Grundlagen der Programmierung 2: Aufgabenblatt 4

Prof. Dr. Robert Gold

Technische Hochschule Ingolstadt
Sommersemester 2022

Aufgabe 1 Die Funktion `search` in folgendem unvollständigen Programm sucht den String `txt` in der Datei mit dem Namen `filename`. Falls der gesuchte String oder der Dateiname leer ist, wird ein char-Array geworfen. Ergänzen Sie den catch-Block.

```
#include <fstream>
#include <iostream>
#include <string>
using namespace std;

void search(const string filename, const string txt)
{
    if (filename.empty())
        throw "file name must be non-empty";
    if (txt.empty())
        throw "search string must be non-empty";

    ifstream infile(filename);
    string line;
    int line_nr = 0;

    while (infile.good())
    {
        getline(infile, line);
        line_nr++;

        if (line.find(txt) != string::npos)
        {
            cout.width(2);
            cout << line_nr << " : " << line << '\n';
        }
    }
```

```
    infile.close();
}

int main()
{
    try
    {
        search("DerWerwolf.txt", "wolf");
    }
    // hier den catch-Block einfuegen









}
```

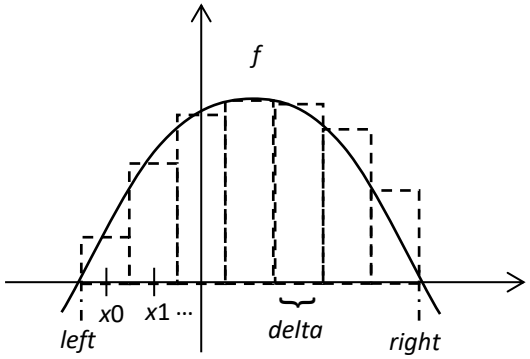
Im folgenden Programm wird eine invalid-argument-Exception geworfen. Ergänzen Sie den catch-Block.

```
#include <fstream>
#include <iostream>
#include <string>
using namespace std;

void search(const string filename, const string txt)
{
    if (filename.empty())
        //throw "file name must be non-empty";
        throw invalid_argument("file name must be non-empty");
    if (txt.empty())
        //throw "search string must be non-empty";
        throw invalid_argument("search string must be non-empty");

    ifstream infile(filename);
    string line;
    int line_nr = 0;

    while (infile.good())
    {
        getline(infile, line);
        line_nr++;

        if (line.find(txt) != string::npos)
        {
            cout.width(2);
            cout << line_nr << " : " << line << '\n';
        }
    }
```

```
    infile.close();
}

int main()
{
    try
    {
        search("DerWerwolf.txt", "wolf");
    }
    // hier den catch-Block einfuegen









}
```

Aufgabe 2 (nach einer Prüfungsaufgabe SS 2018)

In dieser Aufgabe soll ein Programm zur Berechnung eines bestimmten Integrals einer Funktion *f*: double → double erstellt werden.



Zur Berechnung soll das Riemann-Verfahren verwendet werden (Bezeichnungen siehe Abb.):

$$\int_{left}^{right} f(x)dx = \sum_{i=0}^{steps-1} f(xi) \cdot delta$$

Das Berechnungsintervall ist *delta* = (*right* – *left*) / *steps*. Die x-Werte *xi* liegen jeweils in der Mitte eines *delta*-Intervalls.

Wenn *steps* ≤ 0 oder *right* < *left* sollen Exceptions mit aussagekräftigen Fehlermeldungen geworfen werden.

a) Erstellen Sie eine Exception-Klasse `IntegralException` mit Basisklasse `exception`, die ein String-Attribut `message`, einen Konstruktor mit einem String-Parameter und einen Getter für das Attribut besitzt!

b) Erstellen Sie eine Funktion `integrate`.

- Die Methode soll die folgenden vier Parameter haben:
  - Integrationsgrenzen `left` und `right`
  - Anzahl der Schritte `steps`
  - die zu integrierende Funktion `f`
    Die Funktion soll als Funktionspointer übergeben werden.
- Der Rückgabewert ist der Integralwert als double.
- Die `IntegralExceptions` sollen an den Aufrufer weitergereicht werden.

c) Erstellen Sie eine Funktion `poly` mit einem double-Parameter zur Berechnung des Polynoms $-2x^2 + 2$.

d) Erstellen Sie eine main-Funktion, die unter Verwendung der Funktion `integrate` aus Teilaufgabe b) das Integral der Funktion `poly` aus Teilaufgabe c) von –1 bis 1 in 100 Schritten berechnet und auf der Konsole ausgibt!

e) In der Prüfung sollte Funktion `integrate` nicht mit der Funktion `poly` sondern mit einem Lambda-Ausdruck aufgerufen werden. Die Funktion `poly` wird dann gar nicht benötigt.