Fasi dello sviluppo software

1. Analisi (problema)
2. Progettazione (algoritmo)
3. Implementazione (programma)
4. Testing (applicazione)

Algoritmi e strutture di controllo

Algoritmo: una sequenza ordinata di istruzioni ben definite che svolge un dato compito in un tempo finito.

Ordinato significa che le istruzioni possono essere numerate, ma un algoritmo deve avere l'abilità di alterare l'ordine delle istruzioni, utilizzando le strutture di controllo.

Strutture di controllo

Queste ultime rientrano in tre categorie di operatori:

Operazioni sequenziali: Istruzioni eseguite in ordine.

Operazioni condizionali: Strutture di controllo che in base all'esito vero/falso ad un test selezionano il blocco di istruzioni da eseguire.

Operazioni Iterative (cicli): Strutture di controllo che ripetono un blocco di istruzioni

Programmazione strutturata

Tecnica per progettare programmi in cui viene utilizzata una gerarchia di moduli, ciascuno dei quali ha un solo punto di ingresso ed una sola uscita e dove il controllo scende verso il basso lungo la struttura, senza salti incondizionati ai livelli più alti della struttura.

In MATLAB Questi moduli possono essere funzioni predefinite o definite dall'utente.

Vantaggi della programmazione strutturata

1. I programmi strutturati sono di facile scrittura, perché consentono di studiare prima il problema generale, curando i dettagli solo alla fine.

2. I moduli (funzioni) scritti per un'applicazione possono essere utilizzati per altre applicazioni (riusabilità).

3. Il debug dei programmi strutturati è semplice, perché ogni modulo è progettato per svolgere un solo compito e quindi può essere testato separatamente dagli altri moduli.

Vantaggi della programmazione strutturata

4. La programmazione strutturata è efficiente nei lavori di gruppo, perché più persone possono lavorare su un progetto comune, sviluppando ognuno diversi moduli del codice.

5. I programmi strutturati sono più semplici da capire e modificare, specialmente se vengono utilizzati nomi appropriati per i moduli, e se vengono specificati bene i loro compiti nella documentazione / nei commenti.

Consigli per un documentazione efficiente

1. Utilizzare per le variabili dei nomi che ne illustrino il significato

2. Utilizzare i commenti all'interno del codice.

3. Fare i diagrammi di struttura.

4. Fare i diagrammi di flusso.

5. Fare una descrizione verbale del programma, in *pseudocodifica*.

Diagrammi per la documentazione

Per lo sviluppo e la documentazione dei programmi sono utili due tipi di diagrammi:

- diagrammi di struttura
- diagrammi di flusso.

Diagramma di struttura di un gioco

Il diagramma di struttura è una descrizione grafica di come le diverse parti del programma sono interconnesse.

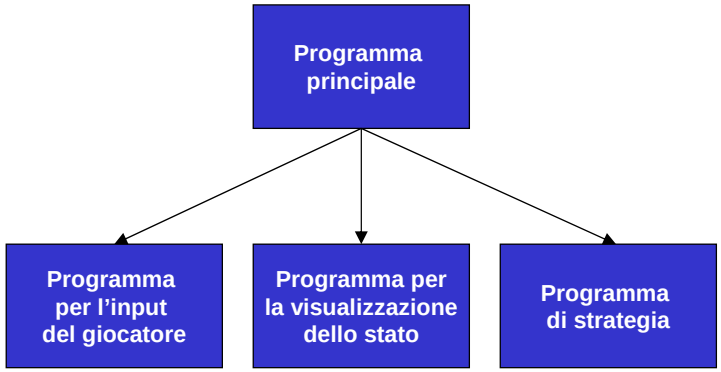


Diagramma di flusso dell'istruzione if

I diagrammi di flusso sono utili per sviluppare e documentare i programmi che contengono istruzioni condizionali, perché possono illustrare i vari "cammini" (detti rami o branch) che un programma può seguire a seconda delle istruzioni condizionali.

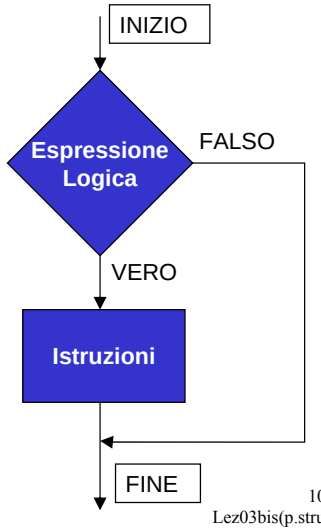
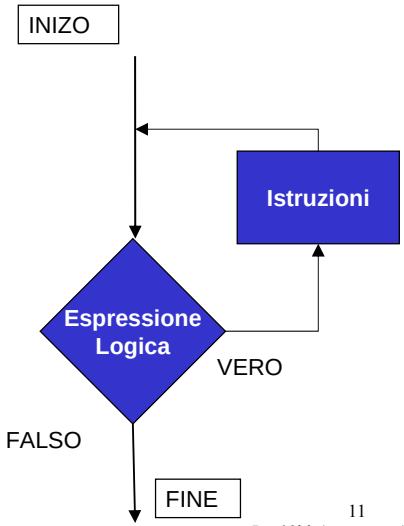


Diagramma di flusso del ciclo while

Il ciclo while consente di ripetere una sequenza di istruzioni finché una condizione rimane vera.



La pseudocodifica

- Per la documentazione possiamo usare anche la pseudocodifica. Questa è una formulazione del programma utilizzando il linguaggio corrente e le espressioni matematiche, senza la sintassi dettagliata.

- Ogni istruzione in pseudocodice può essere numerata, ma deve essere univoca e calcolabile.

Esempio di pseudocodifica

1. Leggi A
2. Leggi B
3. Risultato=0
4 Finché (B>0)
5. Risultato=Risultato+A
6. B=B-1
7. Stampa Risultato
8. Fine