

## 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

```

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 ("%s", &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     "the quieter you become, the more you are able to hear"
29     return 0;
30 }
31
32
33
34 void menu ()
35 {
36     printf ("Benvenuto, sono un assistente digitale, posso aiutarti a sbrigare alcuni compiti!\n");
37     printf ("Come posso aiutarti?\n");
38     printf ("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\n");
39 }
40
41 void moltiplica ()
42 {
43     :set number
44 }

```

```

1 void moltiplica ()
2 {
3     short int a,b = 0;
4     printf ("Inserisci i due numeri da moltiplicare:");
5     scanf ("%d", &a);
6     scanf ("%d", &b);
7
8     short int prodotto = a * b;
9
10    printf ("Il prodotto tra %d e %d e': %d", a,b,prodotto);
11 }
12
13 void dividi ()
14 {
15     int a,b = 0;
16     printf ("Inserisci il numeratore:");
17     scanf ("%d", &a);
18     printf ("Inserisci il denominatore:");
19     scanf ("%d", &b);
20
21     int divisione = a % b;
22
23     printf ("La divisione tra %d e %d e': %d", a,b,divisione);
24 }
25
26 void ins_string ()
27 {
28     char stringa[10];
29     printf ("Inserisci la stringa:");
30     scanf ("%s", &stringa);
31 }
32

```

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
1 #include <stdio.h>
2 #include <stdlib.h> // Include le funzioni atof e atoi
3
4 void menu();
5 void moltiplica();
6 void dividi();
7 void ins_string();
8
9 int main()
10 {
11     char scelta = {'\0'};
12     do
13     {
14         menu();
15         scanf("%c", &scelta);
16         switch (scelta)
17         {
18             case 'A':
19             case 'a':
20                 moltiplica();
21                 break;
22             case 'B':
23             case 'b':
24                 dividi();
25                 break;
26             case 'C':
27             case 'c':
28                 ins_string();
29                 break;
30             case 'X':
31             case 'x':
32                 break;
33             default:
34                 printf("Scelta non valida\n\n\n");
35         }
36     } while (scelta != 'X' && scelta != 'x');
37     return 0;
38 }
39
40
41 void menu()
42 {
```

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"

```

kali@kali: ~/Desktop/C
File Actions Edit View Help
41 void menu()
42 {
43     printf("Benvenuto, sono un assistente digitale, posso aiutarti a sbrigare alcuni compiti\n");
44     printf("Come posso aiutarti?\n");
45     printf("A >> Moltiplicare due numeri\nB >> Dividere due numeri\nC >> Inserire una stringa\nX >> per uscire\n");
46 }
47
48 void moltiplica()
49 {
50     double a = 0, b = 0;
51     char input[100];
52     do
53     {
54         printf("Inserisci il primo numero da moltiplicare: ");
55         scanf("%s", input);
56         a = atof(input);
57         if (a == 0.0)
58         {
59             printf("Inserire un numero valido.\n");
60         } while (a == 0.0);
61     } while (a == 0.0);
62
63     do
64     {
65         printf("Inserisci il secondo numero da moltiplicare: ");
66         scanf("%s", input);
67         b = atof(input);
68         if (b == 0.0)
69         {
70             printf("Inserire un numero valido.\n");
71         } while (b == 0.0);
72     } while (b == 0.0);
73
74     double prodotto = a * b;
75
76     printf("Il prodotto tra %lf e %lf e': %.2lf\n\n", a, b, prodotto);
77 }
78
79 void dividi()
80 {
81     int a = 0, b = 0;
82     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

```

75
76     printf("Il prodotto tra %lf e %lf e': %.2lf\n\n\n", a, b, prodotto);
77 }
78
79 void dividi()
80 {
81     int a = 0, b = 0;
82     char input[100];
83     do
84     {
85         printf("Inserisci il numeratore: ");
86         scanf("%s", input);
87         a = atoi(input);
88         if (a == 0)
89         {
90             printf("Inserire un numero valido diverso da zero.\n");
91         }
92     } while (a == 0);
93
94     do
95     {
96         printf("Inserisci il denominatore: ");
97         scanf("%s", input);
98         b = atoi(input);
99         if (b == 0)
100        {
101            printf("Il denominatore non può essere zero. Riprova.\n");
102        }
103    } while (b == 0);
104
105    float divisione = (float)a / (float)b;
106
107    printf("La divisione tra %d e %d e': %.2f\n\n\n", a, b, divisione);
108 }
109
110 void ins_string()
111 {
112     char stringa[100];
113     printf("[Attenzione max 100 caratteri]\nInserisci la stringa: ");
114     scanf("%100s", stringa);
115     printf("Questa è la tua stringa: %s\n\n\n", stringa);
116 }

```

#### -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.