Piano di Lavoro Annuale del Docente

Anno Scolastico 2021/2022

Classe **4** **sez. B INF**

Disciplina **Tecnologia e progettazione di sistemi informatici**

Docenti **Margherita Lozza , Simona Cutone**

Data di presentazione Ottobre 2021

Presentazione della classe

La classe quarta è composta da 20 alunni (di cui 2 femmine e 18 maschi) evidenzia generalmente un comportamento corretto. L'analisi della situazione di partenza della classe è stata effettuata attraverso momenti di discussione, esercizi individuali alla lavagna ed esercitazioni pratiche in laboratorio tendenti a verificare i prerequisiti e dunque il livello di partenza dei alunni in relazione alle competenze, alle conoscenze e alle capacità. Dagli elementi acquisiti, dal punto di vista cognitivo, i livelli di partenza risultano eterogenei. Alcuni alunni hanno una buona preparazione di base ; una buona parte degli allievi presenta una preparazione di base nell'insieme sufficiente, qualche alunno ha una preparazione frammentaria dovuta alla poca concentrazione durante le lezioni e soprattutto al poco impegno nello studio individuale.

Finalità educative

In accordo con la programmazione annuale del Consiglio di classe, sono state individuate le seguenti finalità educative: Il corso di Informatica ha come fine principale quello di mettere il Perito in Informatica in grado di affrontare (dall'analisi fino alla documentazione) la soluzione di un problema, posto dalla richiesta di un ipotetico committente, scegliendo le metodologie e gli strumenti software più idonei offrendogli la formazione per seguire con una certa autonomia l'evoluzione delle tecnologie informatiche. La disciplina fornisce all'alunno le conoscenze e le abilità necessarie per l'uso di un sistema di elaborazione ai più alti livelli della gerarchia che lo modella (linguaggi ad alto o altissimo livello, linguaggi applicativi). Essa deve essere intesa soprattutto come l'ambiente in cui si sviluppano le capacità di analizzare e risolvere problemi (anche di una certa complessità) di varia natura, e dove di volta in volta vengono proposti i paradigmi e gli strumenti linguistici più idonei alla natura del problema. Si ricorre ripetutamente al concetto di paradigma che, in questo contesto, si intende come chiave di interpretazione dei problemi e come modello di costruzione delle soluzioni (imperativo, logico, funzionale, rivolto agli oggetti agli eventi, alle basi di dati,...). Lo studente, allo scopo di raggiungere una certa flessibilità e la capacità di affrontare nuove prospettive, deve acquisire alcune di queste chiavi e la capacità di impiegarle nei contesti appropriati. Il corso di Informatica non deve, in ogni caso, assumere un carattere nozionistico-sintattico né ridursi ad una collezione di corsi sistematici sui vari linguaggi. I contenuti debbono sempre essere organizzati intorno ai nodi concettuali che vanno sempre affrontati a partire dai problemi ed applicati alla loro soluzione. Gli specifici linguaggi debbono essere visti come mezzi espressivi e come strumenti applicativi

Obiettivi

Comportamentali e culturali

- Far emergere l'auto–consapevolezza dei diversi stili cognitivi;
- Produrre abilità comportamentali di apprendimento autonomo ed efficace;
- percepire le valenze orientative delle diverse aree del sapere (risolvere problemi, lavorare in gruppo per obiettivi/progetti, assumere responsabilità);
- stimolare un'immagine di sé centrata sulla fiducia e l'autostima;
- favorire la percezione di fiducia e dell'accettazione dell'altro;
- identificare il gruppo come pluralità in interazione e influenzamento reciproco orientato a fare assieme, in agire contingente;
- produrre consapevolezza dei vincoli istituzionali come spazi di libertà;

- promozione dell'integrazione come dinamica tra uguaglianza e differenza nonché come equilibrio tra bisogni individuali e di gruppo che genera capacità di collaborazione tramite:
    - partecipazione attiva con tutti;
    - produzioni di relazioni di fiducia e di affidamento alle idee degli altri come: competizione di ipotesi-soluzioni di problemi; flessibilità.

Didattico Cognitivi

La disciplina, nell'ambito della programmazione del Consiglio di classe, concorre in particolare al raggiungimento dei seguenti risultati di apprendimento, relativi all'indirizzo, espressi in termini di competenza:

- utilizzare le strategie del pensiero razionale negli aspetti dialettici ed algoritmici per affrontare situazioni problematiche elaborando opportune soluzioni;
- sviluppare sempre maggiore capacità e consapevolezza nell'uso delle tecnologie;
- acquisire sempre maggiore capacità di autonomia nello svolgere i compiti assegnati.
- redigere relazioni tecniche e documentare le attività individuali e di gruppo relative a situazioni professionali.

