ROMA TRE
UNIVERSITÀ DEGLI STUDI

Dipartimento di Ingegneria
Industriale, Elettronica e Meccanica

Collegio Didattico di Ingegneria Meccanica

Elementi di Informatica

Istruzioni condizionali in C

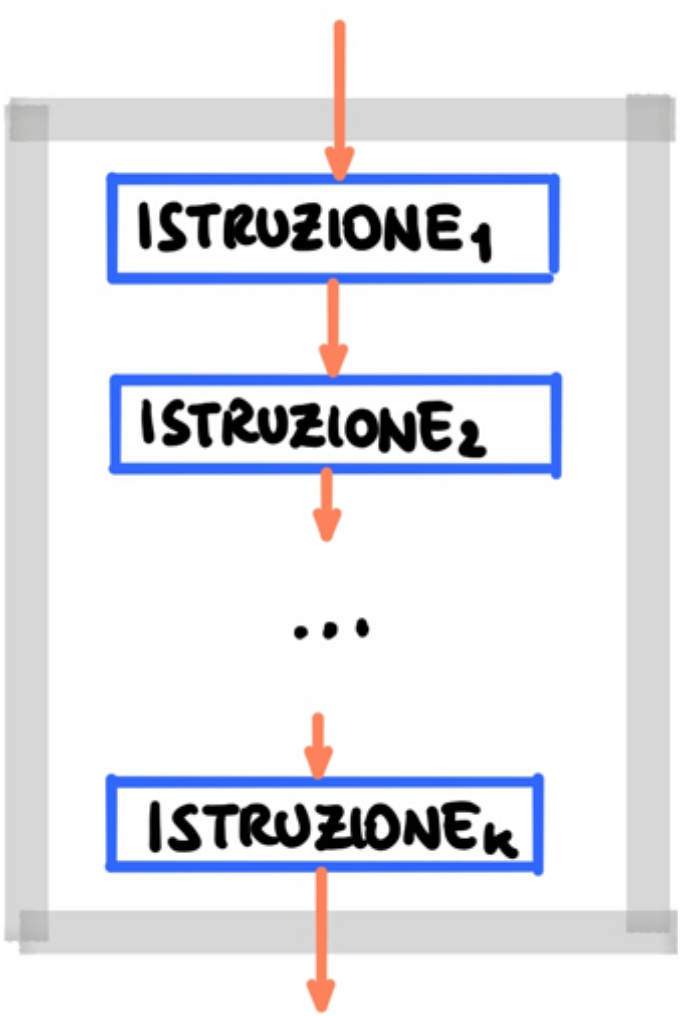
Giordano Da Lozzo e Giuseppe Sansonetti

Previously… Istruzioni di controllo

- Istruzioni che permettono di controllare (nel senso di regolare, governare, dirigere) il flusso di esecuzione, ovvero quali istruzioni devono essere eseguite e in quale ordine.

- Normalmente le istruzioni che compongono un algoritmo vengono eseguite una dopo l'altra, nell'ordine in cui compaiono.

- Le istruzioni di controllo permettono di modificare l'ordine in cui le istruzioni di un algoritmo vengono eseguite.

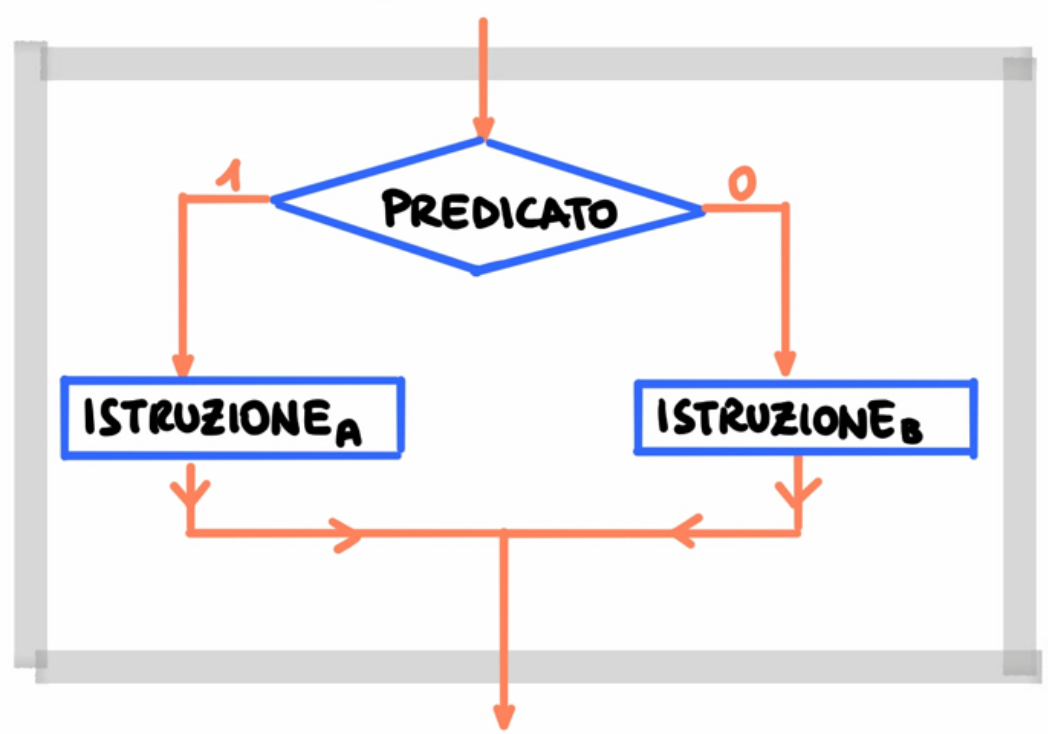


Blocco o sequenza

- Rappresenta: esegui $istruzione_1$, poi esegui $istruzione_2$, … poi esegui $istruzione_k$

⇒Serve quando: hai una sequenza di istruzioni che vuoi eseguire l'una dopo l'altra

