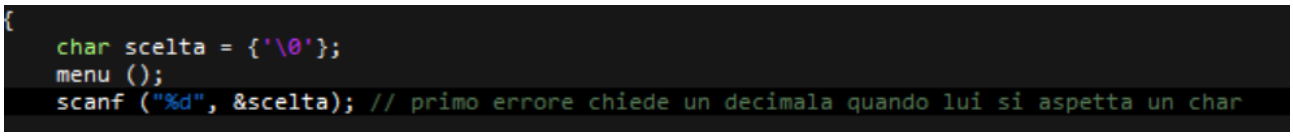
# Osservazione e ottimizzazione del codice

Il compito di oggi era quello di Osservare il codice e capire cosa facesse, correggere eventuali errori di sintassi/logici, fatto questo bisognava individuare casi che il codice non gestiva ed ottimizzarlo.

## 1) Osservazione e comprensione del codice senza esecuzione.

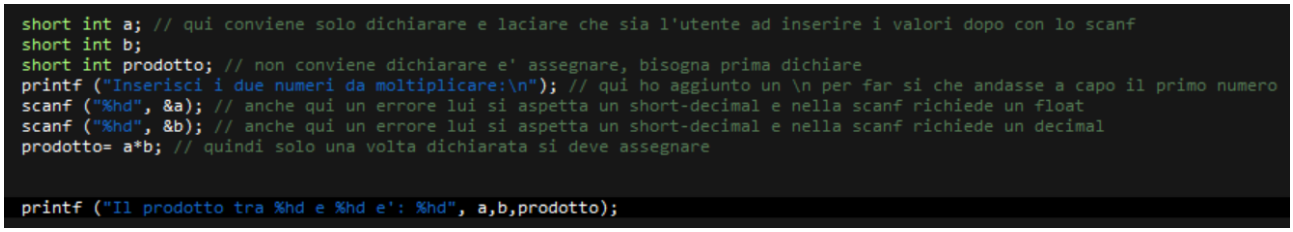
Il codice in questione è molto semplicemente un menù che permette all'utente tramite scelta fatta tramite l'invio di un carattere lo svolgimento di tre operazioni basiche: (A) per svolgere una moltiplicazione tra due numeri, (B) per dividere due numeri, (C) per inserire una stringa. Queste operazioni vengono gestite da uno switch che si trova nel main il quale prende prima il carattere "scelta", che sarà poi passato allo switch che in base al carattere inserito poi andrà a chiamare la funzione per procedere poi con l'operazione da fare.

## 2)Errori di sintassi/logici



```
{
    char scelta = {'\0'};
    menu ();
    scanf ("%d", &scelta); // primo errore chiede un decimala quando lui si aspetta un char
```

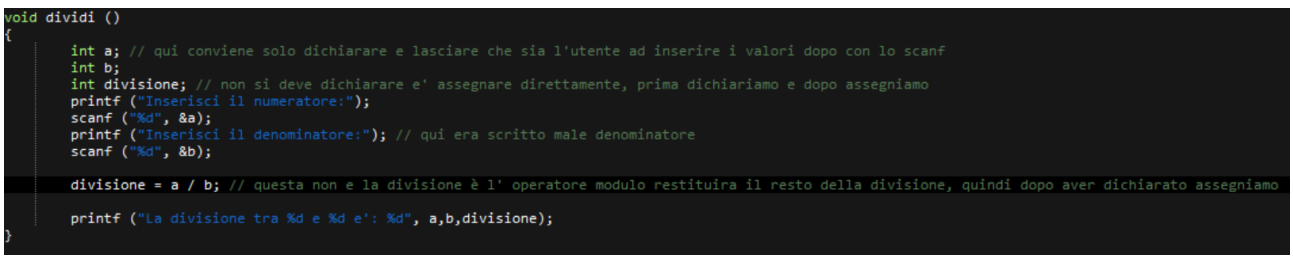
- Per risolvere questo errore dobbiamo inserire nello scanf **("%c",&scelta);**



```
short int a; // qui conviene solo dichiarare e laciare che sia l'utente ad inserire i valori dopo con lo scanf
short int b;
short int prodotto; // non conviene dichiarare e' assegnare, bisogna prima dichiare
printf ("Inserisci i due numeri da moltiplicare:\n"); // qui ho aggiunto un \n per far si che andasse a capo il primo numero
scanf ("%hd", &a); // anche qui un errore lui si aspetta un short-decimal e nella scanf richiede un float
scanf ("%hd", &b); // anche qui un errore lui si aspetta un short-decimal e nella scanf richiede un decimal
prodotto= a*b; // quindi solo una volta dichiarata si deve assegnare


printf ("Il prodotto tra %hd e %hd e': %hd", a,b,prodotto);
```

- Il primo errore qui era che si dichiaravano a, b e gli si assegnavano subito i valori.
- Secondo errore nella scanf richiedeva parametri diversi dai tipi delle variabili.
- Terzo errore prodotto veniva dichiarato è assegnato subito alla moltiplicazione a*b, quindi prima lo dichiariamo e solo dopo lo assegniamo a*b.



```
void dividi ()
{
        int a; // qui conviene solo dichiarare e lasciare che sia l'utente ad inserire i valori dopo con lo scanf
        int b;
        int divisione; // non si deve dichiarare e' assegnare direttamente, prima dichiariamo e dopo assegniamo
        printf ("Inserisci il numeratore:");
        scanf ("%d", &a);
        printf ("Inserisci il denominatore:"); // qui era scritto male denominatore
        scanf ("%d", &b);

        divisione = a / b; // questa non e la divisione è l' operatore modulo restituira il resto della divisione, quindi dopo aver dichiarato assegniamo

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

- Il primo errore qui era che si dichiaravano a, b e gli si assegnavano subito i valori.
- Secondo errore denominatore nella printf era scritto in modo sbagliato.
- Terzo errore divisione veniva sia dichiarato che assegnato, quindi prima lo dichiariamo sopra.
- Quarto errore **'%'** non è l'operatore della divisione ma è l'operatore modulo, restituirà il resto della divisione. L' operatore per la divisione è **'/'.**

