Programmazione in C

Unità
Scelte ed alternative

Scelte ed alternative

- Il controllo di flusso
- Istruzione `if-else`
- Condizioni complesse
- Istruzioni `if-else` annidate
- Istruzione `switch`
- Esercizi proposti
- Sommario

Riferimenti al materiale

- Testi
  - Kernighan & Ritchie: capitolo 3
  - Cabodi, Quer, Sonza Reorda: capitoli 2, 4
  - Dietel & Dietel: capitoli 2, 3
- Dispense
  - Scheda: "Istruzioni di scelta in C"

Scelte ed alternative

Il controllo di flusso

Il controllo di flusso

- Concetto di flusso e di scelta
- Rappresentazione grafica
- Condizioni booleane semplici
- Esempio

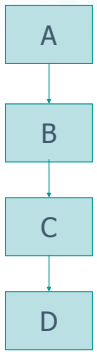
Il controllo di flusso

Concetto di flusso e di scelta

Flusso di esecuzione lineare

- Il programma C viene eseguito, di norma dalla prima istruzione all'ultima
- Il flusso di esecuzione è quindi normalmente di tipo lineare

```
istruzione_A  ;
istruzione_B  ;
istruzione_C  ;
istruzione_D  ;
```



Eccezioni

- In taluni casi il flusso di esecuzione deve variare:
  - in funzione del valore di qualche variabile di ingresso, occorre fare operazioni diverse
  - una certa operazione deve essere ripetuta più volte
  - alcune parti del programma vengono attivate solo se l'utente lo richiede
  - ...
- In tal caso, il flusso di esecuzione non segue più esattamente l'ordine di scrittura delle istruzioni nel codice sorgente

Scelte

- Il tipo più semplice di flusso non lineare è costituito dalle scelte (o alternative)

Se il voto è minore di 18,
allora prenota il prossimo appello,
altrimenti vai in vacanza.
Se il voto è maggiore di 28,
prenota il ristorante.

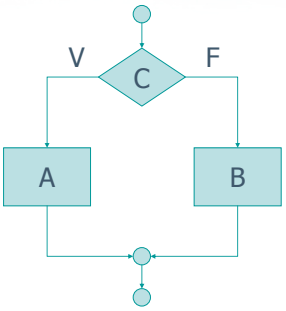
Anatomia di una scelta

- Una condizione di scelta è caratterizzata da tre informazioni:
  - la "condizione" che determina la scelta (il voto è minore di 18)
  - le "operazioni" da svolgere qualora la condizione sia "vera" (prenota il prossimo appello)
  - le "operazioni" da svolgere qualora la condizione sia "falsa" (vai in vacanza)
- Le "operazioni" potrebbero anche essere assenti

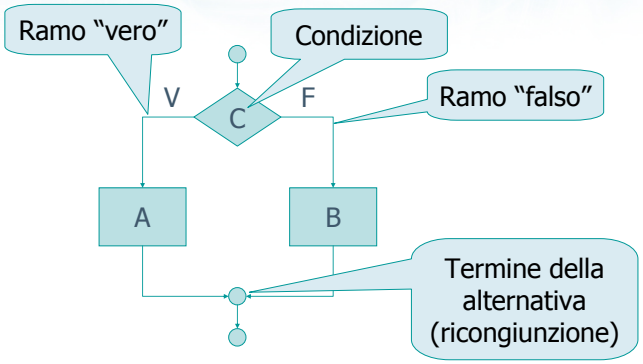
Il controllo di flusso

Rappresentazione grafica

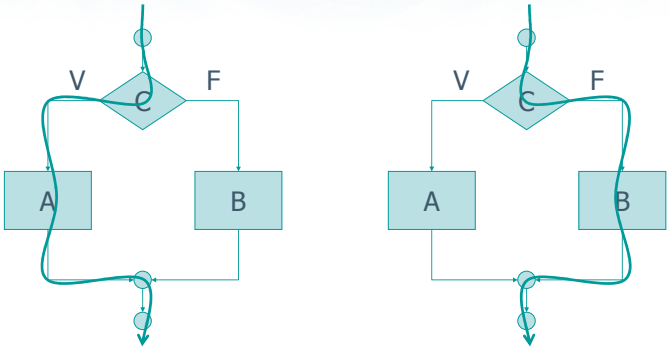
Notazione grafica



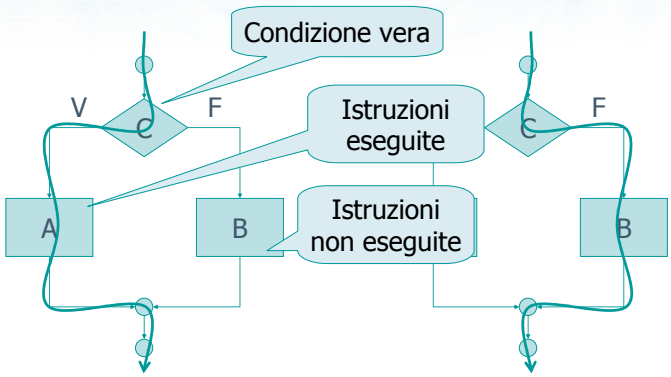
Notazione grafica



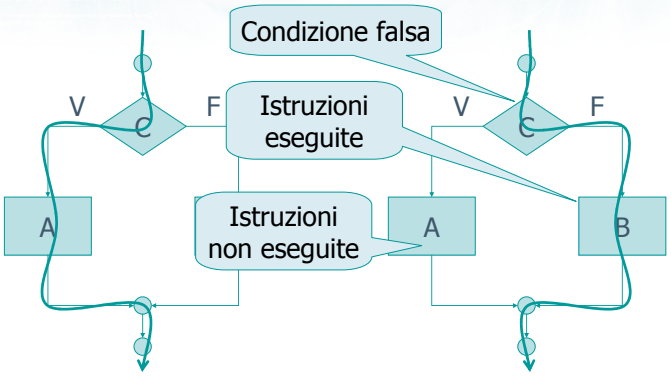
Flussi di esecuzione



Flussi di esecuzione



Flussi di esecuzione



Il controllo di flusso

Condizioni booleane semplici

Il concetto di condizione

- Che tipo di espressioni si utilizzano per esprimere la condizione C ?
  - devono dare una risposta univoca
    - Vero
    - Falso
  - sono basate sui valori delle variabili del programma
- Le espressioni di questo tipo sono dette booleane in quando generano dei valori che rispettano le leggi della logica di Boole (Vero o Falso)

Condizioni di confronto semplici

- Spesso le condizioni sono espresse sotto forma di confronti
- Confronto di uguaglianza
  - Uguale
  - Diverso
- Confronto di ordine
  - Maggiore
  - Minore
  - Maggiore o uguale
  - Minore o uguale

Il controllo di flusso

Esempio

Equazioni di primo grado

- Analizziamo il flusso di esecuzione di un programma in grado di risolvere le equazioni di primo grado:
- Data l'equazione
  - a x + b = 0

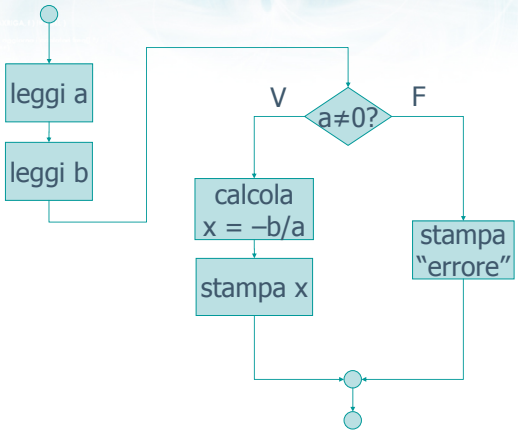
  con a e b inseriti da tastiera, determinare il valore di x che risolve l'equazione

Equazione risolutiva

- Dai corsi di matematica sappiamo che la soluzione dell'equazione:
  - $a\,x + b = 0$

  si può esprimere mediante la formula:
  - $x = -\,b / a$
- Tale formula è valida solo se $a \neq 0$
- Il programma dovrà comportarsi in modo diverso a seconda che a sia nullo o no

Soluzione



Scelte ed alternative

Istruzione `if-else`

Istruzione `if-else`

- Sintassi dell'istruzione
- Operatori di confronto
- Esercizio proposto di esempio
- Risoluzione esercizio (parte I)
- Esecuzione del programma
- Completamento esercizio
- Risoluzione esercizio (parte II)

