dan@Kali: ~/Desktop/Programmazione

File Actions Edit View Help

GNU nano 7.2 calcolatore.cpp *

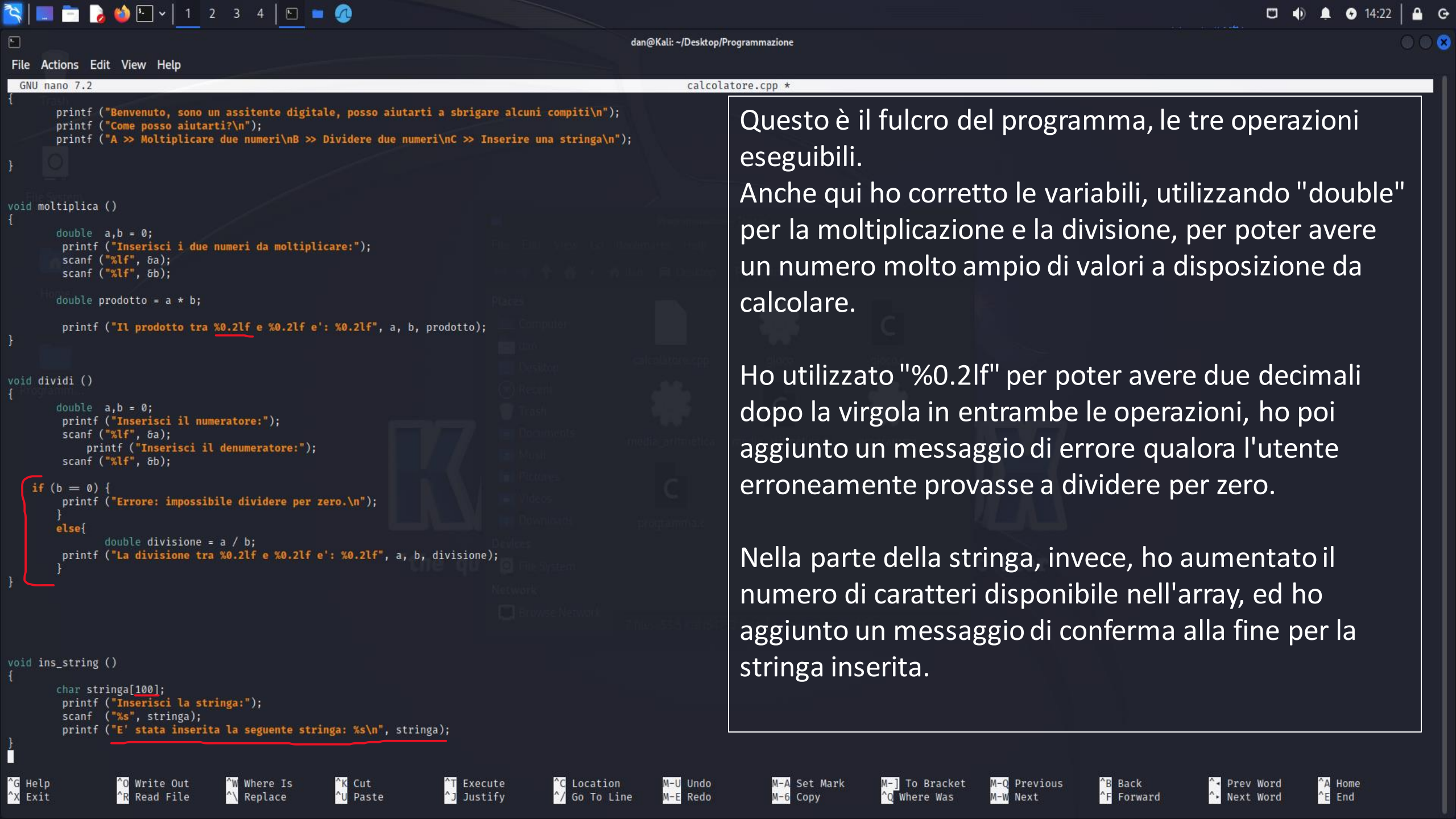
```
int main ()

{
        char scelta = {'\0'};
             menu ();
             scanf (" %c", &scelta);

        switch (scelta)
        {
                case 'A':
                   moltiplica();
                   break;
                case 'B':
                   dividi();
                   break;
                case 'C':
                   ins_string();
                   break;
                default:
                   printf("Seleziona un'opzione corrispondente.\n");
                   break;
        }

return 0;

}


void menu ()
{
        printf ("Benvenuto, sono un assitente digitale, posso aiutarti a sbrigare alcuni compiti\n");
        printf ("Come posso aiutarti?\n");
        printf ("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\n");

}


void moltiplica ()
{
        double  a,b = 0;
         printf ("Inserisci i due numeri da moltiplicare:");
         scanf ("%lf", &a);
         scanf ("%lf", &b);

        double prodotto = a * b;

         printf ("Il prodotto tra %0.2lf e %0.2lf e': %0.2lf", a, b, prodotto);
}
```

