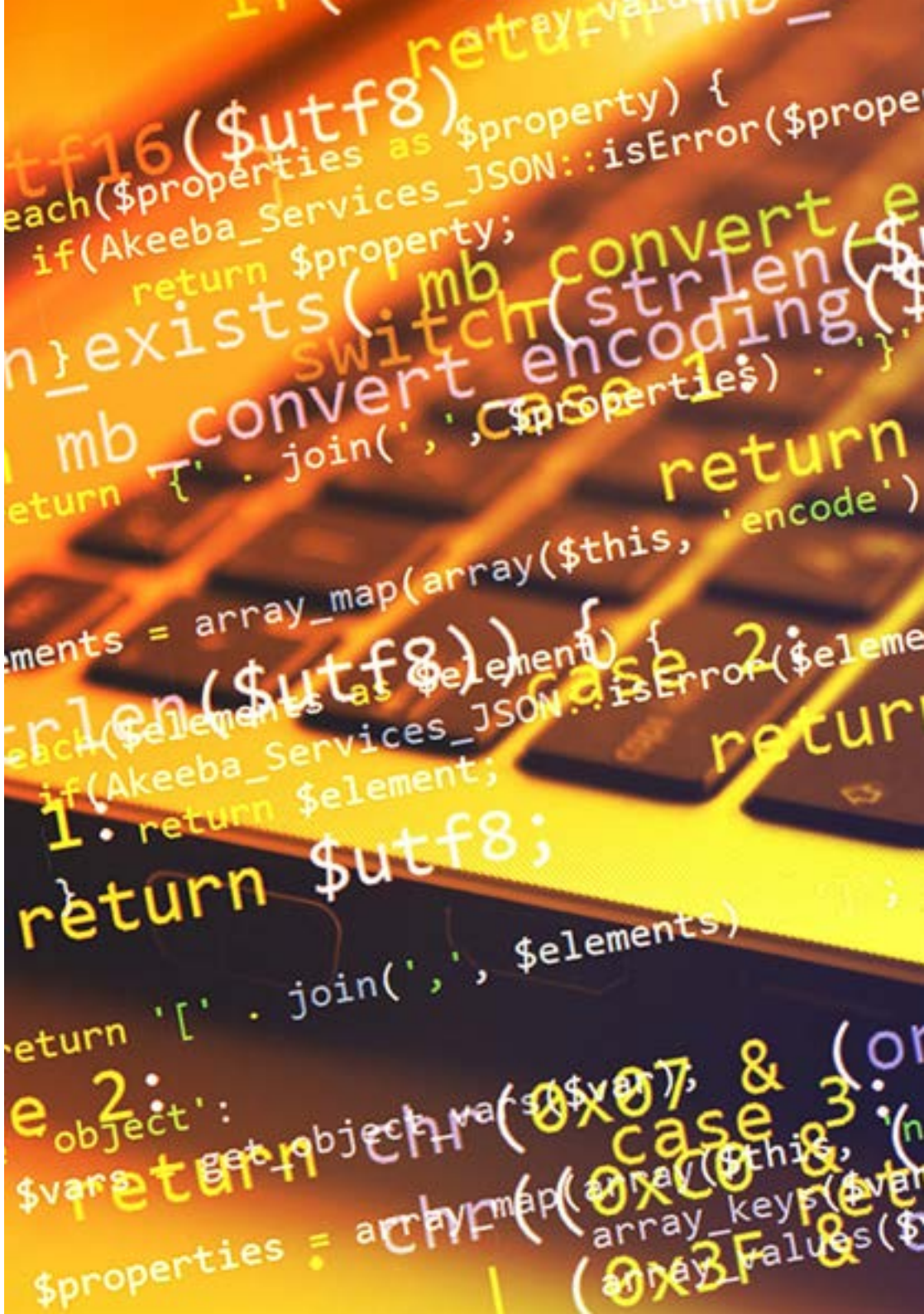


Adquiere la capacidad de realizar nuevos desarrollos informáticos trabajando desde la comprensión y el control de los diferentes lenguajes y algoritmos y su aplicación práctica"

Competencia general

- Realizar de manera correcta las labores vinculadas con la Computación y el Lenguaje informático

Mejora tus competencias para participar en varios proyectos tecnológicos”





Competencias específicas

- Diseñar algoritmos para desarrollar programas informáticos y aplicar el Lenguaje de programación
- Comprender y utilizar la estructura de datos informáticos
- Utilizar los algoritmos necesarios para la resolución de problemas informáticos
- Conocer en profundidad el diseño avanzado de algoritmos, así como los métodos de búsqueda
- Llevar a cabo tareas de programación informática
- Comprender y aplicar la teoría que existe detrás de la informática, como es el caso de las matemáticas
- Conocer la teoría de autómatas y aplicar el Lenguaje informático
- Conocer los fundamentos teóricos de los lenguajes de programación y las técnicas de procesamiento léxico, sintáctico y semántico asociadas
- Comprender los conceptos básicos de matemática y complejidad computacional para aplicarlos a la resolución de problemas informáticos
- Conocer y aplicar los principios fundamentales de la Computación para realizar nuevos desarrollos informáticos

04

Estructura y contenido

La estructura de los contenidos se ha creado de forma que los conocimientos sean asimilados de manera progresiva, consiguiendo una trayectoria de crecimiento que le llevará a la excelencia en su profesión.





Todos los ámbitos de interés que necesitar dominar para trabajar con seguridad y acierto en Computación y Lenguajes, recopilados en un temario de la máxima calidad"

Módulo 1. Fundamentos de programación

1.1. Introducción a la programación
- 1.1.1. Estructura básica de un ordenador
- 1.1.2. Software
- 1.1.3. Lenguajes de programación
- 1.1.4. Ciclo de vida de una aplicación informática

1.2. Diseño de algoritmos
- 1.2.1. La resolución de problemas
- 1.2.2. Técnicas descriptivas
- 1.2.3. Elementos y estructura de un algoritmo

1.3. Elementos de un programa
- 1.3.1. Origen y características del Lenguaje C++
- 1.3.2. El entorno de desarrollo
- 1.3.3. Concepto de programa
- 1.3.4. Tipos de datos fundamentales
- 1.3.5. Operadores
- 1.3.6. Expresiones
- 1.3.7. Sentencias
- 1.3.8. Entrada y salida de datos

1. 4. Sentencias de control
- 1.4.1. Sentencias
- 1.4.2. Bifurcaciones
- 1.4.3. Bucles

1.5. Abstracción y modularidad: funciones
- 1.5.1. Diseño modular
- 1.5.2. Concepto de función y utilidad
- 1.5.3. Definición de una función
- 1.5.4. Flujo de ejecución en la llamada de una función
- 1.5.5. Prototipo de una función
- 1.5.6. Devolución de resultados
- 1.5.7. Llamada a una función: parámetros
- 1.5.8. Paso de parámetros por referencia y por valor
- 1.5.9. Ámbito identificador

1.6. Estructuras de datos estáticas
- 1.6.1. *Arrays*
- 1.6.2. Matrices. Poliedros
- 1.6.3. Búsqueda y ordenación
- 1.6.4. Cadenas. Funciones de E/S para cadenas
- 1.6.5. Estructuras. Uniones
- 1.6.6. Nuevos tipos de datos

