

I CICLI

Agnese Andriani
Davide Giordano



Scuole, Classi e Indirizzi

L'attività è destinata alle classi seconde della Scuola Secondaria di Secondo Grado e si svolgerà negli ultimi mesi dell'Anno Scolastico. L'indirizzo scelto è il Liceo Scientifico - Scienze Applicate, dove lo studio dell'Informatica è introdotto fin dal primo anno.



Scuole, Classi e Indirizzi

- **Riferimenti Ministeriali**

“Lo studente è introdotto ai principi alla base dei linguaggi di programmazione e gli sono illustrate le principali tipologie di linguaggi e il concetto di algoritmo. Sviluppa la capacità di implementare un algoritmo in pseudo-codice o in un particolare linguaggio di programmazione, di cui si introdurrà la sintassi.”


- **Disciplina principale**

Informatica

Scuole, Classi e Indirizzi

L'attività può essere adattata e proposta a studenti di diverse età e indirizzi, a condizione che vengano apportate le opportune modifiche per adeguarla ai differenti livelli di apprendimento.

Dunque, tutto ciò può essere proposto senza modifiche ad una classe di un ITIS dello stesso periodo didattico



Motivazioni e Finalità

I concetti di cicli iterativi, come *while* e *for*, sono fondamentali nella programmazione e rivestono un ruolo cruciale nella risoluzione di problemi computazionali.

La loro comprensione è essenziale per sviluppare un pensiero algoritmico e affrontare con successo sfide in vari ambiti disciplinari.



Motivazioni e Finalità

Inoltre, è **fondamentale** acquisire una solida padronanza di questi strumenti fin dai primi anni di studio, per poter successivamente lavorare con strutture dati più complesse e affrontare tematiche avanzate.

L'attività

L'attività introduce i concetti di cicli iterativi *while* e *for*, spiegandone il funzionamento teorico e proponendo esercizi pratici per consolidare le conoscenze.



Prerequisiti

Abbiamo individuato quattro prerequisiti fondamentali, basati su conoscenze pregresse sviluppate nella secondaria di secondo grado, nel primo anno del biennio delle superiori (fonte MIUR):

Informatici

- Conoscenza dei concetti di base della programmazione: variabili, array, operatori logici e condizionali.
- Concetto di tempo di esecuzione di un programma
- Concetto di blocco di codice

Matematici

Elementi di matematica di base: numeri interi, operazioni aritmetiche, divisibilità.



Contenuti

La spiegazione dell'argomento deve avvenire attraverso una suddivisione della mole di lavoro in due blocchi distinti, che per comodità verranno ricondotte a due lezioni in aula (60 minuti):

- **Lezione 1:** approccio all'idea di ciclo con lavoro riconducibile facilmente al mondo di tutti i giorni
- **Lezione 2:** approccio al concetto di ciclo iterativo attraverso definizioni più rigorose e strutturate

Traguardi e Obiettivi

Secondo le **Indicazioni Nazionali del Ministero dell'Istruzione**, gli obiettivi formativi includono:



Comprendere i fondamenti teorici delle scienze dell'informazione

gli studenti devono acquisire una solida base teorica, che comprende la comprensione delle strutture di controllo come i cicli iterativi (*for*, *while*).

Acquisire padronanza degli strumenti informatici:

gli studenti devono essere in grado di utilizzare linguaggi di programmazione per implementare algoritmi che fanno uso di cicli iterativi, applicando tali strumenti alla risoluzione di problemi significativi.

Utilizzare strumenti informatici per la soluzione di problemi

gli studenti devono saper applicare i cicli iterativi nella programmazione per risolvere problemi complessi, sviluppando competenze pratiche nell'implementazione di algoritmi iterativi.

Traguardi e Obiettivi

Obiettivi di apprendimento

Analizziamo le difficoltà riscontrabili dagli studenti aiutandoci con la Tassonomia di SOLO e la sua divisione gerarchica in livelli. In particolare qui avremo:

- **Prestrutturale (livello di comprensione minima)**

Gli studenti riconoscono un ciclo iterativo, ma non sono ancora in grado di comprenderne il funzionamento o di applicarlo correttamente



Traguardi e Obiettivi

- **Unistrutturale (comprensione di un singolo aspetto)**

Gli studenti sono in grado di descrivere come funziona un ciclo iterativo in termini semplici, comprendendo un aspetto alla volta, come l'uso della variabile di controllo.

- **Multistrutturale (comprensione di più aspetti, ma separati)**

Gli studenti comprendono diversi aspetti del ciclo iterativo (ad esempio, condizione di terminazione, variabili di controllo, aggiornamento della variabile), ma non li collegano ancora in modo complesso.