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo M-A Set Mark M-] To Bracket M-Q Previous ^B Back ^• Prev Word ^A Home
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-6 Copy ^Q Where Was M-W Next ^F Forward ^• Next Word ^E End

In questo esercizio, il nostro obiettivo è di individuare e ottimizzare qualsiasi errore di sintassi oppure logico, presente nel programma. Oltre a risolvere eventuali casistiche non risolte (esempio: gestire l'input utente).

Questo programma è tecnicamente un calcolatore che ci permette di fare, con un menu e delle opzioni, alcune operazioni. Tra cui;
-Una moltiplicazione tra due numeri
-Una divisione tra due numeri
-L'inserimento di una stringa di caratteri

Per prima cosa ho risolto l'argomento della variabile "char" (%c). Ed ho aggiunto alle opzioni dello switch, una casistica di default nel caso in cui l'utente dovesse erroneamente inserire un carattere oltre i tre indicati (A, B, C).



Questo è il fulcro del programma, le tre operazioni eseguibili.
Anche qui ho corretto le variabili, utilizzando "double" per la moltiplicazione e la divisione, per poter avere un numero molto ampio di valori a disposizione da calcolare.

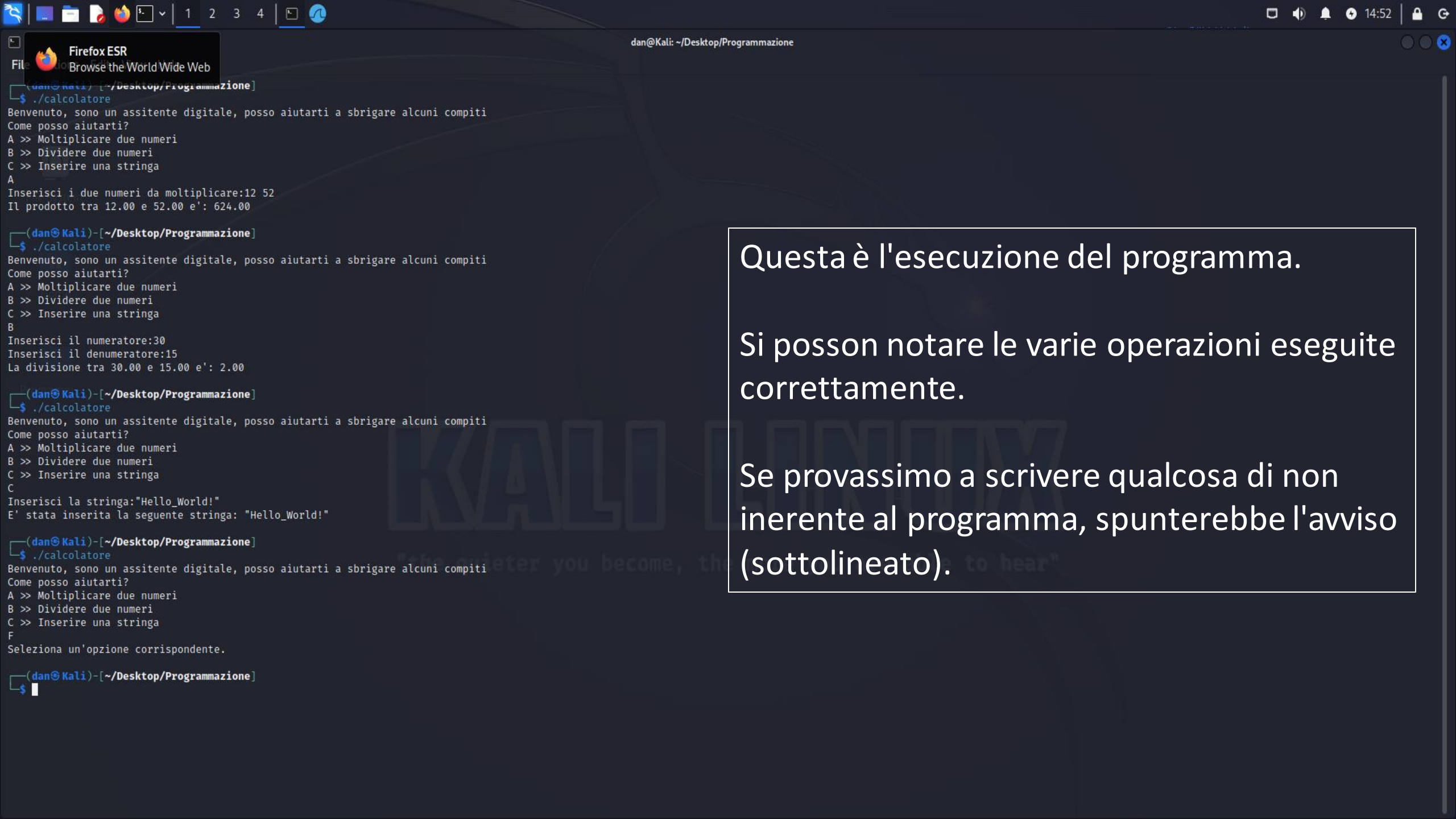
Ho utilizzato "%0.2lf" per poter avere due decimali dopo la virgola in entrambe le operazioni, ho poi aggiunto un messaggio di errore qualora l'utente erroneamente provasse a dividere per zero.

