Repetitorium - Informatik für Ingenieure

Klausur Repetitorium SoSe 22 - Probeklausur 3 (WiSe 20/21)

Prof. Dr.-Ing. Görschwin Fey

Auswertungsbericht für Matrikelnummer mat0043 erstellt am 28.6.2022 um 10:08:02 Uhr.

Übersicht

Aufgabe	Punkte	Ergebnis
Aufgabe 1	6	0.0
Aufgabe 2	8	0.0
Aufgabe 3	5	0.0
Aufgabe 4	4	0.0
Aufgabe 5	2	0.0
Aufgabe 6	2	0.0
Aufgabe 7	3	0.0
Aufgabe 8	3	0.0
Aufgabe 9	4	0.0
Aufgabe 10	4	0.0
Aufgabe 11	4	0.0
Aufgabe 12	5	0.0
Aufgabe 13	6	0.0
Aufgabe 14	4	0.0
Rechteck	8	0.0
Feld füllen	12	0.0
2D-Felder	25	0.0
Vektor	45	0.0
Summe	**150**	**0.0**

Notenschlüssel

Note	Prozent	Punktzahl
1.0	95 %	142.5
1.3	90 %	135.0
1.7	85 %	127.5
2.0	80 %	120.0
2.3	75 %	112.5
2.7	70 %	105.0
3.0	65 %	97.5
3.3	60 %	90.0
3.7	55 %	82.5
4.0	50 %	75.0
5.0	< 50 %	< 75.0

Ergebnis

Sie haben 0.0 Punkte erreicht und erhalten die Note 5.0. Dieses Klausurergebnis ist unverbindlich, bis die Bewertung in TUNE finalisiert wurde.

Auswertungsbericht für mat0043

Aufgabe 1

0.0 / 6 Punkten

Übersetzen Sie die vorzeichenlose Binärzahl `11000001` in die folgenden Zahlensysteme.

Sie erhalten für jede richtig übersetzte Punkte.

Dezimal	richtige Lösung ist 193	**0.00 Punkte**
Oktal	richtige Lösung ist 301	**0.00 Punkte**
Hexadezimal	richtige Lösung ist c1	**0.00 Punkte**

Erstellt am 28.6.2022 um 09:56:22 Uhr

Aufgabe 2

0.0 / 8 Punkten

Füllen Sie die Wahrheitstabelle für den folgenden Booleschen Ausdruck aus:

$$y = [(b \vee c) \wedge \neg(a \vee \neg c)] \bigoplus b$$

Es gilt die Notation aus der Vorlesung:

- $\vee := \mathrm{OR}$
- $\wedge := \mathrm{AND}$
- $\neg := \mathrm{NOT}$
- $\bigoplus := \mathrm{XOR}$

Nicht ausgefüllte Zeilen werden mit 0 Punkten bewertet. Falsch ausgefüllte Zeilen geben Punktabzug; Sie erhalten jedoch niemals weniger als 0 Punkte für diese Aufgabe.

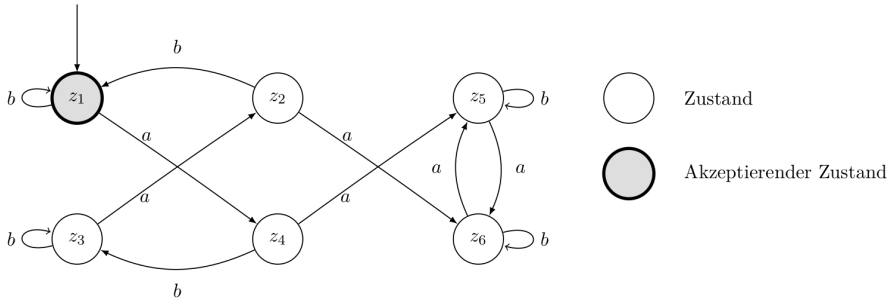
a	b	c	y	y (Lösung)	Bewertung
0	0	0		0	0.00 Punkte
0	0	1		1	0.00 Punkte
0	1	0		1	0.00 Punkte
0	1	1		0	0.00 Punkte
1	0	0		0	0.00 Punkte
1	0	1		0	0.00 Punkte
1	1	0		1	0.00 Punkte
1	1	1		1	0.00 Punkte

Aufgabe 3

0.0 / 5 Punkten

Welche Eingaben werden von folgendem endlichen Automaten akzeptiert?

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.



A ? ✓ ✗ abababab **0.00 Punkte**

B ? ✓ ✗ abbbbbabab **0.00 Punkte**

C ? ✓ ✗ aababb **0.00 Punkte**

D ? ✓ ✗ abbbbbabababab **0.00 Punkte**

E ? ✓ ✗ abaabab **0.00 Punkte**

Aufgabe 4

0.0 / 4 Punkten

Welche der nachfolgenden Aufgaben bzw. Komponenten sind Teil des Betriebssystems?

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.

A ? ✓ ✗ Compiler **0.00 Punkte**

B ? ✓ ✗ Task-Management **0.00 Punkte**

C ? ✓ ✗ Verbindungssystem **0.00 Punkte**

D ? ✓ ✗ Speicherverwaltung **0.00 Punkte**

E ? ✓ ✗ Linker **0.00 Punkte**

F ? ✓ ✗ Hardware-spezifischer Code für Treiber **0.00 Punkte**

G ? ✓ ✗ Dateiverwaltung **0.00 Punkte**

H ? ✓ ✗ Interrupt-Handling **0.00 Punkte**

Aufgabe 5

0.0 / 2 Punkten

Welche Aussagen über Funktionen sind korrekt?

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.

