14.12.2018

Ausgleichsgerade

Projekt ITA

Erstellt: Bielefeld

Jens Höcker, Tim Kirchner, Tobias Fissenebert

Persönliche Erklärung

Hiermit bestätige ich, dass ich die vorliegende Dokumentation in Gruppenarbeit verfasst und keine anderen als die angebenden Hilfsmittel benutzt wurden. Die Stellen der Arbeit, die den Wortlaut oder dem Sinn nach anderen Werken (Internetquellen) entnommen sind, wurden unter Angabe der Quellen kenntlich gemacht.

Ort, Datum, Unterschrift (Jens Höcker)

Hiermit bestätige ich, dass ich die vorliegende Dokumentation in Gruppenarbeit verfasst und keine anderen als die angebenden Hilfsmittel benutzt wurden. Die Stellen der Arbeit, die den Wortlaut oder dem Sinn nach anderen Werken (Internetquellen) entnommen sind, wurden unter Angabe der Quellen kenntlich gemacht.

Ort, Datum, Unterschrift (Tim Kirchner)

Hiermit bestätige ich, dass ich die vorliegende Dokumentation in Gruppenarbeit verfasst und keine anderen als die angebenden Hilfsmittel benutzt wurden. Die Stellen der Arbeit, die den Wortlaut oder dem Sinn nach anderen Werken (Internetquellen) entnommen sind, wurden unter Angabe der Quellen kenntlich gemacht.

Ort, Datum, Unterschrift (Tobias Fissenebert)

I Inhaltsverzeichnis

II Abbildungsverzeichnis

0 Vorschlagsphase

Die Ergebnisse der Vorschlagsphase orientieren sich an der im Unterricht ausgeteilten Aufgabenstellung.

Es soll ein C-Programm zur Berechnung einer Ausgleichsgeraden für eine nicht näher genannte Anzahl Messwerte erstellt werden.

Das Ergebnis besteht aus der folgenden Dokumentation, dem eigentlichen Programm und einer Bedienungsanleitung. Die Abgabe erfolgt in Form eines mobilen Datenträgers und in ausgedruckter Form.

1 Planungsphase

1.1 Arbeits- und Zeitplan

Die Arbeitsplanung orientiert sich, ebenso wie diese Dokumentation, an der Projektplanungsvorlage gemäß O31F.

Der geplante Zeitumfang liegt bei acht Zeiteinheiten zu je 90 Minuten. Die Dokumentation als Gesamtergebnis wird über die gesamte Dauer des Projektes erstellt.

Die Realisierung des eigentlichen Produktes erfolgt ab der dritten Arbeitsphase in Anlehnung an den Ablauf einer Projektrealisierung. Die Erfüllung der Kann-Kriterien ist als zusätzliche Arbeitsleistung geplant.

Der Beginn des Projektes erfolgte am 6. Okt. 2018. Die Auslieferung des Produktes ist auf den 15. Dez. 2018 terminiert.

Arbeitsschritt	Projektschritt	KW40	KW44	KW48	KW49	KW50	KW51
	Start	X					
0	Vorschlagsphase	X					
1	Planungsphase		X				
2	Konzeptphase		X	X			
3	Entwurfsphase			X	X		
4	Implementationsphase				X		
5	Einführungsphase				X		
6	Betrieb des Systems				X		
	Dokumentation	X	X	X	X	X	X
	Abgabe						X

Zeitplan

1.2 Ist-Beschreibung

Am Tag des Projektbeginns ist kein Programm vorhanden, es muss vollständig anhand der ausgeteilten Unterrichtsmaterialien, sowie der Online-Materialbibliothek erarbeitet werden.

Die Kalkulation einer Ausgleichsgeraden, basierend auf einer Messwertetabelle, von Hand stellt die Ausgangssituation dar. Besonders die Berechnung der Steigung und des Y-Achsenabschnitts sowie der dafür benötigten Werte erweist sich als aufwändig. Das Ziel des Produktes ist es diesen Vorgang zu vereinfachen. Der Formalismus ist aus C74A (Anhang) entnommen.

