



UNIVERSITÀ  
degli STUDI  
di CATANIA

# Controlli condizionali in C++

Corso di programmazione I

Corso di Laurea Triennale in Informatica

---

Prof. Giovanni Maria Farinella

Web: <http://www.dmi.unict.it/farinella>

Email: [gfarinella@dmi.unict.it](mailto:gfarinella@dmi.unict.it)

Dipartimento di Matematica e Informatica

1. Costrutto if/else e operatore condizionale in C++
2. Confronti lessicografici
3. Costrutto switch
4. Hand Tracing
5. Operatori logici

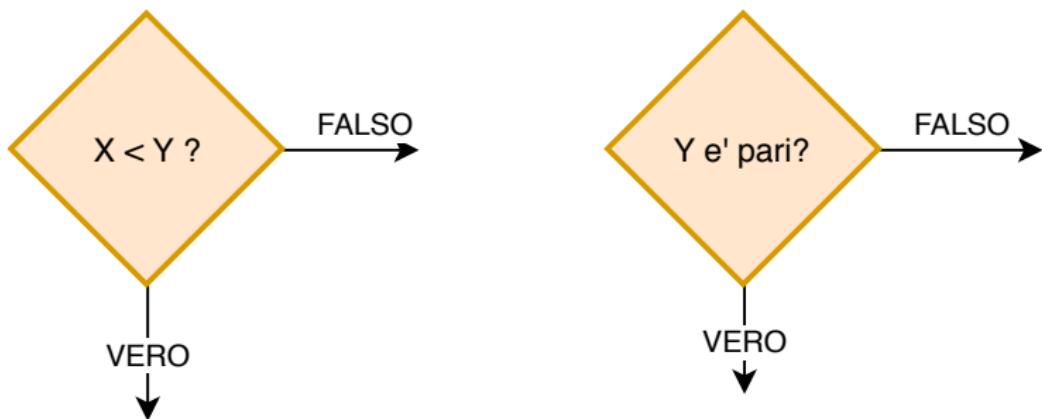
## **Costrutto if/else e operatore condizionale in C++**

---

## Il costrutto if/else in C++

⚡

Il valore di verità del predicato “ $X < Y$ ” (a sinistra) o del predicato “ $Y$  è pari” determina il prossimo passo nel flusso di esecuzione

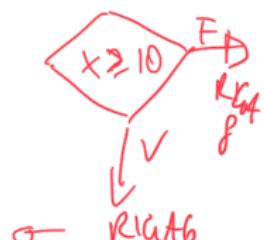


## Il costrutto if/else in C++

La parola chiave **if**, eventualmente seguita da uno o più blocchi **else**, si usa per **condizionare** l'esecuzione di uno o più blocchi di codice in C++.

www

```
1 int x;
2 cout << "Inserire un numero < 10 oppure " << endl;
3 cin >> x;
4
5 if (x>=10)
6     cout << "Numero inserito non valido!";
7 else
8     cout << "Il numero inserito e' " << x;
```



# Il costrutto if/else in C++

## Operatori relazionali

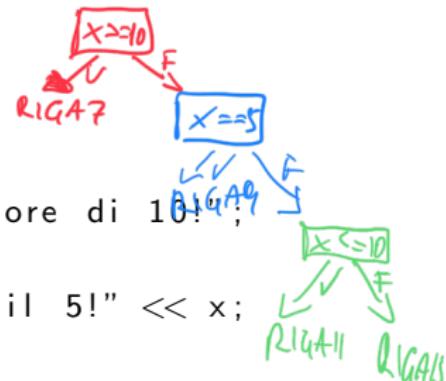
SINTASSI

Notazione Matematica	C++	Descrizione
→ >	>	Maggiore di
→ ≥	≥	Maggiore o uguale
<	<	Minore di
≤	≤	Minore o uguale
→ =	==	Uguale
→ ≠	!=	Diverso