A ? ✓ ✗ Eine Funktion muss immer eine return-Anweisung enthalten. **0.00 Punkte**

B ? ✓ ✗ Eine rekursive Funktion mit weniger als drei Fallunterscheidungen terminiert immer. **0.00 Punkte**

C ? ✓ ✗ Felder bzw. Arrays können an Funktionen über einen Zeiger auf das erste Element übergeben werden. **0.00 Punkte**

D ? ✓ ✗ Eine return-Anweisung beendet die Ausführung einer Funktion. **0.00 Punkte**

Aufgabe 6

0.0 / 2 Punkten

Die Fehlersuche in Programmcode wird als "Debugging" bezeichnet. Bestimmen Sie, welche Aussagen korrekt bzw. inkorrekt sind.

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.

A ? ✓ ✗ Mit einem Debugger kann ein Programm „Schritt-für-Schritt” ausgeführt werden. **0.00 Punkte**

B ? ✓ ✗ Die Verwendung eines Debuggers macht die Programmausführung langsamer. **0.00 Punkte**

C ? ✓ ✗ Beim Debugging muss zunächst das fehlerhafte Verhalten reproduziert und nachvollzogen werden. **0.00 Punkte**

D ? ✓ ✗ Debugging wird beim Compilieren ausgeführt. **0.00 Punkte**

E ? ✓ ✗ Ein Debugger stoppt automatisch an der fehlerhaften Stelle im Programmcode. **0.00 Punkte**

F ? ✓ ✗ GCC ist ein weit verbreiteter Debugger. **0.00 Punkte**

G ? ✓ ✗ Ein Debugger kann nur auf fehlerhaften Programmen eingesetzt werden. **0.00 Punkte**

H ? ✓ ✗ Mit einem Debugger können die Werte von Variablen betrachtet werden. **0.00 Punkte**

Aufgabe 7

0.0 / 3 Punkten

Beim Aufruf einer Funktion sollte geprüft werden, ob die übergebenen Parameter sinnvolle Werte enthalten, so zum Beispiel, ob bei der Flächenberechnung einer geometrischen Form nur positive Werte für Seitenlängen übergeben wurden. Entscheiden Sie, ob folgende Aussagen korrekt sind.

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.

A ? ✓ ✗ Debugger helfen bei der Beseitigung von Laufzeit-Fehlern. **0.00 Punkte**

B ? ✓ ✗ Sind die Parameter-Werte sinnvoll, so liefert die Funktion ein korrektes Ergebnis, da keine weiteren Fehler enthalten sind. **0.00 Punkte**

C ? ✓ ✗ Mit einem Debugger lassen sich die tatsächlichen Parameter-Werte einer Funktion bei der Ausführung überprüfen. **0.00 Punkte**

D ? ✓ ✗ Ein nullptr zeigt auf zuvor allokierten Speicher. **0.00 Punkte**

E ? ✓ ✗ Es ist unnötig, die übergebenen Parameter auf sinnvolle Werte zu kontrollieren, wenn die Implementierung der Funktion mit Unittests überprüft wurde. **0.00 Punkte**

F ? ✓ ✗ In der aufgerufenen Funktion kann nur geprüft werden, ob die Werte der Parameter sinnvoll sind, aber nicht, ob sie korrekt sind. **0.00 Punkte**

Aufgabe 8

0.0 / 3 Punkten

Die Laufzeit- und Speicherkomplexität von Algorithmen wird mittels der O-Notation beschrieben. Entscheiden Sie, ob folgende Aussagen korrekt sind.

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.

A ? ✓ ✗ Falls $f(n) \in \mathcal{O}(n)$ und $g(n) \in \mathcal{O}(\log_2(n))$, so gilt $(f(n) \cdot g(n)) \in \mathcal{O}(n^2)$ **0.00 Punkte**

B ? ✓ ✗ Falls $f(n) \in \mathcal{O}(n)$ und $g(n) = 1024 \cdot n \cdot f(n)$, so gilt $g(n) \in \mathcal{O}(n^2)$. **0.00 Punkte**

C ? ✓ ✗ Falls $f(n) \in \mathcal{O}(n)$ und $g(n) \in \mathcal{O}(n)$, so gilt $(f(n) \cdot g(n)) \in \mathcal{O}(n)$. **0.00 Punkte**

D ? ✓ ✗ Falls $f(n) \in \mathcal{O}(n)$ und $g(n) \in \mathcal{O}(n^2)$, so gilt $(f(n) + g(n)) \in \mathcal{O}(n^2)$ **0.00 Punkte**

E ? ✓ ✗ Falls $f(n) \in \mathcal{O}(n)$ und $g(n) \in \mathcal{O}(n)$, so gilt $(f(n) + g(n)) \in \mathcal{O}(n)$. **0.00 Punkte**

F ? ✓ ✗ Es gilt $\mathcal{O}(log_2(n)) \subset \mathcal{O}(n^2)$. **0.00 Punkte**

G ? ✓ ✗ Falls $f(n) \in \mathcal{O}(n^3)$ und $g(n) = 3 \cdot f(n)$, so gilt $g(n) \in \mathcal{O}(n^2)$. **0.00 Punkte**

H ? ✓ ✗ Es gilt $\mathcal{O}(n \cdot \log_2(n)) \subset \mathcal{O}(\log_2(n))$ **0.00 Punkte**

I ? ✓ ✗ Es gilt $\mathcal{O}(n^4) \supset \mathcal{O}(n^2)$. **0.00 Punkte**

Aufgabe 9

0.0 / 4 Punkten

Welche Aussagen bezüglich des `delete`-Operators sind korrekt?

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.

A ? ✓ ✗ Der `delete`-Operator erhält als Argument eine Zeigervariable. **0.00 Punkte**

B ? ✓ ✗ Die Benutzung von `delete` ist für Variablen, die auf dem `Stack` liegen, notwendig. **0.00 Punkte**

C ? ✓ ✗ Wenn eine Variable mit `new` allokiert wurde, dann sollte mit `delete` der Speicher wieder freigegeben werden. **0.00 Punkte**

D ? ✓ ✗ Mit `delete` kann der Speicher von Feldern bzw. Arrays freigegeben werden. **0.00 Punkte**

Aufgabe 10

0.0 / 4 Punkten

Auf eine Memberfunktion oder -variable einer Klasse, die als `private` deklariert wurde, kann zugegriffen werden von:

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.