1.7. Estructuras de datos dinámicas: punteros
- 1.7.1. Concepto. Definición de puntero
- 1.7.2. Operadores y operaciones con punteros
- 1.7.3. *Arrays* de punteros
- 1.7.4. Punteros y *Arrays*
- 1.7.5. Punteros a cadenas
- 1.7.6. Punteros a estructuras
- 1.7.7. Indirección múltiple
- 1.7.8. Punteros a funciones
- 1.7.9. Paso de funciones, estructuras y *Arrays* como parámetros de funciones

1.8. Ficheros
- 1.8.1. Conceptos básicos
- 1.8.2. Operaciones con ficheros
- 1.8.3. Tipos de ficheros
- 1.8.4. Organización de los ficheros
- 1.8.5. Introducción a los ficheros C++
- 1.8.6. Manejo de ficheros

1.9. Recursividad
- 1.9.1. Definición de recursividad
- 1.9.2. Tipos de recursión
- 1.9.3. Ventajas e inconvenientes
- 1.9.4. Consideraciones
- 1.9.5. Conversión recursivo-iterativa
- 1.9.6. La pila de recursión



1.10. Prueba y documentación
- 1.10.1. Pruebas de programas
- 1.10.2. Prueba de la caja blanca
- 1.10.3. Prueba de la caja negra
- 1.10.4. Herramientas para realizar las pruebas
- 1.10.5. Documentación de programas

Módulo 2. Estructura de datos

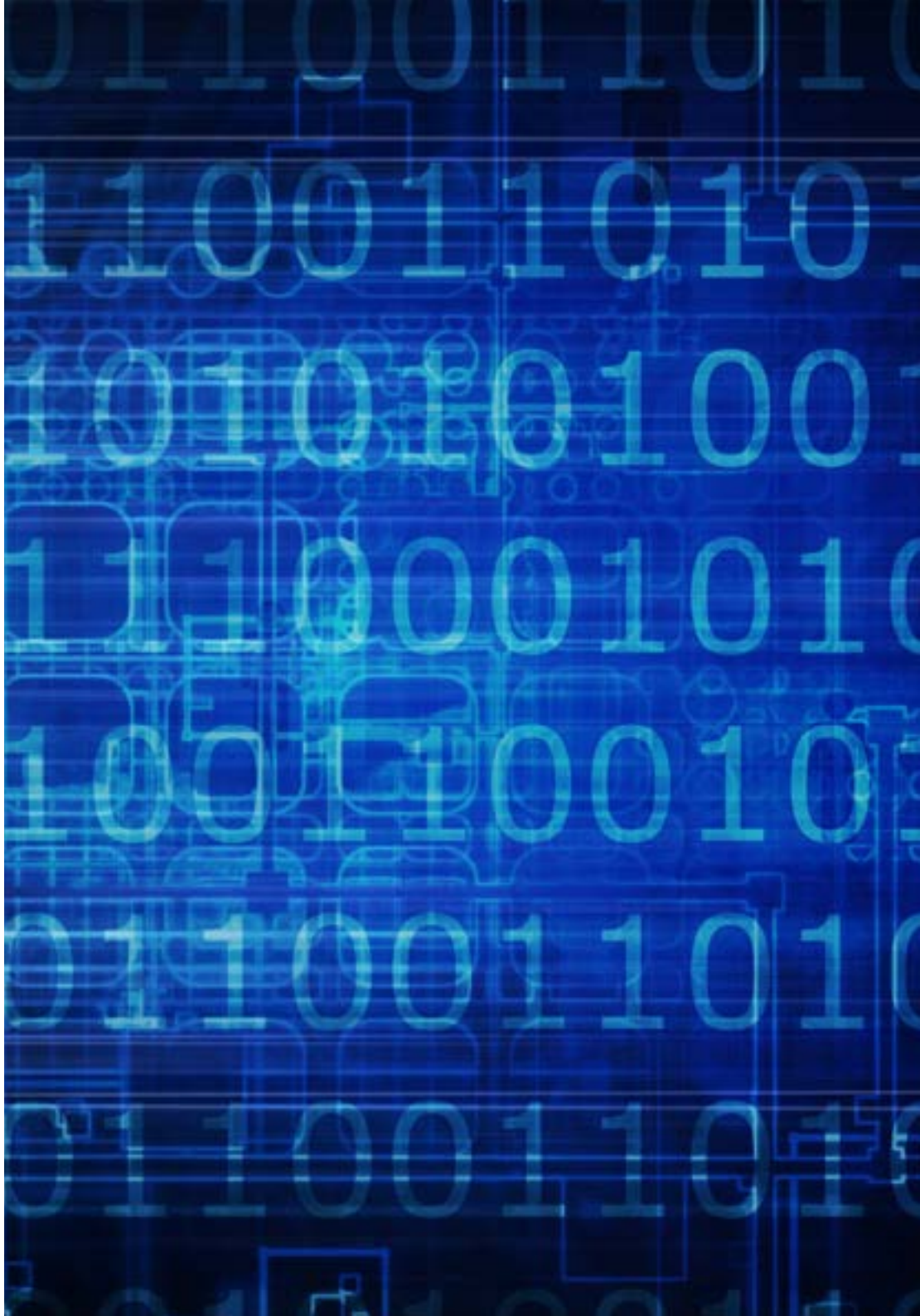
2.1. Introducción a la programación en C++
- 2.1.1. Clases, constructores, métodos y atributos
- 2.1.2. Variables
- 2.1.3. Expresiones condicionales y bucles
- 2.1.4. Objetos

2.2. Tipos abstractos de datos (TAD)
- 2.2.1. Tipos de datos
- 2.2.2. Estructuras básicas y TAD
- 2.2.3. Vectores y arrays

2.3. Estructuras de datos lineales
- 2.3.1. TAD Lista. Definición
- 2.3.2. Listas enlazadas y doblemente enlazadas
- 2.3.3. Listas ordenadas
- 2.3.4. Listas en C++
- 2.3.5. TAD pila
- 2.3.6. TAD cola
- 2.3.7. Pila y cola en C++

2.4. Estructuras de datos jerárquicas
- 2.4.1. TAD árbol
- 2.4.2. Recorridos
- 2.4.3. Árboles n-arios
- 2.4.4. Árboles binarios
- 2.4.5. Árboles binarios de búsqueda

2.5. Estructuras de datos jerárquicas: árboles complejos
  2.5.1. Árboles perfectamente equilibrados o de altura mínima
  2.5.2. Árboles multicamino
  2.5.3. Referencias bibliográficas
2.6. Montículos y cola de prioridad
  2.6.1. TAD montículos
  2.6.2. TAD cola de prioridad
2.7. Tablas hash
  2.7.1. TAD Tabla *Hash*
  2.7.2. Funciones *Hash*
  2.7.3. Función *Hash* en tablas *Hash*
  2.7.4. Redispersión
  2.7.5. Tablas *Hash* abiertas
2.8. Grafos
  2.8.1. TAD Grafo
  2.8.2. Tipos de Grafo
  2.8.3. Representación gráfica y operaciones básicas
  2.8.4. Diseño de Grafos
2.9. Algoritmos y conceptos avanzados sobre Grafos
  2.9.1. Problemas sobre Grafos
  2.9.2. Algoritmos sobre caminos
  2.9.3. Algoritmos de búsqueda o recorridos
  2.9.4. Otros algoritmos
2.10. Otras estructuras de datos
  2.10.1. Conjuntos
  2.10.2. Arrays paralelos
  2.10.3. Tablas de símbolos
  2.10.4. *Tries*



