Istruzioni di controllo

- Basate sulla valutazione di una condizione
- Le istruzioni condizionali permettono di eseguire codice diverso
  - if
  - switch
  - L'operatore ternario ?:
- Le istruzioni di loop permettono di eseguire più volte lo stesso codice
  - Ciclo for (classico, range)
  - Ciclo while
  - Ciclo do-while
- Effetti speciali per mezzo di break e continue
- Codice di esempio
  - https://github.com/egalli64/corso-cpp folder b3

if

- La keyword **if** è seguita da:
  - La condizione che deve essere valutata, tra parentesi tonde
  - Il codice che deve essere eseguito se la condizione è vera
    - Se si tratta di un più statement è necessario delimitarli tra parentesi graffe
    - Per omogeneità si usano le graffe anche nel caso di un singolo statement
- Al blocco **if**, come descritto sopra, può seguire la keyword **else**
  - Seguita dal codice associato, secondo le stesse modalità per l'if
  - Il blocco associato all'else viene eseguito se la condizione valutata per l'if è falsa
  - Dunque in un if-else viene sempre eseguito del codice
- Alla keyword else può far nuovamente seguito la keyword if, che introduce un'altra condizione
  - In un **if**-**else if** viene eseguito il primo blocco associato a una condizione vera, se esiste
  - Se l'istruzione è terminata con un else (senza if), il suo blocco è eseguito se nessuna condizione è vera
  - Se non c'è un else finale, può succedere che nessun blocco sia eseguito
  - Non è mai possibile che siano eseguiti più blocchi in una singola istruzione if-else if

switch

- Lo **switch** permette di eseguire solo controlli di uguaglianza su una singola variabile
  - Alternativo, in una forma più limitata, all'if-else if
- Alla keyword switch segue, tra parentesi tonde, la variabile da controllare, di tipo integrale o enumerativo
  - Segue un blocco delimitato da parentesi graffe in cui si specificano dei "**case**"
  - Alla keyword case segue un valore integrale e due punti
  - Può anche essere presente un blocco di **default**
- Ogni blocco case e l'eventuale blocco default di solito terminano con un **break**
- L'esecuzione dello switch inizia dal case in cui il valore della variabile corrisponde al valore indicato
- Se non c'è alcuna corrispondenza, se esiste un default, l'esecuzione inizia da lì
  - Altrimenti si passa alla prima istruzione successiva allo switch
- Il break ha lo scopo di terminare l'esecuzione interna allo switch
- Se un case non ha un break, l'esecuzione passa al blocco seguente
  - Fenomeno detto "**fall through**", poco leggibile, quindi solitamente evitato

L'operatore ternario ?:

- Usato per assegnare a una variabile un valore in maniera condizionata
- type variable = (condition) ? value1 : value2;
  - Se la condizione è vera
    - Alla variabile è assegnato il valore che segue il punto di domanda
  - Altrimenti
    - Alla variabile è assegnato il valore che segue i due punti
- Equivalente a
  - Dichiarazione della variabile
  - Inizializzazione condizionata via if - else

```
type variable;
if (condition) {
    variable = value1;
}
else {
    variable = value2;
}
```

for

- Permette di esprimere in forma compatta istruzioni ripetute
  - Alla keyword **for** segue una parentesi tonda
  - Nella parentesi ci sono tre clausole separate da punto e virgola
    - 1: Eseguita solo la prima volta, normalmente usata per definire una variabile di loop
    - 2: Condizione controllata prima di eseguire ogni loop
      - Se è falsa il loop termina, si passa allo statement successivo al for
    - 3: Eseguita alla fine di ogni loop, prima di verificare nuovamente la condizione

```
for (/* 1 */; /* 2 */; /* 3 */) {
    // ...
}
```

- "Range based" for
  - Formulazione più robusta di un loop for
    - Meno flessibile, calibrata per l'uso più comune e naturale
  - La variabile di loop è *(di default)* copia by value di un valore dell'array
    - Vedremo che è capace di gestire un qualunque oggetto iterabile

```
for (loop_variable : data) {
    // ...
}
```

while / do-while

- Il **while** è equivalente al loop for, più adatto a cicli indefiniti
  - Deve accadere qualcosa che modifichi la condizione
  - È più facile fare errori logici che portano a loop infiniti indesiderati
- Il **do-while** valuta la condizione *alla fine* di ogni loop
  - Esprime bene il fatto che deve essere eseguito almeno una volta
  - Notare il punto e virgola che termina l'istruzione di loop

```
while (condition) {
    // ...
}
```

```
do {
    // ...
} while (condition);
```

break e continue

- Istruzioni specifiche per i loop
  - Il break è usato anche nello switch
- break
  - Forza la terminazione di un loop
  - Il controllo viene passato al primo statement che segue il loop
- continue
  - Forza la terminazione del ciclo corrente
  - Il controllo viene passato
    - while: al check della condizione
    - for: alla terza clausola, e poi al check della condizione