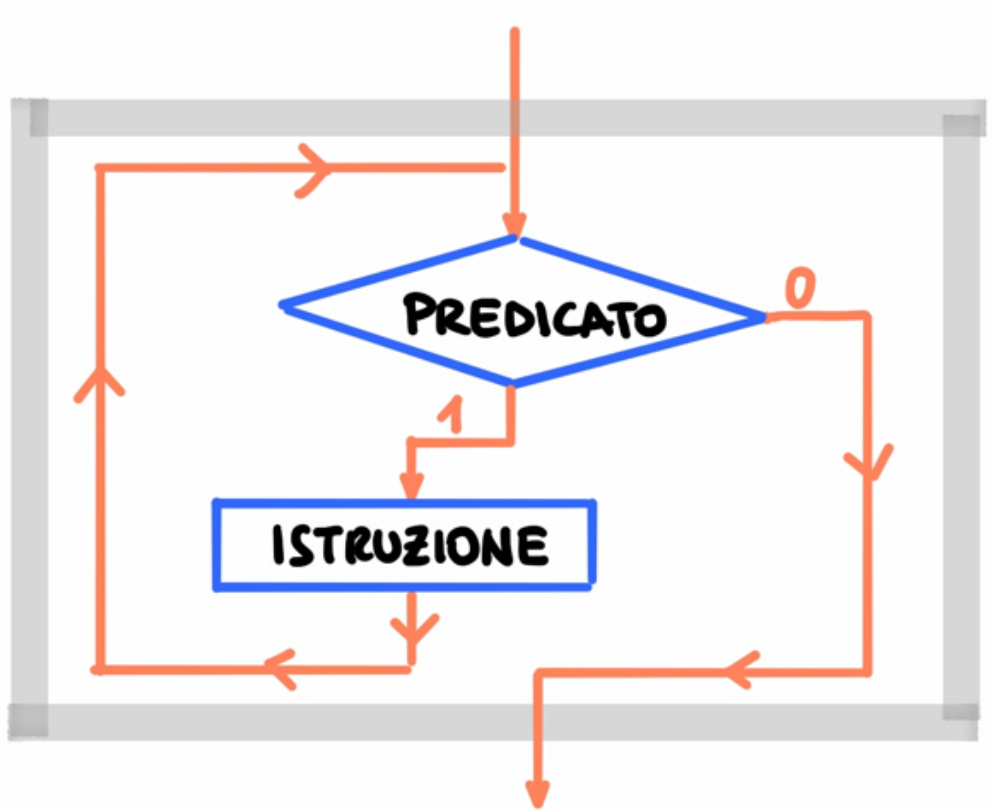
Istruzione condizionale



- Rappresenta: se è vero il predicato, allora esegui $istruzione_a$, altrimenti esegui $istruzione_b$.

⇒ Utile quando: vuoi eseguire una istruzione solo al verificarsi di una condizione, altrimenti ne vuoi eseguire un'altra

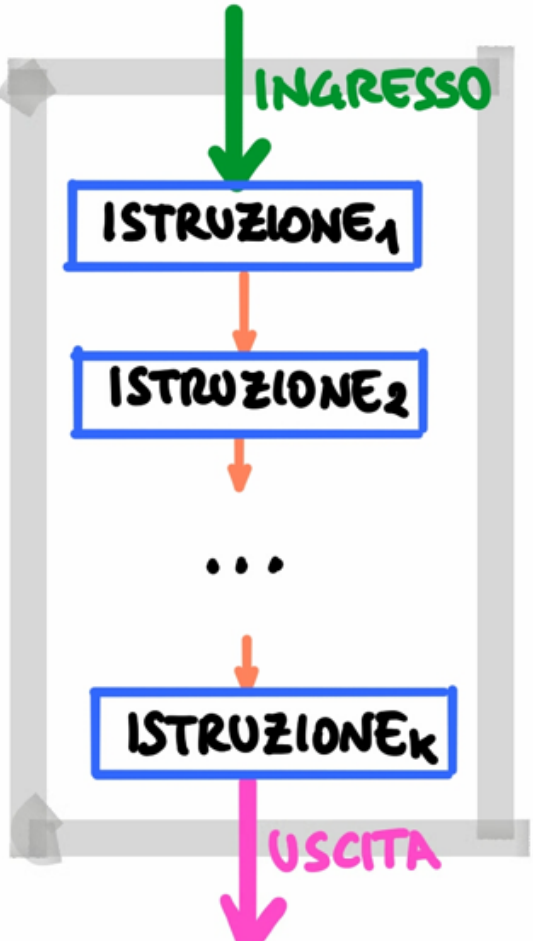
Istruzione ripetitiva



- Rappresenta: fintanto che è vero il predicato, esegui $istruzione$

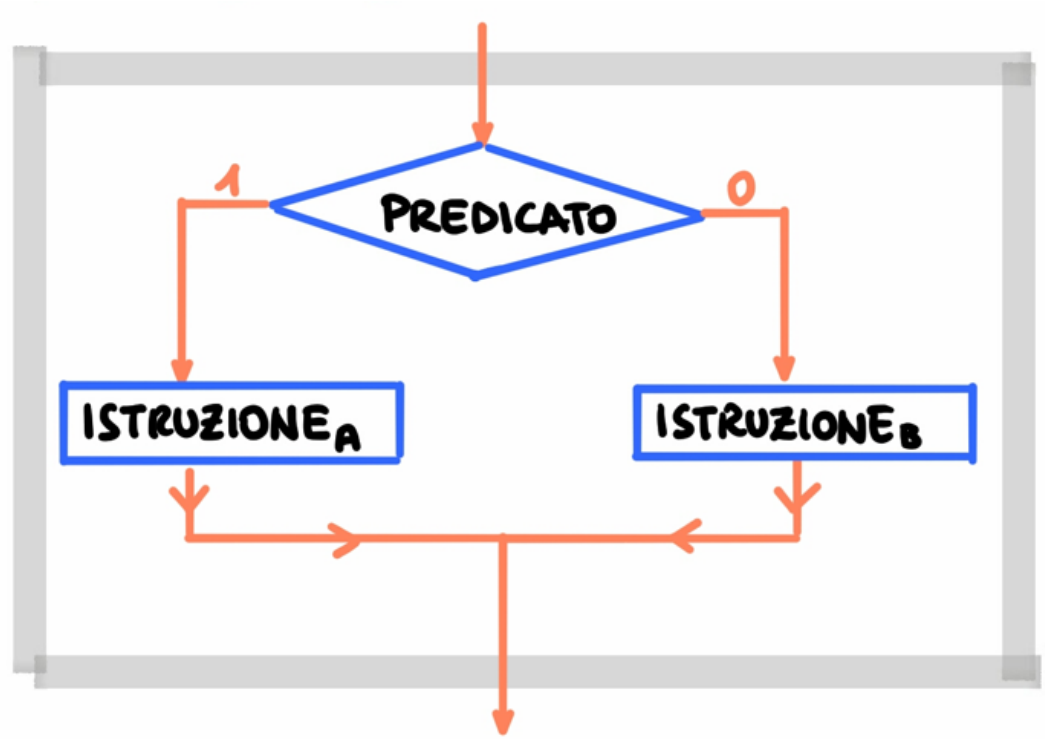
⇒ Utile quando: vuoi eseguire ripetutamente la stessa istruzione fintanto che si verifica una condizione