Módulo 3. Algoritmia y complejidad

3.1. Introducción a las estrategias de diseño de algoritmos
- 3.1.1. Recursividad
- 3.1.2. Divide y conquista
- 3.1.3. Otras estrategias

3.2. Eficiencia y análisis de los algoritmos
- 3.2.1. Medidas de eficiencia
- 3.2.2. Medir el tamaño de la entrada
- 3.2.3. Medir el tiempo de ejecución
- 3.2.4. Caso peor, mejor y medio
- 3.2.5. Notación asintónica
- 3.2.6. Criterios de análisis matemático de algoritmos no recursivos
- 3.2.7. Análisis matemático de algoritmos recursivos
- 3.2.8. Análisis empírico de algoritmos

3.3. Algoritmos de ordenación
- 3.3.1. Concepto de ordenación
- 3.3.2. Ordenación de la burbuja
- 3.3.3. Ordenación por selección
- 3.3.4. Ordenación por inserción
- 3.3.5. Ordenación por mezcla (Merge Sort)
- 3.3.6. Ordenación rápida (QuickSort)

3.4. Algoritmos con árboles
- 3.4.1. Concepto de árbol
- 3.4.2. Árboles binarios
- 3.4.3. Recorridos de árbol
- 3.4.4. Representar expresiones
- 3.4.5. Árboles binarios ordenados
- 3.4.6. Árboles binarios balanceados

3.5. Algoritmos con *Heaps*
- 3.5.1. Los *Heaps*
- 3.5.2. El algoritmo HeapSort
- 3.5.3. Las colas de prioridad

3.6. Algoritmos con Grafos
- 3.6.1. Representación
- 3.6.2. Recorrido en anchura
- 3.6.3. Recorrido en profundidad
- 3.6.4. Ordenación topológica

3.7. Algoritmos *Greedy*
- 3.7.1. La estrategia *Greedy*
- 3.7.2. Elementos de la estrategia *Greedy*
- 3.7.3. Cambio de monedas
- 3.7.4. Problema del viajante
- 3.7.5. Problema de la mochila

3.8. Búsqueda de caminos mínimos
- 3.8.1. El problema del camino mínimo
- 3.8.2. Arcos negativos y ciclos
- 3.8.3. Algoritmo de Dijkstra

3.9. Algoritmos greedy sobre grafos
- 3.9.1. El árbol de recubrimiento mínimo
- 3.9.2. El algoritmo de Prim
- 3.9.3. El algoritmo de Kruskal
- 3.9.4. Análisis de complejidad

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

Módulo 4. Diseño avanzado de algoritmos

4.1. Análisis de algoritmos recursivos y tipo divide y conquista
- 4.1.1. Planteamiento y resolución de ecuaciones de recurrencia homogéneas y no homogéneas
- 4.1.2. Descripción general de la estrategia divide y conquista

4.2. Análisis amortizado
- 4.2.1. El análisis agregado
- 4.2.2. El método de contabilidad
- 4.2.3. El método del potencial

4.3. Programación dinámica y algoritmos para problemas NP
- 4.3.1. Características de la programación dinámica
- 4.3.2. Vuelta atrás: backtracking
- 4.3.3. Ramificación y poda

4.4. Optimización combinatoria
- 4.4.1. Representación de problemas
- 4.4.2. Optimización en 1D

4.5. Algoritmos de aleatorización
- 4.5.1. Ejemplos de algoritmos de aleatorización
- 4.5.2. El teorema Buffon
- 4.5.3. Algoritmo de Monte Carlo
- 4.5.4. Algoritmo Las Vegas

4.6. Búsqueda local y con candidatos
- 4.6.1. *Garcient Ascent*
- 4.6.2. *Hill Climbing*
- 4.6.3. *Simulated Annealing*
- 4.6.4. *Tabu Search*
- 4.6.5. Búsqueda con candidatos

4.7. Verificación formal de programas
- 4.7.1. Especificación de abstracciones funcionales
- 4.7.2. El Lenguaje de la lógica de primer orden
- 4.7.3. El sistema formal de Hoare

4.8. Verificación de programas iterativos
- 4.8.1. Reglas del sistema formal de Hoare
- 4.8.2. Concepto de invariante de iteraciones

4.9. Métodos numéricos
- 4.9.1. El método de la bisección
- 4.9.2. El método de Newton Raphson
- 4.9.3. El método de la secante

4.10. Algoritmos paralelos
- 4.10.1. Operaciones binarias paralelas
- 4.10.2. Operaciones paralelas con grafos
- 4.10.3. Paralelismo en divide y vencerás
- 4.10.4. Paralelismo en programación dinámica

Módulo 5. Programación avanzada

5.1. Introducción a la programación orientada a objetos
- 5.1.1. Introducción a la programación orientada a objetos
- 5.1.2. Diseño de clases
- 5.1.3. Introducción a UML para el modelado de los problemas

5.2. Relaciones entre clases
- 5.2.1. Abstracción y herencia
- 5.2.2. Conceptos avanzados de herencia
- 5.2.3. Polimorfismo
- 5.2.4. Composición y agregación

5.3. Introducción a los patrones de diseño para problemas orientados a objetos
- 5.3.1. Qué son los patrones de diseño
- 5.3.2. Patrón *Factory*
- 5.3.3. Patrón *Singleton*
- 5.3.4. Patrón *Observer*
- 5.3.5. Patrón *Composite*

5.4. Excepciones
- 5.4.1. ¿Qué son las excepciones?
- 5.4.2. Captura y gestión de excepciones
- 5.4.3. Lanzamiento de excepciones
- 5.4.4. Creación de excepciones

5.5. Interfaces de usuarios
- 5.5.1. Introducción a Qt
- 5.5.2. Posicionamiento

5.5.3. ¿Qué son los eventos?
5.5.4. Eventos: definición y captura
5.5.5. Desarrollo de interfaces de usuario
5.6. Introducción a la programación concurrente
5.6.1. Introducción a la programación concurrente
5.6.2. El concepto de proceso e hilo
5.6.3. Interacción entre procesos o hilos
5.6.4. Los hilos en C++
5.6.5. Ventajas e inconvenientes de la programación concurrente
5.7. Gestión de hilos y sincronización
5.7.1. Ciclo de vida de un hilo
5.7.2. La clase *Thread*
5.7.3. Planificación de hilos
5.7.4. Grupos hilos
5.7.5. Hilos de tipo demonio
5.7.6. Sincronización
5.7.7. Mecanismos de bloqueo
5.7.8. Mecanismos de Comunicación
5.7.9. Monitores
5.8. Problemas comunes dentro de la programación concurrente
5.8.1. El problema de los productores consumidores
5.8.2. El problema de los lectores y escritores
5.8.3. El problema de la cena de los filósofos
5.9. Documentación y pruebas de Software
5.9.1. ¿Por qué es importante documentar el Software?
5.9.2. Documentación de diseño
5.9.3. Uso de herramientas para la documentación
5.10. Pruebas de Software
5.10.1. Introducción a las pruebas del Software
5.10.2. Tipos de pruebas
5.10.3. Prueba de unidad
5.10.4. Prueba de integración
5.10.5. Prueba de validación
5.10.6. Prueba del sistema

