LINGUAGGI DI PROGRAMMAZIONE

Nel report di oggi andremo ad analizzare un codice in C svolgendo le seguenti istruzioni:

1. Descrivere il significato funzioni del codice.
2. Individuare codice sorgente e casistiche non standard che il programma non gestisce.
3. Individuare eventuali errori di sintassi/logici.
4. Correggere eventuali errori e proporre una soluzione.

Questo è il codice, adesso procederemo ad analizzarlo in C tramite Kali Linux.

```c
#include <stdio.h>

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


int main ()

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

        switch (scelta)
        {
                case 'A':
                moltiplica();
                break;
                case 'B':
                dividi();
                break;
                case 'C':
                ins_string();
                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 ()
{
        short int  a,b = 0;
        printf ("Inserisci i due numeri da moltiplicare:");
        scanf ("%f", &a);
        scanf ("%d", &b);

        short int prodotto = a * b;

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


void dividi ()
{
        int  a,b = 0;
        printf ("Inserisci il numeratore:");
        scanf ("%d", &a);
        printf ("Inserisci il denumeratore:");
        scanf ("%d", &b);

        int divisione = a % b;

        printf ("La divisione tra %d e %d e': %d", a,b,divisione);
}
```

-#include <stdio.h>: è l'abbreviazione di Standard Input /Output. Per poter usare queste funzioni devo richiamare la libreria stdio. h all'inizio del programma tramite la direttiva #include.

-**V**oid: **è un tipo di dato risultante da una funzione che non restituisce alcun valore al suo chiamante**.
Se usato per l'elenco dei parametri di una funzione, `void` specifica che la funzione non accetta parametri.
Se usato nella dichiarazione di un puntatore, `void` specifica che il puntatore è "universale".

-Int, char e short sono variabili del tipo c, rappresentano diversi tipi di dato, dai grossi numeri ai testi. Int è un numero intero, string è una parola (combinazione tra lettere e/o numeri), char è una lettera singola.

-return 0: se usato nella funzione main determina la fine dell'esecuzione del programma. In questo caso il programma termina restituendo il valore 0 ed indica solo che il programma è terminato correttamente.

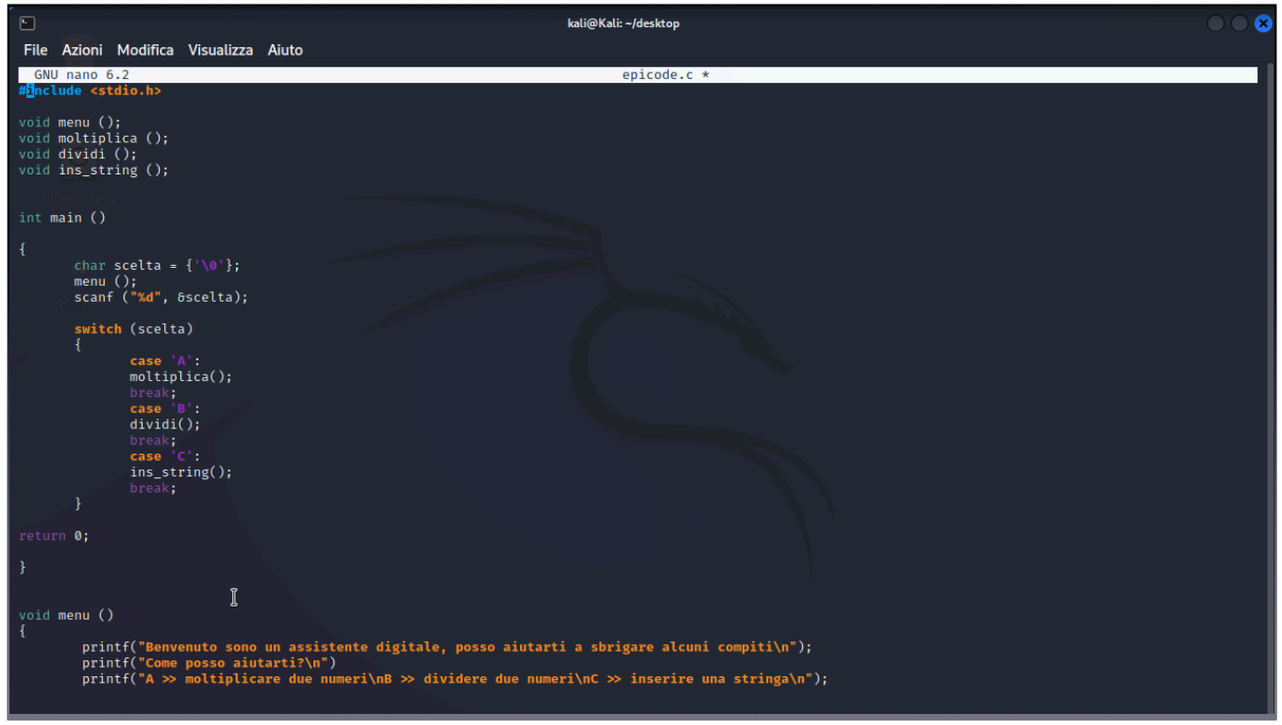
-La funzione printf(): fa parte della Libreria Standard del C e consente all'utente di inviare sul monitor del pc dei numeri / caratteri. Nel caso in cui tale stringa sia composta da solo testo, ovvero solo da caratteri normali, la funzione stampa tutti i caratteri formanti la stringa di formato stessa.

