

PROGETTO UNIT 1.2

MODULO 5

Capire cosa fa il programma senza eseguirlo

Nella funzione `void menu` del programma troviamo la sua descrizione: un assistente digitale che aiuterà nello svolgimento di alcuni compiti. Questi compiti sono divisi in tre opzioni in base all'input dell'utente: moltiplicare due numeri (carattere 'A'), dividere due numeri (carattere 'B') o inserire una stringa (carattere 'C'). In seguito, viene invocata la funzione corrispondente.

Nella direttiva del processore troviamo la libreria standard del linguaggio C `<stdio.h>` (standard input-output header) che contiene definizioni di macro, costanti e dichiarazioni di funzioni e tipi usati per le varie operazioni di input/output.

In seguito, vengono dichiarate i `prototipi` della funzione fatte prima della funzione `int main`, ovvero la funzione principale di un programma in C, dalla quale il processore parte a leggere le istruzioni una volta che il programma è in esecuzione e ne restituirà un valore in base al suo tipo.

Oltre alla funzione principale, identificata con il tipo `int`, un tipo di dato rappresentante un numero intero, troviamo le varie funzioni di tipo `void`, un tipo di dato risultante da una funzione che non restituisce alcun valore al suo chiamante.

La funzione `void moltiplica` chiede all'utente di inserire i due numeri da moltiplicare, esegue il prodotto dei due numeri e ne restituisce il risultato.

La funzione `void dividi` chiede all'utente di inserire il numeratore, successivamente il denominatore ed esegue il modulo della divisione.

La funzione `void ins_string` chiede all'utente di inserire una stringa di massimo 9 caratteri, stampandone il risultato.

Nella funzione `int main` ritroviamo la struttura del programma: assegna il valore `NULL` alla variabile `char`, invoca la funzione `void menu` ed esegue la funzione `scanf` acquisendo una sequenza di caratteri (lettere o cifre) dalla tastiera e memorizzandoli all'interno di opportune variabili.

Successivamente esegue l'istruzione condizionale `switch` (un costrutto a selezione multipla) che si basa sul confronto tra il risultato di un'espressione e un insieme di valori costanti.

Nel caso 'A' viene invocata la funzione void moltiplica, nel caso 'B' viene invocata la funzione void dividi e nel caso 'C' viene invocata la funzione void ins_string. Dopo ogni invocazione, viene terminata l'esecuzione dell'istruzione switch tramite l'istruzione break, terminando l'esecuzione dell'istruzione contenitore.

La funzione int main termina con l'istruzione return, che viene utilizzata per restituire un valore al chiamante, compatibile con il valore della funzione.

Individuare casistiche non standard che il programma non gestisce

Nelle casistiche non standard rientra l'uso del qualificatore di tipo short al tipo int (funzione void moltiplica), implementazione del C++. Anche se il linguaggio C++ provvede dei meccanismi per renderlo compatibile nello stesso programma (con compilatori compatibili), mischiare i linguaggi porta a problemi di portabilità.

Il primo step per far sì che siano compatibili richiede, oltre ad un compilatore compatibile, il tipo di dati come int, float o pointer espressi nello stesso modo.

Errori di sintassi/logici

Un errore di sintassi in un programma è causato dall'utilizzo di una sintassi errata o non contemplata dal linguaggio di programmazione in uso.

Un errore logico è un errore classificato come errore di runtime che può causare un programma che produce un output errato o l'arresto anomalo durante l'esecuzione.

```
kali@kali: ~/Desktop/Report
File Actions Edit View Help
GNU nano 7.2 wrongpro.c *
#include <stdio.h>
void menu ();
void moltiplica ();
void dividi ();
void ins_string();

int main ()
{
    char scelta = {'\0'};
    menu ();
    scanf ("%d", &scelta); //specificatore formato errato

    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");
}
```