Istruzione `if-else`

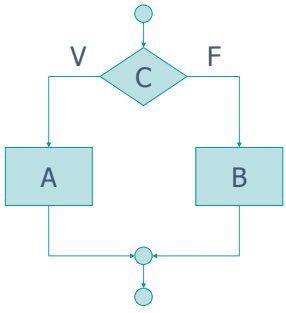
Sintassi dell'istruzione

Istruzioni di scelta in C

- L'operazione di scelta avviene mediante l'istruzione `if-else`
- Le condizioni di scelta possono essere semplici od elaborate
- Le scelte possono essere liberamente combinate tra loro, annidate
- In alcuni casi si può usare l'istruzione `switch` (trattata nella lezione "Istruzione `switch`")

Istruzione `if-else`

```
if ( C )
{
   A ;
}
else
{
   B ;
}
```



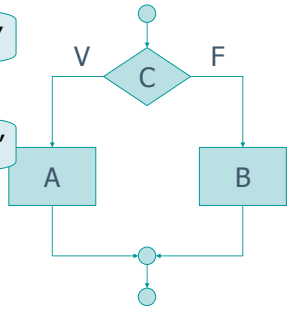
Istruzione `if-else`

```
if ( C )
{
   A ;
}
else
{
   B ;
}
```

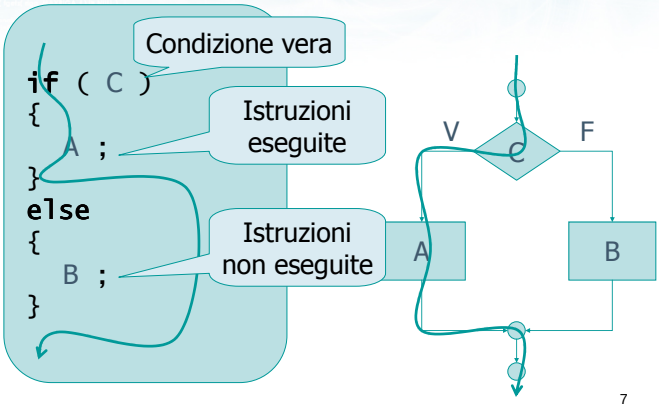
Condizione

Ramo "vero"

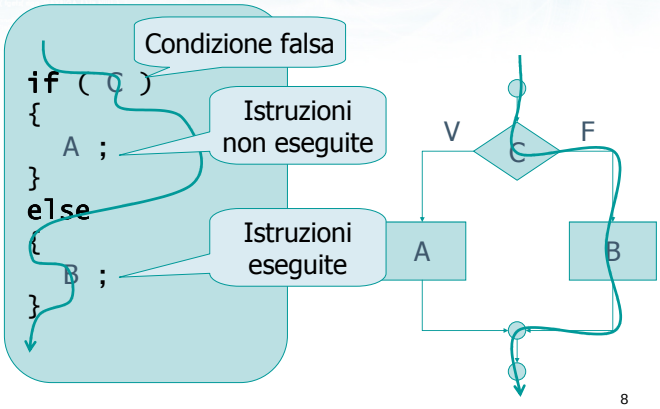
Ramo "falso"



Condizione vera



Condizione falsa



Esempio

es-posneg.c

```
int a, b ;

printf("Immetti un numero: ");
scanf("%d", &a) ;
if ( a > 0 )
{
   printf("positivo\n") ;
   b = a ;
}
else
{
   printf("negativo\n") ;
   b = -a ;
}
printf("Il valore assoluto di %d e' %d", a, b) ;
```


Suggerimento

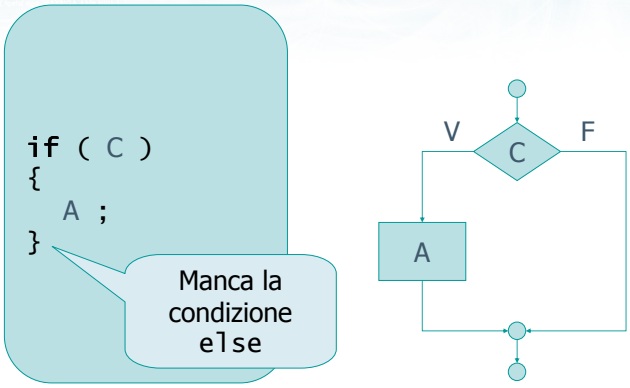
- Ad ogni nuova parentesi graffa aperta {, inserire degli spazi aggiuntivi ad inizio riga
- Tale tecnica, detta indentazione, permette una migliore leggibilità del codice
- È visivamente immediato riconoscere l'inizio e la fine dei blocchi di istruzioni
- Molti ambienti di sviluppo hanno funzioni di indentazione automatica o semi-automatica

Note

```
if ( C )
{
   A ;
}
else
{
   B ;
}
```

- La condizione C può essere semplice o complessa
- Il blocco A può essere composto da una sola istruzione, o da più istruzioni
- Il blocco B può essere composto da una sola istruzione, o da più istruzioni

Caso particolare: istruzione `if`



Esempio

es-valabs.c

```
int a ;

printf("Immetti un numero: ");
scanf("%d", &a) ;
if ( a < 0 )
{
   /* è negativo, gli cambio segno */
   a = -a ;
}
printf("Il valore assoluto e' %d", a) ;
```


Caso particolare: parentesi graffe

```
if ( C )
   A ;
else
{
   B ;
}
```

```
if ( C )
{
   A ;
}
else
   B ;
```

- Se il blocco A è composto da una sola istruzione, allora le parentesi graffe relative si possono omettere.
- Lo stesso vale per il blocco B.

```
if ( C )
   A ;
else
   B ;
```


Esempio

es-valabs2.c

```
int a ;

printf("Immetti un numero: ");
scanf("%d", &a) ;

if ( a < 0 ) /* è negativo, gli cambio segno */
   a = -a ;

printf("Il valore assoluto e' %d", a) ;
```


Errore frequente

- È errato mettere il simbolo di punto-e-virgola ; dopo l'istruzione `if`

```
if ( a > 0 ) ;
   a = -a ;
```

viene interpretato come

```
if ( a > 0 )
   /*nulla*/ ;
a = -a ;
```

forma corretta

```
if ( a > 0 )
   a = -a ;
```


Errore frequente

- È errato fidarsi della sola indentazione

```
if ( a > 0 )
   printf("neg");
   a = -a ;
```

viene interpretato come

```
if ( a > 0 )
   printf("neg");
a = -a ;
```

forma corretta

```
if ( a > 0 )
{
   printf("neg");
   a = -a ;
}
```


Suggerimento

- Anche se vi è una sola istruzione nel blocco "vero" o nel blocco "falso", utilizzare sempre le parentesi graffe
- In tal modo il programma sarà più leggibile, e sarà più facile aggiungere eventuali nuove istruzioni senza incappare in errori

Istruzione `if-else`

Operatori di confronto

Le condizioni

- La condizione C è solitamente basata su un'operazione di confronto
  - determinare se una variabile è uguale o meno a zero, o a un altro valore costante
  - determinare se una variabile è uguale ad un'altra variabile, o è diversa, o è maggiore, o minore, ...
  - determinare se una *espressione*, calcolata a partire da una o più variabili, è uguale, diversa, maggiore, minore, ... di una costante, o una variabile, o un'altra espressione

Operatori di confronto in C

- Uguaglianza
  - Uguale: `a == b`
  - Diverso: `a != b`
- Ordine
  - Maggiore: `a > b`
  - Minore: `a < b`
  - Maggiore o uguale: `a >= b`
  - Minore o uguale: `a <= b`

Esempi

- `if( a == 0 ) ...`
- `if( a == b ) ...`
- `if( a < 0 ) ...`
- `if( a+b > 3 ) ...`
- `if( x*y != y*x ) ...`
- `if( a/2 == (a+1)/2 ) ...`

Esempi

- `if( a == 0 ) ...`
- `if( a == b ) ...`
- `if( a < 0 ) ...`
- `if( a+b > 3 ) ...`
- `if( x*y != y*x ) ...`
- `if( a/2 == (a+1)/2 ) ...`

Esempi

- `if( a == 0 ) ...`
- `if( a == b ) ...`
- `if( a < 0 ) ...`
- `if( a+b > 3 ) ...`
- `if( x*y != y*x ) ...`
- `if( a/2 == (a+1)/2 ) ...`

Esempi

- `if( a == 0 ) ...`
- `if( a == b ) ...`
- `if( a < 0 ) ...`
- `if( a+b > 3 ) ...`
- `if( x*y != y*x ) ...`
- `if( a/2 == (a+1)/2 ) ...`