-Scanf: in linguaggio C è una funzione di input, così come getc e getchar. La funzione scanf in C consente di acquisire una sequenza di caratteri (lettere o cifre) dalla tastiera e di memorizzarli all'interno di opportune variabili.

{ }: serve per bloccare e delimitare blocchi di codice che terminano con << ; >>

-Il costrutto switch è un'altra delle istruzioni mediante le quali si implementa il controllo di flusso in C++. Similarmente all'istruzione if, esso consente infatti di eseguire istruzioni differenti a seconda del risultato prodotto dalla valutazione di un'espressione.

-L'istruzione break è stata introdotta per permettere di uscire dai cicli senza aspettare che l'ultimo valore sia stato raggiunto. Quando l'elaboratore la incontra, salta tute le istruzioni del ciclo rimanenti e prosegue con la prima istruzione successiva al ciclo stesso.

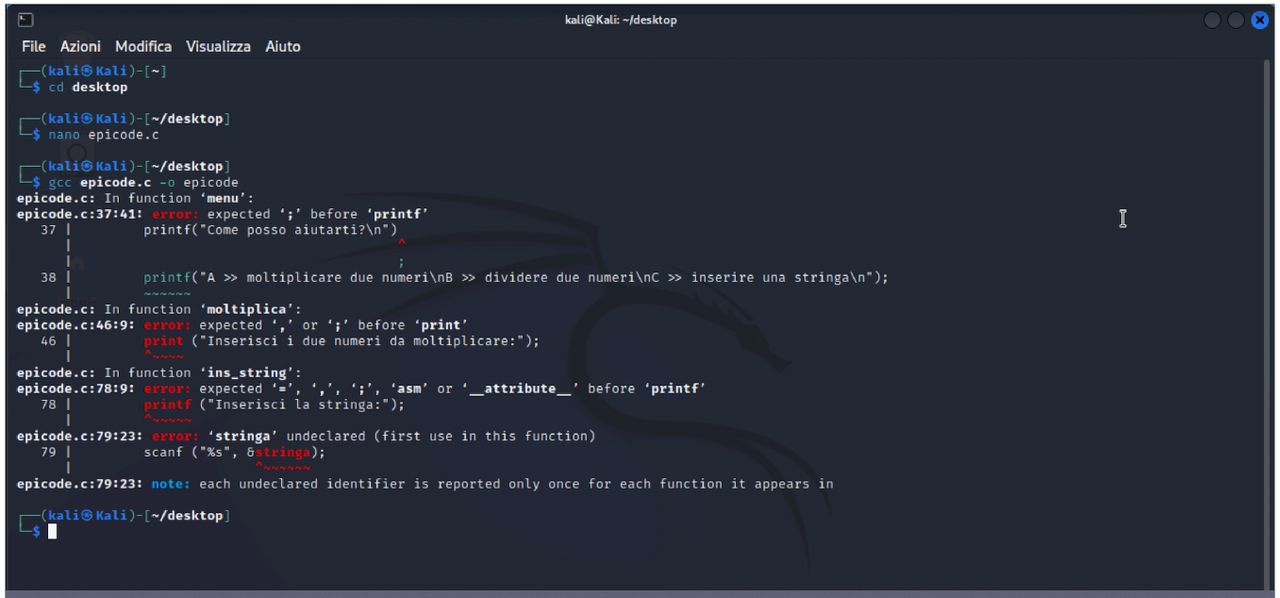

kali@Kali: ~/desktop
File Azioni Modifica Visualizza Aiuto
GNU nano 6.2 epicode.c *
#include <stdio.h>
void menu ();
void moltiplica ();
void dividi ();
void ins_string ();
int main ()
{
char scelta = {'\0'};
menu ();
scanf ("%d", &scelta);
switch (scelta)
{
case 'A':
moltiplica();
break;
case 'B':
dividi();
break;
case 'C':
ins_string();
break;
}
return 0;
}
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\n");

kali@Kali: ~/desktop

File Azioni Modifica Visualizza Aiuto

GNU nano 6.2 epicode.c *

```c
{
void moltiplica ()
{
        short int a,b = 0
        print ("Inserisci i due numeri da moltiplicare:");
        scanf ("%f", &a);
        scanf ("%d", &b);

        short int prodotto = a*b;

        printf ("il prodotto tra %d e %d è: %d", a,b,prodotto);
}



void dividi ()
{
        int  a,b = 0;
        printf("Inserisci il numeratore:");
        scanf ("%d", &a);
        printf("Inserisci il denumeratore:");
        scanf ("%d", &b);

        int divisione = a % b;

        printf("La divisione tra %d e %d e': %d", a,b,divisione);
}




void ins_string ()
{
        char stringa [10]
        printf ("Inserisci la stringa:");
        scanf ("%s", &stringa);
}
```



Ad una attenta analisi il sistema ci indica che ci sono diversi errori, adesso cerchiamo di risolverli in modo da poter avviare il programma correttamente.

