Master Privato

Informatica e Linguaggi di Programmazione

tech università tecnologica

tech università tecnologica

Master Privato

Informatica e Linguaggi di Programmazione

Modalità: **Online**
Durata: **12 mesi**
Titolo: **TECH Università Tecnologica**
Ore teoriche: **1.500**
Accesso al sito web: **www.techtitute.com/it/informatica/master/master-informatica-linguaggi-programmazione**

Indice

01

Presentazione

L'informatico professionista deve mantenere aggiornate le sue competenze per poter continuare a operare in modo ottimale, senza perdere nessuno dei progressi che vengono incorporati in questo settore a un ritmo vertiginoso. Questo aggiornamento è pensato per fornire agli studenti una conoscenza completa e approfondita delle nozioni essenziali e delle innovazioni più interessanti nella progettazione di algoritmi per lo sviluppo di progetti informatici, utilizzando i metodi più innovativi ed efficienti del settore.

Acquisisci le conoscenze fondamentali sull'informatica e su come applicarle con successo nello sviluppo di progetti IT grazie a un Master Privato di alto livello"

Questo Master Privato si concentra sui fondamenti della programmazione e della struttura dei dati, sulla progettazione avanzata di algoritmi e sulla complessità, nonché sui processori di linguaggio e sulla grafica computerizzata, tra gli altri aspetti legati a questo campo dell'informatica.

Questo Master Privato fornisce allo studente strumenti e competenze specifiche per svolgere con successo la sua attività professionale nell'ampia area dell'Informatica e dei Linguaggi di Programmazione. Acquisisci competenze chiave, come le conoscenze sulla realtà e sulla pratica quotidiana nelle diverse aree informatiche, e sviluppa responsabilità nel monitoraggio e nella supervisione del lavoro, così come abilità specifiche imprescindibili in questo campo.

Data la modalità 100% online di questo Master Privato, lo studente non è condizionato da orari fissi o dalla necessità di recarsi in un luogo fisico, ma può accedere ai contenuti in qualsiasi momento della giornata, combinando il suo lavoro o la sua vita personale con quella accademica.

Il personale docente del Master Privato in Informatica e Linguaggi di Programmazione ha selezionato attentamente ognuna delle materie impartite durante questa specializzazione, per offrire allo studente un'opportunità di studio il più completa possibile e sempre attuale.

Questo **Master Privato in Informatica e Linguaggi di Programmazione** possiede il programma più completo e aggiornato del mercato. Le caratteristiche principali del corso sono:

- Sviluppo di casi pratici presentati da esperti in Informatica e Linguaggi di Programmazione
- Contenuti grafici, schematici ed eminentemente pratici che forniscono informazioni scientifiche e pratiche sulle discipline essenziali per l'esercizio della professione
- Esercizi pratici che offrono un processo di autovalutazione per migliorare l'apprendimento
- Speciale enfasi sulle metodologie innovative in Informatica e Linguaggi di Programmazione
- Lezioni teoriche, domande all'esperto, forum di discussione su questioni controverse e compiti di riflessione individuale
- Contenuti disponibili da qualsiasi dispositivo fisso o mobile dotato di connessione a internet

Un'opportunità eccezionale per apprendere in modo comodo e semplice i processi e le conoscenze matematiche di base necessarie per realizzare una programmazione informatica di qualità"

Un Master Privato che basa la sua efficacia sulle tecnologie didattiche più apprezzate del mercato, supportate da sistemi audiovisivi e di studio che ti permetteranno di apprendere più velocemente e comodamente"

I contenuti multimediali, sviluppati in base alle ultime tecnologie educative, forniranno al professionista un apprendimento coinvolgente e localizzato, ovvero inserito in un contesto reale.

La creazione di questo programma è incentrata sull'Apprendimento Basato su Problemi, mediante il quale lo specialista deve cercare di risolvere le diverse situazioni che gli si presentano durante il corso. Lo studente potrà usufruire di un innovativo sistema di video interattivi creati da esperti di rinomata fama.

Mettiamo al tuo servizio un materiale didattico ampio e chiaro, che incorpora tutti gli argomenti attuali di interesse per il professionista che vuole progredire nel campo dell'Informatica e dei Linguaggi di Programmazione.

Uno studio che possiede un alto impatto specializzante e che ti permetterà di adattare l'impegno accademico alle tue esigenze, combinando flessibilità e intensità.

02

Obiettivi

Il Master Privato in Informatica e Linguaggi di Programmazione è stato creato appositamente per il professionista che voglia progredire in questo campo in modo rapido e con qualità reale, sulla base di obiettivi realistici e di alto valore che lo proietteranno a un livelloprofessionale superiore in questo settore.

Il nostro obiettivo è fornire ai professionisti del settore informatico un aggiornamento di alta qualità che permetta loro di intervenire con competenza in materia di Informatica e Linguaggi di Programmazione"

Obiettivo generale

- Preparare scientificamente e tecnologicamente, nonché arricchire la pratica professionale dell'Informatica e dei Linguaggi di Programmazione, il tutto con una qualifica trasversale e versatile adattata alle nuove tecnologie e alle innovazioni del settore

Cogli l'opportunità e aggiornati sulle ultime novità in Informatica e Linguaggi di Programmazione"

Obiettivi specifici

Modulo 1. Fondamenti di programmazione

- Comprendere la struttura di base di un computer, il software e i linguaggi di programmazione di uso generale
- Imparare a progettare e interpretare gli algoritmi, che sono la base necessaria per lo sviluppo del software
- Comprendere gli elementi essenziali di un programma informatico, come i diversi tipi di dati, gli operatori, le espressioni, le dichiarazioni, le istruzioni di I/O e di controllo
- Comprendere le diverse strutture di dati disponibili nei linguaggi di programmazione generici, sia statici che dinamici, e acquisire competenze essenziali nella gestione dei file
- Comprendere le diverse tecniche di test del software e l'importanza di generare una buona documentazione insieme a un buon codice sorgente
- Imparare le basi del linguaggio di programmazione C++, uno dei più utilizzati al mondo

Modulo 2. Struttura dati

- Imparare le basi della programmazione in linguaggio C++, comprese classi, variabili, espressioni condizionali e oggetti
- Comprendere i tipi di dati astratti, i tipi di strutture dati lineari, le strutture dati gerarchiche semplici e complesse e la loro implementazione in C++
- Comprendere il funzionamento di strutture dati avanzate diverse da quelle abituali
- Comprendere la teoria e la pratica relative all'uso di heap e code prioritarie
- Imparare come funzionano le tabelle hash come tipi di dati astratti e funzioni
- Comprendere la teoria dei grafi e dei loro concetti avanzati, nonché degli algoritmi

Modulo 3. Algoritmi e complessità

- Imparare le principali strategie per la progettazione di algoritmi, nonché i diversi metodi e misure per il loro calcolo
- Conoscere i principali algoritmi di ordinamento utilizzati nello sviluppo del software
- Capire come funzionano i diversi algoritmi su alberi, *heap* e grafi
- Comprendere il funzionamento degli algoritmi *Greedy*, la loro strategia e gli esempi del loro utilizzo nei principali problemi noti
- Conoscere anche l'uso degli algoritmi *Greedy* sui grafi
- Imparare le principali strategie di ricerca dei percorsi minimi, con l'approccio ai problemi essenziali del campo e agli algoritmi per la loro risoluzione
- Comprendere la tecnica del *Backtracking* e i suoi principali utilizzi, nonché le tecniche alternative

Modulo 4. Progettazione avanzata di algoritmi

- Approfondire la progettazione di algoritmi avanzati, analizzando algoritmi ricorsivi e tipo divide et impera, nonché eseguendo analisi ammortizzate
- Comprendere i concetti di programmazione dinamica e gli algoritmi per i problemi NP
- Comprendere il funzionamento dell'ottimizzazione combinatoria, i diversi algoritmi randomizzati e gli algoritmi paralleli
- Conoscere e capire come funzionano i diversi metodi di ricerca locale e con candidati
- Imparare i meccanismi della verifica formale dei programmi e di programmi iterativi, compresa la logica del primo ordine e il sistema formale di Hoare
- Imparare il funzionamento di alcuni dei principali metodi numerici come il metodo di bisezione, il metodo di Newton Raphson e il metodo della secante

Modulo 5. Programmazione avanzata

- Approfondire la conoscenza della programmazione, soprattutto per quanto riguarda la programmazione orientata agli oggetti, e i diversi tipi di relazioni tra le classi esistenti
- Conoscere i diversi modelli di progettazione per i problemi orientati agli oggetti
- Imparare a conoscere la programmazione guidata dagli eventi e lo sviluppo dell'interfaccia utente con Qt
- Acquisire le conoscenze essenziali della programmazione concorrente, dei processi e dei thread
- Imparare a gestire l'uso dei thread e della sincronizzazione, nonché a risolvere i problemi più comuni della programmazione concorrente
- Comprendere l'importanza della documentazione e dei test nello sviluppo del software

