technische universität dortmund

fakultät für informatik

Andre Droschinsky
Roman Kalkreuth
Denis Kurz
Bernd Zey

Dortmund, den 20. Dezember 2018

Praktikum zur Vorlesung Einführung in die Programmierung WS 18/19

Blatt 09

Es können 25 Punkte erreicht werden.

Aufgabe 1: ADT Stapel (15 Punkte)

In dieser Aufgabe soll der abstrakte Datentyp (ADT) Stapel mit Hilfe von Templates und dynamischem Speicher für beliebige Elementtypen implementiert werden. Wir wollen die Elemente des Stapels durch ein Hilfs-Struct `StackElement` repräsentieren, das ein Datenfeld des Typs `T` und einen Zeiger auf das nächste Element weiter unten im Stapel (bzw. `nullptr`, falls dies das unterste Element ist) enthält. Der Stapel selbst besitzt lediglich einen Zeiger auf das oberste Element im Stapel.

Einschränkung: Orientieren Sie sich für Ihre Implementierung an den Laufzeitvorgaben aus der Vorlesung!

a) Fügen Sie einem neuen Projekt `Aufgabe_09_1` eine Headerdatei `stack.h` hinzu. Definieren Sie in der Headerdatei eine parametrisierte Klasse `Stack` mit einem Template-Parameter `T`. Fügen Sie in der Klasse `Stack` ein *privates* Hilfs-Struct `StackElement` hinzu, das ein Datenfeld `data` vom Typ `T const` und einen Zeiger auf das nächste Element enthält. Fügen Sie der Klasse `Stack` einen *privaten* Zeiger `m_top` auf ein `StackElement`-Objekt hinzu.

Hinweis: In dieser und den folgenden Aufgaben sollen alle Implementierungen in der Header-Datei erfolgen. Legen Sie also keine zusätzliche `cpp`-Datei an. Grund hierfür ist eine Eigenheit von C++ in Zusammenhang mit Templates. Diese müssen entweder im Header implementiert oder auf unübliche Art und Weise inkludiert werden. Wir entscheiden uns hier für erstere Alternative.

_____ (3)

b) Fügen Sie der Klasse `Stack` einen Default-Konstruktor hinzu, der einen leeren Stapel erzeugt, sowie eine Methode `empty`, die `true` zurückgibt, falls der Stack leer ist, ansonsten `false`.

_____ (2)

c) Schreiben Sie eine Methode `push`, die ein Element `x` vom Typ `T` auf den Stapel legt. Überlegen Sie sich, welchen Übergabemechanismus Sie für den Parameter `x` am besten verwenden.

_____ (2)

d) Schreiben Sie eine Methode `pop`, die das oberste Element vom Stapel entfernt. Achten Sie darauf, den dynamisch allozierten Speicher korrekt wieder freizugeben. Falls der Stapel leer ist, soll eine Fehlermeldung ausgegeben und das Programm mit `exit(1)` beendet werden. Um die Funktion `exit` nutzen zu können, müssen Sie die Bibliothek `cstdlib` einbinden.

_____ (2)

e) Schreiben Sie eine Methode `top`, die eine Referenz auf das oberste Element im Stapel zurückgibt. Falls der Stapel leer ist, soll ebenfalls eine Fehlermeldung ausgegeben und das Programm mit `exit(1)` beendet werden.

_____ (1)

f) Schreiben Sie eine Methode `clear`, die den Stapel leert (also alle Elemente vom Stapel entfernt), sowie einen Destruktor, der den dynamisch allozierten Speicher wieder frei gibt.

_____ (2)

g) Fügen Sie Ihrer Implementierung der Klasse einen Copy-Konstruktor und einen Zuweisungsoperator hinzu.

_____ (2)

h) Testen Sie Ihre Implementierung mit der bereitgestellten Quelldatei `stack_test.cpp`.

_____ (1)

Aufgabe 2: Erweiterung des ADT Liste (10 Punkte)

Der in der Vorlesung vorgestellte ADT Liste soll in dieser Aufgabe erweitert werden. Es sollen Listen implementiert werden, die sowohl vorwärts als auch rückwärts durchlaufen werden können. Um diese Idee möglichst einfach und effizient zu realisieren, sollen sogenannte doppelt verkettete Listen verwendet werden. Bei diesen ist jedes Listenelement mit dem nächsten und dem vorherigen Element durch jeweils einen Zeiger verbunden, d.h. jedes Element hat zwei Zeiger und somit eine Struktur wie in Abbildung 1.