A ? ✓ ✗ Den Objekten, die in der gleichen Datei deklariert wurden **0.00 Punkte**

B ? ✓ ✗ Den Objekten der gleichen Klasse **0.00 Punkte**

C ? ✓ ✗ Dem entsprechenden Objekt selbst **0.00 Punkte**

D ? ✓ ✗ Den Objekten der abgeleiteten Klassen **0.00 Punkte**

Aufgabe 11

0.0 / 4 Punkten

Welche Aussagen über Listen sind korrekt?

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.

A ? ✓ ✗ Die Liste wird für eine feste Anzahl von Elementen instanziiert. **0.00 Punkte**

B ? ✓ ✗ Der Kopf der Liste ist ein Zeiger auf das erste Element. **0.00 Punkte**

C ? ✓ ✗ Jeder Knoten einer doppelt verketteten Liste ist mit dem Vorgänger und dem Nachfolger verknüpft. **0.00 Punkte**

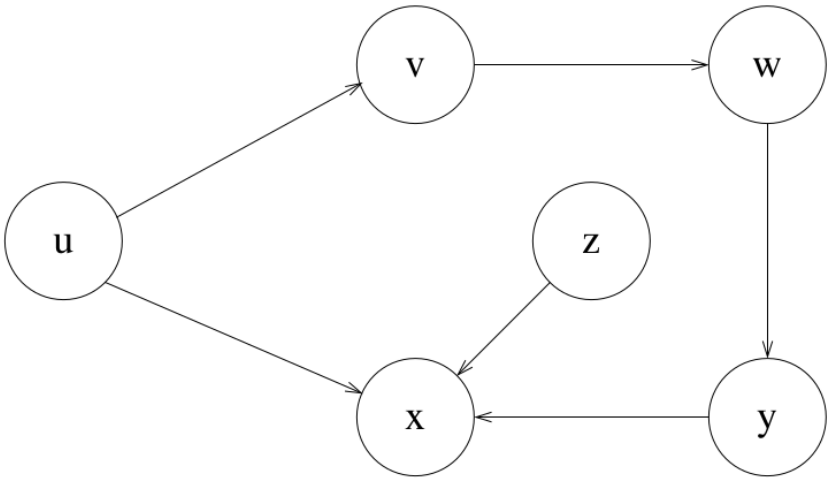
D ? ✓ ✗ Ein Element kann nach dem Einfügen entfernt werden. **0.00 Punkte**

Aufgabe 12

0.0 / 5 Punkten

Betrachten Sie folgenden gerichteten Graphen $G = (V, E)$ mit $V = \{u, v, w, x, y, z\}$. Entscheiden Sie, ob folgende Aussagen korrekt sind.

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.



A ? ✓ ✗ Eine von u startende Breitensuche besucht garantiert Knoten x vor Knoten y. **0.00 Punkte**

B ? ✓ ✗ Der Knoten x hat 3 ausgehende Kanten. **0.00 Punkte**

C ? ✓ ✗ Eine von u startende Tiefensuche besucht alle Knoten. **0.00 Punkte**

D ? ✓ ✗ Der Knoten z ist von keinem anderen Knoten aus erreichbar. **0.00 Punkte**

E ? ✓ ✗ Es existiert eine Kante (u, w). **0.00 Punkte**

Aufgabe 13

0.0 / 6 Punkten

Entscheiden Sie, ob folgende Aussagen bezüglich eines gerichteten Graphen G=(V, E) mit n = |V| und m = |E| korrekt sind.

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.

A ? ✓ ✗ $E \subseteq V \times V$ **0.00 Punkte**

B ? ✓ ✗ Eine Breitensuche erreicht immer alle Knoten eines Graphen. **0.00 Punkte**

C ? ✓ ✗ Jeder Graph hat mehr Kanten als Knoten. **0.00 Punkte**

D ? ✓ ✗ Graphen können zur Modellierung geometrischer Formen genutzt werden. **0.00 Punkte**

E ? ✓ ✗ Es gilt immer $m \leq n^2$. **0.00 Punkte**

F ? ✓ ✗ Falls es zwischen zwei Knoten u und v in einem Graphen einen Pfad gibt, so gibt es auch einen kreisfreien Pfad zwischen u und v. **0.00 Punkte**

Aufgabe 14

0.0 / 4 Punkten

Welche Aussagen sind korrekt?

Wählen Sie für jede Antwortmöglichkeit aus, ob diese richtig oder falsch ist. Für die richige Auswahl erhalten Sie Punkte. Wenn Sie eine falsche Auswahl treffen, dann reduziert dies Ihre Punktzahl. Antwortmöglichkeiten, für die Sie keine Auswahl treffen, werden mit 0 Punkten bewertet.

A ? ✓ ✗ Ein Klassendiagramm beinhaltet die Hierarchien zwischen den Klassen. **0.00 Punkte**

B ? ✓ ✗ Entwurfsmuster können benutzt werden, um wohldefinierte Lösungen auf eine Problemklasse anzuwenden. **0.00 Punkte**

C ? ✓ ✗ UML ist eine Programmiersprache, die von C++ abgeleitet ist. **0.00 Punkte**

D ? ✓ ✗ Entwurfsmuster sind Softwarepakete, die benutzt werden, um Programme bzw. Anwendungen daraus zusammenzusetzen. **0.00 Punkte**

Rechteck

0.0 / 8 Punkten

Aufgabenstellung

Für die Lösung der Aufgaben stehen Ihnen in der *rechteck.cpp*-Datei vorbereitete Funktionsrümpfe zur Verfügung.