Traguardi e Obiettivi

- **Relazionale (connessione tra i vari aspetti)**

Gli studenti sono in grado di comprendere come i vari elementi di un ciclo iterativo (come variabili, condizioni di uscita e l'aggiornamento) interagiscono tra loro in un contesto complesso e di risolvere problemi pratici utilizzando cicli iterativi.

- **Astratto Esteso (applicazione astratta e generalizzazione)**

Gli studenti sono in grado di applicare il concetto di cicli iterativi in situazioni nuove e astratte, resolvendo problemi complessi e adattando il ciclo iterativo a contesti diversi.

Materiali e Strumenti

È importante utilizzare strumenti differenti in base al tipo di lezione a cui ci stiamo dedicando. Dunque, avremo:

● **Lezione 1**

Fogli e schede preparate dal docente;

● **Lezione 2**

Fogli e schede preparate dal docente, LIM, Java Visualizer per la visualizzazione in tempo reale di un piccolo programma (foto seguente);



Materiali e Strumenti

Linguaggio

Come linguaggio la scelta è ricaduta su Java, data la sua estrema semplicità e intuitività, oltre al fatto che sono presenti numerosi programmi software che lo rendono un linguaggio facilmente insegnabile (si veda il funzionamento di Java Visualizer, che per l'appunto inseriamo negli strumenti necessari).

Materiali e Strumenti



Java Visualizer
(beta: [report a bug](#))

Write your Java code here:

```
1 public class ClassNameHere {  
2     public static void main(String[] args) {  
3  
4     }  
5 }
```

options

args: +command-line argument

stdin (also visualizes consumption of [StdIn](#))

Visualize Execution

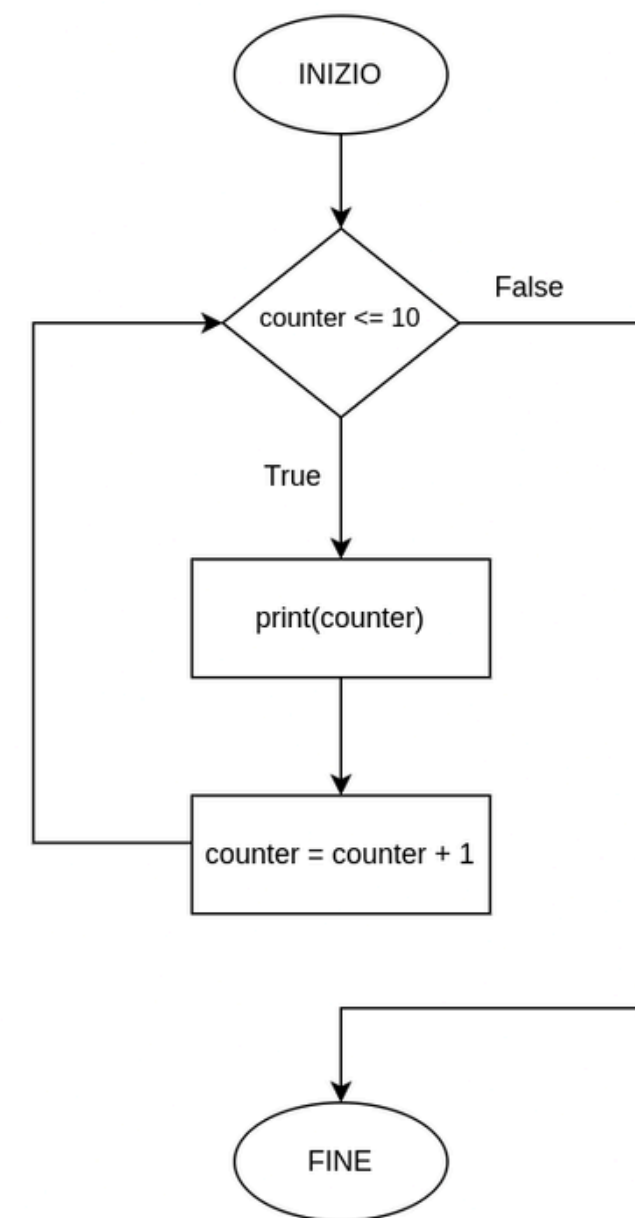


Sviluppo dei contenuti: Lezione 1



Lezione 1: Propedeutica

Il docente divide la classe in gruppi sfruttando la suddivisione nei banchi della classe: ogni studente lavorerà col proprio vicino, con possibilità di lavorare anche a gruppi di tre persone. Ogni gruppo avrà una fotocopia, composta da immagini e testo e nel quale sono trattate situazioni di tutti i giorni la cui gestione si può facilmente ricondurre a quella di un ciclo iterativo.



