Elementi Di Informatica E Programmazione

Prof. Andrea Loreggia

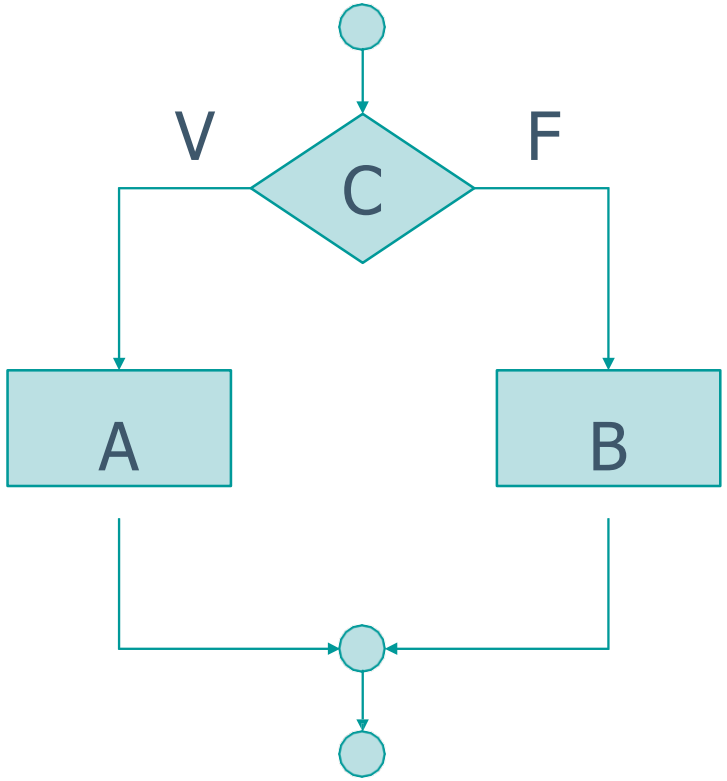
UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

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 ;
}
```



Esempio


```
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

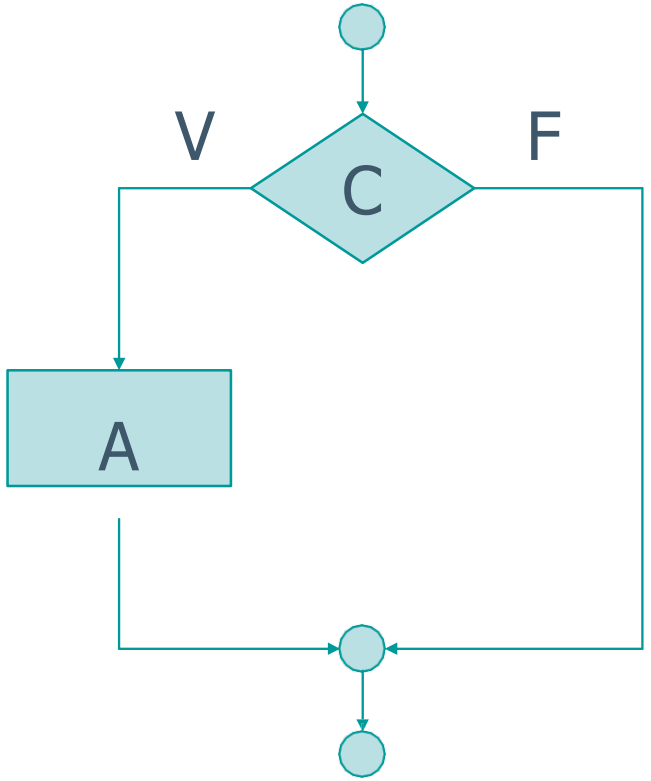
Note

- 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`

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

Manca la condizione `else`



Esempio


```
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`
  - viene interpretato come


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

forma corretta

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

```
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

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 ) ...`

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 =

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 )
```

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 ;
}
```

Condizioni complesse

UNIVERSITÀ
DEGLI STUDI
DI BRESCIA

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

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?
    - $x \geq a$, e contemporaneamente $x \leq b$
  - 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)