## Il costrutto if/else in C++

Un esempio più articolato:

```
1 int x;
2 cout << "Inserire un numero positivo < 10, ma che \
3 non sia 5!" << endl;
4 cin >> x;
5
6 if(x>=10)
7 cout << "Numero inserito maggiore di 10!";
8 else if(x==5)
9 cout << "Hai inserito proprio il 5!" << x;
10 else if(x<=0)
11 cout << "Il numero e' negativo!" << x;
12 else //eseguito se prec. condizioni non verificate
13 cout << "Il numero inserito e':" << x << endl;
```



## Il costrutto if/else in C++

Blocchi di codice (NB: indentazione e parentesi graffe!):

```
1 int x; double result;
2 double alpha = 0.5; +
3 cout << "Inserire un numero positivo < 10: " << endl;
4 cin >> x;
5
6 if(x>=10) {
7     result = (alpha * x) / 10;
8     cout << "Numero inserito maggiore di 10!" ; 0
9
10 else { // ESEGUITO DAVANNO X È MINORE DI 10
11     result = alpha * x;
12     cout << "Inserito numero valido!";
13 }
```

## Il costrutto if/else in C++

Il seguente codice e' **sintatticamente** corretto?

```
1  if (x>=10){  
2      result = (alpha * x) / 10;  
3      cout << "Numero inserito maggiore di 10!" ;  
4  };  
5  else {  
6      result = alpha * x;  
7      cout << "Inserito numero valido!" ;  
8  }
```

Sintassi errata! Compilatore darà errore (blocco else non “legato”).

## Il costrutto if/else in C++

Il seguente codice e' sintatticamente/semanticamente corretto?

```
1  if (x>=10){  
2      result = (alpha * x) / 10;  
3      cout << "Numero inserito maggiore di 10!" ;  
4  }  
5  else;{  
6      result = alpha * x; //  
7      cout << "Inserito numero valido!" ;  
8  }
```

Sintassi OK, ma non è quello che si voleva (blocco else eseguito incondizionatamente)!

## Il costrutto if/else in C++

Il seguente codice e' **semanticamente** corretto?

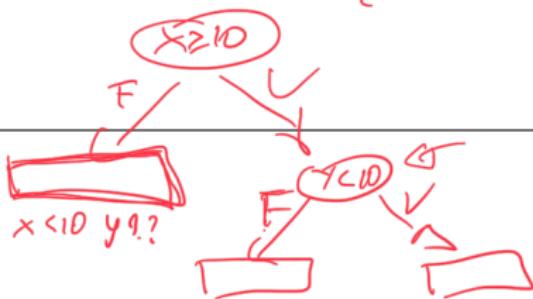
```
1     if (x>=10);  
2         cout << "Numero inserito maggiore di 10!" ;
```

Sintassi OK, ma non è quello che si voleva (Istruzione di output eseguita incondizionatamente)!

## Il costrutto if/else in C++

Indentare è importante per ottenere un codice leggibile.

```
1  if (x>=10){ // if annidati!
2      if(y<10){ // x>=10 e contemporan. y<10
3          result = (alpha * x *y) / 10; ←
4      } else // x>=10 ma y>=10
5          result = (alpha * x * y) / 100; ←
6      } else // x<10, y ???
7          result = -1.0;
```



## Il costrutto if/else in C++

Un errore comune è quello di confondere il simbolo `==` con `=`.

```
1  if(x=10) //assegnamento!
2      y = x/2;
3  else // non sara' mai eseguita!
4      y = x-2;
```

NB: Il compilatore non darà alcun errore!! In C/C++ qualsiasi valore diverso da zero rappresenta il valore true.

## Il costrutto if/else in C++

- a) Se il paese di spedizione è l'Italia, il costo di spedizione sarà di 10 euro; ma per b) la provincia di CT di 15 euro; c) per i paesi diversi dall'Italia sarà di 20 euro.

```
1 double costo_spedizione = 10.00; // a
2 if(paese == "Italia")
3 if(provincia == "Catania") // b
4 costo_spedizione = 15.0;
5 else // c
6 costo_spedizione = 20;

costo_spedizione = 10.0;
```



La soluzione riportata all'interno del frame rispetta le specifiche del problema (correttezza semantica)?