Esempi

- `if( a == 0 ) ...`
- `if( a == b ) ...`
- `if( a < 0 ) ...`
- `if( a+b > 3 ) ...`
- `if( x*y != y*x ) ...`
- `if( a/2 == (a+1)/2 ) ...`

Esempi

- `if( a == 0 ) ...`
- `if( a == b ) ...`
- `if( a < 0 ) ...`
- `if( a+b > 3 ) ...`
- `if( x*y != y*x ) ...`
- `if( a/2 == (a+1)/2 ) ...`

Errore frequente

- Confondere l'operatore di assegnazione = con l'operatore di confronto ==
- Regola pratica: le parentesi tonde richiedono ==
- Regola pratica: il punto-e-virgola richiede =

```
/* assegnazione */
a = b + c ;

/* confronto */
if ( a == b+c )...
```

```
/* assegnazione */
a == b + c ;

/* confronto */
if ( a = b+c )...
```


Istruzione `if-else`

Esercizio proposto di esempio

Esercizio "Controlla A e B"

- Si scriva un programma in linguaggio C che legga due numeri da tastiera, detti A e B, e determini le seguenti informazioni, stampandole a video:
  - determini se B è un numero positivo o negativo
  - determini se A è un numero pari o dispari
  - calcoli il valore di A+B
  - determini quale scelta dei segni nell'espressione (±A) + (±B) porta al risultato massimo, e quale è questo valore massimo.

Struttura generale

- Leggi A e B
- Controlla il segno di B
  - Stampa il messaggio opportuno
- Controlla la parità di A
  - Stampa il messaggio opportuno
- Calcola A+B
  - Stampa il risultato
- ...l'ultimo punto è più difficile
  - ...ci pensiamo dopo!

Lettura dei dati

- Leggi A e B

```
int a, b ;

printf("Immetti A");
scanf("%d", &a) ;

printf("Immetti B");
scanf("%d", &b) ;
```


Controllo del segno

- Controlla il segno di B
  - Stampa il messaggio opportuno

```
if( b > 0 )
{
   printf("B e' positivo\n");
}
else
{
   printf("B e' negativo o nullo\n");
}
```


Controllo della parità

- Controlla la parità di A
  - Stampa il messaggio opportuno

```
if( "a è pari" )
{
   printf("A e' pari\n");
}
else
{
   printf("A e' dispari\n");
}
```


Numero pari

- Come determinare se un numero è pari?
- Calcoliamo la divisione per 2, e controlliamo il resto
  - Se il resto della divisione per 2 vale 0, allora il numero è pari
  - Se il resto della divisione per 2 vale 1, allora il numero è dispari
- Il calcolo del resto si ottiene con l'operatore %

```
if( "a è pari" )
```

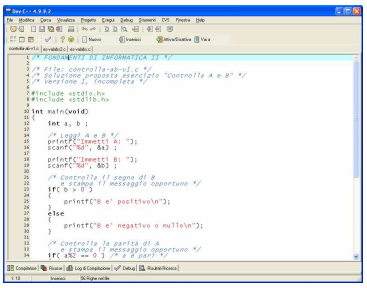
```
if( (a % 2) == 0 )
```


Istruzione if-else

Risoluzione esercizio (parte I)

Risoluzione esercizio "Controlla A e B"

- Risolviamo interattivamente l'esercizio

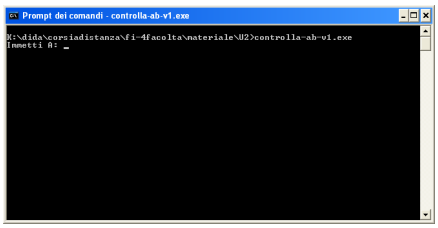


Istruzione if-else

Esecuzione del programma

Verifica esercizio "Controlla A e B"

- Compiliamo il programma
- Eseguiamolo con alcuni dati di prova, verificandone il comportamento corretto



Istruzione if-else

Completamento esercizio

Completamento esercizio "Controlla A e B"

- Abbiamo verificato il corretto funzionamento
- Rimane da implementare il punto:
  - determini quale scelta dei segni nell'espressione (±A) + (±B) porta al risultato massimo, e quale è questo valore massimo.
- Appaiono possibili diverse strategie:
  - Calcolare le 4 combinazioni e scegliere il massimo
  - Riscrivere algebricamente l'espressione

Strategia 1

- La prima strategia prevede di calcolare:
  - R1 = (+A) + (+B)
  - R2 = (+A) + (–B)
  - R3 = (–A) + (+B)
  - R4 = (–A) + (–B)
- Dopo avere calcolato queste 4 variabili, occorre confrontarle per determinare quale è maggiore di tutte le altre.
- In questo caso è inutilmente macchinoso.

Strategia 2

- Ragionando algebricamente, la massima somma che si può ottenere dall'espressione (±A) + (±B) sarà quando
  - ±A è positivo
  - ±B è positivo
- In altre parole, è sufficiente calcolare la somma dei valori assoluti
  - |A| + |B|

Valore assoluto

- Il valore assoluto di una variabile è pari a
  - il valore dell'opposto della variabile
    - se la variabile è negativa
  - il valore della variabile stessa
    - se la variabile è positiva

Valore assoluto

- Il valore assoluto di una variabile è pari a
  - il valore dell'opposto della variabile
    - se la variabile è negativa
  - il valore della variabile stessa
    - se la variabile è positiva

```
if( a<0 )
{
   va = -a ;
}
else
{
   va = a ;
}
```


Valore assoluto

- Il valore assoluto di una variabile è pari a
  - il valore dell'opposto della variabile
    - se la variabile è negativa
  - il valore della variabile stessa
    - se la variabile è positiva

```
if( a<0 )
{
   va = -a ;
}
else
{
   va = a ;
}
```

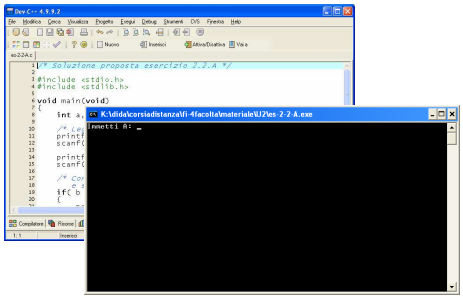
```
if( a<0 )
{
   a = -a ;
}
```


Istruzione `if-else`

Risoluzione esercizio (parte II)

Risoluzione esercizio "Controlla A e B"

- Completiamo l'esercizio, codificandolo e verificandolo



Scelte ed alternative

Condizioni complesse

Condizioni complesse

- Operatori booleani
- Operatori booleani in C
- Esercizio proposto
- Verifica della soluzione

Condizioni complesse

Operatori booleani

Logica Booleana

- Le condizioni "semplici" (es. confronti) forniscono un valore booleano (vero/falso)
- Spesso occorre prendere delle scelte in funzione del valore di più condizioni semplici
  - Es: x è compreso tra a e b?
    - x ≥ a, e contemporaneamente x ≤ b
  - Es: ci sono promossi?
    - voto1 ≥ 18, oppure voto2 ≥ 18
- A questo scopo si possono usare gli operatori booleani

Operatori booleani

- Date due qualsiasi condizioni booleane X ed Y (condizioni semplici, o a loro volta complesse):
- X **AND** Y
  - è vero se sia X che Y sono veri
- X **OR** Y
  - è vero se è vero X (indipendentemente da Y) oppure se è vero Y (indipendentemente da X) o se sono veri entrambi
- **NOT** X
  - è vero se X è falso, è falso se X è vero

Esempi

- x è compreso tra a e b?
  - (x ≥ a) **AND** (x ≤ b)
    - se so già che b ≥ a
  - ((b ≥ a) **AND** (x ≥ a) **AND** (x ≤ b)) **OR** ((b < a) **AND** (x ≤ a) **AND** (x ≥ b))
    - nel caso generale
- ci sono promossi?
  - (voto1 ≥ 18) **OR** (voto2 ≥ 18)

Condizioni complesse

Operatori booleani in C

Operatori booleani in C

Operatore booleano	Sintassi in C	Esempio
AND	&&	(x>=a)&&(x<=b)
OR	||	(v1>=18)||(v2>=18)
NOT	!	!(a>b)

Contesto di utilizzo

- Solitamente gli operatori booleani && || ! si utilizzano all'interno della condizione dell'istruzione `if`, per costruire condizioni complesse
  - Più avanti vedremo come si usano anche nella condizione del costrutto `while`
- Tali operatori lavorano su operandi che solitamente sono:
  - Condizioni semplici (es. `(a>b)||(a!=1)`)
  - Risultati di altri operatori booleani (es. `((a>b)&&(b>c))||(c==0)`)

