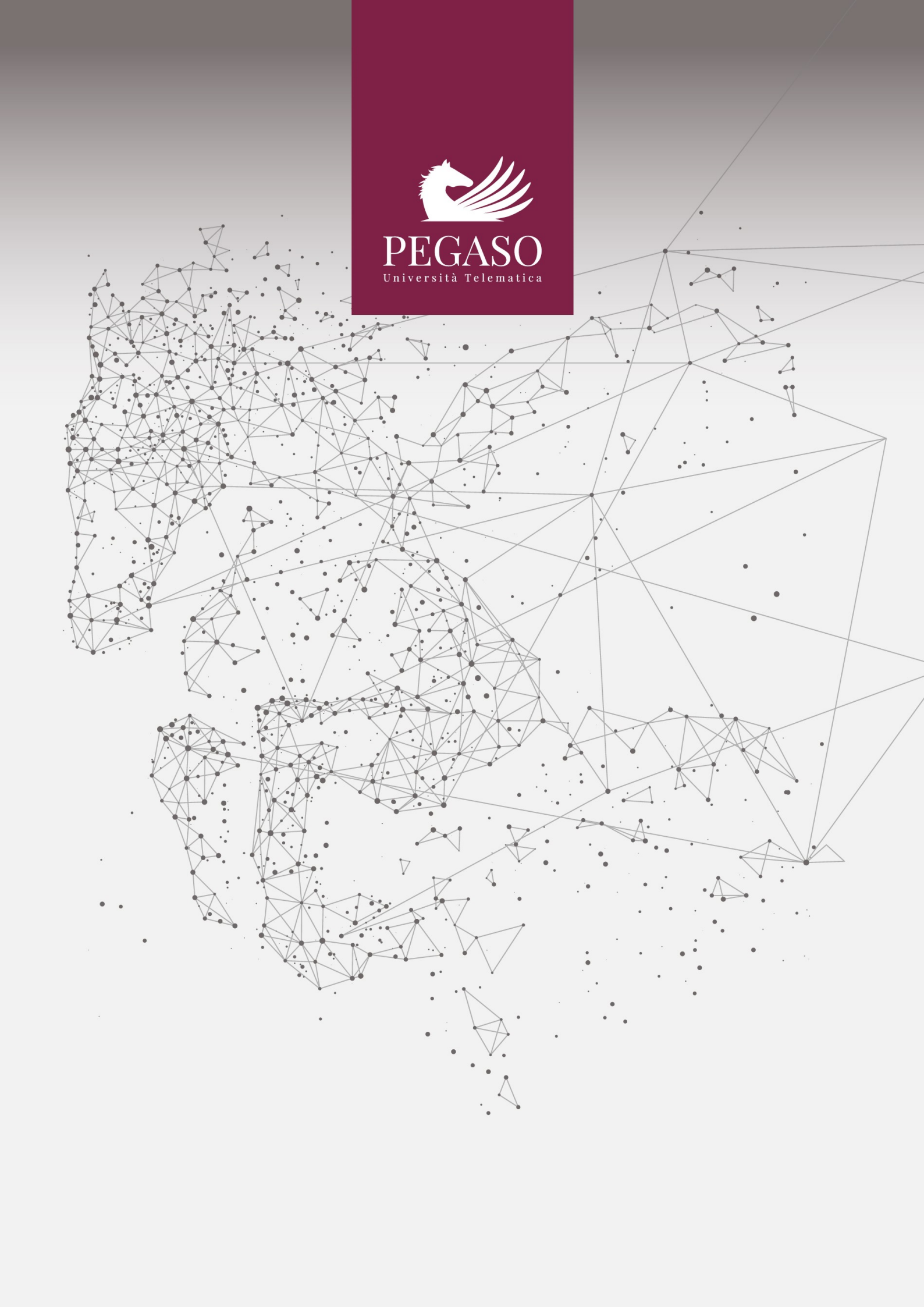

PEGASO
Università Telematica

Indice

1. Introduzione

Le La soluzione di qualsiasi problema di calcolo implica l'esecuzione di una serie di azioni in un ordine specifico.

È chiamato *algoritmo* una *procedura* per la risoluzione di un problema nei termini di:

1. *azioni* da eseguire e

2. *ordine* in cui queste azioni vanno eseguite

L'esempio seguente dimostra quanto sia importante specificare correttamente l'ordine di esecuzione delle azioni.

Considerate "l'algoritmo sveglia!" per alzarsi dal letto e andare al lavoro:

(1) alzarsi dal letto

(2) togliersi il pigiama

(3) farsi una doccia

(4) vestirsi

(5) fare colazione

(6) andare al lavoro.