Modulo 6. Informatica teorica

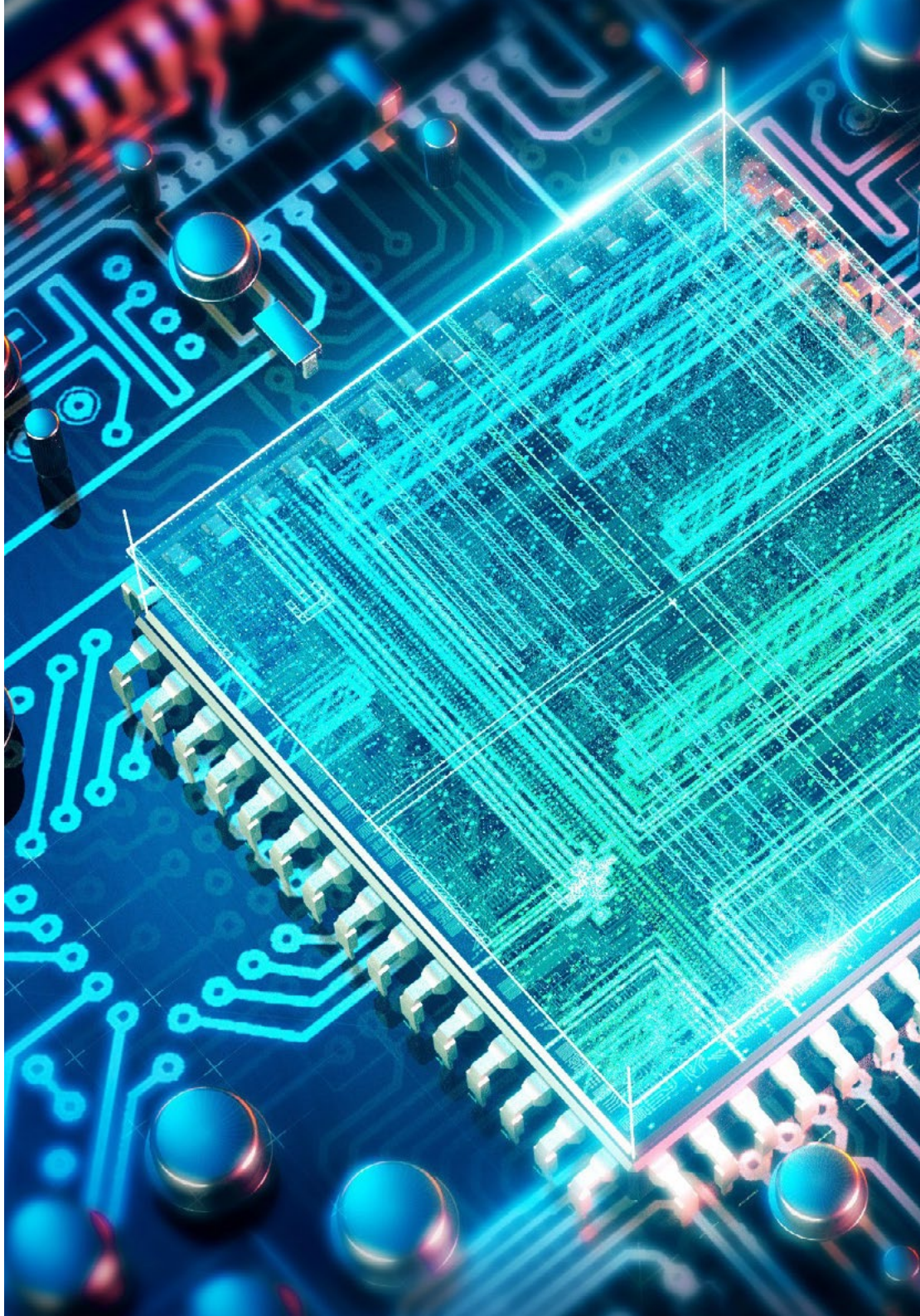
- Comprendere i concetti matematici teorici essenziali alla base dell'informatica, come la logica propositiva, la teoria degli insiemi e gli insiemi numerabili e non numerabili
- Comprendere i concetti di linguaggi formali e grammatiche, nonché di macchine di Turing nelle loro diverse varianti
- Imparare a conoscere i diversi tipi di problemi indecidibili e intrattabili, comprese le diverse varianti di questi e i loro approcci
- Comprendere il funzionamento di diversi tipi di linguaggi basati sulla randomizzazione e di altri tipi di classi e grammatiche
- Conoscere altri sistemi di calcolo avanzati come il Membrane Computing, il DNA Computing e il Quantum Computing

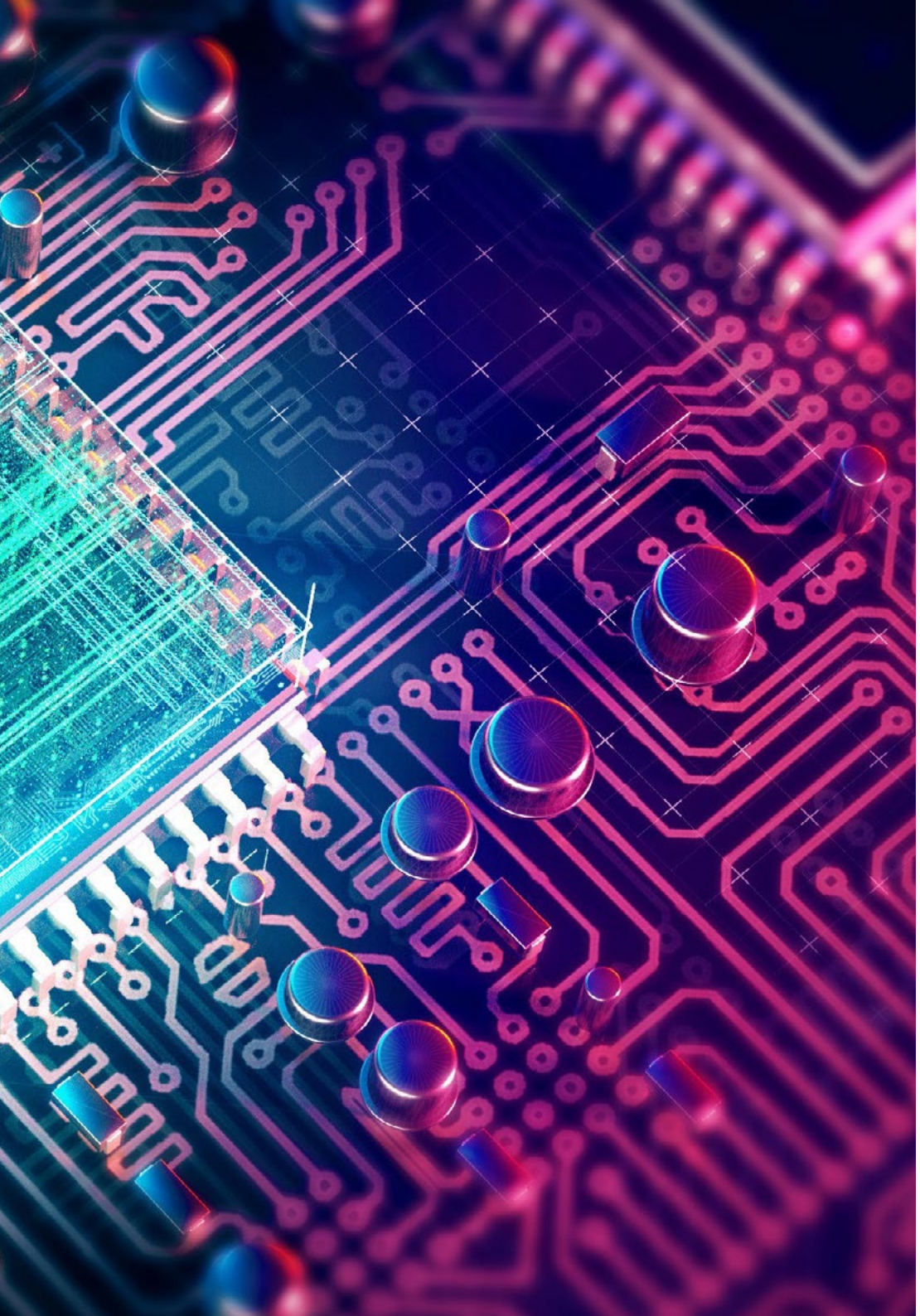
Modulo 7. Teoria degli automi e linguaggi formali

- Comprendere la teoria degli automi e dei linguaggi formali, imparando i concetti di alfabeti, stringhe e linguaggi, nonché a eseguire dimostrazioni formali
- Approfondire la comprensione dei diversi tipi di automi finiti, deterministici o non
- Imparare i concetti di base e avanzati relativi ai linguaggi e alle espressioni regolari, nonché l'applicazione del lemma di pompaggio e la chiusura dei linguaggi regolari
- Comprendere le grammatiche indipendenti dal contesto e il funzionamento degli automi a pila
- Approfondire le forme normali, il lemma di pompaggio delle grammatiche indipendenti dal contesto e le proprietà dei linguaggi indipendenti dal contesto

Modulo 8. Processori linguistici

- Introdurre i concetti relativi al processo di compilazione e ai diversi tipi di analisi: lessicale, sintattica e semantica
- Conoscere il funzionamento di un analizzatore lessicale, la sua implementazione e il recupero degli errori
- Approfondire la conoscenza dell'analisi sintattica, sia top-down che bottom-up, ma con particolare attenzione ai diversi tipi di parser bottom-up
- Comprendere il funzionamento dei parser semantici, la tradizione sintattica, la tabella dei simboli e i vari tipi di parser
- Imparare i diversi meccanismi per la generazione di codice, sia in ambienti runtime che per la generazione di codice intermedio
- Porre le basi dell'ottimizzazione del codice, compreso il riordino delle espressioni e l'ottimizzazione dei cicli





Modulo 9. Computer grafica e visualizzazione

- Introdurre i concetti essenziali della computer grafica e della sua visualizzazione come la teoria del colore e i suoi modelli e le proprietà della luce
- Comprendere il funzionamento delle primitive di output e dei loro algoritmi, sia per il disegno di linee che per il disegno di cerchi e riempimenti
- Studiare in modo approfondito le diverse trasformazioni 2D e 3D, dei sistemi di coordinate e della visualizzazione al computer
- Imparare a realizzare proiezioni e tagli 3D e a rimuovere le superfici nascoste
- Imparare la teoria relativa all'interpolazione e alle curve parametriche, nonché alle curve di Bézier e alle B-spline

