3 - Algoritmo

Ricordiamo cosa è un algoritmo:
Una serie di prescrizioni o istruzioni che specifica l'insieme delle azioni da compiere per poter risolvere un problema

Proprietà di un algoritmo

- **Sequenzialità**: in ogni algoritmo è presente una sequenza, un ordine specificato di istruzioni
- **Determinismo**: Tutti i passi devono essere eseguiti in modo univoco e descritti in una forma eseguibili per l'esecutore
- **Generalità**: La soluzione non deve dipendere da valori predefiniti di dati
- **Terminazione**: L'algoritmo deve sempre terminare, il suo stesso esecutore deve terminare in tempo finito per ogni insieme di valore di ingresso e l'insieme (finito) di istruzioni, infine la stessa istruzione deve essere eseguita in un numero finito di volte
- **Realizzabilità**: L'esecutore deve riuscire a eseguire l'algoritmo con le risorse a sua disposizione
- **Non ambiguità:** L’algoritmo non deve essere costituito da azioni che si contraddicono

Un algoritmo consente di trasformare dati in informazioni e deve descrivere come risolvere una classe di problemi

Un algoritmo può essere

- **Deterministico**: lo stesso algoritmo eseguito più volte sugli stessi dati di input, l'esecutore deve generare sempre gli stessi dati di output
- **Probabilistica**: lo stesso algoritmo eseguito più volte sugli stessi dati di input, l'esecutore deve generare dati di output sempre diversi
  Se si può specificare un algoritmo allora si può automatizzare la stessa soluzione

Processo d'esecuzione

È l'applicazione di un metodo solutivo ad una situazione problematica, cioè l'esecuzione delle operazioni previste dal metodo solutivo, può essere delegato ad un processore diverso dell'estensore del metodo solutivo come

- Essere umano,
- Sistema meccanico
  Un processo esecutivo per poter essere eseguito deve essere perfettamente descritto all'esecutore per poter effettivamente essere eseguito, altrimenti si incontrano delle interpretazioni ambigue e un comportamento non uniforme

L'esecuzione dell'algoritmo non è concorrente ma sequenziale, ma pur essendo sequenziale è possibile prevedere strade alternative da seguire al presentarsi di una certa condizione. Per evitare incomprensioni, la descrizione dell'esecuzione del processo deve definire:

- Gli oggetti su cui operiamo
- La sequenza, dalla prima all'ultima operazione, delle azioni da compiere
- La specifica dei controlli che determinano l'ordine di esecuzione delle azioni, indipendentemente dalla natura dell'esecutore

Programmazione strutturata

La programmazione strutturata è una metodologia di programmazione che è vincolata a due principi fondamentali:

- Sviluppo dell'algoritmo **top-down**
- Le istruzioni che compongono un algoritmo devono essere eseguite in un ordine, stabilito dalle strutture di controllo

Le strutture di controllo possono essere viste come schemi di composizione nel senso che possono essere combinate tra loro

- Nell'iterazione potrebbe essere una struttura di selezione (if)

La programmazione strutturata stabilisce che si possono avere unicamente un punto di ingresso e di uscita

Metodologia top-down

Questa metodologia (dall'alto verso il basso) prevede la scomposizione del problema da risolvere in più sotto problemi, per poi essere ricomposti formando una soluzione al problema originale

Linguaggi di descrizione degli algoritmi

Il linguaggio con cui descrivere gli algoritmi **non è** un linguaggio di programmazione. Esistono linguaggi grafici che fanno uso di simboli grafici e linguaggi lineari con i quali un algoritmo viene descritto con un linguaggio umano, un esempio sono i flowchart oppure gli alberi di decomposizione

Diagrammi di flusso

Linguaggio tipicamente utilizzato per trasmettere ad un esecutore umano la descrizione di un algoritmo o processo tramite un grafo, si parte dal punto iniziale seguendo percorsi indicati, intraprendendo le azioni che si incontrano, altrimenti nel caso di percorsi alternativi se ne sceglie uno a seconda della condizione specificata fino al raggiungimento del punto finale.

