Repubblica Italiana
Provincia Autonoma di Bolzano - Alto Adige

Republik Italien
Autonome Provinz Bozen - Südtirol

Istituto d'Istruzione Secondaria Superiore per le scienze, le tecnologie e i servizi

"GALILEO GALILEI"

Oberschulzentrum für Wissenschaften, Technologie und Dienstleistungen

ISTITUTO TECNICO TECNOLOGICO - LICEO SCIENTIFICO-SCIENZE APPLICATE
ISTITUTO PROFESSIONALE PER L'INDUSTRIA E L'ARTIGIANATO - ISTITUTO PROFESSIONALE ODONTOTECNICO
Fachoberschule für den Technologischen Bereich - Realgymnasium mit Schwerpunkt angewandte Naturwissenschaften
Berufsbildende Oberschule für Industrie und Handel - Berufsbildende Oberschule für Zahntechniker

39100 BOLZANO- via Cadorna 14
Cod. Fisc. 80006520219

39100 Bozen - Cadornastraße 14
St.Nr. 80006520219

PIANO DI LAVORO

Programmazione Didattica per

Competenze Indirizzo Informatica e Telecomunicazioni Articolazione Informatica

DOCENTE:	**MARZOCCHELLA ANTIMO**
ITP:	***ALOISI MARA***
MATERIA:	***INFORMATICA***
CLASSE:	***3° K***
ORE SETTIMANALI:	***7 (di cui 6 in codocenza)***
ANNO SCOLASTICO:	***2019/2020***

LUOGO E DATA

BOLZANO, 24//10/2019

FIRMA

Competenze finali

- ***Competenza n. 1***
  Analizzare dati e interpretarli sviluppando deduzioni e ragionamenti sugli stessi anche con l'ausilio di rappresentazioni grafiche, usando consapevolmente gli strumenti di calcolo e le potenzialità offerte da applicazioni specifiche di tipo informatico

- ***Competenza n. 2***
  Individuare le strategie appropriate per la soluzione di problemi

- ***Competenza n. 3***
  scegliere dispositivi e strumenti in base alle loro caratteristiche funzionali

TAVOLA DI PROGRAMMAZIONE

MATERIA: ***INFORMATICA*** CLASSE: 3°

Modulo N° I: Introduzione ai problemi, agli algoritmi ed ai programmi

Attività/metodologie didattiche	Strumenti didattici	Tipologia verifiche	Tempi: Ore
lezione frontale partecipata, problem-solving, didattica laboratoriale.	libro di testo, appunti del docente, materiale multimediale, laboratorio d'informatica, IDE(G*eany (per C/C++), CodeBlocks), shell* di Linux	Scritte/orali, pratiche	42

Obiettivi disciplinari	Contenuti
COMPETENZA 1: *Analizzare dati e interpretarli sviluppando deduzioni e ragionamenti sugli stessi anche con l'ausilio di rappresentazioni grafiche, usando consapevolmente gli strumenti di calcolo e le potenzialità offerte da applicazioni specifiche di tipo informatico;* **COMPETENZA 2** *Individuare le strategie appropriate per la soluzione di problemi* **ABILITA'** • Distinguere tra i concetti di istruzione, algoritmo, programma e processo • Distinguere tra problemi decidibili e non decidibili (applicazione della tesi di Church) • Saper utilizzare i formalismi per la rappresentazioni degli algoritmi (*Flow-chart PDL*), essere in grado di darne la specifica • Saper analizzare semplici problemi e proporre algoritmi risolutivi • Saper verificare le soluzioni trovate, • utilizzo delle tavole di traccia per le configurazioni istantanee • Saper individuare e correggere gli errori a tempo di compilazione e di esecuzione	**PROGRAMMA GENERALE** ➢ Terminologia generale dell'informatica, elaborazione, automazione e dati; algoritmi e relative tecniche di analisi ((*Flow-chart PDL*) ➢ Specifica degli algoritmi, tipi di dati ed istruzioni (*programmazione imperativa*), strutture dati statiche: array lineari e matrici **CONOSCENZE:** ➢ termini generali della disciplina; proprietà fondamentali degli algoritmi ➢ enunciato della tesi di Church e relative implicazioni nel campo informatico ➢ i vari tipi di dati e la sintassi delle diverse istruzioni: assegnamento, input/output, controllo ➢ Struttura generale di un programma scritto attraverso un text-editor o un IDE in linguaggio di programmazione "C/C++" ➢ errori di sintassi, logici ed in fase di esecuzione ➢ significato di puntatore o indirizzo ➢ tecniche di analisi della complessità spaziale e temporale degli algoritmi **LABORATORIO** Verranno utilizzati gli IDE G*eany (per C/C++)*, CodeBlocks e la *shell* di Linux per lo sviluppo di semplici programmi secondo il *paradigma imperativo*

Modulo N° II: programmazione strutturata ed algoritmi notevoli			
Attività/metodologie didattiche	Strumenti didattici	Tipologia verifiche	Tempi: Ore
lezione frontale partecipata, problem-solving, didattica laboratoriale.	libro di testo, appunti del docente, materiale multimediale, laboratorio d'informatica, IDE(G*eany (per C/C++), CodeBlocks), shell* di Linux	Scritte/orali, pratiche	42

