

Eco-design Digitale di Base per i servizi ICT

Programmazione in C e Python

Obiettivi della lezione

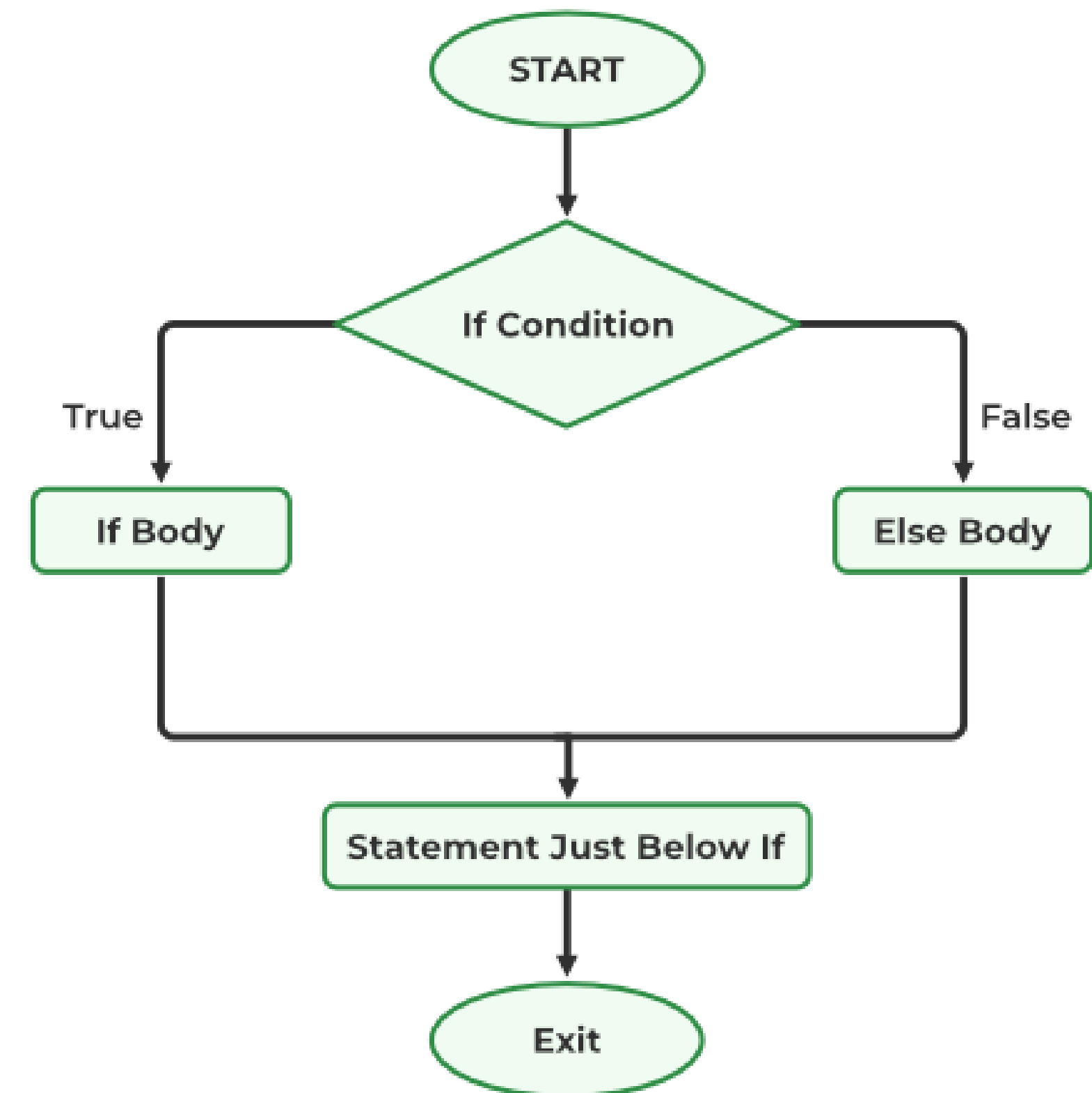
- Comprendere e applicare le strutture di controllo del flusso in C
- Analizzare la struttura e l'utilizzo delle funzioni
- Esplorare il concetto di scope delle variabili

Controllo del flusso: panoramica

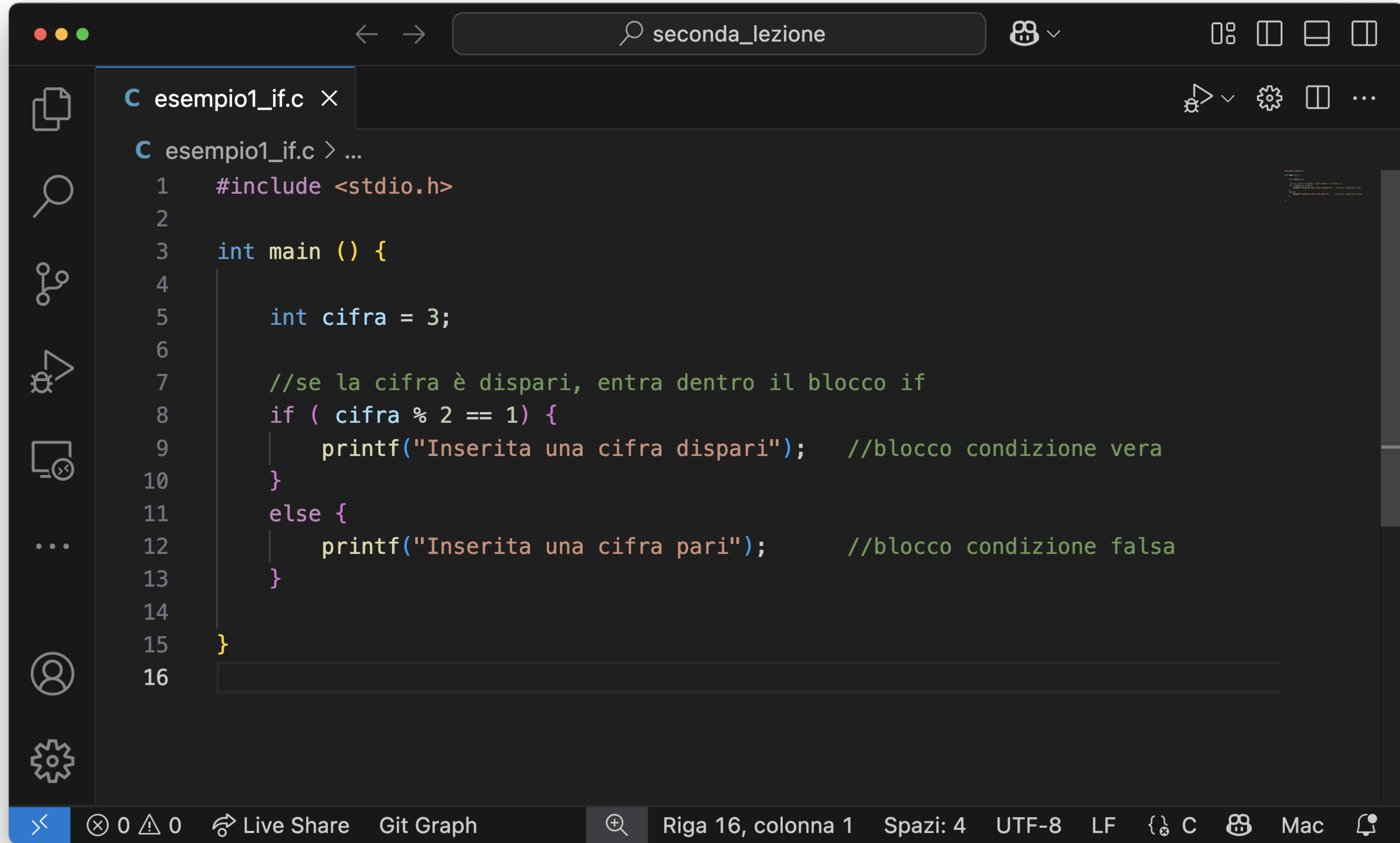
- Il controllo del flusso rappresenta il punto fondamentale della programmazione..
- L'importanza del controllo del flusso si manifesta nella possibilità di creare programmi che non seguono semplicemente una sequenza lineare predefinita, ma che possono reagire a input diversi, elaborare quantità variabili di dati e prendere decisioni logiche.
- Le tre categorie principali: ***selezione***, ***iterazione*** e ***salto***

