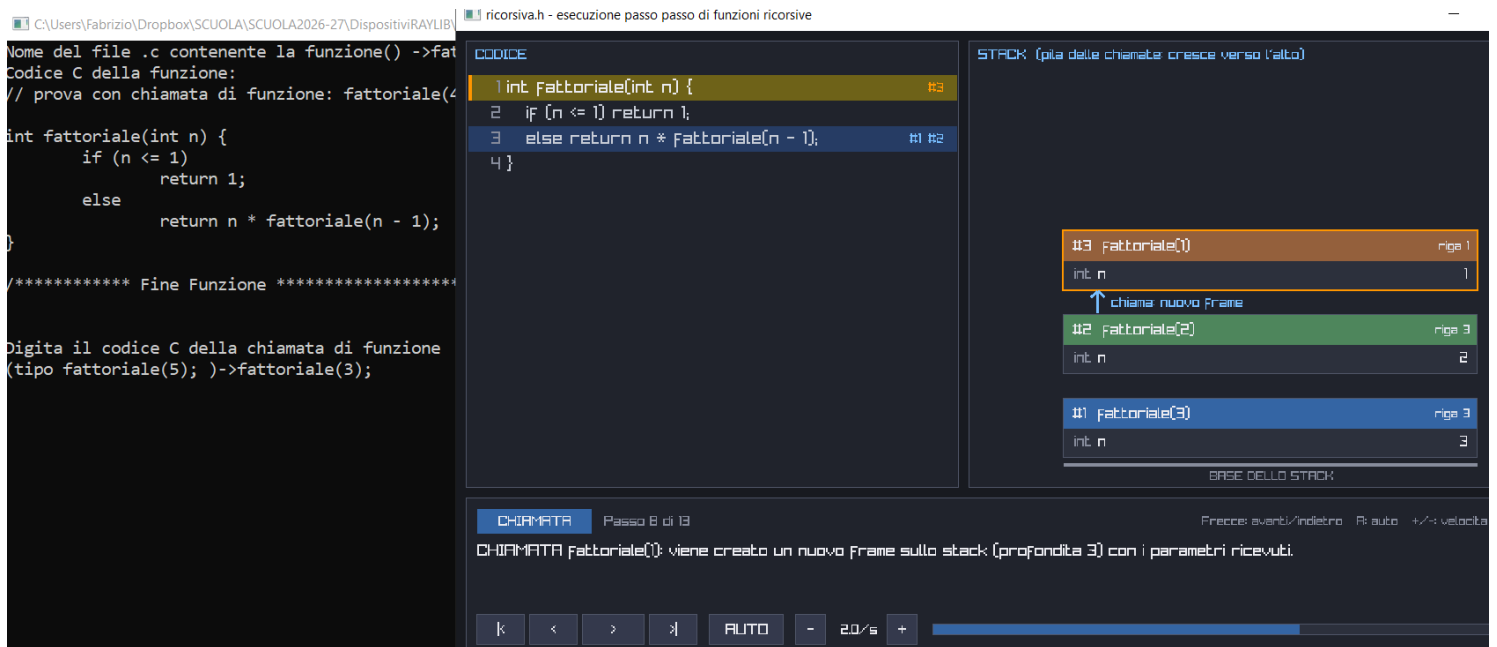


Debug — esecuzione passo passo di funzioni C



1. Cos'è Debug

Debug è un piccolo interprete C con interfaccia grafica (Raylib) che permette di **eseguire passo passo** una funzione scritta in un sottoinsieme di C, mostrando in tempo reale:

- il **codice sorgente** con evidenziata la riga in esecuzione,
- lo **stack delle chiamate** (i vari “frame” con i valori delle variabili),
- l'**output** di eventuali `printf`,
- la **memoria** degli array/stringhe passati come puntatori.

È perfetto per capire visivamente come funziona la **ricorsione** (es. `fattoriale`) e gli **algoritmi iterativi** (es. `bubblesort`).

2. Requisiti per ricostruire Debug (non necessario, puoi scaricare ed usare Debug.exe)

- Windows (o Linux/macOS) con un compilatore C (gcc/MinGW).
- La libreria **Raylib** installata.
- I file:
 - `Debug.c` (il programma principale),
 - `ricorsivaVers2.h` (la libreria header-only),
 - i file `.c` di esempio (`fattoriale.c`, `bubblesort.c`).

Per compilare (esempio con MinGW):

```
gcc Debug.c -o Debug.exe -lraylib -lopengl32 -lgdi32 -lwinmm
```

3. Come si usa

Passo 1 — Avvia Debug .exe

Si apre una finestra del terminale che chiede:

Nome del file .c contenente la funzione() ->

Passo 2 — Digita il nome del file

Per esempio:

fattoriale.c

Il programma legge il file e mostra a video il codice trovato.

Passo 3 — Inserisci la chiamata di funzione

Il programma chiede:

Digita il codice C della chiamata di funzione (tipo fattoriale(5);)->

Digita la chiamata che vuoi analizzare. **Attenzione a rispettare la sintassi dell'esempio commentato nel file.**

- Per fattoriale.c → fattoriale(4);
- Per bubblesort.c → bubblesort({2,-3,0,7,5},5);

💡 **Nota importante:** essendo un interprete semplificato, gli array non si possono dichiarare localmente. Per passarli si usa la sintassi con **liste tra graffe** {2,-3,0,7,5} direttamente nella chiamata.

