

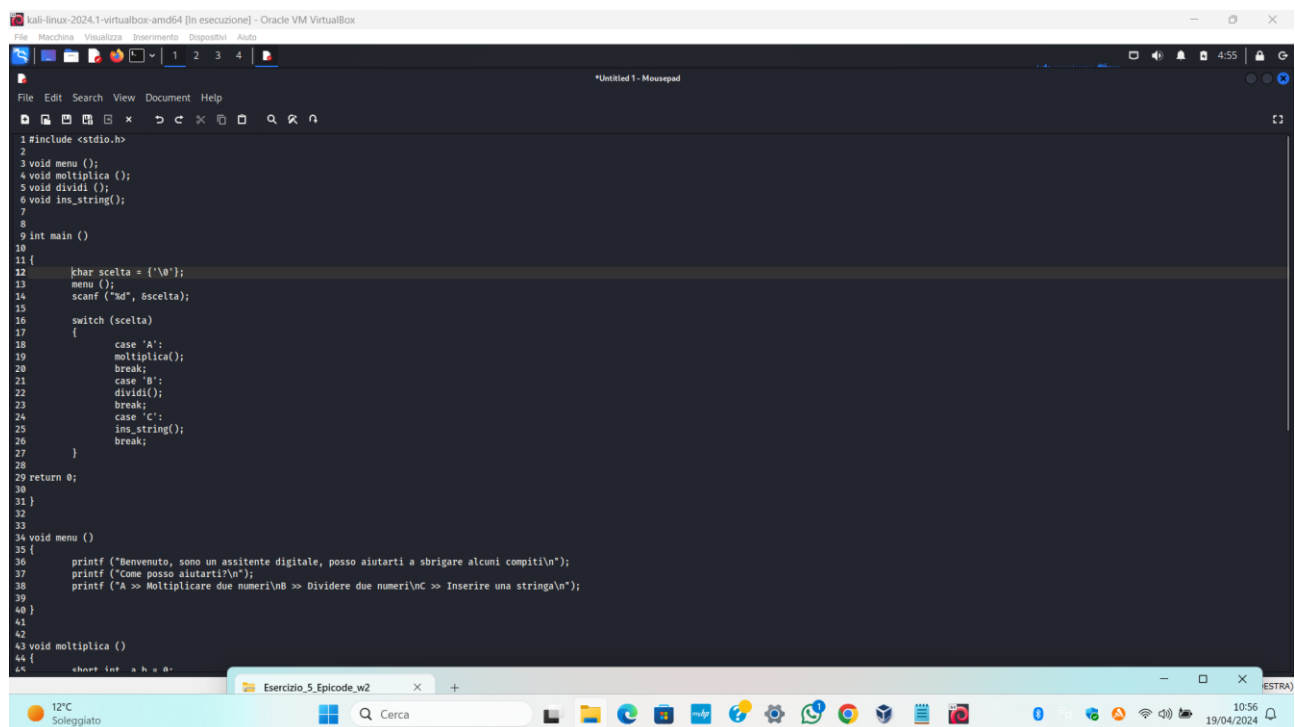
Il programma all'interno dell'esercizio è un assistente virtuale che offre all'utente 3 scelte:

- Moltiplicare 2 numeri
- Dividere 2 numeri
- Inserire una stringa

Tuttavia, sono presenti alcuni errori (o bug) che impediscono la corretta esecuzione del programma.

Nello specifico, i bug sono 4:

Bug 1



```
1 #include <stdio.h>
2
3 void menu ();
4 void moltiplica ();
5 void dividi ();
6 void ins_string();
7
8
9 int main ()
10 {
11     char scelta = {'\0'};
12     menu ();
13     scanf ("%d", &scelta);
14
15     switch (scelta)
16     {
17         case 'A':
18             moltiplica();
19             break;
20         case 'B':
21             dividi();
22             break;
23         case 'C':
24             ins_string();
25             break;
26     }
27
28
29 return 0;
30 }
31
32
33 void menu ()
34 {
35     printf ("Benvenuto, sono un assistente digitale, posso aiutarti a sbrigare alcuni compiti\n");
36     printf ("Come posso aiutarti?\n");
37     printf ("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\n");
38 }
39
40
41 void moltiplica ()
42 {
43     char int  a b = 0;
```

