Progetto week2

Il progetto richiedeva di analizzare un codice in linguaggio C possibilmente senza usare un compilatore/interprete, in particolar modo richiedeva di:

1_Capire cosa fa il programma senza eseguirlo

2_Individuare dal codice sorgente le casistiche non standard che il programma non gestisce (esempio, comportamenti potenziali che non sono stati contemplati)

3_Individuare eventuali errori di sintassi / logici

4_Proporre una soluzione per ognuno di essi

Il codice in questione visualizzato su Linux usando l'editor di test Vim.

1 Cosa fa il programma_

Anche se il codice non ha commenti al suo interno (// ……), si capisce lo stesso quale tipo di programma sia, infatti leggendo i nomi delle funzioni (moltiplica, dividi, ins_stringa) e guardando la struttura di queste funzioni con gli operatori presenti dentro esse (*, %) si arriva alla conclusione che sia un semplice programma matematico che dà la possibilità di fare due operazioni matematiche e di scrivere e una stringa.

---

2 Casistiche non standard_

Guardando il codice si capisce che non sono state prese in considerazione:

-Riposte che non rientrino in quelle elencate nel menu, ovvero che non siano A, B, C, (N.B. anche le minuscole non sono prese in considerazione)

-Nelle funzioni moltiplica() e dividi(), non viene presa in considerazione l'ipotesi che l'utente scriva lettere e altri caratteri speciali.

3.1 Errori di sintassi_

riga 14, %s invece di %d

riga 43 short int, invece di double

riga 45 %f invece di %lf

riga 46 %f invece di %lf

riga 48 short in invece di long double

riga 50 %d invece di %2.lf(il 2. è un'indiscrezione mia fatta per comodità per avere solo due decimali)

riga 61 % invece di / (tutti e due operatori che indicano la divisione ma lo slash / è quello più adatto in questa situazione)

riga 61 sostituire int con float, e (float)a /(float) b invece che a/b

riga 63 %d invece di %f

riga 75 [10] forse è poco--> [100] (più un'imprecisione che un errore)

riga 70 &stringa invece di stringa

3.2 Errori logici_

-Manca la possibilità di uscire dal menu e dal programma.

-Una volta eseguita una funzione il programma finisce, avrebbe più senso rimandare al menù.

-Nella funzione della stringa una volta inserita la nostra stringa, il programma termina, va ampliato con un printf()

-per evitare possibili problemi come lo stack overflow, sarebbe meglio inserire una limitazione ai caratteri inseribili.

---

4 Proposta di soluzione ai problemi_

(Una vera e propria ristrutturazione del codice)