Lezione 1: Propedeutica

Il lavoro sulla scheda consiste nel compilare tre campi fondamentali per ciascuna delle task, che costituiscono rispettivamente tre elementi basilari di un ciclo di programmazione, senza però la richiesta di terminologie specifiche: l'obiettivo è infatti quello di far lavorare l'alunno sull'analisi dei vari elementi di situazioni quotidiane al fine di enfatizzare il concetto di operazione ripetuta.

Lo studente sarà dunque tenuto a individuare invariante di ciclo, condizione di ingresso/condizione di uscita ed eventuali task aggiuntive interne al ciclo stesso senza però che venga richiesta terminologia specifica.

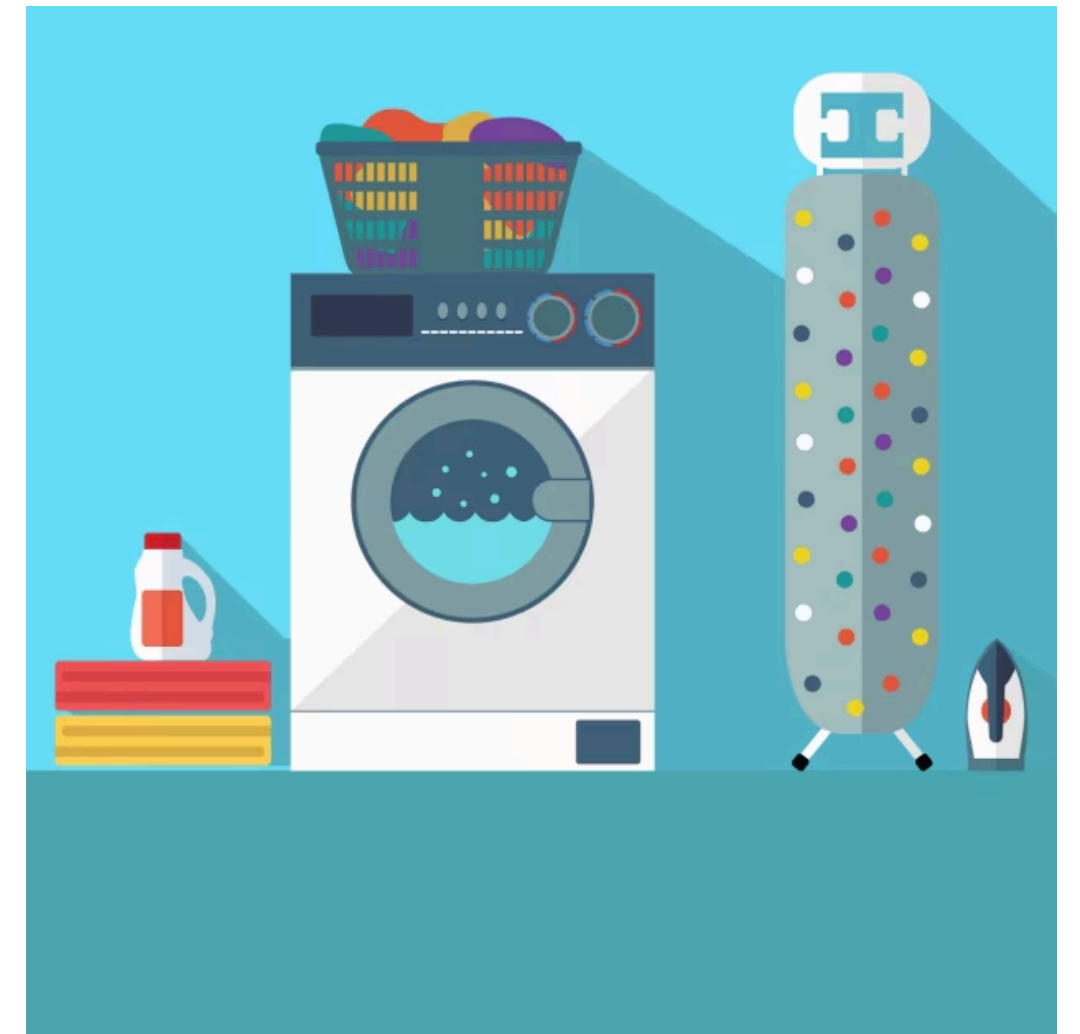
Lezione 1: Propedeutica

Ecco le tre diverse situazioni riportate di seguito:

- **Lavanderia**, in cui verrà introdotto il ciclo iterativo *while* e il suo funzionamento;
- **Squadra**, in cui verrà introdotto il *for* e le differenze con il ciclo precedente;
- **Magazzino**, che si concentra sull'introduzione di condizioni per ciclare;

Esercizio 1: Lavanderia

La signora Antonella è stata male e non ha potuto fare lavatrici per due settimane. Per di più, si è rotta la lavatrice finito il primo ciclo di lavaggi: ora si ritrova con dieci cesti pieni di vestiti da lavare ed è obbligata a portarli in un centro di lavaggio poco distante da casa sua. Ogni cesto è molto grosso e quindi può portarne solo una alla volta. Ogni volta che torna a casa, visto che è un po' golosa, passa dal panettiere per prendersi un dolcetto e posa lo scontrino sul tavolo.