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
    - ||

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!=1) ||
     (b!=1) )
```

```
if ( (a||b) != 1 )
```

forma corretta

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
    - $\text{Ipotenusa}^2 = \text{Cateto}^2 + \text{Cateto}^2$
  - 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

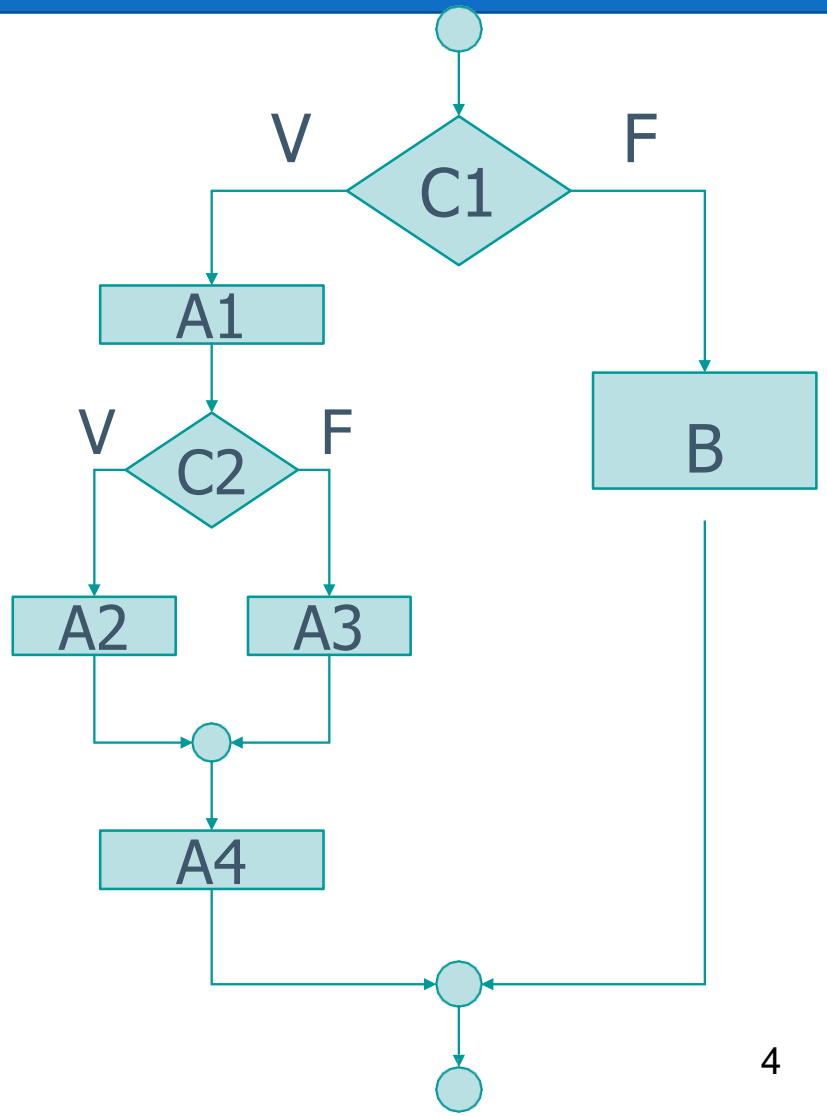
- Equilatero
  - `a==b && b==c`
- Isoscele
  - `a==b || b==c || a==c`
- Scaleno
  - `a!=b && b!=c && a!=c`
- Rettangolo
  - ```
    (a*a == b*b + c*c) ||
    (b*b == a*a + c*c) ||
    (c*c == a*a + b*b)
    ```
    ```