Nella parte della stringa, invece, ho aumentato il numero di caratteri disponibile nell'array, ed ho aggiunto un messaggio di conferma alla fine per la stringa inserita.

Una semplice alternativa che si potrebbe utilizzare per prevenire lo "stack overflow", è mettendo un limite (%99s) nell'argomento della variabile.
Lo "stack overflow" è una tecnica spesso utilizzata dai black hat per mandare in crash il programma, oppure, per inserire tramite puntatori, del materiale malevolo all'interno del programma stesso.
Come si può intuire dal nome, si basa sul mandare in "overflow" (straripare, eccedere), la memoria di "stack", una memoria volatile temporanea che si occupa di elaborare le variabili e i dati.

Con questa specifica, il programma non accetterà più di 99 caratteri per la stringa (uno lasciato libero per il carattere di chiusura stringa), prevenendo un eventuale overflow.

```c
void ins_string ()
{
        char stringa[100];
         printf ("Inserisci la stringa:");
         scanf  ("%99s", stringa);
         printf ("E' stata inserita la seguente stringa: %s\n", stringa);
}
```



Questa è l'esecuzione del programma.

Si posson notare le varie operazioni eseguite correttamente.

Se provassimo a scrivere qualcosa di non inerente al programma, spunterebbe l'avviso (sottolineato).