Prof. Mario Ciampi

Elementi di Informatica

INGEGNERIA MECCANICA E AEROSPAZIALE

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

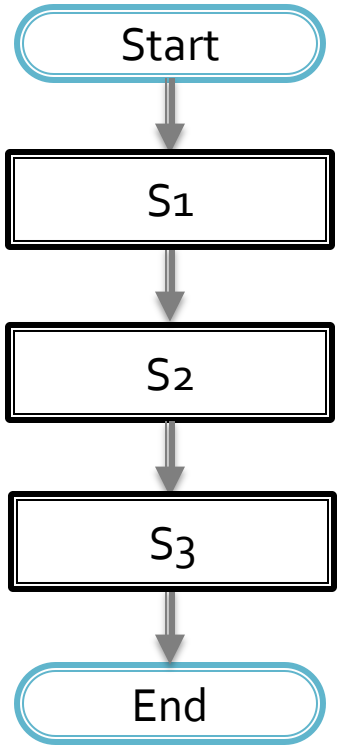
Esercitazione 3

Le strutture di controllo

Sull'importanza delle strutture

- **Teorema di Böhm-Jacopini** (1966)
  - Qualunque algoritmo può essere implementato utilizzando **tre** sole strutture
    - Sequenza
    - Selezione
    - Iterazione o ciclo

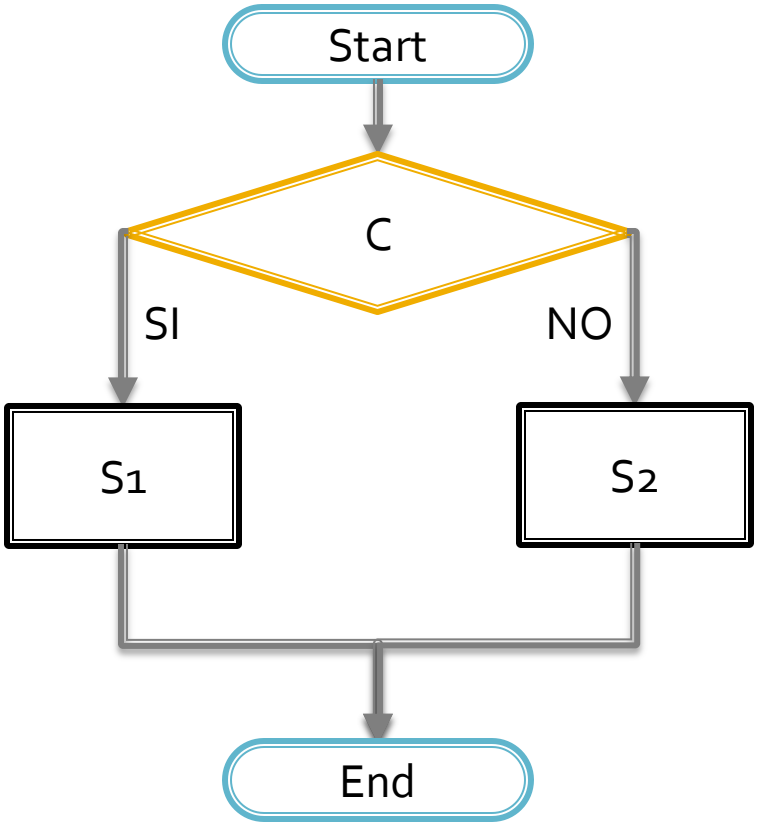
Sequenza



Sull'importanza delle strutture

- **Teorema di Böhm-Jacopini** (1966)
  - Qualunque algoritmo può essere implementato utilizzando **tre** sole strutture
    - Sequenza
    - Selezione
    - Iterazione o ciclo