Supponete che gli stessi passi siano compiuti con un ordine leggermente diverso:

(1) alzarsi dal letto

(2) togliersi il pigiama

(3) vestirsi

(4) farsi una doccia

(5) fare colazione

(6) andare al lavoro

In questo caso ci presenteremmo al lavoro bagnati fradici.

Specificare l'ordine in cui le istruzioni vanno eseguite nel programma di un computer si dice *controllo* del programma.

2. Pseudocodice

Lo pseudocodice è un linguaggio artificiale e informale che aiuta a sviluppare algoritmi. Lo pseudocodice che presentiamo qui è particolarmente utile per lo sviluppo di algoritmi che saranno convertiti in programmi strutturati in C.

Lo pseudocodice è simile al linguaggio di ogni giorno.

È comodo e semplice per l'utente, sebbene non sia un vero e proprio linguaggio di programmazione per computer.

I programmi in pseudocodice non sono eseguiti su computer.

Piuttosto, vi aiutano semplicemente a riflettere bene su un programma prima che tentiate di scriverlo in un linguaggio di programmazione come il C.

Lo pseudocodice consiste puramente di caratteri, così potete comodamente scrivere programmi in pseudocodice in un computer usando un programma editor.

Un programma in pseudocodice preparato con cura può essere facilmente convertito in un corrispondente programma in C.

Questo si fa, in molti casi, semplicemente sostituendo le istruzioni dello pseudocodice con le loro equivalenti in C.

Lo pseudocodice consiste solamente in istruzioni di azione e decisione (quelle che vengono eseguite dopo la conversione del programma dallo pseudocodice al C).

Le definizioni non sono istruzioni eseguibili, ma semplicemente messaggi per il compilatore.

Ad esempio, la definizione

int i;

dice al compilatore il tipo della variabile i e lo istruisce a riservare spazio nella memoria per la variabile.

Ma questa definizione non provoca alcuna azione – come un'operazione di input o di output, oppure un calcolo o un confronto – quando il programma viene eseguito.

Alcuni programmatori scelgono di elencare ogni variabile all'inizio di un programma in pseudocodice e di accennare brevemente allo scopo di ognuna di esse.

3. Diagrammi di flusso

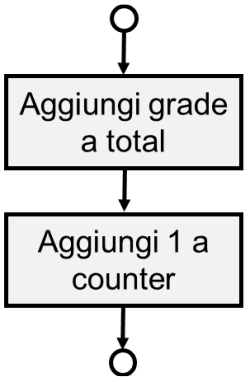
Un diagramma di flusso è una rappresentazione grafica di un algoritmo o di una porzione di un algoritmo.

I diagrammi di flusso sono disegnati usando certi simboli specifici come rettangoli, rombi, rettangoli arrotondati e cerchietti; questi simboli sono collegati da frecce chiamate linee di flusso.

Come lo pseudocodice, i diagrammi di flusso sono utili per sviluppare e rappresentare gli algoritmi, anche se la maggior parte dei programmatori preferisce lo pseudocodice.

I diagrammi di flusso mostrano chiaramente come operano le strutture di controllo.

Consideriamo il seguente diagramma di flusso per una struttura sequenziale.




```
total = total + grade;
```

```
counter = counter + 1;
```

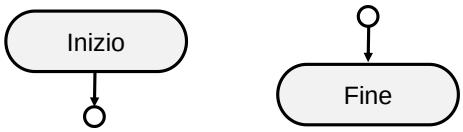
Usiamo il simbolo rettangolo, chiamato anche simbolo di azione, per indicare qualsiasi tipo di azione che include un calcolo o un'operazione di input/output.

Le linee di flusso nella figura indicano l'ordine in cui sono eseguite le azioni: inizialmente grade (voto) viene aggiunto a total (totale), poi 1 viene aggiunto a counter (contatore).

Il C ci permette di avere tutte le azioni che vogliamo in una struttura sequenziale.

Dovunque possa essere inserita una singola azione possiamo inserire diverse azioni in sequenza.

Quando si disegna un diagramma di flusso che rappresenta un algoritmo completo, il primo simbolo usato nel diagramma è il simbolo rettangolo arrotondato contenente la parola "Inizio"; l'ultimo simbolo usato è il simbolo rettangolo arrotondato contenente la parola "Fine".