Istruzioni if-else annidate

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

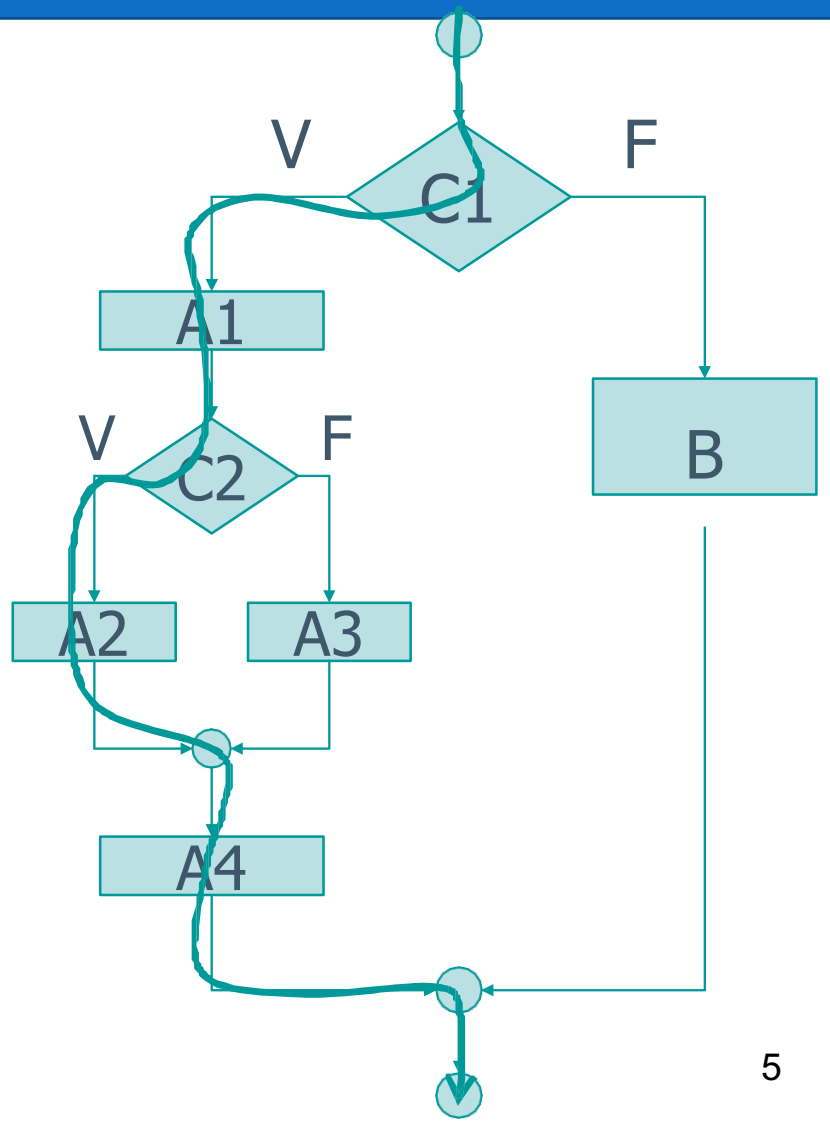
Scelte annidate

- 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



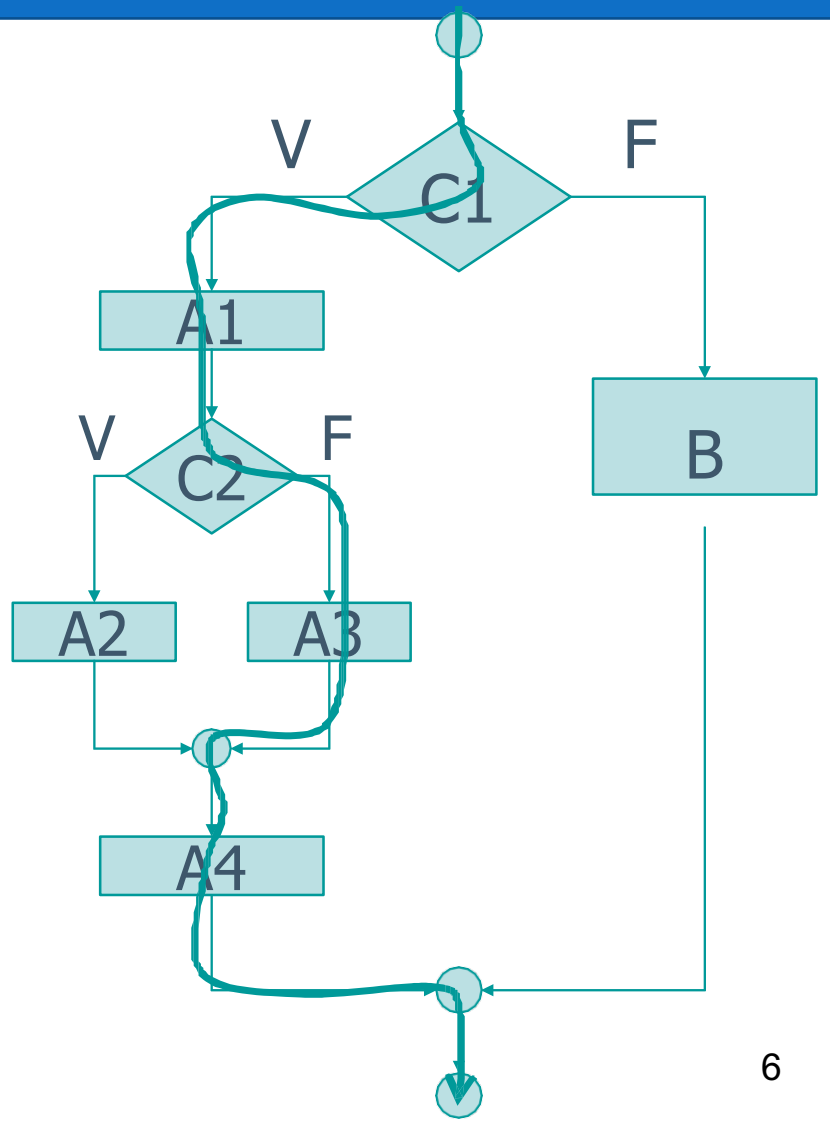