1.3 Pflichtenheft

Das Pflichtenheft ist an die Abschnitte eins bis drei (Anhang: C74A) angelehnt. Im Folgenden die Zielfestschreibung als Muss-Kriterien:

- Die Implementation des Produktes (Software) ist in ANSI-C zu realisieren
- Mehrteilige Programmfunktion:
    - Menü: Auswahl der Unterfunktionalitäten des Produktes
    - Eingabe: Eingabe neuer Messreihen
    - Im-/Exportfunktion: Eingegebene Datensätze sollen in Dateien exportiert und später wieder importiert werden können
    - Kalkulation: Berechnung der Ausgleichsgeraden
    - Ausgabe: Anzeige der kalkulierten Werte zu der Ausgleichsgarde (gem. allgemeiner Geradengleichung)
    - Manipulation: Es soll eine nachträgliche Manipulation der Messdaten möglich sein
- Die Bedienung des Programms soll den Anforderungen gemäß C74A 3 genügen
    - Konsolenanwendung
    - Navigation durch Zifferneingabe
    - Die Dateneingabe und Datenausgabe soll in Tabellarischer Form erfolgen
- Das Phasenmodell soll als Planungsgrundlage verwendet werden
- Realisierung des Projektes unter Anwendung des Top-Down Modells
- Zur Implementation ist die C-Standard Bibliothek zu verwenden
- I18N und L10N ist nach Möglichkeit anzuwenden (Internationalisierung und Lokalisierung)
- Die Verwendung von globalen Variablen und Sprüngen ist zu vermeiden
- Zu jeder Funktion der Implementation soll eine Testmöglichkeit existieren
- Die Anforderungen aus C74A 4 zur Verwendung der Felder (Arrays) fx und fy als Kalkulationsgrundlage in der Main ()- Funktion soll eingehalten werden

Folgend die Auflistung der Kann-Kriterien:

- Grafische Darstellung der Messpunkte sowie der Ausgleichsgerade
- (Pseudo-)Zufälliges generieren von Messwerten
- Löschen von (Teil-) Datenreihen
- Modularisierung von Programmbestandteilen
- Berechnung von statistischen Kennwerten wie z.B. der Standardabweichung
- Einzelne Produktversionen für Linux und Windows-Systeme

2 Konzeptphase

2.1 Überblick

Die Realisierung des geforderten Programms erfolgt gemäß dem Pflichtenheft, in der Programmiersprache C.

Des Weiteren sollen alle Funktionen als Dummies mit finalen Funktionsnamen vorhanden sein, welche jedoch nur den Funktionsnamen selber ausgeben.

2.2 Grobentwurf des Programms

Das Produkt soll sich an das Pflichtenheft halten. Die Erfüllung der Kann-Kriterien ist angestrebt und von der Realisierbarkeit der vorgegebenen Zeitplanung abhängig. Auf eine Realisierung einer grafischen Darstellung wird dabei, nach Rücksprache, verzichtet.

Bezüglich der Funktions-Dummies wird ein eigenes Verzeichnis mit einer vorrangegangen Programmversion mitgeliefert.

In Absprache, wurden die Felder (Arrays) fx und fy, welche laut Pflichtenheft in der `main()`-Funktion hätten definiert werden müssen, durch eine Datenstruktur des Typ struct messreihe_t wie in der Header-Datei `messreihe.h` definiert, ersetzt.

Eine Modularisierung des Programmes wird durch Aufteilung des Programmes in verschiedene Quelldateien erreicht. Diese sind nach zusammengehörigen Funktionsgruppen gegliedert.

Besonders die Datenverwaltung innerhalb des Programmes stellte eine Herausforderung dar. Hierzu existierten diverse Ansätze. Die im Lastenheft definierte Anforderung zweier statischer Datensätze innerhalb der Main()-Funktion ist sehr unflexibel. Auch die Pufferung in Dateien wurde in Erwägung gezogen, ist jedoch zu unperformant. Als ausreichend flexibel zeigte sich eine dynamische Speicherverwaltung.

Auch hier musste zwischen zwei Alternativen abgewogen werden. Die dynamische Datenverwaltung kann einerseits über eine Tabellarische Verwaltung (Array []) im sog. Heap des Hauptspeichers erfolgen und andererseits als Linked-List (bekannt aus allgemeinen Datenstrukturen der Informatik) die ebenfalls im Heap (oder auch Haldenspeicher) liegt.

Um die Anforderung an eine undefinierte Menge an Datensätzen möglichst effizient zu erfüllen, wurde entschieden, dynamische Speicherverwaltung zu nutzen. Eine Linked-List erschien unproportional zur Komplexität des Problems.

Die Verwendung von Algorithmen wie sie in C1.0 bis C6.10 beschrieben sind ist zulässig und angestrebt.