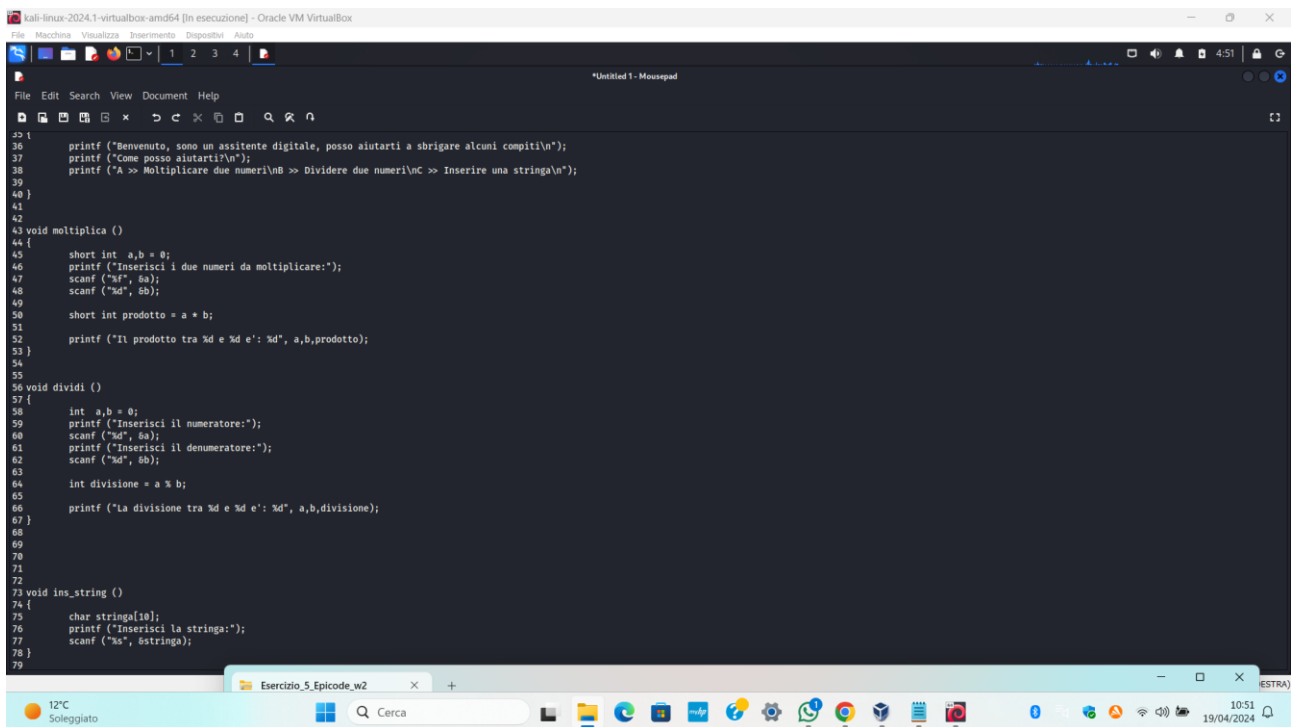
Alla riga 12, nella dichiarazione della variabile scelta, viene utilizzato un inizializzatore di array {}.

Quest'ultimo si utilizza preferibilmente in caso di una sequenza di valori. Tuttavia qui troviamo un solo valore (lo "0"); la scelta dell'array non costituisce un errore a livello di funzionamento del programma ma occupa inutilmente memoria RAM e per questo motivo si può considerare errore a livello di programmazione.

