technische universität dortmund

fakultät für informatik

Lutz Oettershagen
Jurij Kuzmic

Dortmund, den 17. Januar 2019

Übung zur Vorlesung EidP (WS 2018/19)

Blatt 11

Block grün

Es können 4 Punkte und 3 Bonuspunkte erreicht werden.

Abgabedatum: 24. Januar 2019, 23:59 Uhr

Hinweise

- Bitte beachten Sie die aktuellen Hinweise unter

  `https://ls11-www.cs.tu-dortmund.de/teaching/ep1819uebung/`

- Stellen Sie sicher, dass alle von Ihnen abgegebene Dateien reine Textdateien im UTF-8-Format sind.
- Für die Abgabe sind die Dateien `Aufgabe_11_1.txt`, `Aufgabe_11_2a.txt`, `Aufgabe_11_2b.cpp`, `Aufgabe_11_2c.cpp` und `Aufgabe_11_3.cpp` notwendig.
- Für die Kompilierung muss ebenfalls sichergestellt werden, dass sich Ihre Programme mit den Parametern `-pedantic` und `-Werror` kompilieren lassen.
- Für die Programmieraufgaben kopieren Sie immer die Ergebnisse als Block-Kommentar an das Ende der Datei, welche das jeweilige Hauptprogramm enthält.
- **Achtung:** Dies ist das letzte Aufgabenblatt im grünen Block und insgesamt das letzte Übungsblatt zur Vorlesung. Zum Bestehen dieses Blocks sind 8 Punkte erforderlich.

Aufgaben

Aufgabe 1: Grundlagen (1 Punkt)

Legen Sie für Ihre Antworten eine Text-Datei `Aufgabe_11_1.txt` an.

a) Skizzieren Sie mit den Schlüsselwörtern `try` und `catch` den typischen Ausnahmebehandlungsblock. (0.2 Punkte)

b) Welchen Vorteil bietet eine Ausnahmebehandlung gegenüber dem Zurückliefern eines Wertes, welcher einen Fehler darstellen soll? (0.2 Punkte)

c) Nennen Sie zwei Situationen, in denen eine Ausnahmebehandlung dem Beenden eines Programms vorzuziehen ist. (0.2 Punkte)

d) Wie sieht der `catch`-Block aus, der jede beliebige im vorherigen `try`-Block geworfene Ausnahme fängt? (0.2 Punkte)

e) Welche Voraussetzungen müssen vorliegen, damit eine Exception vom Typ `E` im *catch*-Block mit Parametertyp `H` gefangen wird? (0.2 Punkte)

Aufgabe 2: Ausnahmebehandlung (3 Punkte)

Legen Sie für Ihre Antworten die Dateien `Aufgabe_11_2a.txt`, `Aufgabe_11_2b.cpp` bzw. `Aufgabe_11_2c.cpp` an.

a) Um Polymorphie beim Fangen von Ausnahmen zu ermöglichen, haben Sie in der Vorlesung das Konzept *Throw by value, catch by reference* kennengelernt. Erklären Sie, warum Polymorphie beim Fangen von Ausnahmen ein sinnvolles Konzept ist. Welche Probleme können beim *catch by value* entstehen? Verdeutlichen Sie das Problem mit einem Beispiel, bestehend aus zwei Ausnahme-Klassen, einer `main` Funktion und einer Funktion, welche die Ausnahme mit `throw` auslöst.

Hinweis: Sie brauchen keine .cpp-Datei anzulegen; schreiben Sie Ihre Implementierung stattdessen in die Datei `Aufgabe_11_2a.txt`. Die Ausnahme-Klassen brauchen nur eine print-Funktion, um eine Ausgabe auf der Konsole zu ermöglichen. Um das Beispiel möglichst klein zu halten, brauchen Sie keine Konstruktoren zu implementieren. (1 Punkt)

b) Betrachten Sie das unten stehende Programm:

```
1  #include <iostream>
2  using namespace std;
3
4  int PosProdInt(unsigned int size, int const array[]) {
5    int posProd = 1;
6    for (unsigned int i = 0; i < size; ++i) {
7      if (array[i] > 0) posProd *= array[i];
8    }
9    return posProd;
10 }
11
12 double PosProdDbl(unsigned int size, const double array[]) {
13   double posProd = 1;
14   for (unsigned int i = 0; i < size; ++i) {
15     if (array[i] > 0) posProd *= array[i];
16   }
17   return posProd;
18 }
19
20 int main() {
21   int const iA[] = {1, -1, 7, -6, -3, 0};
22   double const dA[] = {1.3, -3.2, 0.1, -2.7, 0.4};
23   cout << PosProdInt(6, iA) << ' ' << PosProdDbl(5, dA) << endl;
24   return 0;
25 }
```

Offensichtlich existieren hier die zwei Funktionen `PosProdInt` und `PosProdDbl`, die zwar das gleiche Verhalten haben, jedoch auf verschiedenen Datentypen arbeiten.

Kommentieren und implementieren Sie mittels der Schablonentechnik aus C++ eine Funktionsschablone `PosProd`, die das Produkt aller positiven Elemente im übergebenen Feld zurückgibt und für jeden eingebauten numerischen Datentyp instantiiert werden kann. Ändern Sie das Hauptprogramm entsprechend ab, so dass es die neue Funktion verwendet. Kopieren Sie das berechnete Ergebnis als Blockkommentar ans Ende der Datei. (1 Punkt)

c) Ändern Sie die Funktionsschablone so, dass sie als Arraygröße eine Variable vom Datentyp `int` erhält, aber dann negative Arraygrößen und Arrays mit der Größe null durch den Mechanismus der Ausnahmebehandlung mit `try`, `throw` und `catch` behandelt. Verwenden Sie für jede Fehlerart eine eigene Fehlerklasse und einen eigenen `catch`-Block. Kopieren Sie das berechnete Ergebnis als Blockkommentar ans Ende der Datei. (1 Punkt)

Aufgabe 3: Zeiger (3 Bonuspunkte)

Die Bearbeitung dieser Aufgabe ist optional. Mit ihr können bis zu 3 Punkte (auch) für zurückliegende Aufgabenblätter erworben werden. Legen Sie für Ihre Antworten eine Textdatei `Aufgabe_11_3.cpp` an.
Wandeln Sie die unten angegebene `for`-Schleife in eine semantisch äquivalente `for`-Schleife um, in der die Indexzugriffe beim Array durch **Zeigeroperationen** ersetzt werden. **Es darf in der von Ihnen erstellten Funktion *keinen* Zugriff über Indizes geben!** Sie können davon ausgehen, dass das Array `t` genügend Speicherplatz für die Kopie bereitstellt.

```
1  void copy(char s[], char t[]) {
2    int i;
3    for(i = 0; s[i] != '\0'; ++i) {
4      t[i] = s[i];
5    }
6    t[i] = 0;
7  }
```