



UNIVERSITÀ
degli STUDI
di CATANIA

Controlli condizionali in C

Corso di programmazione I (A-E / O-Z) AA 2023/24

Corso di Laurea Triennale in Informatica

Fabrizio Messina

fabrizio.messina@unict.it

Dipartimento di Matematica e Informatica

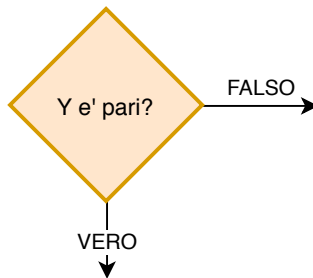
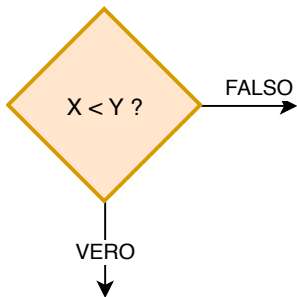
1. Costrutto if/else e operatore condizionale in C
2. Costrutto switch
3. Hand Tracing
4. Operatori logici

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 una o più blocchi `else`, si usa per **condizionare** l'esecuzione di uno o più blocchi di codice in C.

```
1    int x;  
2    printf("Inserire un numero < 10  ");  
3    scanf("%d", &x);  
4  
5    if(x>=10)  
6        printf("Numero non valido..");  
7    else printf("Il numero inserito e'  \n%d ",  x);
```

Operatori relazionali

Il costrutto if/else in C

| Notazione Matematica | C | Descrizione |
|-------------------------|--------------------|-------------------|
| $>$ | <code>></code> | Maggiore di |
| \geq | <code>>=</code> | Maggiore o uguale |
| $<$ | <code><</code> | Minore di |
| \leq | <code><=</code> | Minore o uguale |
| $=$ | <code>==</code> | Uguale |
| \neq | <code>!=</code> | Diverso |

Un esempio più articolato:

Il costrutto if/else in C

```
1    int x;
2    printf("Inserire un numero positivo < 10, ma che \
3        non sia 5!")
4    scanf("%d", &x);
5
6    if(x>=10)
7        printf(...);
8    else if(x==5)
9        printf(...);
10   else if(x<=0)
11       printf(...);
12   else    //eseguito se prec. condizioni non verificate
13       // ...
```

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    printf("Inserire un numero positivo < 10: ");
4    scanf("%d", &x);
5
6    if(x>=10) {
7        result = (alpha * x) / 10;
8        printf("Numero inserito maggiore o uguale a 10!")
9    }
10   else {
11       result = alpha * x;
12       printf("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        printf("Numero inserito maggiore di 10!");  
4    };  
5    else {  
6        result = alpha * x;  
7        printf("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        printf("Numero inserito maggiore di 10!");  
4    }  
5    else; {  
6        result = alpha * x;  
7        printf("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          printf("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, 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==IT)
3      if (provincia==CT) // b
4          costo_spedizione = 15.0;
5  else // c
6      costo_spedizione = 20;
```

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;
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").

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

Semantica:

Se il valore di verità di COND1 è true,

allora valuta EXPR1,

altrimenti valuta EXPR2.

Operatore condizionale

```
1      printf("Max(x,y)=\%d" , (x > y ? x : y));
```

equivalente a ...

```
1      if (x>y)
2          printf("Max(x,y)=\%d" , x);
3      else
4          printf("Max(x,y)=\%d" , y);
```

Note: i) Codificare in C mediante controlli condizionali (if/else e/o operatore condizionale); ii) le variabili menzionate vanno definite ed inizializzate con dati a piacere.

H9.1. Siano a , b e c tre variabili di tipo `int`. Trovare il massimo dei tre numeri usando l'operatore condizionale.

H9.2. Siano a , b e c tre variabili di tipo `int` e k un numero intero inserito dall'utente. Se a è diverso da c e $k > 8$, copia in b la somma di a e c ; Se a è uguale a c e $k < 8$, copia in a il valore $c - b$; in tutti gli altri casi stampa il valore $(a + b + c)/3$ ed infine copia il valore zero in a , b e c .

Costrutto switch

Costrutto switch

```
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 }
```

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


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          printf("Your choice is the first one!");
5          break;
6      case 'b':
7          printf("Your choice is the second one!");
8          break;
9      // altri case ...
10     default:
11         digit_name = NO_VALID_SELECTION;
12         break;
13 }
```

Homework H9.3

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 rappresenta un numero compreso tra 1 e 3, stampa il numero stesso moltiplicato per 10;

Esempi svolti

10_00_if.c

10_01_conditional.c

10_02_switch.c

10_04_printChars.c

Hand Tracing

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

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

- la prima riga (header) 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 qual volta il valore di una variabile di interesse subisce una variazione.

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.

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%

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.