Controllo del flusso: **selezione**

- Valutazione espressione booleana (Vero/Falso)
- Se vera, esegue un blocco di codice associato
- Combinabile con operatori logici (&&, ||, !) per gestire condizioni complesse



Selezione



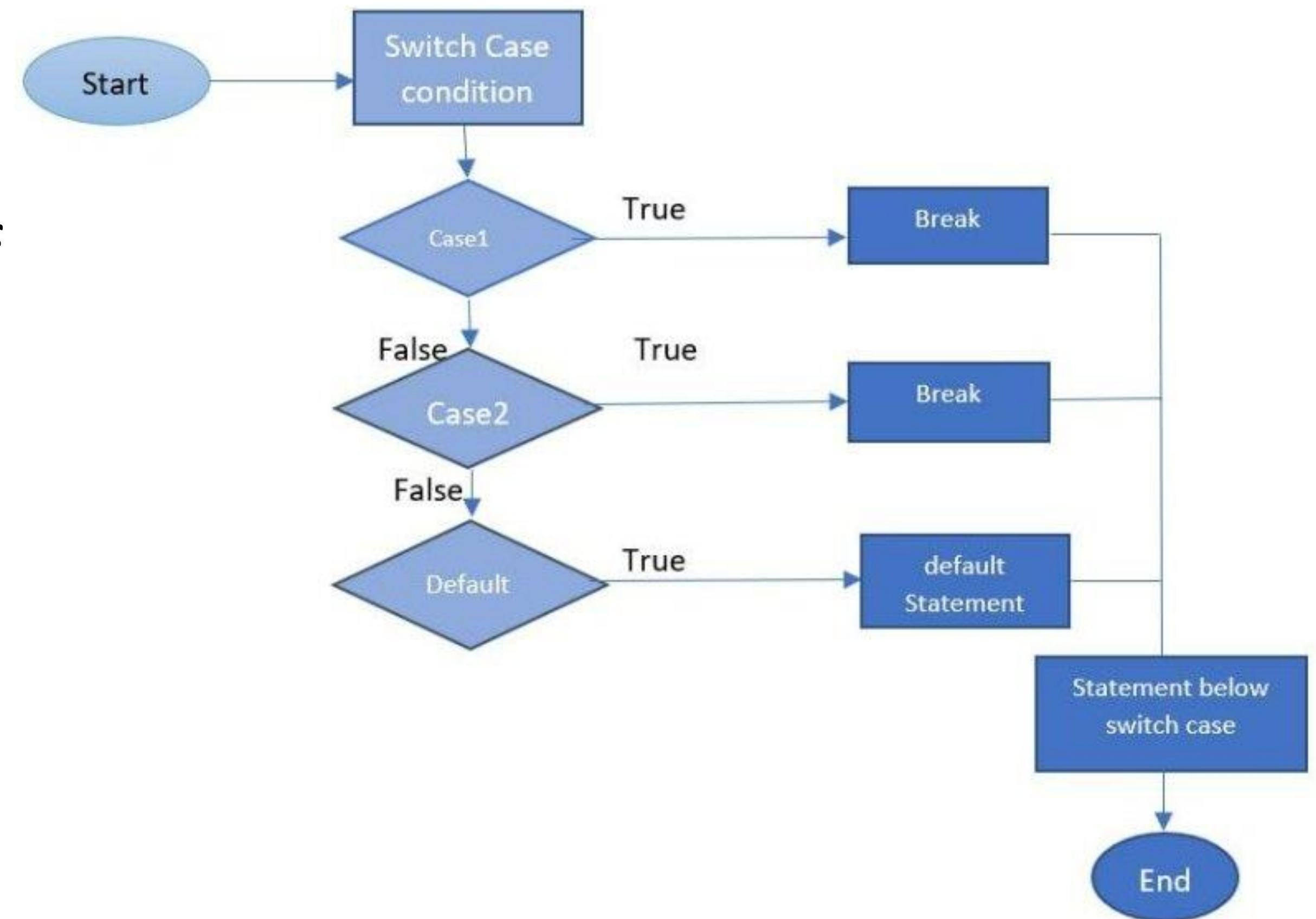
The screenshot shows a code editor window with a dark theme. The title bar at the top includes window control buttons, a search bar containing 'seconda_lezione', and icons for file explorer, settings, and other tools. The editor displays a C program named 'esempio1_if.c'. The code is as follows:

```
1  #include <stdio.h>
2
3  int main () {
4      int cifra = 3;
5
6      //se la cifra è dispari, entra dentro il blocco if
7      if ( cifra % 2 == 1) {
8          printf("Inserita una cifra dispari");    //blocco condizione vera
9      }
10     else {
11         printf("Inserita una cifra pari");        //blocco condizione falsa
12     }
13 }
14
15
16
```

The status bar at the bottom provides additional information: it shows 'Riga 16, colonna 1', 'Spazi: 4', 'UTF-8', 'LF', the file encoding 'C', the operating system 'Mac', and a notification icon.

