Imperative Programmierung
–Aufgabenblatt 03 (22 + 2 Punkte)–

Hinweise

Die Lösungen der Aufgaben sind als PDF-Dokument bzw. C-Quelltext mit Hilfe des Versionskontrollsystems Subversion (SVN) abzugeben. Platzieren Sie das PDF-Dokument mit ihren Antworten im Ordner `a03` innerhalb Ihres Gruppenverzeichnisses[1]. Platzieren Sie die C-Quelltexte im Unterordner `a03/src`. C-Quelltexte müssen fehlerfrei mit den Optionen `-pedantic -Wall -Werror -std=c99` kompiliert werden können. Schreiben Sie in die abgegebenen Dateien die Namen und Matrikelnummern aller Gruppenmitglieder. Verspätete Abgaben oder Abgaben ohne Matrikelnummer werden nicht gewertet! Plagiate jeglicher Art führen zur Meldung beim Prüfungsausschuss und zum nicht bestehen des Moduls.

Ausgabe: 01.11.2024
Abgabe: 17.11.2024 bis 23:59 Uhr

Aufgabe 1 - Einfache Programme (7 Punkte)

Implementieren Sie die folgenden Programme! Erstellen sie zu jedem der Programme eine Tabelle mit 2 Eingaben, die unterschiedliche Ausgaben erzeugen.

(a) Schreiben Sie ein Programm `triangle.c`, das drei Werte vom Typ double einliest. Das Programm soll überprüfen, ob diese Werte als Seitenlängen eines Dreiecks dienen können. Jeder Wert darf dafür nicht größer oder gleich der Summe der anderen beiden sein. Ist dies nicht der Fall, soll das Programm 1 ausgeben, andernfalls 0. (2 Punkte)

(b) Implementieren Sie ein Programm `are_zero.c`, das drei Ganzzahlen einliest. Es soll ausgeben, ob mindestens zwei der eingelesenen Werte ungleich 0 sind. (2 Punkte)

(c) Erstellen Sie ein Programm `odd_zero.c`, das drei Ganzzahlen einliest und ausgibt, ob eine ungerade Anzahl der eingelesenen Werte ungleich Null ist. (3 Punkte)

Aufgabe 2 - FizzBuzz (4 Punkte)

Schreiben Sie ein Programm `fizzbuzz.c`, das die Zahlen angefangen bei 1 bis einschließlich 100 ausgibt! Für Vielfache von 3 soll statt der Zahl die Zeichenkette „Fizz“ ausgegeben werden, für Vielfache von 5 soll statt der Zahl die Zeichenkette „Buzz“ ausgegeben werden! Für Vielfache von 3 und 5 soll statt der Zahl die Zeichenkette „FizzBuzz“ ausgegeben werden! Ihre Ausgabe sollte wie folgt aussehen:

```
1
2
Fizz
4
Buzz
...
14
FizzBuzz
16
17
...
```

Aufgabe 3 - Palindrome (5 Punkte)

Schreiben Sie ein Programm `palindrome.c`, das eine Zahl einliest und ausgibt, ob diese Zahl ein Palindrom ist. Palindrome sind in unserem Falle Zahlen, die von hinten und vorne gelesen gleich sind, wie zum Beispiel „4224“, „424“ oder „2“.

Bonus: Häufig ist eine Optimierung möglich. Optimieren sie den Algorithmus so, dass keine Berechnungen doppelt ausgeführt werden! (2 Bonuspunkte)

[1] Ihr Gruppenverzeichnis ist unter `https://svn.informatik.uni-rostock.de/lehre/ip2024/groups/<group>/` mit <group> $\in \{01, ..., 57\}$ zu finden.

Aufgabe 4 - Primzahlen (6 Punkte)

Eine Primzahl ist eine natürliche Zahl größer eins, die ohne Rest lediglich durch sich selbst und durch eins teilbar ist.

(a) Schreiben Sie ein Programm `is_prime.c` das eine natürliche Zahl einliest und überprüft, ob es sich um eine Primzahl handelt. (2 Punkte)

(b) Schreiben Sie ein Programm `prime_factor.c` das eine natürliche Zahl einliest und die Primfaktorzerlegung dieser Zahl ausgibt. (3 Punkte)

(c) Implementieren Sie ein Programm `prime_analysis.c` das eine Zahl vom Benutzer einliest. Das Programm soll dann überprüfen, ob die eingegebene Zahl eine Primzahl ist. Ist dies der Fall, gibt das Programm eine Meldung aus, dass es sich um eine Primzahl handelt. Andernfalls soll das Programm die Primfaktorzerlegung der Zahl ausgeben. (1 Punkt)

Beispiel: Für die Eingabe `20` soll das Programm `2 * 2 * 5` ausgeben, für die Eingabe `19` aber ausgeben, dass es sich um eine Primzahl handelt.