Il diagramma di flusso viene utilizzato perchè permette di descrivere il flusso di esecuzione in modo grafico su due dimensioni con pochi simboli

Un diagramma di flusso è composto da:

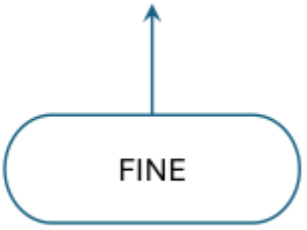
- Blocchi elementari, che descrivono azioni e decisioni
- Archi orientati, che collegano i vari blocchi e la sequenza di svolgimento delle azioni

I blocchi elementari possono essere:

- Blocchi di inizio
- Blocchi di fine
- Blocchi di connessione
- Blocchi di azione generica
- Blocchi di azione di I/O
- Blocchi di decisione binaria (oppure condizionale a due vie)

Blocchi di inizio e fine

Un algoritmo (e la sua rappresentazione grafica) presenta sempre un inizio e una fine, tra queste due ci devono essere sempre almeno un'istruzione

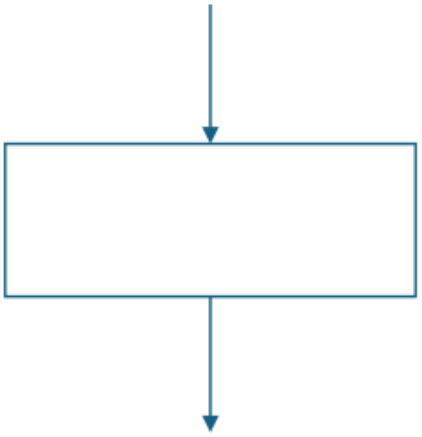


Blocco di connessione

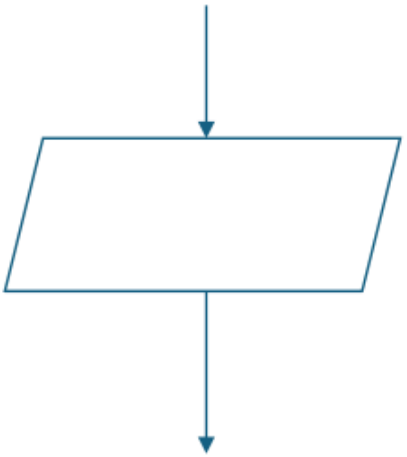
La risoluzione di un problema consiste nell'esecuzione ordinata di una sequenza di operazioni, quest'ordine è fondamentale e nel flowchart è garantito dall'orientamento delle frecce che collegano i blocchi

Blocchi di azione e I/O

✓Azione/Istruzione generica

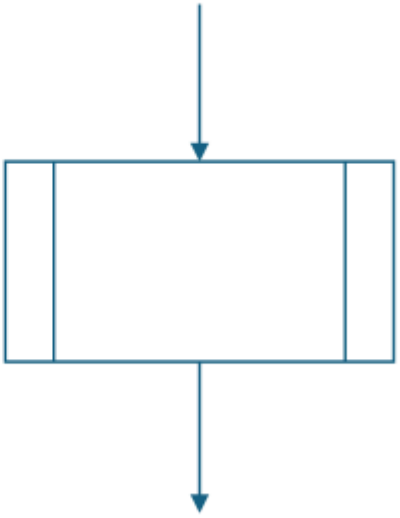


✓Azione I/O



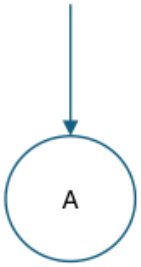
Blocchi sottoproblema/gruppo di azioni

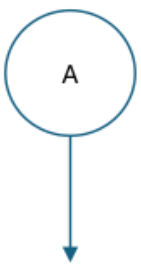
✓Gruppo di azioni che risolvono un sottoproblema



Connessione tra parti di diagrammi di flusso

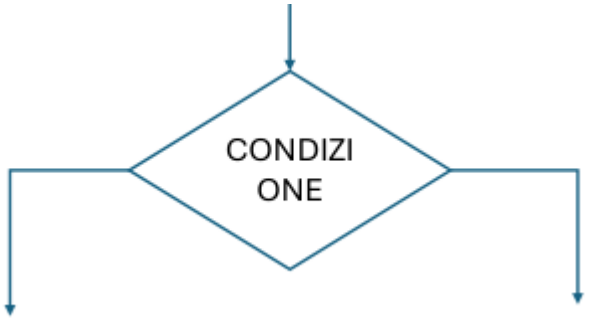
✓Per connettere **parti di UN diagramma di flusso** è possibile usare i connettori





Blocco di decisione binaria

Possono essere presenti istruzioni condizionali, la cui esecuzione dipende da scelte effettuati in base ai dati.
Concettualmente possiamo immaginare che il flusso di esecuzione si ramifichi, per poi far eseguire un operazione in base alla condizione



