

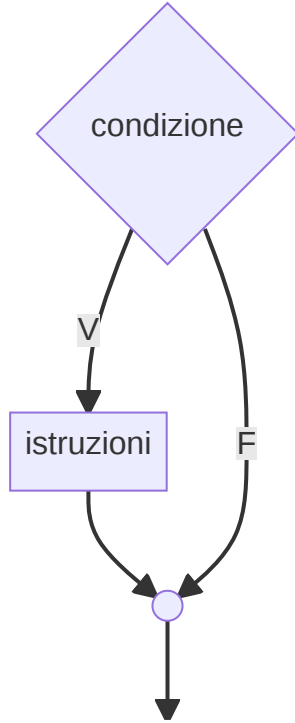
La selezione in C

Sommario

- La selezione semplice
- Operatori relazionali
- La selezione binaria
- Operatori logici
- La selezione nidificata
- L'istruzione switch

Selezione semplice

La selezione semplice



Il calcolatore esegue alcune istruzioni sulla base di una **condizione logica**.

Se la **condizione** risulta **VERA**, vengono eseguite alcune istruzioni.

Se la **condizione** risulta **FALSA**, non viene eseguita alcuna istruzione.

La selezione semplice in C

```
1  if(condizione)
2  {
3      istruzione 1;
4      istruzione 2;
5      ....
6      istruzione N;
7  }
```

Operatori relazionali


Una condizione è un'espressione del linguaggio rappresentata da due elementi messi a confronto da un operatore relazionale.

Operatore	Esempio	Significato
>	$a > b$	maggiore
<	$a < b$	minore
>=	$a \geq b$	maggiore uguale
<=	$a \leq b$	minore uguale
==	$a == b$	uguale
!=	$a != b$	diverso

Errore tipico

Un errore diffuso è quello di utilizzare l'operatore = (**operatore di assegnazione**) per effettuare un confronto. Occorre invece usare l'**operatore == (operatore di confronto)**.

```
1  ...
2  if(a = b)
3  {
4      // do something
5  }
```

 Sbagliato

```
1  ...
2  if(a == b)
3  {
4      // do something
5  }
```

 Corretto

Valore assoluto

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int num;
6
7      printf("Inserire un numero\n");
8      scanf("%d", &num);
9      if(num < 0)
10     {
11         num = num * -1;
12     }
13     printf("Valore assoluto del numero: %d \n", num);
14
15     return 0;
16 }
```

num = -3

condizione **VERA**

Valore assoluto

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int num;
6
7      printf("Inserire un numero\n");
8      scanf("%d", &num);
9      if(num < 0)
10     {
11         num = num * -1;
12     }
13     printf("Valore assoluto del numero: %d \n", num);
14
15     return 0;
16 }
```

num = 5

condizione **FALSA**

Calcolo budget

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      float scarpe, maglietta, pantaloni;
6      float budget;
7
8      printf("Inserire prezzo scarpe:\n");
9      scanf("%f", &scarpe);
10     printf("Inserire prezzo maglietta:\n");
11     scanf("%f", &maglietta);
12     printf("Inserire prezzo pantaloni:\n");
13     scanf("%f", &pantaloni);
14
15     budget = scarpe + maglietta + pantaloni;
16
17     if(budget > 200)
18         printf("Budget superato!\n");
19
20     printf("Totale: %f", budget);
21
22     return 0;
23 }
```

✓ Se in un ramo è presente una singola istruzione, è possibile omettere le parentesi graffe.

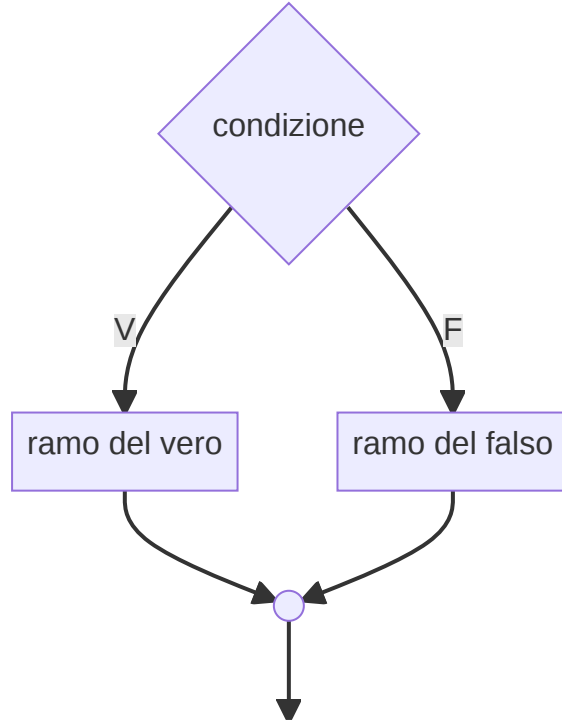
! Attenzione! Occorre indentare il codice per una maggiore leggibilità

Ordinamento di tre variabili

