Container

I. Ziel

In diesem Praktikum implementieren Sie ein dynamisches Array. Als Array-Elemente können Sie die Temperatur-Anomalien verwenden, die Sie bereits aus einem früheren Praktikum kennen; alternativ Ganzzahlen.

II. Aufgaben

1. Daten einlesen und ausgeben

Lesen Sie die benötigten Eingabedaten von der Standardeingabe ein. Geben Sie zunächst als Funktions-Kontrolle jeden einzelnen Datensatz unmittelbar nach dem Einlesen an der Standardausgabe aus.

Haben Sie sich für die Temperaturanomalien entschieden, beginnen Sie am besten mit dem Code, den Sie im entsprechenden Praktikum geschrieben haben. Haben Sie sich für die Ganzzahlen entschieden, verwenden Sie die ersten 20 Primzahlen als Eingabedaten.

Wenn Sie mit den Temperaturanomalien arbeiten, ist es sinnvoll, eine entsprechende Datenstruktur samt Lese- und Ausgabefunktionen in ein separates Paar aus Header- und Quelltextdatei auszulagern.

Die Aufgabenstellungen gehen davon aus, dass Sie Temperaturanomalien verwenden. Falls Sie Ganzzahlen verwenden, ersetzen Sie die jeweiligen Benennungen entsprechend: Beispielsweise `DynamicIntArray` statt `DynamicAnomalyArray`. Diese Benennungen sind allerdings nur Vorschläge.

2. Dynamisches Array anlegen

(a) Legen Sie in einem separaten Paar aus Header- und Quelltextdatei die Container-Datenstruktur für ein dynamisches Array an.

(b) Legen Sie die folgende Funktion an:

```
DynamicAnomalyArray construct_dynamic_anomaly_array(void)
```

Diese Funktion erzeugt ein leeres, dynamisches Array, das bereits eine kleine, vordefinierte Anzahl an Elementen aufnehmen kann.

(c) Legen Sie die folgende Funktion an:

```
void destruct_dynamic_anomaly_array(DynamicAnomalyArray* array)
```

Diese Funktion gibt den im Array reservierten Speicher wieder frei und setzt den Füllstand und die Kapazität auf null.

(d) Testen Sie Ihr Array, indem Sie eine erste Anomalie / eine erste Ganzzahl eintragen und diese wieder ausgeben.

3. Daten hinzufügen

(a) Legen Sie die folgende Funktion an:

```
void push_back_anomaly(DynamicAnomaliesArray* array, Anomaly* anomaly)
```

Diese Funktion fügt dem dynamischen Array ein Element hinzu und passt dabei den Füllstand an.

(b) Testen Sie Ihr Array, indem Sie mit der `push_back`-Funktion Daten ins Array einfügen, bis es gefüllt ist. Überschreiten Sie noch nicht die initiale Kapazität. Geben Sie die enthaltenen Daten aus.

4. Wachsen und weitere Daten hinzufügen

(a) Lassen Sie die push_back-Funktion überprüfen, ob das Array bereits vollständig gefüllt ist. Vergrößern Sie bei Bedarf das Array. Verwenden Sie dazu eine eigene Funktion, z.B.

```
void extend_dynamic_anomaly_array(DynamicAnomalyArray* array)
```

Verwenden Sie als neue Kapazität das Doppelte der vorherigen Kapazität.

(b) Testen Sie Ihr Array, indem Sie mit der push_back-Funktion Daten ins Array einfügen, bis Sie mehrfach die jeweilige Kapazität überschritten haben. Geben Sie die enthaltenen Daten aus. Lassen Sie die extend-Funktion das Wort extend an die Standardausgabe ausgeben, um zu prüfen, ob die Funktion oft genug aufgerufen wird. Vergessen Sie nicht, diese Debug-Ausgabe wieder zu entfernen.

III. WEITERE AUFGABEN

5. Datensatz finden

(a) Implementieren Sie folgende Funktion

```
unsigned int find_anomaly_index(DynamicAnomalyArray* array, int year)
```

Diese Funktion sucht die Anomalie des Jahres year im Array. Dazu iteriert die Funktion über die Arrayelemente und gibt bei einem Treffer den Index des gefundenen Elements zurück. Wird kein passendes Element gefunden, wird der Füllstand des Arrays zurückgegeben.

(b) Testen Sie die Suche, indem Sie den Index eines selbstgewählten Elements suchen lassen und diesen ausgeben.

6. Datensatz im Inneren löschen

(a) Implementieren Sie folgende Funktion

```
void erase_anomaly(DynamicAnomalyArray* array, unsigned int index)
```

Diese Funktion löscht das Element im Array, das durch index bezeichnet wird.

(b) Testen Sie die Funktion, indem Sie ein selbstgewähltes Element löschen und anschließend alle verbliebenen Arrayelemente ausgeben.

7. Datensatz im Inneren hinzufügen

(a) Implementieren Sie folgende Funktion

```
void insert_anomaly(DynamicAnomalyArray* array, unsigned int index,
                    Anomaly* anomaly)
```

Diese Funktion fügt die durch anomaly referenzierte Anomalie *vor* dem Element ein, das durch index bezeichnet wird.

(b) Testen Sie die Funktion, indem Sie ein selbstgewähltes Element an einer selbstgewählten Stelle einfügen und anschließend alle Arrayelemente ausgeben.

Denken Sie daran, das Array ggf. zu vergrößern.

IV. noch mehr Aufgaben

8. Verkettete Liste

Implementieren Sie dieselbe Funktionalität zusätzlich mit einer verketteten Liste. Testen Sie Ihre Implementierung.