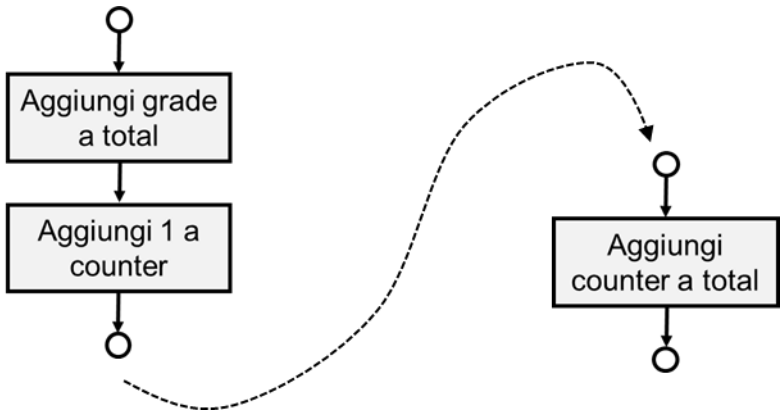
Quando si disegna solo una porzione di un algoritmo, i simboli rettangolo arrotondato sono omessi e al loro posto si usano dei simboli cerchietto, chiamati anche simboli connettori.

La rappresentazione a diagramma di flusso di ogni istruzione di controllo ha due simboli cerchietto:

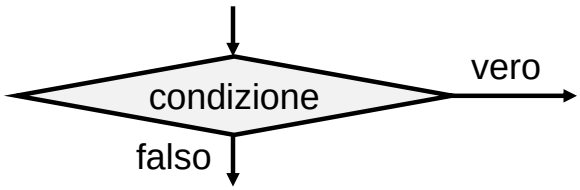
- uno nel punto di ingresso dell'istruzione di controllo
- uno nel punto di uscita

Queste istruzioni di controllo a un solo ingresso e a una sola uscita facilitano la costruzione di programmi chiari.

I segmenti di diagrammi di flusso relativi alle istruzioni di controllo possono essere attaccati l'uno all'altro collegando il punto di uscita di un'istruzione di controllo al punto di ingresso della successiva.



Forse il simbolo più importante di un diagramma di flusso è il simbolo rombo, chiamato anche simbolo di decisione, il quale indica che c'è da prendere una decisione.



Il simbolo di decisione contiene un'espressione, come una condizione, che può essere o vera o falsa.

Il simbolo di decisione ha due linee di flusso che emergono da esso.

Una indica la direzione da prendere quando l'espressione nel simbolo è vera e l'altra la direzione da prendere quando l'espressione è falsa.

3.1 Istruzioni di selezione in C

Il C fornisce tre tipi di **strutture di selezione** nella forma di istruzioni.

- L'istruzione di selezione if seleziona (esegue) un'azione se una condizione è vera o salta l'azione se la condizione è falsa.
- L'istruzione di selezione if...else esegue un'azione se una condizione è vera ed esegue un'azione diversa se la condizione è falsa.
- L'istruzione di selezione switch esegue una fra varie azioni differenti, a seconda del valore di un'espressione.

L'istruzione if è chiamata istruzione di *selezione singola* poiché essa seleziona o ignora un'azione singola.

L'istruzione if...else è chiamata istruzione *di selezione doppia* perché effettua una selezione tra due azioni differenti.

L'istruzione switch è chiamata istruzione di *selezione multipla* perché effettua una selezione tra varie azioni differenti.

3.2 Istruzioni di iterazione in C

Il C fornisce tre tipi di strutture di iterazione nella forma di istruzioni, cioè

- while
- do...while
- for.

Questo è tutto quello che c'è.

Il linguaggio C ha soltanto sette istruzioni di controllo:

- sequenza,
- tre tipi di selezione
- tre tipi di iterazione.

Ogni programma in C è formato dalla combinazione di tante istruzioni di controllo di ognuno di questi tipi quante sono quelle appropriate per l'algoritmo che il programma implementa.

La rappresentazione a diagramma di flusso di ogni istruzione di controllo ha due simboli cerchietto, uno nel punto di ingresso dell'istruzione di controllo e uno nel punto di uscita.

Queste istruzioni di controllo a un solo ingresso e a una sola uscita facilitano la costruzione di programmi chiari.

I segmenti di diagrammi di flusso relativi alle istruzioni di controllo possono essere attaccati l'uno all'altro collegando il punto di uscita di un'istruzione di controllo al punto di ingresso della successiva.

Ciò è molto simile al modo in cui un bambino mette l'uno sull'altro i blocchi di costruzioni, per cui chiamiamo quest'operazione accatastamento delle istruzioni di controllo.

Vi è soltanto un altro modo in cui le istruzioni di controllo possono essere connesse, un metodo chiamato annidamento delle istruzioni di controllo.