File Actions Edit View Help

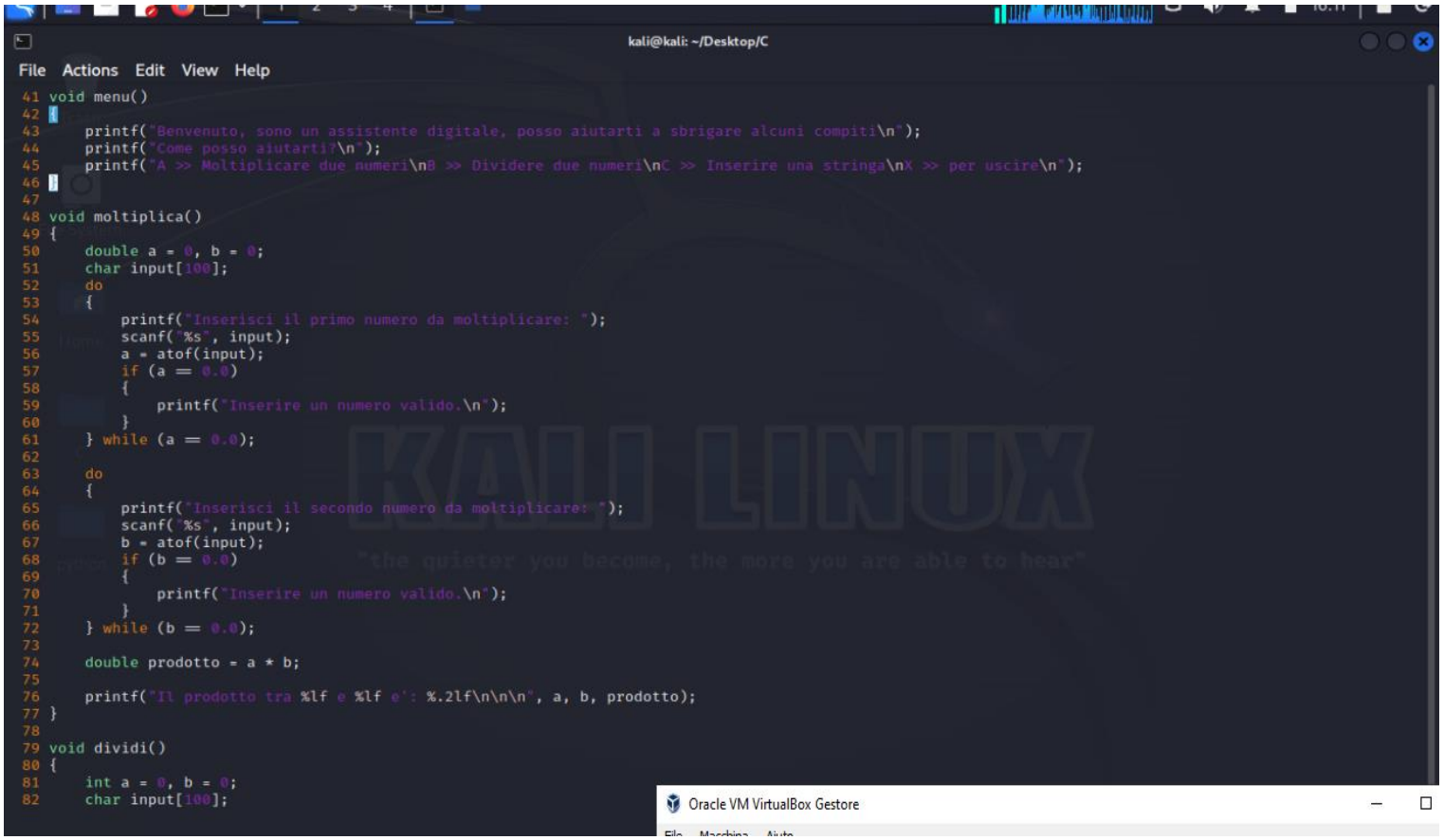
```c
#include <stdio.h>
#include <stdlib.h> // Include le funzioni  atof e atoi

void menu();
void moltiplica();
void dividi();
void ins_string();

int main()
{
    char scelta = {'\0'};
    do
    {
        menu();
        scanf(" %c", &scelta);
        switch (scelta)
        {
        case 'A':
        case 'a':
            moltiplica();
            break;
        case 'B':
        case 'b':
            dividi();
            break;
        case 'C':
        case 'c':
            ins_string();
            break;
        case 'X':
        case 'x':
            break;
        default:
            printf("Scelta non valida\n\n\n");
        }

    } while (scelta != 'X' && scelta != 'x');
    return 0;
}

void menu()
{
```

Novità:

-Libreria, <stdlib.h> che include le funzioni atof e atoi (che spiegherò più avanti)

-Ciclo do-while

-Negli switch sono stati inseriti i “case” anche per le lettere minuscole, il caso X, x per terminare il programma, e il “default” in caso di Digitazione errata con la frase “Scelta non valida”




```c
void menu()
{
    printf("Benvenuto, sono un assistente 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\nX >> per uscire\n");
}

void moltiplica()
{
    double a = 0, b = 0;
    char input[100];
    do
    {
        printf("Inserisci il primo numero da moltiplicare: ");
        scanf("%s", input);
        a = atof(input);
        if (a == 0.0)
        {
            printf("Inserire un numero valido.\n");
        }
    } while (a == 0.0);

    do
    {
        printf("Inserisci il secondo numero da moltiplicare: ");
        scanf("%s", input);
        b = atof(input);
        if (b == 0.0)
        {
            printf("Inserire un numero valido.\n");
        }
    } while (b == 0.0);

    double prodotto = a * b;

    printf("Il prodotto tra %lf e %lf e': %.2lf\n\n\n", a, b, prodotto);
}

void dividi()
{
    int a = 0, b = 0;
    char input[100];
```

-menu():

Il menù è molto simile alla prima versione, semplicemente come già detto è presente nel printf() la voce per l'opzione

"X>> per uscire"

-moltiplica():

Nella funzione moltiplica l'input dell'utente viene letto come una stringa utilizzando scanf("%s", input);.

