Università degli Studi di Milano

Corso di Laurea in
Sicurezza dei Sistemi e delle Reti Informatiche

Lezione 15 e 16 – La programmazione strutturata

FABIO SCOTTI

Laboratorio di programmazione per la sicurezza

Indice

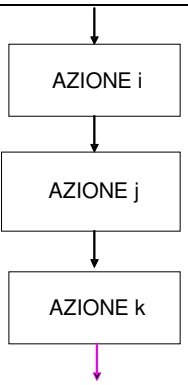
1. La programmazione strutturata

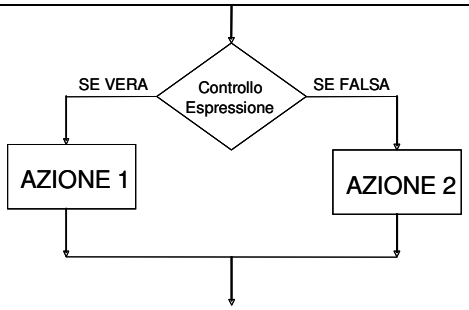
Per programmazione strutturata si intende la tecnica di programmazione che ha lo scopo di semplificare la scrittura dell'algoritmo utilizzando i ***tre*** seguenti requisiti.

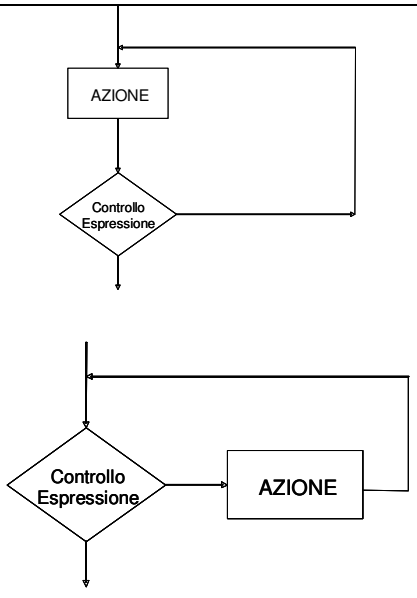
1.1. Solo sequenze, scelte e iterazioni

Il primo requisito impone che per la realizzazione dell'algoritmo e del programma vengano impiegate ***solo*** tre strutture di controllo fondamentali:

Nome costrutto	In linguaggio C	Negli schemi a blocchi

Sequenza;	`azione_i;`
	`azione_j;`
	`azione_k;`
		

Scelta (chiamata anche *selezione*);	`if (Espressione)`
	`{`
	`  azione_1;`
	`}else{`
	`  azione_2;`
	`}`
		

Iterazione.	`do`
	`{`
	`  AZIONE;`
	`}while(Espressione)`
	
	`while(Espressione)`
	`{`
	`  AZIONE;`
	`}`
		

Questi costrutti sono già stati presentati durante il corso.

Il fondamentale teorema di Jacopini-Bohm (1966) ci assicura che le tre strutture di controllo fondamentali formano un insieme di strutture completo, cioè tramite le quali si possono descrivere tutti gli algoritmi.

Ogni algoritmo può essere espresso con le <u>sole</u> tre strutture di controllo fondamentali sequenza, selezione e iterazione.

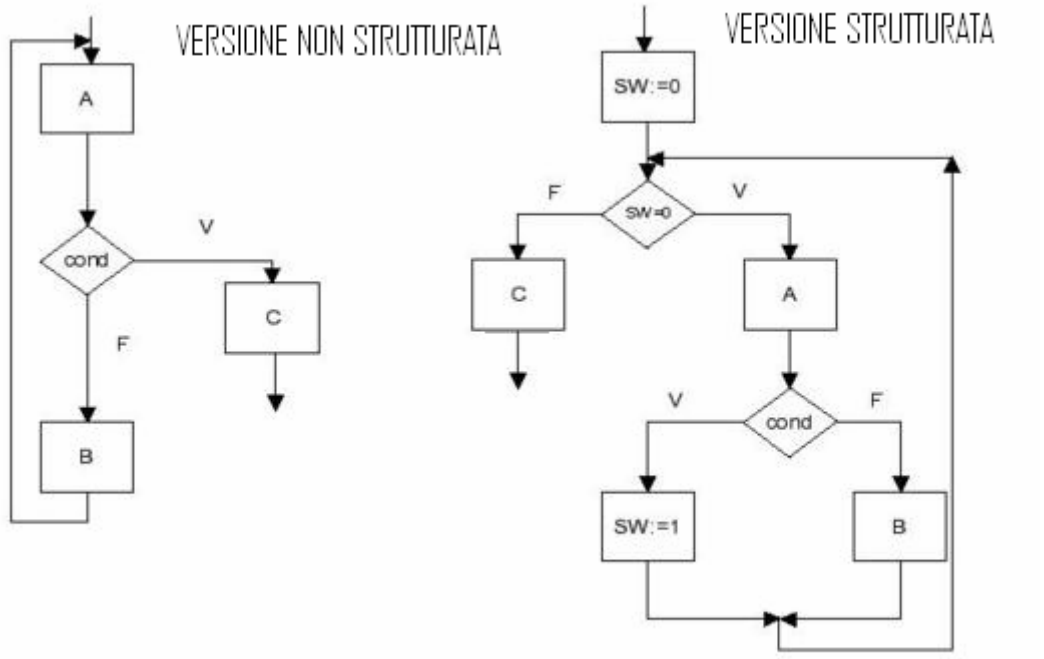
Esprimendo gli stessi concetti usando i costrutti in C, lo stesso teorema può essere espresso dicendo che *ogni algoritmo può essere espresso mediante sequenze di istruzioni,* `if` *e* `while` *(oppure do-while)*. In linea teorica (ed anche pratica) nemmeno il ciclo `for` sarebbe necessario. In generale, l'uso di salti (chiamati non condizionati) nei programmi rende la programmazione non strutturata. Per questo motivo non vanno assolutamente impiegati.