Punti di ingresso ed uscita



Scopo del giorno

Imparare come tradurre in C le istruzioni condizionali ed i blocchi



$$if\,(predicato)$$
$$\quad istruzione_a$$
$$else$$
$$\quad istruzione_b$$

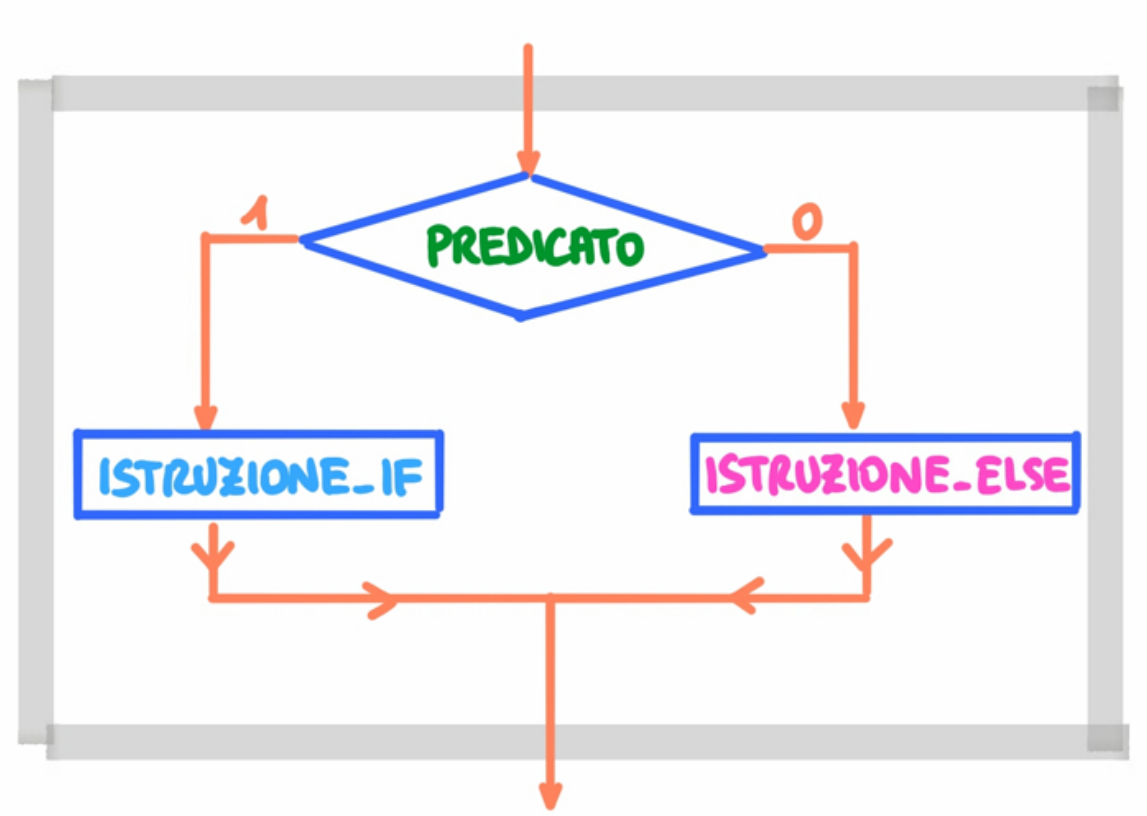
Istruzione condizionale if-else: sintassi

$$
\begin{aligned}
&if\,(predicato)\\
&\quad istruzione_{if}\\
&else\\
&\quad istruzione_{else}
\end{aligned}
$$

Dove $predicato$ rappresenta una condizione logica (ovvero una espressione booleana, che in C può valere 0 o 1), detta condizione dell'istruzione if-else.

Mentre $istruzione_{if}$ è un'istruzione, detta parte if dell'istruzione if-else ed $istruzione_{else}$ è un'istruzione, detta parte else dell'istruzione if-else.

Istruzione condizionale if-else: semantica



Se $predicato$ vale 1, allora esegui $istruzione_{if}$, altrimenti (ovvero se $predicato$ vale 0) esegui $istruzione_{else}$.

Esercizio

Scrivere un programma PositivoONo, che legge un intero introdotto dall'utente e stampa un messaggio che indica se l'intero è positivo oppure no.

Dichiarazione variabili

Input

```
/* programma che legge un intero e stampa un
 * messaggio che indica se l'intero è positivo o no */
int main() {
   int numero; // per memorizzare l'intero
   printf("caro utente introduci un intero!\n");
   scanf("%d", &numero);

   /* controlla se il numero è positivo */
   if(numero>0)
      printf("il numero %c positivo!", 138);
   else
      printf("il numero non %c positivo!", 138);
}
```

Esercizio

Scrivere un programma PariODispari, che legge un intero introdotto dall'utente e stampa un messaggio che indica se l'intero è pari oppure dispari.

Dichiarazione variabili

Input

```c
/* programma che legge un intero e stampa un
 * messaggio che indica se l'intero è pari o dispari*/
int main() {
    int numero; // per memorizzare l'intero
    printf("caro utente introduci un intero!\n");
    scanf("%d", &numero);

    /* controlla se il numero è pari */
    if(numero%2==0)
        printf("Il numero %c pari!", 138);
    else
        printf("Il numero %c dispari!", 138);
}
```

Esercizio

Realizzare un programma Bisestile, che chiede all'utente di inserire un anno e stampa un messaggio che indica all'utente se quell'anno è bisestile oppure no.

- Nota: un anno è bisestile se soddisfa una delle seguenti 2 condizioni:
  - E' divisibile per 4 e non è divisibile per 100
  - E' divisibile per 400
- Esempio: 1900 non è bisestile.
  - E' divisibile per 4 ed è divisibile per 100
  - Non è divisibile per 400

Bisestile

```c
/* programma che chiede all'utente di inserire un anno da tastiera, lo legge e stampa
* un messaggio che dice all'utente se l'anno è bisestile oppure no*/
int main() {
   int anno; // anno da leggere

   /* INPUT */
   printf("caro utente introduci un anno\n");
   scanf("%d", &anno);

   /* OUTPUT */
   if((anno%4==0) && (anno%100!=0) || (anno%400==0)) // anno bisestile
      printf("Anno bisestile\n");
   else
      printf("Anno non bisestile\n");
}
```