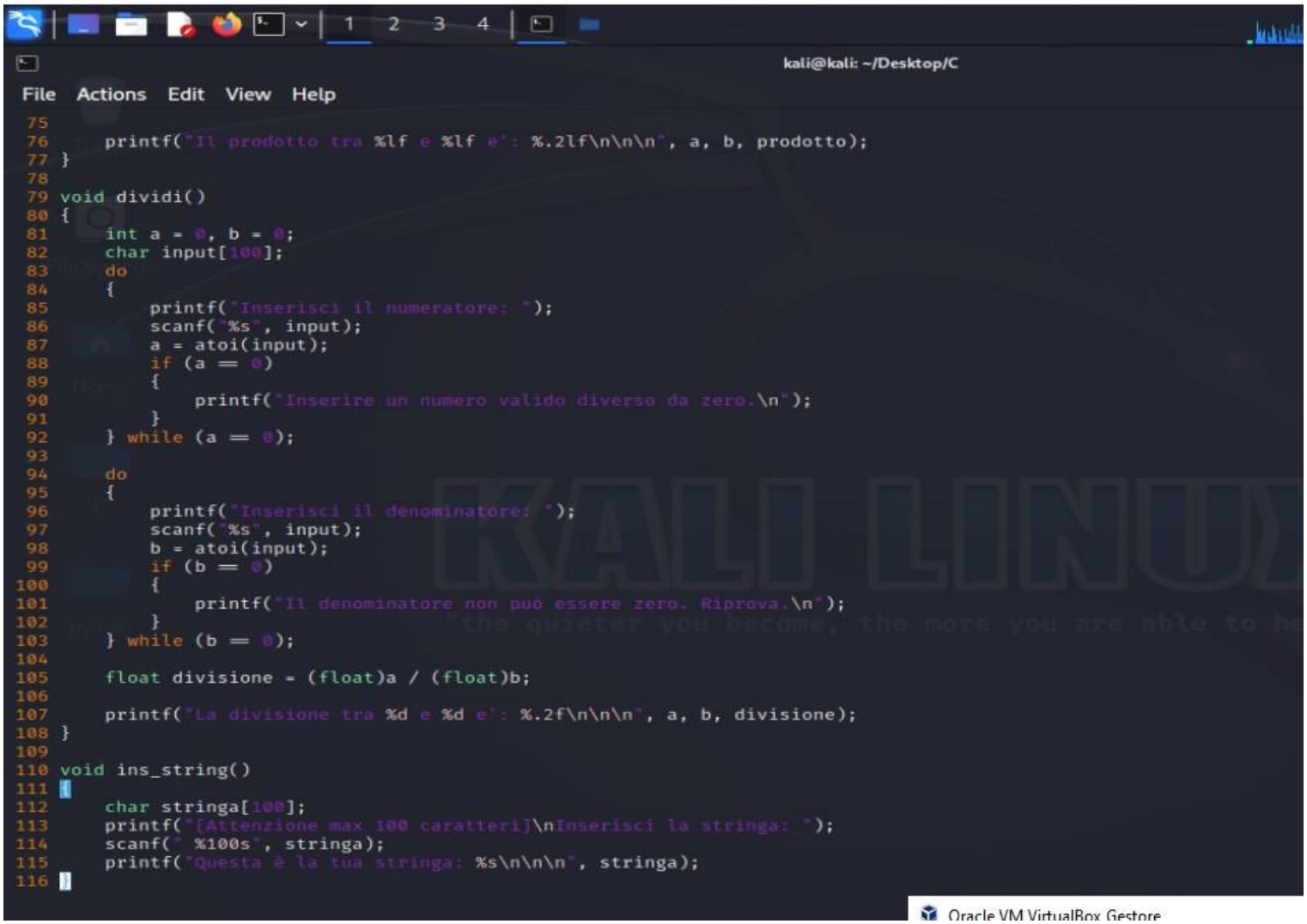
Questa stringa viene poi convertita in un numero in virgola mobile (double) utilizzando atof(input);.

Se l'utente inserisce una lettera o un input non valido, la conversione atof produrrà 0.0.

La funzione verifica se a è uguale a 0.0 e fornisce un messaggio di errore ("Inserire un numero valido.") se l'input non è un numero valido.

La stessa procedura viene ripetuta per l'input del secondo numero (b).

In oltre per dare un criterio estetico e logico i due numeri da moltiplicare vengono chiesti in due "scanf" diversi



-dividi():

L'input dell'utente viene letto come una stringa utilizzando scanf("%s", input);.

Questa stringa viene poi convertita in un intero (int) utilizzando atoi(input);.

Se l'utente inserisce una lettera o un input non valido, la conversione atoi produrrà 0.

La funzione verifica se a è uguale a 0 e fornisce un messaggio di errore ("Inserire un numero valido diverso da zero.") se l'input non è un numero valido.

La stessa procedura viene ripetuta per l'input del denominatore (b).

In entrambe le funzioni (moltiplica e dividi), se l'utente inserisce lettere o input non numerici, il programma fornirà un feedback appropriato e richiederà all'utente di inserire nuovamente un input valido.

Insi_string()

Adesso mostra un messaggio che indica che l'utente può inserire al massimo 100 caratteri: "[Attenzione max 100 caratteri] \n Inserisci la stringa: ".

Utilizza “scanf %100s” per acquisire una stringa di massimo 100 caratteri dall'utente.