Precedenza degli operatori

- Quando più operatori booleani e di confronto sono presenti nella stessa espressione, vengono valutati come segue:
  - Prima gli operatori di confronto
    - == != > < >= <=
  - Poi la negazione NOT
    - !
  - In seguito la congiunzione AND
    - &&
  - Infine la disgiunzione OR
    - ||

Esempi

```
if ( a>0 && b>0 )
```


Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( !(a>0 && b>0) )
```


Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```


Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```

```
if ( b>=a && x>=a && x<=b ||
b<a && x<=a && x>=b )
```


Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```

```
if ( b>=a && x>=a && x<=b ||
b<a && x<=a && x>=b )
```


Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```

```
if ( b>=a && x>=a && x<=b ||
b<a && x<=a && x>=b )
```


Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```

```
if ( b>=a && x>=a && x<=b ||
b<a && x<=a && x>=b )
```

Esempi

```
if ( a>0 && b>0 )
```

```
if ( a<=0 || b<=0 )
```

```
if ( a==0 || (a!=0 && b==0) )
```

```
if ( b>=a && x>=a && x<=b ||
b<a && x<=a && x>=b )
```

```
if ( ((b>=a) && (x>=a) && (x<=b)) ||
     ((b<a)  && (x<=a) && (x>=b))
   )
```


Suggerimento

- In presenza di espressioni complesse, è sempre conveniente abbondare con le parentesi
  - leggibilità
  - indipendenza dalle precedenze degli operatori

~~`if ( b>=a && x>=a && x<=b || b<a && x<=a && x>=b )`~~

```
if ( ((b>=a) && (x>=a) && (x<=b)) ||
     ((b<a)  && (x<=a) && (x>=b))
   )
```


Errore frequente

- È errato usare in successione più operatori di confronto senza collegarli mediante operatori booleani

`if ( a > b > 0 )` → forma corretta → `if ( (a > b) && (b > 0) )`

`if ( a==b==c )` → forma corretta → `if ( (a == b) && (b == c) )`

Errore frequente

- È errato "sottintendere" parte di un confronto
  - Esempio: "se a o b sono diversi da uno"

`if ( a || b != 1 )` → forma corretta

`if ( (a||b) != 1 )` → forma corretta

→ `if ( (a!=1) || (b!=1) )`

Condizioni complesse

Esercizio proposto

Esercizio "Classificazione triangolo 1"

- Si scriva un programma in linguaggio C che legga da tastiera i valori delle lunghezze dei tre lati di un triangolo (detti A, B e C), e determini:
  - se il triangolo è equilatero
  - se il triangolo è isoscele
  - se il triangolo è scaleno
  - se il triangolo è rettangolo
- Nota: si assuma, per il momento, che i valori A, B, C descrivano correttamente un triangolo

Analisi del problema

- Ricordiamo le condizioni matematiche relative alla classificazione dei triangoli:
  - Equilatero: le lunghezze dei tre lati A, B, C sono uguali tra loro
  - Isoscele: le lunghezze di [almeno] due dei tre lati A, B, C sono uguali tra loro
    - ogni triangolo equilatero è anche isoscele
  - Scaleno: le lunghezze dei tre lati A, B, C sono tutte diverse tra loro
  - Rettangolo: possiede un angolo retto
    - vale il teorema di Pitagora

Espressioni matematiche (I)

- Equilatero
  - A = B = C
  - (A = B) AND (B = C) AND (A = C)
  - (A = B) AND (B = C)
- Isoscele
  - (A = B) OR (B = C) OR (A = C)
- Scaleno
  - (A ≠ B) AND (A ≠ C) AND (B ≠ C)

Espressioni matematiche (II)

- Rettangolo
  - Teorema di Pitagora
    - Ipotenusa^2 = Cateto^2 + Cateto^2
  - L'ipotenusa può essere uno qualunque dei lati A, B oppure C
  - ($A^2 = B^2+C^2$) OR ($B^2 = A^2+C^2$) OR ($C^2 = A^2+B^2$)

Condizioni in C

- Equilatero
  - `a==b && b==c`
- Isoscele
  - `a==b || b==c || a==c`
- Scaleno
  - `a!=b && b!=c && a!=c`
- Rettangolo

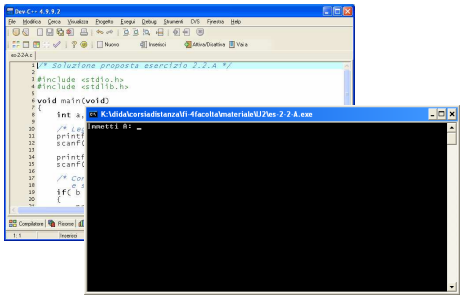
```
(a*a == b*b + c*c) ||
(b*b == a*a + c*c) ||
(c*c == a*a + b*b)
```


Condizioni complesse

Verifica della soluzione

Verifica "Classificazione triangolo 1"

- Analizziamo il codice dell'esercizio e verifichiamone il corretto funzionamento



Scelte ed alternative

Istruzioni `if-else` annidate

Istruzioni `if-else` annidate

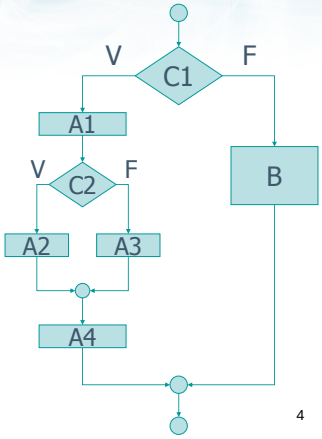
- Annidamento di istruzioni `if-else`
- Opzionalità del ramo `else`
- Catene `if-else if-...-else`
- Esercizio proposto
- Verifica della soluzione

Istruzioni `if-else` annidate

Annidamento di istruzioni `if-else`

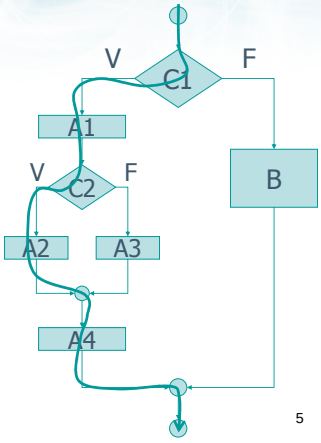
Scelte annidate

- Nelle istruzioni del blocco "vero" o del blocco "else", è possibile inserire altri blocchi di scelta
- In tal caso la seconda scelta risulta annidata all'interno della prima



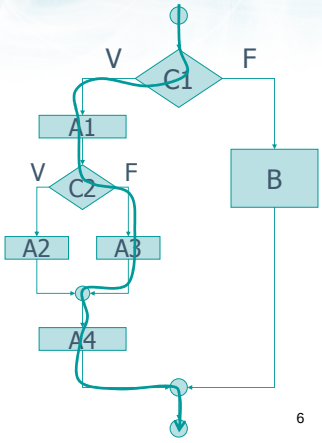
Caso 1

- C1 vero, C2 vero
  - Istruzioni eseguite: A1, A2, A4



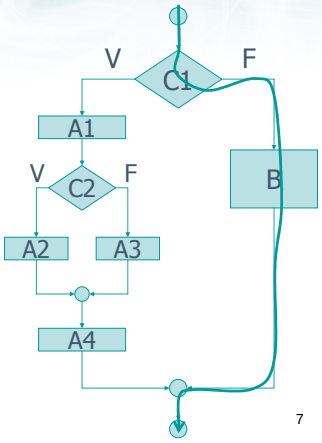
Caso 2

- C1 vero, C2 vero
  - Istruzioni eseguite: A1, A2, A4
- C1 vero, C2 falso
  - Istruzioni eseguite: A1, A3, A4



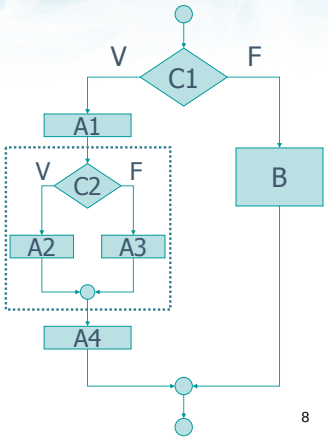
Caso 3

- C1 vero, C2 vero
  - Istruzioni eseguite: A1, A2, A4
- C1 vero, C2 falso
  - Istruzioni eseguite: A1, A3, A4
- C1 falso, C2 indifferente
  - Istruzioni eseguite: B



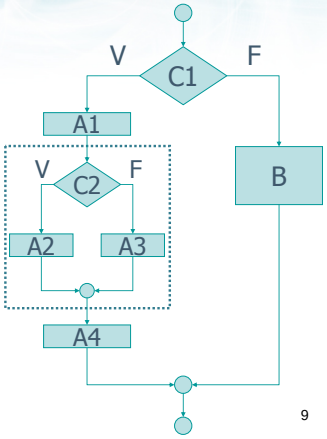
Corretto annidamento

- L'intero blocco di scelta più interno (dalla condizione fino al ricongiungimento) deve essere completamente contenuto all'interno di uno dei rami del blocco più esterno



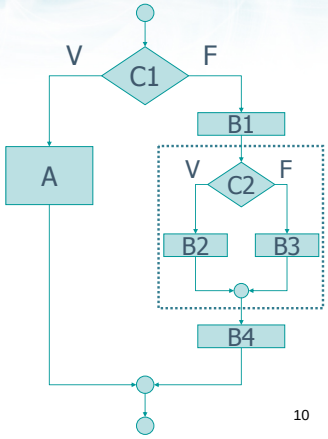
Sintassi C