Caso 1

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



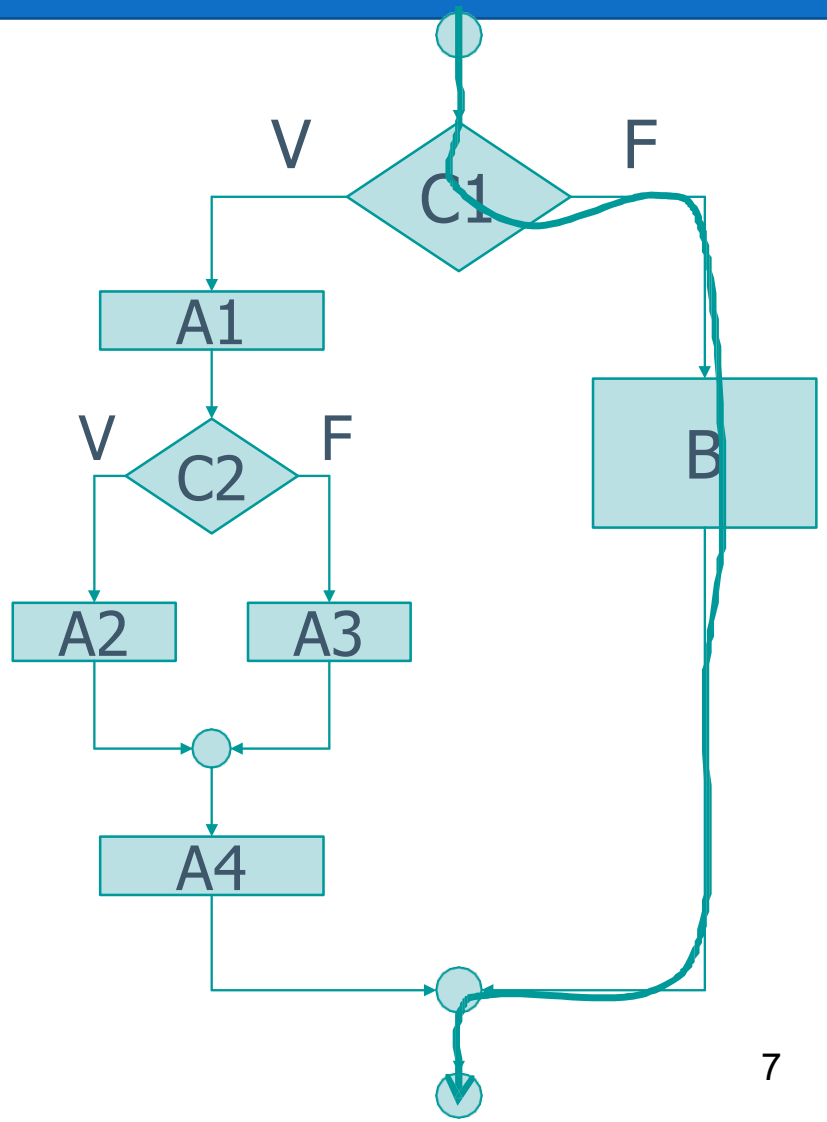
Caso 2

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



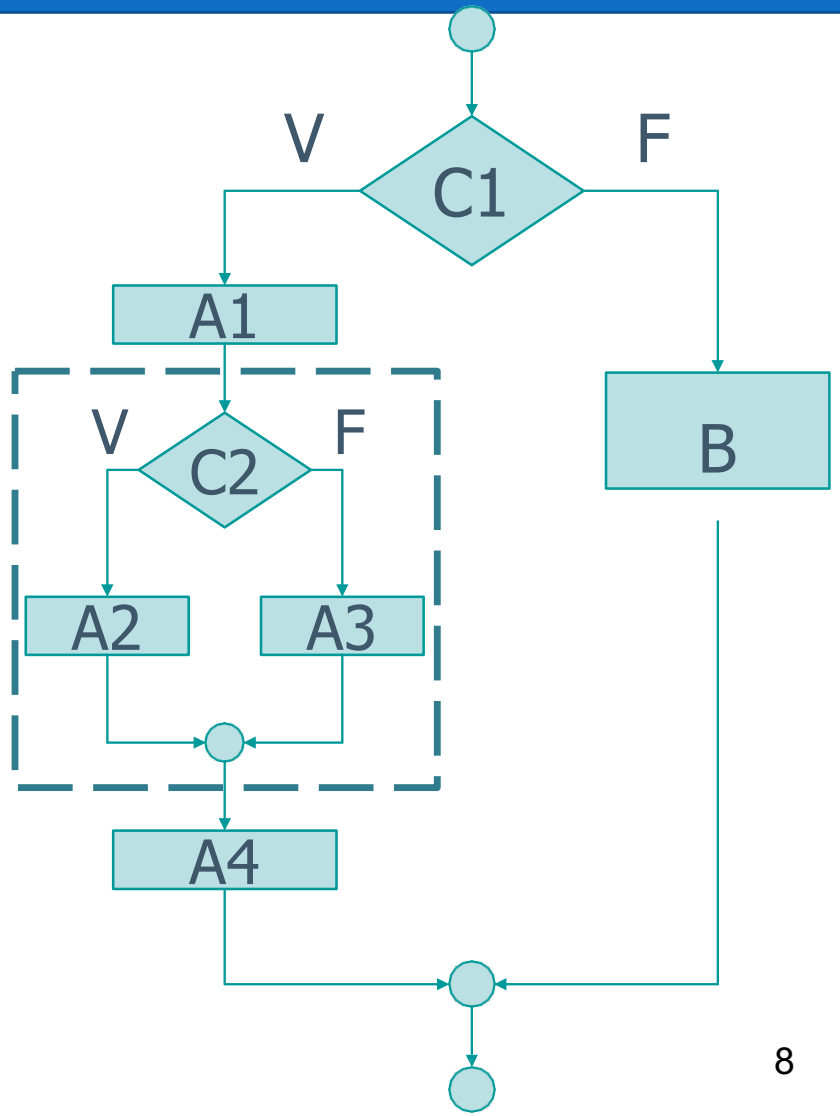
Caso 3