Nella prima parte del programma lo specificatore di formato della funzione `scanf` è errato, `%d` rappresenta variabili di tipo `INT`, mentre la variabile alla quale si riferisce è di tipo `CHAR` (`%c`).

```
kali@kali: ~/Desktop/Report
File Actions Edit View Help
GNU nano 7.2 wrongpro.c *
void moltiplica ()
{
    short int a,b = 0; //qualificatore tipo errato
    printf ("Inserisci i due numeri da moltiplicare:");
    scanf ("%f", &a); //specificatore formato errato
    scanf ("%d", &b);

    short int prodotto = a * b; //qualificatore tipo errato

    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 denominatore:");
    scanf ("%d", &b);

    int divisione = a % b; //operatore aritmetico errato

    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); //s non necessario
}

}
```

Nella seconda parte del programma lo specificatore di formato della prima funzione `scanf` nella funzione void moltiplica è errato. Correggendo lo `short int` con `int`, lo specificatore di formato di entrambe le `scanf` deve essere `%d` (tipo `INT`) e non `%f` (`FLOAT`).

Nella funzione void dividi l'operatore aritmetico utilizzato è errato: `%` (modulo) al posto di `/` (divisione).

Nella funzione void ins_string la funzione `scanf` presenta lo specificatore di formato come `%s` (array di char), esso corrisponde con l'indirizzo di partenza dell'array/stringa, per cui `&` (indirizzo di memoria in cui si trova la variabile) non è necessario.

Soluzione dei problemi

La soluzione consiste nel correggere i vari errori presenti nelle funzioni `scanf`, correggere il qualificatore di tipo `short int` in tipo `int` nella funzione void moltiplica e correggere l'operatore aritmetico nella funzione void dividi. Inoltre, dovrebbe essere implementata una logica per la funzione void ins_string, che non presenta elaborazione nel programma.



```
GNU nano 7.2 wrongpro.c *
#include <stdio.h>
#include <string.h>

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

int main () {
    char scelta = '\0';
    menu();
    scafnf("%c", &scelta);

    switch (scelta) {
        case 'A':
            moltiplica();
            break;
        case 'B':
            dividi();
            break;
        case 'C':
            ins_string();
            break;
        default:
            printf("Scelta non consentita\n"); }

    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 \t\t B >> Dividere due numeri \t\t C >> Inserire una stringa \n\n"); }

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

    int prodotto = a * b;

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

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

    int divisione = a / b;
    int resto = a % b;

    PG Help      PC Write Out  PW Where Is    CX Cut         PT Execute    PC Location    W-U Undo
    EX Exit      CR Read File  PR Replace    CU Paste      PJ Justify    ^_ Go To Line  W-E Redo
```

```
kali@kali: ~/Desktop/Report
File Actions Edit View Help
GNU nano 7.2 wrongpro.c *
char scelta = '\0';
menu();
scanf("%c", &scelta);

switch (scelta) {
case 'A':
    multiplica();
    break;
case 'B':
    dividi();
    break;
case 'C':
    ins_string();
    break;
default:
    printf("Scelta non consentita\n"); }

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 \t\t B >> Dividere due numeri \t\t C >> Inserire una stringa \n\n"); }

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

    int prodotto = a * b;

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

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

    int divisione = a / b;
    int resto = a % b;

    printf("La divisione tra %d e %d e': %d con resto di %d\n", a,b,divisione,resto); }

int ins_string () {
    char stringa[50];
    printf ("Inserisci la stringa: \n");
    fgets(stringa, 50, stdin);
    printf("Hai inserito una stringa lunga %ld caratteri\n",strlen(stringa)); }

Help Exit Write Out Read File Where Is Replace Cut Paste Execute Justify Location Go To Line Undo Redo
```

Dopo le correzioni necessarie ho incluso la libreria `<string.h>` per migliorare la funzione `ins_string` dal momento che non veniva elaborata nel programma.

In questo caso ho deciso di utilizzarla per analizzare una stringa che verrà immagazzinata all'interno di array di caratteri e calcolarne la lunghezza con la funzione `strlen`. In quanto `strlen` restituisce un `long unsigned int`, ho utilizzato lo specificatore `%ld`. Inoltre, per ricevere gli input dei caratteri scritti dall'utente, ho utilizzato la funzione `fgets` per evitare possibili buffer overflow.

Ho aggiunto un caso di `default` nello switch, anche se non necessario, per gestire eventuali input differenti da quelli previsti.

Nella funzione `void dividi` ho aggiunto anche il modulo, oltre alla corretta divisione, per calcolare il resto.