Esercizio 1: Lavanderia

Definire quali sono le condizioni che portano Antonella a non tornare più al centro lavaggio e quali la obbligano ad andare. Inoltre rispondete alle domande:

- Quanti scontrini ci saranno sul tavolo alla fine dei lavaggi?
- Se ogni cesto contenesse vestiti divisi in due categorie (colorati e bianchi) e Antonella dovesse separare i vestiti prima di avviare i lavaggi al centro, quali azioni aggiuntive farebbe ogni volta?

Esercizio 2: Partitella



Davide è uno studente universitario che nel tempo libero allena la squadra di calcio del fratellino minore. Ad ogni allenamento, per far correre i ragazzi, distribuisce sul campo dei conetti e dei birilli creando un percorso che termina con un tiro al portiere. La sua squadra è composta da 7 giocatori e ognuno di essi parte quando l'altro ha calciato in porta. Si continua a correre finché tutti i bimbi hanno segnato almeno un gol.



Esercizio 2: Partitella

Definire condizioni di inizio e fine dell'allenamento, rispondendo anche alle seguenti domande:

- Quante palle almeno ci saranno alla fine della corsa nella rete?
- Se il portiere è davvero bravo e para sempre tutto, per quanto correranno i bambini?



Esercizio 3: Magazzino

Mario è un magazziniere che deve scaricare 15 scatoloni dal suo camion. Gli scatoloni sono molto pesanti, quindi può trasportarne solo uno alla volta. Ogni volta che torna al magazzino dopo aver scaricato uno scatolone, segna il trasporto completato su un registro appeso alla parete.



Esercizio 3: Magazzino

Definire quali sono le condizioni che portano Mario a non dover più scaricare gli scatoloni dal camion e quali lo obbligano a continuare. Inoltre, rispondete alla seguente domanda:

- Quanti segni ci saranno sul registro alla fine del lavoro?
- Se un collega lo aiutasse e potessero trasportare in due tre scatoloni alla volta, quanti giri sarebbero necessari per finire il lavoro?

Lezione 1: Propedeutica

Tempo stimato

- 10 minuti per il professore (spiegazione e eventuali domande)
- 20 minuti (flessibili) per svolgere il lavoro
- 30 per il confronto sulle risposte chiedendo per ogni domanda a uno o più gruppi.

Obiettivi da raggiungere

Aver introdotto la nozione di ciclo e operazione ciclica, primo approccio con gli elementi costitutivi di un ciclo e le condizioni limite che si porta dietro;



Sviluppo dei contenuti: Lezione 2



Lezione 2: Applicazioni pratiche

La lezione si struttura inizialmente in maniera frontale, con il docente che spiega la struttura dei cicli *for* e *while* in Java scendendo nei particolari e analizzando ogni elemento fondamentale nella costruzione di un'iterazione, utilizzando porzioni di codici semplici ma esemplificative e facili da comprendere e sfruttando software come Java Visualizer per rendere ancora più semplice la spiegazione della sequenzialità dell'esecuzione.



Lezione 2: Applicazioni pratiche

Successivamente, vengono distribuite nuove schede su cui lavorare con alcuni esercizi di programmazione con tematiche simili a quelli della lezione precedente (per favorire il riconoscimento di pattern familiari). Gli esercizi, che sono 3, verranno svolti in questo modo:

- **Cestini** (Primo esercizio): guidato dall'insegnante;
- **Squadra** (Secondo esercizio): in autonomia;
- **Super Mario** (Terzo esercizio): compito per casa;



Esercizio 1: Cestini

Scrivi un programma per contare quante volte Antonella passa dal panettiere e posa uno scontrino sul tavolo.

Requisiti:

Usa una variabile numeroCesti per rappresentare i 10 cestini.

Si usi un ciclo for per simulare il processo.

Terminazione

A terminale devono essere stampati:

- Un messaggio per ogni viaggio (Es. “Antonella posa lo scontrino)
- Numero scontrini

Esercizio 1: Cestini (Soluzione)

```
public static void main(String[] args) {  
    int numeroCesti = 10;  
    int scontrini = 0;  
  
    for (int i = 0; i < numeroCesti; i++) {  
        System.out.println("Antonella ha posato lo scontrino " + scontrini + " sul tavolo.");  
        scontrini++;  
    }  
  
    System.out.println("Gli scontrini alla fine sono: " + scontrini);  
}
```

Esercizio 2: Squadre

Marco e Andrea sono i giocatori migliori delle squadre della scuola. Alla fine del campionato, i due team sono a pari punti. Così, Marco decide di usare un programma per calcolare automaticamente il totale dei gol delle squadre, in modo da capire chi abbia vinto il campionato

Requisiti:

Si utilizzi un ciclo *while* per risolvere il problema.