Aufgaben

a. Implementieren Sie die Funktion `flaeche` in der Datei *rechteck.cpp*, die die Fläche eines Rechtecks zurückgibt. Die Zahlen dürfen dabei auch Kommazahlen sein. Ist eine der Seitenlängen kleiner oder gleich 0, soll die Funktion -1 zurückgeben.

b. Implementieren Sie die Funktion `umfang` in der Datei *rechteck.cpp*, die den Umfang eines Rechtecks zurückgibt. Die Zahlen dürfen dabei auch Kommazahlen sein. Ist eine der Seitenlängen kleiner oder gleich 0, soll die Funktion -1 zurückgeben.

Folgende Dateien wurden Ihnen während der Prüfung zur Verfügung gestellt:

main.cpp

```
#include <iostream>
#include <rechteck.hpp>

int main()
{
  float a = 1.0;

  float b = 2.0;

  std::cout << "Fläche des Rechtecks mit Seiten " << a << " und " << b << " ist " << flaeche(a,b) << std::endl;
  std::cout << "Umfang des Rechtecks mit Seiten " << a << " und " << b << " ist " << umfang(a,b) << std::endl;

  return 0;
}
```

rechteck.cpp

```
#include <iostream>
#include <rechteck.hpp>

float flaeche(float a, float b){

    return -3;

}

float umfang(float a, float b){

    return -3;

}
```

rechteck.hpp

```
#ifndef MINMAX_HPP
#define MINMAX_HPP

float flaeche(float a, float b);

float umfang(float a, float b);

#endif //MINMAX_HPP
```

Folgende Dateien haben Sie in der Prüfung bearbeitet bzw. erstellt:

main.cpp

```
#include <iostream>
#include <rechteck.hpp>

int main()
{
  float a = 1.0;

  float b = 2.0;

  std::cout << "Fläche des Rechtecks mit Seiten " << a << " und " << b << " ist " << flaeche(a,b) << std::endl;
```

```
  std::cout << "Umfang des Rechtecks mit Seiten " << a << " und " << b << " ist " << umfang(a,b) << std::endl;


  return 0;
}
```

rechteck.cpp

```
#include <iostream>
#include <rechteck.hpp>

float flaeche(float a, float b){

    return -3;

}

float umfang(float a, float b){

    return -3;

}
```

Bewertung mit Unittests

Testname	Assertions	Bestanden	Punkte
flaeche	1	0	0.00
flaecheFloat	1	0	0.00
flaecheSeiteKleinerNull	1	0	0.00
flaecheSeiteNull	1	0	0.00
umfang	1	0	0.00
umfangFloat	1	0	0.00
umfangSeiteKleinerNull	1	0	0.00
umfangSeiteNull	1	0	0.00

Feld füllen

0.0 / 12 Punkten

Aufgabenstellung

Die Fibonacci-Zahlen sind eine unendliche Folge von Zahlen, beginnend bei $f_0 = 0$ und $f_1 = 1$.
Diese ersten beiden Fibonacci-Zahlen sind per Definition vorgegeben. Alle weiteren Zahlen folgen dem Bildungsgesetz: $f_n=f_{n-1}+f_{n-2}$ mit $n \geq 2$. Die ersten 6 Zahlen der Fibonacci-Folge sind 0, 1, 1, 2, 3, 5.

Aufgabe

Implementieren Sie die Funktion `void feld_fibonacci(int feld[], int n)`, die die ersten *n* Zahlen der Fibonacci-Folge berechnet und in dem übergebenen Feld speichert. Ob Sie die Funktion rekursiv oder iterativ implementieren, bleibt Ihnen überlassen.

Folgende Dateien wurden Ihnen während der Prüfung zur Verfügung gestellt:

main.cpp

```
#include <iostream>
#include <feld_fibonacci.hpp>

using namespace std;

int main()
{
    cout << "Aufgabe - Feld_Fibonacci" << endl;
    cout << "-----------------------------------" << endl;

    unsigned int size = 10;
    int array[size];

    for(unsigned int i = 0; i<size ; i++){
        array[i] = 0;
    }

    feld_fibonacci(array, size);

    for(unsigned int i = 0; i<size ; i++){
        cout << array[i] << " ";
    }

    return 0;
}
```

feld_fibonacci.cpp

```
#include <iostream>
#include <feld_fibonacci.hpp>

void feld_fibonacci(int feld[], unsigned int n){

}
```

feld_fibonacci.hpp

```
#ifndef FIBO_HPP
#define FIBO_HPP

void feld_fibonacci(int feld[], unsigned int n);

#endif //FIBO_HPP
```

Folgende Dateien haben Sie in der Prüfung bearbeitet bzw. erstellt:

main.cpp

```
#include <iostream>
#include <feld_fibonacci.hpp>

using namespace std;

int main()
{
    cout << "Aufgabe - Feld_Fibonacci" << endl;
    cout << "-----------------------------------" << endl;

    unsigned int size = 10;
    int array[size];
```

Auswertungsbericht für mat0043

```
    for(unsigned int i = 0; i<size ; i++){
        array[i] = 0;
    }

    feld_fibonacci(array, size);

    for(unsigned int i = 0; i<size ; i++){
        cout << array[i] << " ";
    }

    return 0;
}
```

feld_fibonacci.cpp

```
#include <iostream>
#include <feld_fibonacci.hpp>

void feld_fibonacci(int feld[], unsigned int n){

}
```

Bewertung mit Unittests

Testname	Assertions	Bestanden	Punkte
fibonacci	1	0	0.00
fibonacci15	1	0	0.00
fibonacci9	1	0	0.00

Erstellt am 28.6.2022 um 09:56:22 Uhr

2D-Felder

0.0 / 25 Punkten

Aufgabenstellung

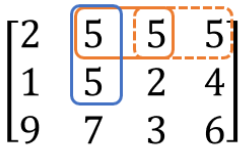
In dieser Aufgabe sollen Sie mit zweidimensionalen Feldern umgehen. Mit dem ersten Index des Felds wird die Zeile, mit dem zweiten Index des Felds die Spalte angegeben, in der das Element steht. Gehen Sie dabei davon aus, dass der Parameter `n` die Anzahl der Zeilen und der Parameter `m` die Anzahl der Spalten vorgibt.

