Nachname: ..... Vorname: ..... MatrNr.: .....

Klausur Programmieren 2

HAW-Hamburg, Fakultät Technik und Informatik, Department Informations- und Elektrotechnik
Prof. Dr. Robert Heß, 18.9.2020, Bearbeitungsdauer: 180 Min.
Erlaubte Hilfsmittel: Vorlesungsunterlagen, Lösungen aus dem Praktikum und C/C++ Einführungsbücher.

Ergebnis: ..... von 100 Punkten Note: ..... Punkte.

---

1 Einleitung

Als angehende Ingenieure der Elektrotechnik sind Sie häufig mit Messergebnissen konfrontiert, die Ihnen in Form langer Zahlenkolonnen vorliegen. Diese gilt es zu analysieren und Sie müssen Ihre Schlüsse aus den Ergebnissen ziehen.

In dieser Programmieraufgabe gehen wir davon aus, dass eine Messung bereits erfolgte und Ihnen die Messreihen als Binärdateien vorliegen. Ihre Aufgabe ist es, diese Werte einzulesen und auszuwerten. Neben den statistischen Kennzahlen *Mittelwert* μ, *Varianz* σ^2 und *Standardabweichung* σ sollen Sie ein Histogramm bestimmen und in eine CSV-Datei speichern.

1.1 Dateifromat der Messreihen

Unter dem Link http://www.rrhess.de/download/data.zip finden Sie drei Messreihen mit den Dateinamen *data1.dat*, *data2.dat* und *data3.dat*, zusammengefasst als ZIP-Datei. Sie beginnen jeweils mit einem 4-Byte Integer für die Anzahl n der gespeicherten Werte. Danach folgen genau n Messwerte vom Typ *double* mit jeweils 8 Byte:

Bytes	Typ	Inhalt
4	int	Anzahl n der folgenden double-Zahlen
8	double	1. Zahl
8	double	2. Zahl
8	double	3. Zahl
...	...	...
8	double	n-te Zahl

Die Messwerte liegen alle im Wertebereich $[0, 4)$, d.h. die Zahlen haben Werte von null bis knapp unter vier.

1.2 Statistik

Sie sollten für jede der drei Messreihen den Mittelwert μ, die Varianz σ^2 und die Standardabweichung σ bestimmen. Mit x_k, $k = 1 \ldots n$ für die eingelesenen Werte folgt:

$$\mu = \frac{1}{n}\sum_{k=1}^{n} x_k \qquad \sigma^2 = \frac{1}{n-1}\left\{\sum_{k=1}^{n} {x_k}^2 - \frac{1}{n}\left(\sum_{k=1}^{n} x_k\right)^2\right\} \qquad \sigma = \sqrt{\sigma^2}$$

1.3 Histogramm

Ein Histogramm ist eine Darstellung der Häufigkeitsverteilung. D.h. es wird z.B. ermittelt, wie oft die 0.0, wie oft die 0.1, die 0.2 u.s.w. in der Zahlenkolonne vorkommt. Da es bei Gleitkommazahlen zwischen zwei unterschiedlichen Zahlen wieder unendlich viele verschiedene Werte gibt, müssen wir hier mit Intervallen arbeiten.

In dieser Aufgabe soll ein Histogramm mit 100 Elementen erstellt werden. In dem ersten Element werden die Werte $0 \leq x_k < 0{,}04$ gezählt. Im zweiten Element werden die Werte $0{,}04 \leq x_k < 0{,}08$ gezählt etc.

Zur Realisierung legen Sie einen Vektor mit 100 unsigned-Elementen an und initialisieren ihn mit Nullen. Danach gehen Sie die Messwerte der Reihe nach durch, bestimmen für jeden Wert den zugehörigen Index im Bereich $0 \ldots 99$ und erhöhen das entsprechende Element des Histogramms. Angenommen, der Vektor für das Histogramm hat den Namen *hist* und der aktuelle eingelesene Wert den Namen *value*, dann folgt:

```
hist[(int)(100*value/4.0)]++;
```

1.4 CSV-Datei

Eine CSV-Datei (*Comma Separated Values*) ist eine Textdatei, in welche die Werte durch Kommata getrennt zeilenweise geschrieben werden. In dieser Aufgabe soll das Histogramm in zwei Spalten gespeichert werden: In die erste schreiben Sie das jeweils untere Ende des Intervalls, also die Werte 0 - 0.04 - 0.08 - 0.12 u.s.w. In die zweite Spalte schreiben Sie die Anzahl der Messwerte, die in das entsprechende Intervall fallen.

Für das Histogramm der Datei *data1.dat* ergibt sich für die ersten vier Zeilen:

```
0,2021
0.04,1909
0.08,1828
0.12,1773
...
```

CSV-Dateien können z.B. mit *LibreOffice Calc* oder *MS Office Excel* eingelesen und weiter verarbeitet werden. Das ist aber nicht Teil dieser Aufgabe.