Módulo 6. Informática teórica

6.1. Conceptos matemáticos utilizados
6.1.1. Introducción a la lógica proposicional
6.1.2. Teoría de relaciones
6.1.3. Conjuntos numerables y no numerables
6.2. Lenguajes y gramáticas formales e introducción a las máquinas de Turing
6.2.1. Lenguajes y gramáticas formales
6.2.2. Problema de decisión
6.2.3. La máquina de Turing
6.3. Extensiones para las máquinas de Turing, máquinas de Turing restringidas y computadoras
6.3.1. Técnicas de programación para las máquinas de Turing
6.3.2. Extensiones para las máquinas de Turing
6.3.3. Máquinas de Turing restringidas
6.3.4. Máquinas de Turing y computadoras
6.4. Indecibilidad
6.4.1. Lenguaje no recursivamente enumerable
6.4.2. Un problema indecidible recursivamente enumerable
6.5. Otros problemas indecibles
6.5.1. Problemas indecidibles para las máquinas de Turing
6.5.2. Problema de correspondencia de Post (PCP)
6.6. Problemas intratables
6.6.1. Las clases P y NP
6.6.2. Un problema NP completo
6.6.3. Problema de la satisfacibilidad restringido
6.6.4. Otros problemas NP completos
6.7. Problemas co-NP y PS
6.7.1. Complementarios de los lenguajes de NP
6.7.2. Problemas resolubles en espacio polinómico
6.7.3. Problemas PS completos

6.8. Clases de lenguajes basados en la aleatorización
- 6.8.1. Modelo de la MT con aleatoriedad
- 6.8.2. Las clases RP y ZPP
- 6.8.3. Prueba de primalidad
- 6.8.4. Complejidad de la prueba de primalidad

6.9. Otras clases y gramáticas
- 6.9.1. Autómatas finitos probabilísticos
- 6.9.2. Autómatas celulares
- 6.9.3. Células de McCullogh y Pitts
- 6.9.4. Gramáticas de Lindenmayer

6.10. Sistemas avanzados de cómputo
- 6.10.1. Computación con membranas: sistemas P
- 6.10.2. Computación con ADN
- 6.10.3. Computación cuántica

Módulo 7. Teoría de autómatas y lenguajes formales

7.1. Introducción a la teoría de autómatas
- 7.1.1. ¿Por qué estudiar teoría de autómatas?
- 7.1.2. Introducción a las demostraciones formales
- 7.1.3. Otras formas de demostración
- 7.1.4. Inducción matemática
- 7.1.5. Alfabetos, cadenas y lenguajes

7.2. Autómatas finitos deterministas
- 7.2.1. Introducción a los autómatas finitos
- 7.2.2. Autómatas finitos deterministas

7.3. Autómatas finitos no deterministas
- 7.3.1. Autómatas finitos no deterministas
- 7.3.2. Equivalencia entre AFD y AFN
- 7.3.3. Autómatas finitos con transiciones

7.4. Lenguajes y expresiones regulares (I)
- 7.4.1. Lenguajes y expresiones regulares
- 7.4.2. Autómatas finitos y expresiones regulares

7.5. Lenguajes y expresiones regulares (II)
- 7.5.1. Conversión de expresiones regulares en autómatas
- 7.5.2. Aplicaciones de las expresiones regulares
- 7.5.3. Álgebra de las expresiones regulares

7.6. Lema de bombeo y clausura de los lenguajes regulares
- 7.6.1. Lema de bombeo
- 7.6.2. Propiedades de clausura de los lenguajes regulares

7.7. Equivalencia y minimización de autómatas
- 7.7.1. Equivalencia de AF
- 7.7.2. Minimización de AF

7.8. Gramáticas independientes de contexto (GIC)
- 7.8.1. Gramáticas independientes de contexto
- 7.8.2. Árboles de derivación
- 7.8.3. Aplicaciones de las GIC
- 7.8.4. Ambigüedad en las gramáticas y lenguajes

7.9. Autómatas a pila y GIC
- 7.9.1. Definición de los autómatas a pila
- 7.9.2. Lenguajes aceptados por un autómata a pila
- 7.9.3. Equivalencia entre autómatas a pila y GIC
- 7.9.4. Autómata a pila determinista

7.10. Formas normales, lema de bombeo de las GIC y propiedades de los LIC
- 7.10.1. Formas normales de las GIC
- 7.10.2. Lema de bombeo
- 7.10.3. Propiedades de clausura de los lenguajes
- 7.10.4. Propiedades de decisión de los LIC

Módulo 8. Procesadores de lenguajes

8.1. Introducción al proceso de compilación
- 8.1.1. Compilación e interpretación
- 8.1.2. Entorno de ejecución de un compilador
- 8.1.3. Proceso de análisis
- 8.1.4. Proceso de síntesis

8.2. Analizador léxico
- 8.2.1. ¿Qué es un analizador léxico?
- 8.2.2. Implementación del analizador léxico
- 8.2.3. Acciones semánticas
- 8.2.4. Recuperación de errores
- 8.2.5. Cuestiones de implementación

8.3. Análisis sintáctico
- 8.3.1. ¿Qué es un analizador sintáctico?
- 8.3.2. Conceptos previos
- 8.3.3. Analizadores descendentes
- 8.3.4. Analizadores ascendentes

8.4. Análisis sintáctico descendente y análisis sintáctico ascendente
- 8.4.1. Analizador LL(1)
- 8.4.2. Analizador LR(0)
- 8.4.3. Ejemplo de analizador

8.5. Análisis sintáctico ascendente avanzado
- 8.5.1. Analizador SLR
- 8.5.2. Analizador LR (1)
- 8.5.3. Analizador LR (k)
- 8.5.4. Analizador LALR

8.6. Análisis semántico (I)
- 8.6.1. Traducción dirigida por la sintaxis
- 8.6.2. Tabla de símbolos

8.7. Análisis semántico (II)
- 8.7.1. Comprobación de tipos
- 8.7.2. El subsistema de tipos
- 8.7.3. Equivalencia de tipos y conversiones

8.8. Generación de código y entorno de ejecución
- 8.8.1. Aspectos de diseño
- 8.8.2. Entorno de ejecución
- 8.8.3. Organización de la memoria
- 8.8.4. Asignación de memoria

8.9. Generación de código intermedio
- 8.9.1. Traducción dirigida por la síntesis
- 8.9.2. Representaciones intermedias
- 8.9.3. Ejemplos de traducciones

8.10. Optimización de código
- 8.10.1. Asignación de registros
- 8.10.2. Eliminación de asignaciones muertas
- 8.10.3. Ejecución en tiempo de compilación
- 8.10.4. Reordenación de expresiones
- 8.10.5. Optimización de bucles

Módulo 9. Informática gráfica y visualización

9.1. Teoría del color
- 9.1.1. Propiedades de la luz
- 9.1.2. Modelos de color
- 9.1.3. El estándar CIE
- 9.1.4. *Profiling*