Implementieren Sie die Funktionsrümpfe in der Datei `felder.cpp`. Wann immer eine der Dimensionen (`n` oder `m`) kleiner oder gleich `0` ist, sollen die Funktionen den Wert `-3` zurückgeben.

Aufgabe

1. Implementieren Sie die Funktion `int summe(int** feld, int n, int m)`, welche die Summe der Zahlen im übergebenen zweidimensionalen Feld berechnet und zurückgibt.
2. Implementieren Sie Funktionen, um die Anzahl an Paaren von Fünfen (zwei benachbarte Fünfen) in einem zweidimensionalen Feld zu finden.

   Beispiel:

   Horizontale Paare: 2
   Vertikale Paare: 1
   Gesamtzahl Paare: 3

   a. Implementieren Sie die Funktion `int paare_horizontal(int** feld, int n, int m)`, welche die Anzahl horizontaler Fünfer-Paare im Feld `feld` bestimmt und zurückgibt.

   b. Implementieren Sie die Funktion `int paare_vertikal(int** feld, int n, int m)`, welche die Anzahl vertikaler Fünfer-Paare im Feld `feld` bestimmt und zurückgibt.

   c. Implementieren Sie die Funktion `int paare_hor_ver(int** feld, int n, int m)`, welche die Anzahl aller Fünfer-Paare (horizontal und vertikal) im Feld `feld` bestimmt und zurückgibt.

Folgende Dateien wurden Ihnen während der Prüfung zur Verfügung gestellt:

main.cpp

```
#include <iostream>
#include "felder.hpp"

using namespace std;

int main()
{
   const int array_size = 5;

    cout << "Berechnung der Summe / Calculation of sum:" << endl;
    int* feld_ones[array_size];

    for(int i=0; i < array_size; i++){
        feld_ones[i] = new int[array_size];
        for(int j = 0; j < array_size;j++){
            feld_ones[i][j] = 1;
            cout << feld_ones[i][j] << " ";
        }
        cout << endl;
    }

    int sum = summe(feld_ones, array_size, array_size);

    for(int i=0; i < array_size; i++){
        delete[] feld_ones[i];
    }

    cout << "Summe/Sum: " << sum << endl;
    cout << "-----------------------------------" << endl;

    cout << "Paare suchen/Find pairs:" << endl;

    int feld1[] = {2, 5, 5, 5};
    int feld2[] = {1, 5, 2, 4};
    int feld3[] = {9, 7, 3, 6};

    int* feld[3];
    feld[0] = feld1;
    feld[1] = feld2;
    feld[2] = feld3;

    for(int i = 0; i < 3; i++) {
        for(int j = 0; j < 4;j++){
            cout << feld[i][j] << " ";
        }
        cout << endl;
```

```
    }

    cout << "Horizontale Paare / Horizontal pairs:     " << paare_horizontal(feld, 3, 4) << endl;
    cout << "Vertikale Paare / Vertical pairs:         " << paare_vertikal(feld, 3, 4) << endl;
    cout << "Gesamtzahl Paare / Total number of pairs: " << paare_hor_ver(feld, 3, 4) << endl;


    return 0;
}
```

felder.cpp

```
#include <iostream>
#include "felder.hpp"

/**
 * @brief Berechnet die Summe des Felds.
 *
 * @param feld Zeiger auf ein zweidimensionales Feld.
 * @param n Anzahl der Zeilen.
 * @param m Anzahl der Spalten.
 *
 * @return Das Ergebnis oder -3, wenn n <= 0 oder m <= 0.
 */
/**
 * @brief Calulates sum of array.
 *
 * @param feld Pointer to a two-dimensional array.
 * @param n Number of rows.
 * @param m Number of columns.
 *
 * @return The result or -3 if n <= 0 or m <= 0.
 */
int summe(int** feld, int n, int m){
    return 0;
}

/**
 * @brief Zählt die Anzahl der horizontal benachbarten Paare von Fünfen.
 *
 * @param feld Zeiger auf ein zweidimensionales Feld.
 * @param n Anzahl der Zeilen.
 * @param m Anzahl der Spalten.
 *
 * @return Das Ergebnis oder -3, wenn n <= 0 oder m <= 0.
 */
/**
 * @brief Counts the number of horizontal pairs of fives.
 *
 * @param feld Pointer to a two-dimensional array.
 * @param n Number of rows.
 * @param m Number of columns.
 *
 * @return The result or -3 if n <= 0 or m <= 0.
 */
int paare_horizontal(int** feld, int n, int m){
    return 0;
}

/**
 * @brief Zählt die Anzahl der vertikal benachbarten Paare von Fünfen.
 *
 * @param feld Zeiger auf ein zweidimensionales Feld.
 * @param n Anzahl der Zeilen.
 * @param m Anzahl der Spalten.
 *
 * @return Das Ergebnis oder -3, wenn n <= 0 oder m <= 0.
 */
/**
 * @brief Counts the number of vertical pairs of fives.
 *
 * @param feld Pointer to a two-dimensional array.
 * @param n Number of rows.
 * @param m Number of columns.
 *
 * @return The result or -3 if n <= 0 or m <= 0.
 */
int paare_vertikal(int** feld, int n, int m){
    return 0;
}

/**
 * @brief Zählt die Anzahl der benachbarten Paare von Fünfen.
 *
 * @param feld Zeiger auf ein zweidimensionales Feld.
 * @param n Anzahl der Zeilen.
 * @param m Anzahl der Spalten.
 *
 * @return Das Ergebnis oder -3, wenn n <= 0 oder m <= 0.
 */
/**
 * @brief Counts the overall number of pairs of fives.
 *
 * @param feld Pointer to a two-dimensional array.
 * @param n Number of rows.
 * @param m Number of columns.
 *
 * @return The result or -3 if n <= 0 or m <= 0.
```

```
 */

int paare_hor_ver(int** feld, int n, int m){
    return 0;
}
```

felder.hpp

```
#pragma once

int summe(int** feld, int n, int m);

int paare_horizontal(int** feld, int n, int m);

int paare_vertikal(int** feld, int n, int m);

int paare_hor_ver(int** feld, int n, int m);
```