Si faccia in modo di inserire i dati delle partite da riga di comando, per renderlo il più scalabile possibile

Suggerimenti

Usate questi dati (COPIATE IL CODICE)

```
int[] golSquadra1 = {10, 4, 8};
```

```
int[] golSquadra2 = {17, 2, 1};
```


Esercizio 2: Squadra (Soluzione)

```
public static void main(String[] args) {  
    // CODICE DATO DAL PROFESSORE  
    int[] golSquadra1 = {10, 4, 8};  
    int gol1 = 0;  
    int[] golSquadra2 = {17, 3, 2};  
    int gol2 = 0;  
  
    // CICLO CALCOLO GOLSQUADRA1  
    for (int i = 0; i < golSquadra1.length; i++) {  
        gol1 = gol1 + golSquadra1[i];  
    }  
  
    //CICLO CALCOLO GOLSQUADRA2  
    for (int i = 0; i < golSquadra2.length; i++) {  
        gol2 = gol2 + golSquadra2[i];  
    }  
  
    //CONFRONTO RISULTATI  
    if (gol1 == gol2) {  
        System.out.println("Differenza reti UGUALE");  
    } else if (gol1 > gol2) {  
        System.out.println("Vince SQUADRA1 con: " + gol1);  
    } else {  
        System.out.println("Vince SQUADRA2 con: " + gol2);  
    }  
}
```

Esercizio 3: Super Mario

Luigi è un grandissimo appassionato di videogiochi e il suo preferito è Super Mario Bros. A lezione ha scoperto di essere in grado di simulare il funzionamento del suo gioco del cuore, ma purtroppo giocando nell'intervallo si è infortunato alla mano. Aiutalo tu! Per completarlo, ti basteranno tre variabili: Vita, danno e turno.

Requisiti:

Vita: Deve essere impostata a un valore iniziale (es. 100 punti salute)

Danno: rappresenta il danno subito da Mario quando viene colpito da un nemico. Deve essere un valore positivo e diverso ad ogni turno.

Turno: Rappresenta l'avanzamento nel gioco o il ciclo in cui si verifica un'azione.

Terminazione

Stampa un messaggio a ogni turno che mostri il numero del turno, i punti vita attuali di Mario e il danno subito.



Esercizio 3: Super Mario

Suggerimenti

Usate questi dati (copiate il codice, la libreria è da importare e non da scrivere nel main)

```
import java.util.Random; // libreria da utilizzare
```

```
int danno = 0;
```

```
Random random = new Random(); // crea un'istanza della classe Random no usages
```

```
danno = random.nextInt(10) + 1; // Genera un numero tra 1 e 10
```

Esercizio 3: Super Mario(Soluzione)

```
import java.util.Random;

public class Main {
    public static void main(String[] args) {
        int vita = 100;
        int turno = 1;
        int danno = 0;

        Random random = new Random();

        System.out.println("Benvenuto, Mario inizia con 100 punti vita");

        while(vita > 0){
            danno = random.nextInt(10) + 1; // Genera un numero tra 1 e 10

            vita -= danno;

            if(vita > 0){
                turno++;
                System.out.println("Turno: " + turno + "Mario ha " + vita + " punti vita");
            }else{
                System.out.println("Mario è stato sconfitto");
            }
        }
    }
}
```

Lezione 2: Dettaglio

Tempo stimato

- 20 minuti per il professore (spiegazione della struttura dei cicli e eventuali domande)
- 20 minuti per il lavoro sulle schede da parte degli studenti
- 20 minuti per il lavoro sulla stesura di codice

Obiettivi da raggiungere

Comprendere in maniera completa il funzionamento dei cicli, sviluppare capacità di problem solving e astrazione dei problemi.



Guida per insegnanti



Guida per gli insegnanti

Consigli su come utilizzare il materiale didattico

Stampare e distribuire le schede con le situazioni (Lavanderia, Partitella, Magazzino).

Assicurarsi di avere accesso a un proiettore e a un software come Java Visualizer per la lezione frontale.

Prevedere una copia delle soluzioni degli esercizi per guidare la correzione in classe.

Guida per gli insegnanti

Consigli su come utilizzare il materiale didattico

Analizzare eventuali errori comuni e risolverli insieme, evidenziando buone pratiche.

Durante la spiegazione, utilizzare esempi concreti presenti nella scheda, collegando ogni elemento narrativo a un concetto teorico (es. “Antonella prende un dolcetto → incremento di una variabile”).