Obiettivi disciplinari	Contenuti
COMPETENZA 1: *Analizzare dati e interpretarli sviluppando deduzioni e ragionamenti sugli stessi anche con l'ausilio di rappresentazioni grafiche, usando consapevolmente gli strumenti di calcolo e le potenzialità offerte da applicazioni specifiche di tipo informatico;* **COMPETENZA 2** *Individuare le strategie appropriate per la soluzione di problemi* **ABILITA'** • Saper utilizzare le principali operazioni su liste lineari statiche: array, record, stack(pila) e queue(coda) • Saper applicare gli algoritmi di ricerca su liste di tipo lineari statiche • Riuscire ad applicare gli algoritmi fondamentali di ordinamento sulle strutture lineari • Saper utilizzare le funzioni (in C/C++) ovvero in genere i sottoprogrammi allo scopo di modulare la struttura del programma. • Saper utilizzare il passaggio dei parametri per valore e per indirizzo comprendendone l'utilità • Essere in grado di applicare la ricorsione (metodo top-down) quando la natura bottom-up della soluzione è di difficile impostazione algoritmica	**PROGRAMMA GENERALE** ➢ Strutture dati astratte lineari statiche ➢ Metodologia top-down e bottom-up ➢ funzioni e sotto-algoritmi/sotto-programmi ➢ Algoritmi ricorsivi **CONOSCENZE:** ➢ Strutture dati lineari statiche: array e record, stack(pila) e coda ➢ Principali algoritmi di ricerca su strutture dati lineari statiche ➢ Problema dell'ordinamento e relativi algoritmi per le applicazioni informatiche ➢ Definizione del prototype di una funzione (sotto-algoritmo) e del relativo codice ai fini della programmazione strutturata ➢ Significato del passaggio di parametri per valore e per indirizzo ➢ Funzionamento ed importanza della ricorsione, in termini di funzionamento della CPU e di algoritmi complessi ➢ **LABORATORIO** Verranno utilizzati gli IDE G*eany (pe C/C++)*, CodeBlocks e la *shell* di Linux per lo sviluppo di semplici programmi secondo il *paradigma imperativo*

Modulo N° III: Puntatori ed allocazione dinamica della memoria			
Attività/metodologie didattiche	Strumenti didattici	Tipologia verifiche	Tempi: Ore
lezione frontale partecipata, problem-solving, didattica laboratoriale.	libro di testo, appunti del docente, materiale multimediale, laboratorio d'informatica, IDE(G*eany (per C/C++), CodeBlocks), shell* di Linux	Scritte/orali, pratiche	42

Obiettivi disciplinari	Contenuti
COMPETENZA 1: *Analizzare dati e interpretarli sviluppando deduzioni e ragionamenti sugli stessi anche con l'ausilio di rappresentazioni grafiche, usando consapevolmente gli strumenti di calcolo e le potenzialità offerte da applicazioni specifiche di tipo informatico;* **COMPETENZA 2** *Individuare le strategie appropriate per la soluzione di problemi* **ABILITA'** • Sapere utilizzare i puntatori come riferimento a regioni strutturate di memoria • Essere in grado di implementare, attraverso i puntatori, le strutture dati dinamiche, lineari e non lineari • Essere in grado di eseguire le operazioni di inserimento, modifica e cancellazione in una linked-list • Saper attraversare un albero in maniera pre-order, in-order, post-order e level-order, sia in maniera ricorsiva che iterativa • Essere in grado di eseguire le operazioni di inserimento, modifica e cancellazione in un binary-search-tree per gestire un insieme ordinato di dati • Saper utilizzare un albero di decisione per la determinazione del numero minimo di passi per la risoluzione di un problema.	**PROGRAMMA GENERALE** ➢ Aritmetica dei puntatori ➢ Strutture dati astratte lineari e non lineari(cenni) di tipo dinamico ➢ Implementazione di linked-list ed alberi binari ➢ Implementazione dinamica delle strutture dati lineari **CONOSCENZE:** ➢ significato di puntatore, di indirizzo e di allocazione dinamica della memoria ➢ differenza tra strutture dati statiche e dinamiche ➢ strutture dati linked-list ed albero binario ➢ Sapere il significato di attraversamento pre-order, in-order, post-order e level-order di un albero binario ➢ Capire l'importanza, ai fini della complessità, che riveste il binary-search-tree per molte applicazioni informatiche che richiedono la disposizione ordinata delle informazioni ➢ Conoscere il concetto di albero di decisione **LABORATORIO** Verranno utilizzati gli IDE G*eany (pe C/C++)*, CodeBlocks e la *shell* di Linux per lo sviluppo di semplici programmi secondo il *paradigma imperativo*

Modulo N° IV: Elementi fondamentali di gestione dei file			
Attività/metodologie didattiche	Strumenti didattici	Tipologia verifiche	Tempi: Ore
lezione frontale partecipata, problem-solving, didattica laboratoriale.	libro di testo, appunti del docente, materiale multimediale, laboratorio d'informatica, IDE(G*eany (per C/C++), CodeBlocks), shell* di Linux	Scritte/orali, pratiche	28

