Universität Augsburg
Institut für Informatik

Übung zur Vorlesung Informatik 1 WiSe 2022/2023
Fakultät für Angewandte Informatik
Institut für Informatik
Dr. M. Frieb, Dr. J. Mische, V. Schmid, L. Petrak, P. Schalk 09.12.2022

Übungsblatt 8

Abgabe: 19.12.2022 9:00 Uhr

- Dieses Übungsblatt muss im Team abgegeben werden (Einzelabgaben sind nicht erlaubt!).
- Die **Zeitangaben** geben zur Orientierung an, wie viel Zeit für eine Aufgabe später in der Klausur vorgesehen wäre; gehen Sie davon aus, dass Sie zum jetzigen Zeitpunkt wesentlich länger brauchen und die angegebene Zeit erst nach ausreichender Übung erreichen.

* leichte Aufgabe / ** mittelschwere Aufgabe / *** schwere Aufgabe

Achtung: Um an der Klausur für die Veranstaltung Informatik 1 teilnehmen zu können, ist zwingend eine Anmeldung im System Studis notwendig.[1]
Die Studis-Anmeldephase beginnt am 05.12.2022 um 12:00 Uhr und endet am 19.12.2022 um 12:00 Uhr. Melden Sie sich rechtzeitig zur Prüfung an, um Komplikationen bei der Anmeldung zu vermeiden.

Offener Zoom-Arbeitsraum (OZR):
Da zwischenzeitlich ein falscher Link zum offenen Zoom-Raum veröffentlicht war, finden Sie hier den korrekten Link:

https://uni-augsburg.zoom.us/j/97733319417?pwd=RnZjOVhpZktINlVKTlVPcm1uTEEyUT09

Der offene Zoom-Raum findet immer freitags von 14:00 bis 17:00 Uhr statt. Dort sind ein Mitarbeiter und einige Tutoren anwesend, denen Sie Fragen zum aktuellen Stoff und zum aktuellen Übungsblatt stellen können. Außerdem können Sie dort in Breakout-Sessions mit Ihrem Team oder anderen Studierenden gemeinsam am Übungsblatt arbeiten.

- Alle erstellten Programme sollen grundsätzlich mit den Compilerschaltern `-ansi -pedantic -Wall -Wextra` ohne Fehler und Warnungen kompilierbar und ausführbar sein.
- Soweit nicht explizit verlangt, dürfen ungültige Parameterwerte in Funktionen ignoriert werden. Anders ausgedrückt: Gibt es keine Vorgabe für das Verhalten einer Funktion bei Übergabe eines ungültigen Werts, bleibt das Verhalten der Funktion in diesem Fall undefiniert. In der Regel wird solches undefiniertes Verhalten einer Funktion in der Dokumentation der Funktion beschrieben.

[1] https://studiswebstud.zv.uni-augsburg.de/FN2AUTH/FN2AuthServlet?op=Login&nologout=true

Aufgabe 29 *(Statische Variablen & Eingabe, 18 Minuten)*

Erstellen Sie kompilierbare und ausführbare C-Programme mit je einer `main`-Funktion nach unten folgenden Vorgaben. Kompilieren Sie Ihre Programme mit den Compilerschaltern `-ansi -pedantic -Wall -Wextra` und führen Sie sie aus.

a) *(*, 5 Minuten)*
Schreiben Sie ein Programm, das als einfacher Taschenrechner dient: Es sollen zwei vom Benutzer eingegebene ganze Zahlen getrennt durch ein `'+'`-Zeichen **ohne** Berücksichtigung von Puffer-Fehlern eingelesen, addiert und die durchgeführte Rechnung samt Ergebnis ausgegeben werden. Bei ungültigen Eingaben soll, wenn nötig, der Puffer geleert werden.

b) *(***, 10 Minuten)*
Erweitern Sie ihr Programm aus Teilaufgabe a) um zwei Funktionen:

- Der Taschenrechner soll solange verfügbar sein, bis das Programm beendet wird.
- Das letzte Ergebnis soll zwischengespeichert und in einer Rechnung verwendbar sein: Führen Sie dazu eine statische Variable ein, in der Sie nach jeder Berechnung das Zwischenergebnis speichern. Passen Sie außerdem das Einlesen so an, dass die Eingabe zusätzlich ein Schlüsselworts für das vorherige Ergebnis (z.B. `"ans"`) erlaubt. Bei ungültigen Eingaben soll, wenn nötig, der Puffer geleert werden.
  Hinweis: Beschränken Sie sich auf den Fall, dass das Schlüsselwort nur als erster Summand verwendet werden darf, um die Anzahl der zu betrachtenden Fälle zu reduzieren.

c) *(**, 3 Minuten)*
Welche Anpassungen müssten Sie vornehmen, damit Ihr Programm zusätzlich zur Addition auch Subtraktion, Multiplikation und Division erlaubt? Beschreiben Sie die notwendigen Anpassungen in zwei oder drei Sätzen. Teilen Sie Ihre Antwort auf in Code-Bereiche, die angepasst werden müssen, und in Code, der neu hinzugefügt werden muss.

Aufgabe 30 ** *(Übersetzungseinheiten, 20 Minuten)*

a) (**, 8 Minuten)
Erstellen Sie eine allgemein einsetzbare Übersetzungseinheit, bestehend aus einer Header- und einer C-Datei, zur Berechnung von Binomialkoeffizienten. Die Übersetzungseinheit soll folgende Funktionen enthalten:

- Eine Funktion, um die Fakultät einer ganzen Zahl n zu bestimmen: $n! = n \cdot (n-1) \cdot \ldots 1$ für $n \geq 1$
- Eine Funktion, um den Binomialkoeffizient für zwei ganze Zahlen n und k mit folgender Formel zu berechnen: $\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!}$ mit $n \geq 1$, $n \geq k \geq 0$

Verwenden Sie für die Ein- und Rückgabewerte der Funktionen den Datentyp `unsigned long`.

b) (**, 12 Minuten) Testen Sie ihre Übersetzungseinheit aus Teilaufgabe a) in einem eigenen Hauptprogramm und führen Sie folgende Schritte aus:

- Bestimmen Sie den maximalen Wert n, für den die Fakultät (und damit jeder Binomialkoeffizient mit n) noch korrekt berechnet werden kann.
  Hinweis: Wenn $n!$ noch korrekt berechnet werden kann, kann $(n+1)!$ dann nicht korrekt berechnet werden, wenn $(n+1) \cdot n!$ größer als der maximale Wert ist, der von Variablen vom Typ `unsigned long` angenommen werden kann. Lösen Sie die resultierende Ungleichung nach $(n+1)$ auf.

- Geben Sie die ersten n Zeilen des Pascal'schen Dreiecks ordentlich formatiert aus. *Hinweis: Eine linksbündige Ausgabe ist ausreichend, z.B. so für $n = 3$:*

  ```
  1 1
  1 2 1
  1 3 3 1
  ```

- Bestimmen Sie für jede Zeile des Pascal'schen Dreiecks die Summe der enthaltenen Werte und geben Sie diese ebenfalls aus. Was fällt ihnen auf?

Aufgabe 31 * *(Übersetzungseinheiten, 22 Minuten)*

a) (*, 14 Minuten)
Erstellen Sie eine allgemein einsetzbare Übersetzungseinheit, bestehend aus einer Header- und einer C-Datei, zur Verwaltung von Längenangaben von Strecken in Kilometern oder Seemeilen[2] in `double`-Feldern. Die Übersetzungseinheit soll folgende Funktionen und Makros enthalten:

- Jeweils ein Makro zur Umrechnung einer Längenangabe von Seemeilen in Kilometer bzw. Kilometern in Seemeilen
- Eine Funktion zur Berechnung der Gesamtlänge einer Strecke, deren Teilstrecken in einem `double`-Feld gespeichert sind
- Eine Funktion zur Berechnung der durchschnittlichen Länge der Teilstrecken in einem `double`-Feld
- Eine Funktion zur sicheren Eingabe der Länge einer weiteren Teilstrecke durch den Benutzer (`scanf`) in ein bestehendes `double`-Feld **ohne** Berücksichtigung von Pufferfehlern. Machen Sie anhand des Rückgabewerts der Funktion erkennbar, ob das Einlesen funktioniert hat oder nicht. Leeren Sie bei ungültigen Eingaben bei Bedarf den Eingabepuffer.