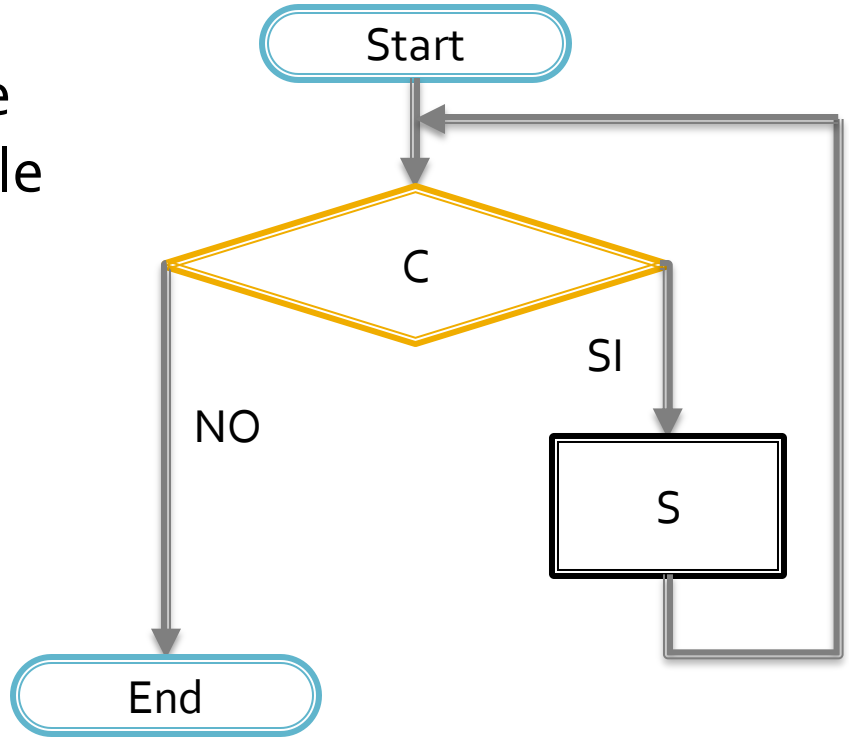
Selezione



Sull'importanza delle strutture

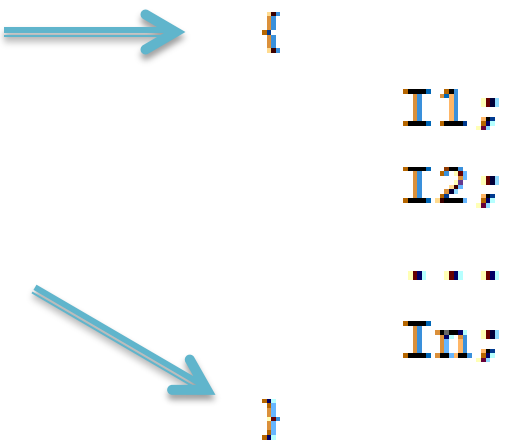
- **Teorema di Böhm-Jacopini** (1966)
  - Qualunque algoritmo può essere implementato utilizzando **tre** sole strutture
    - Sequenza
    - Selezione
    - Iterazione o ciclo

Iterazione



Blocco

- Elemento base per la costruzione di un programma
- Formato dalla sequenza S di n istruzioni
  - $I_1; I_2; \ldots; I_n$
- L'inizio del blocco è indicato da una parentesi graffa aperta
- La fine del blocco è indicata da una parentesi graffa chiusa
- Le istruzioni vengono eseguite secondo l'ordine di lettura
  - Dall'alto verso il basso



Blocco

- La presenza del carattere di terminazione «;» consente di scrivere le istruzioni anche su di uno stesso rigo

```
{
    cout << "ciao ";
    cout << "a ";
    cout << "tutti";
}
```