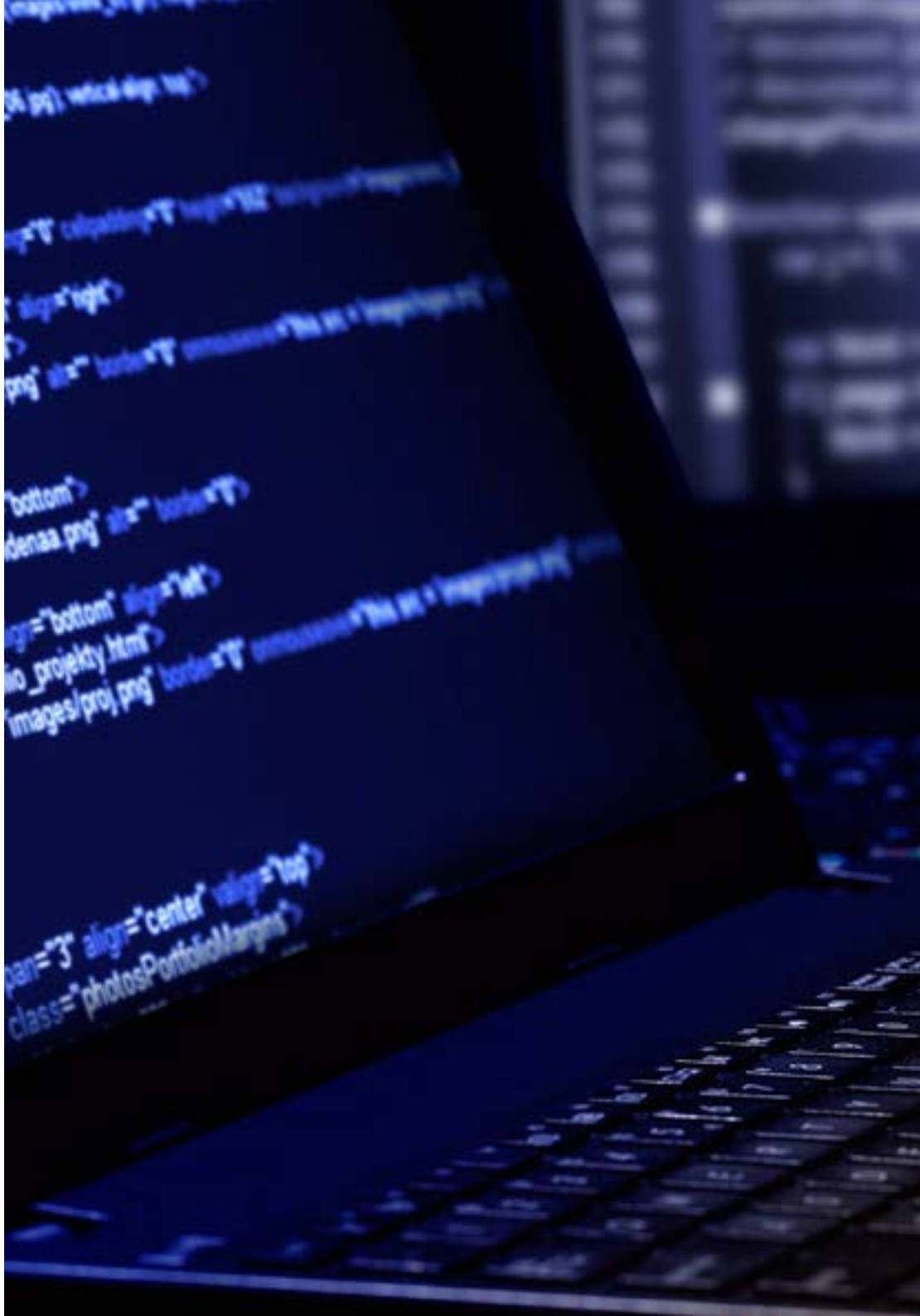
9.2. Primitivas de salida
- 9.2.1. El controlador de vídeo
- 9.2.2. Algoritmos de dibujo de líneas
- 9.2.3. Algoritmos de dibujo de circunferencias
- 9.2.4. Algoritmos de relleno

9.3. Transformaciones 2D y sistemas de coordenadas y recorte 2D
- 9.3.1. Transformaciones geométricas básicas
- 9.3.2. Coordenadas homogéneas
- 9.3.3. Transformación inversa
- 9.3.4. Composición de transformaciones
- 9.3.5. Otras transformaciones
- 9.3.6. Cambio de coordenada
- 9.3.7. Sistemas de coordenadas 2D
- 9.3.8. Cambio de coordenadas
- 9.3.9. Normalización
- 9.3.10. Algoritmos de recorte

9.4. Transformaciones 3D
- 9.4.1. Translación
- 9.4.2. Rotación
- 9.4.3. Escalado
- 9.4.4. Reflexión
- 9.4.5. Cizalla

9.5. Visualización y cambio de coordenadas 3D
- 9.5.1. Sistemas de coordenadas 3D
- 9.5.2. Visualización
- 9.5.3. Cambio de coordenadas
- 9.5.4. Proyección y normalización

9.6. Proyección y recorte 3D
- 9.6.1. Proyección ortogonal
- 9.6.2. Proyección paralela oblicua
- 9.6.3. Proyección perspectiva
- 9.6.4. Algoritmos de recorte 3D



9.7. Eliminación de superficies ocultas
- 9.7.1. *Back face removal*
- 9.7.2. Z-buffer
- 9.7.3. Algoritmo del pintor
- 9.7.4. Algoritmo de Warnock
- 9.7.5. Detección de líneas oculta

9.8. Interpolación y curvas paramétricas
- 9.8.1. Interpolación y aproximación con polinomios
- 9.8.2. Representación paramétrica
- 9.8.3. Polinomio de Lagrange
- 9.8.4. *Splines* cúbicos naturales
- 9.8.5. Funciones base
- 9.8.6. Representación matricial

9.9. Curvas Bézier
- 9.9.1. Construcción algebraica
- 9.9.2. Forma matricial
- 9.9.3. Composición
- 9.9.4. Construcción geométrica
- 9.9.5. Algoritmo de dibujo

9.10. *B-Splines*
- 9.10.1. El problema del control local
- 9.10.2. B-splines cúbicos uniformes
- 9.10.3. Funciones base y puntos de control
- 9.10.4. Deriva al origen y multiplicidad
- 9.10.5. Representación matricial
- 9.10.6. *B-Splines* no uniformes

Módulo 10. Computación bioinspirada

10.1. Introducción a la Computación bioinspirada
- 10.1.1. Introducción a la Computación bioinspirada

10.2. Algoritmos de adaptación social
- 10.2.1. Computación bioinspirada basada en colonia de hormigas
- 10.2.2. Variantes de los algoritmos de colonias de hormigas
- 10.2.3. Computación basada en nubes de partículas

10.3. Algoritmos genéticos
- 10.3.1. Estructura general
- 10.3.2. Implementaciones de los principales operadores

10.4. Estrategias de exploración-explotación del espacio para algoritmos genéticos
- 10.4.1. Algoritmo CHC
- 10.4.2. Problemas multimodales