Obiettivi specifici disciplinari:

CONOSCENZE:

- **Sapere cosa succede all'accensione del PC**
- **Conoscere i compiti del sistema operativo**
- Conoscere a storia dei sistemi operativi
- **Conoscere il ciclo di vita di un processo**
- **Conoscere i meccanismi di caricamento di caricamento del programma in memoria**
- Conoscere le tecniche di partizionamento della memoria
- Conoscere le tecniche di virtualizzazione della memoria: paginazione e segmentazione
- **Conosce i principali comandi da shell per la gestione dei file e delle directory**
- **Conoscere la differenza tra processi e thread**
- Conoscere i metodi di realizzazione di un thread
- **Conoscere il ciclo di vita di un thread**
- **Acquisire il concetto di programmazione concorrente**
- **Acquisire il concetto di interazione tra processi**
- Conoscere le caratteristiche di un linguaggio concorrente
- **Conoscere la funzione fork**
- **Conoscere i costrutti forkjoin e cobegin-coend**
- Conoscere le funzioni principali per l'implementazione dei thread
- Conoscere il modello ad ambiente locale e globale
- **Conoscere le proprietà richieste ai programmi concorrenti**
- **Conoscere la funzione dei semafori binari**
- **Conoscere la funzione dei semafori di Dijkstra**

- Conoscere il problema dei produttori/consumatori
- Conoscere il problema dei lettori/scrittori
- **Conoscere il problema del deadlock**
- Conoscere i monitor come soluzione ai deadlock
- Conoscere le funzioni delle librerie signal.h, e semaphore.h
- Conoscere il codice che permette di osservare la schedulazione dei thread
- Conoscere il codice che permette la condivisione di dati tra due thread attraverso l'uso dei semafori
- Conoscere le contion variable per la sospensione di un thread
- **Conoscere l'uso dei monitor in C**
- **Conoscere l'importanza della fase di analisi**
- **Conoscere il concetto di requisito utente e di sistema**
- **Conoscere il concetto di fase di esplorazione**
- **Conoscere il concetto di scenario e caso d'uso**
- Comprendere le caratteristiche di un SRS
- Sapere quali sono i documenti necessari in un progetto
- Conoscere il concetto di documentazione interna ed esterna
- **Conoscere i principali tool di documentazione automatica del codice**

COMPETENZE:

- **Saper classificare i sistemi operativi**
- Saper dire quale è stata l'evoluzione dei sistemi operativi negli anni
- **Saper descrivere il ciclo di vita di processo**
- **Comprendere le problematiche relative all'allocazione dei processi in memoria**
- **Saper utilizzare interfacce di tipo CUI**
- **Descrivere l'interazione processi-risorse col grafo di Holt**
- Realizzare e semplificare il grafo delle precedenze
- **Scrivere programmi concorrenti utilizzando l'istruzione fork-join**
- Scrivere programmi concorrenti utilizzando l'istruzione cobegin-coend
- **Eseguire un programma in Cygwin**
- **Scrivere un programma multiprocessore in linguaggio C**
- **Comprendere l'esigenza di sincronizzazione**
- Comprendere il concetto di indivisibilità di una primitiva
- Individuare le tipologie di errori nei processi paralleli
- Definire e utilizzare i semafori di basso livello
- Utilizzare gli strumenti di sincronizzazione per thread in C
- Utilizzare le condition variable in C
- Implementare i monitor in C
- **Saper individuare i requisiti utente**
- **Saper individuare i requisiti di sistema**
- Utilizzare le tecniche di esplorazione
- Individuare gli scenari d'uso
- Analizzare il documento di Specifica dei Requisiti Software (SRS)

- Acquisire la struttura di un SRA
- Saper organizzare la documentazione del progetto
- Saper definire uno standard di documentazione
- Saper formattare il codice
- Saper effettuare la documentazione del codice

CAPACITA':

- **Saper utilizzare in modo appropriato la terminologia tecnica**
- **Saper riconoscere le caratteristiche principali di un sistema operativo**
- **Saper riconoscere i vantaggi e svantaggi delle politiche di scheduling**
- **Saper individuare le problematiche di cooperazione tra processi**
- **Saper classificare le memorie**
- Saper proteggere i file
- **Sa eseguire operazioni su file e directory da linee di comando in Windows e in Linux**
- **Sa compilare un file in linguaggio c ed eseguirlo da linea di comando**
- Installare e configurare il software Cygwin
- **Compilare i programmi C col compilatore gcc**
- Utilizzare il linguaggio C in thread
- Saper spiegare il problema del deadlock e starvation
- Saper mettere a confronto i semafori binari con i semafori di Dijkstra
- **Saper applicare i semafori**
- Risolvere le situazioni di starvation
- Risolvere le situazioni di deadlock
- Risolvere i problemi di produttore/consumatore in C
- Risolvere il problema dei filosofi in C
- **Saper scrivere in UML i casi d'uso**
- Saper descrivere in UML il diagramma di contesto
- Saper documentare i casi d'uso
- Saper compilare il documento di Specifica dei Requisiti Software (SRS)
- Validare le specifiche di un SRS
- Installare e utilizzare Doxygen come strumento di documentazione automatica
- Installare e configurare Subversione TortoiseSVN
- Utilizzare TortoiseSVN per effettuare il controllo delle versioni

Obiettivi minimi

Gli obiettivi minimi sono indicati in grassetto sopra.