=

```
{
    cout << "ciao "; cout << "a "; cout << "tutti";
}
```

- La prima forma favorisce la lettura

Blocco

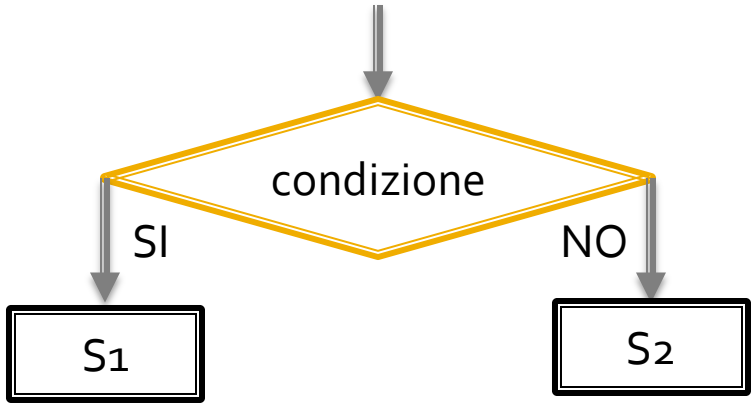
- Un blocco può contenerne altri
- Incolonnamento
  - Evidenzia la presenza di blocchi innestati
    - Migliora la leggibilità del codice

```
{
    Blocco1-I1;
    Blocco1-I2;
    {
        Blocco2-I1;
        Blocco2-I2;
        {
            Blocco3-I1;
            Blocco3-I2;
        }
        Blocco2-I3;
    }
    Blocco1-I3;
}
```

Strutture di controllo
selettive

if … else

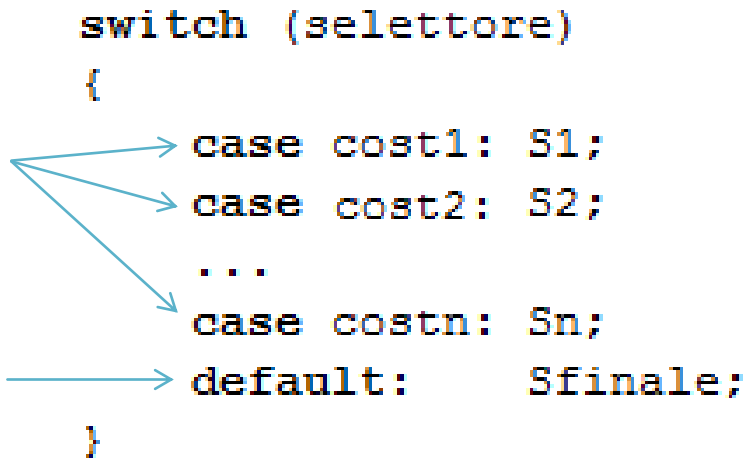
- Con `if` … `else` si effettua la scelta tra due blocchi di istruzioni
- La condizione è una espressione di tipo logico
- Il ramo `else` può non esistere




```
if (condizione)
{
    S1;
}
else
{
    S2;
}
```

switch

- Con il costrutto `switch` si sceglie l'istruzione di inizio della esecuzione in una sequenza di istruzioni
- La scelta avviene:
  - Calcolando il valore dell'espressione *selettore*
  - Confrontando tale valore con una serie di valori costanti indicati con la parola chiave `case`
- Può comprendere una condizione finale di `default`
  - viene eseguita quando il valore del selettore è diverso dalle costanti riportate nelle frasi case