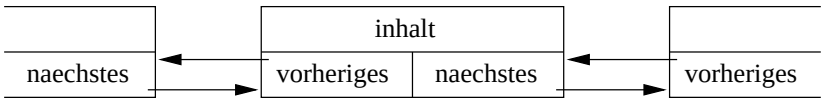

Abbildung 1: Doppelt verkettetes Listenelement

Im Folgenden soll die bereits aus Aufgabenblatt 8 bekannte (einfach verkettete) Liste wie oben beschrieben erweitert werden. Dazu ist Ihnen die Datei `eidpliste.h` vorgegeben. Legen Sie ein neues Projekt `Aufgabe_09_2` an, dem Sie die Header-Datei hinzufügen. Darin sollen nachfolgend alle Änderungen oder Ergänzungen vorgenommen werden.

EINSCHRÄNKUNG: Orientieren Sie sich für Ihre Änderungen oder Ergänzungen an den Laufzeitvorgaben aus der Vorlesung!

a) Betrachten Sie das Hilfs-Struct `ListElement`. Notieren Sie kurz, welche Veränderungen daran nötig sind, damit statt der bisher implementierten einfachen Verkettung nun eine doppelte Verkettung möglich ist.

_____ (1)

b) Betrachten Sie nun die gesamte Klasse `Liste`. Notieren Sie kurz, in welchen Methoden Änderungen nötig sind, damit statt der bisher implementierten einfachen Verkettung nun eine doppelte Verkettung möglich ist. Notieren Sie darüber hinaus für jede der identifizierten Methoden kurz, warum eine Änderung nötig ist.

HINWEIS: Sie sollten Teilaufgaben a) and b) testieren lassen, bevor Sie die weiteren Teilaufgaben bearbeiten.

_____ (1)

c) Implementieren Sie alle nötigen Änderungen, sodass die Klasse `Liste` anschließend doppelt verkettete Elemente enthält.

Um die von Ihnen erweiterte Funktionalität zu testen, soll diese sofort sinnvoll genutzt werden. Implementieren Sie dazu eine Methode `void print(bool directionForward)`. Diese soll alle Elemente der Liste per `cout` ausgeben. Ist `directionForward == true`, soll die Ausgabe vom ersten zum letzten Element stattfinden. Andernfalls soll vom letzten zum ersten Element ausgegeben werden. Formatieren Sie Ihre Ausgabe gemäß der folgenden Beispielausgabe:

```
1 2 3 4 5 6
```

HINWEIS: Testen Sie Ihre Ergebnisse vor Abgabe mit der bereitgestellten Testumgebung (`eidpliste_test.cpp`). Der darin enthaltene Code ist in verschiedene Bereiche eingeteilt, die den unterschiedlichen Teilaufgaben zugeordnet sind. Damit die Testumgebung kompilieren kann, auch wenn Sie noch nicht alle Teilaufgaben bearbeitet haben, müssen Sie den Code, der für die folgenden Teilaufgaben bestimmt ist, auskommentieren. Testen Sie auch **alle nachfolgenden** Teilaufgaben vor Abgabe mit der Testumgebung.

_____ (4)

d) Implementieren Sie eine Methode `reverse`, die die Reihenfolge der Elemente in der Liste umdreht. Das erste Element soll also anschließend am Ende stehen, das Letzte am Anfang usw.

EINSCHRÄNKUNG: Legen Sie keine Hilfslisten oder Hilfslistenlemente an und verschieben Sie nicht den Inhalt der Liste. Arbeiten Sie stattdessen nur mit den Zeigern der Listenelemente und (nach Bedarf) mit Hilfszeigern.

_____ (2)

e) Implementieren Sie eine Methode `void deleteAt(unsigned int position)`. Diese Methode soll ein Element an der Stelle der Liste löschen, die durch den Parameter `position` spezifiziert wird. Der dynamische Speicher des entfernten Listenelements soll ebenfalls freigegeben werden.

HINWEIS: Achten Sie darauf, die Zeigerstruktur der Liste nicht zu zerstören.

_____ (2)