10.5. Modelos de Computación evolutiva (I)
- 10.5.1. Estrategias evolutivas
- 10.5.2. Programación evolutiva
- 10.5.3. Algoritmos basados en evolución diferencial

10.6. Modelos de Computación evolutiva (II)
- 10.6.1. Modelos de evolución basados en estimación de distribuciones (EDA)
- 10.6.2. Programación genética

10.7. Programación evolutiva aplicada a problemas de aprendizaje
- 10.7.1. Aprendizaje basado en reglas
- 10.7.2. Métodos evolutivos en problemas de selección de instancias

10.8. Problemas multiobjetivo
- 10.8.1. Concepto de dominancia
- 10.8.2. Aplicación de algoritmos evolutivos a problemas multiobjetivo

10.9. Redes neuronales (I)
- 10.9.1. Introducción a las redes neuronales
- 10.9.2. Ejemplo práctico con redes neuronales

10.10. Redes neuronales (II)
- 10.10.1. Casos de uso de las redes neuronales en la investigación médica
- 10.10.2. Casos de uso de las redes neuronales en la economía
- 10.10.3. Casos de uso de las redes neuronales en la visión artificial

05

Metodología

Este programa de capacitación ofrece una forma diferente de aprender. Nuestra metodología se desarrolla a través de un modo de aprendizaje de forma cíclica: ***el Relearning***.

Este sistema de enseñanza es utilizado, por ejemplo, en las facultades de medicina más prestigiosas del mundo y se ha considerado uno de los más eficaces por publicaciones de gran relevancia como el ***New England Journal of Medicine.***





Descubre el Relearning, un sistema que abandona el aprendizaje lineal convencional para llevarte a través de sistemas cíclicos de enseñanza: una forma de aprender que ha demostrado su enorme eficacia, especialmente en las materias que requieren memorización"

Estudio de Caso para contextualizar todo el contenido

Nuestro programa ofrece un método revolucionario de desarrollo de habilidades y conocimientos. Nuestro objetivo es afianzar competencias en un contexto cambiante, competitivo y de alta exigencia.

Con TECH podrás experimentar una forma de aprender que está moviendo los cimientos de las universidades tradicionales de todo el mundo"



Accederás a un sistema de aprendizaje basado en la reiteración, con una enseñanza natural y progresiva a lo largo de todo el temario.



El alumno aprenderá, mediante actividades colaborativas y casos reales, la resolución de situaciones complejas en entornos empresariales reales.

Un método de aprendizaje innovador y diferente

El presente programa de TECH es una enseñanza intensiva, creada desde 0, que propone los retos y decisiones más exigentes en este campo, ya sea en el ámbito nacional o internacional. Gracias a esta metodología se impulsa el crecimiento personal y profesional, dando un paso decisivo para conseguir el éxito. El método del caso, técnica que sienta las bases de este contenido, garantiza que se sigue la realidad económica, social y profesional más vigente.

> *Nuestro programa te prepara para afrontar nuevos retos en entornos inciertos y lograr el éxito en tu carrera"*

El método del caso ha sido el sistema de aprendizaje más utilizado por las mejores escuelas de Informática del mundo desde que éstas existen. Desarrollado en 1912 para que los estudiantes de Derecho no solo aprendiesen las leyes a base de contenidos teóricos, el método del caso consistió en presentarles situaciones complejas reales para que tomasen decisiones y emitiesen juicios de valor fundamentados sobre cómo resolverlas. En 1924 se estableció como método estándar de enseñanza en Harvard.

Ante una determinada situación, ¿qué debería hacer un profesional? Esta es la pregunta a la que te enfrentamos en el método del caso, un método de aprendizaje orientado a la acción. A lo largo del curso, los estudiantes se enfrentarán a múltiples casos reales. Deberán integrar todos sus conocimientos, investigar, argumentar y defender sus ideas y decisiones.

Relearning Methodology

TECH aúna de forma eficaz la metodología del Estudio de Caso con un sistema de aprendizaje 100% online basado en la reiteración, que combina elementos didácticos diferentes en cada lección.

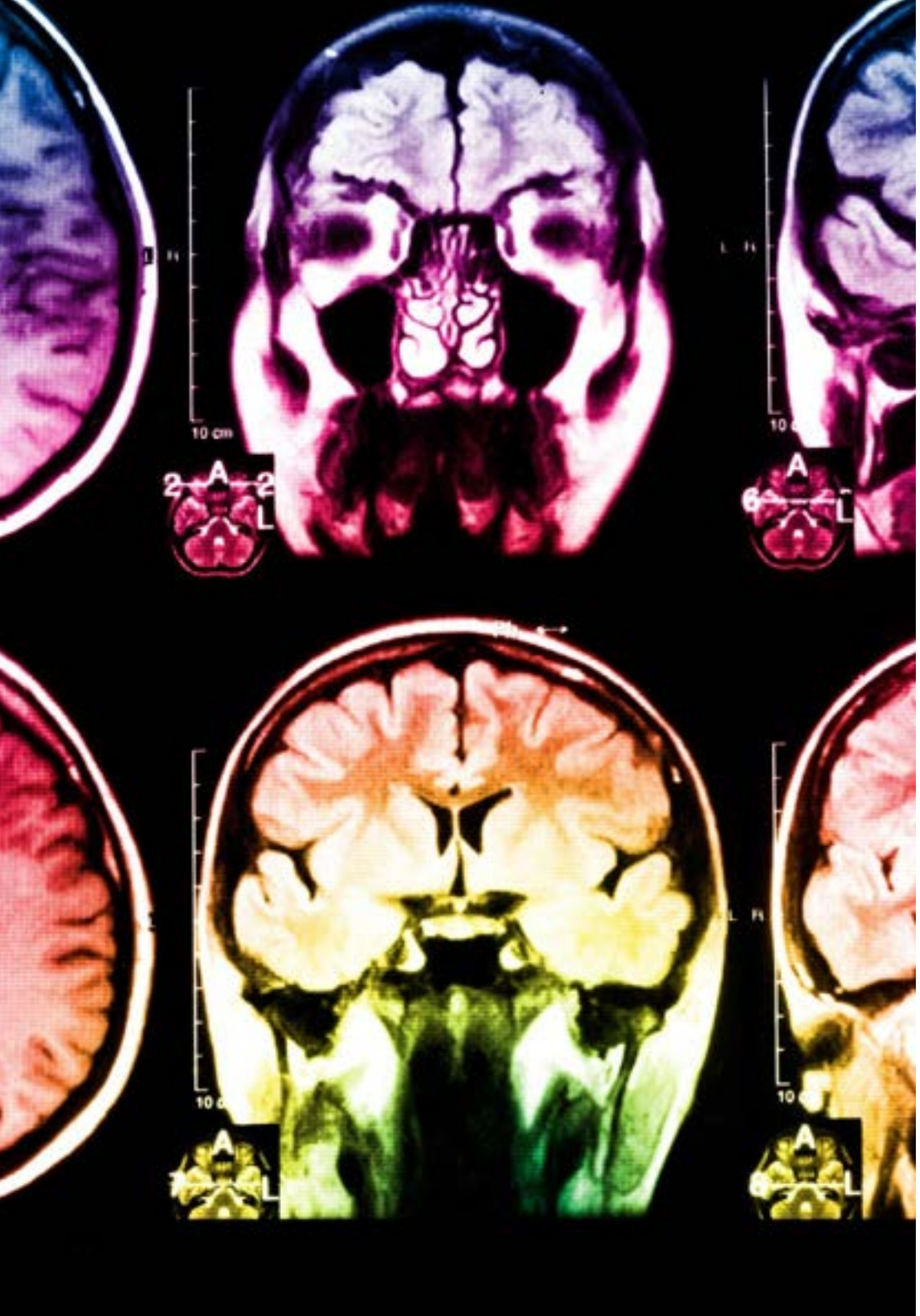
Potenciamos el Estudio de Caso con el mejor método de enseñanza 100% online: el Relearning.