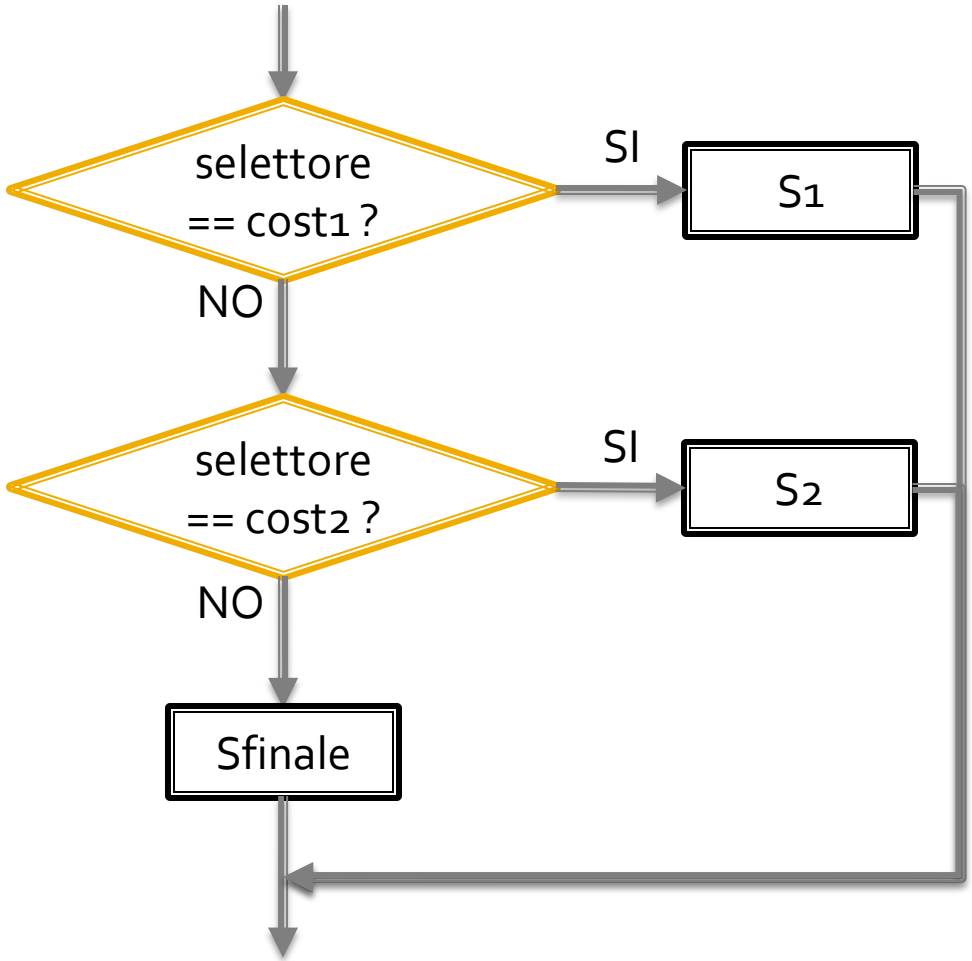
switch

- La struttura `switch` diventa un selettore di blocchi se
  - tutte le sequenze etichettate con le frasi `case` si concludono con l'istruzione `break`
    - che produce come effetto l'uscita dallo switch

```
switch (selettore)
{
    case cost1: S1;
                break;
    case cost2: S2;
                break;
    ...
    case costn: Sn;
                break;
    default:    Sfinale;
}
```

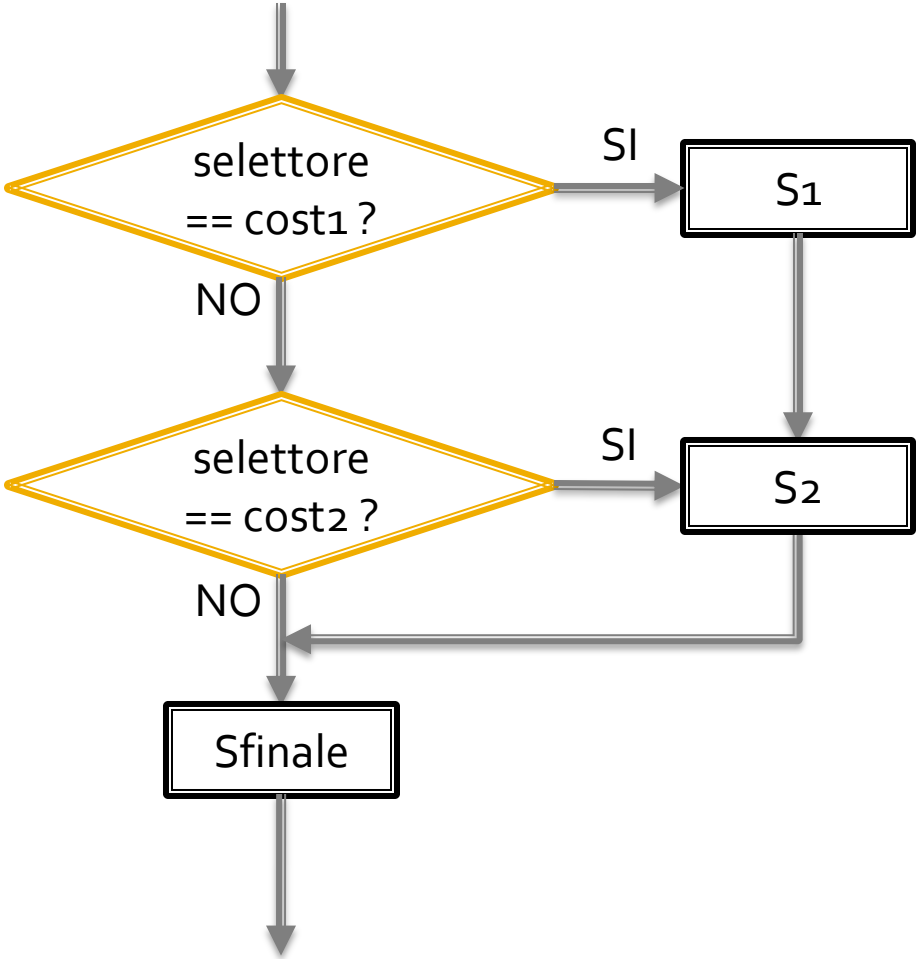
switch

```
switch (selettore)
{
    case cost1: S1;
                break;
    case cost2: S2;
                break;
    default:    Sfinale;
}
```