## Il costrutto if/else in C++

```
1 double costo_spedizione=10.00; // a
2 if(paese=="Italia") {
3     if(provincia=="Catania") { // b
4         costo_spedizione = 15.0; } // yy
5     else // c
6     costo_spedizione = 20;
```

La soluzione riportata all'interno del frame rispetta le specifiche del problema (correttezza semantica)?

NO!  $\Rightarrow$  “Dangling else” (else “penzolante”).

## Operatore condizionale

Sintassi: **(COND1 ? EXPR1 : EXPR2);**

Semantica:

**Se** il valore di verità di COND1 è true,

**allora** valuta EXPR1,

**altrimenti** valuta EXPR2.

```
IF (cond1){  
    x=EXPR1;  
} ELSE {  
    x=EXPR2;  
}
```

## Operatore condizionale

```
1 cout << "Max(x,y)=" << (x > y ? x : y) << endl;
```

EXP1

COND I

EXP2

equivalente a ...

```
1 if (x>y)
2     cout << "Max(x,y)=" << x < endl;
3 else
4     cout << "Max(x,y)=" << y < endl;
```

# Homework

Da codificare in C++ mediante controlli condizionali (if/else e/o operatore condizionale). NB: si assuma che le variabili menzionate siano già state definite ed opportunamente inizializzate.

**H10.1.** Siano  $a$ ,  $b$  e  $c$  tre variabili di tipo int. Trovare il massimo dei tre numeri usando l'operatore condizionale.

**H10.2.** Siano  $a$ ,  $b$  e  $c$  tre variabili di tipo int ed  $str$  una variabile di tipo string. Se  $a$  è diverso da  $c$  e la lunghezza della stringa  $str$  è minore di 8, allora copia in  $b$  la somma di  $a$  e  $c$ ; Se  $a$  è uguale a  $c$  e la lunghezza della stringa  $str$  è minore di 8, allora copia in  $a$  il valore  $c - b$ ; in tutti gli altri casi ponи  $a$ ,  $b$  e  $c$  a zero.

**H10.3.** Siano  $s1$ ,  $s2$  ed  $s3$  tre oggetti di tipo string e  $mys$  un altro oggetto di tipo string.

- Se (A) la lunghezza di una delle tre stringhe  $s1$ ,  $s2$ ,  $s3$  è maggiore di 10 e se (B) almeno una di esse è lunga almeno 20 caratteri, allora copia nella variabile di tipo string denominata  $mys$  la concatenazione dei primi tre caratteri in  $s2$  e degli ultimi tre caratteri in  $s3$ .
- Se la condizione A è verificata ma non la B, allora copia in  $mys$  la concatenazione delle tre stringhe, in ordine.
- In tutti gli altri casi stampa la somma delle lunghezze delle tre stringhe.

## Confronti lessicografici

---

## Confronti lessicografici

È possibile usare gli operatori relazionali per confrontare stringhe e singoli caratteri.

```
1     char a = 'a'; char b = 'b';
2     cout << (a < b ? "a < b" : "b > a") << endl;
```

```
1     string myname = "Pippo";
2     string yourname = "Paperino";
3     string selected;
4     if(myname < yourname)
5         selected = nyname;
6     else
7         selected = yourname;
```

FRANCO  
FLANEI

= =

FEDDEO

## Confronti lessicografici

Date due stringhe  $S_1$  e  $S_2$  quale sarà il valore di verità corrispondente alla espressione  $\underline{S_1} < \underline{S_2}$ ?