```
1
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int num1, num2, num3, min, med, max, temp;
7
8      printf("Inserire num1:\n");
9      scanf("%d", &num1);
10     printf("Inserire num2:\n");
11     scanf("%d", &num2);
12     printf("Inserire num3:\n");
13     scanf("%d", &num3);
14
15     max = num1;
16     min = num2;
17     med = num3;
18
19     if(min > max)
20     {
21         temp = min;
22         min = max;
```

Selezione binaria

La selezione binaria



Il calcolatore esegue **alternativamente** alcune istruzioni sulla base di una **condizione logica**.

Se la **condizione** risulta **VERA**, vengono eseguite le istruzioni del ramo del vero.

Se la **condizione** risulta **FALSA**, vengono eseguite le istruzioni del ramo del falso.

La selezione binaria in C

```
1  if(condizione)
2  {
3      istruzione x;
4      istruzione y;
5      ....
6      istruzione z;
7  }
8  else
9  {
10     istruzione j;
11     istruzione k;
12     ....
13     istruzione m;
14 }
```

Numero positivo o negativo

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int num;
6
7      printf("Inserire numero:\n");
8      scanf("%d", &num);
9
10     if(num > 0)
11     {
12         printf("Numero positivo!\n");
13     }
14     else
15     {
16         printf("Numero negativo!\n");
17     }
18
19     return 0;
20 }
```

num = 5

condizione **VERA**

Numero positivo o negativo

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int num;
6
7      printf("Inserire numero:\n");
8      scanf("%d", &num);
9
10     if(num > 0)
11     {
12         printf("Numero positivo!\n");
13     }
14     else
15     {
16         printf("Numero negativo!\n");
17     }
18
19     return 0;
20 }
```

num = -3

condizione **FALSA**

Pari o dispari

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int num;
6
7      printf("Inserire numero:\n");
8      scanf("%d", &num);
9
10     if(num % 2 == 0)
11         printf("Numero pari!\n");
12     else
13         printf("Numero dispari!\n");
14
15     return 0;
16 }
```

✓ Se in un ramo è presente una singola istruzione, è possibile omettere le parentesi graffe.

! Attenzione! Occorre indentare il codice per una maggiore leggibilità

Ordinare due numeri (usando min e max)

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      float a, b, min, max;
6
7      printf("Inserire a:\n");
8      scanf("%f", &a);
9      printf("Inserire b:\n");
10     scanf("%f", &b);
11
12     if (a > b)
13     {
14         max = a;
15         min = b;
16     }
17     else
18     {
19         max = b;
20         min = a;
21     }
22
```

Operatori logici

Fino ad ora sono state trattate condizioni logiche semplici.

Una condizione logica composta è un insieme di condizioni logiche semplici legati da **operatori logici**.

Esempio:

`(x >= 3) && (y <= 5)`

condizione composta

operatore logico

condizione A

condizione B

Operatori logici

Operatore	Esempio	Significato
&&	a && b	AND (Prodotto logico)
 	a b	OR (Somma logica)
!	!a	NOT (Negazione)

L'operatore NOT (!) in realtà non mette in relazione due o più condizioni semplici.

AND (Prodotto logico)

L'espressione logica $A \ \&\& \ B$ è **VERA** se e solo se entrambe le condizioni A e B sono VERE.

A	B	A && B
F	F	F
F	V	F
V	F	F
V	V	V

AND (Prodotto logico)

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int eta;
6      printf("Inserire età:\n");
7      scanf("%d", &eta);
8
9      if ((eta > 2) && (eta < 90))
10         printf("Costo ingresso: 8 euro.\n");
11     else
12         printf("Ingresso gratuito\n");
13
14     return 0;
15 }
```

Il ramo del VERO sarà eseguito se il valore della variabile `eta` soddisferà contemporaneamente entrambe le condizioni.

OR (Somma logica)

L'espressione logica $A \parallel B$ è **VERA se e solo se almeno una delle condizioni A oppure B sono VERE**.