Metodologie e strategie didattiche

Il traguardo formativo potrebbe essere raggiunto privilegiando momenti di scoperte e di successiva generalizzazione a partire da casi semplici e stimolanti. Gli allievi vengono così impegnati in attività che favoriscono il consolidamento di meccanismi mentali di base. Si procederà per moduli didattici, verrà utilizzato il metodo induttivo. Gli argomenti trattati saranno studiati cercando sempre di seguire i ritmi di apprendimento degli alunni, ma senza perdere di vista obiettivi e finalità che vanno comunque conseguiti.

Il docente dovrà apparire come una guida e, fornendo agli allievi la sua esperienza, analizzerà con essi degli esempi concreti di applicazione di quanto si sta studiando, privilegiando così l'aspetto applicativo rispetto a quello teorico.

Sarà privilegiata la lezione dialogata, poiché tale tecnica consente rapidamente di valutare lo stato di apprendimento ed apportare tempestivamente azioni di recupero e correttive.

Mezzi e strumenti

- **Lezioni frontali:** Il docente descrive con l'aiuto degli strumenti disponibili, gli aspetti importanti dell'argomento trattato. Il docente non si limita all'esposizione degli argomenti di studio, ma stimola la partecipazione costruttiva della classe, privilegiando il metodo deduttivo e cercando quindi di far giungere passo-passo gli allievi stessi, ove sia possibile, alle conclusioni.
- **Verifiche formative orali:** Sono parte integrante dell'attività didattica, essenzialmente sono un momento in cui l'intera classe prende atto del proprio grado di apprendimento e interviene con domande critiche chiarificatrici, mentre il singolo alunno "interrogato" ha modo di esercitare le proprie abilità espressive, valutando, inoltre, la propria preparazione e correggendo, grazie all'intervento dell'insegnante, i propri errori.
- **Lavoro di gruppo:** Durante le attività di gruppo gli studenti tenderanno a sviluppare diverse strategie formative. L' attenzione principale sarà dedicata ai seguenti aspetti:
  a) La cooperazione. Gli studenti opereranno all'interno del gruppo per svolgere il lavoro loro affidato, aiutandosi vicendevolmente. Durante questa attività il compito del docente sarà quello di osservare il lavoro dei componenti i gruppi ed in qualche caso di partecipazione alle attività dei gruppi.
  b) Il confronto. Gli studenti affronteranno piccoli esercizi/progetti in modo "competitivo" con altri gruppi, al fine di evidenziare le proprie particolarità, gli aspetti di creatività e l'acquisizione di nuove competenze. Il ruolo del docente sarà maggiormente orientato alla gestione dei progetti ed al controllo della tempistica del lavoro da svolgere.
  c) Aspetti di professionalità. In questo caso saranno proposti, come attività di gruppo dei semplici problemi. La finalità del lavoro è quella di abituare a risolvere casi reali progressivamente più difficili. I gruppi saranno invitati a relazionare agli altri studenti il lavoro svolto.
- **Discussione:** Situazione di confronto su tematiche inerente agli argomenti trattati al fine di far emergere problemi, dubbi e contributi utili al rafforzamento dell'azione formativa.
- **Esercitazioni pratiche ed in laboratorio:** Il docente, dopo aver illustrato gli aspetti teorici dell'argomento, assegna agli allievi la realizzazione di un compito da svolgere al fine di produrre un risultato pratico. Questo aspetto è fondamentale per rafforzare l'identità e la vocazione occupazionale dell'allievo.
  Grazie alla pratica laboratoriale sono provate e sperimentate le abilità progettuali e realizzative acquisite.

Verifiche e valutazioni

Le verifiche formative saranno fatte durante lo svolgimento delle unità di apprendimento tramite domande a risposta multipla o aperta saranno inoltre somministrati esercizi on-line e off-line.

Le verifiche avranno lo scopo di guidare gli allievi e verificare il raggiungimento degli obiettivi ed eventualmente poter colmare le lacune.

Gli strumenti utilizzati saranno colloqui, interrogazioni ed esercitazioni al computer. Le verifiche sommative serviranno a valutare la competenza disciplinare acquisita e il raggiungimento degli obiettivi cognitivi prefissati. Gli strumenti utilizzati saranno verifiche orali individuali, verifiche scritte, test o risposte aperte e chiuse, soluzione di problemi.

Il numero delle verifiche sommative previste per il primo periodo è pari a tre (una scritta, una orale e una pratica); per il secondo pentamestre sono previste almeno due verifiche orali e due scritte e due pratiche.