Folgende Dateien haben Sie in der Prüfung bearbeitet bzw. erstellt:

main.cpp

```
#include <iostream>
#include "felder.hpp"

using namespace std;

int main()
{
   const int array_size = 5;

    cout << "Berechnung der Summe / Calculation of sum:" << endl;
    int* feld_ones[array_size];

    for(int i=0; i < array_size; i++){
        feld_ones[i] = new int[array_size];
        for(int j = 0; j < array_size;j++){
            feld_ones[i][j] = 1;
            cout << feld_ones[i][j] << " ";
        }
        cout << endl;
    }

    int sum = summe(feld_ones, array_size, array_size);

    for(int i=0; i < array_size; i++){
        delete[] feld_ones[i];
    }

    cout << "Summe/Sum: " << sum << endl;
    cout << "-----------------------------------" << endl;

    cout << "Paare suchen/Find pairs:" << endl;

    int feld1[] = {2, 5, 5, 5};
    int feld2[] = {1, 5, 2, 4};
    int feld3[] = {9, 7, 3, 6};

    int* feld[3];
    feld[0] = feld1;
    feld[1] = feld2;
    feld[2] = feld3;

    for(int i = 0; i < 3; i++) {
        for(int j = 0; j < 4;j++){
            cout << feld[i][j] << " ";
        }
        cout << endl;
    }

    cout << "Horizontale Paare / Horizontal pairs:     " << paare_horizontal(feld, 3, 4) << endl;
    cout << "Vertikale Paare / Vertical pairs:         " << paare_vertikal(feld, 3, 4) << endl;
    cout << "Gesamtzahl Paare / Total number of pairs: " << paare_hor_ver(feld, 3, 4) << endl;


    return 0;
}
```

felder.cpp

```
#include <iostream>
#include "felder.hpp"

/**
 * @brief Berechnet die Summe des Felds.
 *
 * @param feld Zeiger auf ein zweidimensionales Feld.
 * @param n Anzahl der Zeilen.
 * @param m Anzahl der Spalten.
```

```
 *
 * @return Das Ergebnis oder -3, wenn n <= 0 oder m <= 0.
 */
/**
 * @brief Calulates sum of array.
 *
 * @param feld Pointer to a two-dimensional array.
 * @param n Number of rows.
 * @param m Number of columns.
 *
 * @return The result or -3 if n <= 0 or m <= 0.
 */
int summe(int** feld, int n, int m){
    return 0;
}

/**
 * @brief Zählt die Anzahl der horizontal benachbarten Paare von Fünfen.
 *
 * @param feld Zeiger auf ein zweidimensionales Feld.
 * @param n Anzahl der Zeilen.
 * @param m Anzahl der Spalten.
 *
 * @return Das Ergebnis oder -3, wenn n <= 0 oder m <= 0.
 */
/**
 * @brief Counts the number of horizontal pairs of fives.
 *
 * @param feld Pointer to a two-dimensional array.
 * @param n Number of rows.
 * @param m Number of columns.
 *
 * @return The result or -3 if n <= 0 or m <= 0.
 */
int paare_horizontal(int** feld, int n, int m){
    return 0;
}

/**
 * @brief Zählt die Anzahl der vertikal benachbarten Paare von Fünfen.
 *
 * @param feld Zeiger auf ein zweidimensionales Feld.
 * @param n Anzahl der Zeilen.
 * @param m Anzahl der Spalten.
 *
 * @return Das Ergebnis oder -3, wenn n <= 0 oder m <= 0.
 */
/**
 * @brief Counts the number of vertical pairs of fives.
 *
 * @param feld Pointer to a two-dimensional array.
 * @param n Number of rows.
 * @param m Number of columns.
 *
 * @return The result or -3 if n <= 0 or m <= 0.
 */
int paare_vertikal(int** feld, int n, int m){
    return 0;
}

/**
 * @brief Zählt die Anzahl der benachbarten Paare von Fünfen.
 *
 * @param feld Zeiger auf ein zweidimensionales Feld.
 * @param n Anzahl der Zeilen.
 * @param m Anzahl der Spalten.
 *
 * @return Das Ergebnis oder -3, wenn n <= 0 oder m <= 0.
 */
/**
 * @brief Counts the overall number of pairs of fives.
 *
 * @param feld Pointer to a two-dimensional array.
 * @param n Number of rows.
 * @param m Number of columns.
 *
 * @return The result or -3 if n <= 0 or m <= 0.
 */

int paare_hor_ver(int** feld, int n, int m){
    return 0;
}
```

Bewertung mit Unittests

Testname	Assertions	Bestanden	Punkte
PaareGesamtBeispiel	1	0	0.00
PaareGesamtNKleinerNull	1	0	0.00
PaareGesamtNeueMatrix	1	0	0.00
PaareHorizontal	1	0	0.00
PaareHorizontalNKleinerNull	1	0	0.00
PaareVertikal	1	0	0.00
PaareVertikalNKleinerNull	1	0	0.00
Summe	1	0	0.00

Auswertungsbericht für mat0043

Testname	Assertions	Bestanden	Punkte
SummeGemischteZahlen	1	0	0.00
SummeNKleinerNull	1	0	0.00

Erstellt am 28.6.2022 um 09:56:22 Uhr

Vektor

0.0 / 45 Punkten

Aufgabenstellung

Motivation

Jede dynamische Speicherallokation benötigt etwas Laufzeit. Deshalb kann es sinnvoll sein, lieber wenige Speicherallokationen durchzuführen und dabei zunächst zuviel Speicher zu allokieren, als viele kleine Speicherbereiche zu allokieren. Dies wird zum Beispiel mit einem `std::vector` aus der Standard-Bibliothek realisiert. Eine ähnliche Funktionalität soll im Folgenden realisiert werden, allerdings ohne automatische Reallokation, falls das Feld zu klein oder zu groß wird, z.B. beim Aufruf von `push_back`.