Legen Sie dabei selbst den Namen von Header- und C-Datei, den Namen der Makros, sowie die Prototypen der Funktionen geeignet fest. Verwenden Sie bei Bedarf ergänzend geeignete symbolische Konstanten.

b) (*, 8 Minuten)
Testen Sie alle Details Ihrer Implementierung durch Eingabe, Verarbeitung und Ausgabe einiger Teilstrecken.

Aufgabe 32 ** *(Übersetzungseinheiten & statische Variablen, 24 Minuten)*

a) (**, 14 Minuten)
Erstellen Sie eine allgemein einsetzbare Übersetzungseinheit, bestehend aus einer Header- und einer C-Datei, zum Umgang mit Zahlenfolgen. Die Übersetzungseinheit soll folgende Funktionen und Makros enthalten:

- ein Makro, um ohne Fehlerbehandlung eine ganze Zahl vom Benutzer einzulesen (`scanf`)
- eine Funktion `unsigned int fibonacci(void)`, die beim n-ten Aufruf die n-te **Fibonacci-Zahl** f_n zurückgibt.

  Die Fibonacci-Zahlen sind dabei wie folgt induktiv definiert:
  - $f_1 := 0$
  - $f_2 := 1$
  - $f_{n+1} := f_n + f_{n-1}$ für $n \geq 2$

  Hinweis: Verwenden Sie zwei statische Variablen.

[2] Eine Seemeile entspricht 1852.0 Metern.

- Eine Funktion `unsigned int brady_seq(void)`, die genauso wie die Finbonacci-Funktion funktioniert, jedoch für beliebige Startwerte b_1 und b_2, sowie $b_{n+1} = b_n + b_{n-1}$ für $n \geq 2$
- Eine Funktion `void init_brady(unsigned int b1, unsigned int b2)`, die die statischen Variablen von `brady_seq()` auf die übergebenen Werte setzt.

- eine Funktion `void calc_sequence_ratio(unsigned int seq[], double ratio[], int len)`, die das Verhältnis der aufeinanderfolgenden Zahlen im übergebenen `unsigned int`-Feld bestimmt und im übergebenen `double`-Feld speichert.
  Beispiel: Für das unsigned int Feld {`1, 2, 4, 8, 16`} *enthält das double-Feld die Werte:*

  ```
  {2.0, 2.0, 2.0, 2.0}
  ```

Legen Sie dabei selbst den Namen von Header- und C-Datei, den Namen der Makros, sowie die Prototypen der Funktionen geeignet fest. Verwenden Sie bei Bedarf ergänzend geeignete symbolische Konstanten.

b) (**, 10 Minuten)
Implementieren Sie ein Hauptprogramm, um Ihre Übersetzungseinheit aus der vorherigen Teilaufgabe zu testen:

- Erstellen sie drei `unsigned int`-Felder sowie drei `double`-Felder fester Länge.
- Verwenden Sie die Werte `2308U` und `4261U` als Startwerte für die erste Brady-Folge.
- Speichern Sie in den ersten `unsigned int`-Feldern die ersten Werte der Fibonacci- bzw. ersten Brady-Folge.
- Lesen Sie zwei ganze Zahlen vom Benutzer ein und verwenden Sie diese als Startwerte der zweiten Brady-Folge
- Speichern Sie das Verhältnis der Folgen in den drei `double`-Feldern und geben Sie die Verhältnisse aus. Was fällt Ihnen dabei auf?