switch

```
switch (selettore)
{
    case cost1: S1;
                break;
    case cost2: S2;
                break;
    default:    Sfinale;
}
```



Strutture di controllo
iterative

while

- Impone che l'esecuzione del blocco di istruzioni sia ripetuta fino a quando la condizione non diventa FALSE
  - Viene calcolata la *condizione*
    - Se FALSE la sequenza S non viene eseguita
    - Se TRUE si esegue S
      - al suo termine si ricalcola la condizione e si riesegue S se è ancora TRUE
        - Si continua fino a che la condizione non diventa FALSE

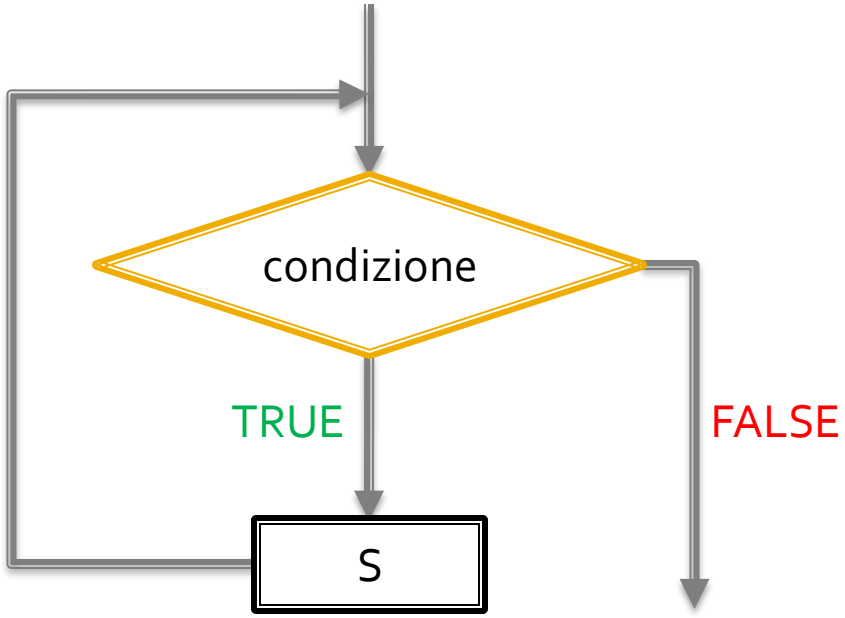
Struttura iterativa
A CONTROLLO INIZIALE