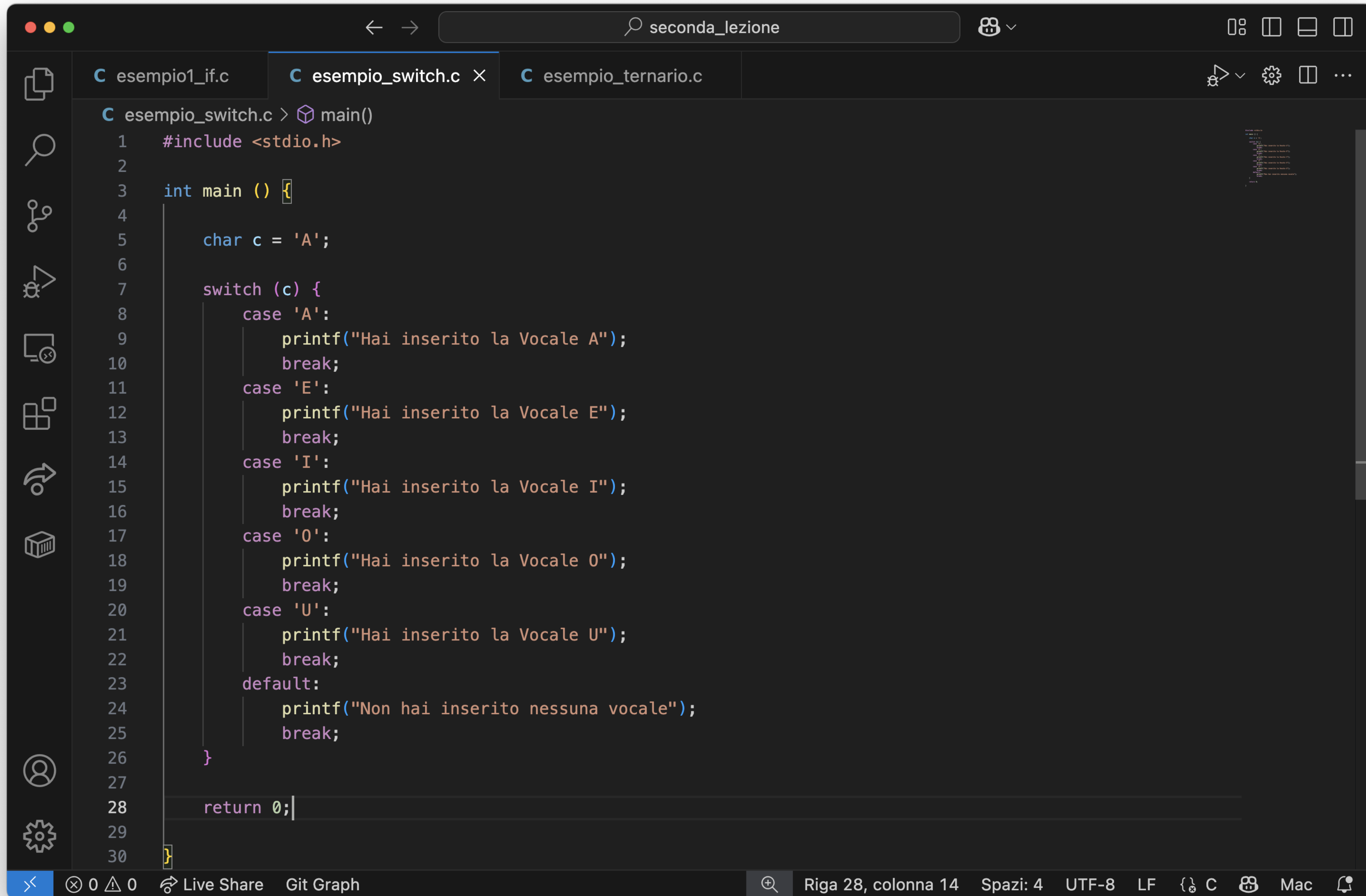
Controllo del flusso: **selezione**

- Utile per eseguire blocchi di codice in base al valore di una variabile
- Alternativa leggibile a molte if-else if
- Ogni case termina con break per evitare il fall-through
- Il blocco default viene eseguito se nessun case corrisponde



Switch Case Flow Chart

Selezione (switch)