```
if ( C1 )
{
   A1 ;
   if ( C2 )
   {
      A2 ;
   }
   else
   {
      A3 ;
   }
   A4 ;
}
else
{
   B ;
}
```

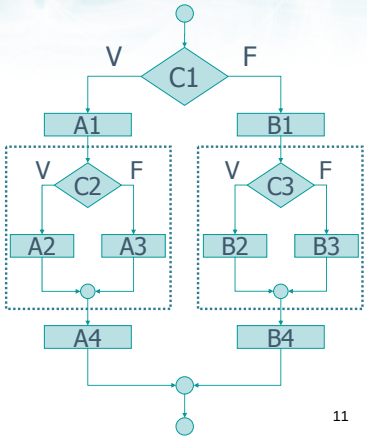


Sintassi C

```
if ( C1 )
{
   A ;
}
else
{
   B1 ;
   if ( C2 )
   {
      B2 ;
   }
   else
   {
      B3 ;
   }
   B4 ;
}
```



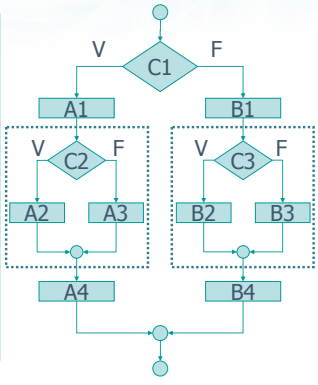
Caso generale



Sintassi C

```
if ( C1 )
{
   A1 ;
   if ( C2 )
   {
      A2 ;
   }
   else
   {
      A3 ;
   }
   A4 ;
}
```

```
else
{
   B1 ;
   if ( C3 )
   {
      B2 ;
   }
   else
   {
      B3 ;
   }
   B4 ;
}
```



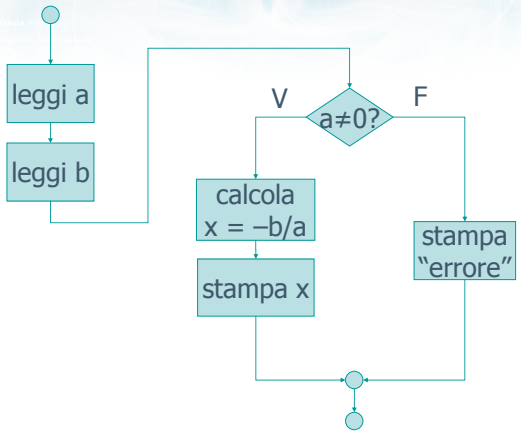
Sintassi C

- Un'istruzione `if` può comparire ovunque
  - anche all'interno del blocco "vero" o "falso" di un'altra istruzione `if`
- Occorre garantire il corretto annidamento delle istruzioni
  - le istruzioni annidate vanno completamente contenute tra le parentesi graffe {...}

Esempio

- Ricordiamo l'esercizio sull'algoritmo risolutivo delle equazioni di primo grado
  - a x + b = 0
- La soluzione è:
  - x = – b / a
    - solo se a≠0

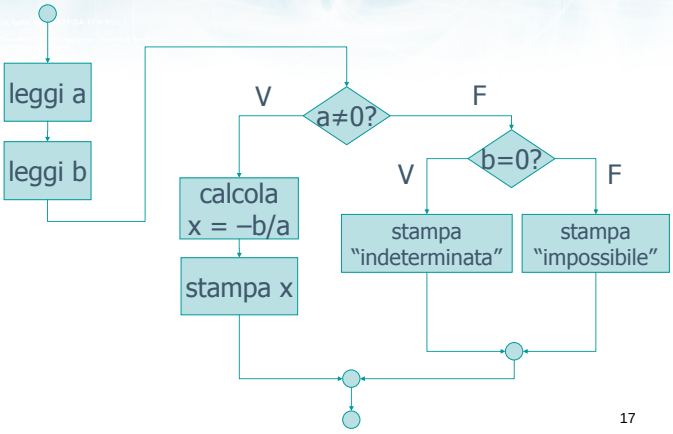
Soluzione (parziale)



Esempio

- Ricordiamo l'esercizio sull'algoritmo risolutivo delle equazioni di primo grado
  - a x + b = 0
- La soluzione è:
  - x = – b / a
    - solo se a≠0
  - x = indeterminato (infinite soluzioni)
    - se a=0 e b=0
  - x = impossibile (nessuna soluzione)
    - se a=0 e b≠0

Soluzione (completa)



Soluzione in C (1/2)

primogrado.c

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    float a, b ;
    float x ;

    printf("Risoluzione eq. di primo grado\n");
    printf("Equazione: a x + b = 0\n") ;

    /* Leggi A e B */
    printf("Immetti coefficiente a: ");
    scanf("%f", &a) ;

    printf("Immetti coefficiente b: ");
    scanf("%f", &b) ;
```

Soluzione in C (2/2)

```
    if( a != 0 )
    {
        x = - b / a ;
        printf("La soluzione e' x = %f\n", x) ;
    }
    else
    {
        if( b==0 )
        {
            printf("Equazione indeterminata\n");
        }
        else
        {
            printf("Equazione impossibile\n");
        }
    }
}
```

Istruzioni `if-else` annidate

Opzionalità del ramo `else`

Forme abbreviate

- Ricordiamo che:
  - Se il ramo "vero" oppure il ramo "falso" è composto da una sola istruzione, allora le parentesi graffe {...} sono opzionali
  - Se il ramo "falso" non contiene istruzioni, allora la clausola `else` si può omettere
- Nel contesto di `if` annidati, queste regole possono creare una potenziale ambiguità

Esempio

```
if( a>0 )
{
  c = a ;
}
else
{
  if( b>0 )
  {
    c = b ;
  }
  else
  {
    c = 0 ;
  }
}
```

=

```
if( a>0 )
  c = a ;
else
  if( b>0 )
    c = b ;
  else
    c = 0 ;
```


Esempio problematico

```
if( a>0 )
if( b>0 )
c = a + b ;
else
c = 0 ;
```

?

```
if( a>0 )
  if( b>0 )
    c = a + b ;
  else
    c = 0 ;
```

```
if( a>0 )
  if( b>0 )
    c = a + b ;
else
  c = 0 ;
```


Regola

- Ogni clausola `else`, in assenza di parentesi graffe che ne esplicitino l'attribuzione, è da intendersi riferita all'istruzione `if` più vicina (per la quale non sia ancora stata attribuita una clausola `else`)

```
if( a>0 )
if( b>0 )
c = a + b ;
else
c = 0 ;
```

```
if( a>0 )
{
  if( b>0 )
    c = a + b ;
  else
    c = 0 ;
}
```

Suggerimento

- In presenza di `if` annidate, è conveniente abbondare con le parentesi graffe

```
if( a>0 )
if( b>0 )
c = a + b ;
else
c = 0 ;
```

```
if( a>0 )
{
  if( b>0 )
    c = a + b ;
}
else
  c = 0 ;
```

```
if( a>0 )
{
  if( b>0 )
    c = a + b ;
  else
    c = 0 ;
}
```

Istruzioni `if-else` annidate

Catene `if-else` `if-...-else`

Catene di istruzioni condizionali

- Talvolta occorre verificare, in sequenza, una serie di condizioni particolari, trovando la prima condizione vera tra quelle possibili
- Esempio:
  - Dato un numero intero tra 1 e 12, che rappresenta il mese corrente, stampare il nome del mese per esteso ("Gennaio" ... "Dicembre")