Per la valutazione minima di sufficienza nei colloqui orali l'alunno deve dimostrare la conoscenza, anche se non approfondita dei concetti oggetto del programma, deve saperli esporre in modo semplice con termini appropriati e deve saper risolvere semplici esercizi e problemi. Inoltre nella valutazione complessiva scritta per il raggiungimento degli obiettivi minimi si terrà conto della completezza dell'elaborato, della correttezza, dell'organicità nell'esecuzione e della giustificazione delle procedure attuate, delle conoscenze, competenze e abilità acquisite da ogni singolo allievo in relazione ai livelli di partenza e ai livelli finali raggiunti, tenuto conto dell'impegno, della partecipazione e della costanza nello studio.

Criteri e metodi di valutazione

La misurazione delle prestazioni consentirà di acquisire informazioni continue ed analitiche sul modo in cui gli allievi procedono nell'apprendimento e potrà essere effettuata con prove strutturate e non strutturate (scritte ed orali).

Tipologia di verifiche e misurazione delle prestazioni

Verifica		Misurazione delle prestazioni	
1	Orale	Viene attribuito un punteggio in base alle capacità dello studente di esprimere correttamente le proprie conoscenze, motivandole con gli opportuni riferimenti ed utilizzando un linguaggio appropriato.	
2	Test	Quesiti a risposta multipla	Viene attribuito un punteggio in base al numero di risposte esatte, di quelle errate e di quelle a cui lo studente non ha risposto.
		Quesiti a risposta chiusa	Viene attribuito un punteggio in base al numero di risposte esatte, di quelle errate e di quelle a cui lo studente non ha risposto.
		Quesiti a risposta aperta	Viene attribuito un punteggio in base alla correttezza e alla completezza della risposta. Nelle richieste relative a questa tipologia sarà specificato il numero di righe entro il quale lo studente deve formulare la risposta ed inoltre sarò fornito un modello di risposta ottimale: "risposta criterio". Per le risposte parziali saranno precisati punteggi parziali.
		Trattazione sintetica	Viene attribuito un punteggio sulla base di griglie da stabilire

		di argomenti	in itinere.
3	Pratica	Viene attribuito un punteggio in base alla correttezza dell'impostazione risolutiva	

La valutazione costituisce un punto cruciale per tutta l'azione didattico educativa e non può semplicemente ridursi all'accertamento del profitto individuale dello studente classificandone il livello di apprendimento, bensì deve essere intesa come conoscenza che influisce direttamente sulle dinamiche del processo di insegnamento-apprendimento.

È uno strumento indispensabile per lo studente, infatti gli consente di prendere coscienza delle proprie potenzialità e di svilupparle in modo concreto e coerente incentivandone la motivazione allo studio ed alla partecipazione alle attività didattiche.

Per le griglie di valutazione si rimanda alla programmazione di dipartimento.

Strutturazione della programmazione disciplinare

La programmazione disciplinare è stata suddivisa nelle seguenti Macro-UDA, suddivisi a loro volta in unità di apprendimento più piccole per rendere più snella, sia la fase di trattazione che quella di verifica dell'avvenuta assimilazione degli argomenti affrontati.

n°	UDA	n° u.d.	Unità didattiche	tempi
1	Recupero dei prerequisiti: Sistemi operativi	1	Generalità sui sistemi operativi	3
		2	La gestione del processore	4
		3	La gestione della memoria	4
		4	Interfaccia CUI	4
2	I processi	1	Modello a processi, risorse e condivisione	6
		2	I thread	6
		3	Elaborazione sequenziale e concorrente	6
		4	Laboratorio: esecuzione parallela di processi e thread	12
3	Tecniche di programmazione concorrente	1	Comunicazione e sincronizzazione tra processi	7
		2	Problemi classici della programmazione concorrente	7
		3	Laboratorio: comunicazione tra processi mediante segnali asincroni	16
4	Progettazione software e documentazione del codice	1	I requisiti: raccolta, analisi e documentazione. Attori casi d'uso e scenari	12

		2	Documentazione del progetto e del codice	12

Descrizione analitica delle UDA

TITOLO: Recupero dei prerequisiti: Sistemi Operativi

Competenze:

- Saper classificare i sistemi operativi
- Saper dire quale è stata l'evoluzione dei sistemi operativi negli anni
- Saper descrivere il ciclo di vita di processo
- Comprendere le problematiche relative all'allocazione dei processi in memoria
- Saper utilizzare interfacce di tipo CUI

Abilità:

- Saper utilizzare in modo appropriato la terminologia tecnica
- Saper riconoscere le caratteristiche principali di un sistema operativo
- Saper riconoscere i vantaggi e svantaggi delle politiche di scheduling
- Saper individuare le problematiche di cooperazione tra processi
- Saper classificare le memorie
- Sa eseguire operazioni su file e directory da linee di comando in Windows e in Linux
- Sa compilare un file in linguaggio c ed eseguirlo da linea di comando

