

2.3 if annidati

Il costrutto `if` è un'istruzione e quindi può comparire all'interno di un altro `if`, come si deduce dalla sintassi generale, nel ruolo di *istruzione*. Quando ciò si verifica si parla di `if` annidati ■. Nell'esempio:

```
if (i<100)
    if (i>0)
        printf("minore di 100 e maggiore di zero");
```

il secondo controllo (`i>0`) viene effettuato soltanto se il primo (`i<100`) ha dato esito positivo. Se anche il secondo `if` risulta vero si visualizza il messaggio. Si osservi che, dopo il primo `if`, non è necessario inserire il secondo `if` e la `printf` all'interno di parentesi graffe, in quanto queste costituiscono già un'unica istruzione semplice.

Aggiungiamo ora al nostro esempio un ramo `else`:

```
if (i<100)
    if (i>0)
        printf("minore di 100 e maggiore di zero");
    else
        printf("minore di 100 ma non maggiore di zero");
```

Come fa capire il messaggio della seconda `printf`, l'`else` che abbiamo aggiunto si riferisce al secondo `if`, cioè a quello più interno, il quale insieme alle due `printf` e all'`else` costituiscono ancora un'unica istruzione. Per fare in modo che l'`else` si riferisca al primo `if` bisogna usare le parentesi graffe:

```
if (i<100) {
    if (i>0)
        printf("minore di 100 e maggiore di zero");
}
else
    printf("maggiore o uguale a 100");
```

Se invece avessimo anche l'`else` dell'`if` più interno le parentesi graffe sarebbero superflue, e si potrebbe scrivere:

```
if (i<100)
    if (i>0)
        printf("minore di 100 e maggiore di zero");
    else
        printf("minore di 100 ma non maggiore di zero");
else
    printf("maggiore o uguale a 100");
```

Ciò è reso possibile dal fatto che il tutto viene considerato come un'unica istruzione. Modifichiamo ora il nostro esempio in modo che vengano visualizzati due messaggi diversi a seconda che la variabile `i` sia maggiore oppure uguale a 100, e non soltanto un messaggio generico come sopra:

```
if (i<100)
    if (i>0)
        printf("minore di 100 e maggiore di zero");
    else
        printf("minore di 100 ma non maggiore di zero");
else
    if (i==100)
        printf("uguale a 100");
    else
        printf("maggiore di 100");
```

Come si può osservare, anche la parte istruzione dell'`else` può essere un'istruzione `if`, la quale, a sua volta, può avere un proprio ramo `else`. L'operatore `==` ha il significato di *uguale a*, risponde vero se l'operando che lo precede e quello che lo segue sono uguali, falso altrimenti.


Un'ulteriore modifica del nostro esempio consiste nel dare più informazioni nel caso che `i` sia minore di 100:

```
1  if (i<100)
2      if (i>0)
3          printf("minore di 100 e maggiore di zero");
```

```

4     else
5         if(i==0)
6             printf("uguale a zero");
7         else
8             printf("minore di zero");
9     else
10        if(i==100)
11            printf("uguale a 100");
12        else
13            printf("maggiore di 100");

```

È importante notare che nell'esempio precedente non sono richieste parentesi graffe ; ciò in virtù del fatto che:

- le righe 1..13 sono un'unica istruzione if-else la quale ha per *istruzione1* le righe 2..8 e per *istruzione2* le righe 10..13;
- le righe 2..8 sono un'unica istruzione if-else la quale ha per *istruzione1* la riga 3 e per *istruzione2* le righe 5..8;
- le righe 5..8 sono un'unica istruzione if-else la quale ha per *istruzione1* la riga 6 e per *istruzione2* la riga 8;
- le righe 10..13 sono un'unica istruzione if-else la quale ha per *istruzione1* la riga 11 e per *istruzione2* la riga 13.

✓ NOTA

Quanto sopra può essere scritto in modo più compatto:

```

if(i<100)
    if(i>0)
        printf("minore di 100 e maggiore di zero");
    else if(i==0)
        printf("uguale a zero");
    else
        printf("minore di zero");
else if(i==100)
    printf("uguale a 100");
else
    printf("maggiore di 100");

```

Questo schema è frequente nei programmi C ed è molto comodo per simulare l'istruzione *elseif*, tipica di altri linguaggi.