A	B	$A \parallel B$
F	F	F
F	V	V
V	F	V
V	V	V

OR (Somma logica)

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int eta;
6      printf("Inserire età:\n");
7      scanf("%d", &eta);
8
9      if ((eta < 2) || (eta > 90))
10         printf("Ingresso gratuito\n");
11     else
12         printf("Costo ingresso: 8 euro.\n");
13
14     return 0;
15 }
```

Il ramo del VERO sarà eseguito se il valore della variabile `eta` soddisferà almeno una delle condizioni.

Vero e Falso in C

Nei linguaggi moderni, oltre ai tipi di dato `int`, `float`, `char`, ... , esiste anche il tipo `bool`.

Il tipo di dato `bool` consente di creare variabili in grado di assumere solamente due valori: VERO o FALSO.

Nel linguaggio C non esiste il tipo `bool`, ma è comunque possibile esprimere il concetto di VERO o FALSO.

Infatti, una condizione è:

- VERA (`true`) se assume valore diverso da 0
- FALSA (`false`) se assume valore uguale a 0

Vero e Falso in C

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int condizione = 1;
6
7      if(condizione)
8          printf("ramo del vero\n");
9      else
10         printf("ramo del falso\n");
11
12     return 0;
13 }
```

Siccome la variabile `condizione = 1`, il suo valore sarà VERO.

Vero e Falso in C

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int condizione = 0;
6
7      if(condizione)
8          printf("ramo del vero\n");
9      else
10         printf("ramo del falso\n");
11
12     return 0;
13 }
```

Siccome la variabile `condizione = 0`, il suo valore sarà FALSO.

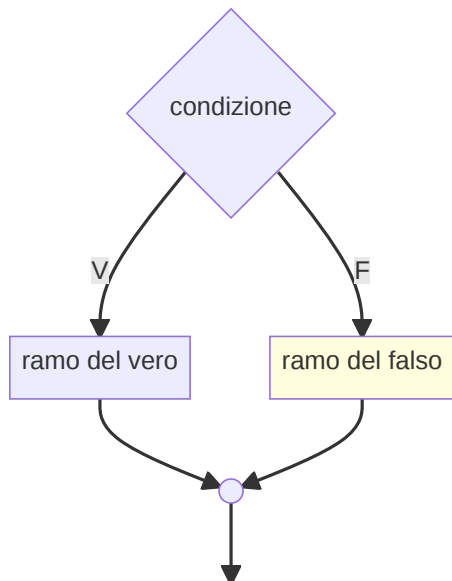
NOT (Negazione)

L'espressione logica !A esegue il cambiamento del valore di verità assunto dalla condizione: **se l'enunciato era VERO, negandolo diventa FALSO e viceversa.**

A	!A
F	V
V	F

Selezione nidificata

La selezione nidificata



I rami della selezione possono contenere qualsiasi tipo di codice:

```
1  if(condizione)
2  {
3      ....
4  }
5  else
6  {
7      ...
8  }
```

La selezione nidificata

Il ramo del falso contiene un'altra selezione binaria.

Questa situazione si chiama selezione nidificata.

La selezione nidificata in C

```
1  if(condizione)
2  {
3      istruzione x;
4      istruzione y;
5      istruzione z;
6  }
7  else
8  {
9      if(condizione)
10     {
11         istruzione j;
12         istruzione k;
13         istruzione l;
14     }
15     else
16     {
17         istruzione m;
18         istruzione n;
19         istruzione o;
20     }
21 }
```


Maggiore, minore, uguale

Verificare se due numeri sono uguali, maggiore di..., minore di...

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int num1, num2;
6
7      printf("Inserire num1:\n");
8      scanf("%d", &num1);
9      printf("Inserire num2:\n");
10     scanf("%d", &num2);
11
12     if(num1 == num2)
13     {
14         printf("I due numeri sono uguali\n");
15     }
16     else
17     {
18         if(num1 > num2)
19         {
20             printf("num1 > num2\n");
21         }
22         else
```

Temperature

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int t;
6
7      printf("Inserire temperatura:\n");
8      scanf("%d", &t);
9
10     if(t > 30)
11     {
12         printf("molto caldo!\n");
13     }
14     else
15     {
16         if(t > 20)
17         {
18             printf("caldo!\n");
19         }
20         else
21         {
22             if(t > 10)
```

Temperatura	Messaggio
$t > 30$	molto caldo
$20 < t < 30$	caldo
$10 < t < 20$	ideale
$0 < t < 10$	freddo

Temperature (else - if)

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int t;
6
7      printf("Inserire temperatura:\n");
8      scanf("%d", &t);
9
10     if(t > 30)
11     {
12         printf("molto caldo!\n");
13     }
14     else if(t > 20)
15     {
16         printf("caldo!\n");
17     }
18     else if(t > 10)
19     {
20         printf("ideale\n");
21     }
22     else
```

