1 Liste

Esercizio 1.1

Implementare

- una funzione per calcolare la lunghezza di una lista;
- una funzione in grado di invertire l'ordine di una lista L senza utilizzare una lista di supporto;
- una funzione che stabilisca se una lista è palindroma.

```
int lunghezza(Lista);
void inverti(Lista &);
bool palindoma(Lista);
```

Le funzioni possono anche essere implementate come metodi della classe astratta LinearList.

Esercizio 1.2

Liste lineari ordinate: *particolare tipo di lista in cui l'ordine sequenziale degli elementi è legato ad una relazione d'ordine definita sugli elementi*.
Esempio: se gli elementi sono stringhe si suppone che l'odinamento sia per chiave alfabetica. A tale scopo sarà necessario definire gli operatori '<', '<=' e '>' per il tipo di dato stringa (Tali operatori sono già definiti per il tipo di dato **string**, definito nella libreria **string**.).

- Modificare l'operatore **insLista** per liste lineari ordinate (oppure definire un nuovo operatore insListaOrdinata);
- implementare l'algoritmo di *ricerca lineare* ordinata;
- implementare l'algoritmo di *fusione* di liste ordinate;
- applicare le funzioni a liste lineari di stringhe ordinate per chiave alfabetica.

Esercizio 1.3

Multilista: supponendo di voler rappresentare un numero intero I mediante una lista, tale che il valore dell'elemento i-esimo della lista corrisponda a quello dell'i-esima cifra di I, definire una lista di interi ed implementare le funzioni per

- l'acquisizione di n interi
- la visualizzazione di interi
- la somma di interi

```
l1 --> 1 3 5 6
l2 --> 3 4
l3 --> 5 6 7
l4 --> 1
```

Suggerimenti per la realizzazione con cursori

Definire la variabile SPAZIO come membro della classe lista di tipo **static**.

Di norma oggetti diversi della stessa classe non condividono risorse di memoria. Per poter realizzare una *comunicazione di ambiente condiviso* bisogna dichiarare l'attributo (comune a tutte le istanze) come static. Gli attributi static possono essere visti come elementi propri della classe, non dell'istanza.

```
class Lista
  {

  public:
    ...
    Lista();    //costruttore
    ~Lista();   //distruttore

    // operatori
    void crealista();
    ...

  private:
    Static componenteSpazio SPAZIO[100];
    // metodi per la manipolazione dell'attributo SPAZIO
    ...
    // altri dati privati
  };
```

componenteSpazio è un tipo strutturato costituito da una componente 'elemento' di tipo tipoelem e da una componente successivo di tipo posizione.