Konzept

Die Klasse `static_vector` stellt ein Feld fester Größe zur Verfügung, welches `int`-Werte enthält. Der Nutzer muss sich im Gegensatz zur direkten dynamischen Allokation mittels Zeiger-Variablen nicht um die Speicherallokation und -freigabe kümmern, da dies von der Klasse übernommen werden soll.

Hinweise zur Implementierung

Ein `static_vector` hat eine maximale Größe für die Speicher allokiert ist und eine Anzahl tatsächlich bereits enthaltener Elemente. Die maximale Größe wird in der Membervariablen `memory_size` und die Anzahl tatsächlich bereits enthaltener Elemente in `num_elements` gespeichert. Der Speicher für das Feld selbst wird in der Membervariablen `memory` bereitgestellt.

Aufgaben

Implementieren Sie alle fehlenden Memberfunktionen. Die Funktionsrümpfe der Memberfunktionen sind bereits in der Datei `static_vector.cpp` verfügbar. Der Konstruktor ohne Argumente wurde bereits implementiert. Die Klasse hat folgende Memberfunktionen:

a. `static_vector()` ist bereits implementiert.

b. `static_vector(int max_elements)` allokiert `max_elements` in `memory` und initialisiert `memory_size` sowie `num_elements` entsprechend. Falls `max_elements` kleiner gleich 0 ist, wird die maximale Größe auf 0 gesetzt.

c. `~static_vector()` gibt den gesamten allokierten Speicher frei.

d. `int capacity() const` liefert die maximale Anzahl an Elementen, die gespeichert werden können.

e. `int size() const` liefert die Anzahl der enthaltenen Elemente.

f. `int& at(int pos)` liefert das Element mit dem Index `pos`. Die Memberfunktion führt eine Prüfung der Grenzen durch und wirft eine `std::out_of_range`-Exception, falls `pos` außerhalb der aktuellen Grenze von gespeicherten Elementen liegt. Die Indizes beginnen wie üblich mit 0.

g. `int& operator[](int pos)` liefert das Element mit dem Index `pos` aber ohne die Grenzen zu prüfen.

h. `void push_back(const int& value)` fügt den Wert `value` am Ende des Feldes an - also hinter dem letzten bisher genutzten Element. Falls kein Speicher mehr für das neue Element zur Verfügung steht, wird eine `std::length_error`-Exception ausgelöst.

i. `void pop_back()` entferent das letzte gespeicherte Element. Falls keine Elemente enthalten sind, wird eine `std::runtime_error`-Exception ausgelöst.

j. `int& back()` liefert das letzte gespeicherte Element. Falls keine Elemente enthalten sind, wird eine `std::runtime_error`-Exception ausgelöst.

k. `void reserve(int max_elements)` allokiert ein neues Feld der Größe `max_elements`, welches auch alle bisherigen Elemente enthält. Falls `max_elements` kleiner oder gleich der aktuellen Kapazität `memory_size` ist, wird nichts ausgeführt.

l. `void shrink_to_fit()` reduziert die maximale Größe auf die aktuelle Anzahl gespeicherter Elemente.

Folgende Dateien wurden Ihnen während der Prüfung zur Verfügung gestellt:

main.cpp

```
#include <static_vector.hpp>

#include <iostream>

int main() {
  static_vector a;
  try {
    a = static_vector(10);
```

```
    std::cout << "Capacity sollte 10 liefern. Liefert " << a.capacity() << "!" << std::endl;
    std::cout << "Size sollte 0 liefern. Liefert " << a.size() << "!" << std::endl;

  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Füge Elemente an: [3, 4]" << std::endl;
  try {
    a.push_back(3);
    a.push_back(4);
    std::cout << "Letztes Element sollte 4 sein. Liefert " << a.back() << "!" << std::endl;
  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Entferne letztes Element" << std::endl;
  try {
    a.pop_back();
    std::cout << "Back sollte 3 liefern. Liefert " << a.back() << "!" << std::endl;
  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Füge Elemente ein: [1, 2, 3, 4, 5]" << std::endl;
  try {
    a.push_back(1);
    a.push_back(2);
    a.push_back(3);
    a.push_back(4);
    a.push_back(5);
    std::cout << "Element an Position 0 sollte 3 sein. Liefert " << a.at(0) << "!" << std::endl;
    std::cout << "Size sollte 6 liefern. Liefert " << a.size() << "!" << std::endl;
  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Verkleinere Vektor auf die Anzahl der Elemente" << std::endl;
  try {
    a.shrink_to_fit();
    std::cout << "Capacity sollte 6 liefern. Liefert " << a.capacity() << "!" << std::endl;
  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Versuche push_back. Erwarte Exception." << std::endl;
  try {
    a.push_back(0);
    std::cout << "Keine Exception erhalten." << std::endl;
  } catch (...) {
    std::cout << "Exception erhalten." << std::endl;
  }
  std::cout << "Erweitere maximale Größe auf 12 Elemente" << std::endl;
  try {
    a.reserve(12);
    std::cout << "Capacity sollte 12 liefern. Liefert " << a.capacity() << "!" << std::endl;
  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Versuche push_back. Keine Exception erwartet." << std::endl;
  try {
    a.push_back(0);
    std::cout << "Keine Exception erhalten." << std::endl;
  } catch (...) {
    std::cout << "Exception erhalten." << std::endl;
  }

  return 0;
}
```