```c
void ins_string ()
{
    char stringa[10];
        printf ("Inserisci la stringa:");
        scanf ("%s", &stringa);
        printf("la tua stinga e' %s",stringa); // qui non ci sono errori di sintassi ma facendo cosi mostriamo la stringa che abbiamo scritto.
}
```

- Non ci sono errori, ma cosi facendo mostriamo a schermo la stringa appena scritta.

## 3) **Individuare casistiche non standard e proporre soluzioni per queste.**

```c
scanf ("%c", &scelta); // primo caso qui inserndo una cosa diversa da A,B,C il programma termina.
```

```c
while(scelta!='A' && scelta!='B' && scelta!='C' ){
printf("ATTENZIONE INSERSCI UN VALORE ESATTO!!!\n"); // viene mostrata a video questa scritta in caso di errore nella scelta
fflush(stdin); // pulisce il buffer del input
menu(); // ripropone la funzione menu con le scelte
scanf("%c",&scelta); // per rinserire la scelta
```

- Con questo ciclo while in caso l'utente inserisca un valore che non si (A),(B),(C) gli verrà mostrato un codice di errore e il menù.

```c
do{
printf ("Inserisci i due numeri da moltiplicare:\n");
fflush(stdin);
prova_a=scanf ("%hd", &a); // facendo cosi assegniamo a prova_a il valore del funzione scanf di a quindi in caso di insrimento di una lettera il risultato sara 1
prova_b=scanf ("%hd", &b); // stessa cosa per b
if(prova_a==0 || prova_b==0) // qui controlliamo se il valore della funzione scanf sia 0 se invece fosse 1 procede con la moltiplicazione
{
    printf ("ATTENZIONE SONO CONSENTITI SOLO NUMERI INTERI!!!\n");   // messaggio per l'errore
}
else if(a>180 || b>180 || a<-180 || b<-180){ // con questo controllo faccio in modo che l'utente possa moltiplicare al massimo i numeri consentiti dallo short int
printf ("ATTENZIONE PUOI MOLTIPLICARE AL MASSIMO 180*180!!!\n");
}
else{
    prodotto= a*b;
    printf ("Il prodotto tra %hd e %hd e': %hd", a,b,prodotto);
    j++; //contatore per uscire dal while in caso lui inserisca numeri troppo grandi per lo short int
}
ile(prova_a==0 || prova_b==0 || j==0); // qui in caso di valore 1 della funzione scanf uscira dal while oppure in caso di 0 riparte il ciclo o se j e 1
```

- Per la moltiplicazione c'erano varie falle
- Quando si metteva una lettera o un carattere il programma si chiudeva senza permettere la moltiplicazione.
- Adesso con prova_a e prova_b in cui ci va il risultato della funzione scanf controlliamo con l'if, in caso sia una lettera o un carattere viene mostrato a schermo un errore, la condizione del while diventa quindi falsa e riparte il do, se invece il risultato della funzione scanf sia 0 quindi un numero procede con il controllo del range.
- Un altro problema era il range della moltiplicazione. l'else if gestisce questo problema del range della moltiplicazione nel caso dello short int il limite massimo è 180*180(o -180) quindi nel caso l'utente inserisca un valore maggiore di 180 gli verrà richiesto il valore.

```c
void dividi ()
{
        int a;
        int prova_a;
        int prova_b;
        int b;
        int divisione;
        do{
    fflush(stdin);
    printf("inserisci numeratore\n");
    prova_a=scanf ("%hd", &a);
    printf("inserisci denominatore\n");
    prova_b=scanf ("%hd", &b);
    if(prova_a==0 || prova_b==0)
    {
        printf ("ATTENZIONE SONO CONSENTITI SOLO NUMERI INTERI!!!\n");
    }
    else{
        divisione = a / b;
        printf ("La divisione tra %hd e %hd e': %hd", a,b,divisione);
        }
        }while(prova_a==0 || prova_b==0);
}
```

- Anche qui il problema si presentava quando l'utente inseriva una cosa diversa da un numero, ho riusato la stessa formattazione del do while della moltiplicazione
- Anche qui si presenta il problema del range che in questo caso con l'int intero non sono riuscito a gestire, avrei potuto anche in questo caso impostare un range come nella moltiplicazione con un altro else if e un contatore nel while.

```c
void ins_string ()
{
    char stringa[10];
        printf ("Inserisci la stringa:");
        fflush(stdin);
        fgets(stringa,10,stdin); // facendo questo anche in caso l' utente scriva stringa piu' grande di quella possibile lui la tronca
        printf("la tua stinga e' %s",stringa);
}
```

- In questo caso l'unico problema che si presentava era quello del range, quindi quando l'utente inseriva una stringa più grande di quella dovuto c'era un errore.
- Ho risolto usando fgets che anche in caso in cui l'utente inserisca una stringa più grande di quella concessa, fgets provvederà a tagliarla e renderla della dimensione impostata.