Nel nostro corso useremo soltanto i costrutti `if`, `while`, `do-while`, `for`, `case` e ovviamente le sequenze di istruzioni. In nessun caso useremo altri costrutti.

1.2. Un ingresso ed una uscita per ogni blocco

Il secondo requisito della programmazione strutturata impone che ogni struttura di controllo dell'algoritmo, immaginata mediante il formalismo degli schemi a blocchi, debba essere ***un blocco con una sola freccia in entrata ed una sola in uscita.***

L'applicazione del primo e del secondo requisito, è apparentemente banale. Ad esempio, si noti come lo schema a blocchi sotto riportato nella parte sinistra della figura non soddisfi il primo requisito. Questo accade perché il costrutto impiegato per realizzare l'iterazione non è di quelli specificati per la programmazione strutturata.

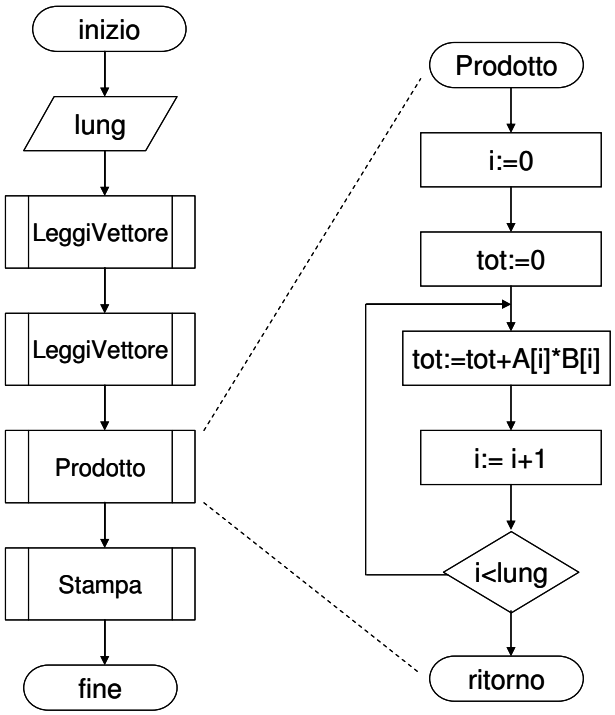


Sulla destra invece è presente uno schema a blocchi che realizza lo stesso algoritmo, ma soddisfa i requisiti della programmazione strutturata. Per realizzarlo si è impiegata una variabile dal nome *SW* inizializzata a zero.

1.3. Un blocco può essere a sua volta una struttura

Ogni blocco interno ad una struttura di controllo non è detto che sia semplice (singola istruzione) ***ma può essere a sua volta una struttura***. Come abbiamo già visto nel corso, questo concetto è legato alle tecniche di tecniche di risoluzione ***top-down***.

Esaminiamo questo semplice schema a blocchi che implementa un algoritmo per il calcolo del prodotto scalare di due vettori di numeri `A` e `B` di lunghezza `lung` elementi. La soluzione è composta da soluzioni di sottoproblemi: la lettura del vettore A di lung elementi, la lettura del vettore B di lung elementi, il calcolo del prodotto scalare degli elementi e la stampa del prodotto scalare calcolato.



La soluzione proposta mostra la decomposizione del sottoprogramma Prodotto che si occupa di calcolare il prodotto scalare di due vettori. Gli altri sottoprogrammi LeggiVettore e Stampa posso essere sviluppati in modo analogo e per brevità vengono omessi.

La soluzione presentata nella forma di uno schema a blocchi può essere presentata in modo equivalente scrivendo un programma C con un `main` che chiama altre funzioni, come nel caso che segue.

```
int main()
{     …
      scanf(“%d”, &lung);
      LeggiVettore(A, lung);
      LeggiVettore(B, lung);
      tot = Prodotto (A, B, lung);
```

```
	Stampa(tot);
	…
} // fine del main

// definizione della funzione Prodotto
float Prodotto( float A[], float B[], int lung)
{
	int i;
	float tot;
	i = 0;
	tot = 0;
	do{
		tot = tot + A[i] * B[i];
		i = i + 1;
	}while(i<lung)
	return tot;
}
```

Per questo motivo consideriamo come *ben scritto* un codice che ha un `main` snello, ovvero composto solo dalla chiamata di alcune principali funzioni.

1.4. Vantaggi della programmazione strutturata

Imporre che una soluzione algoritmica di un problema sia scritta seguendo questi dettami non è un puro esercizio di programmazione ma porta dei vantaggi diretti fra cui:

- maggiore facilità di uso di tecniche top-down;
- algoritmi più leggibili;
- maggiore facilità di individuazione degli errori.

E' da notare che la programmazione strutturata non è la più moderna metodologia per scrivere programmi, anzi, forse, è da considerarsi come una delle prime. Tuttavia è riconosciuta ancora oggi come una delle fondamentali. I concetti della programmazione strutturata sono stati subito impiegati nell'industria e oggigiorno sono stati recepiti da tutti i linguaggi di programmazione moderni.

I ricercatori di informatica teorica e di software engineering ritennero che i vantaggi descritti non fossero ancora soddisfacenti e inventarono altre metodologie per la creazione di programmi con vantaggi ancora maggiori per il programmatore come la *programmazione modulare* e successivamente la *programmazione ad oggetti*.

In questo modulo, l'obiettivo è quello di esercitarsi nell'impiego dei tipi di dato astratto associato e nel corretto soddisfacimento dei requisiti della programmazione strutturata.