U. D. n°1 〈 Generalità sui sistemi operativi 〉

Competenze	*Conoscenze*	*Abilità*
▪ Saper classificare i sistemi operativi ▪ Saper dire quale è stata l'evoluzione dei sistemi operativi negli anni	▪ Sapere cosa succede all'accensione del PC ▪ Conoscere i compiti del sistema operativo ▪ Conoscere a storia dei sistemi operativi	▪ Saper utilizzare in modo appropriato la terminologia tecnica ▪ Saper riconoscere le caratteristiche principali di un sistema operativo

Contenuti	Tempi in ore	metodologia	Mezzi e strumenti
Il sistema operativo	3	- Lezioni frontali - Discussioni guidate	- Libro - Appunti - Laboratorio - PC
La struttura onion-skin			
I sistemi operativi in commercio			

Sistemi dedicati			**verifiche**
Sistemi interattivi			- Verifiche orali e scritte - Verifiche pratiche
Home computing			**collegamenti intedisciplinari**
Sistemi odierni e sviluppi futuri			- Sistemi - **UDA Interdisciplinare**

U. D. n°2 ⟨ La gestione del processore ⟩

Competenze	*Conoscenze*	*Abilità*
▪ Saper descrivere il ciclo di vita di processo	▪ Conoscere i meccanismi di caricamento di caricamento del programma in memoria	▪ Saper riconoscere i vantaggi e svantaggi delle politiche di scheduling ▪ Saper individuare le problematiche di cooperazione tra processi

Contenuti	Tempi in ore	metodologia	Mezzi e strumenti
Sistemi monoprogrammati e multiprogrammati	4	- Lezioni frontali - Discussioni guidate	- Libro - Appunti - Laboratorio - PC
I processi e i programmi			
Stati di un processo			
La scheduazione dei processi			**verifiche**
User mode e kernel mode			- Verifiche orali e scritte - Verifiche pratiche
I criteri di scheduling			**collegamenti intedisciplinari**
			- Sistemi

U. D. n°3 ⟨ la gestione della memoria ⟩

Competenze	*Conoscenze*	*Abilità*
▪ Comprendere le problematiche relative all'allocazione dei processi in memoria	▪ Conoscere le tecniche di partizionamento della memoria ▪ Conoscere le tecniche di virtualizzazione della memoria: paginazione e segmentazione	▪ Saper classificare le memorie ▪ Saper mettere a confronto

Contenuti	Tempi in ore	metodologia	Mezzi e strumenti
Caricamento di un programma	4	- Lezioni frontali	- Libro

Partizionamento della memoria		- Discussioni guidate	- Appunti - Laboratorio - PC
Memoria virtuale e paginazione			**verifiche**
Memoria virtuale e segmentazione			- Verifiche orali e scritte - Verifiche pratiche
			collegamenti intedisciplinari
			- Sistemi

U. D. n°4 〈 Interfaccia CUI 〉

Competenze	*Conoscenze*	*Abilità*
▪ Saper utilizzare interfacce di tipo CUI	▪ Conosce i principali comandi da shell per la gestione dei file e delle directory	▪ Sa eseguire operazioni su file e directory da linee di comando in Windows e in Linux ▪ Sa compilare un file in linguaggio c ed eseguirlo da linea di comando

Contenuti	Tempi in ore	metodologia	Mezzi e strumenti
La shell dei comandi di windows e di linux	4	- Lezioni frontali - Discussioni guidate	- Libro - Appunti - Laboratorio - PC
I comandi di gestione delle directory			
I comandi per i file			**verifiche**
Comandi di copia			- Verifiche pratiche
Comandi di rete			**collegamenti intedisciplinari**
Compilazione tramite comando gcc			- Sistemi

TITOLO: I processi

Competenze:

- Descrivere l'interazione processi-risorse col grafo di Holt
- Realizzare e semplificare il grafo delle precedenze
- Scrivere programmi concorrenti utilizzando l'istruzione fork-join
- Scrivere programmi concorrenti utilizzando l'istruzione cobegin-coend
- Eseguire un programma in Cygwin
- Scrivere un programma multiprocessore in linguaggio C

Abilità:

- Installare e configurare i software Cygwin

- Compilare i programmi C col compilatore gcc
- Utilizzare il linguaggio C in thread
-

U. D. n°1 〈 Modello a processi, risorse e condivisione 〉

Competenze	*Conoscenze*	*Abilità*
▪ Descrivere l'interazione tra processi con il grafo di holt ▪ Realizzare e semplificare il grafo delle precedenze	▪ Conoscere i modelli di elaborazione dei processi ▪ Conoscere il ciclo di vita dei processi ▪ Acquisire il concetto di risorsa condivisa ▪ Distinguere le richieste e le modalità di accesso alle risorse ▪ Apprendere l'utilizzo del grafo di Holt per descrivere processi e risorse	▪ Progettare e realizzare applicazioni che interagiscono con le funzionalità dei sistemi operativi ▪ Progettare e realizzare applicazioni in modalità concorrente