Modulo 10. Informatica bio-ispirata

- Introdurre il concetto di Informatica bio-ispirata e comprendere il funzionamento di diversi tipi di algoritmi di adattamento sociale e di algoritmi genetici
- Approfondire lo studio dei diversi modelli di calcolo evolutivo, conoscendone le strategie, la programmazione, gli algoritmi e i modelli basati sulla stima delle distribuzioni
- Comprendere le principali strategie di esplorazione e sfruttamento dello spazio per gli algoritmi genetici
- Comprendere il funzionamento della programmazione evolutiva applicata a problemi di apprendimento e a problemi multi-obiettivo
- Imparare i concetti essenziali relativi alle reti neurali e capire come funzionano in casi d'uso reali applicati ad aree diverse come la ricerca medica, l'economia e la visione artificiale

03

Competenze

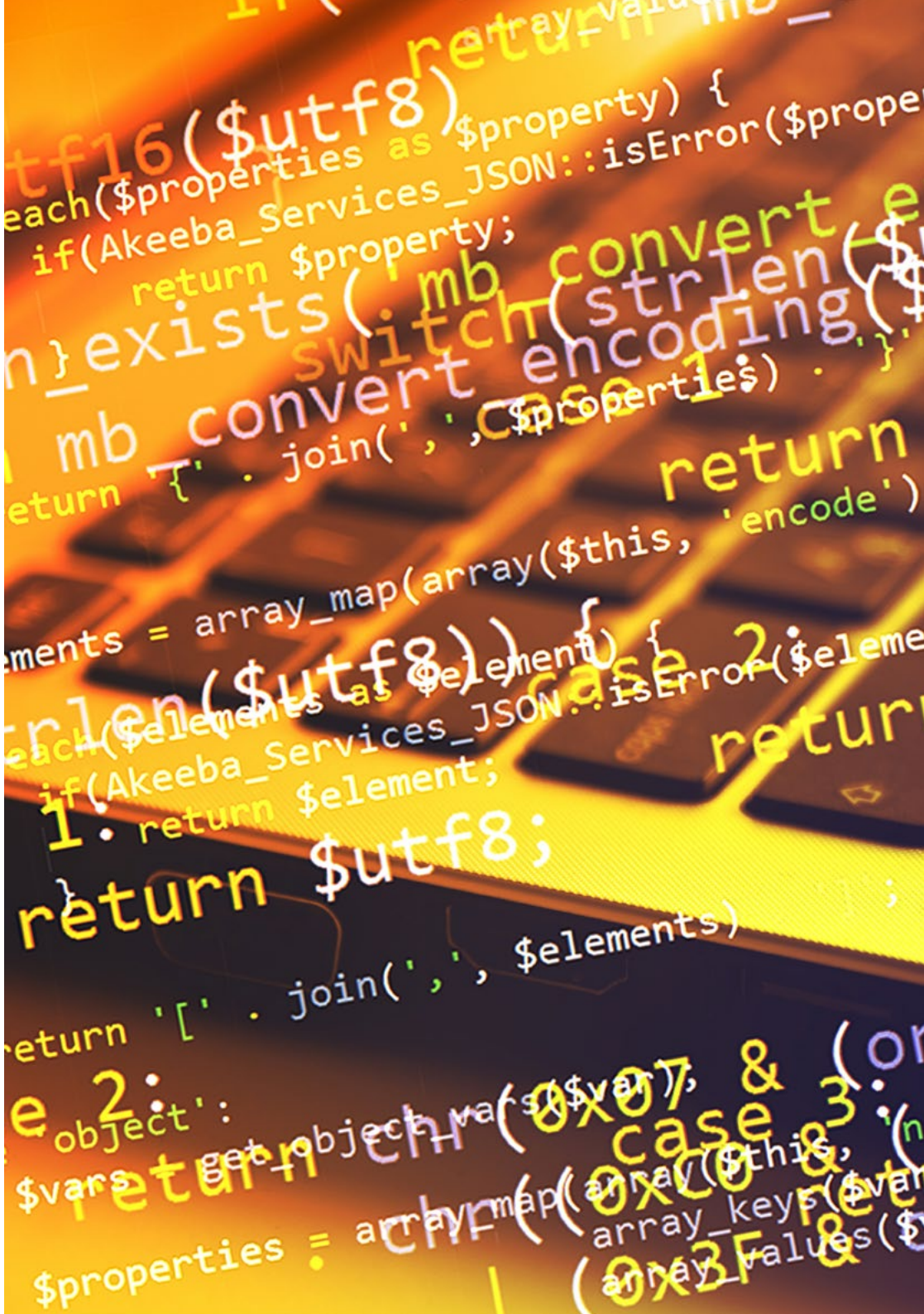
Dopo aver superato le valutazioni del Master Privato in Informatica e Linguaggi di Programmazione, il professionista avrà acquisito le competenze necessarie per conoscere i principi fondamentali dell'informatica e possiederà la capacità di lavorare con linguaggi di programmazione e dati.

Acquisisci la capacità di realizzare nuovi sviluppi informatici partendo dalla comprensione e dal controllo dei diversi linguaggi di programmazione e gli algoritmi e dalla loro applicazione pratica"

Competenza generale

- Svolgere correttamente il lavoro relativo all'Informatica e al Linguaggio di Programmazione

Migliora le tue competenze per essere in grado di partecipare a vari progetti tecnologici"



Competenze specifiche

- Progettare algoritmi per sviluppare programmi informatici e applicare il linguaggio di programmazione
- Comprendere e utilizzare la struttura dei dati informatici
- Utilizzare gli algoritmi necessari per risolvere i problemi informatici
- Conoscere in profondità la progettazione di algoritmi avanzati e i metodi di ricerca
- Eseguire lavori di programmazione informatica
- Comprendere e applicare la teoria alla base dell'informatica, come per esempio la matematica
- Conoscere la teoria degli automi e applicare il linguaggio informatico
- Conoscere le basi teoriche dei linguaggi di programmazione e le relative tecniche di elaborazione lessicale, sintattica e semantica
- Comprendere i concetti di base della matematica e della complessità computazionale per applicarli alla risoluzione di problemi informatici
- Conoscere e applicare i principi fondamentali della computazione per realizzare nuovi sviluppi informatici

04

Struttura e contenuti

La struttura dei contenuti è stata creata in modo tale che le conoscenze vengano assimilate progressivamente, realizzando un percorso di crescita che ti porterà all'eccellenza nella tua professione.

Conosci tutte le aree di interesse per lavorare con successo e in modo sicuro nel campo dell'Informatica e dei Linguaggi di Programmazione, riunite in un programma di alta qualità"

Modulo 1. Fondamenti di programmazione

1.1. Introduzione alla programmazione
- 1.1.1. Struttura base di un computer
- 1.1.2. Software
- 1.1.3. Linguaggi di programmazione
- 1.1.4. Ciclo di vita di un'applicazione informatica

1.2. Progettazione di algoritmi
- 1.2.1. Risoluzione dei problemi
- 1.2.2. Tecniche descrittive
- 1.2.3. Elementi e struttura di un algoritmo

1.3. Elementi di un programma
- 1.3.1. Origine e caratteristiche del linguaggio C++
- 1.3.2. Ambienti di sviluppo
- 1.3.3. Concetto di programma
- 1.3.4. Tipi di dati fondamentali
- 1.3.5. Operatori
- 1.3.6. Espressioni
- 1.3.7. Strutture
- 1.3.8. Ingresso e uscita di dati

1.4. Strutture di controllo
- 1.4.1. Sentenze
- 1.4.2. Biforcazioni
- 1.4.3. Loop

1.5. Astrazione e modularità: le funzioni
- 1.5.1. Progettazione modulare
- 1.5.2. Concetto di funzione e utilità
- 1.5.3. Definizione di una funzione
- 1.5.4. Flusso di esecuzione in una chiamata di funzione
- 1.5.5. Prototipo di una funzione
- 1.5.6. Restituzione dei risultati
- 1.5.7. Chiamata di una funzione: parametri
- 1.5.8. Passaggio di parametri per riferimento e per valore
- 1.5.9. Ambito di identificazione

1.6. Strutture dati statiche
- 1.6.1. *Array*
- 1.6.2. Matrici. Poliedri
- 1.6.3. Ricerca e ordinamento
- 1.6.4. Stringhe. Funzioni di I/O per le stringhe
- 1.6.5. Strutture. Unioni
- 1.6.6. Nuovi tipi di dati

1.7. Strutture dati dinamiche: puntatori
- 1.7.1. Concetto. Definizione di puntatore
- 1.7.2. Operatori e operazioni con i puntatori
- 1.7.3. *Array* di puntatori
- 1.7.4. Puntatori e *Array*
- 1.7.5. Puntatori a stringhe
- 1.7.6. Puntatori a strutture
- 1.7.7. Indirezione multipla
- 1.7.8. Puntatori a funzioni
- 1.7.9. Passaggio di funzioni, strutture e *Array* come parametri di funzione

1.8. File
- 1.8.1. Concetti di base
- 1.8.2. Operazioni sui file
- 1.8.3. Tipi di file
- 1.8.4. Organizzazione dei file
- 1.8.5. Introduzione ai file C++
- 1.8.6. Gestione dei file

1.9. Ricorsione
- 1.9.1. Definizione di ricorsione
- 1.9.2. Tipi di ricorsione
- 1.9.3. Vantaggi e svantaggi
- 1.9.4. Considerazioni
- 1.9.5. Conversione ricorsiva-iterativa
- 1.9.6. Lo stack di ricorsione

1.10. Prove e documentazione
- 1.10.1. Test del programma
- 1.10.2. Test della scatola bianca
- 1.10.3. Test della scatola nera
- 1.10.4. Strumenti per i test
- 1.10.5. Documentazione del programma

Modulo 2. Struttura dati

2.1. Introduzione alla programmazione C++
- 2.1.1. Classi, costruttori, metodi e attributi
- 2.1.2. Variabili
- 2.1.3. Espressioni condizionali e loop
- 2.1.4. Obiettivi