2 Programmieraufgaben

Aufgabe 1 (5 Punkte)

Legen Sie im Hauptprogramm main() (neben möglichen weiteren) folgende Variablen an:

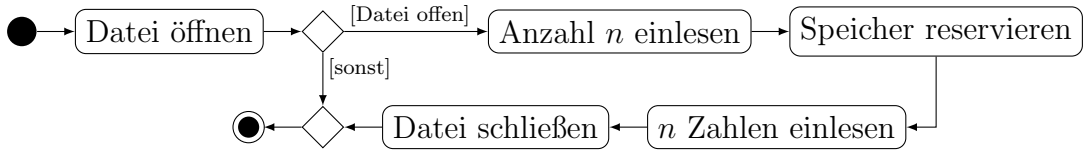
Typ	Name	Beschreibung
double*	data	Zeiger für den dynamischen Vektor der Messwerte
int	size	Anzahl der eingelesenen Messwerte
unsigned[100]	histogram	Histogramm der Messwerte
double	mean	Mittelwert der Messwerte
double	variance	Varianz der Messwerte

Aufgabe 2 (15 Punkte)

Erstellen Sie eine Funktion zum Einlesen einer Binärdatei mit folgender Deklaration:

```
int readData(const char *filename, double **data, int *size);
```

Die zu erstellende Funktion *readData()* erhält im ersten Parameter den Namen der Datei. Der zweite und dritte Parameter wird für die Reservierung des Speichers benötigt. Die Funktion öffnet die Datei, ermittelt die Anzahl der Messwerte, reserviert entsprechend Speicher und ließt die Messwerte ein. Es ergibt sich folgender Ablauf:



Die Funktion gibt bei Erfolg den Wert null, bei Misserfolg einen negativen Wert zurück.

Aufgabe 3 (15 Punkte)

Erstellen Sie eine Funktion, welche aus den eingelesenen Daten Mittelwert und Varianz bestimmt. Verwenden Sie folgende Deklaration:

```
void getStatistics(double *data, int size, double *mean, double *variance);
```

Da die Standardabweichung σ einfach die Quadratwurzel der Varianz σ^2 ist, braucht diese nicht separat zurückgeben werden.

Aufgabe 4 (15 Punkte)

Erstellen Sie eine Funktion, die aus den eingelesenen Messwerten ein Histogramm bildet:

```
void getHistogram(double *data, int size, unsigned *hist);
```

Das Histogramm soll immer 100 Elemente haben, so dass die Größe nicht übergeben werden muss.

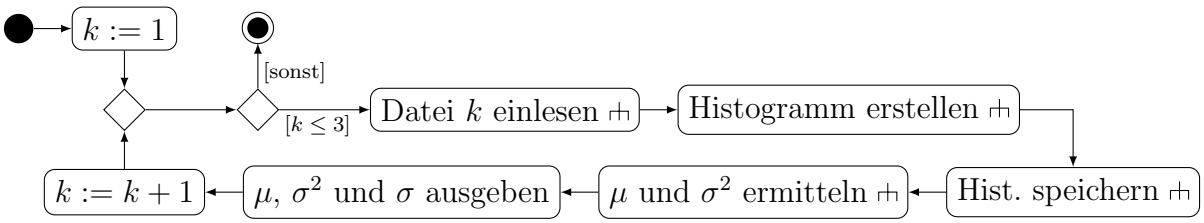
Aufgabe 5 (14 Punkte)

Erstellen Sie eine Funktion, welche das Histogramm in eine CSV-Datei schreibt:

```
int writeHistogram(const char *filename, unsigned *hist);
```

Aufgabe 6 (10 Punkte)

Fügen Sie die erstellten Funktionen zu einem lauffähigen Programm zusammen. Es soll gemäß dem folgenden Aktivitätsdiagramm die drei zur Verfügung gestellen Dateien verarbeiten:



Die drei CSV-Dateien sollen entsprechend die Namen *data1.csv*, *data2.csv* und *data3.csv* erhalten. Verwenden Sie z.B. die Funktion *sprintf()*, um in einer Schleife die Dateinamen automatisch zu generieren.

3 Verständnisfragen

Gehen Sie in diesem Abschnitt von einem 32-Bit System aus.

Aufgabe 7 (8 Punkte)

Definieren Sie

a) einen Zeiger auf *int*

b) einen Doppelzeiger auf *double*

c) einen Zeigervektor mit acht Elementen auf *short*

d) einen Vektorzeiger auf einen *char*-Vektor mit 12 Elementen

e) Bestimmen Sie den Speicherplatz in Bytes für:

a) ..... b) ..... c) ..... d) .....

Aufgabe 8 (4 Punkte)

Finden Sie jeweils eine kompaktere bzw. übersichtlichere Schreibweise:

a) (∗pData).a = 5;

b) x = ∗(pValue+2);

Aufgabe 9 (4 Punkte)

Welche Befehle werden zum Lesen und Schreiben von Binär-Dateien verwendet?

Aufgabe 10 (3 Punkte)

Warum belegt eine *union*-Struktur im Speicher weniger Platz als eine *struct*-Struktur mit den gleichen Elementen?

Aufgabe 11 (4 Punkte)

Erläutern Sie die Unterschiede zwischen den Befehlen *malloc()* und *calloc()*:

Aufgabe 12 (3 Punkte)

Warum muss bei rekursiver Programmierung die Rekursionstiefe begrenzt werden?