2.3 Vergleich / Votum

Der Vergleich sowie eine interne Abstimmung hat ergeben, dass in Bezug auf die geforderte Komplexität des Produktes (Software) und dem gegebenen Zeitrahmen eine planungsorientierte Dokumentation der Planungsphase am effizientesten ist. Die Implementation soll im genannten Umfang zum vereinbarten Produkt führen. Als Grundlage für die Dokumentation wird eine Programmversion mit Dummie-Modulen verwendet. Die Umsetzung des Projektes wird in der Entwurfsphase fortgeführt..

3 Entwurf

3.1 Benutzer Interface

Das entworfene Programm wird in Form einer Konsolenanwendung bedient.

Mit Hilfe des Programms wird es ermöglicht folgende Operationen durch zu führen:

- Daten eingeben
- Daten ausgeben
- Daten korrigieren
- Daten speichern
- Daten laden
- Ausgleichsgerade ermitteln

Der Aufbau und die Funktionsweise des Interfaces wird dabei durch die Aufgabenstellung vorgegeben und kann selbigen unter Abschnitt 3 entnommen werden.

Aufgrund der in Abschnitt 2.2 aufgeführten und in Abschnitt 2.3 beschriebenen, freiwilligen Erweiterung des Programms, unterscheidet sich das finale Interface von der Vorlage, ist aber dennoch stark an selbiger angelehnt.

3.2 Datenbeschreibung

Die Datenstrukturen sind gemäß MVC-Prinzip getrennt von der Implementation der Programmfunktionen zu betrachten.

Mit Hilfe der Structs `messwert_t` und `messreihe_t` werden sowohl neu eingegebene als auch eingelesene Messwerte verwaltet. Dabei dient das Struct `messwert_t` für die Verarbeitung der X- und Y- Koordinaten, während das Struct `messreihe_t` die Anzahl an Messwertpaaren in einem Array organisiert und Informationen über die Anzahl an Werten und die Gesamtgröße enthält.

Die Realisierung hat gezeigt, dass die Zusatzinformationen (Anzahl der Messwerte und die Gesamtgröße) regelmäßig benötigt werden. Somit kann über das dirkete mitführen der Zusatzinformationen ein deutlich optimiertes Datenhandling erfolgen.

Eingegebene Datensätze werden außerhalb des Programms in einer `messreihe.ttj`-Datei abgespeichert und können später wieder eingelesen und geändert werden. Der Dateiinhalt selber besteht aus 2 Spalten (X-und Y-Wert), welche per Komma getrennt sind, und n Zeilen wobei n der Anzahl an Wertpaaren entspricht.

Die erste Zeile der Definitionsdatei stellt eine optionale Information für das Produkt dar. Diese gibt in Klartext die Menge des Nutzinhaltes an. Dieser wird wiederum für ein laufzeitoptimiertes

einlesen der Datei genutzt. Ohne diese Zeile wird die Datei sequentiell gelesen und der Messdatenspeicher dynamisch erweitert bis das Dateiende erreicht ist.

Die Notwendigkeit für die Datenstruktur im Programm liegt in der angestrebten, beliebigen Größe der Messreihe und der daraus resultierenden Verwaltung wie unter 3.3 beschrieben.