2.2. Tipi di dati astratti (ADT)
- 2.2.1. Tipi di dati
- 2.2.2. Strutture di base e ADT
- 2.2.3. Vettori e array

2.3. Strutture dati lineari
- 2.3.1. ADT Lista. Definizione
- 2.3.2. Liste collegate e doppiamente collegate
- 2.3.3. Liste ordinate
- 2.3.4. Liste in C++
- 2.3.5. ADT pila
- 2.3.6. ADT coda
- 2.3.7. Pila e coda in C++

2.4. Strutture dati gerarchiche
- 2.4.1. ADT albero
- 2.4.2. Percorsi
- 2.4.3. Alberi n-ari
- 2.4.4. Alberi binari
- 2.4.5. Alberi binari di ricerca

2.5. Strutture dati gerarchiche: alberi complessi
- 2.5.1. Alberi perfettamente bilanciati o di altezza minima
- 2.5.2. Alberi multipercorso
- 2.5.3. Riferimenti bibliografici

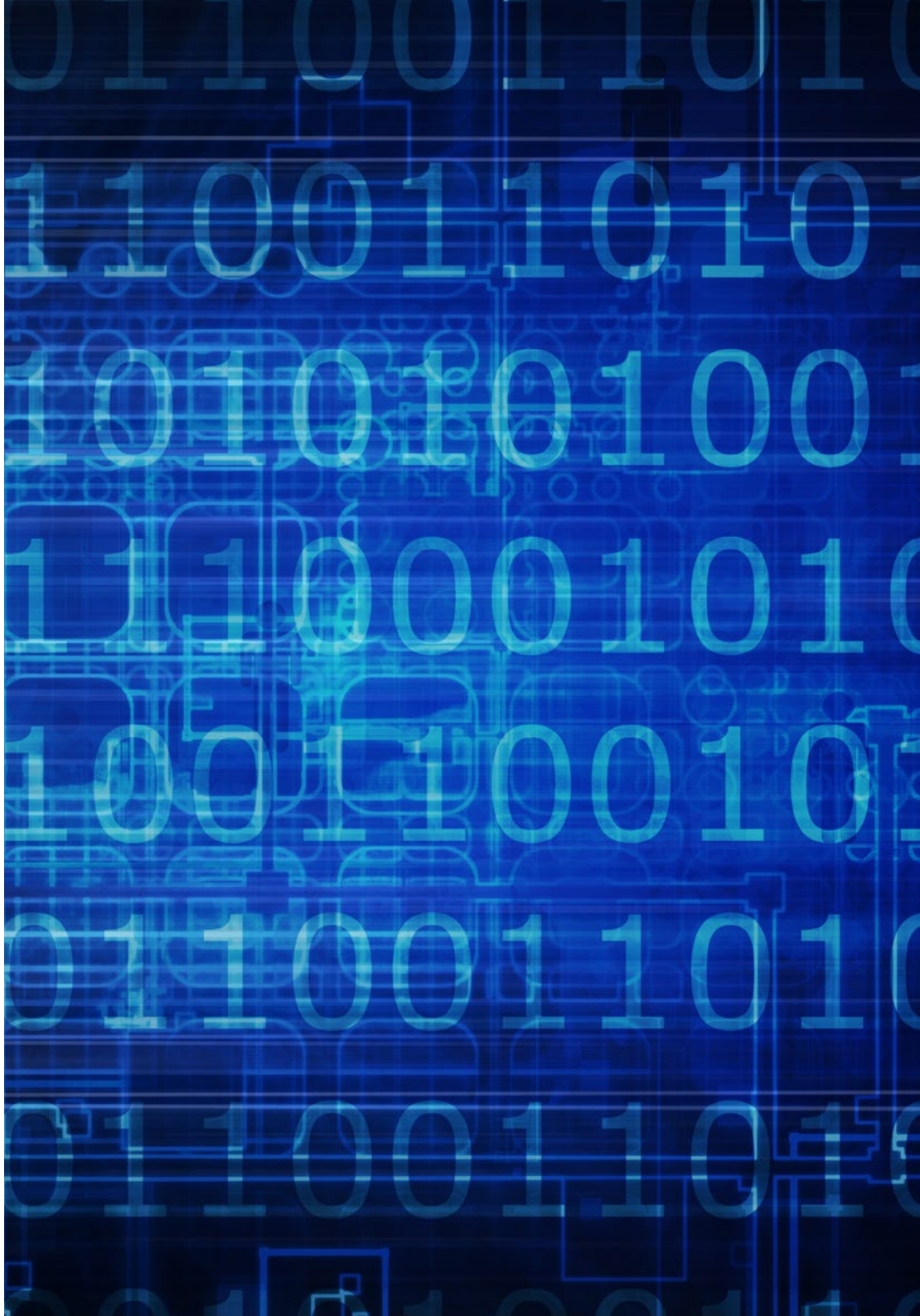
2.6. Heap e code di priorità
- 2.6.1. ADT heap
- 2.6.2. ADT coda di priorità

2.7. Tabelle Hash
- 2.7.1. ADT Tabella *Hash*
- 2.7.2. Funzioni *Hash*
- 2.7.3. Funzioni *Hash* nelle tabelle *Hash*
- 2.7.4. Ridispersione
- 2.7.5. Tabelle *Hash* aperte

2.8. Grafi
- 2.8.1. ADT grafo
- 2.8.2. Tipi di grafo
- 2.8.3. Rappresentazione grafica e operazioni di base
- 2.8.4. Progettazione di grafi

2.9. Algoritmi e concetti avanzati sui grafi
- 2.9.1. Problemi sui grafi
- 2.9.2. Algoritmi sui percorsi
- 2.9.3. Algoritmi di ricerca o percorsi
- 2.9.4. Altri algoritmi

2.10. Altre strutture di dati
- 2.10.1. Insiemi
- 2.10.2. Array paralleli
- 2.10.3. Tabelle di simboli
- 2.10.4. *Trie*



Modulo 3. Algoritmi e complessità

3.1. Introduzione alle strategie di progettazione degli algoritmi
- 3.1.1. Ricorsione
- 3.1.2. Divide et impera
- 3.1.3. Altre strategie

3.2. Efficienza e analisi degli algoritmi
- 3.2.1. Misure di efficienza
- 3.2.2. Misurare le dimensioni dell'ingresso
- 3.2.3. Misurare il tempo di esecuzione
- 3.2.4. Caso peggiore, caso migliore e caso medio
- 3.2.5. Notazione asintotica
- 3.2.6. Criteri di analisi matematica per algoritmi non ricorsivi
- 3.2.7. Analisi matematica di algoritmi ricorsivi
- 3.2.8. Analisi empirica degli algoritmi

3.3. Algoritmi di ordinamento
- 3.3.1. Concetto di gestione
- 3.3.2. Ordinamento per bolla
- 3.3.3. Ordinamento per selezione
- 3.3.4. Ordinamento per inserimento
- 3.3.5. Merge Sort
- 3.3.6. QuickSort

3.4. Algoritmi con alberi
- 3.4.1. Concetto di albero
- 3.4.2. Alberi binari
- 3.4.3. Percorsi di alberi
- 3.4.4. Rappresentare espressioni
- 3.4.5. Alberi binari ordinati
- 3.4.6. Alberi binari bilanciati

3.5. Algoritmi con *Heap*
- 3.5.1. Gli *Heap*
- 3.5.2. L'algoritmo HeapSort
- 3.5.3. Code di priorità

3.6. Algoritmi con grafi
- 3.6.1. Rappresentazione
- 3.6.2. Percorso in larghezza
- 3.6.3. Percorso in profondità
- 3.6.4. Ordinamento topologico

3.7. Algoritmi *Greedy*
- 3.7.1. La strategia *Greedy*
- 3.7.2. Elementi della strategia *Greedy*
- 3.7.3. Scambio di monete
- 3.7.4. Problema del viaggiatore
- 3.7.5. Problema dello zaino

3.8. Ricerca di percorsi minimi
- 3.8.1. Il problema del percorso minimo
- 3.8.2. Archi negativi e cicli
- 3.8.3. Algoritmo di Dijkstra

3.9. Algoritmi greedy sui grafi
- 3.9.1. L'albero di copertura minimo
- 3.9.2. Algoritmo di Prim
- 3.9.3. Algoritmo di Kruskal
- 3.9.4. Analisi della complessità

3.10. *Backtracking*
- 3.10.1. Il *Backtracking*
- 3.10.2. Tecniche alternative

Modulo 4. Progettazione avanzata di algoritmi

4.1. Analisi di algoritmi ricorsivi e divide et impera
- 4.1.1. Porre e risolvere equazioni di ricorrenza omogenee e non omogenee
- 4.1.2. Panoramica della strategia divide et impera

4.2. Analisi ammortizzata
- 4.2.1. Analisi aggregata
- 4.2.2. Il metodo di contabilità
- 4.2.3. Il metodo del potenziale

4.3. Programmazione dinamica e algoritmi per problemi NP
- 4.3.1. Caratteristiche della programmazione dinamica
- 4.3.2. Backtracking
- 4.3.3. Ramificazione e potatura

4.4. Ottimizzazione combinatoria
- 4.4.1. Rappresentazione dei problemi
- 4.4.2. Ottimizzazione 1D

4.5. Algoritmi randomizzati
- 4.5.1. Esempi di algoritmi randomizzati
- 4.5.2. Il teorema Buffon
- 4.5.3. Algoritmo Monte Carlo
- 4.5.4. Algoritmo Las Vegas

4.6. Ricerca locale e con candidati
- 4.6.1. *Garcient Ascent*
- 4.6.2. *Hill Climbing*
- 4.6.3. *Simulated Annealing*
- 4.6.4. *Tabu Search*
- 4.6.5. Ricerca con candidati

4.7. Verifica formale dei programmi
- 4.7.1. Specifica delle astrazioni funzionali
- 4.7.2. Il linguaggio della logica di prim'ordine
- 4.7.3. Sistema formale di Hoare

4.8. Verifica dei programmi iterativi
- 4.8.1. Regole del sistema formale di Hoare
- 4.8.2. Concetto invariante di iterazioni