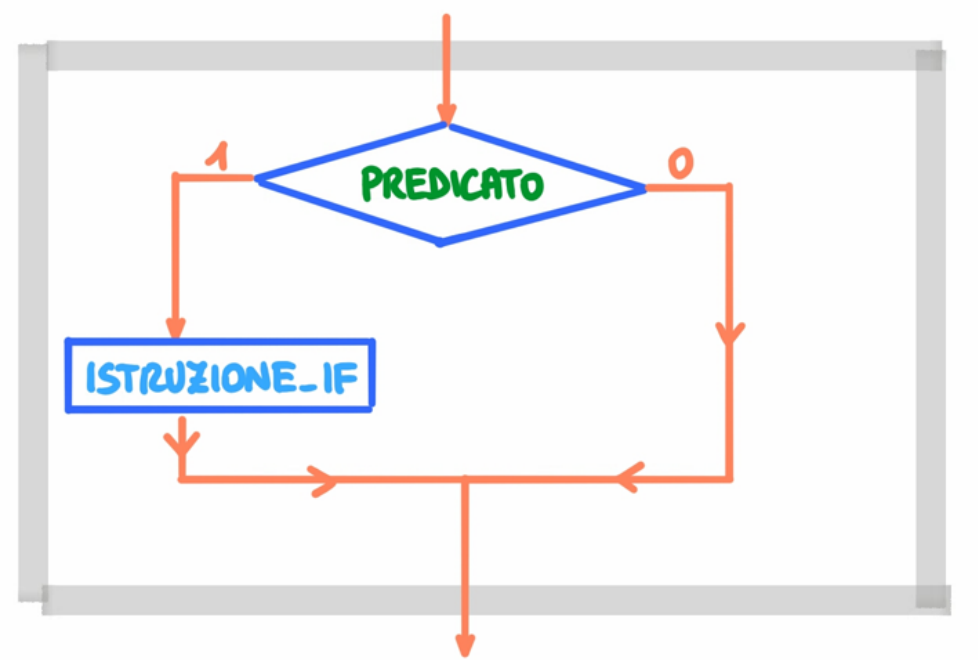
Istruzione condizionale if: sintassi

$$if\,(predicato)$$
$$\quad istruzione_{if}$$

Dove $predicato$ rappresenta una condizione logica (ovvero una espressione booleana, che in C può valere 0 o 1), detta condizione dell'istruzione if

Mentre $istruzione_{if}$ è un'istruzione, detta parte if dell'istruzione if.

Istruzione condizionale if: semantica



Se $predicato$ vale 1, allora esegui $istruzione_{if}$.

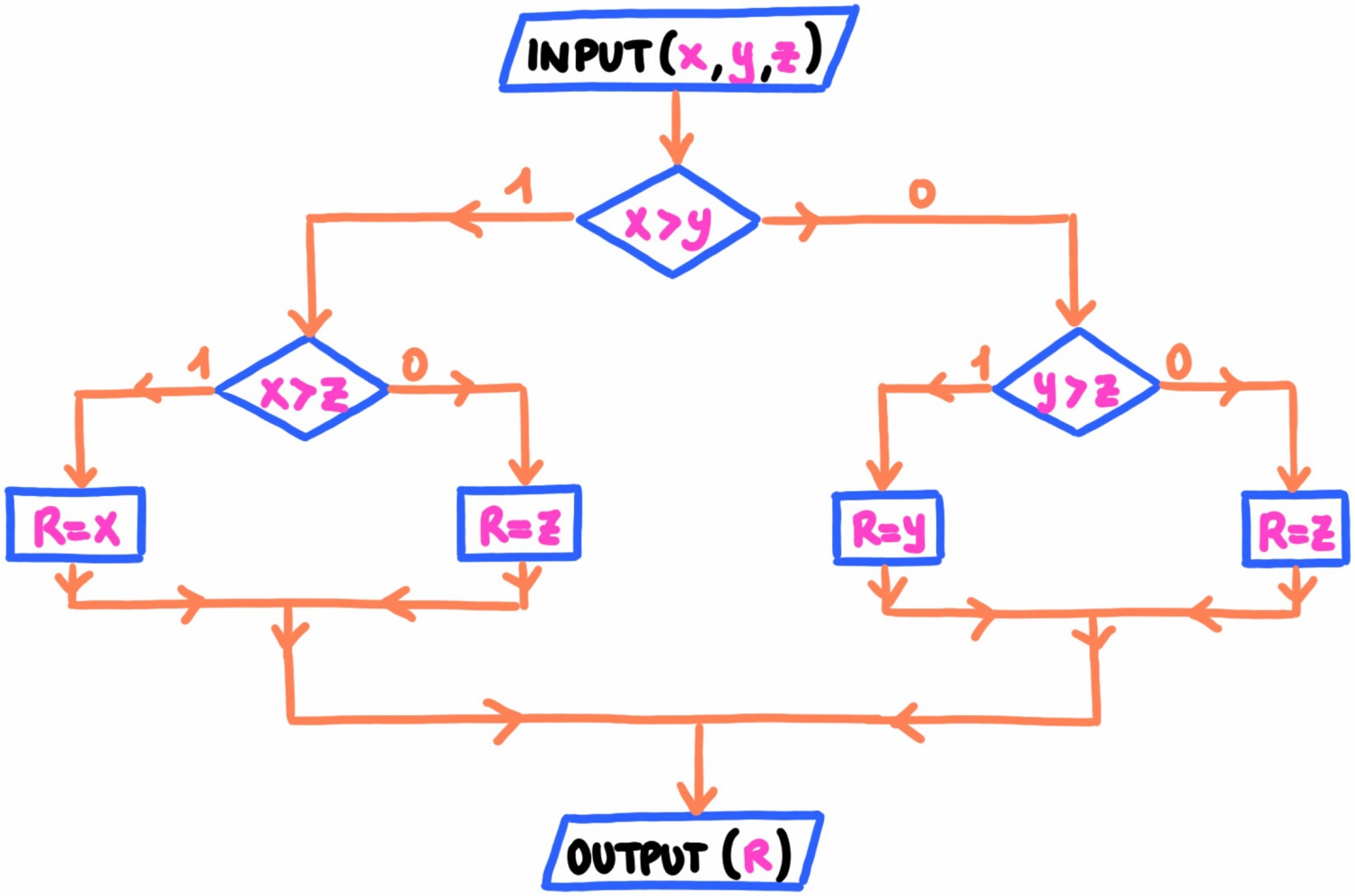
Massimo fra 3 interi

Scrivere un’applicazione Massimo3Interi, che chiede all’utente di inserire 3 interi da tastiera, legge gli interi inseriti dall’utente e stampa un messaggio che informa l’utente del valore massimo fra i tre interi.

Massimo fra 3 interi: pseudo-codice Alg. 1

1. $Input(x, y, z)$
2. $Se(x > y)$
   - 2.1 $Se\ (x > z)$
     - 2.1.1 $r = x \longleftarrow x > y\ \&\&\ x > z$
     - $Altrimenti$
     - 2.1.2 $r = z \longleftarrow x > y\ \&\&\ x \leq z$
   - $Altrimenti$
   - 2.2 $Se(y > z)$
     - 2.2.1 $r = y \longleftarrow x \leq y\ \&\&\ y > z$
     - $Altrimenti$
     - 2.2.2 $r = z \longleftarrow x \leq y\ \&\&\ y \leq z$
3. $Output(r)$

Massimo fra 3 interi: diagramma a blocchi Alg. 1



Massimo fra 3 interi: programma 1

```
int main() {
   int x,y,z; // variabili per memorizzare tre interi
   int massimo; // il massimo fra i tre

   /* INPUT */
   printf("Introduci tre interi!\n");
   scanf("%d%d%d", &x,&y,&z);

   /* CALCOLA IL MASSIMO */
   if(x>y)
      if(x>z)
         massimo=x;
      else
         massimo=z;
   else
      if(y>z)
         massimo=y;
      else
         massimo=z;
    /* OUTPUT */
   printf("Il massimo valore %c %d", 138, massimo);
}
```