Numero di esecuzioni:
[0,n]

```
while (condizione)
{
    S;
}
```

while

```
while (condizione)
{
    S;
}
```



do while

- Come il while, ma la condizione viene scritta dopo la sequenza S
  - L'esecuzione del blocco avviene almeno una volta
  - Si esegue la sequenza S
  - Si calcola la *condizione*
    - Se TRUE si riesegue S
      - Si continua fino a che la condizione non diventa FALSE

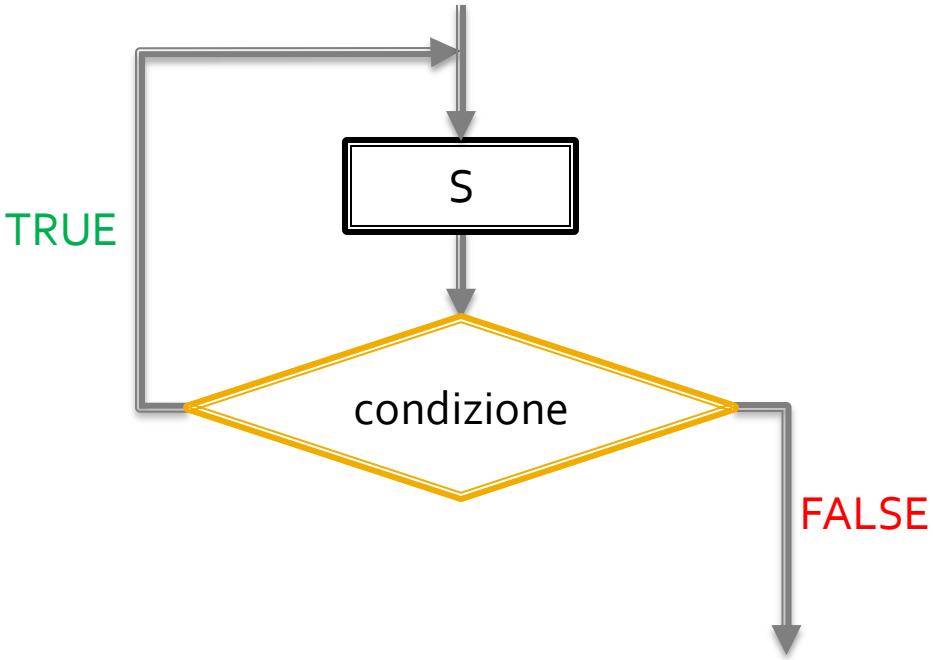
Struttura iterativa
A CONTROLLO FINALE

Numero di esecuzioni:
[1,n]

```
do
{
    S;
}
while (condizione)
```

do while

```
do
{
    S;
}
while (condizione)
```



for

- Il ciclo `for` consente di esprimere
  - una o più istruzioni di inizializzazione per il ciclo
  - una o più istruzioni di variazione delle condizioni di iterazione

```
for (inizializzazioni; condizione; variazioni)
{
    S;
}
```

for

- Il ciclo `for` prescrive
  - L'esecuzione delle istruzioni di inizializzazione
  - Il calcolo della condizione
  - L'esecuzione della sequenza S se è vera la condizione
    - In caso contrario l'esecuzione termina
  - L'esecuzione delle istruzioni di variazione al termine di S
  - La rivalutazione della condizione con il ripetersi dei passi precedenti fino alla determinazione della falsità della condizione

```
for (inizializzazioni; condizione; variazioni)
{
    S;
}
```