static_vector.cpp

```
#include <static_vector.hpp>

static_vector::static_vector() {
    num_elements = 0;
    memory_size = 0;
    memory = nullptr;
}

static_vector::static_vector(int max_elements) {
    throw std::runtime_error("Not implemented");
}

static_vector::~static_vector() {
}

int static_vector::capacity() const {
    throw std::runtime_error("Not implemented");
}

int static_vector::size() const {
    throw std::runtime_error("Not implemented");
}

int &static_vector::at(int pos) {
    throw std::runtime_error("Not implemented");
```

```
}


int &static_vector::operator[](int pos) {
    throw std::runtime_error("Not implemented");
}

void static_vector::push_back(const int &value) {
    throw std::runtime_error("Not implemented");
}

void static_vector::pop_back() {
    throw std::runtime_error("Not implemented");
}

int &static_vector::back() {
    throw std::runtime_error("Not implemented");
}

void static_vector::reserve(int max_elements) {
    throw std::runtime_error("Not implemented");
}

void static_vector::shrink_to_fit() {
    throw std::runtime_error("Not implemented");
}
```

static_vector.hpp

```
#pragma once

#include <stdexcept>

class static_vector {
public:
    static_vector();
    static_vector(int max_elements);
    ~static_vector();

    int capacity() const;
    int size() const;
    int& at(int pos);
    int& operator[](int pos);
    void push_back(const int& value);
    void pop_back();
    int& back();
    void reserve(int max_elements);
    void shrink_to_fit();

public:
    int num_elements;
    int memory_size;
    int* memory;
};
```

Folgende Dateien haben Sie in der Prüfung bearbeitet bzw. erstellt:

main.cpp

```
#include <static_vector.hpp>

#include <iostream>

int main() {
  static_vector a;
  try {
    a = static_vector(10);
    std::cout << "Capacity sollte 10 liefern. Liefert " << a.capacity() << "!" << std::endl;
    std::cout << "Size sollte 0 liefern. Liefert " << a.size() << "!" << std::endl;
  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Füge Elemente an: [3, 4]" << std::endl;
  try {
    a.push_back(3);
    a.push_back(4);
    std::cout << "Letztes Element sollte 4 sein. Liefert " << a.back() << "!" << std::endl;
  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Entferne letztes Element" << std::endl;
  try {
    a.pop_back();
    std::cout << "Back sollte 3 liefern. Liefert " << a.back() << "!" << std::endl;
  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Füge Elemente ein: [1, 2, 3, 4, 5]" << std::endl;
  try {
    a.push_back(1);
```

```
    a.push_back(2);
    a.push_back(3);

    a.push_back(4);
    a.push_back(5);
    std::cout << "Element an Position 0 sollte 3 sein. Liefert " << a.at(0) << "!" << std::endl;
    std::cout << "Size sollte 6 liefern. Liefert " << a.size() << "!" << std::endl;
  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Verkleinere Vektor auf die Anzahl der Elemente" << std::endl;
  try {
    a.shrink_to_fit();
    std::cout << "Capacity sollte 6 liefern. Liefert " << a.capacity() << "!" << std::endl;
  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Versuche push_back. Erwarte Exception." << std::endl;
  try {
    a.push_back(0);
    std::cout << "Keine Exception erhalten." << std::endl;
  } catch (...) {
    std::cout << "Exception erhalten." << std::endl;
  }
  std::cout << "Erweitere maximale Größe auf 12 Elemente" << std::endl;
  try {
    a.reserve(12);
    std::cout << "Capacity sollte 12 liefern. Liefert " << a.capacity() << "!" << std::endl;
  } catch (std::runtime_error& e) {
    std::cout << e.what() << std::endl;
  }

  std::cout << "Versuche push_back. Keine Exception erwartet." << std::endl;
  try {
    a.push_back(0);
    std::cout << "Keine Exception erhalten." << std::endl;
  } catch (...) {
    std::cout << "Exception erhalten." << std::endl;
  }

  return 0;
}
```

static_vector.cpp

```
#include <static_vector.hpp>

static_vector::static_vector() {
    num_elements = 0;
    memory_size = 0;
    memory = nullptr;
}

static_vector::static_vector(int max_elements) {
    throw std::runtime_error("Not implemented");
}

static_vector::~static_vector() {
}

int static_vector::capacity() const {
    throw std::runtime_error("Not implemented");
}

int static_vector::size() const {
    throw std::runtime_error("Not implemented");
}

int &static_vector::at(int pos) {
    throw std::runtime_error("Not implemented");
}

int &static_vector::operator[](int pos) {
    throw std::runtime_error("Not implemented");
}

void static_vector::push_back(const int &value) {
    throw std::runtime_error("Not implemented");
}

void static_vector::pop_back() {
    throw std::runtime_error("Not implemented");
}

int &static_vector::back() {
    throw std::runtime_error("Not implemented");
}

void static_vector::reserve(int max_elements) {
    throw std::runtime_error("Not implemented");
}

void static_vector::shrink_to_fit() {
    throw std::runtime_error("Not implemented");
}
```

Auswertungsbericht für mat0043

Bewertung mit Unittests

Testname	Assertions	Bestanden	Punkte
back	1	0	0.00
capacity	1	0	0.00
constructor	1	0	0.00
operator=	4	0	0.00
pop_back	1	0	0.00
push_back	1	0	0.00
reserve	1	0	0.00
shrink_to_fit	1	0	0.00
size	1	0	0.00

Erstellt am 28.6.2022 um 09:56:22 Uhr