BHT Berliner Hochschule für Technik

Algorithmen & Datenstrukturen

WS 2025

Hausaufgabe 2

In dieser Aufgabe erweitern Sie Ihre Implementierung der doppelt verketteten Liste um die Möglichkeiten die Liste zu sortieren, sortiert einzufügen und sortiert darin zu suchen. Damit die Umsetzung mit beliebigen Daten funktioniert, sollen die Vergleichsfunktionen als Funktionszeiger übergeben werden.

a) Insertion Sort

Ihre Aufgabe in dieser Übung ist die Implementierung von **Insertion Sort** für die doppelt verkettete Liste aus Hausaufgabe 1.

Dazu finden Sie in Moodle eine Textdatei `dlist_new_api.txt`, in welcher die neuen Funktionen der Liste beschrieben sind. Dort finden Sie auch die Definition von `comparefunc_t`. Dieser Datentyp passt auf alle Funktionen, welche eine Zahl vom Typ `int` zurückgeben und zwei formale Parameter vom Zeiger-Datentyp `void*` erhalten. Fügen Sie also den Inhalt dieser Textdatei zuerst Ihrer Header-Datei `dlist.h` hinzu.

Beachten Sie: die Vergleichsfunktion soll **die Werte** der übergebenen Zeiger als passenden Datentyp interpretieren und vergleichen, also **nicht** die Zeigeradressen selbst. Ist der Wert des ersten Parameters kleiner als der des zweiten, wird -1 zurückgegeben. Ist der Wert des ersten Parameters hingegen größer, so wird 1 zurückgegeben. Sind die beiden Werte gleich, ist der Rückgabewert die Zahl 0.

1. Implementieren Sie die Funktion

   ```
   void dlist_sort(dlist_t* list, comparefunc_t cmp);
   ```

   a) Die Funktion erhält einen Zeiger auf die doppelt verkettete Liste und einen Funktionszeiger auf eine Vergleichsfunktion wie oben beschrieben.

   b) Die Funktion implementiert den Algorithmus **Insertion Sort**.

   c) Die übergebene Liste wird **in-place** sortiert, d.h. es soll keine neue Liste erstellt werden.

b) Sortierter Zugriff auf die Daten der Liste

Als nächster Schritt soll das sortierte Einfügen und Suchen in der Liste umgesetzt werden.

1. Implementieren Sie die Funktion

```
void dlist_insert_sorted(
    dlist_t* list, void* data, comparefunc_t cmp
);
```

   a) Die Funktion erhält die Liste in die eingefügt werden soll, den Zeiger das einzufügende Datum und die zu verwendende Vergleichsfunktion. Sie können annehmen, dass die Liste bereits sortiert ist.

2. Implementieren Sie die Funktion

```
int dlist_find_sorted(
    dlist_t* list, void* data, comparefunc_t cmp
);
```

   a) Die Funktion erhält die Liste in die eingefügt werden soll, den Zeiger auf das einzufügende Datum und die zu verwendende Vergleichsfunktion. Sie können annehmen, dass die Liste bereits sortiert ist. Die Funktion gibt die gefundene Position in der Liste zurück. Werden die Daten nicht gefunden, wird -1 zurückgegeben.

c) Abgabe

1. Bereiten Sie für die Abgabe ihr Verzeichnis so vor, dass dieses Ihre Quellcode-Dateien, das `Makefile` und die erweiterte Datei `dlist.h` enthält.

2. Komprimieren Sie das Verzeichnis in eine .zip-Datei und laden diese in der Moodle-Abgabe für die Übung hoch.

3. Die folgenden Anforderungen werden in der Abgabe geprüft:

   a) Die Datei `dlist.h` wurde mit dem Inhalt der neuen API erweitert, aber sonst nicht verändert.
   b) Die Test-Anwendung kann mit dem `Makefile` übersetzt werden (Target `test`) und die kompilierte Anwendung hat den Namen `main`.
   c) Die implementierten Funktionen funktionieren wie in der Dokumentation beschrieben.
   d) Keine Probleme durch Verwendung von dynamischem Speicher werden gefunden.
   e) Eine Plagiatsprüfung findet keine Treffer.