Guida per gli insegnanti

Lezione 1: Propedeutica

Gli esercizi proposti saranno tre:

- Lavanderia
- Partitella
- Magazzino

Da svolgere in autonomia con correzione collettiva. Analizzare eventuali errori comuni e risolverli insieme.

NB!

E' importante che l'insegnante sia chiaro nel fornire le istruzioni per lo svolgimento del lavoro con le schede e che dia spazio a tutti i gruppi nella fase finale di confronto sulle risposte.

Lezione 1: Approccio didattico

L'approccio didattico usato è il POGIL, infatti abbiamo:

Apprendimento attivo e guidato

Gli studenti lavorano su situazioni pratiche e quotidiane, che richiedono un'analisi autonoma e una riflessione sul problema. Questo è coerente con il principio POGIL, in cui gli studenti costruiscono attivamente la loro conoscenza.

Collaborazione in piccoli gruppi

La divisione in gruppi di 2-3 studenti rispecchia la struttura tipica di POGIL, dove ogni studente ha un ruolo attivo e contribuisce al raggiungimento degli obiettivi del gruppo.



Lezione 1: Approccio didattico

Guida dell'insegnante

L'insegnante agisce da facilitatore, fornendo un framework strutturato (le fotocopie con le situazioni) che aiuta gli studenti a esplorare i concetti iterativi, senza spiegazioni dirette o lezioni frontali.

Obiettivi cognitivi e metacognitivi

L'attività mira non solo a sviluppare competenze tecniche (come l'identificazione di cicli iterativi e condizioni di uscita), ma anche abilità di analisi, pensiero critico e collaborazione.



Lezione 1: Approccio didattico

Focus sui processi fondamentali

Anche senza usare la terminologia tecnica (invariante di ciclo, condizione di uscita), l'attività porta gli studenti a identificare questi concetti attraverso la pratica. Questo approccio riflette il principio di POGIL di concentrarsi sui processi essenziali.

Situazioni reali e significative

Le situazioni di vita quotidiana rendono l'attività pertinente e accessibile, un altro pilastro fondamentale di POGIL.



Guida per gli insegnanti

Lezione 2: Applicazione pratiche

Viene richiesto agli studenti di scrivere porzioni di codice che utilizzino cicli, facendo riferimento alla situazione descritta nella scheda.

Durante l'attività l'insegnante deve monitorare il lavoro degli studenti e chiarire eventuali dubbi. Infine, è bene guidare gli studenti verso il collegamento tra le situazioni proposte e i concetti di base del ciclo iterativo.

Lezione 2: Approccio didattico

L'approccio didattico usato è PRIMM, infatti abbiamo:

Sequenzialità delle fasi

La lezione è strutturata in modo progressivo, partendo dall'analisi guidata (Predict, Run, Investigate) e conducendo gli studenti verso una maggiore autonomia (Modify, Make).

Focus sulla comprensione

L'uso di Java Visualizer e l'analisi dettagliata dei cicli si allinea al principio di PRIMM di enfatizzare la comprensione prima della produzione.



Lezione 2: Applicazioni pratiche

Esercizi graduati

La progressione da un esercizio guidato a uno autonomo e infine a un compito a casa è coerente con il graduale trasferimento di responsabilità che PRIMM promuove.

Di seguito ripercorriamo i passi dell'approccio:

- Predict
- Run;
- Investigate
- Modify;
- Make;

Lezione 2: Fasi di PRIMM

Predict

Durante la lezione frontale, il docente analizza i cicli for e while, spiegandone i dettagli e mostrando esempi semplici. L'uso di strumenti come Java Visualizer aiuta gli studenti a immaginare e prevedere il comportamento del codice.

Run

Attraverso il software di visualizzazione e le porzioni di codice esemplificative, gli studenti possono osservare l'esecuzione del codice e vedere come funziona la sequenzialità nei cicli (= elemento chiave del metodo PRIMM).

Lezione 2: Fasi di PRIMM

Investigate

L'esercizio guidato sui **Cestini** rientra in questa fase. Con il supporto dell'insegnante, gli studenti analizzano il funzionamento del ciclo iterativo e cercano di comprendere i vari elementi (condizione, invariante, ecc.).

Modify

L'esercizio **Squadra**, svolto in autonomia, può essere visto come un passo verso la fase di modifica. Gli studenti applicano quanto appreso per affrontare un problema simile, adattando le loro conoscenze a un nuovo contesto.

Lezione 2: Fasi di PRIMM

Make

L'esercizio Super Mario, assegnato come compito a casa, rappresenta la fase in cui gli studenti devono creare un programma più complesso da zero, consolidando le competenze apprese. Questo è un chiaro esempio della fase Make.