kali@Kali: ~

File Azioni Modifica Visualizza Aiuto

GNU nano 6.2 test.c *

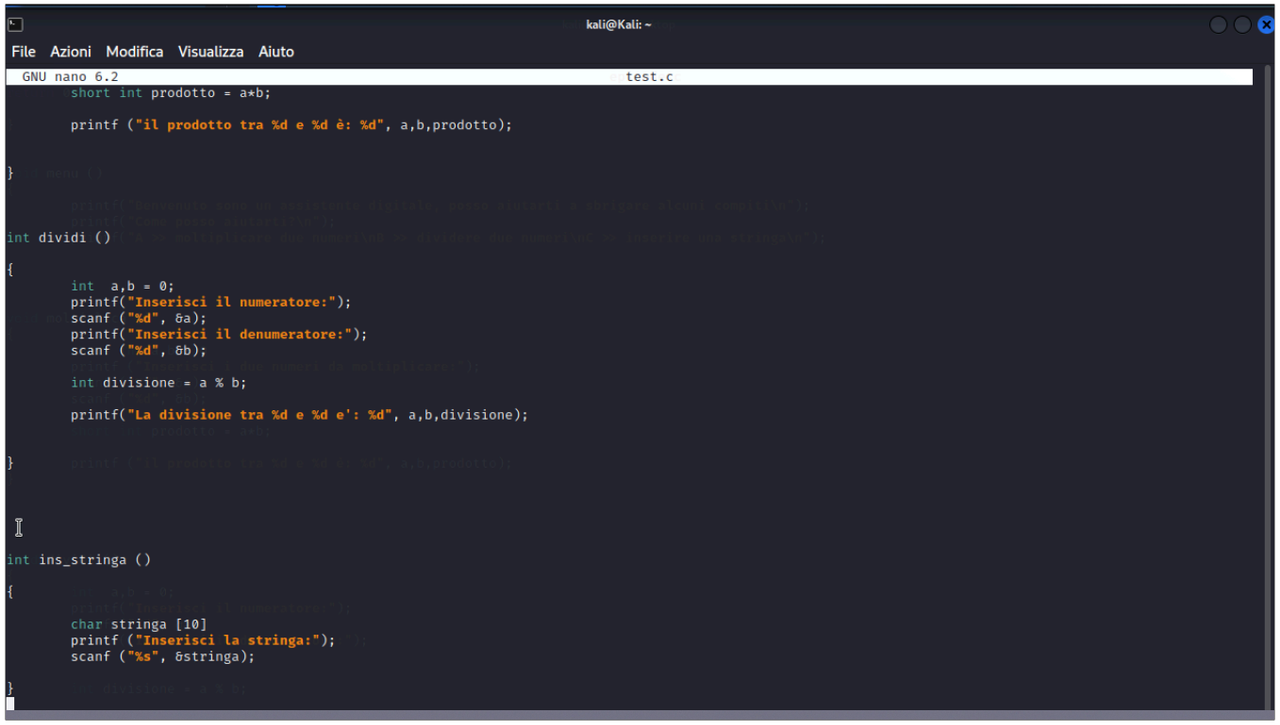
```c
#include <stdio.h>


void menu ()
int elabora (int a; int b);

{

void main (){
int a=20, b=20
elabora(a,b);
(a ⩾ b) ? printd ("a maggiore uguale a b\n" : printf("a minore di b\n");


       scanf("%d", &moltiplicazione);
       scanf("%f", divisione);
       scanf("%d", resto_divisione);


return 0;
}

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\n");

}


int moltiplica ()
{
```

kali@Kali: ~

File Azioni Modifica Visualizza Aiuto

GNU nano 6.2 test.c

```c
int moltiplica ()
{

        short int a,b = 0
        printf ("Inserisci i due numeri da moltiplicare:");
        scanf ("%f", &a);
        scanf ("%d", &b);

        short int prodotto = a*b;

        printf ("il prodotto tra %d e %d è: %d", a,b,prodotto);


}


int dividi ()

{
        int  a,b = 0;
        printf("Inserisci il numeratore:");
        scanf ("%d", &a);
        printf("Inserisci il denumeratore:");
        scanf ("%d", &b);

        int divisione = a % b;

        printf("La divisione tra %d e %d e': %d", a,b,divisione);


}


int ins_stringa ()
```


kali@Kali: ~
File Azioni Modifica Visualizza Aiuto
GNU nano 6.2 test.c
short int prodotto = a*b;
printf ("il prodotto tra %d e %d è: %d", a,b,prodotto);
}
int dividi ()
{
int a,b = 0;
printf("Inserisci il numeratore:");
scanf ("%d", &a);
printf("Inserisci il denumeratore:");
scanf ("%d", &b);
int divisione = a % b;
printf("La divisione tra %d e %d e': %d", a,b,divisione);
}
int ins_stringa ()
{
char stringa [10]
printf ("Inserisci la stringa:");
scanf ("%s", &stringa);
}

Conclusione

Ho provato a modificare vari parametri del codice per riuscire a riprodurre le varie operazioni e ridurre eventuali errori, ciò nonostante non sono riuscito a far partire il programma…