Soluzione Bug 1 :

Alla riga 12, sostituire "char scelta = {'\0'};" "char scelta = '\0';"

Bug 2



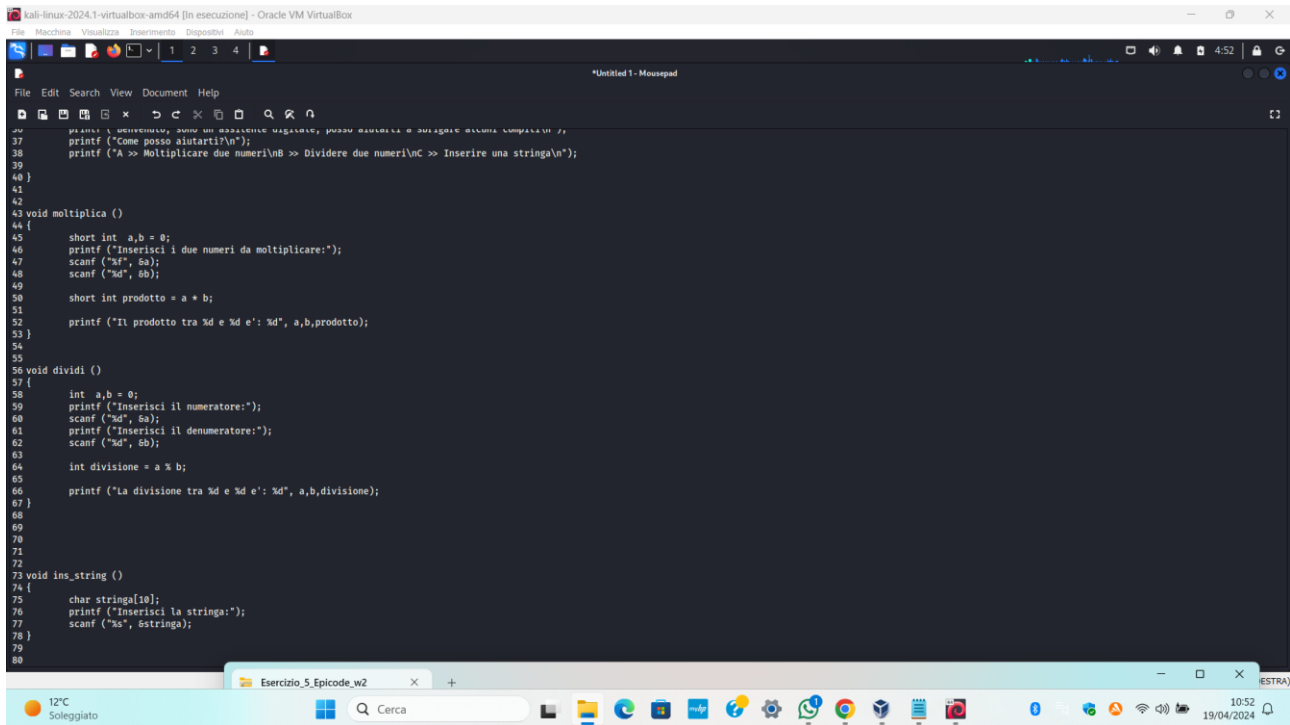
```
25 {
26     printf("Benvenuto, sono un assistente digitale, posso aiutarti a sbrigare alcuni compiti!\n");
27     printf("Come posso aiutarti?\n");
28     printf("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\n");
29 }
30
31
32
33
34
35
36 void multiplica ()
37 {
38     short int a,b = 0;
39     printf("Inserisci i due numeri da moltiplicare:");
40     scanf("%f", &a);
41     scanf("%d", &b);
42
43     short int prodotto = a * b;
44
45     printf("Il prodotto tra %d e %d e': %d", a,b,prodotto);
46 }
47
48 void dividi ()
49 {
50     int a,b = 0;
51     printf("Inserisci il numeratore:");
52     scanf("%d", &a);
53     printf("Inserisci il denominatore:");
54     scanf("%d", &b);
55
56     int divisione = a % b;
57
58     printf("La divisione tra %d e %d e': %d", a,b,divisione);
59 }
60
61
62
63
64
65
66 void ins_string ()
67 {
68     char stringa[10];
69     printf("Inserisci la stringa:");
70     scanf("%s", &stringa);
71 }
72
73
74
75
76
77
78
79
```