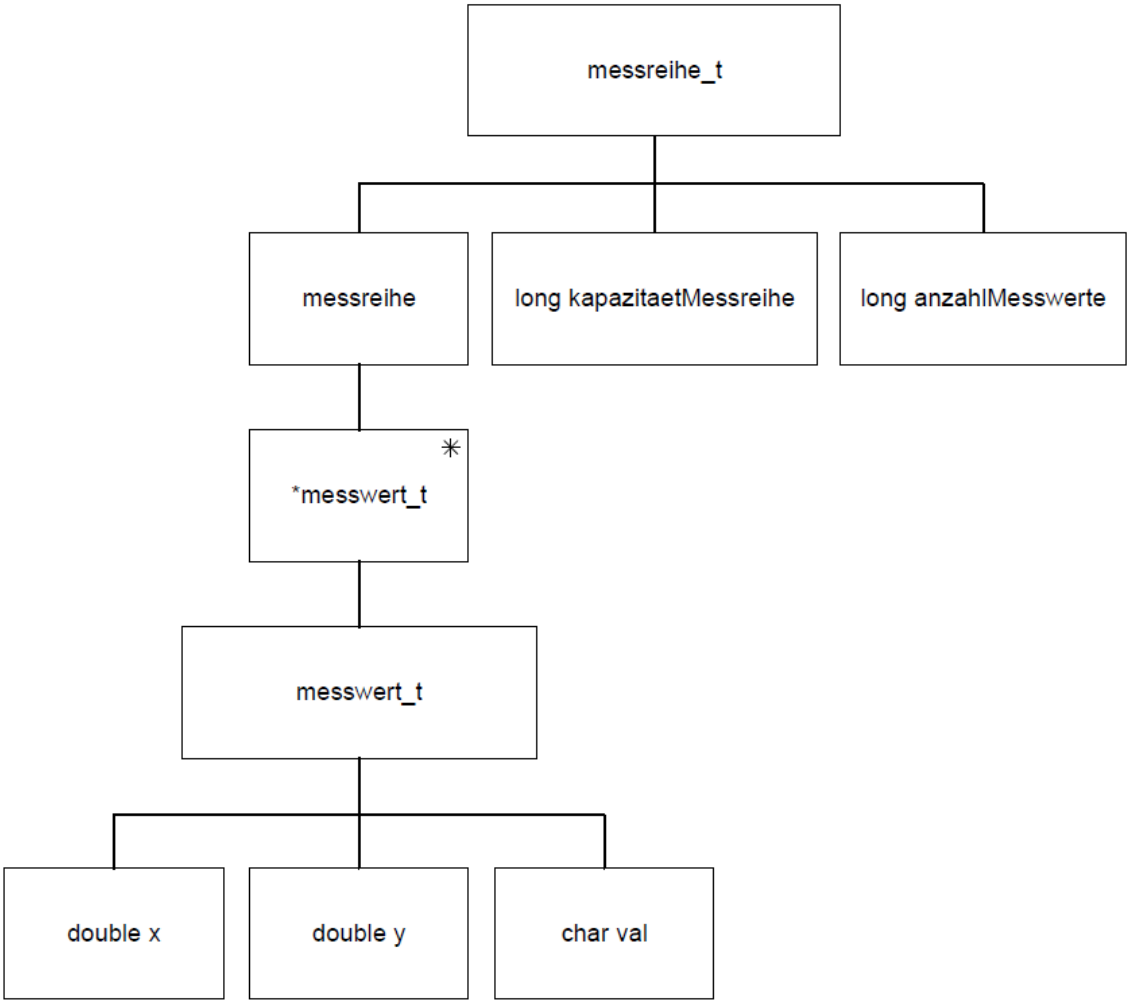

Abbildung 1: Jackson-Diagramm Messreihe

3.3 Systemaufbau

Das Gesamtprogramm wird gemäß dem Pflichtenheft aufgebaut. Die innere Struktur ist unter 3.4 weitergehend beschrieben und im Quelltext zusätzlich kommentiert.

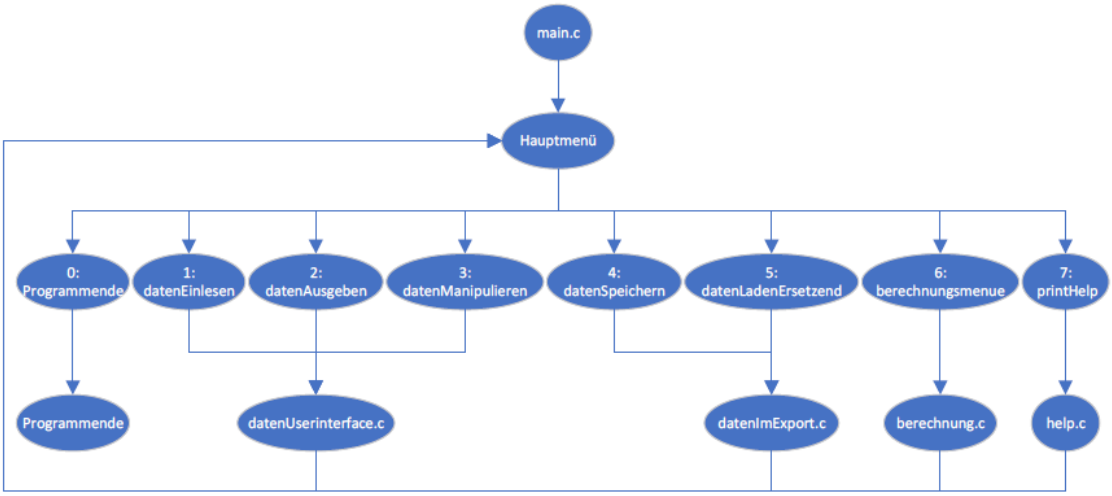

Abbildung 2:Systemaubau

Jede Programmdatei verfügt über eine eigene Header-Datei zur quelloffenen Schnittstellendefinition. Die zusätzliche Header-Datei `ioHilfen.h` dient zur Erkennung des Betriebssystems und definiert den Umgang mit Umlauten unter Unix- und Windows-Systemen. Des Weiteren wird der für Umlaute zu verwendende Zeichensatz definiert. Je Betriebssystem ist definiert, ob UTF8 codierte Zeichen verwendet werden können oder ein Zugriff über HEX-Werte erfolgen soll.