Input

Output

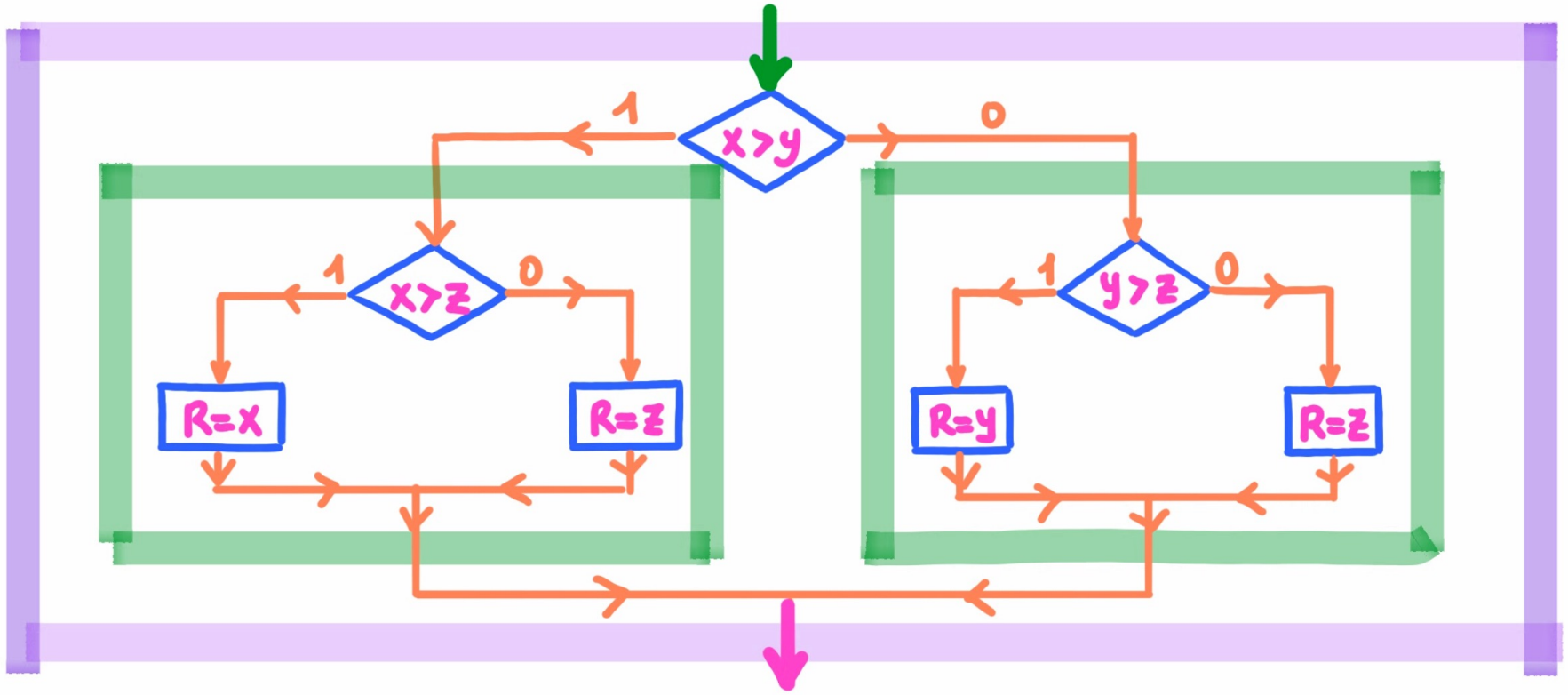
Nella codifica, l'annidamento delle istruzioni si rende con l'**indentazione** del codice

In C, l'indentazione è un solo supporto alla comprensione del codice.

- Non ha impatto sulla semantica delle istruzioni.

Istruzioni strutturate

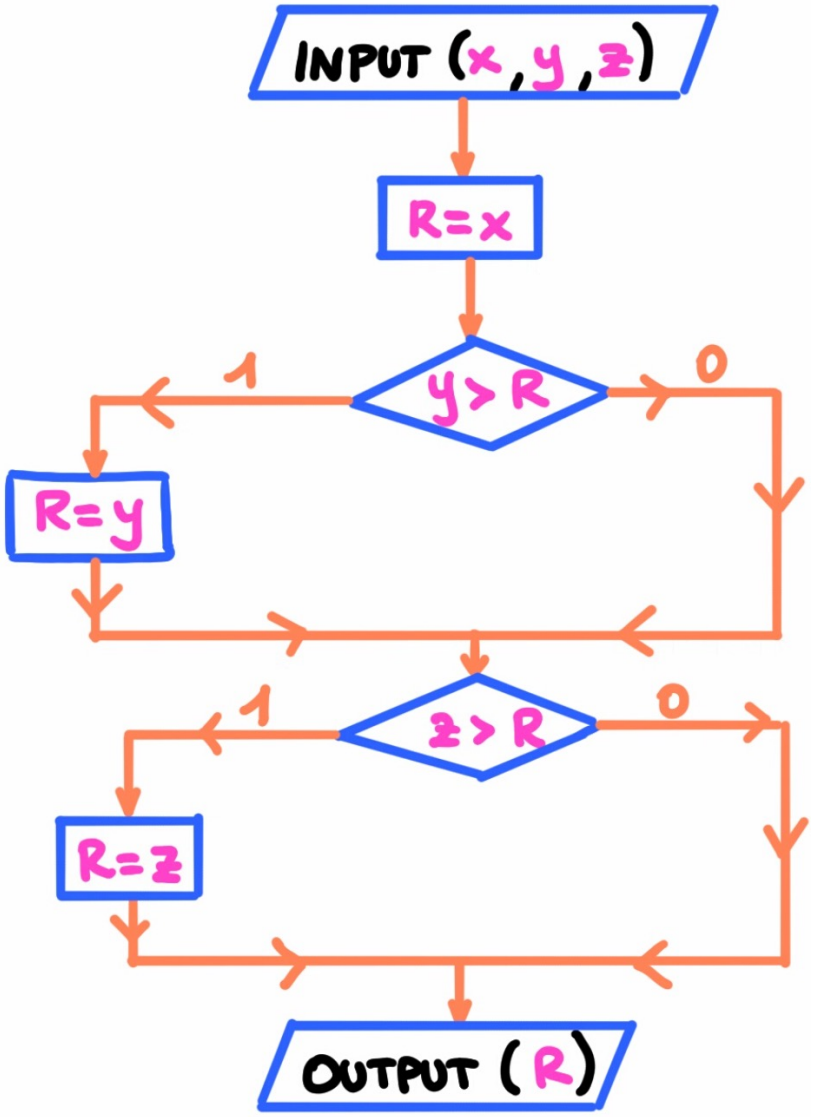
Il fatto che le istruzioni di controllo abbiano un punto di ingresso ed un punto di uscita, permette di vedere ciascuna istruzioni di controllo come una singola istruzione e permette quindi l'annidamento.



