ByteRebels 08-03-2024

Ad una prima analisi effettuata sul codice appaiono evidenti degli errori sintattici e che impediscono la corretta compilazione del codice.

Facendo riferimento al codice originario gli **errori sintattici** più evidenti sono seguenti:

-Problematiche relative ai segnaposto: alle righe 47-48 del codice entrambe le funzioni “scanf” utilizzano il segnaposto sbagliato, entrambe le variabili infatti sono di tipo “short int” e richiedono quindi il segnaposto “%hd” contrariamente da quanto indicato.

-Valutazione del carattere Newline a riga 14, la funzione “main” in cui è necessario aggiungere uno spazio prima del segnaposto nella funzione “scanf” o, più correttamente, aggiungendo una funzione “getchar()” per consumare il newline.

-Scambiati i comandi % e / nella funzione “Dividi” a riga 64.

-Referenziazione in utile del vettore “stringa” nella funzione “scanf” di riga 77. I vettori sono puntatori alla prima cella di sé stessi, questo significa che contengono di fatto già l’indirizzo della prima cella di memoria che fa riferimento alla cella 0 del vettore, per tanto è sbagliato cercare di estrarre l’indirizzo dell’indirizzo con “&”.

-Problemi relativi alla gestione degli spazi nell’acquisizione della striga nella funzione “ins_string()”.

Il codice presenta anche degli **errori logici**:

-Il mancato controllo case sensitive per quanto concerne lo switch case nel main.

-L’assenza di un default case nel medesimo switch case.

-Il mancato controllo della divisione per zero nella funzione “dividi” dedicata.

Risoluzione problemi:

Il codice proposto risolve i problemi di sintassi ed introduce strategie per ovviare ai problemi.

In primo luogo si sottolineano appunto l’introduzione del default case e di una funzione che rende maiuscolo ogni carattere inserito dall’utente in fase di scelta, annullando di fatto le problematiche relative all’inserimento di un’opzione che non esiste o erroneamente inserita in minuscolo

Si desidera poi porre l’attenzione sulla riscrittura della funzione “ins_stringa” che, sebbene non stravolga il funzionamento della stessa e non tenga in considerazione i caratteri inseriti oltre il decimo, pone però rimedio ai problemi di acquisizione legati alla presenza di spazi, che nella versione originale interrompevano l’acquisizione.

Complessità:

La complessità del nuovo codice risulta essere O(1) nella funzione “menu” e in quella “ins_string” (perché nonostante presenti una stringa si può approssimare ad una complessità tempo-costante O(1)), le funzioni “moltiplica()” e “dividi()” invece hanno complessità variabile a seconda dei numeri inseriti dall’utente.

La complessità dello script è invece O(1) poiché utilizza una quantità cpstante di memoria.