Programmazione strutturata

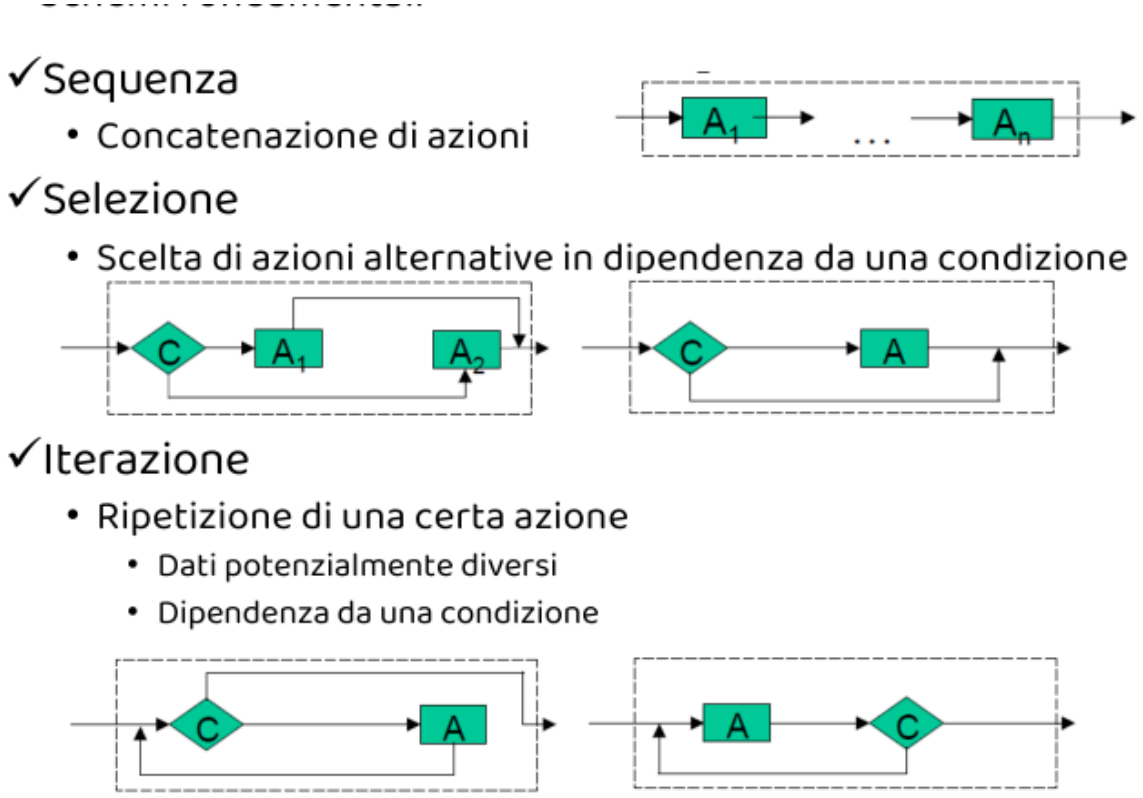
La programmazione strutturata è una metodologia che si basa su alcuni principi fondamentali per migliorare la leggibilità, la manutenibilità e la correttezza del codice.

- **Uso di schemi fondamentali**:
  - **Diagrammi strutturati**: La programmazione strutturata utilizza solo diagrammi che seguono schemi ben definiti. Questi diagrammi rappresentano il flusso di controllo del programma in modo chiaro e ordinato.
  - **Configurazioni standard di blocchi elementari**: Questi blocchi sono comuni a molti processi della vita quotidiana e includono azioni come l'inizio, la fine, le decisioni condizionali e le iterazioni. Utilizzare configurazioni standard aiuta a mantenere la coerenza e la comprensibilità del codice.
- **Sviluppo per raffinamenti successivi**:
  - **Punto di ingresso e uscita unici**: Ogni schema fondamentale deve avere un solo punto di ingresso e un solo punto di uscita. Questo principio evita la complessità e rende il flusso di controllo più facile da seguire.
  - **Sostituibilità ad un blocco di azione**: Ogni schema può essere trattato come un blocco di azione singolo. Questo significa che uno schema complesso può essere sostituito da un blocco di azione più semplice, mantenendo la struttura del programma.
  - **Omissione dei blocchi di inizio e fine**: Quando si inserisce uno schema all'interno di un altro, è possibile omettere i blocchi di inizio e fine dello schema inserito. Questo semplifica la rappresentazione e mantiene il focus sulle azioni principali.

