Politecnico di Torino

Algoritmi e struttire dati
laboratorio 08 - esercizio 03

Riccardo Rosin - s284211

Gen 2023

Indice

1 Struttura dati **2**
1.1 Modulo Data 2
1.2 Modulo Titolo 2
1.3 Modulo List 3
1.4 Modulo Quotazione 3
1.5 Modulo BST 4

2 Scelte algoritmiche **5**

3 Commenti finali **6**

1 Struttura dati

Come richiesto dalla consegna, la struttura dati è organizzata su tre livelli:

- Un primo livello per la collezione dei titoli
- Un secondo livello per la collezione selle sue transazioni
- Un livello aggiuntivo, alla base di entrambi, per rappresentare la data

I titoli e le transazioni sono due istanze del modulo ITEM visto a lezione, denominati rispettivamente ***Titolo*** e ***Quotazione***, raggruppati nei rispettivi moduli.

1.1 Modulo Data

La data e l'ora sono rappresentate assieme come quasi ADT di tipo ***Datetime***, implementata come segue.

```
typedef struct date_time{
    int year;
    unsigned int month;
    unsigned int day;
    unsigned int hour;
    unsigned int minute;
} Datetime;
```

Il modulo contiene le funzioni base di lettura e scrittura, oltre a quelle di confronto tra due date. Per necessità, sono state create delle funzioni per gestire soltanto i campi dedicati alla data, ma non le corrispettive per l'ora, in quanto all'interno del programma verrà ignorato il campo orario.

1.2 Modulo Titolo

Il modulo ***Titolo*** è una rivisitazione del modulo ***Item*** visto a lezione, implementato come ADT di prima classe.

```
typedef struct titolo_s{
    char codice[LEN];
    BST qt;
}* Titolo;
```

L'item è identificato univocamente dal campo *codice*, definito nel header del modulo come tipo ***KeyT***, mentre il campo *qt* è il puntatore alla struttura di secondo livello che salva le varie transazioni.
Il modulo implementa le classiche funzioni del tipo Item.

1.3 Modulo List

Il primo livello della struttura dati è organizzato in una lista ordinata rappresentata come ADT di prima calsse di tipo ***List***.

```
typedef struct node_s{
    Titolo val;
    struct node_s *prev;
    struct node_s *next;
} node, *link;

typedef struct list_s{
    link head;
    link tail;
    int N;
    } *List;
```

La lista è implementata come una lista doppiamente concatenata, con un puntatore alla testa e un puntatore alla coda per generalità, anche se questo non comporta un alcun miglioramento alle prestazioni della struttura.

1.4 Modulo Quotazione

L'algoritmo è stato impostato per non memorizzare le singole transazioni, in quanto non necessarie per la richiesta. Ogni elemento rappresenta l'insieme giornaliero delle operazioni, memorizzando le informazioni strettamente necessarie per il calcolo richiesto.

```
typedef struct quatazione_giornaliera{
    Datetime dt;
    unsigned int numerator;
    unsigned int denominator;
    unsigned short int null;
} Quotazione;
```

Il tipo ***Quotazione*** è rappresentato come quasi ADT, ed utilizza il campo *dt* come identificatore univoco dell'elemento, identificato con il tipo ***KeyQ***.
I campi *numerator* e *denominator* sono i numeratore e il denominatore della quotazione giornaliera. Per evitare di salvare tutte la varie operazioni, il tipo ***Quotazione*** implementa direttamente il calcolo richiesto, per fare ciò è stato necessario studiare la Funzione: essa viene spezzata nel numeratore e denominatore che vengono salvati e manipolati indipendentemente, La funzione si poteva scomporre in altri modi che però utilizzavano la virgola mobile, ed essendo imprecisa, si è preferito ricadere su una rappresentazione attraverso interi.
Il campo *null* è un flag che indica se l'item è stato inizializzato o meno.

1.5 Modulo BST

Il secondo livello della struttura dati è organizzato in un albero binario di ricerca, rappresentato come ADT di prima classe di tipo ***BST***.

```
typedef struct BSTnode {
    Quotazione item;
    link p;
    link l;
    link r;
    int N;
}* link;
struct binarysearchtree {
    link root;
    link z;
};
```

Il modulo implementa, in aggiunta alle funzioni fornite, una funzione di ricerca del massimo in un intervallo di chiavi ed una funzione di calcolo dello sbilancio dell'albero.
Il dato usa la chiave dell'Item ***Quotazione*** come sua chiave di ordinamento.

2 Scelte algoritmiche

Il programma è gestito runtime tramite linea di comando: all'avvio del programma compare un primo menù testuale per differenziare le operazioni di manipolazione e visualizzazione da quelle di iserimento.
La funzione di lettura riceve il nome del file da cui estrarre i dati e, scandendo la struttura allocata, verifica di riga in riga l'inserimento o l'aggiornamento del dato. A seguito della richiesta dell'utente di visualizzare una quotazione, il programma propone un secondo menù che illustra le sue varie funzionalità.
La funzione di ricerca nell'intervallo di date del BST sfrutta una visita post order per cercare il massimo ed il minimo. La stessa funzione viene chiamata con l'elemento minimo e massimo del BST per effettuare una ricerca su tutto l'intervallo.

3 Commenti finali

La parte meno soddisfacente è la gestione, nel modulo ***Titolo***, della chiave. Essendo il tipo definito come ADT di prima classe ed allocato dinamicamente, esso dipende molto dalla rappresentazione interna che lo implementa e lascia poco margine di errore al client: esso dovrà infatti fare attenzione a tutte le assegnazioni in quanto possono sovrascrivere il dato precedente e lasciare del memory leack nel programma. Durante il programma è stato necessario richeidere all'utente una singola chiave per effettuare delle operazioni. La strategia adottata è stata quella di allocare un item ed inizializzargli soltato il campo chiave tramite una funzione apposita del modulo. Per ricevere eventuali valori di ritorno delle funzioni chiamate con questa chiave, è stato necessario utilizzare una variabile d'appoggio per permettere la corretta deallocazione.