Contenuti	**Tempi in ore**	**metodologia**	**Mezzi e strumenti**
Modello a processi	6	- Lezioni frontali - Lezioni in laboratorio - Discussioni guidate	- Libro - Appunti - Laboratorio - PC
Stato dei processi			
Comandi per la creazione, sospensione e terminazione dei processi			
Risorse e condivisione			**verifiche** - Verifiche orali e scritte - Verifiche pratiche
Grafo di Holt			**collegamenti intedisciplinari** - Sistemi - Informatica

U. D. n°2 〈 I thread 〉

Competenze	*Conoscenze*	*Abilità*
▪ Descrivere l'interazione processi-risorse col grafo di Holt ▪ Realizzare e semplificare il grafo delle precedenze	▪ Conoscere la differenza tra processi e thread ▪ Conoscere i metodi di realizzazione di un thread ▪ Conoscere il ciclo di vita di un thread	▪ Utilizzare il linguaggio C in thread

Contenuti	**Tempi in ore**	**metodologia**	**Mezzi e strumenti**
I thread e loro utilizzo	6		

Processi ”pesanti” e “processi leggeri”		- Lezioni frontali - Lezioni di laboratorio - Discussioni guidate	- Libro - Appunti - Laboratorio - PC
Single threading vs Multithreading			
Realizzazione di thread e standard POSIX			**verifiche**
Stati di un thread			- Verifiche orali e scritte - Verifiche pratiche
			collegamenti intedisciplinari
			- Sistemi - Informatica

U. D. n°3 〈 Elaborazione sequenziale e concorrente 〉

Competenze	*Conoscenze*	*Abilità*
▪ Scrivere programmi concorrenti utilizzando l’istruzione fork-join ▪ Scrivere programmi concorrenti utilizzando l’istruzione cobegin-coend. ▪ Realizzare e semplificare il grafo delle precedenze	▪ Acquisire il concetto di programmazione concorrente ▪ Acquisire il concetto di interazione tra processi ▪ Conoscere le caratteristiche di un linguaggio concorrente	▪ Utilizzare il linguaggio C in thread

Contenuti	Tempi in ore	metodologia	Mezzi e strumenti
Elaborazione sequenziale e concorrenza	6	- Lezioni frontali - Lezioni in Laboratorio - Discussioni guidate	- Libro - Appunti - Laboratorio - PC
Processi non sequenziali e grafo di precedenza			
Scomposizione di un processo non sequenzial			**verifiche**
Esecuzione parallela : fork-join e cobegin-coend			- Verifiche orali e scritte - Verifiche pratiche
Semplificazione delle precedenze			**collegamenti intedisciplinari**
			- Sistemi

U. D. n°4 〈 Laboratorio: esecuzione parallela di processi e thread〉

Competenze	*Conoscenze*	*Abilità*
▪ Scrivere programmi concorrenti utilizzando l’istruzione fork-join ▪ Scrivere programmi concorrenti utilizzando l’istruzione cobegincoend	▪ Conoscere la funzione fork ▪ Conoscere i costrutti forkjoin e cobegin-coend ▪ Conoscere le funzioni principali per	▪ Installare e configurare i software Cygwin ▪ Compilare i programmi C col compilatore gcc

▪ Eseguire un programma in Cygwin ▪ Scrivere un programma multiprocessore in linguaggio C	l'implementazione dei thread	▪ Saper generare processi figlio con le funzioni fork ▪ Saper utilizzare i costrutti forkjoin e cobegin-coend ▪ Utilizzare il linguaggio C in thread

Contenuti	Tempi in ore	metodologia	Mezzi e strumenti
Emulatore Cygwin	12	- Lezioni frontali - Lezioni in Laboratorio - Discussioni guidate	- Libro - Appunti - Laboratorio - PC
La fork in C			
Fork-join e cobegin-coend			**verifiche**
I thread in C			- Verifiche orali e scritte - Verifiche pratiche
Thread e parametri			
La comunicazione tra processi mediante segnali asincroni			**collegamenti intedisciplinari**
Thread e schedulazione			- Sistemi - Informatica

TITOLO: Tecniche di programmazione concorrente

Competenze:

- Comprendere l'esigenza di sincronizzazione
- Comprendere il concetto di indivisibilità di una primitiva
- Individuare le tipologie di errori nei processi paralleli
- Definire e utilizzare i semafori di basso livello
- Utilizzare gli strumenti di sincronizzazione per thread in C
- Utilizzare le condition variable in C
- Implementare i monitor in C

Abilità:

- Saper spiegare il problema del deadlock e starvation
- Saper mettere a confronto i semafori binari con i semafori di Dijkstra
- Saper applicare i semafori
- Risolvere le situazioni di starvation
- Risolvere le situazioni di deadlock
- Risolvere i problemi di produttore/consumatore in C
- Risolvere il problema dei filosofi in C