Da die Anzahl der zu bearbeitenden Messreihe nicht definiert ist, werden selbige im Heap verwaltet. Normalerweise (im Stack – statischer Speicherbereich) müsste die Größe des verwendeten Arrays fest deklariert werden um den nötigen Speicherplatz zu reservieren. Mit Hilfe der Funktion `realloc()` aus der Bibliothek `stdlib.c` können wir den Speicher im Heap dynamisch reservieren, frei geben und die Daten in andere Bereiche des Speichers kopieren, basierend auf der Anzahl eingegebener Messwerte. Diese Form der Speicherverwaltung erlaubt eine effektivere Nutzung wie bereits im Programmentwurf beschrieben.

3.4 Definition der Module

Die Module des Programms werden artverwandte Funktionen definiert und wurden aufgabenspezifisch erstellt und selbsterklärend benannt. Das Produkt ist in mehreren Source-Dateien erstellt. Zum Kompilieren sind nur Bibliotheken notwendig die zum Standardumfang der gängigen C-Compiler gehören. Funktionalitäten von Drittanbietern werden nicht verwendet.

Aufgrund der vorhandenen Dokumentation in den Header-/Quelldateien selbst, wird an dieser Stelle lediglich die Funktion der Module beschrieben.

3.4.1 Main-Modul

In der main.c sind die Funktionen `main()` und `menue()` definiert.

Dem Benutzer wird ein Menü angezeigt, mit der Möglichkeit durch Eingabe einer Zahl auf eine entsprechende Funktion des Programms zuzugreifen.

```
(1) Daten aufnehmen
(2) Daten ausgeben
(3) Daten korrigieren
(4) Daten speichern
(5) Daten laden
(6) Ausgleichsgerade ermitteln
(7) Hilfe anzeigen
(0) Ende

(9) Debug-Modus
```

Im Fall einer fehlerhaften bzw. falschen Eingabe wird eine Fehlermeldung ausgegeben und der Benutzer hat die Möglichkeit eine weitere Eingabe zu tätigen.

Dabei übernimmt die Funktion `menue()` die Anzeige der Menü-Punkte sowie das Einlesen und die Fehlerprüfung der Benutzereingabe. Bei einer korrekten Eingabe, wird diese an die `main()`-Funktion übergeben.

In der `main()`-Funktion wird, basierend auf der Benutzereingabe, entweder das Programm beendet (Eingabe =0) oder die entsprechenden Informationen ausgegeben. Dies Erfolg über eine `Switch/Case` Funktion und greift auf Funktionalitäten zu, welche von anderen Modulen bearbeitet und bereitgestellt werden.

Als Entwicklungsstand (vgl. 1.3 Testmöglichkeit) wird hier eine Programmversion mit zur Verfügung gestellt, in der lediglich der angewählte Menüpunkt angezeigt wird bzw. eine Funktionalität eines Externen Moduls aufgerufen wird, die lediglich ihren Namen anzeigt.

3.4.2 berechnung-Modul

berechnung.c übernimmt die Verarbeitung der Messreihe und führt alle nötigen Berechnungen aus. Dabei dient die vorgegebene Formel aus C74A als Grundlage.

$$b = \frac{\sum y * x^2 * - \sum x \sum xy}{n * \sum x^2 - (\sum x)^2}$$

$$m = \frac{n * \sum yx - \sum x \sum y}{n * \sum x^2 - (\sum x)^2}$$

Folgende Werte werden in diesem Modul berechnet:

- Y-Achsenabschnitts
- Steigung
- Anzahl der Messreihen
- Nenner für die Formel, vorgegeben durch die Aufgabenstellung (C74A)
- Summe der X-Werte
- Summer der Y-Werte
- X^2
- Produkt aus X und Y-Wert

Realisiert wurde die Berechnung als Zusammenspiel einiger Funktionen um eine hohe Modularisierung und Wiederverwendbarkeit zu erzielen. Beispielsweise wird für die Summenbildung ein Zeiger auf eine Funktion() verwendet um zu definieren wie die Funktion (math.) der Summenformel aussieht.

