Übung zur Vorlesung Programmieren 2
SoSe 2024

# Blatt 2

Datentypen, Kontrollstrukturen, Funktionen, Mehrere Dateien

| | |
|---|---|
| Erfordert die VL bis Kapitel | Mehrere Dateien |
| Abgabe bis | 30. April 2024, 16:00 Uhr |

**Richtlinien für Abgaben:**

Lösungen, die diese Richtlinien nicht erfüllen, werden mit *0 Punkten* gewertet!
Abgegeben werden alle vorgegebenen, oder für die Aufgaben neu erstellten, Code/Text-Dateien.
Abgaben erfolgen über den Git-Server (↗) des Instituts.
Jeder in der Gruppe ist für die (pünktliche) Abgabe mitverantwortlich. Dafür gilt nicht die Commit-Zeit, sondern der Push-Zeitpunkt auf dem Server.

In Git ...
- wird für jedes Übungsblatt ein neuer Ordner angelegt: *Gruppe_XxYZ/blatt1*, */blatt2*, */blatt3*, ...
- dürfen keine temporären/automatisch generierten Dateien hinzugefügt werden (→ *.gitignore*).
- wird nur auf den Server gepusht, wenn der Code im aktuellen Commit lauffähig ist.
  Nicht lauffähige lokale Commits (für Zwischenstände) sind OK und sogar empfohlen. Diese sollten allerdings mit `"WIP:"` (work in progress) am Anfang der Commit-Nachricht gekennzeichnet werden.

Es wird kein vollständiger Code-Style vorgegeben, aber ...
- innerhalb jeder Abgabe muss die Code-Formatierung konsistent sein.
- es wird sinnvoll und konsistent eingerückt.
- Variablen haben aussagekräftige Namen.
- Compiler-Warnungen im *eigenen* Code, die in der Konsole ausgegeben werden, gelten als Fehler.

**Allgemeine Infos:**

Die Aufgaben und Erklärungen orientieren sich an (C++17) CMake-Projekten in der CLion IDE (↗), für Studenten kostenlos verfügbar für Windows, macOS und Linux.

Richtlinien & Infos werden bei Bedarf angepasst/erweitert. Sie gelten entsprechend je Übungsblatt.

---

**Hinweise zu Blatt 2:**

*Feiertag:* Wegen dem Aufschub für die Präsentationen von Blatt 1 verschieben sich auch die Präsentationen für Blatt 2 entsprechend um eine Woche. Das betrifft jedoch nicht die Abgabefrist.

## Aufgabe 1

2 Punkte

> Banknoten haben verschiedene Sicherheitsmerkmale. Eines davon ist die Seriennummer, bekannt aus den unzähligen Banküberfällen der Hollywood-Studios, zur Identifizierung der berüchtigten "markierten Scheine". Aber auch gegen Fälschungen helfen Seriennummern, da anhand einer enthaltenen Prüfziffer berechnet werden kann, ob es sich um eine gültige Seriennummer handelt.

Entpackt und öffnet das C++-Projekt im Verzeichnis *Repo/blatt2*.
Implementiert die Funktion `inputNextSerialNumber()` und ruft sie entsprechend auf.

**Abgabe/Präsentation:**

1. *main.cpp*:
   - Implementierung und Aufruf von `inputNextSerialNumber()`

## Aufgabe 2

3 Punkte

> Zu beachten ist, dass es zwei Versionen von Euro-Banknoten gibt: die ursprünglichen Scheine von 2002, und die neueren/sichereren Scheine, die seit 2013 gedruckt werden.

Deklariert (*euroCheck.h*) und implementiert (*euroCheck.cpp*) die Funktion
`getEuroSerialNumberVersion(serialNumber)`.

**Abgabe/Präsentation:**

1. *euroCheck.h/cpp*:
   - `getEuroSerialNumberVersion(serialNumber)`

## Aufgabe 3

6 Punkte

Inkludiert die Funktionen aus den *euroCheck.h/cpp*-Dateien in *main.cpp* und implementiert die Funktion `checkSerialNumberAndPrintResult(serialNumber)`.

Definiert dazu in *euroCheck.cpp* die Funktionen `checkEuroSerialNumber2013(serialNumber)` und `checkEuroSerialNumber2002(serialNumber)` zunächst ohne echten Inhalt (`{return true;}`).

**Abgabe/Präsentation:**

1. *main.cpp*:
   - Korrekte `include`-Anweisungen
   - `checkSerialNumberAndPrintResult(serialNumber)`
2. Das Programm gibt die korrekte Version der Euro-Banknote aus und ruft die entsprechende Funktion (`checkEuroSerialNumber2013/2002(serialNumber)`) auf.

## Aufgabe 4

2 Punkte

Vervollständigt die Funktion `main()`.

**Abgabe/Präsentation:**

1. *main.cpp*:
   - `main()`
2. Das Programm nimmt endlos neue Seriennummern entgegen und gibt jeweils die Version aus.
3. Das Programm endet durch die Eingabe von `"fertig"` statt einer Seriennummer.

## Aufgabe 5

8 Punkte

> Eine ausführlichere Beschreibung der Berechnung der Prüfziffer findet Ihr auf der Webseite `https://www.geldschein.at/euro-banknoten/euro_seriennummer.html`.

Implementiert die Prüfung der Seriennummer einer 2002er Euro-Banknote in der Funktion `checkEuroSerialNumber2002(serialNumber)`.

– Der Buchstabe wird durch die Zahl seiner Position im Alphabet ersetzt (also A=1, Z=26).
– Aus der ermittelten Zahl und der 10stelligen Nummer wird eine Quersumme gebildet.
– Die Quersumme wird durch 9 dividiert und der Rest aus dieser Division ermittelt.
– Dieser Rest wird nun von 8 abgezogen, das Ergebnis ergibt die Prüfziffer.
– Wenn die Subtraktion 0 ergibt ist die Prüfziffer 9.

**Abgabe/Präsentation:**

1. *euroCheck.cpp*:
   - `checkEuroSerialNumber2002()`
2. Das Programm gibt für echte 2002er Euro-Banknoten `"Gueltig"` und sonst `"Ungueltig"` aus.

## Aufgabe 6

8 Punkte

Implementiert die Prüfung der Seriennummer einer 2013er Euro-Banknote in der Funktion `checkEuroSerialNumber2013(serialNumber)`.

– Die 2 Buchstaben werden durch die Zahl ihrer Reihenfolge im Alphabet ersetzt (also A=1, Z=26).
– Aus den zwei ermittelten Zahlen und der 9stelligen Nummer wird eine Quersumme gebildet.
– Die Quersumme wird durch 9 dividiert und der Rest aus dieser Division ermittelt.
– Dieser Rest wird nun von 7 abgezogen, das Ergebnis ergibt die Prüfziffer.
– Wenn die Subtraktion 0 ergibt ist die Prüfziffer 9, ergibt die Subtraktion -1 ist die Prüfziffer 8.

**Abgabe/Präsentation:**

1. *euroCheck.cpp*:
   - `checkEuroSerialNumber2013()`
2. Das Programm gibt für echte 2013er Euro-Banknoten `"Gueltig"` und sonst `"Ungueltig"` aus.

## Aufgabe 7 (Bonus) (4 Punkte)

Stellt sicher, dass Seriennummern mit falschem Format (entsprechend der Berechnungsvorschrift) nicht als eine der beiden Versionen (2002/2013) erkannt werden, euer Programm also immer `"Ungueltig"` ausgibt und nicht abstürzt.

---

Gesamt: 29 Punkte