U. D. n°1 ⟨ Comunicazione e sincronizzazione tra processi⟩

Competenze	*Conoscenze*	*Abilità*
▪ Comprendere l'esigenza di sincronizzazione ▪ Comprendere il concetto di indivisibilità di una primitiva ▪ Individuare le tipologie di errori nei processi paralleli ▪ Definire e utilizzare i semafori di basso livello ▪ Utilizzare gli strumenti di sincronizzazione per thread in C	▪ Conoscere il modello ad ambiente locale e globale ▪ Conoscere le proprietà richieste ai programmi concorrenti ▪ Conoscere la funzione dei semafori binari ▪ Conoscere la funzione dei semafori di Dijkstra	▪ Saper spiegare il problema del deadlock e starvation ▪ Saper mettere a confronto i semafori binari con i semafori di Dijkstra ▪ Saper applicare i semafori

Contenuti	**Tempi in ore**	**metodologia**	**Mezzi e strumenti**
La comunicazione tra processi	6	- Lezioni frontali - Lezioni in laboratorio - Discussioni guidate	- Libro - Appunti - Laboratorio - PC
La sincronizzazione tra processi			
I semafori: semafori binari vs semafori di Dijkstra			
Semafori e mutua esclusione			**verifiche** - Verifiche orali e scritte - Verifiche pratiche
Semafori come vincoli di precedenza			**collegamenti intedisciplinari** - Sistemi - Informatica

U. D. n°2 ⟨ Problemi classici della programmazione concorrente ⟩

Competenze	*Conoscenze*	*Abilità*
▪ Utilizzare gli strumenti di sincronizzazione per thread in C ▪ Utilizzare le condition variable in C ▪ Implementare i monitor in C	▪ Conoscere il problema dei produttori/consumatori ▪ Conoscere il problema dei lettori/scrittori ▪ Conoscere il problema del deadlock ▪ Conoscere i monitor come soluzione ai deadlock	▪ Risolvere le situazioni di deadlock ▪ Risolvere i problemi di produttore/consumatore in C ▪ Risolvere il problema dei filosofi in C

Contenuti	**Tempi in ore**	**metodologia**	**Mezzi e strumenti**
Problemi classici della programmazione concorrente: produttori/consumatori	6	- Lezioni frontali	- Libro

Problemi classici della programmazione concorrente: lettori/scrittori Problemi classici della programmazione concorrente: deadlock I monitor come soluzione ai deadlock		- Lezioni di laboratorio - Discussioni guidate	- Appunti - Laboratorio - PC **verifiche** - Verifiche orali e scritte - Verifiche pratiche **collegamenti intedisciplinari** - Sistemi

U. D. n°3 〈 Laboratorio: comunicazione tra processi mediante segnali asincroni 〉

Competenze	*Conoscenze*	*Abilità*
▪ Scrivere programmi concorrenti utilizzando l'istruzione fork-join ▪ Scrivere programmi concorrenti utilizzando l'istruzione cobegin-coend. ▪ Realizzare e semplificare il grafo delle precedenze	▪ Acquisire il concetto di programmazione concorrente ▪ Acquisire il concetto di interazione tra processi ▪ Conoscere le caratteristiche di un linguaggio concorrente	▪ Utilizzare il linguaggio C in thread

Contenuti	Tempi in ore	metodologia	Mezzi e strumenti
Elaborazione sequenziale e concorrenza Processi non sequenziali e grafo di precedenza Scomposizione di un processo non sequenzial Esecuzione parallela : fork-join e cobegin-coend Semplificazione delle precedenze	6	- Lezioni frontali - Lezioni in Laboratorio - Discussioni guidate	- Libro - Appunti - Laboratorio - PC **verifiche** - Verifiche orali e scritte - Verifiche pratiche **collegamenti intedisciplinari** - Sistemi

U. D. n°4 〈 Laboratorio: esecuzione parallela di processi e thread〉

Competenze	*Conoscenze*	*Abilità*
▪ Comprendere l'esigenza di sincronizzazione ▪ Comprendere il concetto di indivisibilità di una primitiva ▪ Individuare le tipologie di errori nei processi paralleli ▪ Definire e utilizzare i semafori di	▪ Conoscere le funzioni delle librerie signal.h, e semaphore.h ▪ Conoscere il codice che permette di osservare la schedulazione dei thread ▪ Conoscere il codice che permette la condivisione di dati tra due thread	▪ Risolvere le situazioni di starvation ▪ Risolvere le situazioni di deadlock ▪ Risolvere i problemi di produttore/consumatore in C ▪ Risolvere il problema dei filosofi in C

basso livello ▪ Utilizzare gli strumenti di sincronizzazione per thread in C ▪ Utilizzare le condition variable in C ▪ Implementare i monitor in C	attraverso l'uso dei semafori ▪ Conoscere le contion variable per la sospensione di un thread ▪ Conoscere l'uso dei monitor in C	