Valutazione dell'apprendimento



Valutazione dell'apprendimento

Durante le lezioni verrà utilizzata la **valutazione formativa**:
gli studenti ricevono un feedback immediato durante lo svolgimento degli
esercizi in gruppo.

Il docente fornirà correzioni o suggerimenti che aiuteranno gli studenti a
riflettere sui propri errori e a comprendere come migliorare il proprio
approccio.

Questo li guiderà nella risoluzione corretta degli esercizi, promuovendo anche
l'autovalutazione e le capacità di analisi e problem solving.

Esercizio 1: Cestini (Block Model)

Il codice del primo esercizio rispetta le regioni **AT** (Atoms-Text Surface) e **BP** (Blocks-Program Execution) del Block Model.

- La regione AT rappresenta tutte le operazioni atomiche, nel nostro caso l'incremento della variabile scontrini;
- La regione BP si occupa della struttura di controllo del programma e dell'ordine di esecuzione dei blocchi. Nel codice, questa regione è rappresentata dal ciclo *for* e dalla sua logica di esecuzione delle istruzioni;

In questo primo esercizio si vuole presentare un codice semplice e lineare che possa fornire una solida base per esercizi più complessi.

Esercizio 2: Squadra (Block Model)

Il codice del primo esercizio rispetta le regioni **BT** (Blocks /Text surface), **AF** (Atoms/Functions) e **RP**(Relations/Program Execution) del Block Model.

- Ogni ciclo o blocco condizionale rappresenta una regione coerente che realizza un sotto-obiettivo (B) e la rappresentazione del codice è chiara e traduce direttamente i requisiti in operazioni di esecuzione e controllo del flusso (T);
- I blocchi sono collegati da un flusso chiaro di dati. Le relazioni tra cicli di calcolo e il confronto sono ben definite. Il flusso di esecuzione segue una sequenza logica e lineare, con aggiornamenti progressivi dei dati (gol1, gol2);
- Troviamo dichiarazioni, operazioni aritmetiche, confronti e chiamate a metodo (A). Inoltre, le funzioni di ciascun atomo sono chiare e contribuiscono allo scopo generale del programma.

Esercizio 3: Super Mario (Block Model)

Il codice del terzo esercizio rispetta le regioni **RF** (Relations-Functions), **MT** (Macro Structure-Text Surface) e **RP** (Relations-Program Execution) del Block Model.

- La regione RF si concentra sulle relazioni tra le diverse funzioni del programma e su come queste contribuiscano al raggiungimento degli obiettivi. Ogni parte del codice è collegata e lavora per il raggiungimento di uno scopo comune: simulare il calo dei punti vita di Mario.
- La regione MT si occupa della struttura complessiva del programma, a partire dall'inizializzazione delle variabili, passando all'esecuzione del ciclo *while*. Questa struttura si riflette su quello che sarà l'output prodotto.
- La regione RP si occupa del legame tra lo stato interno del programma e il suo comportamento durante l'esecuzione. Nel nostro caso lo stato del programma è rappresentato dalle variabili *vita*, *danno* e *turno*. L'output prodotto riflette lo stato del programma dopo ogni iterazione, fornendo i punti vita, il turno giocato e i danni ricevuti.

Rubrica valutativa (1)

	Indicatori	Descrittori	Livelli
Comprensione teorica	Capacità di comprendere e descrivere i concetti di ciclo (while, for) e i relativi elementi (condizioni, variabili di controllo, iterazione).	Descrive con chiarezza i cicli, collegando tutti gli elementi in modo approfondito.	4
		Comprende i cicli e descrive i principali elementi in modo completo.	3
		Riconosce i cicli e descrive uno o due elementi con chiarezza.	2
		Non riconosce i cicli o ne dà una descrizione confusa e incompleta.	1

Rubrica valutativa (2)

	Indicatori	Descrittori	Livelli
Rispetto dei tempi	Capacità di organizzare il lavoro in modo da rispettare le scadenze previste per ciascuna fase dell'attività.	Organizza il lavoro in modo efficiente, rispettando pienamente i tempi previsti.	4
		Completa la maggior parte delle attività nei tempi previsti, con lievi ritardi.	3
		Rispetta parzialmente i tempi, completando solo alcune parti entro le scadenze.	2
		Non rispetta i tempi previsti, con significativi ritardi o disorganizzazione.	1

Rubrica valutativa (3)

	Indicatori	Descrittori	Livelli
Applicazione pratica	Capacità di tradurre i concetti teorici in porzioni di codice funzionante e coerente con la traccia fornita.	Il codice è completo, funzionante, ottimizzato e ben strutturato.	4
		Il codice è funzionante, con lievi imprecisioni o limitata ottimizzazione.	3
		Il codice è parzialmente corretto, ma presenta errori logici o sintattici che influiscono sul risultato.	2
		Il codice è incompleto o non funzionante.	1