Soluzione

```
if( mese == 1 )
    printf("Gennaio\n") ;
else
{
    if( mese == 2 )
        printf("Febbraio\n") ;
    else
    {
        if( mese == 3 )
            printf("Marzo\n") ;
        else
        {
            if( mese == 4 )
                printf("Aprile\n") ;
            else
            {
                ..... /* continua fino a 12 */
            }
        }
    }
}
```

mesi.c

Analisi della soluzione

```
if( mese == 1 )
    printf("Gennaio\n") ;
else
{
    if( mese == 2 )
        printf("Febbraio\n") ;
    else
    {
        if( mese == 3 )
            printf("Marzo\n") ;
        else
        {
            if( mese == 4 )
                printf("Aprile\n") ;
            else
            {
                if( mese == 5 )
                    printf("Maggio\n") ;
                else
                {
                    if( mese == 6 )
                        printf("Giugno\n") ;
                    else
                    {
                        if( mese == 7 )
                            printf("Luglio\n") ;
                        else
                        {
                            if( mese == 8 )
                                printf("Agosto\n") ;
                            else
                            {
                                if( mese == 9 )
                                    printf("Settembre\n") ;
                                else
                                {
                                    if( mese == 10 )
                                        printf("Ottobre\n") ;
                                    else
                                    {
                                        if( mese == 11 )
                                            printf("Novembre\n") ;
                                        else
                                        {
                                            if( mese == 12 )
                                                printf("Dicembre\n") ;
                                            else
                                                printf("MESE ERRATO!\n") ;
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
```

- Annidamento eccessivo
- Scarsa leggibilità
- Difficile identificazione delle {...} corrispondenti
- Esistono formattazioni migliori?

Soluzione più leggibile

```
if( mese == 1 )
    printf("Gennaio\n") ;
else if( mese == 2 )
    printf("Febbraio\n") ;
else if( mese == 3 )
    printf("Marzo\n") ;
else if( mese == 4 )
    printf("Aprile\n") ;
else if( mese == 5 )
    printf("Maggio\n") ;
.............................
else if( mese == 9 )
    printf("Settembre\n") ;
else if( mese == 10 )
    printf("Ottobre\n") ;
else if( mese == 11 )
    printf("Novembre\n") ;
else if( mese == 12 )
    printf("Dicembre\n") ;
else
    printf("MESE ERRATO!\n") ;
```

mesi2.c

In generale...

- In ogni ramo "falso" c'è una sola istruzione: la `if-else` annidata
  - anche se a sua volta contiene altre istruzioni, è comunque considerata una sola istruzione
- È possibile omettere le parentesi nel ramo `else`
- È conveniente rimuovere l'indentazione
- Ricordarsi dell'`else` finale

```
if ( C1 )
{
    A1 ;
}
else if ( C2 )
{
    A2 ;
}
else if ( C3 )
{
    A3 ;
}
.........
else
{
    An ;
}
```


Istruzioni `if-else` annidate

Esercizio proposto

Esercizio "Classificazione triangolo 2"

- Si scriva un programma in linguaggio C che legga da tastiera i valori delle lunghezze dei tre lati di un triangolo (detti A, B e C), e determini:
  - se il triangolo è equilatero
  - se il triangolo è isoscele
  - se il triangolo è scaleno
  - se il triangolo è rettangolo
- Il programma, prima di classificare il triangolo, controlli se i numeri A, B, C rappresentano correttamente un triangolo

Analisi

- Data una terna di numeri A, B, C, non è detto che si possa costruire un triangolo con tali lunghezze dei lati
  - La lunghezza di ciascun lato deve essere un numero positivo
  - Ogni lato deve essere minore della somma degli altri due
  - Ogni lato deve essere maggiore della differenza degli altri due

Struttura proposta

```
if (i lati non sono positivi)
{
    printf("errore") ;
}
else if (ogni lato non è minore della somma degli altri)
{
    printf("errore") ;
}
else if (ogni lato non è maggiore della differenza degli altri)
{
    printf("errore") ;
}
else
{
    /* caso normale */
    ...vedi triangolo.c
}
```

Condizioni booleane

- "i lati non sono positivi"
  - `a<=0 || b<=0 || c<=0`
- "ogni lato non è minore della somma degli altri"
  - `a>=b+c || b>=a+c || c>=a+b`
- "ogni lato non è maggiore della differenza degli altri"
  - attenzione: la differenza va presa in valore assoluto!
  - prima di calcolare A-B, occorre verificare che A>B, altrimenti bisogna calcolare B-A

Condizioni booleane (2)

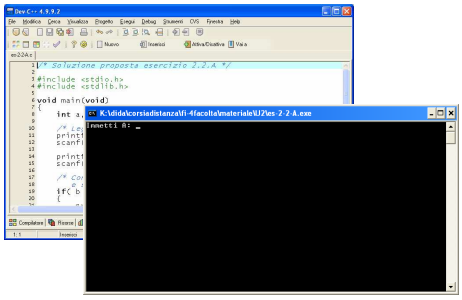
- "ogni lato non è maggiore della differenza degli altri"
  - `( (b>c)  && a <= b-c ) ||`
  - `( (b<=c) && a <= c-b ) ||`
  - `( (a>c)  && b <= a-c ) ||`
  - `( (a<=c) && b <= c-a ) ||`
  - `( (a>b)  && c <= b-a ) ||`
  - `( (a<=b) && c <= a-b )`

Istruzioni `if-else` annidate

Verifica della soluzione

Verifica "Classificazione triangolo 2"

- Analizziamo il codice dell'esercizio e verifichiamone il corretto funzionamento



Scelte ed alternative

Istruzione switch

Istruzione switch

- Sintassi dell'istruzione
- Particolarità dell'istruzione
- Esercizio proposto
- Verifica della soluzione

Istruzione switch

Sintassi dell'istruzione

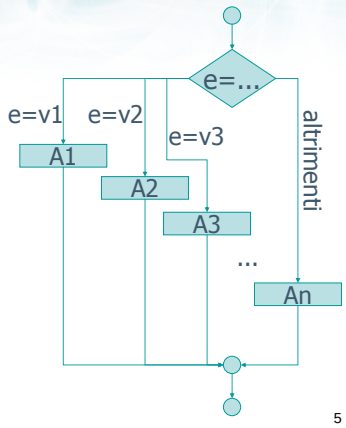
Scelte multiple

- Quando occorre compiere una sequenza di scelte, in funzione del valore di una variabile, occorre una catena di `if-else`
- Lo stesso risultato si può ottenere in forma più compatta mediante l'istruzione `switch`

```
if( mese == 1 )
    printf("Gennaio\n") ;
else if( mese == 2 )
    printf("Febbraio\n") ;
else if( mese == 3 )
    printf("Marzo\n") ;
else if( mese == 4 )
    printf("Aprile\n") ;
else if( mese == 5 )
    printf("Maggio\n") ;
.........................
else if( mese == 9 )
    printf("Settembre\n") ;
else if( mese == 10 )
    printf("Ottobre\n") ;
else if( mese == 11 )
    printf("Novembre\n") ;
else if( mese == 12 )
    printf("Dicembre\n") ;
else
    printf("MESE ERRATO!\n") ;
```

Sintassi istruzione switch

```
switch ( e )
{
    case v1:
        A1 ;
    break ;

    case v2:
        A2 ;
    break ;

    case v3:
        A3 ;
    break ;
    ..........
    default:
        An ;
}
```



Precisazioni (1/2)

- L'espressione e può essere una variabile oppure un'espressione aritmetica
  - Il tipo di dato deve essere `int`, `char` o `enum`
- Ciascun caso è identificato da una costante
  - L'espressione e viene confrontata con il valore delle costanti v1...vn
  - Il tipo di dato deve essere compatibile
- Ciascun caso è delimitato da `case`...`break`
  - Non vi sono parentesi graffe {...}

Precisazioni (2/2)

- I casi possono apparire in qualsiasi ordine
  - Devono essere tutti diversi
- Verrà selezionato al più un caso
- Il caso `default` viene valutato se e solo se nessuno degli altri casi è stato considerato
  - Opzionale, ma sempre consigliato

L'istruzione `break`

- Il significato di `break` è di portare l'esecuzione del programma fino al termine del costrutto `switch`
  - "Salta alla chiusa graffa": }
- In assenza di `break`, l'esecuzione proseguirebbe attraverso il caso successivo
  - Né il prossimo `case`, né eventuali parentesi graffe, possono fermare l'esecuzione lineare

Casi multipli

- Potrebbe essere necessario eseguire lo stesso codice in corrispondenza di diversi valori dell'espressione
- È possibile accomunare più casi, indicandoli consecutivamente

```
switch( ora )
{
  case 12:
    pranzo = 1 ;
    break;
  case 13:
    pranzo = 1 ;
    break;
}
```

→

```
switch( ora )
{
  case 12:
  case 13:
    pranzo = 1 ;
    break;
}
```


Esempio

```
switch( mese )
{
    case 1:
        printf("Gennaio\n") ;
        break ;
    case 2:
        printf("Febbraio\n") ;
        break ;
    case 3:
        printf("Marzo\n") ;
        break ;
    case 4:
        printf("Aprile\n") ;
        break ;
    ..........................
    case 12:
        printf("Dicembre\n") ;
        break ;
    default:
        printf("MESE ERRATO!\n") ;
}
```

mesi3.c

Istruzione `switch`

Particolarità dell'istruzione

Istruzione atipica

- L'istruzione `switch` è anomala sotto diversi punti di vista:
  - Non utilizza le parentesi graffe per l'annidamento delle istruzioni interne
  - Prevede solamente il controllo di uguaglianza `e==v`
  - Richiede che i valori da confrontare siano costanti
  - `break` e `default` sono opzionali
- Occorre una forte disciplina nell'utilizzarla correttamente!