4.9. Metodi numerici
- 4.9.1. Il metodo della bisezione
- 4.9.2. Metodo di Newton Raphson
- 4.9.3. Il metodo delle secanti

4.10. Algoritmi paralleli
- 4.10.1. Operazioni binarie parallele
- 4.10.2. Operazioni parallele con grafi
- 4.10.3. Parallelismo nel divide et impera
- 4.10.4. Parallelismo nella programmazione dinamica

Modulo 5. Programmazione avanzata

5.1. Introduzione alla programmazione orientata agli oggetti
- 5.1.1. Introduzione alla programmazione orientata agli oggetti
- 5.1.2. Progettazione di classi
- 5.1.3. Introduzione a UML per la modellazione dei problemi

5.2. Relazioni tra classi
- 5.2.1. Astrazione ed ereditarietà
- 5.2.2. Concetti avanzati di ereditarietà
- 5.2.3. Polimorfismi
- 5.2.4. Composizione e aggregazione

5.3. Introduzione ai modelli di progettazione per i problemi orientati agli oggetti
- 5.3.1. Cosa sono i modelli di progettazione
- 5.3.2. Modello *Factory*
- 5.3.3. Modello *Singleton*
- 5.3.4. Modello *Observer*
- 5.3.5. Modello *Composite*

5.4. Eccezioni
- 5.4.1. Cosa sono le eccezioni?
- 5.4.2. Cattura e gestione delle eccezioni
- 5.4.3. Avvio delle eccezioni
- 5.4.4. Creazione di eccezioni

5.5. Interfacce utente
- 5.5.1. Introduzione a Qt
- 5.5.2. Posizionamento
- 5.5.3. Cosa sono gli eventi?
- 5.5.4. Eventi: definizione e cattura
- 5.5.5. Sviluppo dell'interfaccia utente

5.6. Introduzione alla programmazione concorrente
- 5.6.1. Introduzione alla programmazione concorrente
- 5.6.2. Il concetto di processo e di thread
- 5.6.3. Interazione tra processi o thread
- 5.6.4. Thread in C++
- 5.6.5. Vantaggi e svantaggi della programmazione concorrente

5.7. Gestione e sincronizzazione dei thread
- 5.7.1. Ciclo di vita dei thread
- 5.7.2. La classe *Thread*
- 5.7.3. Pianificazione dei thread
- 5.7.4. Gruppi di thread
- 5.7.5. Thread daemon
- 5.7.6. Sincronizzazione
- 5.7.7. Meccanismi di bloccaggio
- 5.7.8. Meccanismi di comunicazione
- 5.7.9. Monitor

5.8. Problemi comuni della programmazione concorrente
- 5.8.1. Il problema dei produttori-consumatori
- 5.8.2. Il problema dei lettori e degli scrittori
- 5.8.3. Il problema della cena dei filosofi

5.9. Documentazione e test del software
- 5.9.1. Perché è importante documentare il software?
- 5.9.2. Documento di progettazione
- 5.9.3. Utilizzo di strumenti per la documentazione

5.10. Test del software
- 5.10.1. Introduzione ai test del software
- 5.10.2. Tipi di test
- 5.10.3. Test unitario
- 5.10.4. Test di integrazione
- 5.10.5. Test di validazione
- 5.10.6. Test del sistema

Modulo 6. Informatica teorica

6.1. Concetti matematici utilizzati
- 6.1.1. Introduzione alla logica proposizionale
- 6.1.2. Teoria delle relazioni
- 6.1.3. Insiemi numerabili e non numerabili

6.2. Linguaggi formali e grammatiche e introduzione alle macchine di Turing
- 6.2.1. Linguaggi e grammatiche formali
- 6.2.2. Problema decisionale
- 6.2.3. La macchina di Turing

6.3. Estensioni per macchine di Turing, macchine di Turing vincolate e computer
- 6.3.1. Tecniche di programmazione per macchine di Turing
- 6.3.2. Estensioni per macchine di Turing
- 6.3.3. Macchine di Turing vincolate
- 6.3.4. Macchine di Turing e computer

6.4. Indecidibilità
- 6.4.1. Linguaggio non ricorsivo enumerabile
- 6.4.2. Un problema indecidibile ricorsivamente enumerabile

6.5. Altri problemi indecidibili
- 6.5.1. Problemi indecidibili per macchine di Turing
- 6.5.2. Post Problema di Corrispondenza (PCP)

6.6. Problemi intrattabili
- 6.6.1. Le classi P e NP
- 6.6.2. Un problema NP completo
- 6.6.3. Problema di soddisfacibilità ristretta
- 6.6.4. Altri problemi NP completi

6.7. Problemi di Co-NP e PS
- 6.7.1. Complementari ai linguaggi NP
- 6.7.2. Problemi risolvibili nello spazio polinomiale
- 6.7.3. Problema PS completo

6.8. Classi di linguaggi basati sulla randomizzazione
- 6.8.1. Modello per fenomeni aleatori
- 6.8.2. Le classi RP e ZPP
- 6.8.3. Test di primalità
- 6.8.4. Complessità del test di primalità

6.9. Altre classi e grammatiche
- 6.9.1. Automi finiti probabilistici
- 6.9.2. Automi cellulari
- 6.9.3. Cellule di McCullogh e Pitts
- 6.9.4. Grammatiche di Lindenmayer

6.10. Sistemi informatici avanzati
- 6.10.1. Computing a membrana: sistemi P
- 6.10.2. Computing con DNA
- 6.10.3. Computing quantistico

Modulo 7. Teoria degli automi e linguaggi formali

7.1. Introduzione alle teoria degli automi
- 7.1.1. Perché studiare la teoria degli automi?
- 7.1.2. Introduzione alle dimostrazioni formali
- 7.1.3. Altre forme di dimostrazioni
- 7.1.4. Induzione matematica
- 7.1.5. Alfabeti, stringhe e linguaggi

7.2. Automi finiti deterministi
- 7.2.1. Introduzione agli automi finiti
- 7.2.2. Automi finiti deterministi

7.3. Automi finiti non deterministi
- 7.3.1. Automi finiti non deterministi
- 7.3.2. Equivalenza tra ASF e ASFND
- 7.3.3. Automi finiti con transizioni

7.4. Linguaggio ed espressioni regolari (I)
- 7.4.1. Linguaggio ed espressioni regolari
- 7.4.2. Automi finiti ed espressioni regolari

7.5. Linguaggio ed espressioni regolari (II)
- 7.5.1. Conversione di espressioni regolari in automi
- 7.5.2. Applicazioni delle espressioni regolari
- 7.5.3. Algebra delle espressioni regolari

7.6. Pompaggio del lemma e chiusura dei linguaggi regolari
- 7.6.1. Pompaggio del lemma
- 7.6.2. Proprietà di chiusura dei linguaggi regolari

7.7. Equivalenza e minimizzazione degli automi
- 7.7.1. Equivalenza di ASF
- 7.7.2. Minimizzazione di ASF

7.8. Grammatiche libere dal contesto (CFG)
- 7.8.1. Grammatiche libere dal contesto
- 7.8.2. Alberi di derivazione
- 7.8.3. Applicazioni delle CFG
- 7.8.4. Ambiguità nelle grammatiche e nei linguaggi

7.9. Automi a pila e CFG
- 7.9.1. Definizione di automi a pila
- 7.9.2. Linguaggi accettati da un automa a pila
- 7.9.3. Equivalenza tra automi a pila e CFG
- 7.9.4. Automa a pila determinista

7.10. Forme normali, schema di pompaggio CFG e proprietà dei linguaggi liberi dal contesto
- 7.10.1. Forme normali di CFG
- 7.10.2. Pompaggio del lemma
- 7.10.3. Proprietà di chiusura dei linguaggi
- 7.10.4. Proprietà di decisionale dei linguaggi liberi dal contesto

Modulo 8. Processori linguistici

8.1. Introduzione al processo di compilazione
- 8.1.1. Compilazione e interpretazione
- 8.1.2. Ambiente di esecuzione del compilatore
- 8.1.3. Processo di analisi
- 8.1.4. Processo di sintesi

8.2. Analizzatore lessicale
- 8.2.1. Cos'è un analizzatore lessicale?
- 8.2.2. Implementazione dell'analizzatore lessicale
- 8.2.3. Azioni semantiche
- 8.2.4. Recupero degli errori
- 8.2.5. Problemi di implementazione

8.3. Parsing
- 8.3.1. Cos'è un parser?
- 8.3.2. Concetti preliminari
- 8.3.3. Parser top-down
- 8.3.4. Parser bottom-up

8.4. Parsing top-down e parsing bottom-up
- 8.4.1. Parser LL(1)
- 8.4.2. Parsing LR(0)
- 8.4.3. Esempio di parser

8.5. Parsing bottom-up avanzato
- 8.5.1. Parser SLR
- 8.5.2. Parser LR(1)
- 8.5.3. Parser LR(k)
- 8.5.4. Parser LALR

8.6. Analizzatore semantico (I)
- 8.6.1. Traduzione guidata dalla sintassi
- 8.6.2. Tabelle di simboli

8.7. Analizzatore semantico (II)
- 8.7.1. Verifica dei tipi
- 8.7.2. Il sottosistema dei tipi
- 8.7.3. Equivalenza dei tipi e conversioni

8.8. Generazione di codice e ambiente di esecuzione
- 8.8.1. Aspetti progettuali
- 8.8.2. Ambiente di esecuzione
- 8.8.3. Organizzazione della memoria
- 8.8.4. Assegnazione della memoria

8.9. Generazione di codice intermedio
- 8.9.1. Traduzione guidata dalla sintesi
- 8.9.2. Rappresentazioni intermedie
- 8.9.3. Esempi di traduzioni

8.10. Ottimizzazione del codice
- 8.10.1. Assegnazione dei registri
- 8.10.2. Eliminazione delle assegnazioni morte
- 8.10.3. Esecuzione in tempo di compilazione
- 8.10.4. Riordinamento delle espressioni
- 8.10.5. Ottimizzazione del loop