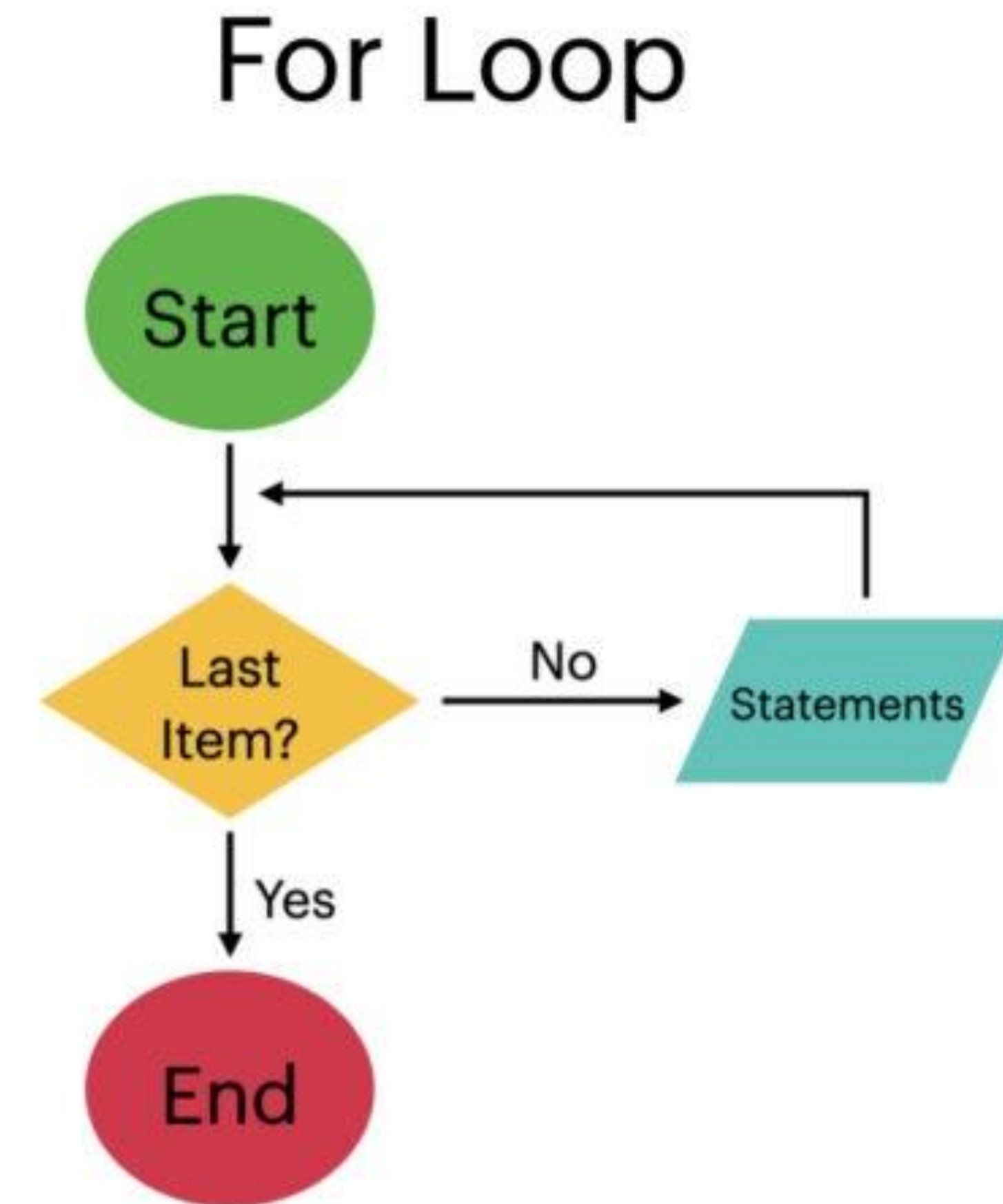
The screenshot shows a code editor with three tabs: `esempio1_if.c`, `esempio_switch.c` (active), and `esempio_ternario.c`. The active tab displays a C program that uses a `switch` statement to check for vowels. The code is as follows:

```
C esempio_switch.c > main()
1  #include <stdio.h>
2
3  int main () {
4
5      char c = 'A';
6
7      switch (c) {
8          case 'A':
9              printf("Hai inserito la Vocale A");
10             break;
11          case 'E':
12              printf("Hai inserito la Vocale E");
13              break;
14          case 'I':
15              printf("Hai inserito la Vocale I");
16              break;
17          case 'O':
18              printf("Hai inserito la Vocale O");
19              break;
20          case 'U':
21              printf("Hai inserito la Vocale U");
22              break;
23          default:
24              printf("Non hai inserito nessuna vocale");
25              break;
26      }
27
28      return 0;
29
30 }
```

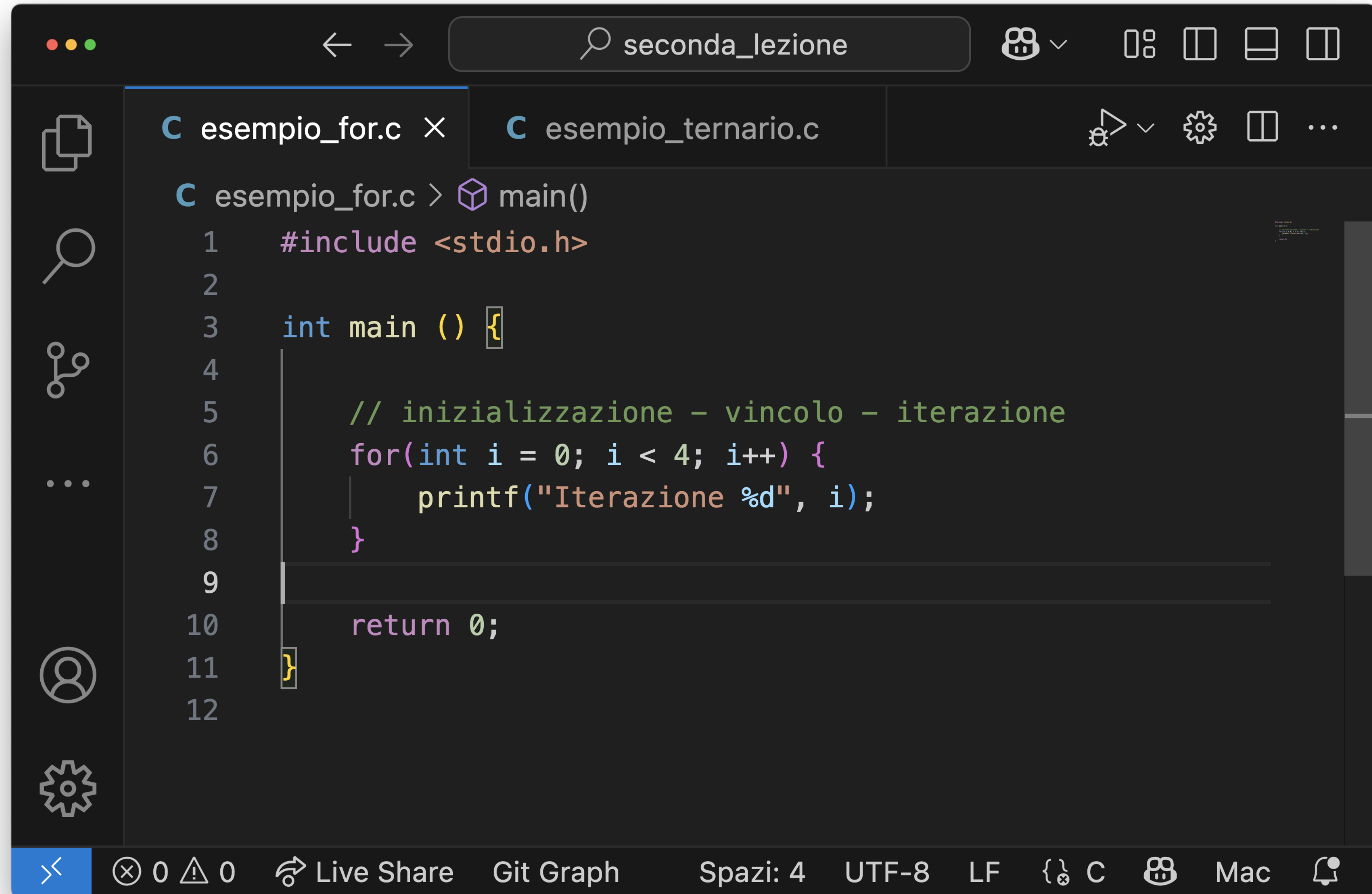
The editor interface includes a sidebar with icons for file explorer, search, source control, and other tools. The top bar shows the file name `seconda_lezione` and window management icons. The bottom status bar indicates the current position (Riga 28, colonna 14), spacing (Spazi: 4), encoding (UTF-8), line endings (LF), and the operating system (Mac).