- 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



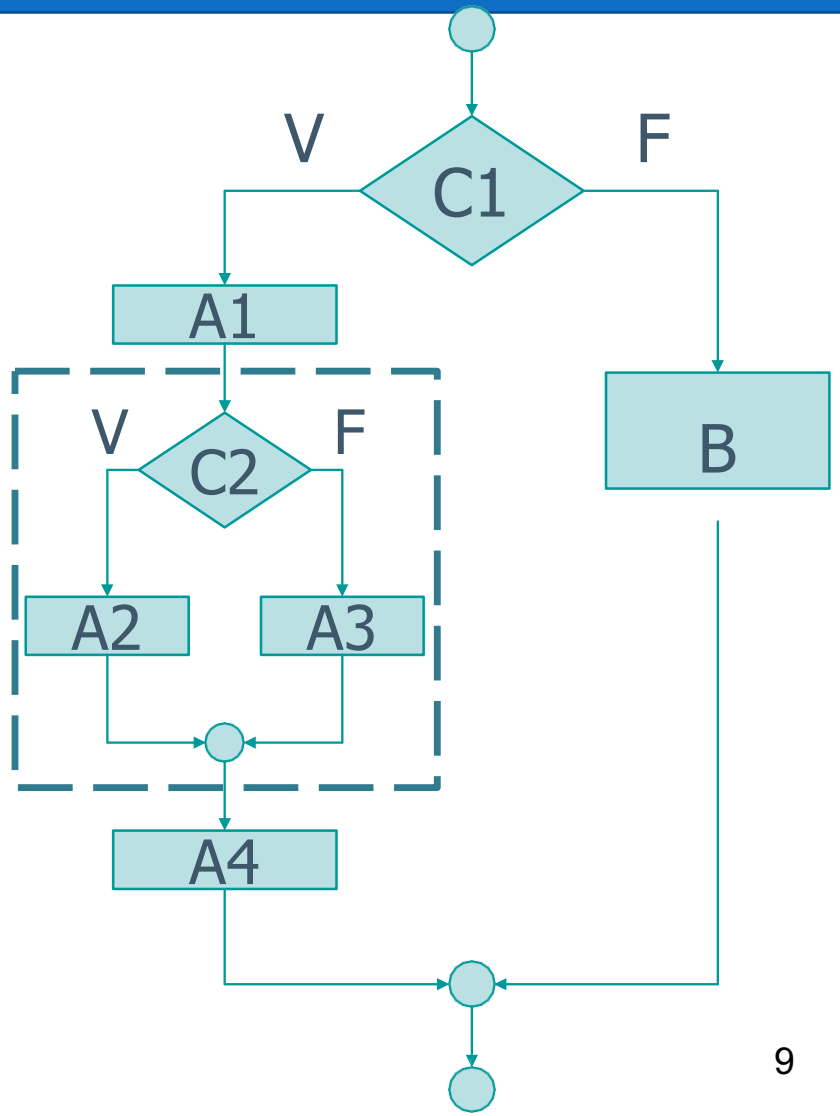
Corretto annidamento

- 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



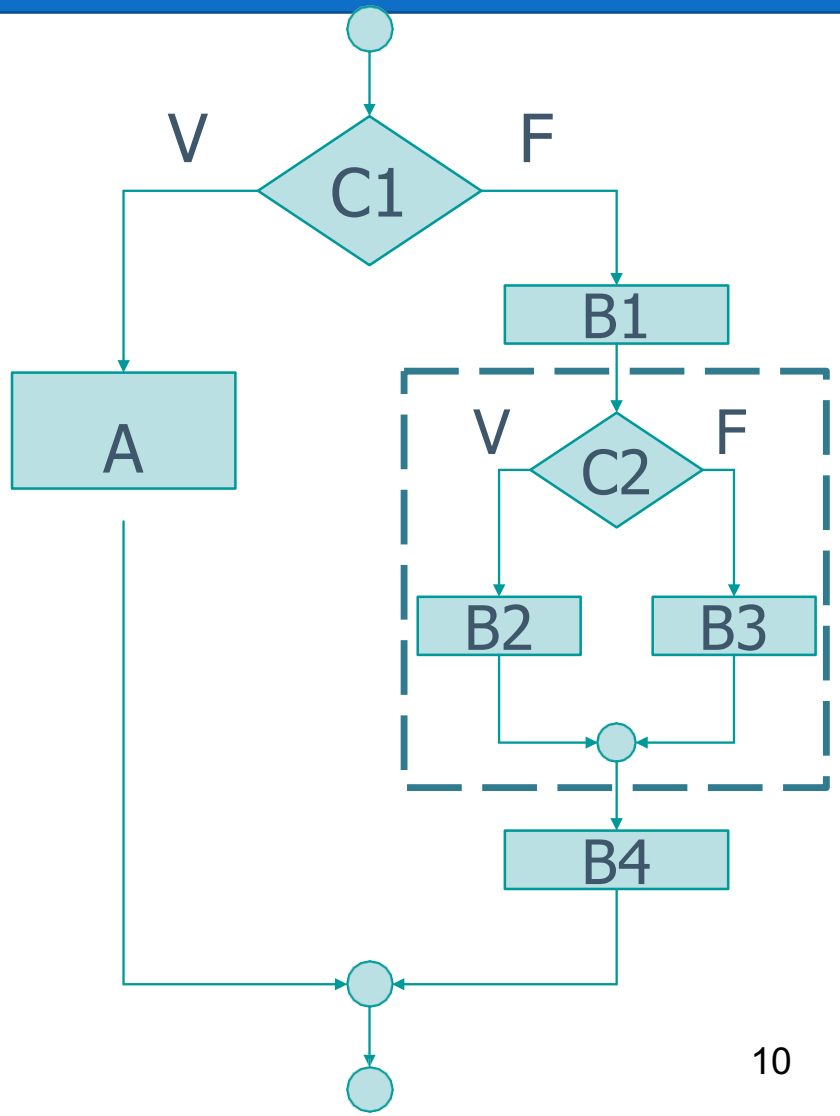
Sintassi C