for

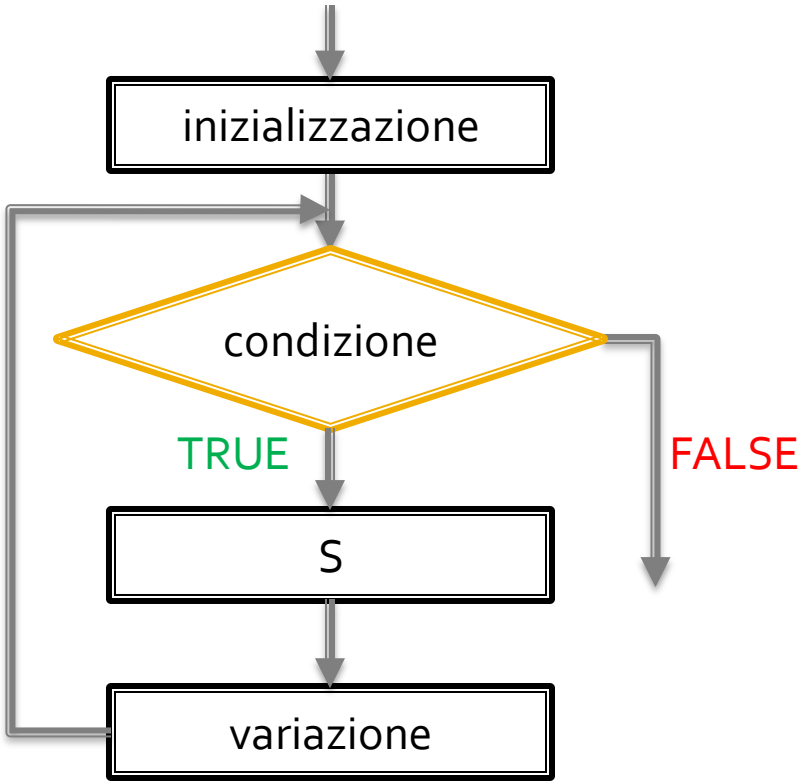
- Le istruzioni di inizializzazione e di variazione non sono obbligatorie
  - mentre la condizione è obbligatoria
- Se inizializzazione e variazione non vengono espresse, i cicli `for` e `while` sono equivalenti

```
for (inizializzazioni; condizione; variazioni)
{
    S;
}
```

- In generale, i cicli for vengono utilizzati quando il numero delle operazioni richieste è ben definito

for

```
for (inizializzazioni; condizione; variazioni)
{
     S;
}
```



Strutture di controllo in sequenza / innestate

- Struttura di controllo innestata
  - Contenuta totalmente in un'altra

```
if (condizioneA)
{
   if (condizioneB)
      istruzione1;
   else
      istruzione2;
}
else
   istruzione3;
```

- Struttura di controllo in sequenza
  - Il suo punto di ingresso è collegato al punto di uscita di un'altra struttura

```
if (condizioneA)
   istruzione1;
else
   istruzione2;
if (condizioneB)
   istruzione3;
else
   istruzione4;
```

Istruzioni non strutturate

goto / break / continue

- Le istruzioni non strutturate sono istruzioni di salto che possono violare i principi della programmazione strutturata
  - Sono da evitare
  - `goto`
    - Provoca il trasferimento incondizionato del flusso di controllo del programma all'istruzione identificata dall'etichetta <label>

```
goto <label>;
label: istruzione;
```

  - `break`
    - Comporta l'uscita da `while`, `do-while`, `switch`, `for`
    - È normalmente usata solo per lo `switch`
  - `continue`
    - Può essere usata nel
      - `while` e `do-while`, in cui è equivalente al salto alla verifica della condizione
      - `for`, in cui è equivalente al salto alla variazione delle condizioni del ciclo

Esercizi

Esercizi

- Conversione da minuscolo a maiuscolo
  - Scrivere un programma che trasforma in maiuscola una lettera inserita da tastiera
    - Dopo aver verificato che la lettera inserita è minuscola
  - Regola
    - La posizione nella tabella ASCII di una lettera maiuscola è data dalla posizione della stessa lettera minuscola MENO 32