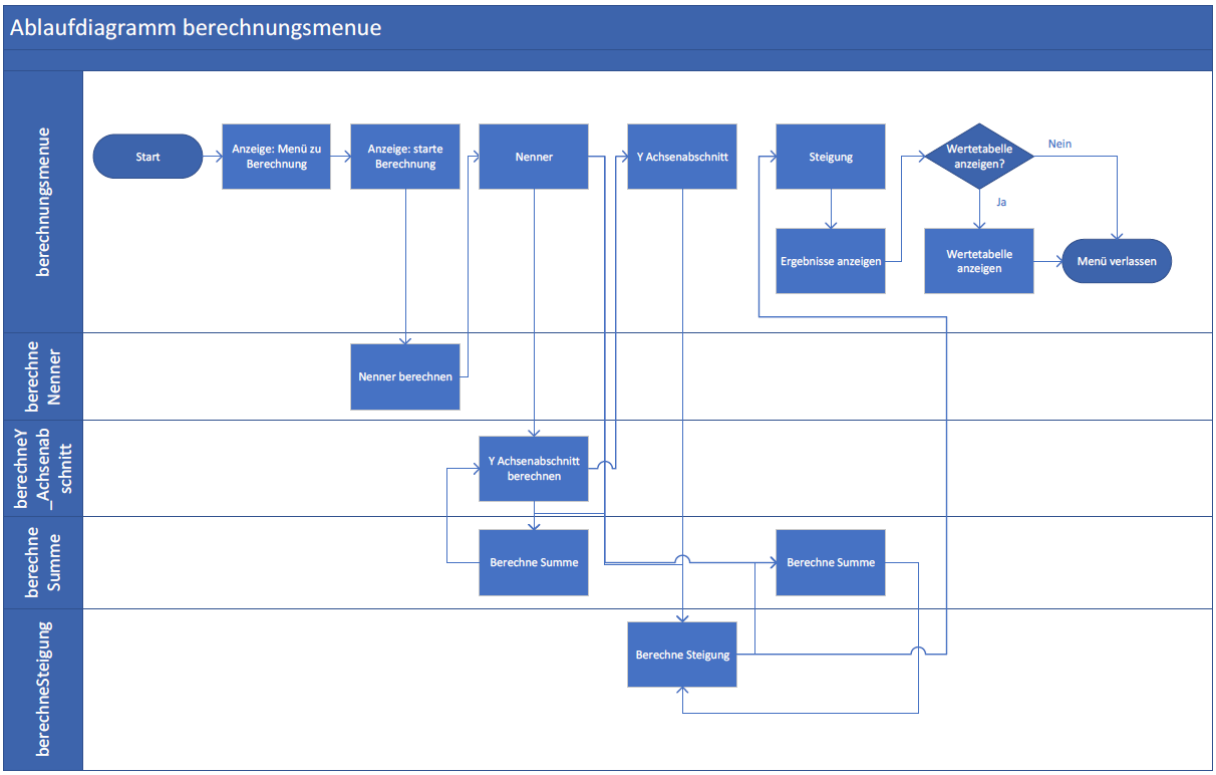

Abbildung 3: Ablaufdiagramm Berechnung

3.4.3 datenHandling-Modul

Wie im Abschnitt 3.3 erläutert, werden die nötigen Daten des Programmes im Heap verwaltet. Die genaue Funktionsweise wird in diesem Modul definiert.

Das Ziel dieses Moduls ist eine Optimierte Auslastung des Hauptspeichers. Dafür besteht dieses Modul im Wesentlichen aus zwei Funktionen().

Als systemnah kann die Funktion `messreiheAllocate(long neueAnzahl, messreihe_t *p_messreihe)` gesehen werden. Diese reserviert die benötigte Bytemenge im dynamischen Bereich des Prozessspeichers über die Funktionen aus der `stdlib.h` (`malloc()`, `realloc()` und `free()`).

An dieser Stelle sei einmal der Aufbau des Hauptspeichers eines Prozesses skizziert. Der Speicherbereich fängt mit dem sogenannten Programmspeicher an. Dieser fasst die Befehle die bearbeitet werden sollen. Vom Ende her wird der sogenannte Stack angesiedelt. Dieser stellt den statischen Bereich des Prozessspeichers dar. Datenfelder die zum Zeitpunkt des Kompilierens bekannt sind liegen in diesem Bereich. Der Stack ist heutzutage wenige Megabyte groß. Der gesamte Zwischenraum (zwischen Programmspeicher und Stack) wird Heap genannt. In diesem Bereich kann dynamisch, also zur Laufzeit eines Prozesses Speicher reserviert werden. Dies wiederum geschieht mit den bereits genannten Funktionen der `stdlib.h`.