Massimo fra 3 interi: pseudo-codice Alg. 2

1. $Input(x, y, z)$
2. $r = x$
3. $Se(y > r)$
   3.1 $r = y$
4. $Se(z > r)$
   4.1 $r = z$
5. $Output(r)$

Massimo fra 3 interi: diagramma a blocchi Alg. 2



Massimo fra 3 interi: programma 2

```
int main() {
   int x,y,z; // variabili per memorizzare tre interi
   int massimo; // il massimo fra i tre

   /* INPUT */
   printf("Introduci tre interi!\n");
   scanf("%d%d%d", &x,&y,&z);

   /* CALCOLA IL MASSIMO */
   massimo=x
   if(y>massimo)
      massimo=y;
   if(z>massimo)
      massimo=z;

   /* OUTPUT */
   printf("Il massimo valore %c %d", 138, massimo);
}
```

Test 1:
x=10, y=5, z=1

Test 2:
x=-5, y=-8, z=-2

Nota: l'istruzione *printf* viene eseguita indipendentemente dal valore delle condizioni delle istruzioni condizionali (un punto di uscita).

Massimo fra 3 interi: pseudo-codice Alg. 3

1. $Input(x, y, z)$
2. $Se(x > y \;\&\&\; x > z)$
   - 2.1 $r = x$

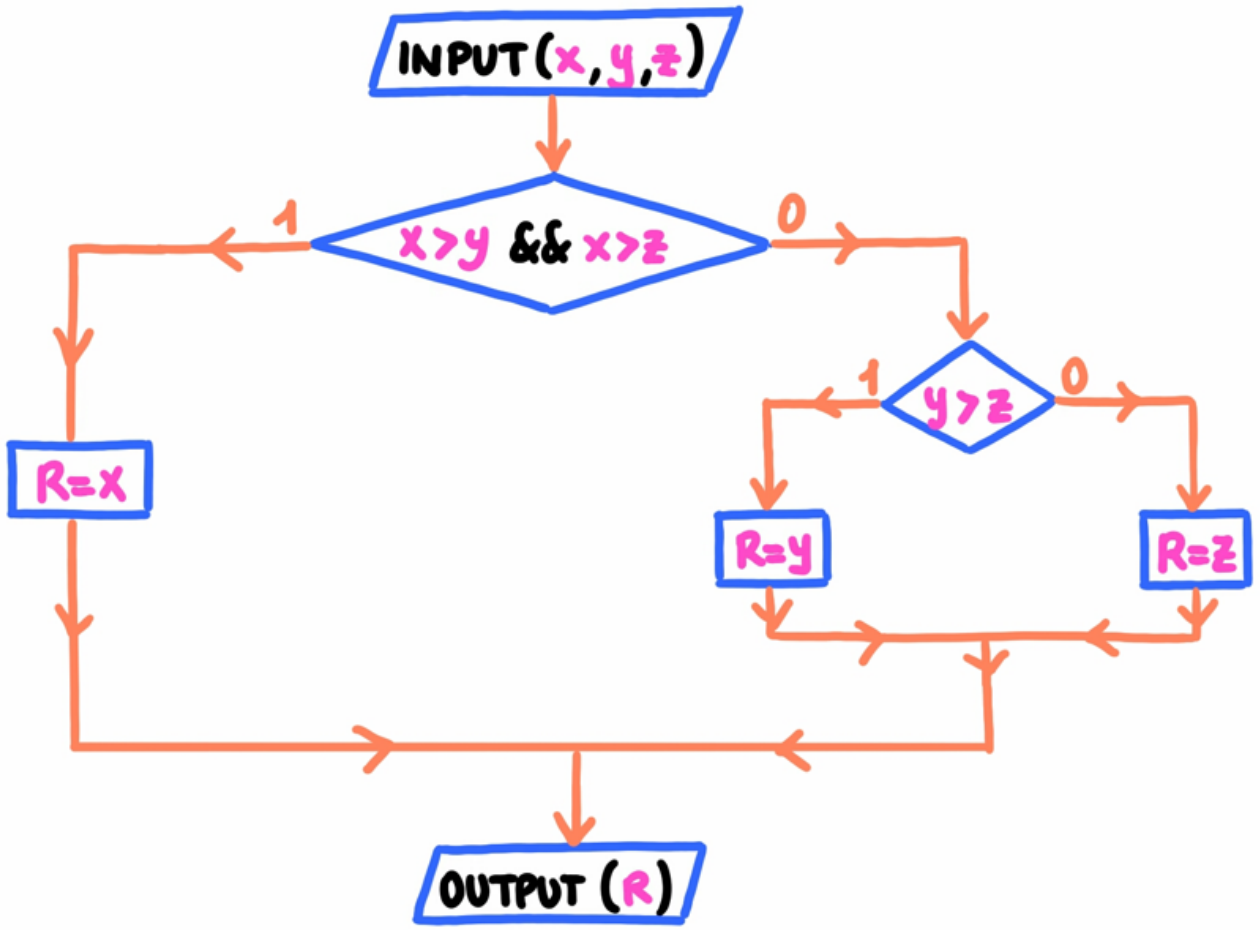
   $Altrimenti$ ← $!(x > y \;\&\&\; x > z) \equiv (x \leq y \;||\; x \leq z)$