Errore frequente

- È errato dimenticare l'istruzione `break` al termine di ogni caso

```
switch( ora )
{
  case 12:
    pranzo = 1 ;
  case 20:
    cena = 1 ;
    break;
}
```

viene interpretato come

```
switch( ora )
{
  case 12:
    pranzo = 1 ;
    cena = 1 ;
    break;
  case 20:
    cena = 1 ;
    break;
}
```

forma corretta

```
switch( ora )
{
  case 12:
    pranzo = 1 ;
    break;
  case 20:
    cena = 1 ;
    break;
}
```

Errore frequente

- È errato utilizzare variabili come valori dei singoli `case`
- Se non è possibile usare costanti, allora utilizzare delle catene di `if-else`

forma corretta

```
switch( ora )
{
  case orapranzo:
    pranzo = 1 ;
    break;
  case oracena:
    cena = 1 ;
    break;
}
```

```
if( ora==orapranzo )
{
  pranzo = 1 ;
}
else if( ora==oracena )
{
  cena = 1 ;
}
```

Errore frequente

- È errato pensare di poter fare confronti di ordine, o confronti multipli
- Utilizzare sequenze di `if-else`
- Non usare `else if` se i casi non sono mutuamente esclusivi

forma corretta

```
switch( ora )
{
  case <12:
    mattino = 1 ;
    break;
  case 12 || 20:
    pasti = 1 ;
    break;
}
```

```
if( ora<12 )
{
  mattino = 1 ;
}
if( ora==12 || 20 )
{
  pasti = 1 ;
}
```

Suggerimento

- Utilizzare sempre l'istruzione `default`
- Anche se non vi è ragione apparente, è opportuno che il programma intercetti valori errati della variabile

```
switch( ora )
{
  ......

  default:
    printf("Errore: valore inatteso
            %d per la variabile ora\n", ora) ;
}
```


Suggerimento

- Posizionare l'istruzione `default` come ultimo caso dello `switch`
  - Potrebbe stare ovunque
  - Maggiore leggibilità
- L'istruzione `break` finale si può omettere

Istruzione `switch`

Esercizio proposto

Esercizio "Semplice calcolatrice"

- Si scriva un programma in linguaggio C che implementi una semplice calcolatrice in grado di compiere le 4 operazioni (+ − × ÷) tra numeri interi
- Il programma presenti un semplice menù da cui l'utente indichi (con un numero tra 1 e 4) l'operazione da svolgere
- In seguito il programma acquisirà da tastiera i due operandi e stamperà il risultato dell'operazione

Soluzione (1/4)

calcola.c

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int op ;
    int a, b, c ;
    int err ;

    printf("Semplice calcolatrice\n\n") ;

    printf("Inserisci 1 per la somma\n");
    printf("Inserisci 2 per la sottrazione\n");
    printf("Inserisci 3 per la moltiplicazione\n");
    printf("Inserisci 4 per la divisione\n");

    printf("La tua scelta:") ;
    scanf("%d", &op) ;
```

Soluzione (2/4)

```
printf("Inserisci il primo operando: ") ;
scanf("%d", &a) ;

printf("Inserisci il secondo operando: ") ;
scanf("%d", &b) ;
```


Soluzione (3/4)

```
    err = 0 ;
    switch( op )
    {
        case 1:
            c = a + b ;
            break ;
        case 2:
            c = a - b ;
            break ;
        case 3:
            c = a * b ;
            break ;
        case 4:
            c = a / b ;
            break ;
        default:
            printf("Operazione errata\n") ;
            err = 1 ;
    }
```

Soluzione (3/4)

```
err = 0 ;
switch( op
{
    case 1
        c
        br
    case 2
        c
        br
    case 3
        c
        br
    case 4:
        c = a / b ;
        break ;
    default:
        printf("Operazione errata\n") ;
        err = 1 ;
}
```

```
case 4:
    if( b == 0 )
    {
        printf("Divisione per zero!\n");
        err = 1 ;
    }
    else
    {
        c = a / b ;
    }
    break ;
```

Soluzione (4/4)

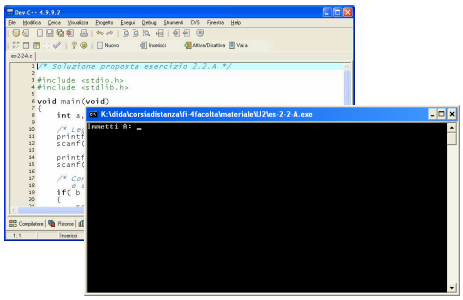
```
    if( err == 0 )
    {
        printf("Il risultato vale: %d\n", c) ;
    }

}
```

Istruzione switch

Verifica della soluzione

Verifica "Semplice calcolatrice"

- Analizziamo il codice dell'esercizio e verifichiamone il corretto funzionamento



Scelte ed alternative

Esercizi proposti

Esercizi proposti

- Esercizi sul calcolo del massimo
- Esercizio "Equazione di secondo grado"
- Esercizio "Re e Regina"

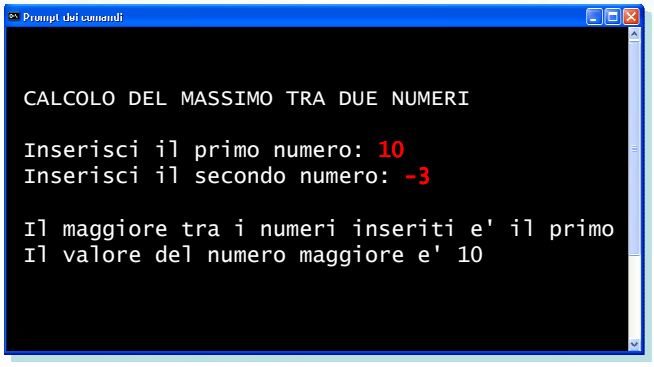
Esercizi proposti

Esercizi sul calcolo del massimo

Esercizio "Calcolo del massimo"

- Si scriva un programma in linguaggio C che acquisisca due numeri interi da tastiera e:
  - determini, stampando un messaggio opportuno quale dei due numeri (il primo o il secondo) sia maggiore
  - stampi il valore di tale numero
- Si trascuri il caso in cui i due numeri siano uguali

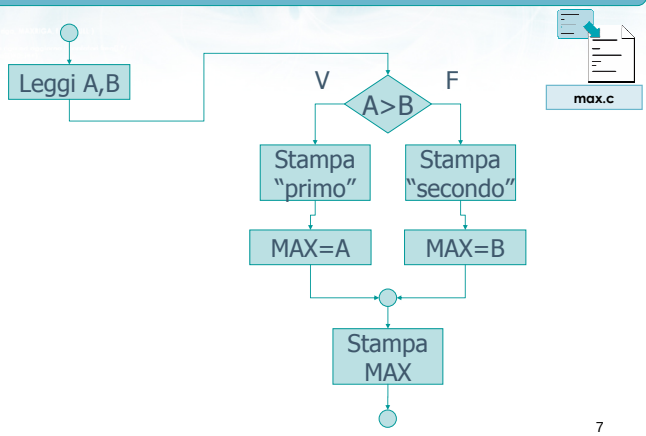
Esempio



Analisi

- Chiamiamo A e B i due numeri introdotti dall'utente
- Chiamiamo MAX il valore del maggiore tra i due
- Occorre verificare la condizione A>B
  - Se A>B, allora MAX sarà pari ad A
  - Altrimenti, MAX sarà pari a B

Diagramma di flusso



max.c

Esercizio "Calcolo del massimo a 3"

- Si scriva un programma in linguaggio C che acquisisca **tre** numeri interi da tastiera e:
  - determini, stampando un messaggio opportuno quale dei **tre** numeri (il primo, il secondo o il terzo) sia maggiore
  - stampi il valore di tale numero
- Si trascuri il caso in cui i numeri siano uguali

Esempio