Schemi fondamentali

- **Sequenza**: Questo schema rappresenta una serie di azioni che vengono eseguite una dopo l'altra. Ogni azione segue direttamente quella precedente senza deviazioni. È il tipo più semplice di controllo del flusso.

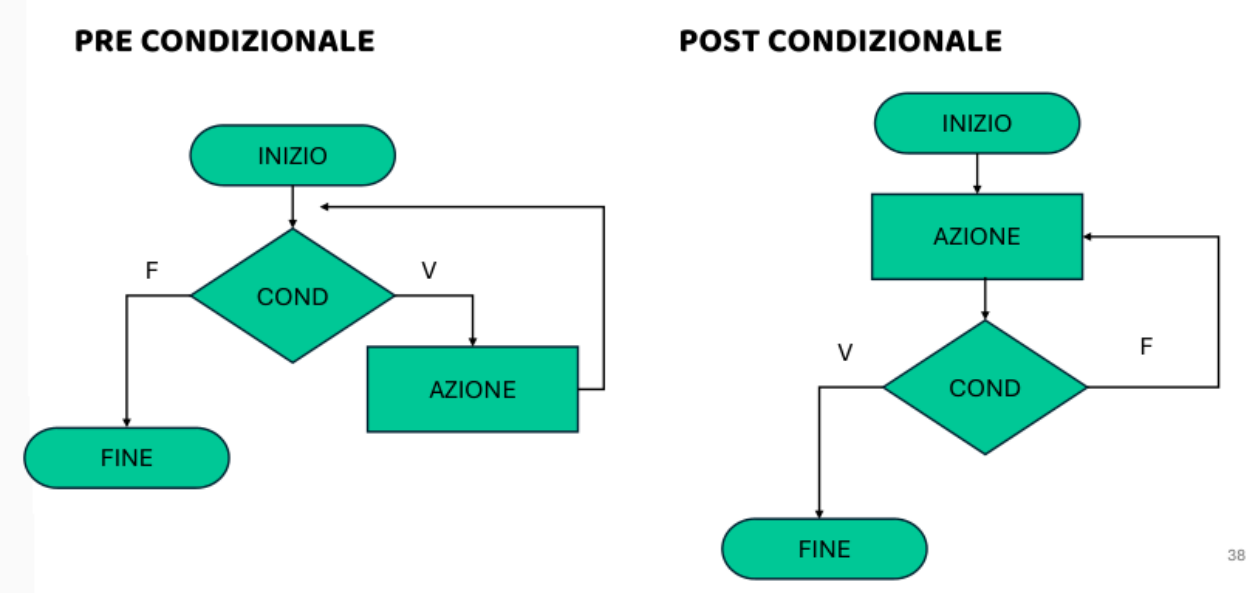
- **Selezione**: Conosciuta anche come struttura condizionale, permette di scegliere tra due o più percorsi di esecuzione basati su una condizione. Ad esempio, un'istruzione `if-else` in cui, se una condizione è vera, viene eseguito un blocco di codice, altrimenti ne viene eseguito un altro.
- **Iterazione**: Questa struttura consente di ripetere una serie di azioni fino a quando una condizione specifica non viene soddisfatta.



Iterazione

L'iterazione è fondamentale per eseguire operazioni ripetitive. Esistono due tipi principali di iterazione:

- **Pre-condizionale**: La condizione viene verificata prima di eseguire l'azione. Se la condizione è vera, l'azione viene eseguita. Questo ciclo continua fino a quando la condizione diventa falsa. Un esempio comune è il ciclo `while`.
- **Post-condizionale**: La condizione viene verificata dopo l'esecuzione dell'azione. Questo garantisce che l'azione venga eseguita almeno una volta. Un esempio comune è il ciclo `do-while`.



I diagrammi hanno comunque svantaggi molto importanti, sono ingombranti e difficili da seguire, non sono naturalmente strutturati e sono diversi dal linguaggio di programmazione.

Nei diagrammi di flusso non sono necessarie le istruzioni di salto (per il teorema di Bohm-Jacopini) e potenzialmente dannose perché rendono difficoltoso seguire il flusso del controllo con scarsa modificabilità e interazioni impreviste

Teorema di Bohm-Jacopini

Dato un processo P e un diagramma che lo descrive, è sempre possibile determinare un processo Q, equivalente a P, che sia descrivibile con un diagramma strutturato
• Processi equivalenti se producono lo stesso effetto

Un processo o metodo solutivo può essere sempre descritto tramite diagrammi strutturati
Un qualsiasi algoritmo può essere espresso usando solo combinazioni di sequenza, selezione e iterazione.