Damit keine unerwünschte Auslastung des Hauptspeichers erfolgt soll der Speicherbereich für die Messdaten eben so groß wie nötig aber so schmal wie möglich gehalten werden. Die zweite Funktion dieses Moduls beschäftigt sich mit diesem Problem. Hier wird die Menge der benötigten Speicherfelder gezählt und wenn sinnvoll reagiert. Zum einen geschieht dies auf der Basis von Prozent-Grenzen die fest definiert sind und zum anderen beruhend auf der Speicherreserve, wie diese in der `datenHandling.h` definiert ist.

3.4.4 DatenInExport-Modul

Die Datei für den Im- oder Export kann über eine Standarddatei oder über eine Benutzereingabe definiert werden.

Die Funktion `int datenSpeichern(messreihe_t *p_messreihe)` erstellt und öffnet die externe Datei „`messwerte.ttj`“ in welcher die Messwerte abgespeichert werden. Die Messwerte werden dabei aus dem Array gelesen und in die Datei als double-Werte gespeichert. Wie unter 3.2 beschrieben zuerst die Anzahl der Werteund anschließend die Werte zeilenweise sequentiell.

Das Öffnen und laden der genannten Datei erfolgt über die Funktion `int datenLadenErsetzend(messreihe_t *p_messreihe)`, welche die X- und Y-Werte direkt in das vorgesehene Array lädt.

Dabei wird zuerst geprüft ob eine Anzahl an zu erwartenden Messwerten zur Verfügung steht. Auf diese Größe wird, wenn verfügbar, der Messdatenspeicher angepasst. Anschließend wird im vereinbarten Format soweit Messpunt für Messpunkt eingelesen bis das `eof` erreicht ist.

3.4.5 datenUserInterface-Modul

Dieses Modul dafür zuständig die Informationen aus der Speicherdatei anzuzeigen und dem Benutzer die Bearbeitung selbiger zu erlauben, entweder durch Veränderung existierender oder Anlegen neuer Messwerte die an den bestehenden Datensatz angehängt werden.

Die Funktion `int datenAusgeben(messreihe_t *p_messreihe)` übernimmt die Anzeige der bisherigen Messwerte. Der Benutzer kann dabei die Anzahl der auszugebenden Messwerte, bis eine Benutzeraktion erwartet wird, selber angeben. Sollte eine Eingabe 0 oder unzulässig sein, so erfolgt stattdessen die Ausgabe aller Messwerte auf einmal.

`int datenEinlesen(messreihe_t *p_messreihe)` erlaubt es dem Benutzer, neue Messwerte einzugeben. Die Eingabe erfolgt dabei fortlaufend und wird mit Hilfe der erneuten Eingabe des ersten, eingegebenen Wertes beendet. Dieser wird zur Vereinfachung immer mit angezeigt.

Wie in 1.3 beschrieben wird der lokale Zeichensatz verwendet, so ist also ein ‘,‘ bei Dezimalzahlen zu verwenden. Alle Werte werden im `double` Format abgelegt.

Fehlerhafte Eingabe werden in Abhängigkeit behandelt. Kann aus einer Eingabe eine Zahl abgeleitet werden, so wird diese auf diesen Anteil gekürzt (z.B. 6,3z wird zu 6,3). Eine Eingabe aus der keine Zahl gebildet werden kann (z.B. z6) wird als fehlerhaft gemeldet und es wird eine erneute Eingabe für diesen Wert erwartet.

Mit Hilfe der Funktion `int datenManipulieren(messreihe_t *p_messreihe)` wird ermöglicht, bereits eingegebene Messwerte zu verändern. Folgende Optionen können über das Nummernmenü ausgewählt werden:

- Daten löschen
- Datensatz anzeigen
- Datensatz bearbeiten
- Datensatz löschen

Mit der Option „Daten löschen“ werden alle Daten, nicht nur ein Datensatz, gelöscht. Die Abfrage ob die Daten wirklich gelöscht werden sollen, ist mit y oder n zu beantworten. Jede Antwort ungleich „y“ lässt die Daten weiter bestehen und die Funktion des Daten Löschens ist beendet.

Alle weiteren Optionen erfordern die Eingabe von Datensatznummern, um die genaue Position der gesuchten Datensätze zu ermitteln.

