Programmiertechnik II

Sommersemester 2017 – Aufgabenblatt 1

Hasso-Plattner-Institut
Fachgebiet Computergrafische Systeme

Anmerkungen zu Aufgabenblatt 1

Der zu verwendende Programmrahmen steht als Zip-Archiv `uebung1.zip` im Moodle unter der Adresse `https://moodle.hpi3d.de/course/view.php?id=96` zum Download zur Verfügung.

Allgemeine Hinweise
Versuchen Sie als erstes, den Programmrahmen zu verstehen. Ermitteln Sie dann im Programmrahmen die Stellen, die Sie selbst implementieren sollen, um die Aufgaben korrekt zu lösen; diese Stellen sind meist besonders gekennzeichnet. Kommentieren Sie bei Bedarf bitte Ihren Quellcode.

Bevor Sie mit der Lösung beginnen, lesen Sie das gesamte Aufgabenblatt und überprüfen Sie, ob der Programmrahmen auf Ihrem System fehlerfrei kompiliert und alle Programme ausführbar sind. Nehmen Sie die Übungstermine wahr.

Zielsetzung
Ziel dieses Aufgabenblattes ist es, sich mit der elementaren C++-Programmierung sowie mit CMake als Build-System und einem C++-Compiler Ihrer Wahl grundlegend vertraut zu machen.

Viel Spaß und Erfolg bei der Bearbeitung der Aufgaben!

Aufgabe 1.1: Ausgabe von Dreieckszahlen (10 Punkte 2+6+2)

Gegeben ist eine Formel zur Berechnung der Summe aller positiven natürlichen Zahlen von 1 bis zu einer Obergrenze n („Dreieckszahlen“):

$$\Delta(n) = \sum_{i=1}^{n} i, n \in \mathbb{N}^{+} \tag{1}$$

a) Das Programm *triangular* soll für einen Eingabeparameter n die Summe unter Anwendung von Formel 1 ausgeben. Vervollständigen Sie dazu die Funktion `triangular()` in `triangular.cpp` und geben Sie die Summe auf der Konsole via `std::cout` in `main()` aus.

b) Überprüfen Sie, ob die Eingabe im gültigen Definitionsbereich für Werte vom Typ `int` (also `signed int`) liegt. Der gültige Definitionsbereich für Eingabewerte (n) ist der Bereich, für den die korrespondierenden Dreieckszahlen mit `signed int` abgebildet werden können. Sofern die Eingabe ausserhalb dieses Definitionsbereiches liegt, geben Sie den gültigen Definitions- und Wertebereich von Δ auf der Konsole im folgenden Format aus: `domain = [<dmin>;<dmax>], codomain = [<cdmin>;<cdmax>]`.

c) Verbessern Sie die Lesbarkeit der Ausgabe aller Dezimalzahlen (Dreieckszahlen sowie Definitionsbereich), indem Sie deren Formatierung eigenständig um Dezimaltrennzeichen erweitern. Beispielausgabe für $n = 12345$ ist `76.205.685`. Implementieren und nutzen Sie dazu die Funktion `pretty_print()`, welche ihren Eingabeparameter entsprechend formatiert auf der Konsole ausgeben soll.

Aufgabe 1.2: Berechnung von Fibonacci-Zahlen (10 Punkte 3+2+2+3)

In der Fibonacci-Folge ergibt die Summe zweier benachbarter Zahlen die unmittelbar folgende Zahl. Jede Zahl der Folge kann mit einem rekursiven Bildungsgesetz berechnet werden:

$$\mathrm{fib}\,(n) = \begin{cases} 1, & n \leq 2 \\ \mathrm{fib}(n-1) + \mathrm{fib}(n-2), & n > 2 \end{cases}, n \in \mathbb{N}^+ \tag{2}$$

a) Implementieren Sie die Funktion `fibonacci()` in `fibonacci_recursive.cpp` zur Berechnung der n-ten Fibonacci-Zahl. Verwenden Sie dazu die *rekursive* Formel 2. Achten Sie darauf, dass die Eingabeparameter für Anfragen von ungültigen oder mittels `signed int` nicht ausdrückbaren Fibonacci-Zahlen abgefangen werden, indem Ihre Berechnungen frühzeitig 0 zurückgeben.

b) Implementieren Sie die Funktion `fibonacci()` auf Basis eines iterativen Berechnungsschemas in `fibonacci_iterative.cpp`. Fangen Sie auch hier ungültige Eingaben ab.

