Tutorato 07_11_2022

Strutture e Allocazione dinamica

Esercizio 0

Data un struttura `punto` contenente le coordinate `x` e `y` implementare le seguenti funzioni:

- Inizializzazione del punto del punto
- Stampa del punto
- Stampa dell'equazione della retta passante per i due punti, con tutti i controlli del caso

Esercizio 1

Data una struttura `arraylist`, la quale rappresenta un array la cui dimensione varia in base al numero di elementi al suo interno,

```
struct arraylist{
    const float expansion_factor = 2;
    const float reduce_factor = 0.25;
    int current_length = 0;
    int current_max_length = 0;
    int* list = nullptr;
}
```

Implementare le seguenti funzioni:

- `arraylist* init(const int)` la quale ritorna un arraylist allocato dinamicamente
- `void expand(arraylist&)` la quale deve incrementare la lunghezza massima dell'array, di un fattore `expansion_factor`, quando si cerca di inserire un nuovo elemento e l'array è pieno
- `void reduce(arraylist&)` la quale deve ridurre la lunghezza massima dell'array, di un fattore `reduce_factor`, quando il numero di elementi presenti nell'array è inferiore al `25%` della sua lunghezza massima
- `void add(arraylist&, const int)`, la quale permette di inserire un elemento in coda all'array
- `void remove_last(arraylist&)`, la quale rimuove l'ultimo elemento in coda all'array
- `void print(arraylist& )`, stampa a video tutte le informazioni contenute nella struttura corrente

Note

- Deallocare tutto il necessario(non solo la struttura prima di uscire dal programma!)
- Inserimento, rimozione e stampa devono essere controllati dall'utente attraverso un menù scritto nel main

Esercizio 2

Date le strutture `person` e `node`, implementare una `lista concatenata` di persone attraverso le seguenti funzioni:

```
struct person {
    char name[35];
    int age;
};

struct node {
    person p;
    node* next;
};

person init_person(char name[35], int age);
void print_person(person p);
node* init_node(person p, node* next);
void insert_ordered(node* &l, person p);
void print_list(node* l);
void delete_list(node* &l);
```

Esercizio 3

Implementare una lista doppiamente concatenata definendo le seguenti funzioni:

- inserimento ordinato
- rimozione nodo per valore
- stampa lista (valore corrente, precedente e prossimo)

Permettere all'utente di scegliere se aggiungere/ eliminare un nodo oppure stampare la lista tramite un apposito menù di scelta nel main. Ricordarsi di deallocare tutti gli elementi della lista al termine del programma.

```
struct node {
    int val;
    node* next;
    node* prev;
};

void inserisci_in_testa(node*& lista, int val);
void init_lista(node*& lista, int val);
void inserisci_ordinato(node*& lista, int val);
void rimuovi_nodo(node*& lista, int val);
void stampa(node* lista);
void dealloca_lista(node*& lista);
```