Report Gioco Quiz in C

```
┌──(kali㉿kali)-[~/Desktop]
└─$ emacs esercizioC.c
```

Creazione di un file emacs per scrivere il codice.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ nano esecizioC.c
```

Modifica del file dove vado a scrivere il codice C per il gioco del Quiz

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Struttura per rappresentare una domanda
typedef struct {
    char testo[100];
    char risposte[3][50];
    int rispostaCorretta;
} Domanda;

// Funzione per mostrare il menu iniziale e ottenere la scelta dell'utente
char mostraMenuIniziale() {
    char scelta;
    printf("Benvenuto al quiz di Epicode!\n");
    printf("A) Iniziare una nuova partita\n");
    printf("B) Uscire dal gioco\n");
    printf("Scegli: ");
    scanf(" %c", &scelta);
    return scelta;
}

// Funzione per gestire una nuova partita
void nuovaPartita() {
    int punteggio = 0;
    char nome[50];

    // Ottieni il nome del giocatore
    printf("Inserisci il tuo nome: ");
    scanf("%s", nome);

    // Array di domande
    Domanda domande[5] = {

        {"Con solo device di rete livello 2, cosa possono separare i domini di prodcast?",{"Valn","Wan","Pan"}, 0},
        {"Da quanti bit è composto un IPv4?",{"28","64","32"}, 2},
        {"Quale commando si usa da terminale Windows per controllare le impostazioni di rete ?",{"ifconfig","ipconfig","ipconf"}, 1},
        {"Come sono anche detti i paccchetti di data link?",{"Datagrammi","Frame","Bit"}, 2},
        {"Quale dei seguenti protocolli, sono protocolli di trasporto?",{"UDP MAC","TCP IP","UDP TCP"}, 2}

  };
```

```c
    // Loop attraverso le domande
    for (int i = 0; i < 5; i++) {
        printf("\nDomanda %d: %s\n", i + 1, domande[i].testo);
        printf("A) %s\nB) %s\nC) %s\n", domande[i].risposte[0], domande[i].risposte[1], domande[i].risposte[2]);

        char rispostaUtente;
        printf("Inserisci la tua risposta (A, B o C): ");
        scanf(" %c", &rispostaUtente);

        // Valuta la risposta
        if (rispostaUtente == 'A' || rispostaUtente == 'B' || rispostaUtente == 'C') {
            if (rispostaUtente - 'A' == domande[i].rispostaCorretta) {
                printf("Corretto!\n");
                punteggio++;
            } else {
                printf("Sbagliato. La risposta corretta è %c)\n", 'A' + domande[i].rispostaCorretta);
            }
        } else {
            printf("Risposta non valida. Per favore, inserisci A, B o C.\n");
            i--;  // Decrementa l'indice per ripetere la stessa domanda
        }
    }

    // Stampa il punteggio finale
    printf("\n%s, il tuo punteggio finale è: %d/5\n", nome, punteggio);
}

int main() {
    char scelta;

    do {
        scelta = mostraMenuIniziale();

        switch (scelta) {
            case 'A':
                nuovaPartita();
                break;
            case 'B':
                printf("Grazie per aver giocato. Arrivederci!\n");
                break;
            default:
                printf("Scelta non valida. Per favore, inserisci A o B.\n");
        }
    } while (scelta != 'B');

    return 0;
}
```

Dopo la modifica del file, il codice che scrivo al suo interno attraverso il commando nano nomefile, e salvarlo.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ gcc -g esecizioC.c -o m

┌──(kali㉿kali)-[~/Desktop]
└─$
```

Con Il comando `gcc -g esercizioC.c -o m` è utilizzato per compilare un programma in C con il compilatore GCC (GNU Compiler Collection). Di seguito una breve spiegazione:

1. **gcc:** Compilatore C della GNU Compiler Collection.

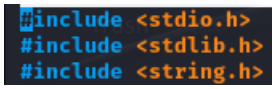
2. **-g:** Opzione per includere informazioni di debug nel file eseguibile, utili per il debugging con GDB.

3. **esercizioC.c:** Nome del file sorgente in C contenente il codice del programma.

4. **-o m:** Opzione per specificare il nome del file eseguibile di output, che sarà "m".

In sintesi, il comando compila "esercizioC.c" e genera l'eseguibile chiamato "m", includendo informazioni di debug. L'eseguibile può essere eseguito sul sistema.

Con il commando ‘./m’ eseguo il mio codice nel terminale.



Questo è il blocco delle intestazioni (**#include**). Include alcune librerie standard di C che il programma utilizzerà. In questo caso, **stdio.h** per l'input/output standard, **stdlib.h** per funzioni di utilità standard, e **string.h** per operazioni sulle stringhe.