   - 2.2 $Se(y > z)$
     - 2.2.1 $r = y$

     $Altrimenti$

     - 2.2.2 $r = z$
3. $Output(r)$

Massimo fra 3 interi: diagramma a blocchi Alg. 3



Massimo fra 3 interi: programma 3

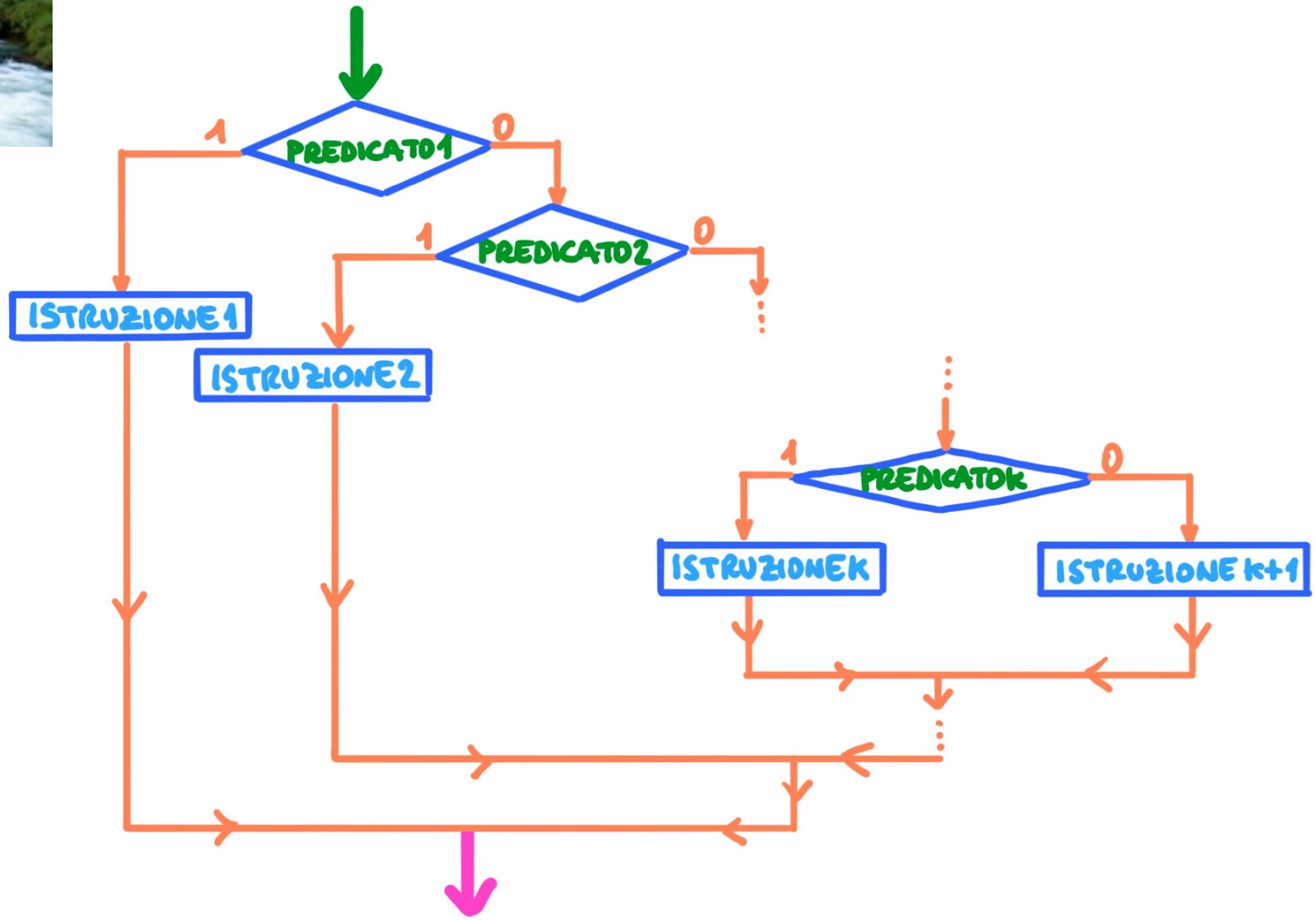
```
int main() {
   int x,y,z; // variabili per memorizzare tre interi
   int massimo; // il massimo fra i tre

   /* INPUT */
   printf("Introduci tre interi!\n");
   scanf("%d%d%d", &x,&y,&z);

   /* CALCOLA IL MASSIMO */
   if(x>y && x>z)
      massimo=x;
   else
      if(y>z)
         massimo=y;
      else
         massimo=z;
   /* OUTPUT */
   printf("Il massimo valore %c %d", 138, massimo);
}
```

Osserva le indentazioni!

Cascata di istruzioni if-else



Cascata di istruzioni if-else

- Composizione di istruzioni if-else, che si realizza inserendo ripetutamente un'istruzione if-else all'interno della parte else di un'istruzione if-else.
- Viene eseguita solo una delle k+1 istruzioni, in base al valore delle k condizioni. In particolare, viene eseguita $istruzione_j$, con $1 \leq j \leq k$, se $predicato_j$ vale 1 mentre $predicato_i$ vale 0, per ogni $1 \leq i \leq j-1$. Viene eseguita $istruzione_{k+1}$ se $predicato_i$ vale 0, per ogni $1 \leq i \leq k$.
- L'istruzione if-else più annidata potrebbe essere un'istruzione if. Allora ci sarebbero k istruzioni a fronte di k condizioni. Se le k condizioni valgono tutte 0, non viene eseguita nessuna istruzione.

If annidati (problema dell'else mancante)

- Mentre annidare un'istruzione if-else all'interno della parte else di un'altra istruzione if-else è pratica comune (cascata di istruzioni if-else), l'annidamento di un'istruzione if-else all'interno della parte if di un'altra istruzione if-else è meno comune, meno intuitivo, e può facilmente portare ad errori ⇒ Fatelo con attenzione!!
- Esempio: con il seguente frammento di codice vorremmo stampare la frase "positivo grande" se x>10, la frase "negativo" se x<0 e non vorremmo stampare niente altrimenti.
- Q: che cosa stampa questo frammento di codice se x=-3? E se x=5?

```
if(x>=0)
        if(x>10)
                printf("positivo grande");
else
        printf("negativo");
```

If annidati (problema dell'else mancante)

- Regola: una parte else appartiene all'istruzione condizionale dell'ultimo if.
  - In particolare, l'indentazione non ha alcun effetto sul programma.

```
if(x>=0)
        if(x>10)
                printf("positivo grande");
else
        printf("negativo");
```

- Q: che cosa stampa questo frammento di codice se x=-3? E se x=5?

niente negativo

- E' possibile modificare il comportamento descritto delimitando esplicitamente la parte if e/o la parte else di un'istruzione condizionale per mezzo di parentesi graffe.

```
if(x>=0) {
        if(x>10)
                printf("positivo grande");
        }
else
        printf("negativo");
```

Maggiore e minore

Con il seguente frammento di codice vorremmo salvare il valore massimo fra quelli memorizzati in a e b nella variabile maggiore, ed il valore minimo nella variabile minore

error: 'else' without a previous 'if'