Controllo del flusso: iterazione controllata

- Struttura iterativa con inizializzazione, condizione, incremento
- Esegue il blocco finché la condizione è vera
- Ottimo per iterazioni con conteggi noti



Iterazione (for)



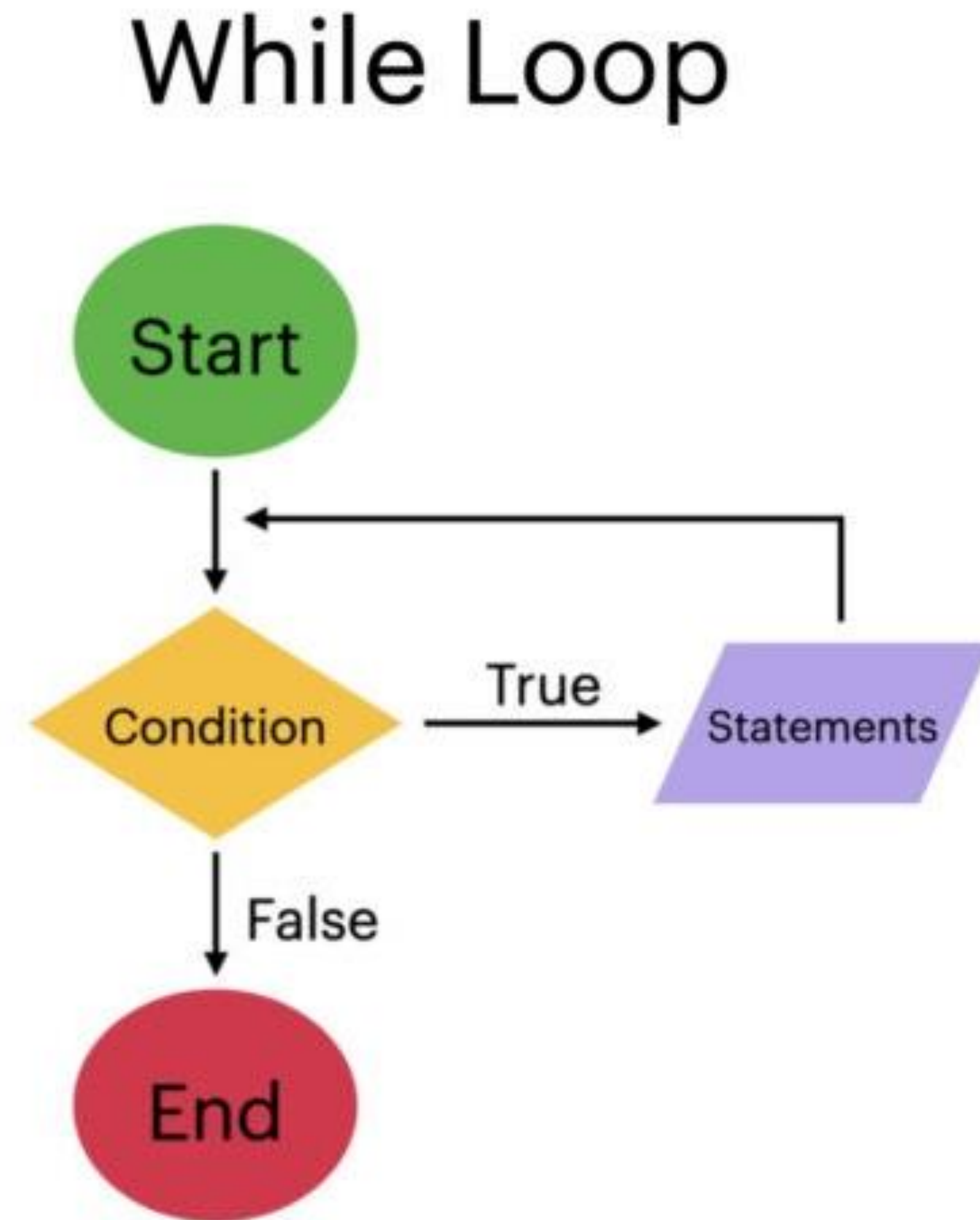
The screenshot shows a code editor window with two tabs: `esempio_for.c` and `esempio_ternario.c`. The active tab is `esempio_for.c`, which contains the following C code:

```
C esempio_for.c > main()
1  #include <stdio.h>
2
3  int main () {
4
5      // inizializzazione - vincolo - iterazione
6      for(int i = 0; i < 4; i++) {
7          printf("Iterazione %d", i);
8      }
9
10     return 0;
11 }
12
```