Nella funzione “multiplica()” alla riga 45, le variabili “a” e “b” vengono dichiarate come “short int”, ma vengono lette come “float” e “int” attraverso i comandi “scanf (“%f”, &a);” e “scanf (“%d”, &b);” che si trovano alle righe 47 e 48. La non corrispondenza tra il tipo di variabile dichiarato e la variabile utilizzata per la lettura fa sì che i valori non vengano letti correttamente. Infatti, con il comando “scanf (“%f”, &a);” ci aspetteremmo un numero con la virgola in input; anche “scanf (“%d”, &b);” non è adatto per leggere gli input di tipo “SHORT”. Per questo compito è preferibile il comando “scanf (“%hd”, &a);”.

Soluzione Bug 2

Sostituire “scanf (“%f”, &a);” e “scanf (“%d”, &b);” delle rispettive righe 47 e 48 entrambe con “scanf (“%hd”, &a);”

Bug 3



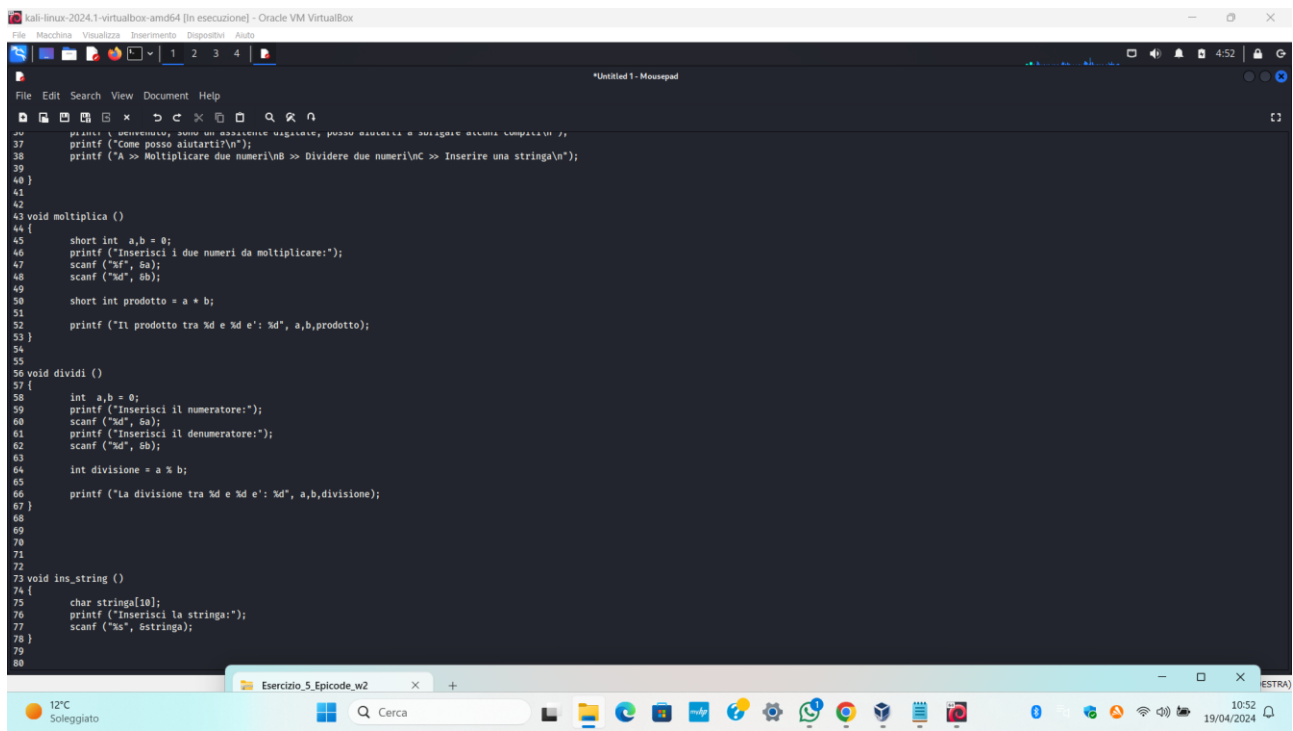
```
37 printf("Come posso aiutarli?\n");
38 printf("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\n");
39 }
40
41
42
43 void moltiplica ()
44 {
45     short int a,b = 0;
46     printf("Inserisci i due numeri da moltiplicare:");
47     scanf("%d", &a);
48     scanf("%d", &b);
49
50     short int prodotto = a * b;
51
52     printf("Il prodotto tra %d e %d e': %d", a,b,prodotto);
53 }
54
55
56 void dividi ()
57 {
58     int a,b = 0;
59     printf("Inserisci il numeratore:");
60     scanf("%d", &a);
61     printf("Inserisci il denominatore:");
62     scanf("%d", &b);
63
64     int divisione = a % b;
65
66     printf("La divisione tra %d e %d e': %d", a,b,divisione);
67 }
68
69
70
71
72
73 void ins_string ()
74 {
75     char stringa[10];
76     printf("Inserisci la stringa:");
77     scanf("%s", &stringa);
78 }
79
80
```