Rubrica valutativa (4)

	Indicatori	Descrittori	Livelli
Autovalutazione	Capacità di analizzare il proprio lavoro e correggere errori basandosi sui feedback ricevuti o sull'autoanalisi critica.	Individua e corregge autonomamente errori complessi, migliorando notevolmente il proprio lavoro.	4
		Identifica gli errori principali e li corregge in modo adeguato.	3
		Individua solo alcuni errori con difficoltà e necessita di aiuto per correggerli.	2
		Non è in grado di individuare errori né migliorare il lavoro.	1

Rubrica valutativa (5)

	Indicatori	Descrittori	Livelli
Uso del linguaggio specifico	Capacità di utilizzare correttamente termini tecnici e specifici relativi al contesto della programmazione e dei cicli iterativi.	Utilizza un linguaggio tecnico preciso e appropriato in modo costante e fluido.	4
		Utilizza correttamente i principali termini tecnici relativi al contesto dell'attività.	3
		Utilizza alcuni termini tecnici correttamente, ma in modo limitato.	2
		Non utilizza termini tecnici o li usa in modo errato.	1

Rubrica valutativa (6)

	Indicatori	Descrittori	Livelli
Creatività	Capacità di proporre soluzioni originali o approcci alternativi nell'implementazione delle attività.	Propone soluzioni innovative, dimostrando un approccio personale e un pensiero creativo nell'intero lavoro.	4
		Dimostra creatività proponendo soluzioni personali e applicando approcci originali in alcune parti.	3
		Dimostra un minimo di creatività, ma le soluzioni rimangono limitate o non originali.	2
		Non dimostra creatività, limitandosi a copiare o seguire rigidamente gli esempi forniti.	1

Rubrica valutativa (7)

	Indicatori	Descrittori	Livelli
Interazione orizzontale (con i compagni)	Capacità di collaborare efficacemente con i compagni, contribuendo alla risoluzione dei problemi e al confronto costruttivo.	Contribuisce in modo determinante al lavoro di gruppo, favorendo la collaborazione e il confronto costruttivo.	4
		Collabora attivamente con i compagni, proponendo soluzioni e ascoltando le idee degli altri.	3
		Collabora in modo limitato, interagendo solo quando sollecitato.	2
		Non collabora con i compagni o limita la comunicazione al minimo, senza contribuire significativamente al lavoro di gruppo.	1

Rubrica valutativa (8)

Giudizi	Voti	Livelli
Avanzato	9/10	4
Adeguito	7/8	3
Base	6	2
Non raggiunto	≤ 5	1



Misconception



Misconception

Durante il lavoro, gli studenti potranno incappare in ben più di un problema. Dunque, abbiamo diviso le possibili misconception in due macroaree:

- Misconception generali sui cicli;
- Misconception specifiche per gli esercizi suggeriti;

Misconception generali

Di seguito ecco alcune delle misconception generali in cui si può incappare nello studio dei cicli iterativi:

- utilizzare i cicli *for* e *while* in modo intercambiabile: i cicli *for* sono usati quando il numero di iterazioni è noto, mentre i cicli *while* sono più adatti quando non si sa in anticipo quante iterazioni servano;
- dimenticare di aggiornare la variabile di controllo (nei cicli *while*)
- modificare la variabile di controllo nel ciclo *for*
- pensare che la condizione venga verificata solo una volta all'inizio

Misconception specifiche

Di seguito ecco alcune delle misconception specifiche per gli esercizi suggeriti:

- Non sapere come usare un ciclo *for* per iterare esattamente n volte. Ad esempio, potrebbero iniziare il ciclo da 1 e terminare a n inclusivo, dimenticando che i contatori in molti linguaggi iniziano da 0. (EX1)
- Potrebbero non stampare il numero finale degli scontrini separatamente, includendolo magari in ogni messaggio iterativo. E' importante chiarire che l'output totale degli scontrini va mostrato solo una volta, dopo il ciclo. (EX2)

Misconception specifiche

- Gli studenti potrebbero non capire come scorrere gli array utilizzando un indice o accedere correttamente ai loro elementi. Ad esempio, potrebbero utilizzare un indice fuori dai limiti dell'array. Oppure, confondere i due array (golSquadra1 e golSquadra2) mescolando i dati. Gli array devono essere iterati con attenzione, mantenendo l'indice entro i limiti e distinguendo chiaramente i dati delle due squadre. (EX2)

Misconception specifiche

- Gli studenti potrebbero non implementare correttamente la terminazione del gioco, ad esempio lasciando che "Vita" scenda sotto 0 o non terminando il ciclo quando Mario muore. Il gioco dovrebbe terminare immediatamente quando "Vita" scende a 0 o al termine di un numero prefissato di turni. (EX3)



Grazie per l'attenzione!

