Programmazione ad Oggetti
Homework 1 (ADT)

Esercizio 1

Modificare la ADT Insieme, descritta nelle slides proiettate a lezione (il codice è presente nella cartella src del file ADT_PO.zip). La modifica dovrà consentire di rappresentare un insieme potenzialmente infinito: il numero massimo di elementi non è assegnato in fase di inizializzazione. A tale scopo sarà necessario modificare la funzione add() in maniera tale da allocare nuova memoria se l'array si riempie. Vanno inoltre opportunamente modificate le funzioni init(), full() e clear(). La nuova specifica sarà:

```
typedef struct Set{
    void init();   // funzione di inizializzazione.
    int  size();   // restituisce la cardinalità dell'insieme.
    bool empty();  // restituisce TRUE se è vuoto.
    bool full();   // restituisce TRUE se è pieno.
    void clear();  // elimina tutti gli elementi dall'insieme
    void add(TipoValue val); // aggiunge il valore val
    void remove(TipoValue val); // elimina il valore val
    bool member(TipoValue val);  // restituisce TRUE se val è presente.
  private:
    void change_dim (float f); // Aumenta/riduce il vettore quando
                               // necessario
    TipoValue *v;
    int  n, len;
    elim(int pos);
}
```

la funzione di "servizio" `change_dim()` va utilizzata per consentire l'aumento o la riduzione delle dimensioni del vettore, a seconda dei casi:

- se il vettore è pieno (n == len), le dimensioni vanno aumentate: la funzione change_dim va chiamata passandole un parametro > 1.0.
- se il numero di elementi presenti è molto minore della memoria allocata (n << len): la funzione change_dim va chiamata passandole un parametro < 1.0.

Per quanto riguarda l'entità di variazione delle dimensioni, si definiscano due costanti: la prima, maggiore di 1.0, determina di quanto aumentare il vettore. La seconda, invece, minore di 1.0, determina l'entità di riduzione del vettore. È opportuno inoltre definire anche un'ulteriore costante per determinare il riempimento minimo al di sotto del quale il vettore verrà ridotto.

Per quanto riguarda la funzione `clear()`, si faccia anche qui un ragionamento sulla riduzione del vettore: conviene ridurlo? Se sì, di quanto?

La nuova versione della funzione `init()`, infine, necessità anch'essa della definizione di una costante (intera), che determini le dimensioni iniziali del vettore.

Esercizio 2

Modificare la ADT dell'esercizio precedente memorizzando gli elementi in un vettore ordinato. A tale scopo bisognerà modificare le seguenti funzioni:

- `add()`

– `member()`
– `remove()`

La funzione `member()` deve implementare la ricerca binaria.

Suggerimenti

Per la nuova versione della funzione add() si usi la funzione ins della ADT list (file list.cpp). Aggiungendola come funzione private all'interno della struct.

Per la ricerca binaria si aggiunga all'interno della struct come funzione private, la seguente funzione:

```
bool search_bin(int vet[], int n, int x);
```

Si tenga presente che in questo caso il vettore v è membro della struct e può essere acceduto senza passato come parametro.

La ADT insieme è stata definita in maniera generale, senza specificare nessun tipo di dato. L'insieme ordinato prevede che ci sia appunto un ordinamento, definito da una funzione che avrà questo prototipo:

```
bool less_than(TipoValue a,  TipoValue b);
```

che restituisce il valore **`true`** se a è minore di b secondo l'ordinamento previsto da quel tipo di dato.

Si preveda inoltre la presenza di una funzione di confronto, sempre fornita dall'utente, che restituisce **`true`** sse a==b. Il prototipo della funzione sarà il seguente:

```
bool is_equal(TipoValue a,  TipoValue b);
```

Entrambi le funzioni dovranno essere definite dall'utente utilizzatore della ADT.

Le funzioni di ordinamento di cui sopra possono essere gestite per mezzo di puntatori a funzioni. Questi puntatori, che dovranno essere opportunamente richiamati dalle funzioni che prevedono un confronto (cerca_bin e remove)

A tale scopo, nel file header si inseriscano le seguenti dichiarazioni:

```
bool (*eq_fptr)  (TipoValue, TipoValue);
bool (*lt_fptr) (TipoValue, TipoValue);
```

Una volta implementata, la ADT potrà essere testata definendo le funzioni di cui sopra per il tipo int (oppure float) e assegnando i puntatori alle funzioni definiti. Queste operazioni potranno essere fatte all'interno del main del programma.