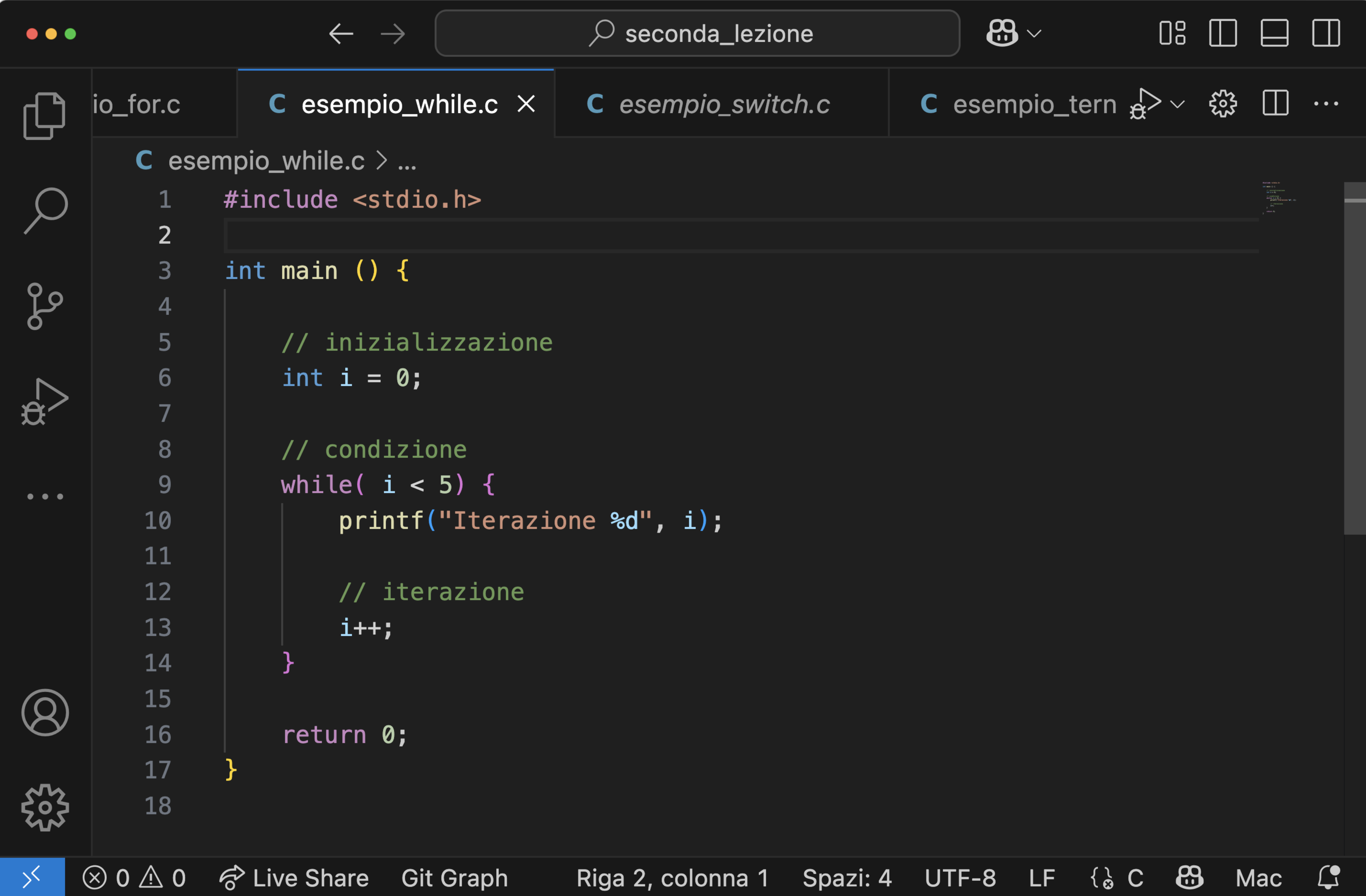
The editor interface includes a sidebar on the left with icons for file explorer, search, source control, and settings. The top bar shows a search bar with the text `seconda_lezione` and various window management icons. The bottom status bar displays information such as line counts (0 errors, 0 warnings), Live Share status, Git Graph, indentation (4 spaces), encoding (UTF-8), line endings (LF), language (C), and the operating system (Mac).

Controllo del flusso: iterazione condizionale

- Esegue il blocco finché la condizione è vera
- Valuta la condizione **prima** di entrare nel ciclo
- Ideale per condizioni di continuazione non note a priori



Iterazione (while)



The image shows a code editor window with a dark theme. The title bar at the top includes window control buttons, a search bar containing 'seconda_lezione', and icons for file management and settings. The editor has several tabs open: 'io_for.c', 'esempio_while.c' (active), 'esempio_switch.c', and 'esempio_tern'. The active tab displays a C program that demonstrates a while loop. The code is as follows:

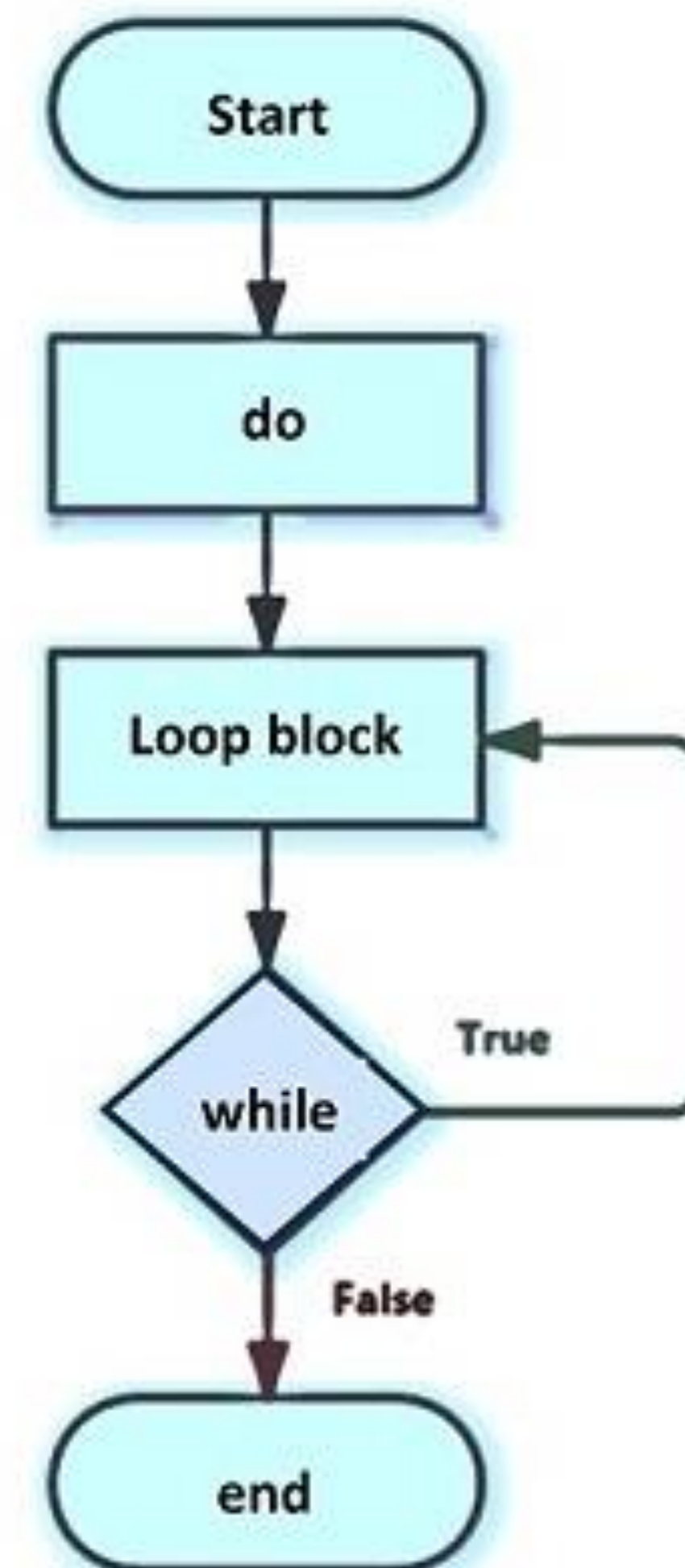
```
1  #include <stdio.h>
2
3  int main () {
4
5      // inizializzazione
6      int i = 0;
7
8      // condizione
9      while( i < 5) {
10         printf("Iterazione %d", i);
11
12         // iterazione
13         i++;
14     }
15
16     return 0;
17 }
18
```

The status bar at the bottom provides additional information: it shows a zoom level of 0%, 0 warnings or errors, a 'Live Share' button, 'Git Graph' integration, the current cursor position 'Riga 2, colonna 1', and various configuration details including 'Spazi: 4', 'UTF-8', 'LF', 'C' language, 'Mac' platform, and a bell icon for notifications.