Modulo 9. Computer grafica e visualizzazione

9.1. Teoria del colore
- 9.1.1. Proprietà della luce
- 9.1.2. Modelli di colore
- 9.1.3. Lo standard CIE
- 9.1.4. *Profiling*

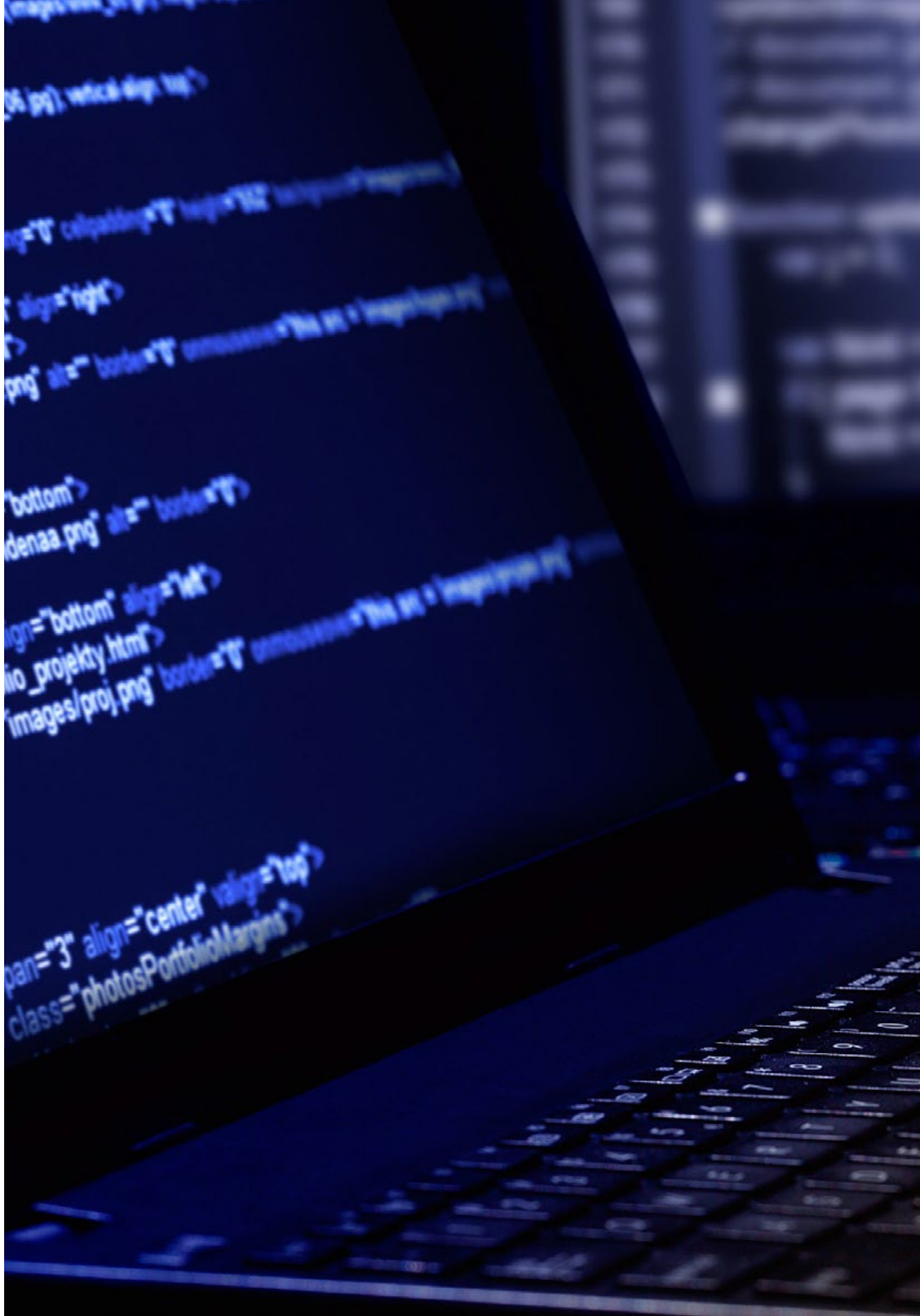
9.2. Primitive di uscita
- 9.2.1. Il controller video
- 9.2.2. Algoritmi di disegno di linee
- 9.2.3. Algoritmi di disegno di linee
- 9.2.4. Algoritmi di riempimento

9.3. Trasformazioni 2D, sistemi di coordinate 2D e ritaglio 2D
- 9.3.1. Trasformazioni geometriche di base
- 9.3.2. Coordinate omogenee
- 9.3.3. Trasformazione inversa
- 9.3.4. Composizione di trasformazioni
- 9.3.5. Altre trasformazioni
- 9.3.6. Cambiamento di coordinata
- 9.3.7. Sistemi di coordinate 2D
- 9.3.8. Cambiamento di coordinate
- 9.3.9. Normalizzazione
- 9.3.10. Algoritmi di ritaglio

9.4. Trasformazioni 3D
- 9.4.1. Traslazione
- 9.4.2. Rotazione
- 9.4.3. Scalare
- 9.4.4. Riflessione
- 9.4.5. Cesoia

9.5. Visualizzazione e modifica delle coordinate 3D
- 9.5.1. Sistemi di coordinate 3D
- 9.5.2. Visualizzazione
- 9.5.3. Cambiamento di coordinate
- 9.5.4. Proiezione e standardizzazione

9.6. Proiezione e taglio 3D
- 9.6.1. Proiezione ortogonale
- 9.6.2. Proiezione parallela obliqua
- 9.6.3. Proiezione prospettica
- 9.6.4. Algoritmi di ritaglio 3D



9.7. Rimozione di superfici nascoste
- 9.7.1. *Back face removal*
- 9.7.2. Z-buffer
- 9.7.3. Algoritmo del pittore
- 9.7.4. Algoritmo di Warnock
- 9.7.5. Rilevamento delle linee nascoste

9.8. Interpolazione e curve parametriche
- 9.8.1. Interpolazione e approssimazione polinomiale
- 9.8.2. Rappresentazione parametrica
- 9.8.3. Polinomio di Lagrange
- 9.8.4. *Spline* cubici naturali
- 9.8.5. Funzioni di base
- 9.8.6. Rappresentazione della matrice

9.9. Curve Bézier
- 9.9.1. Costruzione algebrica
- 9.9.2. Forma a matrice
- 9.9.3. Composizione
- 9.9.4. Costruzione geometrica
- 9.9.5. Algoritmo di disegno

9.10. *B-Splin*
- 9.10.1. Il problema del controllo locale
- 9.10.2. B-Splin cubiche uniformi
- 9.10.3. Funzioni di base e punti di controllo
- 9.10.4. Deriva verso l'origine e la molteplicità
- 9.10.5. Rappresentazione della matrice
- 9.10.6. *B-Splin* non uniformi

Modulo 10. Informatica bio-ispirata

10.1. Introduzione all'Informatica bio-ispirata
- 10.1.1. Introduzione all'Informatica bio-ispirata

10.2. Algoritmi di adattamento sociale
- 10.2.1. Informatica bio-ispirata basata su colonie di formiche
- 10.2.2. Varianti degli algoritmi di colonia di formiche
- 10.2.3. Computing basato su nubi di particelle

10.3. Algoritmi genetici
- 10.3.1. Struttura generale
- 10.3.2. Implementazioni dei principali operatori

10.4. Strategie di esplorazione e sfruttamento dello spazio per gli algoritmi genetici
- 10.4.1. Algoritmo CHC
- 10.4.2. Problemi multimodali

10.5. Modelli di computing evolutivo (I)
- 10.5.1. Strategie evolutive
- 10.5.2. Programmazione evolutiva
- 10.5.3. Algoritmi basati sull'evoluzione differenziale

10.6. Modelli di computing evolutivo (II)
- 10.6.1. Modelli di evoluzione basati sulla stima delle distribuzioni
- 10.6.2. Programmazione genetica

10.7. Programmazione dello sviluppo applicata ai disturbi dell'apprendimento
- 10.7.1. Apprendimento basato sulle regole
- 10.7.2. Metodi evolutivi nei problemi di selezione delle istanze

10.8. Problemi multi-obiettivo
- 10.8.1. Concetto di dominanza
- 10.8.2. Applicazione di algoritmi evolutivi a problemi multi-obiettivo

10.9. Reti neuronali (I)
- 10.9.1. Introduzione alle reti neurali
- 10.9.2. Esempio pratico con le reti neurali

10.10. Reti neuronali (II)
- 10.10.1. Casi di utilizzo delle reti neurali nella ricerca medica
- 10.10.2. Casi di utilizzo delle reti neurali nell'economia
- 10.10.3. Casi di utilizzo delle reti neurali nella visione artificiale

05

Metodologia

Questo programma ti offre un modo differente di imparare. La nostra metodologia si sviluppa in una modalità di apprendimento ciclico: ***il Relearning***.

Questo sistema di insegnamento viene applicato nelle più prestigiose facoltà di medicina del mondo ed è considerato uno dei più efficaci da importanti pubblicazioni come il ***New England Journal of Medicine.***

Scopri il Relearning, un sistema che abbandona l'apprendimento lineare convenzionale, per guidarti attraverso dei sistemi di insegnamento ciclici: una modalità di apprendimento che ha dimostrato la sua enorme efficacia, soprattutto nelle materie che richiedono la memorizzazione"

Caso di Studio per contestualizzare tutti i contenuti

Il nostro programma offre un metodo rivoluzionario per sviluppare le abilità e le conoscenze. Il nostro obiettivo è quello di rafforzare le competenze in un contesto mutevole, competitivo e altamente esigente.

Con TECH potrai sperimentare un modo di imparare che sta scuotendo le fondamenta delle università tradizionali in tutto il mondo"

Avrai accesso a un sistema di apprendimento basato sulla ripetizione, con un insegnamento naturale e progressivo durante tutto il programma.