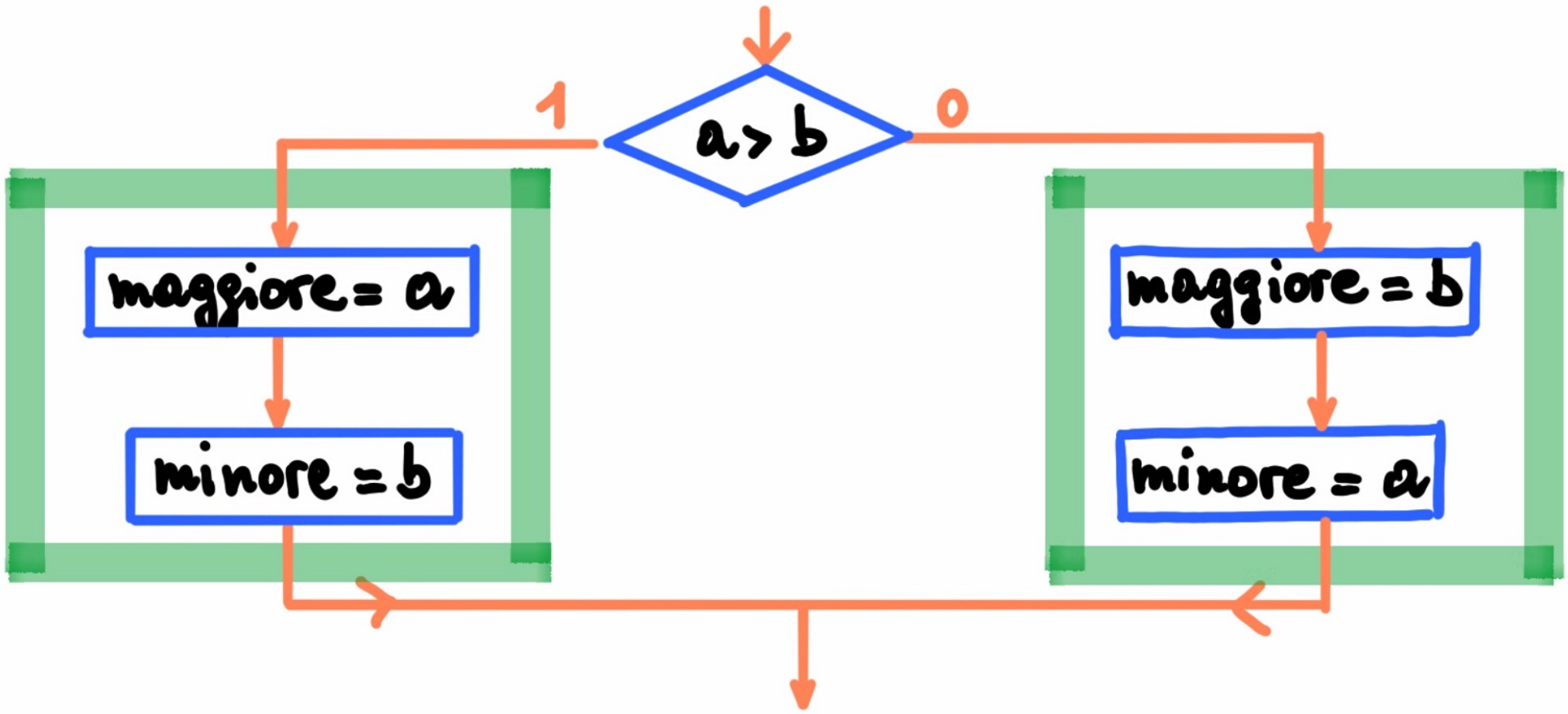
```
if(a>b)
        maggiore=a;
        minore=b;
else
        maggiore=b;
        minore=a;
```

Non fa parte dell'if

La parte if e la parte else devono essere singole istruzioni

Nel codice sopra, maggiore=a; è la parte if dell'istruzione condizionale. Visto che la parola successiva alla parte if non è else, allora l'istruzione condizionale è un'istruzione if e non un'istruzione if-else, quindi il successivo else non ha un if corrispondente.

Il diagramma che avevamo in mente



Blocco

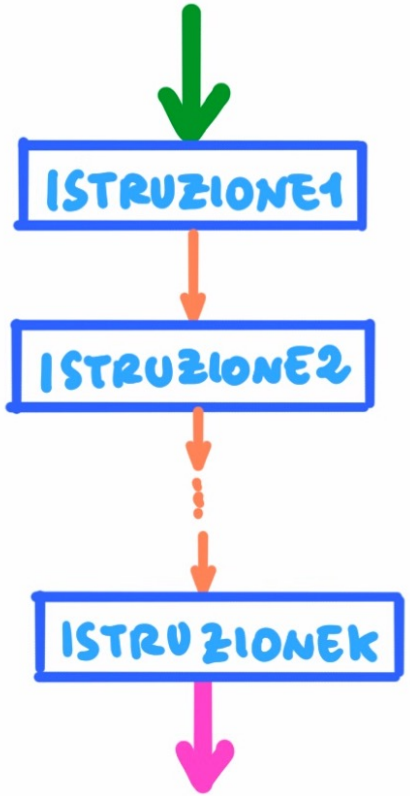
Istruzione strutturata composta da una sequenza di istruzioni

SINTASSI:

```
{
  ISTRUZIONE 1
  ISTRUZIONE 2
      ...
  ISTRUZIONE k
}
```

SEMANTICA:

ESEGUI ISTRUZIONE 1,
POI ISTRUZIONE 2,
...
POI ISTRUZIONE k.



```
if(a>b){
        maggiore=a;
        minore=b;
} else{
        maggiore=b;
        minore=a;
}
```

```
if(a>b){
      maggiore=a;
      minore=b;
} else{
       maggiore=b;
       minore=a;
}
```

Parentesi graffe

Sono sintatticamente necessarie per le istruzioni di blocco.

E' possibile (e consigliato) usare parentesi graffe per delimitare la parte if e/o la parte else di una istruzione condizionale, anche se spesso non è necessario.

warning: suggest explicit braces to avoid abiguous 'else'

```
if(a>0) {
   if(a>100)
      printf("positivo grande");
   else
      printf("positivo piccolo");
}
```

↕ **Equivalenti** ↕

```
if(a>0)
   if(a>100)
      printf("positivo grande");
   else
      printf("positivo piccolo");
```

```
if(a>0) {
   if(a>100)
      printf("positivo grande");
   else
      printf("positivo piccolo");
}
else {
   if(a>-100)
      printf("negativo grande");
   else
      printf("negativo piccolo");
}
```

↔ **Equivalenti** ↔

```
if(a>0)
   if(a>100)
      printf("positivo grande");
   else
      printf("positivo piccolo");
else
   if(a>-100)
      printf("negativo grande");
   else
      printf("negativo piccolo");
```

Osservazioni

- La sintassi delle istruzioni if ed if-else non richiede il *punto e virgola* dopo il predicato. Scrivere if(predicato); fa sì che la parte if sia una istruzione vuota.

- E' possibile dichiarare variabili all'interno di blocchi. In tal caso, le variabili sono visibili (ovvero possono essere utilizzate) solamente all'interno dei blocchi stessi.

```
if(a>b) {
   int x = 5;
   printf("se uso x=%d qui va bene", x);
}
printf("se uso x=%d qui non va bene", x);
```

Error: 'x' undeclared

- I blocchi, le istruzioni condizionali e le istruzioni ripetitive sono delle singole istruzioni. Sono delle istruzioni strutturate, a differenza di istruzioni come printf(…); che sono istruzioni semplici.

Piccolo riepilogo

$if\,(predicato)$
$\quad istruzione_{if}$
$else$
$\quad istruzione_{else}$

$predicato$: espressione/funzione che restituisce un valore booleano
$istruzione_{if}$: istruzione
$istruzione_{else}$: istruzione

Istruzione:
- Istruzione semplice
- Istruzione strutturata

Istruzione semplice:
- Assegnazione
- Incremento o decremento
- Invocazione di funzione
- Istruzione di return
- …

Istruzione strutturata:
- Istruzione blocco
- Istruzione condizionale
- Istruzione ripetitiva

Istruzione blocco: {istruzione$_1$; istruzione$_2$; … ; istruzione$_k$}

Altre risorse

- Bellini, Guidi: Linguaggio C — Capitolo 4.6 (cenni), 4.7, 7.1-7.3