DAP2 Praktikum für ETIT und IKT, SoSe 2018, Kurzaufgabe K6

Fällig am 09.07 um 18:00

Es gelten die üblichen Programmierregeln in der gewohnten Härte. Hässlicher Code führt zu Punktabzug. Assertions sind nicht notwendig.

Ergänzen Sie das Klassen-Template `template<class T> class Heap` so dass man es für einfache Datentypen zum Beispiel mit `Heap<int> eger;` instanziieren kann. Heap soll eine Prioritätswarteschlage sein, realisiert als Max-Heap.

Beachten Sie folgende Anmerkungen:

- In dem zum Download bereitstehenden Gerüst `Halde.cpp` befinden sich bereits einige der benötigten Teile der Klasse, insbesondere alle benötigten Variablen der Klasse, Konstruktor und Destruktor sowie die nützlichen Funktionen `IsEmpty()` und `HeapSize()`. Der zu verwendende Konstruktor `Heap(int size)` legt die maximale Größe des Heap bei der Erzeugung fest. Das Einbinden weiterer Header ist nicht erlaubt.

- Folgende `public` Funktionen sind noch unvollständig:

`void Insert(T what)`	Fügt ein neues Element in den Heap ein. Doppelte Werte in dem Heap sollen ganz normal gespeichert und nicht entfernt werden.
`T ExtractMax()`	Entnimmt das größte Element aus dem Heap und entfernt es gleichzeitig aus dem Heap.
`operator[]`	Realisiert den lesenden Zugriff auf den Speicher der Werte des Heap, in exakt der Reihenfolge, wie sie im Speicher abgelegt sind.
`T *AscendingContent()`	Liefert einen Pointer auf einen dynamisch neu geschaffenen Speicherbereich. In diesem befindet sich eine Kopie aller belegten Einträge des Heap in aufsteigend sortierter Reihenfolge (das kleinste zuerst). Bei der Implementierung müssen Sie die Heap-Eigenschaften ausnutzen, sonst gibt es in Aufgabeteil K6.3 keine Punkte.

- Diese Funktionen müssen Sie vervollständigen. Neben dem benötigtem Code muss natürlich bei den letzten drei Funktionen der Rückgabewert nach `return` auf einen sinnvollen Wert geändert werden. Sie dürfen keine weiteren `public` Funktionen oder Variablen in das Template einfügen. Im `private` Teil dürfen Sie Hilfsfunktionen, aber keine weiteren Variablen ins Template einfügen.

- Folgende Exceptions vom Typ `const char*` sind zu implementieren:
  - `Insert` erzeugt eine Exception, wenn der Heap schon voll ist.
  - `ExtractMax, operator[]` und `AscendingContent` erzeugen eine Exception, wenn der Heap leer ist.

- `AscendingContent`
  erzeugen eine Exception, wenn `new` fehlschlägt.
- `operator[]`
  erzeugt eine Exception, wenn auf ein nicht existentes Element zugegriffen wird.

- Bitte beachten Sie unbedingt: Die Elemente innerhalb des Heap sind in `T *array` abgelegt. Durch eine im Quelltext vorhandene Modifikation des Pointers kann man das erste Element des Heaps intern nicht wie in C/C++ üblich beginnend mit dem Index `0` ansprechen, sondern mit dem Index `1`. Also:

`array[0]`	ist falsch, hier ist kein Speicher.
`array[1]`	ist das erste Element im Heap.
`array[HeapSize]`	ist das letzte belegte Element im Heap.
`array[MaxHeapSize]`	ist das letzte Element im Heap für das Speicher vorhanden ist.

  Dies ist nicht gemacht worden, um Sie zu ärgern, sondern weil sich der Heap intern auf diese Weise durch Sie einfacher realisieren läßt. Die von `AscendingContent` zurückgegebenen Zeiger auf das Ergebnis müssen aber von Ihnen so implementiert werden, dass sie von außen ganz normal auf das erste Element zeigen. Demzufolge ist

  ```
  x=MeineHalde.AscendingContent(); cout << x[0] << endl;
  ```

  wie üblich ein erlaubter Zugriff auf das erste Element des zurückgelieferten sortierten Inhalts des Heaps. Ebenfalls muss mit `operator[]` von `0` bis `Size()-1` indiziert werden können.

Allgemeine Aufgabenbeschreibung

Sie sind selbst dafür verantwortlich, Ihre Implementierung zu testen. Sie dürfen dazu selbstverständlich das bestehende `main` nutzen. Wenn Ihre Implementierung nur vorgibt, ein Heap zu sein, erhalten Sie keine Punkte.

Aufgabe K6.1 (0,4 Punkte)

Vervollständigen Sie `ExtractMax, Insert` und `Operator[]` Lassen Sie das Programm `Halde` laufen. Ihre Implementierung wird mit Zufallszahlen aufgerufen. Der Heap wird zufällig gefüllt und gelehrt. Der Inhalt des Heap wird nach jedem Füllvorgang und nach jedem Entnahmevorgang ausgegeben. Erklären Sie Ihrem Betreuer anhand der Ausgabe die Funktionsweise des Heap. Es empfiehlt sich die Breite der Konsole auf mindestens 132 Zeichen zu stellen. Wenn Sie die Funktionsweise nicht erklären können oder der Heap kein Heap ist, gibt es keine Punkte.

Aufgabe K6.2 (0,8 Punkte)

Das Programm führt einen Test der einfachen Funktionen `ExtractMax, Insert` automatisch durch. Es überprüft auch das richtige Erzeugen der Exceptions. Nur wenn Ihre Implementierung den Test besteht, können Sie Sie die Punkte für diesen Aufgabenteil erhalten.

Aufgabe K6.3 (0,8 Punkte)

Das Programm führt einen zweiten Test durch, der jetzt `AscendingContent` und `Operator[]` in den Test einbezieht. Nur wenn Ihre Implementierung den Test besteht, können Sie die Punkte für diesen Aufgabenteil erhalten.