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

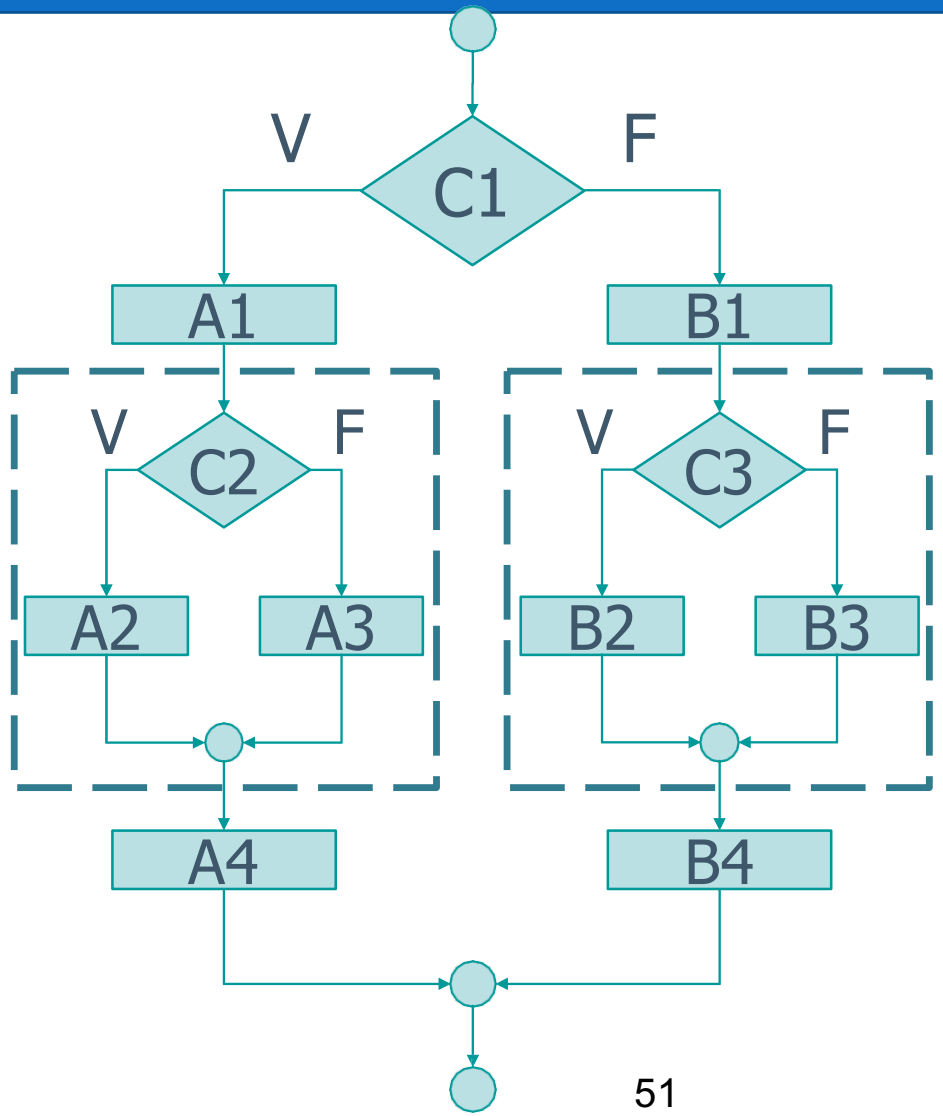


Sintassi C

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



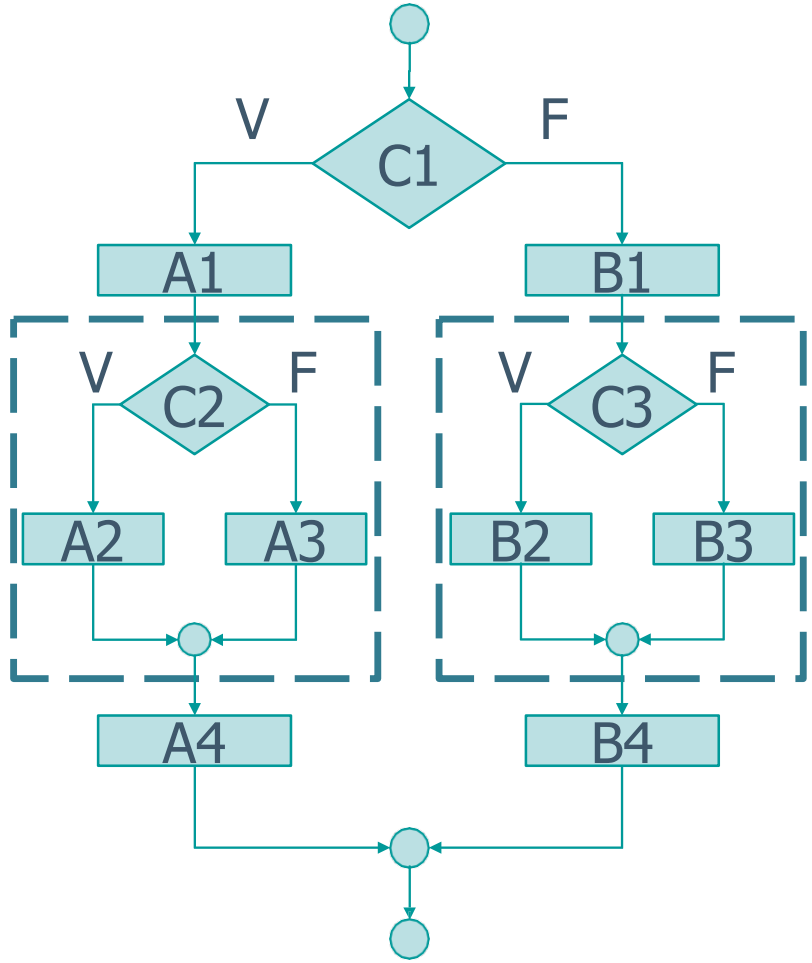
Caso generale



Sintassi C