Alla riga 64 viene dichiarata la divisione tra "a" e "b" con "int divisione = a % b", ma in realtà dovremmo avere "int divisione = a / b". Infatti, il valore "%" si utilizza per restituire il resto di una divisione.

Soluzione Bug 3

La riga 64 dovrebbe presentarsi come "int divisione = a / b" invece che "int divisione = a % b"

Bug 4



```
37 printf("Come posso aiutarli?\n");
38 printf("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\n");
39
40 }
41
42
43 void moltiplica ()
44 {
45     short int a,b = 0;
46     printf("Inserisci i due numeri da moltiplicare:");
47     scanf("%d", &a);
48     scanf("%d", &b);
49     short int prodotto = a * b;
50     printf("Il prodotto tra %d e %d e': %d", a,b,prodotto);
51 }
52
53
54
55 void dividi ()
56 {
57     int a,b = 0;
58     printf("Inserisci il numeratore:");
59     scanf("%d", &a);
60     printf("Inserisci il denominatore:");
61     scanf("%d", &b);
62     int divisione = a % b;
63     printf("La divisione tra %d e %d e': %d", a,b,divisione);
64 }
65
66
67
68
69
70
71
72
73 void ins_string ()
74 {
75     char stringa[10];
76     printf("Inserisci la stringa:");
77     scanf("%s", &stringa);
78 }
79
80
```

Nella funzione “ins_string()” alla riga 73, si utilizza il comando “scanf()” alla riga 77 per leggere una stringa, tuttavia il comando “scanf()” è in grado di leggere solo fino al primo spazio. È preferibile utilizzare il comando “fgets()” per leggere una riga intera. Inoltre, alla riga 77 viene richiamata la stringa ma non viene specificata la dimensione massima del file da leggere. Questo potrebbe causare un buffer overflow. Inoltre, in ambito di sicurezza informatica ciò si traduce nella possibilità, per un eventuale attaccante, di inserire piccole parti di un codice malevolo (che potrebbe tradursi in un virus) all’interno dell’operazione di richiamo della stringa, qualora venga impiegata più memoria del necessario.

In più, il valore “%s” andrà sostituito con “%9s” in quanto la lunghezza massima che la stringa può avere è di 10, come indicato nella riga 75.

Soluzione Bug 4

La riga 77 si presenterà come di seguito: `scanf ("%9s", stringa);`

Infine, per quanto riguarda la divisione, non è stata considerata l’eventualità che l’utente inserisca “0” al denominatore. In questi casi dovrebbe esserci un controllo che restituisca la scritta “errore” per impedire all’utente di inserire valori non contemplati. Nel nostro caso, il problema si può risolvere con l’inserimento, (sotto la riga 62) del codice:

```
if (b != 0) {
    printf("La divisione tra %d e %d è: %d\n", a, b, divisione);
} else {
    printf("Impossibile dividere per zero.\n");
}
```