Contenuti	Tempi in ore	metodologia	Mezzi e strumenti
Comunicazione sincrona tra processi con l'uso della libreria signal.h	16	- Lezioni frontali - Lezioni in Laboratorio - Discussioni guidate	- Libro - Appunti - Laboratorio - PC
Thread e schedulazione			
I semafori binari in C			
La soluzione del deadlock dei filosofi in C			
I semafori in C con la libreria semaphore.h			**verifiche**
I monitor con le variabili condition in C			- Verifiche orali e scritte - Verifiche pratiche
			collegamenti intedisciplinari
			- Sistemi

TITOLO: Progettazione software e documentazione del codice

Competenze:

- Saper individuare i requisiti utente
- Saper individuare i requisiti di sistema
- Utilizzare le tecniche di esplorazione
- Individuare gli scenari d'uso
- Analizzare il documento di Specifica dei Requisiti Software (SRS)
- Acquisire la struttura di un SRA
- Saper organizzare la documentazione del progetto
- Saper definire uno standard di documentazione
- Saper formattare il codice
- Saper effettuare la documentazione del codice

Abilità:

- Saper scrivere in UML i casi d'uso
- Saper descrivere in UML il diagramma di contesto
- Saper documentare i casi d'uso
- Saper compilare il documento di Specifica dei Requisiti Software (SRS)
- Validare le specifiche di un SRS
- Installare e utilizzare Doxygen come strumento di documentazione automatica
- Installare e configurare Subversione TortoiseSVN

- Utilizzare TortoiseSVN per effettuare il controllo delle versioni

U. D. n°1 ⟨ I requisiti: raccolta, analisi e documentazione. Attori casi d'uso e scenari⟩

Competenze	*Conoscenze*	*Abilità*
▪ Saper individuare i requisiti utente ▪ Saper individuare i requisiti di sistema ▪ Utilizzare le tecniche di esplorazione ▪ Individuare gli scenari d'uso ▪ Analizzare il documento di Specifica dei Requisiti Software (SRS) ▪ Acquisire la struttura di un SRA	▪ Conoscere l'importanza della fase di analisi ▪ Conoscere il concetto di requisito utente e di sistema ▪ Conoscere il concetto di fase di esplorazione ▪ Conoscere le tecniche di fase di esplorazione ▪ Conoscere il concetto di scenario e caso d'uso ▪ Comprendere le caratteristiche di un SRS	▪ Saper scrivere in UML i casi d'uso ▪ Saper descrivere in UML il diagramma di contesto ▪ Saper documentare i casi d'uso ▪ Saper compilare il documento di Specifica dei Requisiti Software (SRS) ▪ Validare le specifiche di un SRS

Contenuti	**Tempi in ore**	**metodologia**	**Mezzi e strumenti**
La specifica dei requisiti	12	- Lezioni frontali - Lezioni in laboratorio - Discussioni guidate	- Libro - Appunti - Laboratorio - PC
Raccolta e analisi dei requisiti			
Attori, casi d'uso e scenari			
La documentazione dei requisiti			**verifiche**
			- Verifiche orali e scritte - Verifiche pratiche
			collegamenti intedisciplinari
			- Sistemi - Informatica

U. D. n°2 ⟨ Gestione e documentazione del codice concorrente ⟩

Competenze	*Conoscenze*	*Abilità*
▪ Saper organizzare la documentazione del progetto ▪ Saper definire uno standard di documentazione ▪ Saper formattare il codice ▪ Saper effettuare la documentazione	▪ Sapere quali sono i documenti necessari in un progetto ▪ Conoscere il concetto di documentazione interna ed esterna ▪ Conoscere i principali tool di	▪ Installare e utilizzare Doxygen come strumento di documentazione automatica ▪ Installare e configurare Subversione TortoiseSVN ▪ Utilizzare TortoiseSVN per effettuare

del codice	documentazione automatica del codice	il controllo delle versioni

Contenuti	Tempi in ore	metodologia	Mezzi e strumenti
La documentazione del progetto La documentazione del codice	12	- Lezioni frontali - Lezioni di laboratorio - Discussioni guidate	- Libro - Appunti - Laboratorio - PC **verifiche** - Verifiche orali e scritte - Verifiche pratiche **collegamenti intedisciplinari** - Sistemi

Scansione temporale

n°UDA	titolo	tempi	Periodo
1	Recupero dei prerequisiti: Sistemi operativi	15	Settembre-Ottobre
2	I processi	30	Ottobre-Novembre-Dicembre-Gennaio
3	Tecniche di programmazione concorrente	30	Gennaio-Febbraio-Marzo-Aprile
4	Progettazione software e documentazione del codice	24	Aprile-Maggio-Giugno

Cassino, 20/10/2021

I docenti

Margherita Lozza
Simona Cutone