Tabella ASCII

0		32		64	@	96	`	128	Ç	160	á	192	└	224	Ó
1	☺	33	!	65	A	97	a	129	ü	161	í	193	┴	225	ß
2	☻	34	"	66	B	98	b	130	é	162	ó	194	┬	226	Ô
3	♥	35	#	67	C	99	c	131	â	163	ú	195	├	227	Ò
4	♦	36	$	68	D	100	d	132	ä	164	ñ	196	─	228	õ
5	♣	37	%	69	E	101	e	133	à	165	Ñ	197	┼	229	Õ
6	♠	38	&	70	F	102	f	134	å	166	ª	198	ã	230	µ
7	•	39	'	71	G	103	g	135	ç	167	º	199	Ã	231	þ
8	◘	40	(	72	H	104	h	136	ê	168	¿	200	╚	232	Þ
9	○	41	)	73	I	105	i	137	ë	169	®	201	╔	233	Ú
10	◙	42	*	74	J	106	j	138	è	170	¬	202	╩	234	Û
11	♂	43	+	75	K	107	k	139	ï	171	½	203	╦	235	Ù
12	♀	44	,	76	L	108	l	140	î	172	¼	204	╠	236	ý
13	♪	45	-	77	M	109	m	141	ì	173	¡	205	═	237	Ý
14	♫	46	.	78	N	110	n	142	Ä	174	«	206	╬	238	¯
15	☼	47	/	79	O	111	o	143	Å	175	»	207	¤	239	´
16	►	48	0	80	P	112	p	144	É	176	░	208	ð	240	-
17	◄	49	1	81	Q	113	q	145	æ	177	▒	209	Ð	241	±
18	↕	50	2	82	R	114	r	146	Æ	178	▓	210	Ê	242	‗
19	‼	51	3	83	S	115	s	147	ô	179	│	211	Ë	243	¾
20	¶	52	4	84	T	116	t	148	ö	180	┤	212	È	244	¶
21	§	53	5	85	U	117	u	149	ò	181	Á	213	ı	245	§
22	▬	54	6	86	V	118	v	150	û	182	Â	214	Í	246	÷
23	↨	55	7	87	W	119	w	151	ù	183	À	215	Î	247	¸
24	↑	56	8	88	X	120	x	152	ÿ	184	©	216	Ï	248	°
25	↓	57	9	89	Y	121	y	153	Ö	185	╣	217	┘	249	¨
26	→	58	:	90	Z	122	z	154	Ü	186	║	218	┌	250	·
27	←	59	;	91	[	123	{	155	ø	187	╗	219	█	251	¹
28	∟	60	<	92	\	124	|	156	£	188	╝	220	▄	252	³
29	↔	61	=	93	]	125	}	157	Ø	189	¢	221	¦	253	²
30	▲	62	>	94	^	126	~	158	×	190	¥	222	Ì	254	■
31	▼	63	?	95	_	127	⌂	159	ƒ	191	┐	223	▀	255	

Soluzione:
Conversione da minuscolo a maiuscolo

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main()
{
    char lettera;

    cout << "Inserire lettera minuscola: ";
    cin >> lettera;

    if (lettera >= 'a' && lettera <= 'z')
    {
       lettera -= 32;
       cout << "La maiuscola e': " << lettera << endl;
    }
    else
       cout << "Non hai inserito una lettera minuscola" << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}
```

NOTA:
Se un blocco si compone di una sola istruzione, le parentesi graffe possono essere omesse

Esercizio

- Valutare la differenza tra questi due programmi:

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main()
{
    char lettera;

    cout << "Inserire lettera minuscola: ";
    cin >> lettera;

    switch (lettera)
    {
        case 'a':   cout << "a1" << endl;
                    cout << "a2" << endl;

        case 'b':   cout << "b1" << endl;

        default:    cout << "default" << endl;
    }
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main()
{
    char lettera;

    cout << "Inserire lettera minuscola: ";
    cin >> lettera;

    switch (lettera)
    {
        case 'a':   cout << "a1" << endl;
                    cout << "a2" << endl;
                    break;
        case 'b':   cout << "b1" << endl;
                    break;
        default:    cout << "default" << endl;
    }
    system("PAUSE");
    return EXIT_SUCCESS;
}
```

Esercizi

- Fattoriale 1
  - Scrivere un programma che calcola il fattoriale di un numero intero non negativo inserito da tastiera
- Fattoriale 2
  - Aggiungere nel programma precedente un controllo sul valore inserito da tastiera
    - Viene continuamente chiesto di inserire un numero in ingresso fino a che il valore inserito non è un numero intero non negativo

Domande