```c
// Struttura per rappresentare una domanda
typedef struct {
    char testo[100];
    char risposte[3][50];
    int rispostaCorretta;
} Domanda;
```

Questo è il blocco della definizione della struttura (**typedef struct**). Qui viene definita una struttura di dati chiamata **Domanda**, che rappresenta una singola domanda del gioco. La struttura contiene il testo della domanda, un array di risposte (3 opzioni), e l'indice della risposta corretta.

```c
// Funzione per mostrare il menu iniziale e ottenere la scelta dell'utente
char mostraMenuIniziale() {
    char scelta;
    printf("Benvenuto al quiz di Epicode!\n");
    printf("A) Iniziare una nuova partita\n");
    printf("B) Uscire dal gioco\n");
    printf("Scegli: ");
    scanf(" %c", &scelta);
    return scelta;
}
```

Questo è il blocco della funzione **mostraMenuIniziale()**. La funzione mostra un messaggio di benvenuto e un menu iniziale all'utente, quindi ottiene e restituisce la scelta dell'utente.

```c
// Funzione per gestire una nuova partita
void nuovaPartita() {
    int punteggio = 0;
    char nome[50];
```

```c
void nuovaPartita() {
    int punteggio = 0;
    char nome[50];

    // Ottieni il nome del giocatore
    printf("Inserisci il tuo nome: ");
    scanf("%s", nome);

    // Array di domande
    Domanda domande[5] = {
        {"Qual è la capitale dell'Italia?", {"Roma", "Parigi", "Berlino"}, 0},
        {"Quanto fa 2 + 2?", {"3", "4", "5"}, 1},
        {"Chi è l'autore di Romeo e Giulietta?", {"Shakespeare", "Dante", "Hemingway"}, 0},
        {"Quale pianeta è conosciuto come la 'Sorella della Terra'?", {"Marte", "Venere", "Giove"}, 1},
        {"Quante corde ha una chitarra classica?", {"4", "6", "8"}, 1}
    };

    // Loop attraverso le domande
    for (int i = 0; i < 5; i++) {
        printf("\nDomanda %d: %s\n", i + 1, domande[i].testo);
        printf("A) %s\nB) %s\nC) %s\n", domande[i].risposte[0], domande[i].risposte[1], domande[i].risposte[2]);

        char rispostaUtente;
        printf("Inserisci la tua risposta (A, B o C): ");
        scanf(" %c", &rispostaUtente);

        // Valuta la risposta con switch
        switch (toupper(rispostaUtente)) {
            case 'A':
            case 'B':
            case 'C':
                if (toupper(rispostaUtente) - 'A' == domande[i].rispostaCorretta) {
                    printf("Corretto!\n");
                    punteggio++;
                } else {
                    printf("Sbagliato. La risposta corretta è %c)\n", 'A' + domande[i].rispostaCorretta);
                }
                break;
            default:
                printf("Risposta non valida. Per favore, inserisci A, B o C.\n");
                i--;  // Decrementa l'indice per ripetere la stessa domanda
                break;
        }
    }

    // Stampa il punteggio finale
    printf("\n%s, il tuo punteggio finale è: %d/5\n", nome, punteggio);
}
```

Questo è il blocco della funzione **nuovaPartita()**. Questa funzione gestisce l'intero flusso di una nuova partita. Include la logica per ottenere il nome del giocatore, presentare una serie di domande, valutare le risposte, calcolare il punteggio finale e stampare il risultato.

```c
int main() {
    char scelta;

    do {
        scelta = mostraMenuIniziale();

        switch (scelta) {
            case 'A':
                nuovaPartita();
                break;
            case 'B':
                printf("Grazie per aver giocato. Arrivederci!\n");
                break;
            default:
                printf("Scelta non valida. Per favore, inserisci A o B.\n");
        }
    } while (scelta != 'B');

    return 0;
}
```

Questo è il blocco principale (**main**). Il programma entra in un ciclo che continua fino a quando l'utente sceglie di uscire. Durante ogni iterazione, mostra il menu iniziale, gestisce la scelta dell'utente, e chiama la funzione **nuovaPartita()** se l'utente sceglie di iniziare una nuova partita. Il programma continua a eseguire questo ciclo finché l'utente sceglie di uscire.