Le due stringhe vengono confrontate carattere per carattere ( $S_1[1]$  vs  $S_2[1]$ ,  $S_1[2]$  vs  $S_2[2]$  e così via ...). Il confronto termina quando:

- **due caratteri  $a \in S_1$  e  $b \in S_2$  aventi lo stesso indice sono differenti**, allora  $S_1 < S_2$  se  $a < b$ , ovvero se il carattere  $a$  precede  $b$  nell'ordinamento dei caratteri, e viceversa.
- Si raggiunge la fine di una delle stringhe, dopo una sequenza di caratteri identici; **Allora la stringa più corta sarà considerata la "minore".**

GIANPAOLO  
|||||  
GIAN

## Confronti lessicografici

Un carattere *a* si dice minore del carattere *b* se il valore con il quale esso viene codificato (tipicamente rappresentato con 1 byte) è minore del corrispondente valore di *b*.

**Codifica ASCII** ( American Standard Code for Information Interchange - 1968)

- 7 bit (+1 per controllo di parità);
- codifica da 0 a 127
- caratteri “stampabili” da 32 a 126;
- il resto sono caratteri “non stampabili” come DELETE (127) oppure il carattere “NULL” ('\0');

## Confronti lessicografici

Esempio: codifica di "A": 65. Codifica di "a": 97. Quindi il confronto ('A' < 'a') avrà valore true.

### H10.4

Scrivere del codice C++ per testare le seguenti disuguaglianze:

"sale" < "sole"? VERO

"uova" < "suola"? FALSO

"Marco" < "marco"? VERO

"asta" < "canasta"?

"123prova" < "Abaco"?

## **Costrutto switch**

---

## Costrutto switch

~~if (digit == 0);~~

```
1  if (digit == 1) { digit_name = "one"; }
2  else if (digit == 2) { digit_name = "two"; }
3  else if (digit == 3) { digit_name = "three"; }
4  else if (digit == 4) { digit_name = "four"; }
5  else if (digit == 5) { digit_name = "five"; }
6  else if (digit == 6) { digit_name = "six"; }
7  else if (digit == 7) { digit_name = "seven"; }
8  else if (digit == 8) { digit_name = "eight"; }
9  else if (digit == 9) { digit_name = "nine"; }
10 else { digit_name = ""; }
```

Tedioso, poco leggibile...

## Costrutto switch

```
1 switch (digit)
2 {
3     case 1:
4         digit_name = "one";
5         break;
6     case 2:
7         digit_name = "two";
8         break;
9     // altri case ... ~
10    default:
11        digit_name = "NO DIGIT";
12        break;
13 }
```

CASE 3:  
digit\_name = "THREE";  
Break;

## Costrutto switch

```
1  switch ( digit )
2  {
3      case 1:
4          digit_name = "one";
5          break;
6      case 2:
7          digit_name = "two";
8          break;
9      // altri case ...
10 }
```

Se digit assume effettivamente un valore tra quelli inseriti nei costrutti case, il flusso seguirà il blocco di istruzioni che si trova tra **i due punti ed l'istruzione break**.

## Costrutto switch

```
1  switch ( digit )
2  {
3      case 1:
4          digit_name = "one";
5          //break; Cosa accadrebbe ?
6      case 2:
7          digit_name = "two";
8          break;
9      // altri case ...
10 }
```

**In assenza della istruzione break**, verranno eseguite anche le istruzioni del case successivo, ed eventualmente di quello dopo, fino alla prima istruzione break!!

## Costrutto switch

In casi come questo l'assenza dei break viene sfruttata opportunamente..

```
1  switch (digit) {  
2      case 1:    "1"  
3      case 2:  
4      case 3:  
5      case 4:  
6          digit_name = "A number less than five";  
7          break;  
8          //Altri casi...  
9      }  
10 }
```

Switch (MATH) }  
CASE "GIOVANNI":

## Costrutto switch

```
1  switch ( digit )
2  {
3      case 1:
4          digit_name = "One";
5          break;
6      // altri case ...
7      default:
8          digit_name = "NO DIGIT";
9          break;
10 }
```

Il blocco di codice in corrispondenza di **default** sarà eseguito se il valore della variabile digit non è presente negli altri casi;