```
Prompt dei comandi

CALCOLO DEL MASSIMO TRA TRE NUMERI

Inserisci il primo numero: 10
Inserisci il secondo numero: -3
Inserisci il terzo numero: 15

Il maggiore tra i numeri inseriti e' il terzo
Il valore del numero maggiore e' 15
```


Analisi

- Chiamiamo A, B, C i tre numeri inseriti
- Si può procedere in due modi
  - Usando istruzioni `if` annidate
    - se A>B, allora controlla se A>C ...

max3-if.c

Analisi

- Chiamiamo A, B, C i tre numeri inseriti
- Si può procedere in due modi
  - Usando istruzioni `if` annidate
    - se A>B, allora controlla se A>C ...

max3-if.c

  - Usando espressioni condizionali complesse
    - se A>B e A>C ...

max3-and.c

Esercizi proposti

Esercizio "Equazione di secondo grado"

Esercizio "Equazione di secondo grado"

- Data l'equazione
  - $a\,x^2 + b\,x + c = 0$

  con a, b e c inseriti da tastiera, determinare il valore (o i valori) di x che risolvono l'equazione

Analisi

- Ricordiamo la formula risolutiva per le equazioni di secondo grado
- La quantità sotto radice è il discriminante Δ
- Vi sono vari casi possibili
  - se a≠0 o a=0
  - se Δ>0, Δ=0 o Δ<0

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$\Delta = b^2 - 4ac$$

Casi possibili

Caso	Situazione	Soluzione/i
$a = 0$	Equazione di primo grado	$x = -c/b$ Impossibile se $b = 0, c \neq 0$ Indeterminata se $b = 0, c = 0$
$a \neq 0$ $\Delta > 0$	Due soluzioni reali distinte	$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
$a \neq 0$ $\Delta = 0$	Due soluzioni reali coincidenti	$x_1 = x_2 = \frac{-b}{2a}$
$a \neq 0$ $\Delta < 0$	Due soluzioni complesse coniugate	$x_{1,2} = R \pm iC$ $R = -b/2a, \quad C = \sqrt{\lvert b^2 - 4ac \rvert}/2a$

Soluzione

- Acquisire a, b, c
- Se a=0, risolvere l'equazione di primo grado
  - Ri-usiare il codice già scritto, facendo attenzione al nome delle variabili
- Calcolare il discriminante Δ
  - Il nome della variabile sarà delta
- In funzione del segno di delta, usare la formula opportuna

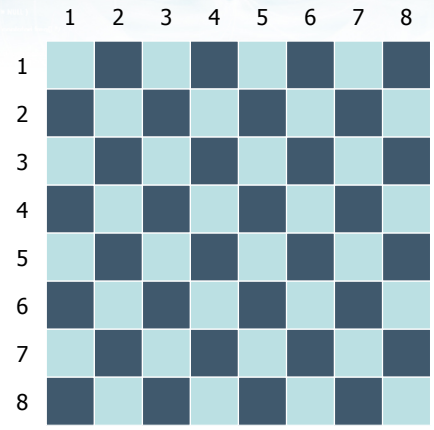
Esercizi proposti

Esercizio "Re e Regina"

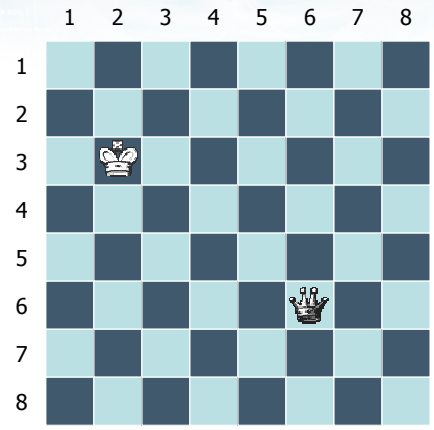
Esercizio "Re e Regina"

- Su una scacchiera 8x8 sono posizionati due pezzi: il Re bianco e la Regina nera
- Si scriva un programma in linguaggio C che, acquisite le posizioni del Re e della Regina, determini se la Regina è in posizione tale da poter mangiare il Re
  - Le posizioni dei due pezzi sono identificate mediante la riga e la colonna su cui si trovano, espresse come numeri interi tra 1 e 8

Analisi



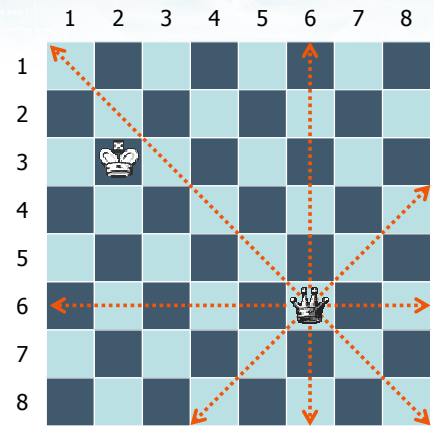
Analisi



Re bianco:
riga 3
colonna 2

Regina nera:
riga 6
colonna 6

Analisi

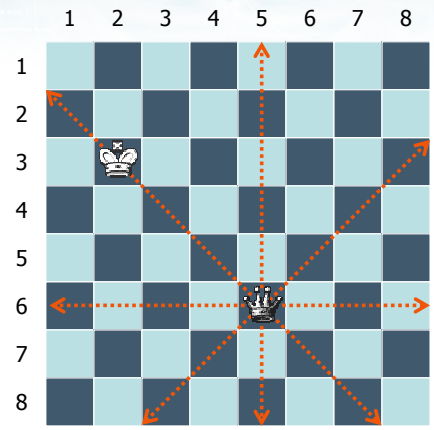


Re bianco:
riga 3
colonna 2

Regina nera:
riga 6
colonna 6

Il Re è salvo

Analisi



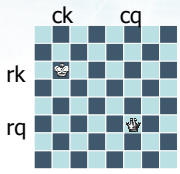
Re bianco:
riga 3
colonna 2

Regina nera:
riga 6
colonna 5

Il Re è
sotto scacco

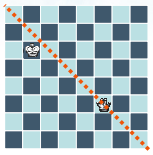
Soluzione (1/2)

- Acquisire le coordinate dei pezzi
  - Re: `rk`, `ck`
  - Regina: `rq`, `cq`
- Controllare se la Regina
  - è sulla stessa riga del Re
    - `rq==rk`
  - è sulla stessa colonna del Re
    - `cq==ck`
  - è sulla stessa diagonale discendente del Re
  - è sulla stessa diagonale ascendente del Re

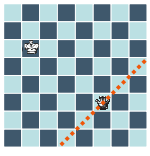


Soluzione (1/2) - Diagonali

- Diagonale discendente
  - r–c costante
  - `rq-cq == rk-ck`



- Diagonale ascendente
  - r+c costante
  - `rq+cq == rk+ck`



Soluzione (2/2)

- Conviene utilizzare una variabile logica `scacco`
  - inizializzare `scacco=0`
  - fare i vari tipi di controlli
  - se si verifica una condizione di scacco, porre `scacco=1`
- Al termine dei controlli, in funzione del valore di `scacco`, stampare il messaggio opportuno
  - se `scacco==0`, il Re è salvo
  - se `scacco==1`, il Re è sotto scacco

Scelte ed alternative

Sommario

Argomenti trattati

- Ramificazione del flusso di esecuzione
- Istruzione `if-else`
- Condizioni Booleane semplici e complesse
- Annidamento di istruzioni `if-else`
- Istruzione `switch`

Tecniche di programmazione

- Catene di istruzioni `if-else if-...-else`
- Annidamento delle istruzioni o condizioni Booleane complesse
- Uso di variabili logiche per tenere traccia delle condizioni incontrate
- Istruzione `switch` per sostituire alcuni tipi di catene `if-else if`
- Uso di `else` e `default` per catturare condizioni anomale

Suggerimenti

- Analizzare sempre tutti i casi possibili prima di iniziare a scrivere il programma
- Abbondare con le parentesi graffe
- Curare l'indentazione
- Aggiungere commenti in corrispondenza della clausola `else` e della graffa di chiusura

Materiale aggiuntivo

- Sul CD-ROM
  - Testi e soluzioni degli esercizi trattati nei lucidi
  - Scheda sintetica
  - Esercizi risolti
  - Esercizi proposti
- Esercizi proposti da altri libri di testo