Temperatura	Messaggio
$t > 30$	molto caldo
$20 < t < 30$	caldo
$10 < t < 20$	ideale
$0 < t < 10$	freddo

Equazione di II grado

Un'equazione di secondo grado ha la seguente forma:

$$ax^2 + bx + c = 0$$

le cui radici sono:

$$x_1, x_2 = \frac{-b \pm \sqrt{\Delta}}{2a} \quad \text{con} \quad \Delta = b^2 - 4ac$$

impossibile

$$\Delta < 0$$

radici coincidenti

$$\Delta = 0$$

radici distinte

$$\Delta > 0$$

Equazione di II grado

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main(void)
5  {
6      float a, b, c, delta, x1, x2;
7
8      printf("Equazione di II grado\n");
9      printf("Inserire a:\n");
10     scanf("%f", &a);
11     printf("Inserire b:\n");
12     scanf("%f", &b);
13     printf("Inserire c:\n");
14     scanf("%f", &c);
15
16     if(a == 0)
17         printf("Equazione di primo grado. Formula inutilizzabile\n");
18     else
19     {
20         delta = b*b - 4*a*c;
21
22         if(delta < 0)
```

Calcolo di min, med e max

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int n1, n2, n3, min, med, max;
6
7      printf("Inserire n1:\n");
8      scanf("%d", &n1);
9      printf("Inserire n2:\n");
10     scanf("%d", &n2);
11     printf("Inserire n3:\n");
12     scanf("%d", &n3);
13
14     if(n1 > n2)
15     {
16         max = n1;
17         min = n2;
18     }
19     else
20     {
21         max = n2;
22         min = n1;
```

Dandlig else

Analizziamo il seguente codice:

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int x = 11;
6      int y = 9;
7
8      if(x<10)
9          if(y>10)
10             printf("*****");
11     else
12         printf("####");
13         printf("$$$$");
14
15     return 0;
16 }
```

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int x = 11;
6      int y = 9;
7
8      if(x<10)
9          if(y>10)
10             printf("*****");
11     else
12         printf("####");
13         printf("$$$$");
14
15     return 0;
16 }
```

Dandlig else

Esistono situazioni ambigue, in cui si associa erroneamente l'istruzione **else** ad un **if** "sbagliato".

Questa situazione è detta "dandling else".

Per risolvere questo problema, è sufficiente indicare la coppia di parentesi graffe anche in caso di istruzione singola.

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int x = 11;
6      int y = 9;
7
8      if(x<10)
9          if(y>10)
10             printf("*****");
11     else
12         printf("#####");
13         printf("$$$$$");
14
15     return 0;
16 }
```

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int x = 11;
6      int y = 9;
7
8      if(x<10)
9          if(y>10)
10             printf("*****");
11     else
12         printf("#####");
13         printf("$$$$$");
14
15     return 0;
16 }
```


L'istruzione switch

Consente di effettuare una selezione multipla.

Viene valutato il valore di un'espressione e, in base a tale valore, si eseguono blocchi diversi di istruzioni.

```
1  switch(espressione)
2  {
3      case valore1:
4          istruzioni1;
5          break;
6
7      case valore2:
8          istruzioni2;
9          break;
10     ...
11
12     case valoreN:
13         istruzioniN;
14         break;
15
16     default:
17         istruzioni_default;
18         break;
19 }
```

- **espressione** può assumere solamente valori interi
- viene confrontata con ciò che compare dopo **case**.
- viene seguito il blocco in cui viene trovata la corrispondenza
- se non viene trovata alcuna corrispondenza, viene eseguito il blocco di **default**
- **break** serve per terminare un blocco di istruzioni

"Come quando fuori piove"

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int seme;
6
7      printf("Inserire seme\n");
8      scanf("%d", &seme);
9
10     switch(seme)
11     {
12         case 1:
13             printf("picche");
14             break;
15
16         case 2:
17             printf("fiori");
18             break;
19
20         case 3:
21             printf("cuori");
22             break;
```

Condizioni composte

Normalmente occorre sempre l'istruzione di **break** per terminare un blocco di istruzioni.

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      int mese, giorni;
6
7      printf("Inserire mese\n");
8      scanf("%d", &mese);
9
10     switch(mese)
11     {
12         case 1:
13         case 3:
14         case 5:
15         case 7:
16         case 8:
17         case 10:
18         case 12:
19             giorni = 31;
20             break;
21
22         case 4:
```

Per esprimere **condizioni composte**, è possibile utilizzare una sequenza di **case** senza **break**.