# Costrutto switch

Anche caratteri!

```
1  switch (char_digit)
2  {
3      case ('a'):
4          cout << "Your choice is the first one!" << "\n";
5          break;
6      case ('b'):
7          cout << "Your choice is the second one!" << "\n";
8          break;
9      // altri case ...
10     default():
11         digit_name = "NO VALID SELECTION";
12         break;
13 }
```

## Costrutto switch

### Homework H10.5

Codificare un programma in C++ che chiede all'utente di inserire un carattere. Il programma dovrà dare il seguente output:

- Se il carattere è una vocale minuscola, stampa il numero che rappresenta la sua codifica;
- Se il carattere è una vocale maiuscola, stampa il carattere stesso sullo standard output;
- Se il carattere è un numero compreso tra 1 e 3, stampa il numero stesso moltiplicato 10;

# Esempi svolti

## Esempi svolti

A10\_00\_if.cpp

A10\_01\_conditional.cpp

A10\_02\_switch.cpp

A10\_03\_compareStrings.cpp

A10\_04\_compareChars.cpp

# Hand Tracing

---

Fare *Hand Tracing* significa **simulare manualmente l'esecuzione di un programma**.

Si costruisce una **tabella** come segue::

- la prima riga riporta i nomi delle **variabili di interesse**;
- le righe successive servono per **mantenere traccia (tracing)** dei valori delle variabili di interesse;
- **una nuova riga va creata** ogni volta il valore di una o piu' variabili di interesse subisce una variazione.

## Hand Tracing: Esempio E10.1

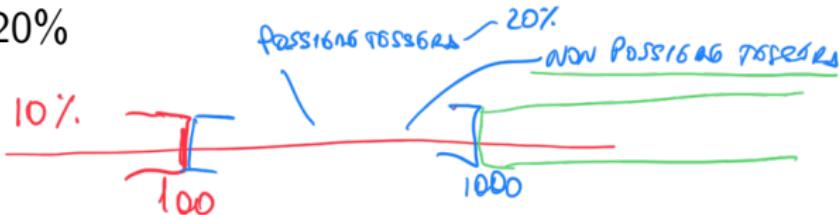
Scrivere un programma che prende i seguenti input da tastiera:

- l'importo totale di un ordine
- un ulteriore dato in input che indica se uno o più articoli hanno un prezzo promozionale,
- infine, un ulteriore input che indica se l'ordinante ha una tessera cliente.

## Hand Tracing: Esempio E10.1

Il programma dovrà stampare l'importo totale dell'ordine aggiornato in base alle seguenti specifiche:

- A Se l'importo dell'ordine e' minore o uguale a 100 euro, applica uno sconto del 10%.
- B Se l'importo dell'ordine e' maggiore di 100 euro e minore o uguale a 1000 euro, allora
  - B1 se l'utente possiede la tessera cliente applica uno sconto del 20%



## Hand Tracing: Esempio E10.1

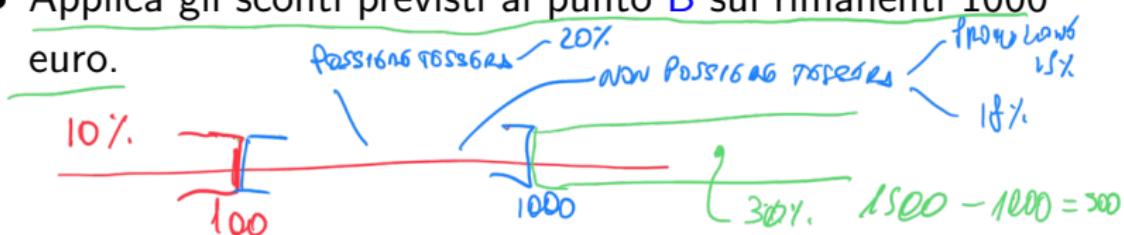
B2 se l'utente non possiede la tessera cliente:

B21 se ad uno o piu' articoli e' stato applicato un prezzo promozionale, applica uno sconto del 15%;

B22 altrimenti applica uno sconto del 18%;

C Se l'importo dell'ordine e' maggiore di 1000 euro

- applica uno sconto del 30% alla differenza tra l'importo totale dell'ordine e 1000 euro.
- Applica gli sconti previsti al punto B sui rimanenti 1000 euro.