c) Ergänzen Sie die Ausgabe beider Programme um die Anzahl der Berechnungsschritte (Summierungen) im folgenden Format: `<n> : <fib(n)> : #<number_of_summations>`. Die Ausgabe für $n = 8$ wäre beispielsweise in der rekursiven Variante `8 : 21 : #41`.

d) Gegeben sei eine Treppe mit n Stufen. Um diese Treppe hinauf (oder herab) zu steigen, können je Schritt eine oder zwei Stufen genommen werden. Implementieren Sie aufbauend auf Ihren Erkenntnissen über die Fibonacci-Folge die Funktion `combinations()` in `walking_stairs.cpp` zur Berechnung der Anzahl aller möglichen Schrittkombinationen, um eine Treppe mit n Stufen hinauf oder herab zu steigen. Erweitern Sie dabei die *gesamte* Funktion so, dass eine *schnelle* Berechnung für 92 Stufen möglich ist.
Hinweis: Machen Sie sich dazu mit den Grunddatentypen von C++ vertraut.

Zusatzaufgabe: Wechselgeldrückgabe (3 Bonuspunkte)

Für die Wechselgeldrückgabe eines automatischen Kassensystems soll die Differenz zwischen dem gezahlten und dem zu zahlenden Betrag immer über die kleinstmögliche Anzahl von Münzen und Scheinen erfolgen. Für die Aufgabe wird vereinfachend angenommen, dass stets genügend Münzen und Scheine im System enthalten sind.

Implementieren Sie die Methode `change` in `change.cpp`, welche für die Eingabeparameter `due` (zu zahlender Betrag) und `paid` (gezahlter Betrag) ermittelt. Die Ausgabe gibt dabei den Cent-Betrag der Münze bzw. des Scheins und die ermittelte Anzahl an. Dabei sollen ausschließlich Münzen und Scheine bis 50 Euro unterstützt werden.

Zur Vereinfachung sollen alle Euro-Beträge in Cent kodiert werden, d. h. ein zu zahlender Betrag von 7,21 Euro entspricht 721 bzw. ein 10-Euro-Schein entspricht 1000.

Die Konsolenausgabe soll im CSV-Format (Comma Separated Values) erfolgen. Für einen zu zahlenden Betrag von 0,89 Euro und einem gezahlten Betrag von 10,00 Euro (`change 89 1000`) wird beispielsweise ausgegeben:

```
coin,num
500,1
200,2
10,1
1,1
```

Die Ausgabe soll mit der Option `-o <Filename>` in eine Textdatei geschrieben werden. Sorgen Sie dafür, dass Ihr Programm alle ungültigen Eingaben ohne Ausgabe abfängt.

Allgemeine Hinweise zur Bearbeitung und Abgabe

- Die Aufgaben sollen maximal zu zweit bearbeitet werden; Ausnahmen sind nicht vorgesehen. Je Gruppe ist nur eine Abgabe notwendig.
- Bonuspunkte können innerhalb des Übungsblattes verlorene Punkte ausgleichen. Sie sind nicht auf andere Übungsblätter übertragbar und werden nicht über die reguläre Punktzahl hinaus angerechnet.
- Bitte reichen Sie Ihre Lösungen bis spätestens **Montag, den 8. Mai um 17:00 Uhr** ein.
- Die Implementierung kann auf einer üblichen Plattform (Windows, Linux, OS X) erfolgen, darf aber keine plattformspezifischen Elemente enthalten, d. h. die Implementierung soll plattformunabhängig entwickelt werden. Lesen Sie bei Fragen zum Kompilieren und Ausführen bitte genau die dem Programmrahmen beiliegende `README.TXT`.

  Bestehen weitere Fragen und Probleme, kontaktieren Sie den Übungsleiter oder nutzen Sie das Forum im Moodle.
- Archivieren Sie zur Abgabe Ihren bearbeiteten Programmrahmen als Zip-Archiv und ergänzen Sie Ihre Namen im Bezeichner des Zip-Archivs entsprechend im folgenden Format: **`uebung1_vorname1_nachname1_vorname2_nachname2.zip`**. Beachten Sie, dass dabei nur die vollständigen Quelltexte, die CMake-Konfiguration sowie eventuelle Zusatzdaten gepackt werden (alle Dateien, die im gegebenen Programmrahmen vorhanden waren). Laden Sie keine Kompilate und temporären Dateien (*.obj, *.pdb, *.ilk, *.ncb etc.) hoch. Testen Sie vor dem Hochladen, ob die Abgabe fehlerfrei kompiliert und ausgeführt werden kann.
- Reichen Sie Ihr Zip-Archiv im Moodle ein:
  `https://moodle.hpi3d.de/course/view.php?id=96`.