Passo 4 — Si apre la finestra grafica

Vedrai 4 pannelli:

Pannello	Cosa mostra
CODICE	Il sorgente con la riga corrente evidenziata in giallo
OUTPUT	Solo se usi printf: i testi prodotti
STACK	I frame delle chiamate (cresce verso l'alto)
MEMORIA	Solo se usi array/stringhe: i loro valori

4. Comandi nella finestra

Tasto	Azione
→ / Spazio / Invio	Passo avanti
← / Backspace	Passo indietro
Home / End	Vai all'inizio / alla fine
A	Avvia / ferma l'esecuzione automatica
+ / -	Aumenta / diminuisci la velocità automatica
Rotella mouse (sullo stack)	Scorre lo stack se non entra tutto
Clic sulla barra in basso	Salta a quel passo
In basso trovi anche dei pulsanti < < > > AUTO - + cliccabili con il mouse, e una descrizione testuale del passo corrente (es. “la condizione è VERA: si esegue il ramo dell'if”).	

5. Esempio guidato: fattoriale (4)

Il file `fattoriale.c` contiene:

```
int fattoriale(int n) {  
    if (n <= 1)  
        return 1;  
    else  
        return n * fattoriale(n - 1);  
}
```

Premendo → vedrai questa sequenza:

1. **Chiamata iniziale** → nasce il frame #1 **fattoriale(n=4)**.
2. La condizione `n <= 1` è **falsa** → si va nel ramo `else`.
3. Si valuta `n * fattoriale(n-1)` — ma prima serve il risultato della chiamata.
4. Nasce il frame #2 **fattoriale(n=3)** **sopra** il precedente → **lo stack cresce verso l'alto**.
5. Si ripete per `n=2` → frame #3.
6. Per `n=1` la condizione è **vera** → `return 1` → **il frame #4 viene rimosso**.
7. Ogni ritorno “risale”: `1 → 2 → 6 → 24`.
8. Lo stack si svuota e appare in grande il **risultato: 24**.

È il modo migliore per vedere che la ricorsione: ogni frame aspetta il risultato di quello sotto.

6. Esempio guidato: bubblesort ({2, -3, 0, 7, 5}, 5)

Chiamata da usare:

```
bubblesort({2, -3, 0, 7, 5}, 5);
```

- Compare il pannello **MEMORIA** con la lista {2, -3, 0, 7, 5}.
- Ad ogni passo, nel pannello memoria vedrai le celle cambiare colore quando avviene uno **scambio** (`vet[i] > vet[j]`).
- Nel pannello **STACK** vedrai una sola chiamata (nessuna ricorsione) ma potrai seguire l'evoluzione di `i`, `j`, deposito e del puntatore `vet`.

Alla fine l'array sarà ordinato: {-3, 0, 2, 5, 7}.

7. Cosa è supportato / cosa no

Il linguaggio è un **sottoinsieme di C**. In breve:

- ✓ **Supportato** - Tipi base: `int`, `long`, `char`, `float`, `double`, `void`, puntatori, `int vet[]` come parametro. - `if/else`, `while`, `do-while`, `for`, `break`, `continue`, `return`, operatore ternario `?:`. - Ricorsione (anche indiretta) e più funzioni nello stesso file. - `printf`, `putchar`, `puts`, `abs`, `labs`, `fabs`, `sqrt`, `pow`, `floor`, `ceil`. - Array/stringhe passati come parametri.
 - ✗ **Non supportato** - `struct`, array locali, variabili globali, `&` (indirizzo di), `malloc`, `#define`. - Le **liste tra graffe** {1, 2, 3} si possono usare **solo nella chiamata iniziale**.
-

8. Errori comuni e cosa fare

Messaggio

Attenzione, file [...] non trovato!

atteso ';' ma trovato ...

STACK OVERFLOW: la ricorsione supera 100 chiamate...

Raggiunto il limite di 6000 passi

gli array locali non sono supportati

La finestra si apre ma non mostra nulla

Causa / Soluzione

Il file .c non è nella stessa cartella di Debug.exe o hai sbagliato il nome.

Manca un ; oppure la sintassi non è supportata. Guarda la riga evidenziata nel pannello CODICE.

Probabilmente manca il **caso base** o non viene mai raggiunto.

Ciclo infinito o ricorsione troppo lunga. Puoi alzare il limite definendo RICORSIVA_MAX_PASSI prima dell'#include.

Passa l'array come parametro e crealo nella **chiamata iniziale** con {...}.

Il codice non conteneva funzioni valide: controlla il .c.

9. Consigli per l'uso didattico

- Usa fattoriale(4) o fattoriale(5): con valori troppo alti la pila diventa lunga e difficile da leggere.
- Prova a introdurre un **caso base mancante** (es. rimuovi `if (n <= 1) return 1;`): vedrai lo STACK OVERFLOW a video.
- Prova a scrivere la versione **iterativa** del fattoriale nello stesso file e confronta i due stack: nello iterativo c'è un solo frame.
- Nel bubblesort, metti la velocità automatica al massimo (A++): vedrai le celle scambiarsi in tempo reale.
- Usa la **barra di avanzamento** in basso per tornare indietro e rivedere un punto preciso senza rifare tutti i passi.

10. Riepilogo rapido (cheat-sheet)

[facoltativo 0) Compila: gcc Debug.c -o Debug.exe -lraylib -lopengl32 -lgdi32 -lwinmm]

1) Avvia: Debug.exe

2) Scrivi: fattoriale.c (o qualsiasi altra funzione C)

4) Scrivi: fattoriale(4); (o la chiamata della tua funzione C)

5) Comandi: → avanti, ← indietro, A auto, +/- velocità

Buon debug!  