## Hand Tracing: Esempio E10.1

Soluzione al problema in C++.

A10\_05\_ordini.cpp

# Hand Tracing: Esempio E10.1

## Rappresentazione di alcune costanti

```
1 #define SCONTO_A 0.10
2 #define SCONTO_B1 0.20
3 #define SCONTO_B21 0.15
4 #define SCONTO_B22 0.18
5 #define SCONTO_C 0.30
6 #define SOGLIA_A 100
7 #define SOGLIA_B 1000
8 #define YES 'Y'
9 #define NO 'N'
10 int main(){
11     //...
12 }
```

## Hand Tracing: Esempio E10.1

Equivalente alle direttive #define

```
1 const double SCONTO_A=0.10
2
3 const double SCONTO_B1=0.20
4 const double SCONTO_B2=0.15
5 const double SCONTO_B3=0.18
6
7 const double SCONTO_C=0.30
8
9 const double SOGLIA_A=100
10 const double SOGLIA_B=1000
11
12 const char YES='Y'
13 const char NO='N'
```

# Hand Tracing: Esempio E10.1

## Variabili

```
1 double totale_ordine = 0;  
2 double eccesso_B = -1;  
3 bool tessera = false;  
4 bool articoli_in_promozione = false;  
5 char risposta;
```

## Hand Tracing: Esempio E10.1

Codice per la gestione dello input da tastiera

```
1 cout << "Inserire totale ordine: " << endl;
2 cin >> totale_ordine;
3 cout << "Articoli in promozione (Y/N)? " << endl;
4 cin >> risposta;
5
6 if( risposta!=YES)
7     articoli_in_promozione = false;
8 else articoli_in_promozione = true;
9
10 cout << "Tessera (Y/N)? " << endl;
11 cin >> risposta;
12 if( risposta!=YES)
13     tessera = false;
14 else tessera = true;
```

# Hand Tracing: Esempio E10.1

## Parte centrale della soluzione

```
1 if (totale_ordine <= SOGLIA_A)
2     totale_ordine = totale_ordine * (1 - SCONTI_A);
3 else { //casi B e C
4     eccesso_B = totale_ordine - SOGLIA_B;
5     totale_ordine = (totale_ordine >= SOGLIA_B ? SOGLIA_B \
6         : totale_ordine);
7     if (tessera)
8         totale_ordine = totale_ordine * (1.0 - SCONTI_B1);
9     else if (articoli_in_promozione)
10        totale_ordine = totale_ordine * (1.0 - SCONTI_B21);
11    else
12        totale_ordine = totale_ordine * (1.0 - SCONTI_B22);
13    if (eccesso_B > 0)
14        totale_ordine += eccesso_B * (1-SCONTI_C);
15 }
```

## Hand Tracing: Esempio E10.1

Tabella di hand tracing

<b>totale_ordine</b>	<b>eccesso_B</b>	<b>tessera</b>	<b>articoli_in_promozione</b>
...	...	...	...

**Input di test:**

Ordine di 900 euro.

Il cliente possiede la tessera.

All'interno dell'ordine non ci sono articoli in promozione.

## Hand Tracing: Esempio E10.1

```
double totale_ordine = 0;  
double eccesso_B = -1;  
bool tessera = false;  
bool articoli_in_promozione = false;  
char risposta;
```

totale_ordine	eccesso_B	tessera	articoli_in_promozione
0	-1	false	false

## Hand Tracing: Esempio E10.1

Input (totale ordine, tessera, promozioni)

```
cout << "Inserire totale ordine: " << endl;
cin >> totale_ordine;
// ...
if (risposta != YES)
    tessera = false;
else tessera = true;
```

(Per codice completo vedi ([slide](#)))