Imparerai, attraverso attività collaborative e casi reali, la risoluzione di situazioni complesse in ambienti aziendali reali.

Un metodo di apprendimento innovativo e differente

Questo programma di TECH consiste in un insegnamento intensivo, creato ex novo, che propone le sfide e le decisioni più impegnative in questo campo, sia a livello nazionale che internazionale. Grazie a questa metodologia, la crescita personale e professionale viene potenziata, effettuando un passo decisivo verso il successo. Il metodo casistico, la tecnica che sta alla base di questi contenuti, garantisce il rispetto della realtà economica, sociale e professionale più attuali.

> *Il nostro programma ti prepara ad affrontare nuove sfide in ambienti incerti e a raggiungere il successo nella tua carriera"*

Il Metodo Casistico è stato il sistema di apprendimento più usato nelle migliori Scuole di Informatica del mondo da quando esistono. Sviluppato nel 1912 affinché gli studenti di Diritto non imparassero la legge solo sulla base del contenuto teorico, il metodo casistico consisteva nel presentare loro situazioni reali e complesse per prendere decisioni informate e giudizi di valore su come risolverle. Nel 1924 fu stabilito come metodo di insegnamento standard ad Harvard.

Cosa dovrebbe fare un professionista per affrontare una determinata situazione? Questa è la domanda con cui ti confrontiamo nel metodo dei casi, un metodo di apprendimento orientato all'azione. Durante il corso, gli studenti si confronteranno con diversi casi di vita reale. Dovranno integrare tutte le loro conoscenze, effettuare ricerche, argomentare e difendere le proprie idee e decisioni.

Metodologia Relearning

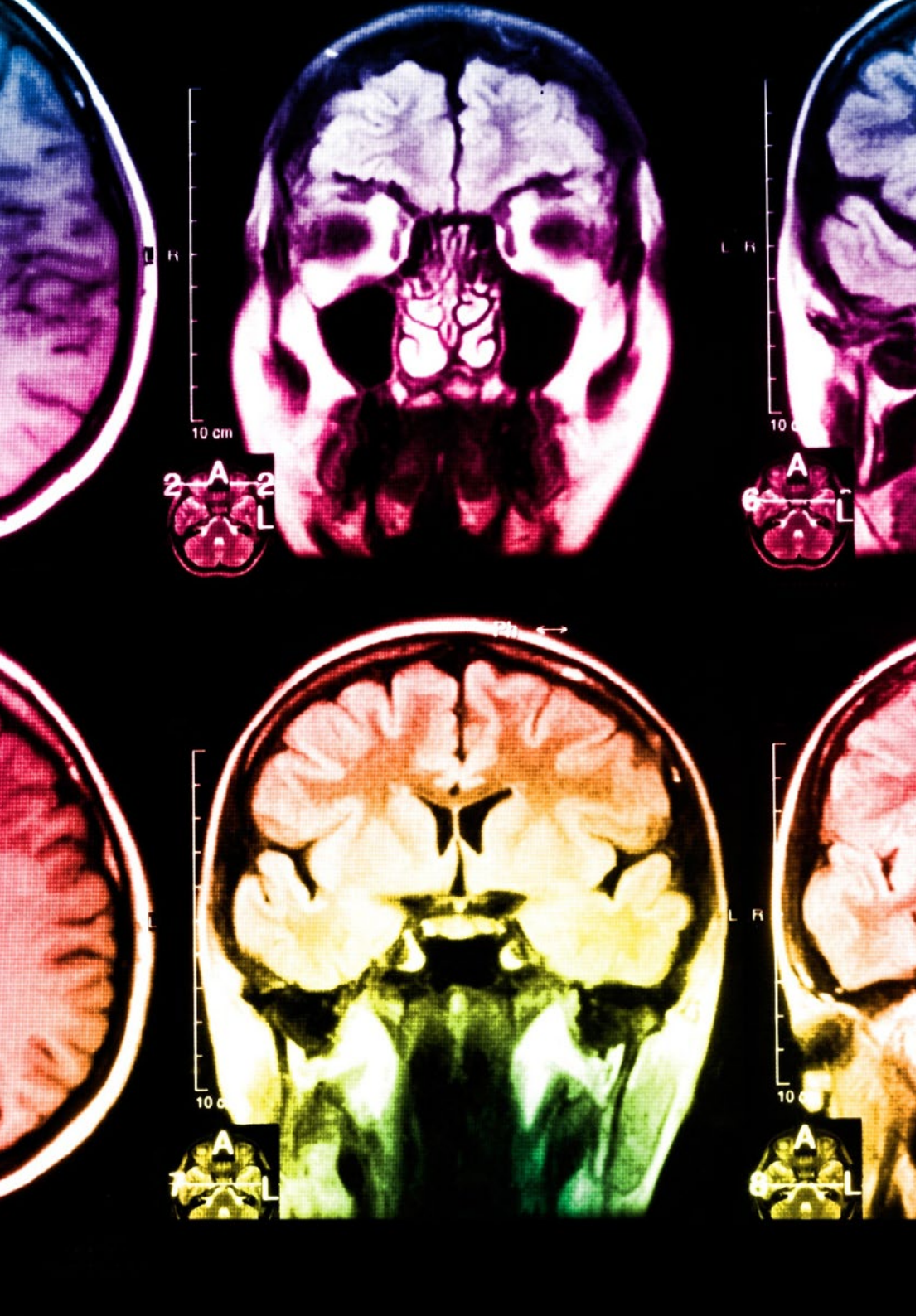
TECH coniuga efficacemente la metodologia del Caso di Studio con un sistema di apprendimento 100% online basato sulla ripetizione, che combina diversi elementi didattici in ogni lezione.

Potenziamo il Caso di Studio con il miglior metodo di insegnamento 100% online: il Relearning.

Nel 2019 abbiamo ottenuto i migliori risultati di apprendimento di tutte le università online del mondo.

In TECH imparerai con una metodologia all'avanguardia progettata per formare i manager del futuro. Questo metodo, all'avanguardia della pedagogia mondiale, si chiama Relearning.

La nostra università è l'unica autorizzata a utilizzare questo metodo di successo. Nel 2019, siamo riusciti a migliorare il livello di soddisfazione generale dei nostri studenti (qualità dell'insegnamento, qualità dei materiali, struttura del corso, obiettivi...) rispetto agli indicatori della migliore università online.



Nel nostro programma, l'apprendimento non è un processo lineare, ma avviene in una spirale (impariamo, disimpariamo, dimentichiamo e re-impariamo). Pertanto, combiniamo ciascuno di questi elementi in modo concentrico. Questa metodologia ha formato più di 650.000 laureati con un successo senza precedenti in campi diversi come la biochimica, la genetica, la chirurgia, il diritto internazionale, le competenze manageriali, le scienze sportive, la filosofia, il diritto, l'ingegneria, il giornalismo, la storia, i mercati e gli strumenti finanziari. Tutto questo in un ambiente molto esigente, con un corpo di studenti universitari con un alto profilo socio-economico e un'età media di 43,5 anni.

Il Relearning ti permetterà di apprendere con meno sforzo e più performance, impegnandoti maggiormente nella tua specializzazione, sviluppando uno spirito critico, difendendo gli argomenti e contrastando le opinioni: un'equazione diretta al successo.

Dalle ultime evidenze scientifiche nel campo delle neuroscienze, non solo sappiamo come organizzare le informazioni, le idee, le immagini e i ricordi, ma sappiamo che il luogo e il contesto in cui abbiamo imparato qualcosa è fondamentale per la nostra capacità di ricordarlo e immagazzinarlo nell'ippocampo, per conservarlo nella nostra memoria a lungo termine.

In questo modo, e in quello che si chiama Neurocognitive Context-dependent E-learning, i diversi elementi del nostro programma sono collegati al contesto in cui il partecipante sviluppa la sua pratica professionale.

Questo programma offre i migliori materiali didattici, preparati appositamente per i professionisti:

Materiale di studio

Tutti i contenuti didattici sono creati appositamente per il corso dagli specialisti che lo impartiranno, per fare in modo che lo sviluppo didattico sia davvero specifico e concreto.

Questi contenuti sono poi applicati al formato audiovisivo che supporterà la modalità di lavoro online di TECH. Tutto questo, con le ultime tecniche che offrono componenti di alta qualità in ognuno dei materiali che vengono messi a disposizione dello studente.

Master class

Esistono evidenze scientifiche sull'utilità dell'osservazione di esperti terzi.

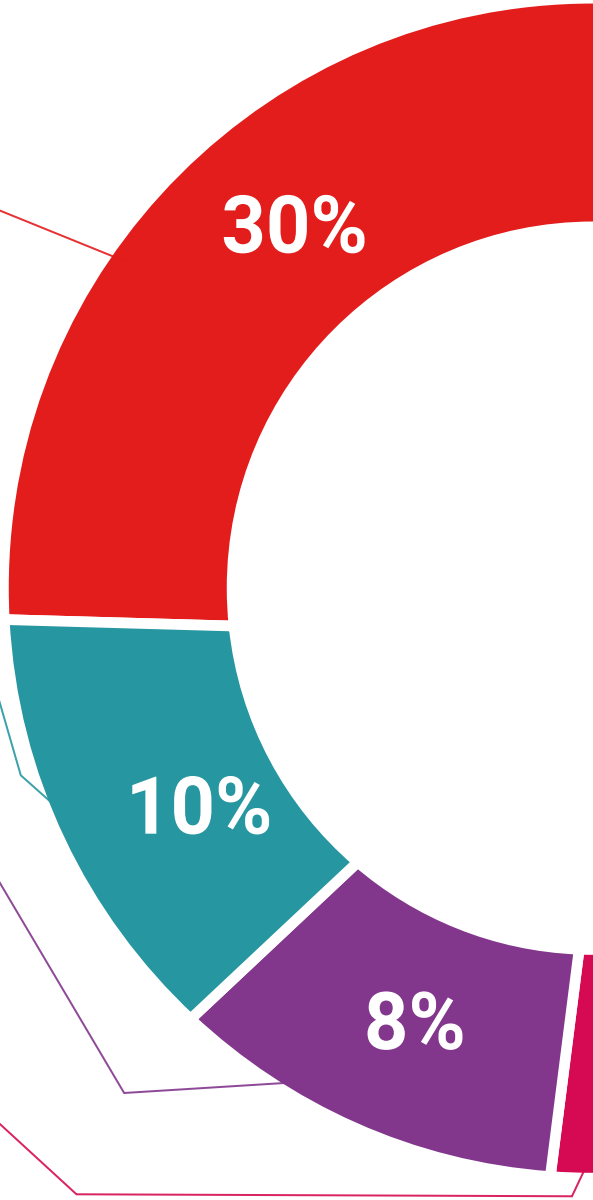
Imparare da un esperto rafforza la conoscenza e la memoria, costruisce la fiducia nelle nostre future decisioni difficili.

