Hashtabelle mit Kollisionsbehandlung

Die Struktur `struct HashTable` stellt eine *Hashtabelle* (im Folgenden *Tabelle*) mit Kollisionsbehandlung durch einfach verkettete Listen dar. Die an der Datenstrukture beteiligten Stukturen sehen so aus:

```
typedef struct Book T; // stored type

struct Node
{
  struct Node *next; // next node in list (or NULL if last node in list)
  T value;           // stored value
};

struct HashTableEntry
{
  struct Node *list; // first node in list (or NULL if list is empty)
};

struct HashTable
{
  struct HashTableEntry *data;    // array indexed by the hashed value
  size_t size;                    // size of the array
  size_t count;                   // number of aktive entries
  size_t (*hash)(const T *value); // hash function for entries/keys
  bool (*compare)(const T *value1,
                  const T *value2); // compare function for entries/keys
};
```

Die von Ihnen zu entwickelnden Funktionen sind:

`hashtable_new`: Dynamisches Tabellen-Objekt erzeugen

```
struct HashTable *hashtable_new(size_t size, size_t (*hash)(const T *value),
                                bool (*compare)(const T *value1, const T *value2));
```

Funktionsbeschreibung: Alloziert ein `struct HashTable`-Objekt, initialisert es und gibt einen Zeiger darauf zurück.

Parameter

- `size`: Initiale Feldgröße
- `hash`: Funktionszeiger auf die Funktion, die zum Generieren eines Hashes verwendet wird.
- `compare`: Funktionszeiger auf die Funktion, die zum Vergleichen zweier Werte verwendet wird.

Rückgabewert: Zeiger auf das dynamisch allozierte Objekt.

`hashtable_delete`: Dynamisches Vector-Objekt freigeben

```
void hashtable_delete(struct Vector **self);
```

Funktionsbeschreibung: Deallozierd ein per doppelter Referenz übergebenes alloziertes `struct HashTable`-Objekt, gibt zuvor den durch die Tabelle gehaltenen dynamischen Speicher frei und setzt den Zeiger auf die Tabelle auf `NULL`.

Parameter:

- `self`: Zeiger auf einen Zeiger auf eine dynamisch allozierte Tabelle (Doppelzeiger).

`hashtable_clear`: Alle Einträge entfernen

```
void hashtable_clear(struct HashTable *self);
```

Funktionsbeschreibung: Entfernt alle Einträge aus der Tabelle.

Parameter:

- `self`: Zeiger auf eine dynamisch allozierte Tabelle.

`hashtable_insert`: Eintrag einfügen

```
bool hashtable_insert(struct HashTable *self, const T *value);
```

Funktionsbeschreibung: Fügt den übergebenen Wert in die Tabelle ein. Zum Berechnen des Hashes wird die Funktion `hash` verwendet. Um zu überprüfen ob der dem einzufügenden Wert zugeordnete Schlüssel bereits vorhanden ist, wird die Funktion `compare` verwendet. Ist der Schlüssel schon in der Tabelle vorhanden wird die Tabelle nicht verändert.

Parameter:

- `self`: Zeiger auf eine dynamisch allozierte Tabelle.
- `value`: Zeiger auf den einzufügenden Wert (Wertepaar).

Rückgabewert: `false` falls der Schlüssel bereits vorhanden war, und `true` falls übergebene der Wert eingetragen wurde.