Dopo esecuzione intero blocco di gestione input avremo:

totale_ordine	eccesso_B	tessera	articoli_in_promozione
0 900	-1 -1	false true	false false

## Hand Tracing: Esempio E10.1

```
if (totale_ordine <= SOGLIA_A)
    // ...
else { // casi B e C
    eccesso_B = totale_ordine - SOGLIA_B;
    900           1000
```

totale_ordine	eccesso_B	tessera	articoli_in_promozione
0	-1	false	false
900	-1	true	false
900	-100	true	false

## Hand Tracing: Esempio E10.1

```
totale_ordine = ( totale_ordine >= SOGLIA_B ? SOGLIA_B \
: totale_ordine);
```

totale_ordine	eccesso_B	tessera	articoli_in_promozione
0	-1	false	false
900	-1	true	false
900	-100	true	false
900	-100	true	false

## Hand Tracing: Esempio E10.1

```
if (tessera)
    totale_ordine = totale_ordine * (1.0 - SCONTI_B1);
```

totale_ordine	eccesso_B	tessera	articoli_in_promozione
0	-1	false	false
900	-1	true	false
900	-1	true	false
900	-100	true	false
720	-100	true	false

NB: 20% di 900 è 180

## Hand Tracing: Esempio E10.1

```
else if(articoli_in_promozione) // NO
    totale_ordine = totale_ordine * (1.0 - SCONTI_B21);
else // NO
    totale_ordine = totale_ordine * (1.0 - SCONTI_B22);
if(eccesso_B > 0) // Non verificata
    totale_ordine += eccesso_B * (1-SCONTI_C); // FINE
```

totale_ordine	eccesso_B	tessera	articoli_in_promozione
0	-1	false	false
900	-1	true	false
900	-1	true	false
900	-100	true	false
<b>720</b>	<b>-100</b>	<b>true</b>	<b>false</b>

A Eseguire lo Hand Tracing della soluzione presentata nello esempio **E10.1** per i seguenti input:

- Ordine di 80 euro, l'utente ha la tessera cliente e non sono presenti articoli in promozione;
- Ordine di 100 euro, l'utente non ha la tessera cliente e non sono presenti articoli in promozione;
- Ordine di 1000 euro, l'utente ha la tessera cliente e sono presenti articoli in promozione;
- Ordine di 1500 euro, l'utente non ha la tessera e non sono presenti articoli in promozione;

- B Disegnare un diagramma di flusso per rappresentare la soluzione in C++ proposta per l'esempio **E10.1**.
- C Proporre una soluzione alternativa per la risoluzione del problema descritto in **E10.1**. Disegnare un diagramma di flusso o a blocchi per tale soluzione.

Un falegname realizza scaffali di legno per ambienti interni o esterni. I clienti si recano presso il suo laboratorio con alcune richieste (input) in base alle quali il falegname opera alcune scelte:

1. ambiente: esterno o interno
2. massimo carico (in kg) che il singolo ripiano deve essere in grado di sopportare;
3. lunghezza in metri di ogni ripiano;

## Homework H10.7

---

Il falegname dovrà operare nel modo seguente:

1. Se lo scaffale va sistemato all'esterno allora va impiegato legno di castagno, altrimenti legno di pino;
2. lo spessore di ogni ripiano va calcolato in base alla formula  $S = [B + \max(0, L - 1) \times P] \times Q$ , dove:
  - $L$  è la lunghezza (in metri) di ogni ripiano fornita dal cliente;
  - $B = 0.018$  metri se si usa il castagno,  $B = 0.02$  metri se si usa il pino;
  - $P = 0.02$  per il castagno,  $P = 0.022$  per il pino

## Homework H10.7

---

- $Q = 1.1$  se il legno scelto è il castagno e se il massimo carico che il singolo ripiano deve supportare è maggiore di 100 kg, altrimenti  $Q = 1.0$ ;
- $Q = 1.2$  se il legno scelto è il pino e se il carico massimo che il singolo ripiano deve supportare è maggiore di 80kg, altrimenti  $Q = 1.0$ .