Sollte ein Datensatz außerhalb des Wertebereichs angewählt werden, so wird eine entsprechende Fehlermeldung ausgegeben. Mit der Eingabe einer „-1“ wird der Vorgang beendet. Auch Fehlerhafte Eingaben der Wertnummer oder des Menüpunktes führen zur Beendigung der Subroutine und das Hauptprogramm wird fortgesetzt. Dieses Verhalten ermöglicht es einfacher aus der fortlaufenden Abfrage einer Datensatznummer zu kommen. Besonders für unerfahrene Benutzer kann die Beendigung mit „-1“ unerwartet sein. Eine abweichende Reaktion (Beispielsweise „abwegige“ Tastatureingabe) führt so auch zum gewünschten Ergebnis.

3.4.6 help-Modul

Um dem Benutzer des Programmes zu unterstützen, enthält das help-Modul Hinweise und Hilfestellungen zum Programm und kann über das Hauptmenü aufgerufen werden.

4 Einführungsphase

4.1 Übergabeprotokoll

Die Implementation erfolgt anhand dieser Dokumentation in ANSI-C. Die Lauffähigkeit ist auf UNIX (Ubuntu) und Win64- Systemen getestet. Für die jeweiligen Architekturen ist das Programm in der jeweiligen Umgebung zu kompilieren. Die Präcompiler-Direktiven (zur Bestimmung des Betriebssystems) beziehen sich auf Umgebungsvariablen die durch `gcc` definiert sind.

Als Entwicklungsumgebung werden verschiedene Software-Pakete benutzt. Diese sind nicht Bestandteil dieses Produktes, jedoch frei verfügbar. Auf Windows-Betriebssystemen wird VisualStudio 2017 -CE und VisualStudio Code verwendet. Unter Linux (Ubuntu/ debian) wird gedit verwendet.

Als Compiler dient jeweils gcc. Unter Windows wird als Paket MinGW zur Bereitstellung verwendet. (Siehe Programmverzeichnis)

4.2 Weitere Software

Zur (grafischen) Dokumentation werden weitere Softwarepakete verwendet. Auch hier wurde auf frei verfügbare Software (bzw. frei für Bildungszwecke) Wert gelegt. (Siehe Programmverzeichnis)

4.3 Hauptprogramme:

- Vollversion

4.4 Testprogramm:

- Vorgängerversion

4.5 Nachweis der Funktionalität

Die Funktionalität des Gesamtproduktes sowie der einzelnen Bestandteile sind sorgfältig und ausführlich getestet. Diese Tests kontrollieren die Richtigkeit der Ausgaben sowie das Verhalten bei Fehleingaben. Desweieteren wurde durch Freiwillige Beta-Tester der Schutz sowohl gegen Fehleingaben als auch gegen mutwillige Fehlbedienung getestet.

Disclaimer

Die Software sowie ihre Bestandteile sind ausführlich getestet. Jedoch kann es zu unerwarteten Fehler kommen die einen Programmabsturz erzeugen. Es gilt keine Verbindlichkeit auf die Richtigkeit der Ergebnisse und auch die Interpretation der Ausgaben ist dem Benutzer vorbehalten.

5 Einführung

Die Einführung ist mit der Aushändigung des Datenträgers dieser Dokumentation abgeschlossen.

5.1 Lieferumfang

1x Bedienungsanleitung

1x Dokumentation

1x Ausgleichsgeradenberechnung als C-Programm (Vollversion)

1x Programm mit Funktions-Dummies (Vorgängerversion)

Diverse Dokumentationen

Source-Dateien der Dokumentationsbestandteile

Literaturverzeichnis

Betriebsunterlagen/ Quellverzeichnis:

https:\\ tutorialspoint.com/c_standard_library/

http://www.c-howto.de/

https://stackoverflow.com

Ausbildungsunterlagen:

Die Ausbildungsunterlagen sind unter folgenden Link zu finden:

http://gyrator.de/material/

IV Softwareverzeichnis

IDEs

- (Gedit)
- Microsoft Visual Studio Code (Version 1.28.2)
- Microsoft Visual Studio 2017 CE-Edition (Version 15.7.2)

Dokumentation

- Microsoft Word 2016 (365)
- Microsoft Visio 2018
- egypt (Version 1.10) [Anwendung zur Erstellung von Ablaufdiagrammen] (http://www.gson.org/egypt/)
    - zusätzlich die benötigten Abhängigkeiten Perl und Graphviz
- Okular (als Teil einer Ubuntu Dist.) [Anwendung für .ps zu .pdf]
- Github
- Sourcetree (Version 3.0.8)

V Anhang