Obiettivi disciplinari	Contenuti
COMPETENZA 1: *Analizzare dati e interpretarli sviluppando deduzioni e ragionamenti sugli stessi anche con l'ausilio di rappresentazioni grafiche, usando consapevolmente gli strumenti di calcolo e le potenzialità offerte da applicazioni specifiche di tipo informatico;* **COMPETENZA 2** *Individuare le strategie appropriate per la soluzione di problemi* **ABILITA'** • Essere in grado di organizzare i dati attraverso i file e di confrontarne le possibili organizzazioni (sequenziale, random, con indici(cenni)) • Saper utilizzare le funzioni e le procedure di gestione dei file in linguaggio C/C++ • Essere in grado di implementare le operazioni di creazione, scansione, ricerca per chiave, inserimento di record in file ordinati e non(cenni) • Essere in grado di implementare la modifica di record, cancellazione logica e fisica di record, la compattazione(cenni), l'ordinamento(cenni) e la fusione di file	**PROGRAMMA GENERALE** ➢ Concetti elementari fondamentali: file, record, campo, chiave di accesso ➢ Organizzazione logica e fisica di un file ➢ File di testo e non ➢ Tecniche di accesso ad un file: sequenziale, random o diretto, indicizzato(cenni) ➢ Operazioni fondamentali e notevoli sui file **CONOSCENZE:** ➢ Conoscere le caratteristiche dei file, le operazioni possibili su di essi e le problematiche relative alla loro gestione (logica e di S.O.) ➢ Conoscere le funzioni e procedure per il trattamento dei file in linguaggio C/C++ ➢ Conoscere le operazioni di creazione, scansione, ricerca per chiave, inserimento di record in file ordinati e non ordinati (cenni) ➢ Conoscere le operazioni di modifica di record, cancellazione logica e fisica di record, la compattazione (cenni), l'ordinamento(cenni) e la fusione di file **LABORATORIO** Verranno utilizzati gli IDE G*eany (pe C/C++)*, CodeBlocks e la *shell* di Linux per lo sviluppo di semplici programmi secondo il *paradigma imperativo*

Modulo N° V: Introduzione agli elementi fondamentali della O.O.P, E.D.P e V.P.			
Attività/metodologie didattiche	Strumenti didattici	Tipologia verifiche	Tempi: Ore
lezione frontale partecipata, problem-solving, didattica laboratoriale.	libro di testo, appunti del docente, materiale multimediale, laboratorio d'informatica, IDE(G*eany (per C/C++), Code::Blocks, Qt Creator), shell* di Linux	Scritte/orali, pratiche	60

Obiettivi disciplinari	Contenuti
COMPETENZA 1: *Analizzare dati e interpretarli sviluppando deduzioni e ragionamenti sugli stessi anche con l'ausilio di rappresentazioni grafiche, usando consapevolmente gli strumenti di calcolo e le potenzialità offerte da applicazioni specifiche di tipo informatico;* **COMPETENZA 2** *Individuare le strategie appropriate per la soluzione di problemi* **COMPETENZA 3** *scegliere dispositivi e strumenti in base alle loro caratteristiche funzionali* **ABILITA'** • Essere in grado di programmare attraverso un ambiente di sviluppo visuale, distinguendo tra fase di progettazione del codice e sviluppo dell'interfaccia grafica • Saper utilizzare gli elementi di base di un'ambiente di sviluppo, essere in grado di progettare e costruire interfacce grafiche user-friendly • Saper applicare i principi e le regole fondamentali della programmazione orientata agli oggetti ed agli eventi per la risoluzione di problemi reali • Riuscire ad applicare i metodi e gli eventi dei principali oggetti/classi di C++ • Riutilizzare il codice anche attraverso lo sviluppo di proprie classi e librerie esterne	**PROGRAMMA GENERALE** ➢ Gli ambienti RAD di sviluppo: le finestre, sintassi delle istruzioni, tipi di dati, introduzione alle classi e agli oggetti, metodi ed eventi, proprietà ➢ La costruzione di interfacce grafiche user-friendly: ➢ Utilizzo e sviluppo di classi/oggetti: **CONOSCENZE:** ➢ Conoscere l'esistenza di strumenti automatici (ovvero di ambienti RAD) per lo sviluppo di software e comprenderne l'utilità ➢ Principi e caratteristiche di base fondamentali della programmazione visuale ➢ Conoscere gli elementi di base della programmazione event-driven e della programmazione object-oriented in genere ➢ Conoscere le proprietà, i metodi e gli eventi dei principali oggetti/classi di C++ ➢ Comprendere il significato e l'utilizzo delle classi ai fini del riutilizzo del codice, uso di costruttori, principio di information hiding/incapsulamento del codice, ereditarietà e polimorfismo. **LABORATORIO** Verranno utilizzati gli IDE G*eany (pe C/C++)*, Code::Blocks, *Qt Creator* e la *shell* di Linux per lo sviluppo di semplici programmi secondo il *paradigma ad oggetti*