## Homework H10.7

Descrivere una soluzione del problema tale che, dati in input i parametri ambiente, lunghezza L e carico massimo del singolo ripiano dello scaffale, produca in output:

- essenza da usare (pino o castagno)
- spessore ripiani.

In particolare:

1. Realizzare un diagramma di flusso (o a blocchi) per la risoluzione del problema del falegname;
2. Codificare in C++ un programma che si base sul diagramma di flusso realizzato per il punto precedente;

3. Testare il programma in C++ mediante hand tracing per i seguenti input:
  - a Carico max 80kg, scaffale non destinato all'esterno, lunghezza ripiani 150cm;
  - b Carico max 100kg, scaffale destinato all'esterno, lunghezza ripiani 100cm;
  - c Carico max 120kg, scaffale non destinato all'esterno, lunghezza ripiani 100cm;

## Operatori logici

---

# Operatori logici

---

Gli operatori logici o booleani permettono di combinare opportunamente espressioni booleane. Essi sono

- AND. In C++ è denotato con `&&` (operatore binario)
- OR. In C++ è denotato con `||` (operatore binario)
- NOT. In C++ è denotato con `!` (operatore unario)

# Operatori logici

A	B	A <u>&amp;&amp;</u> B	A    B
true	true	true	true
false	true	false	true
true	false	false	true
false	false	false	false

A	! A
false	true
true	false

# Operatori logici

Altri operatori che si ottengono dai precedenti

A	B	NAND(A,B)	NOR(X,Y)	XOR(X,Y)
→ true	true	false	false	false ✓
false	true	true	false	true
true	false	true	false	true
→ false	false	true	true	false ✓

**Homework H10.8:** Ricavare una espressione logica per XOR facendo uso di AND, OR, NOT.

# Operatori logici

NB: Forma che fa uso di AND logico più comprensibile.

```
1 alpha=1.0;  
2 if(a!=2){ // a non deve essere né 2 né 3!!  
3   if(a!=3)  
4     alpha = 0.5; //  
5   }  
6 else  
7   alpha=1.0;
```

```
1 if (a!=2 && a!=3) // equivalente  
2   alpha = 0.5;  
3 else  
4   alpha=1.0;
```

# Operatori logici

NB: Forma che fa uso di OR logico più comprensibile.

```
1 if (a==2) // se a e' 2
2   alpha = 1.0; ↗
3 else if (a==3) // oppure a e' 3
4   alpha=1.0; ↗
5 else
6   alpha = 0.5; ↗
```

Σ

A      B

```
1 if (A || B) // equivalente
2   alpha = 1.0;
3 else
4   alpha=0.5; ↗
```

A	B
T	F
F	T
I	I
T	F

# Operatori logici

La **valutazione a corto circuito o di McCarthy** di una espressione booleana **si arresta** nel momento in cui **il valore della prima espressione booleana è sufficiente** per determinare il risultato dell'espressione.

Ad esempio, se il primo argomento dell'operatore logico AND falso, allora il valore dell'intera espressione sarà false. Quindi **Exp2 non sarà valutata**.

**Exp1 && Exp2**

**La valutazione a corto circuito o di McCarthy** di una espressione booleana **si arresta** nel momento in cui **il valore dei precedenti operandi a sinistra è sufficiente** per determinare il risultato dell'espressione.

Allo stesso modo, se il primo argomento dell'operatore logico OR vero allora il risultato della intera espressione sarà true. Quindi **Exp2 non sarà valutata**.

**Exp1 || Exp2**

NB: Potrebbe essere molto dispendioso valutare Exp2..

# Operatori logici

Leggi di DeMorgan

$$1. \underline{!(A \&\& B)} \Leftrightarrow !A \mid\mid !B$$

$$2. !(A \mid\mid B) \Leftrightarrow !A \&\& !B$$

A	B	!(A&&B)	!A    !B
T	T	F	F
T	F	T	F
F	T	T	T
F	F	T	F

# FINE