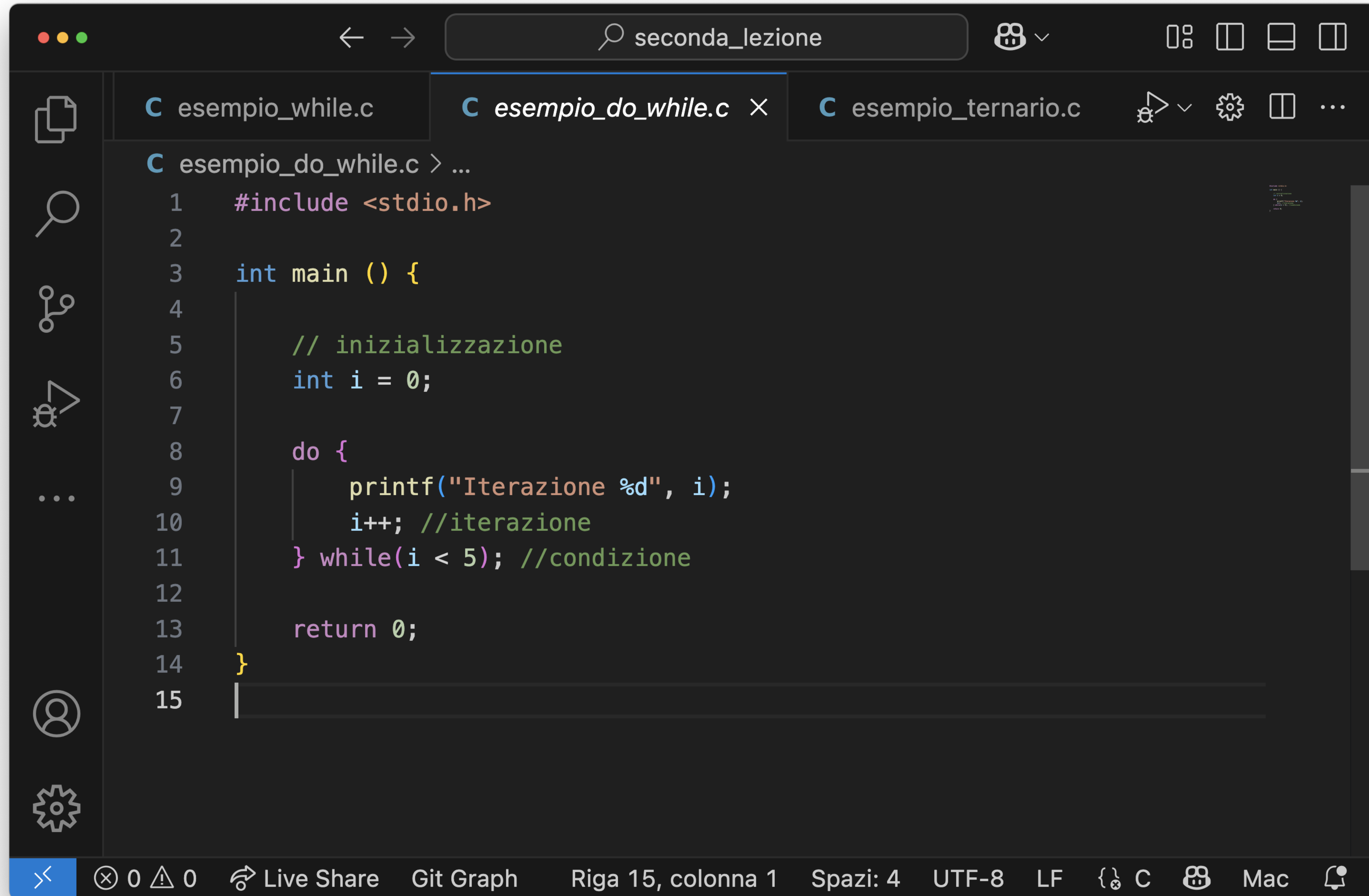
Controllo del flusso: iterazione post-condizionale

- Come while, ma la condizione è valutata **dopo** il primo ciclo
- Garantisce almeno una esecuzione del blocco
- Utile per menu e input interattivi

DO-WHILE



Iterazione (do while)



The screenshot shows a code editor with three tabs: `esempio_while.c`, `esempio_do_while.c` (active), and `esempio_ternario.c`. The active tab contains the following C code:

```
1  #include <stdio.h>
2
3  int main () {
4
5      // inizializzazione
6      int i = 0;
7
8      do {
9          printf("Iterazione %d", i);
10         i++; //iterazione
11     } while(i < 5); //condizione
12
13     return 0;
14 }
15
```

The editor interface includes a top bar with navigation arrows, a search bar containing `seconda_lezione`, and window management icons. A left sidebar contains icons for file explorer, search, source control, and other tools. The bottom status bar displays icons for file operations, error/warning counts, and various settings like encoding (UTF-8), line endings (LF), and theme (Mac).