Qualsiasi programma in C che avremo mai bisogno di costruire potrà quindi essere costruito partendo soltanto da sette tipi differenti di istruzioni di controllo combinate in due soli modi.

3.3 Istruzione di selezione if

Le istruzioni di selezione sono usate per scegliere tra linee di azione alternative.

Ad esempio, supponiamo che il voto minimo per superare un esame sia 18.

L'istruzione in pseudocodice

Se il voto dello studente è maggiore o uguale a 18

Stampa "Passed"

verifica se la condizione "il voto dello studente è maggiore o uguale a 18" è vera o falsa. Se la condizione è vera, allora viene stampato "Passed" e viene "eseguita" nell'ordine la successiva istruzione nello pseudocodice (ricordate che lo pseudocodice non è un vero linguaggio di programmazione).

Se la condizione è falsa, la stampa viene ignorata e viene eseguita nell'ordine la successiva istruzione nello pseudocodice.

La precedente istruzione in pseudocodice *Se...* può essere scritta in C come:

```
if ( grade >= 18 ) {
   puts( "Passed\n" );
} // fine di if
```

Si noti che il codice C corrisponde strettamente allo pseudocodice (naturalmente dovrete dichiarare la variabile grade come int).

Questa è una delle proprietà dello pseudocodice che lo rende uno strumento utile per lo sviluppo di un programma.

La seconda riga di questa struttura di selezione è indentata.

Tale indentazione è facoltativa, ma altamente raccomandata, in quanto consente di mettere in rilievo la struttura propria dei programmi strutturati.

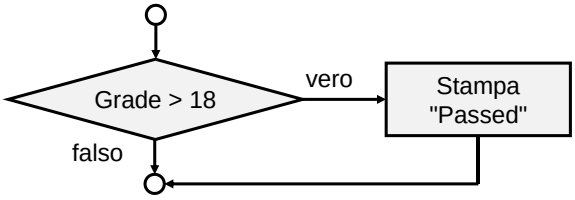
Il compilatore C ignora i caratteri di spaziatura, come spazi, tabulazioni e newline usati per l'indentazione e la spaziatura verticale.

Le decisioni nell'istruzione if possono basarsi su condizioni contenenti operatori relazionali o di uguaglianza.

In realtà, una decisione può basarsi su qualsiasi espressione:

- se l'espressione ha valore uguale a zero è trattata come falsa
- se ha valore diverso da zero è trattata come vera.

Anche l'istruzione if è un'istruzione a una sola entrata e a una sola uscita.



Anche diagrammi di flusso per le rimanenti strutture di controllo possono contenere (oltre ai simboli cerchietto e alle linee di flusso) solamente simboli rettangolo per indicare le azioni da eseguire e simboli rombo per indicare le decisioni da prendere.

Questo è il modello di programmazione azione/decisione.

Il compito di un programmatore C, quindi, è quello di assemblare un programma partendo da tante istruzioni di controllo di ogni tipo quante ne richiede l'algoritmo, combinandole in due soli modi possibili (accatastandole o annidandole) e inserendo poi le azioni e le decisioni in un modo appropriato per l'algoritmo.

4. Strutture di controllo

Normalmente le istruzioni in un programma sono eseguite una dopo l'altra nell'ordine in cui sono scritte.

Questa semplice modalità è chiamata esecuzione *sequenziale*.

Varie istruzioni in C vi permetteranno di specificare come istruzione successiva da eseguire una diversa da quella successiva nella sequenza.

Questa modalità è chiamata *trasferimento del controllo*.

Durante gli anni Settanta divenne chiaro che l'uso indiscriminato del trasferimento del controllo stava alla radice di una grande quantità di problemi incontrati dagli sviluppatori del software. Il dito era puntato contro l'istruzione goto, che permette di specificare un trasferimento del controllo a uno dei tanti punti possibili in un programma.

Il principio della cosiddetta programmazione strutturata divenne quasi sinonimo di "eliminazione del goto".

I programmi prodotti con tecniche strutturate, prive di istruzioni goto, si sono dimostrati più chiari, più facili da correggere e da modificare e privi di errori sin dal primo momento.

Tutti i programmi potrebbero essere scritti in termini di sole tre strutture di controllo:

- la struttura *sequenziale*,
- la struttura di *selezione*
- la struttura di *iterazione*

La struttura sequenziale è semplice: a meno che non venga specificato diversamente, il computer esegue le istruzioni in C una dopo l'altra nell'ordine in cui sono scritte.

Bibliografia

- Paul Deitel, Harvey Deitel, "Il linguaggio C – Fondamenti e tecniche di programmazione", Libro edito da Pearson Italia. Include anche utili esercizi di autovalutazione.