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

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



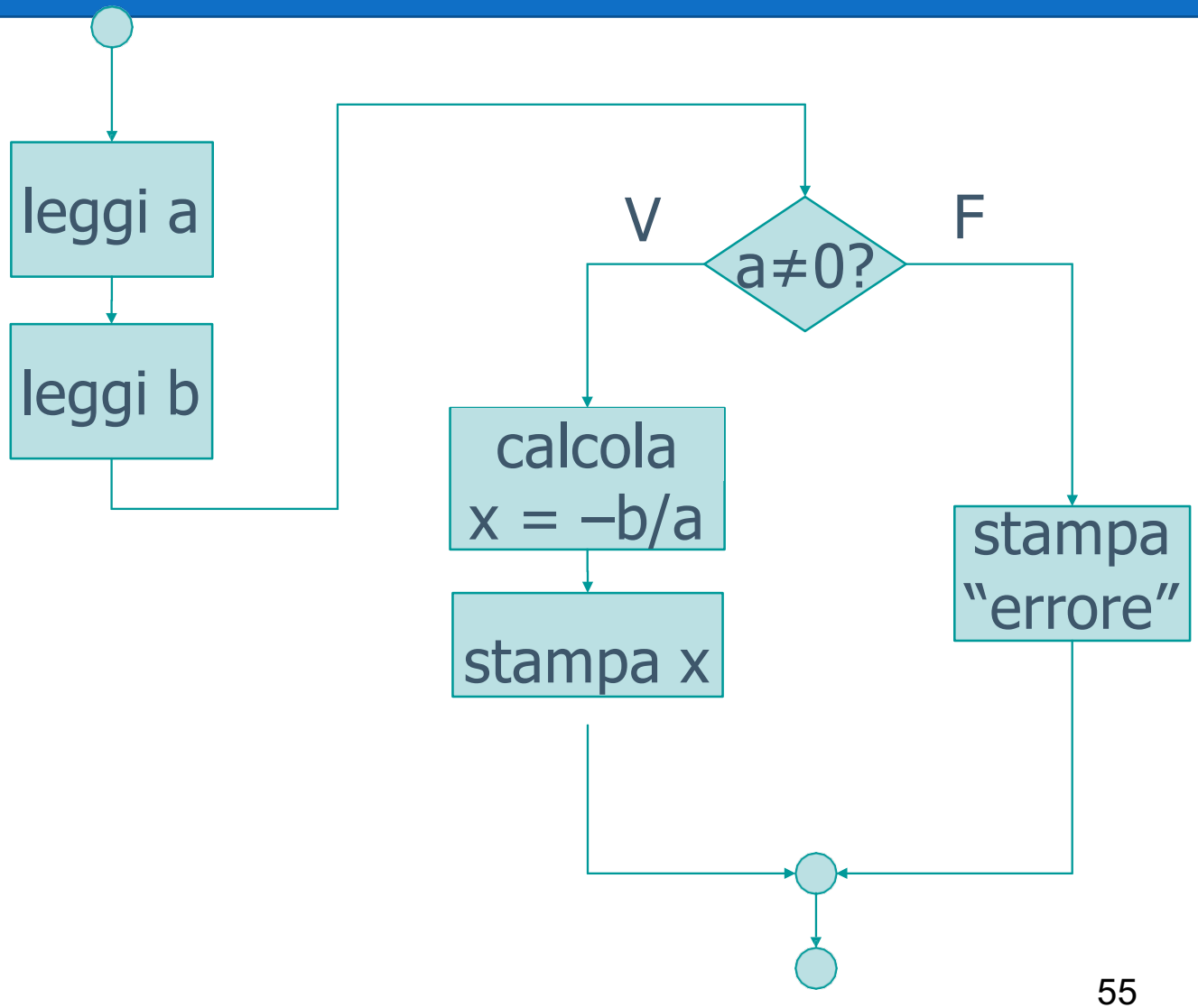
Sintassi C

- 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 $a \neq 0$

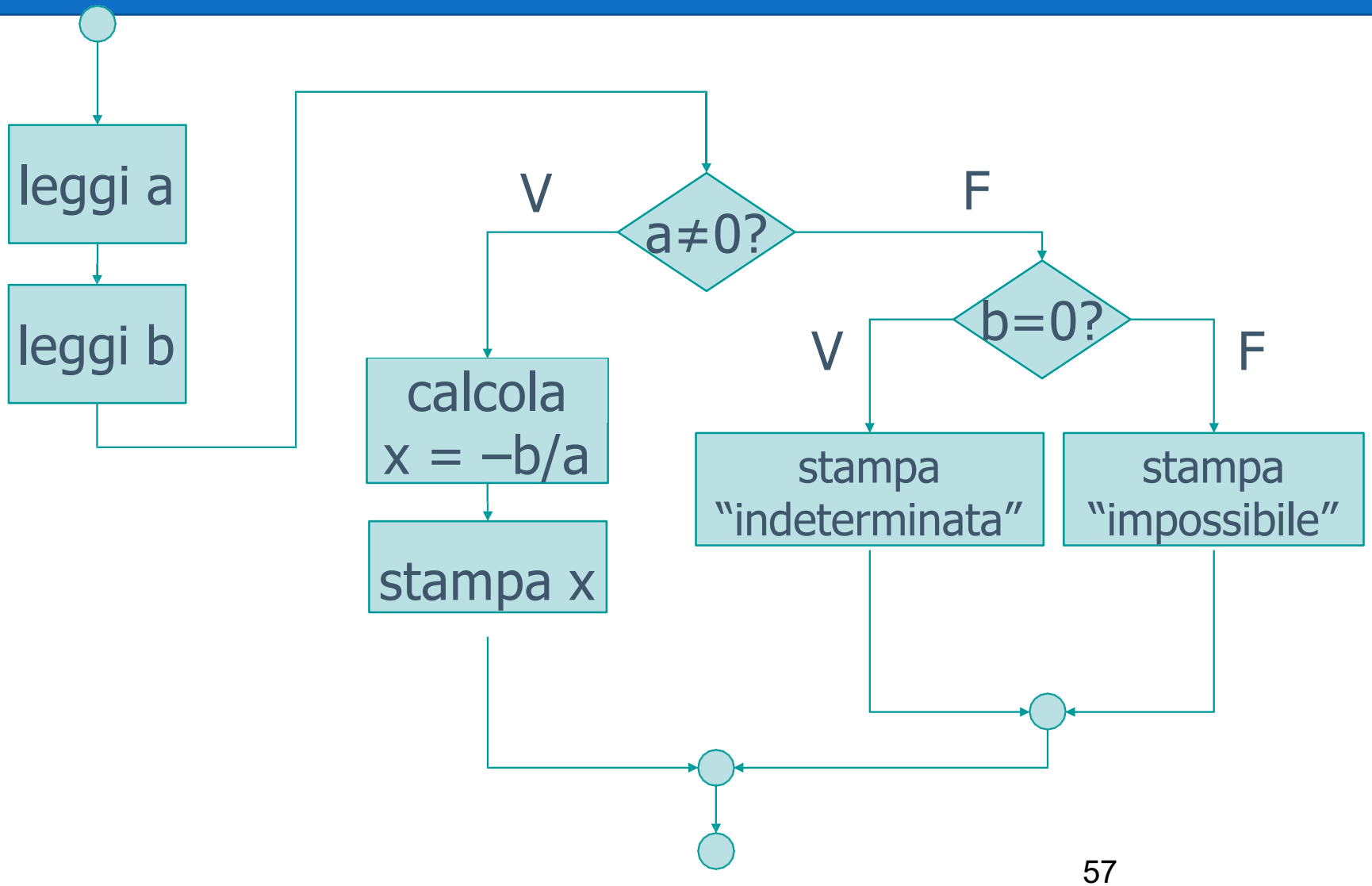
Soluzione (parziale)



Esempio

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

Soluzione (completa)



Soluzione in C (1/2)


```
#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");
        }
    }
}
```