Esame di Programmazione Cl. B (Parte 2)

12 Luglio 2024

Ingegneria e Scienze Informatiche A.A. 2023-2024

Questo esame prevede lo sviluppo di un programma per gestire una lista di punti di interesse geografici. Ogni punto è definito da latitudine, longitudine e descrizione. Gli studenti implementeranno e testeranno quattro funzioni per manipolare una lista semplicemente concatenata.

Punteggio Totale: 16 Punti

Il punteggio è suddiviso equamente tra le quattro funzioni, con 4 punti assegnati a ciascuna funzione implementata correttamente.

Ambiente di Sviluppo

Gli studenti dovranno utilizzare il progetto Visual Studio già configurato per ANSI C, che include i file main.c, lib.h e il file lib.c vuoto. Tutte le modifiche devono essere apportate esclusivamente al file lib.c, che è l'unico file che gli studenti dovranno consegnare.

Regole dell'Esame

- **Strumenti Consentiti**: È consentito l'uso esclusivo di Visual Studio per lo sviluppo del codice. Non è permesso l'uso di internet, Google, ChatGPT o qualsiasi altro strumento o risorsa online.
- **File da Consegnare**: Gli studenti devono consegnare soltanto il file lib.c.
- **Collaborazione**: Non è consentita la collaborazione tra studenti o con terze parti.

Librerie Necessarie

- **stdio.h**: Per le operazioni di input/output.
- **stdlib.h**: Per la gestione della memoria.
- **string.h**: Per la manipolazione delle stringhe.

Struttura dei Dati

```c
typedef struct PuntoInteresse {
    float latitudine;
    float longitudine;
    char descrizione[100];
    struct PuntoInteresse *next;
} PuntoInteresse;
```

Funzioni da Implementare

Inserisci Punto (4 punti)

```
void inserisciPunto(PuntoInteresse **testa, float latitudine, float longitudine,
char *descrizione);
```

Inserisce un nuovo punto alla fine della lista. Utilizza malloc() per allocare memoria per un nuovo nodo e strcpy() per copiare la descrizione.

Rimuovi Punto (4 punti)

```
void rimuoviPunto(PuntoInteresse **testa, char *descrizione);
```

Rimuove un punto dalla lista basato sulla descrizione. Utilizza strcmp() per confrontare le descrizioni e free() per liberare la memoria del nodo rimosso.

Cerca Punto (4 punti)

```
void cercaPunto(PuntoInteresse *testa, char *descrizione);
```

Cerca un punto per descrizione e stampa le sue coordinate se trovato. Utilizza strcmp() per il confronto delle stringhe.

Stampa Lista (4 punti)

```
void stampaLista(PuntoInteresse *testa);
```

Stampa tutti i punti nella lista.

File di Intestazione `lib.h`

```h
#ifndef LIB_H
#define LIB_H

#include <stdlib.h> // Include per malloc e free
#include <string.h> // Include per strcpy e strcmp

// Definizione della struttura PuntoInteresse
typedef struct PuntoInteresse {
    float latitudine;
    float longitudine;
    char descrizione[100];
    struct PuntoInteresse *next;
} PuntoInteresse;

// Dichiarazione delle funzioni che manipolano la lista di PuntiInteresse
void inserisciPunto(PuntoInteresse **testa, float latitudine, float longitudine, char *descrizione);
void rimuoviPunto(PuntoInteresse **testa, char *descrizione);
void cercaPunto(PuntoInteresse *testa, char *descrizione);
void stampaLista(PuntoInteresse *testa);

#endif
```

Riepilogo delle Funzioni di Stringa

```
char *strcpy(char *dest, const char *src);
```

La funzione `strcpy()` viene utilizzata per copiare il contenuto di una stringa in un'altra. È particolarmente utile per assegnare il valore di una stringa a una nuova variabile di tipo array di caratteri, come nel caso della descrizione di un punto di interesse nel nostro esame.

Parametri:

- `dest`: puntatore alla stringa di destinazione, dove verrà copiata la stringa sorgente.
- `src`: puntatore alla stringa sorgente da copiare.

Ritorno: La funzione restituisce un puntatore alla stringa di destinazione.

```
int strcmp(const char *s1, const char *s2);
```

La funzione `strcmp()` confronta due stringhe carattere per carattere. Se le stringhe sono uguali, la funzione restituisce 0. Se sono diverse, restituisce un valore negativo se la prima stringa è minore della seconda o un valore positivo se maggiore.

Parametri:

- `s1`: puntatore alla prima stringa da confrontare.
- `s2`: puntatore alla seconda stringa da confrontare.

Ritorno: Restituisce `0` se le stringhe sono uguali, un valore `<0` se `s1` è lessicograficamente minore di `s2`, e `>0` se `s1` è maggiore di `s2`.