En 2019 obtuvimos los mejores resultados de aprendizaje de todas las universidades online en español en el mundo.

En TECH aprenderás con una metodología vanguardista concebida para capacitar a los directivos del futuro. Este método, a la vanguardia pedagógica mundial, se denomina Relearning.

Nuestra universidad es la única en habla hispana licenciada para emplear este exitoso método. En 2019, conseguimos mejorar los niveles de satisfacción global de nuestros alumnos (calidad docente, calidad de los materiales, estructura del curso, objetivos…) con respecto a los indicadores de la mejor universidad online en español.





En nuestro programa, el aprendizaje no es un proceso lineal, sino que sucede en espiral (aprender, desaprender, olvidar y reaprender). Por eso, se combinan cada uno de estos elementos de forma concéntrica. Con esta metodología se han capacitado más de 650.000 graduados universitarios con un éxito sin precedentes en ámbitos tan distintos como la bioquímica, la genética, la cirugía, el derecho internacional, las habilidades directivas, las ciencias del deporte, la filosofía, el derecho, la ingeniería, el periodismo, la historia o los mercados e instrumentos financieros. Todo ello en un entorno de alta exigencia, con un alumnado universitario de un perfil socioeconómico alto y una media de edad de 43,5 años.

El Relearning te permitirá aprender con menos esfuerzo y más rendimiento, implicándote más en tu capacitación, desarrollando el espíritu crítico, la defensa de argumentos y el contraste de opiniones: una ecuación directa al éxito.

A partir de la última evidencia científica en el ámbito de la neurociencia, no solo sabemos organizar la información, las ideas, las imágenes y los recuerdos, sino que sabemos que el lugar y el contexto donde hemos aprendido algo es fundamental para que seamos capaces de recordarlo y almacenarlo en el hipocampo, para retenerlo en nuestra memoria a largo plazo.

De esta manera, y en lo que se denomina Neurocognitive context-dependent e-learning, los diferentes elementos de nuestro programa están conectados con el contexto donde el participante desarrolla su práctica profesional.

Este programa ofrece los mejores materiales educativos, preparados a conciencia para los profesionales:

Material de estudio

Todos los contenidos didácticos son creados por los especialistas que van a impartir el curso, específicamente para él, de manera que el desarrollo didáctico sea realmente específico y concreto.

Estos contenidos son aplicados después al formato audiovisual, para crear el método de trabajo online de TECH. Todo ello, con las técnicas más novedosas que ofrecen piezas de gran calidad en todos y cada uno los materiales que se ponen a disposición del alumno.

Clases magistrales

Existe evidencia científica sobre la utilidad de la observación de terceros expertos.

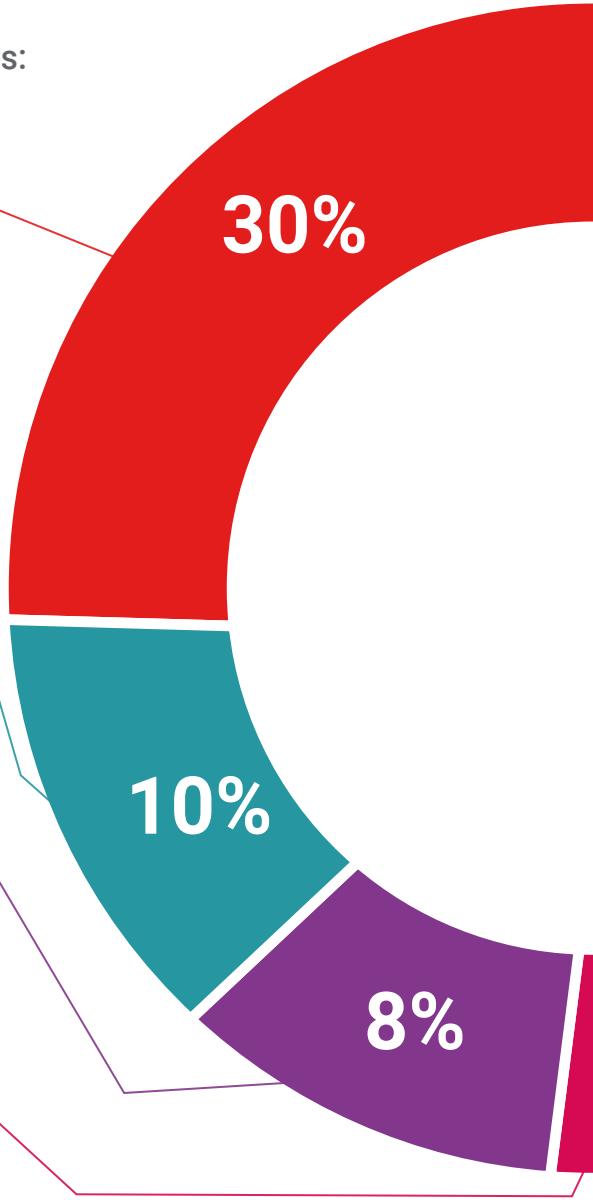
El denominado Learning from an Expert afianza el conocimiento y el recuerdo, y genera seguridad en las futuras decisiones difíciles.

