NETRAIDERS

Matteo Leoni

BUG HUNTING

Si analizza il programma dato. Vedendo il codice salta subito all'occhio che si tratta di un piccolo programma che permette di fare una divisione tra due numeri, una moltiplicazione tra due numeri e l'inserimento di una stringa. Andiamo a vedere più nel dettaglio gli eventuali errori. Nella relazione metterò due screenshot in comparazione, quello di sinistra con codice errato, mentre lo screenshot di destra sarà col codice revisionato e corretto.
Partiamo con gli errori di sintassi/logici.

ERRORI SINTASSI/LOGICI

```
int main ()

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

```
int main ()

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

Nella funzione main(), lo scanf della variabile scelta non deve essere %d, in quanto si riferisce a numeri interi, bensì %c, in quanto la variabile scelta sarà di tipo char, quindi un carattere.

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

        int prodotto = a * b;

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

Nella funzione moltiplica(), dopo il primo dei due scanf, c'è %f, ed è sbagliato, in quanto questa notazione è riservata alle variabili di tipo float, mentre qui siamo in presenza di variabili di tipo int, quindi abbiamo bisogno di %d, come giustamente scritto nella riga successiva.

Non è un errore sintattico/logico, ma abbiamo comunque pensato di aumentare il range di utilizzo di questa funzione, togliendo lo short davanti all'int sia per i valori a e b, sia per la variabile prodotto.

```
int divisione = a % b;
```

```
float divisione = (float)a/(float)b;
```

In questo caso si cambia il segno della percentuale (%), col segno appropriato per la divisione (/), in quanto il segno di percentuale dà il resto della divisione, e non il risultato effettivo.

Anche in questo caso viene modificata la variabile divisione da int a float, per dare un risultato matematicamente più corretto nel caso in cui non sia una divisione a resto 0, e quindi con valori decimali. Questo lo si fa per evitare un possibile errore logico del programma, dando un risultato non perfetto.

```
void menu ()
{
        printf ("Benvenuto, sono un assitente digitale, posso aiutarti a sbrigare alcuni compiti\n");
```

In questo caso non è un errore riferito al codice, ma un semplice errore di battitura, con la parola Assistente scritta male, che potrebbe far storcere il naso all'utente finale.

```
        printf ("Inserisci il denumeratore:");
```

Altro errore di scrittura, in quanto non esiste la parola denumeratore, bensì denominatore.

IMPLEMENTAZIONE PROGRAMMA

Per un migliore utilizzo, si è pensato di implementare il programma, effettuando le seguenti modifiche:

```
int main ()

{
        char scelta = {'\0'};
        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;
                default:
                        printf("Scelta sbagliata, chiusura in corso,\nsi prega di riavviare scegliendo tra le lettere disponibili!\n");
        }

return 0;

}
```

E' stata inserita una casistica di default nello switch della funzione main(). In questo caso se l'utente sbaglia a inserire il carattere richiesto, il programma ci dice che si è sbagliato a digitare il carattere, avvisa della chiusura del programma, e prega di riavviarlo scegliendo tra le opzioni corrette.

Inoltre, abbiamo deciso di includere nei Case anche i caratteri minuscoli, per evitare problemi in cui si digiti il carattere giusto, ma non in forma maiuscola.

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

        int prodotto = a * b;

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

L'implementazione della funzione moltiplica() è stata già spiegata precedentemente, togliendo lo short davanti all'int per un range più ampio di utilizzo, in quanto short int è formato da due byte, quindi 65536 valori, mentre int è formato da 4 byte, ovvero 4294967296 valori.

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

        if(b==0){
                printf("Non si può dividere per 0!\n");
}
        else{
                float divisione = (float)a/(float)b;
                printf ("La divisione tra %d e %d e' : %.2f\n", a, b, divisione);

    }

}
```

In questo caso, nel programma viene implementato anche la casistica nella quale un qualsiasi numero venga diviso per 0. Nel caso in cui l'utente inserisse un denominatore pari a 0, il programma lo avvisa col printf("Non si può dividere per 0!"), evitando quindi un errore di runtime.

```c
void ins_string ()
{
        char stringa[100];
        printf ("Inserisci la stringa:");
        scanf (" %99[^\n]s", stringa);
        printf("Hai inserito: %s\n", stringa);
}
```

Per quanto riguarda l'ultima funzione, oltre a quello che si è già detto precedentemente, si è deciso di aumentare il valore della stringa, per aumentare le funzionalità della stessa, al fine di avere la possibilità di inserire più caratteri. Inoltre, si è deciso di inserire anche quella notazione particolare nello scanf, in quanto con un semplice %s, nel caso in cui ci troviamo davanti ad una stringa con più parole separate da spazi, il programma legge solo la prima parola. In questo modo, con scanf(" %99[^\n]s", stringa) il programma legge 100 caratteri, compresi gli spazi.