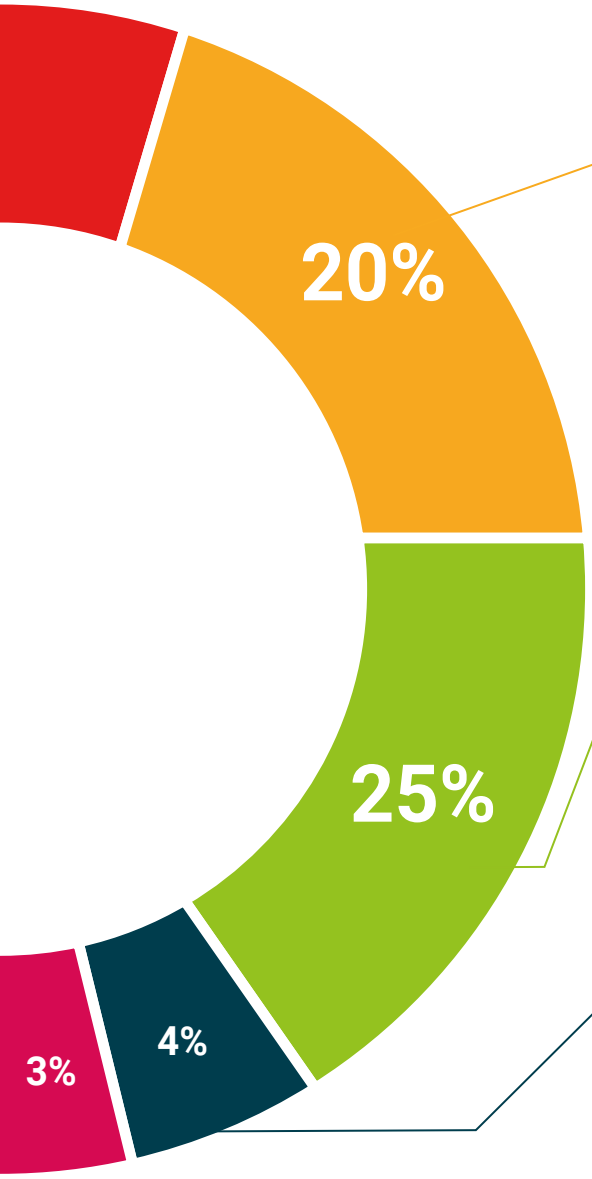
Pratiche di competenze e competenze

Svolgerai attività per sviluppare competenze e capacità specifiche in ogni area tematica. Pratiche e dinamiche per acquisire e sviluppare le competenze e le abilità che uno specialista deve sviluppare nel quadro della globalizzazione in cui viviamo.

Letture complementari

Articoli recenti, documenti di consenso e linee guida internazionali, tra gli altri. Nella biblioteca virtuale di TECH potrai accedere a tutto il materiale necessario per completare la tua specializzazione.





Casi di Studio

Completerai una selezione dei migliori casi di studio scelti appositamente per questo corso. Casi presentati, analizzati e monitorati dai migliori specialisti del panorama internazionale.

Riepiloghi interattivi

Il team di TECH presenta i contenuti in modo accattivante e dinamico in pillole multimediali che includono audio, video, immagini, diagrammi e mappe concettuali per consolidare la conoscenza.

Questo esclusivo sistema di specializzazione per la presentazione di contenuti multimediali è stato premiato da Microsoft come "Caso di successo in Europa".

Testing & Retesting

Valutiamo e rivalutiamo periodicamente le tue conoscenze durante tutto il programma con attività ed esercizi di valutazione e autovalutazione, affinché tu possa verificare come raggiungi progressivamente i tuoi obiettivi.

06

Titolo

Il Master Privato in Informatica e Linguaggi di Programmazione ti garantisce, oltre alla preparazione più rigorosa e aggiornata, l'accesso a una qualifica di Master Privato rilasciata da TECH Università Tecnologica.

Porta a termine questo programma e ricevi la tua qualifica universitaria senza spostamenti o fastidiose formalità"

Questo **Master Privato in Informatica e Linguaggi di Programmazione** possiede il programma più completo e aggiornato del mercato.

Dopo aver superato la valutazione, lo studente riceverà mediante lettera certificata* con ricevuta di ritorno, la sua corrispondente qualifica di **Master Privato** rilasciata da **TECH Università Tecnologica.**

Il titolo rilasciato da **TECH Università Tecnologica** esprime la qualifica ottenuta nel Master Privato, e riunisce tutti i requisiti comunemente richiesti da borse di lavoro, concorsi e commissioni di valutazione di carriere professionali.

Titolo: **Master Privato in Informatica e Linguaggi di Programmazione**

N. Ore Ufficiali: **1.500**


tech università tecnologica

Conferisce il presente

DIPLOMA

a

Dott./Dott.ssa _____, con documento d'identità n°_____

Per aver completato con esito positivo e accreditato il programma di

MASTER PRIVATO

in

Informatica e Linguaggi di Programmazione

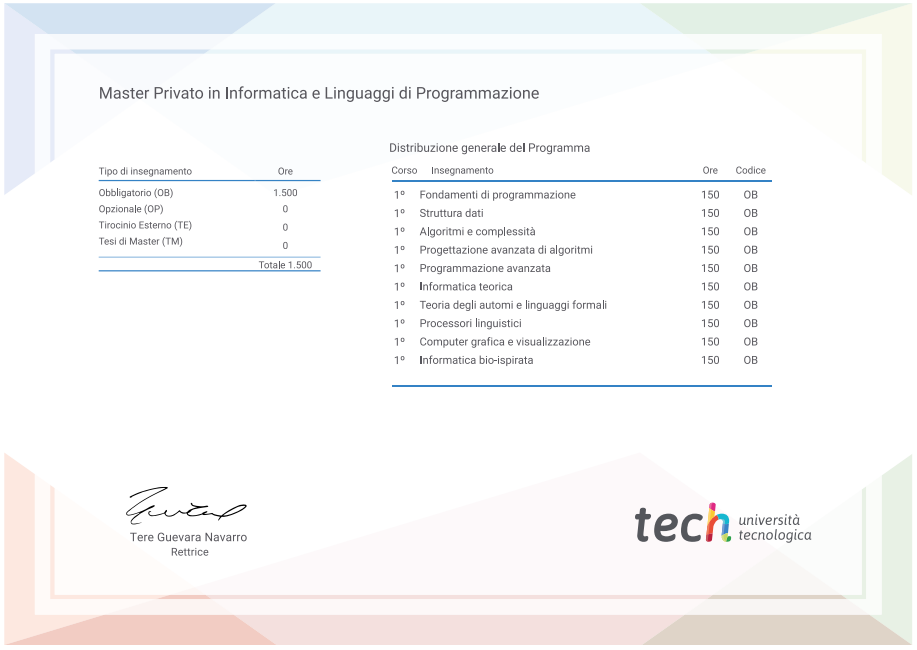
Si tratta di un titolo rilasciato da questa Università ed equivalente a 1.500 ore, con data di inizio gg/mm/aaaa e con data di fine gg/mm/aaaa.

TECH è un Istituto Privato di Istruzione Superiore riconosciuto dal Ministero della Pubblica Istruzione a partire dal 28 giugno 2018.

In data 17 Giugno 2020

Tere Guevara Navarro
Rettrice

Questo titolo deve essere sempre accompagnato da un titolo universitario rilasciato dall'autorità competente per l'esercizio della pratica professionale in ogni paese. codice unico TECH: AFWORD23S techtitute.com/titulos


Master Privato in Informatica e Linguaggi di Programmazione

Tipo di insegnamento	Ore
Obbligatorio (OB)	1.500
Opzionale (OP)	0
Tirocinio Esterno (TE)	0
Tesi di Master (TM)	0
	Totale 1.500

Distribuzione generale del Programma

Corso	Insegnamento	Ore	Codice
1°	Fondamenti di programmazione	150	OB
1°	Struttura dati	150	OB
1°	Algoritmi e complessità	150	OB
1°	Progettazione avanzata di algoritmi	150	OB
1°	Programmazione avanzata	150	OB
1°	Informatica teorica	150	OB
1°	Teoria degli automi e linguaggi formali	150	OB
1°	Processori linguistici	150	OB
1°	Computer grafica e visualizzazione	150	OB
1°	Informatica bio-ispirata	150	OB

Tere Guevara Navarro
Rettrice

tech università tecnologica

*Se lo studente dovesse richiedere che il suo diploma cartaceo sia provvisto di Apostille dell'Aia, TECH EDUCATION effettuerà le gestioni opportune per ottenerla pagando un costo aggiuntivo.

tech università tecnologica

Master Privato

Informatica e Linguaggi di Programmazione

Modalità: **Online**

Durata: **12 mesi**

Titolo: **TECH Università Tecnologica**

Ore teoriche: **1.500**

Master Privato

Informatica e Linguaggi di Programmazione

tech università tecnologica