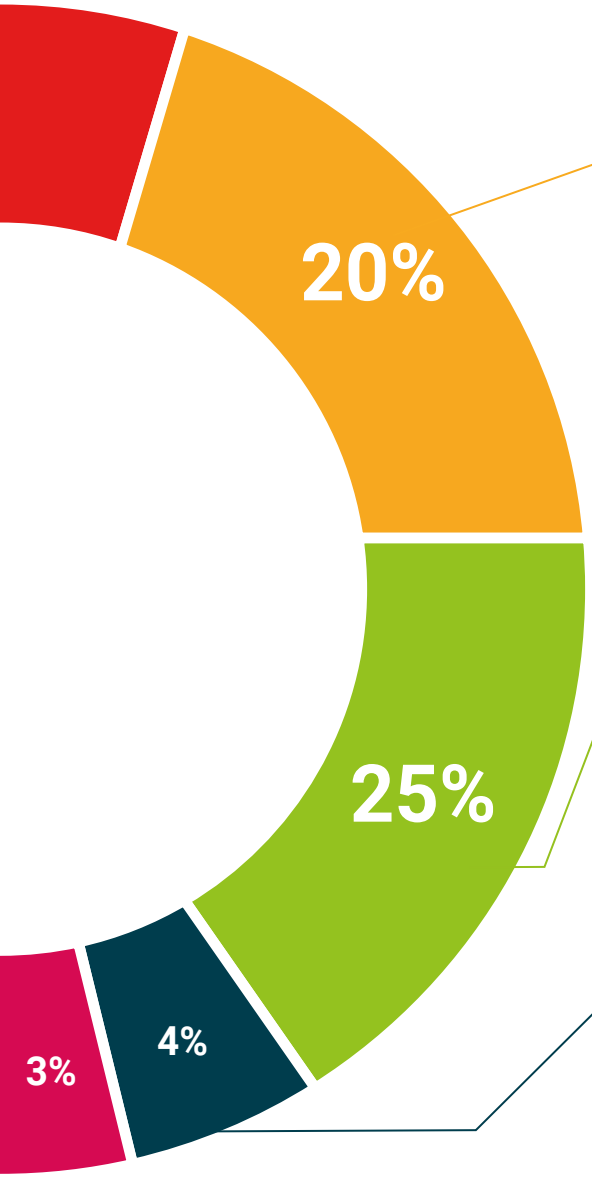
Prácticas de habilidades y competencias

Realizarán actividades de desarrollo de competencias y habilidades específicas en cada área temática. Prácticas y dinámicas para adquirir y desarrollar las destrezas y habilidades que un especialista precisa desarrollar en el marco de la globalización que vivimos.

Lecturas complementarias

Artículos recientes, documentos de consenso y guías internacionales, entre otros. En la biblioteca virtual de TECH el estudiante tendrá acceso a todo lo que necesita para completar su capacitación.





Case studies

Completarán una selección de los mejores casos de estudio elegidos expresamente para esta titulación. Casos presentados, analizados y tutorizados por los mejores especialistas del panorama internacional.

Resúmenes interactivos

El equipo de TECH presenta los contenidos de manera atractiva y dinámica en píldoras multimedia que incluyen audios, vídeos, imágenes, esquemas y mapas conceptuales con el fin de afianzar el conocimiento.

Este exclusivo sistema educativo para la presentación de contenidos multimedia fue premiado por Microsoft como "Caso de éxito en Europa".

Testing & Retesting

Se evalúan y reevalúan periódicamente los conocimientos del alumno a lo largo del programa, mediante actividades y ejercicios evaluativos y autoevaluativos para que, de esta manera, el estudiante compruebe cómo va consiguiendo sus metas.

06

Titulación

El Máster Título Propio en Computación y Lenguajes garantiza, además de la capacitación más rigurosa y actualizada, el acceso a un título de Máster Propio expedido por TECH Universidad Tecnológica.





Supera con éxito este programa y recibe tu titulación universitaria sin desplazamientos ni farragosos trámites”

Este **Máster Título Propio en Computación y Lenguajes** contiene el programa más completo y actualizado del mercado.

Tras la superación de la evaluación, el alumno recibirá por correo postal* con acuse de recibo su correspondiente título de **Máster Propio** emitido por **TECH Universidad Tecnológica**.

El título expedido por **TECH Universidad Tecnológica** expresará la calificación que haya obtenido en el Máster Título Propio, y reunirá los requisitos comúnmente exigidos por las bolsas de trabajo, oposiciones y comités evaluadores de carreras profesionales.

Título: **Máster Título Propio en Computación y Lenguajes**

N.º Horas Oficiales: **1.500 h.**


tech universidad tecnológica

Otorga la presente

CONSTANCIA

a

C. _____, con documento de identificación nº_____

Por haber superado con éxito y acreditado el programa de

MÁSTER TÍTULO PROPIO

en

Computación y Lenguajes

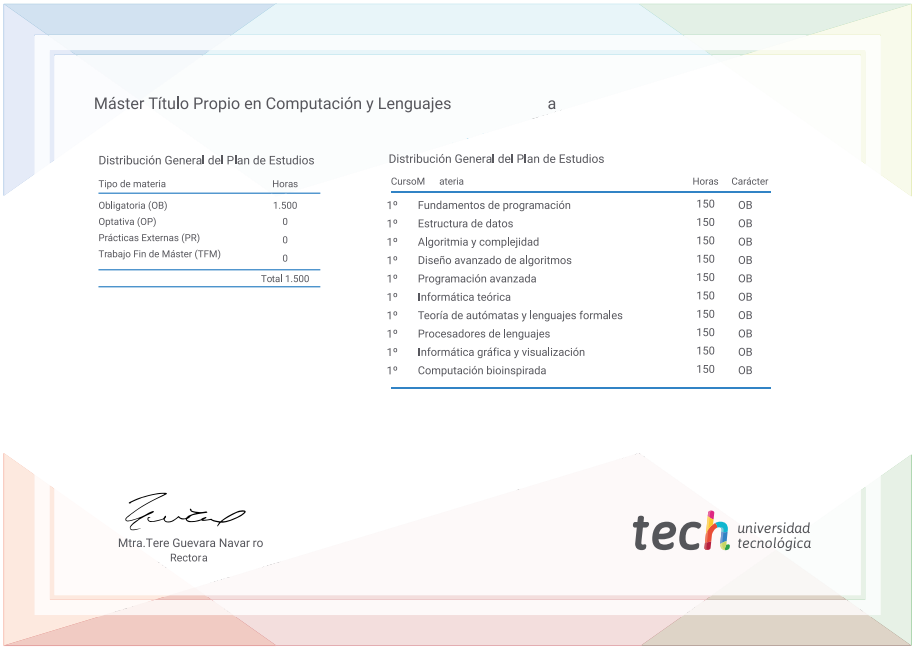
Se trata de un título propio de esta Universidad con una duración de 1.500 horas, con fecha de inicio dd/mm/aaaa y fecha de finalización dd/mm/aaaa.

TECH es una Institución Particular de Educación Superior reconocida por la Secretaría de Educación Pública a partir del 28 de junio de 2018.

A 17 de junio de 2020

Mtra.Tere Guevara Navarro
Rectora

código único TECH: AFWOR23S techtitute.com/titulos


Máster Título Propio en Computación y Lenguajes a

Distribución General del Plan de Estudios

Tipo de materia	Horas
Obligatoria (OB)	1.500
Optativa (OP)	0
Prácticas Externas (PR)	0
Trabajo Fin de Máster (TFM)	0
	Total 1.500

Distribución General del Plan de Estudios

CursoM	ateria	Horas	Carácter
1º	Fundamentos de programación	150	OB
1º	Estructura de datos	150	OB
1º	Algoritmia y complejidad	150	OB
1º	Diseño avanzado de algoritmos	150	OB
1º	Programación avanzada	150	OB
1º	Informática teórica	150	OB
1º	Teoría de autómatas y lenguajes formales	150	OB
1º	Procesadores de lenguajes	150	OB
1º	Informática gráfica y visualización	150	OB
1º	Computación bioinspirada	150	OB

Mtra.Tere Guevara Navar ro
Rectora

tech universidad tecnológica

*Apostilla de La Haya. En caso de que el alumno solicite que su título en papel recabe la Apostilla de La Haya, TECH EDUCATION realizará las gestiones oportunas para su obtención, con un coste adicional.

tech universidad tecnológica

Máster Título Propio
Computación y Lenguajes

Modalidad: **Online**
Duración: **12 meses**
Titulación: **TECH Universidad Tecnológica**
Horas lectivas: **1.500 h.**

Máster Título Propio
Computación y Lenguajes



tech universidad tecnológica