Istruzioni di salto

- **break:** esce immediatamente dal ciclo o dallo switch
- **continue:** salta all'iterazione successiva
- **return:** esce da una funzione e restituisce (eventualmente) un valore

Esercizio guidato

Scrivi un programma in linguaggio C che implementi un **gioco di indovinare il numero**, con la possibilità di scegliere tra tre livelli di difficoltà:

1. **Facile**: numero tra 1 e 20, 8 tentativi
2. **Medio**: numero tra 1 e 50, 6 tentativi
3. **Difficile**: numero tra 1 e 100, 4 tentativi

Il programma deve includere anche:

- Opzione per visualizzare le **statistiche** della sessione
- Opzione per **uscire** dal gioco

Secondo esercizio

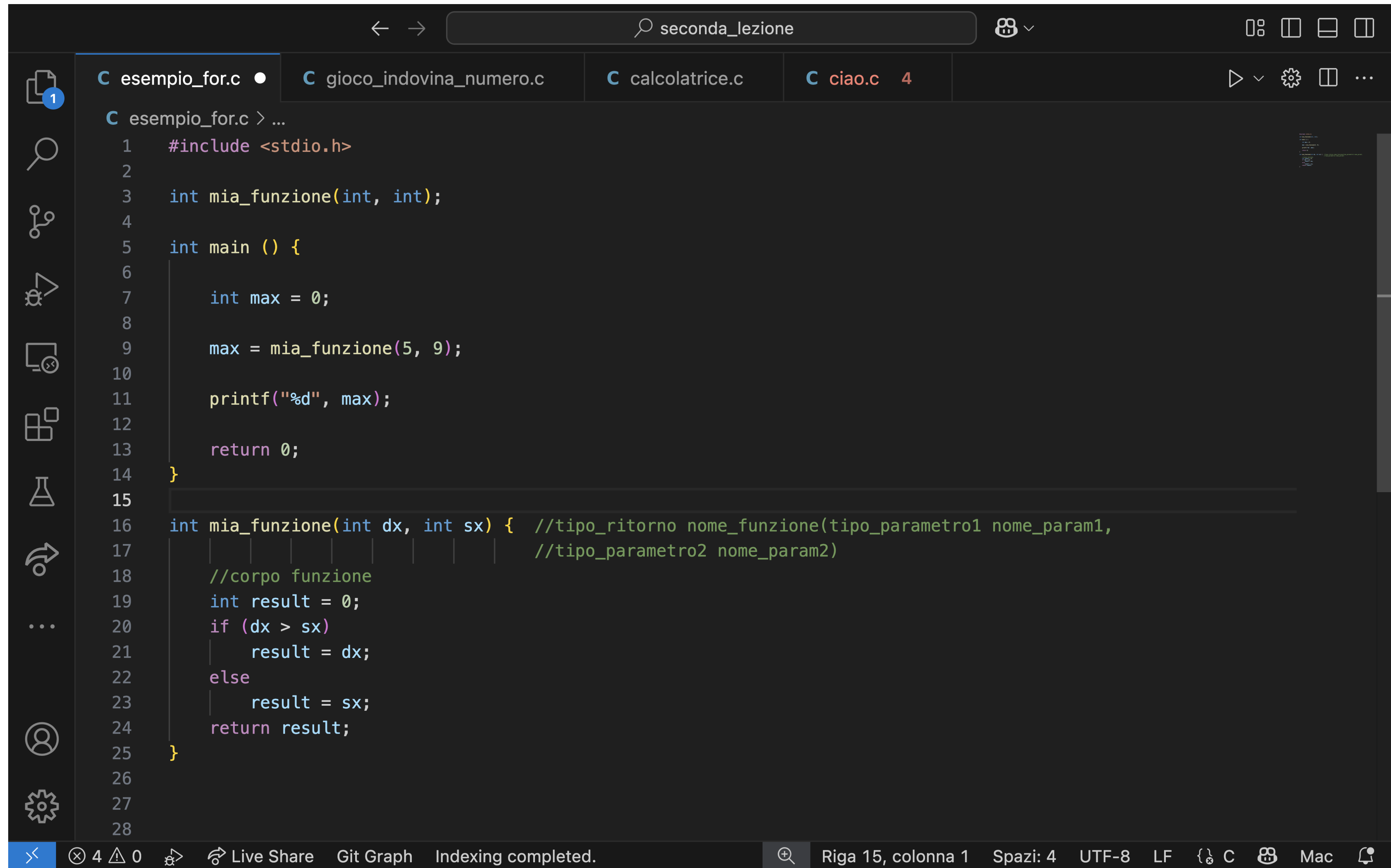
Scrivi un programma in linguaggio C che implementi una **calcolatrice interattiva** con menu, che consenta all'utente di scegliere l'operazione da eseguire tra:

1. Addizione
2. Sottrazione
3. Moltiplicazione
4. Divisione (con controllo della divisione per zero)
5. Uscita dal programma

Funzioni in C: panoramica

- Un blocco di codice riutilizzabile, definito una volta, chiamato più volte.
- Le funzioni favoriscono:
 - Riutilizzo del codice
 - Modularità
 - Leggibilità e manutenibilità

Funzioni in C: sintassi



The screenshot shows a code editor with a dark theme. At the top, there's a search bar with the text "seconda_lezione". Below it, there are four tabs: "esempio_for.c", "gioco_indovina_numero.c", "calcolatrice.c", and "ciao.c" (which is highlighted in red and has a "4" next to it). The main editor area shows the code for "esempio_for.c". The code is as follows:

```
1  #include <stdio.h>
2
3  int mia_funzione(int, int);
4
5  int main () {
6
7      int max = 0;
8
9      max = mia_funzione(5, 9);
10
11     printf("%d", max);
12
13     return 0;
14 }
15
16 int mia_funzione(int dx, int sx) { //tipo_ritorno nome_funzione(tipo_parametro1 nome_param1,
17     //                                //tipo_parametro2 nome_param2)
18     //corpo funzione
19     int result = 0;
20     if (dx > sx)
21         result = dx;
22     else
23         result = sx;
24     return result;
25 }
26
27
28
```

At the bottom of the editor, there's a status bar with various icons and text: "Riga 15, colonna 1", "Spazi: 4", "UTF-8", "LF", "C", "Mac", and "Indexing completed."

Funzioni in C: buona pratica

- Funzioni brevi e con responsabilità ben definite
- Nomi descrittivi per funzione e parametri
- Commentare solo quando il codice non è autoesplicativo
- Separare logica da I/O dove possibile

Scope delle variabili

- **Locale:** dichiarata dentro una funzione o un blocco {}
- **Globale:** dichiarata fuori da tutte le funzioni
- Scope determina **visibilità e durata** della variabile
- Variabili locali vengono create e distrutte nel blocco in cui vivono

Altri esercizi

- Valutazione voto

Scrivere un programma che legga un voto numerico e stampi:

“Ottimo” se ≥ 28

“Buono” se 24–27

“Sufficiente” se 18–23

“Insufficiente” altrimenti

- Somma dei numeri da 1 a N

Chiedere un numero N, poi sommare tutti i numeri da 1 a N usando while

Altri esercizi

- Calcolo del fattoriale

Scrivere una funzione `int fattoriale(int n)` e usarla nel main per calcolare il fattoriale di un numero