Soluzione al problema in C.

```
10_05_ordini.c
```

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 E9.1

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'
```

Alcune definizioni

```
1  #define TRUE 1
2  #define FALSE 0
```

Variabili

```
1  double totale_ordine = 0;
2  double eccesso_B = -1;
3  unsigned short int tessera = FALSE;
4  unsigned short int articoli_in_promozione = FALSE;
5  char risposta;
```

Hand Tracing: Esempio E9.1

```
1  printf(" Inserire totale ordine: ");
2  scanf("%1c", totale_ordine);
3  printf(" Articoli in promozione (Y/N)? ");
4  scanf("%1c", risposta);
5
6  if(risposta!=YES)
7      articoli_in_promozione = FALSE;
8  else articoli_in_promozione = TRUE;
9
10 printf("\n Tessera (Y/N)? ");
11 scanf("%1c", risposta);
12 if(risposta!=YES)
13     tessera = FALSE;
14 else tessera = TRUE;
```

Hand Tracing: Esempio E9.1

Parte centrale della soluzione

```
1  if (totale_ordine <= SOGLIA_A)
2      totale_ordine = totale_ordine * (1 - SCONTO_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 - SCONTO_B1);
9      else if (articoli_in_promozione)
10         totale_ordine = totale_ordine * (1.0 - SCONTO_B21);
11     else
12         totale_ordine = totale_ordine * (1.0 - SCONTO_B22);
13     if (eccesso_B > 0)
14         totale_ordine += eccesso_B * (1-SCONTO_C);
15 }
```

Tabella di hand tracing

| totale_ordine | eccesso_B | tessera | articoli_in_promozione |
|----------------------|------------------|----------------|-------------------------------|
| ... | ... | ... | ... |

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 E9.1

```
double totale_ordine = 0;  
double eccesso_B = -1;  
unsigned short int tessera = FALSE;  
unsigned short int articoli_in_promozione = FALSE;  
char risposta;
```

| totale_ordine | eccesso_B | tessera | articoli_in_promozione |
|----------------------|------------------|----------------|-------------------------------|
| 0 | -1 | false | false |

Hand Tracing: Esempio E9.1

Input (totale ordine, tessera, promozioni)

```
printf("Inserire totale ordine: ");  
scanf("%f", 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 | -1 | false | false |
| 900 | -1 | true | false |

Hand Tracing: Esempio E9.1

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

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

Hand Tracing: Esempio E9.1

```
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 E9.1

```
if (tessera)
    totale_ordine = totale_ordine * (1.0 - SCONTO_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 E9.1

```
else if(articoli_in_promozione) // NO
    totale_ordine = totale_ordine * (1.0 - SCONTO_B21);
else // NO
    totale_ordine = totale_ordine * (1.0 - SCONTO_B22);
if(eccesso_B > 0) // Non verificata
    totale_ordine += eccesso_B * (1-SCONTO_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 |
| 720 | -100 | true | false |

A Usare la tecnica di Hand Tracing della soluzione presentata nell'esempio **E9.1** 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 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 **E9.1**. Disegnare un diagramma di flusso o a blocchi per tale soluzione.

Un falegname realizza scaffali in 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. carico massimo (in kg) che il singolo ripiano deve essere in grado di sopportare;
3. lunghezza in metri di ogni ripiano;

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

- $Q = 1.1$ 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 H9.5

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. Impiegare la tecnica di hand tracing per verificare il funzionamento del programma 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

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

- AND. In linguaggio C ecco è "&&" (operatore binario)
- OR. In linguaggio C esso è "||" (operatore binario)
- NOT. In linguaggio C esso è "!" (operatore unario)

| A | B | A && 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 |

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 H9.6: Ricavare una espressione logica per XOR facendo uso di AND, OR, NOT.

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

```
1  double alpha=1.0; int a =...;  
2  if(a!=2) // a non deve essere ne 2 ne 3!!  
3      if(a!=3)  
4          alpha = 0.5;
```

```
1  double alpha=1.0; int a =...;  
2  if(a!=2 && a!=3) // equivalente  
3      alpha = 0.5;
```

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;
```

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

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** è falso, allora il valore dell'intera espressione sarà falso. 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** è vero allora il risultato della intera espressione sarà true. Quindi **Exp2 non sarà valutata**.

Exp1 || Exp2

NB: Potrebbe essere molto dispendioso valutare Exp2..

Leggi di DeMorgan

$$1. \neg(A \ \&\& \ B) \Leftrightarrow \neg A \ || \ \neg B$$

$$2. \neg(A \ || \ B) \Leftrightarrow \neg A \ \&\& \ \neg B$$

Homework **H9.7**.

Quale dei tre operatori “&&”, “||” e “!” ha precedenza maggiore?
Quale dei tre ha invece precedenza minore?

Codificare alcuni test a piacere per verificare la mutua precedenza tra gli operatori.

FINE