Requirements für den Programmentwurf C/C++ TI24

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg

1. Inhaltsverzeichnis

2. Einführung

2.1. Gültigkeit

Diese Version ist Version 1.0

2.2. Abgabe der Prüfungsleistung

Jeder Student und Studentin gibt eigenständig seinen Programmentwurf bei Moodle ab, spätestens bis zum vorgesehenen Abgabezeitpunkt der unter Moodle vermerkt ist.
E-Mail Abgaben und Entwürfe können **nicht** berücksichtigt werden.

Bei Fragen bitte an thomas_staudacher@yahoo.de wenden.

> Anmerkung: Studien- und Prüfungsordnung für die Bachelorstudiengänge im Studienbereich Technik der Dualen Hochschule Baden-Württemberg (DHBW) (Studien- und Prüfungsordnung DHBW Technik – StuPrO DHBWTechnik) § 11 Versäumnis, Rücktritt, Täuschung, Ordnungsverstoß / 1

2.3. Aufgabenstellung

Es soll ein Tool entwickelt werden (Name: **morse**) mit dem sowohl Texte in Morsezeichen wie auch einzelne/mehrere Morsezeichen wieder in Text verwandelt werden können.

Nähere Information zur Umsetzung sind in den Requirements der folgenden Kapitel detailliert aufgeführt.

3. Allgemeine Beschreibung

3.1. Funktionale Anforderungen

Funktionale Anforderungen beschreiben, was das Programm ausführen soll. Im Gegensatz zu nichtfunktionalen Anforderungen, sind sie direkt in der Software zu finden, d.h. es wird kein Tooling beschrieben, sondern was im Programmablauf zu beachten ist. Optionale Anforderung sind nicht bewertet, helfen aber die Qualität des Endprodukt zu steigern.

4. Optionen

4.1. Requirements Kommandozeilenoptionen

Um dem Nutzer die Steuerung Ihres Programms zu erleichtern soll, das Programm ein mit getopt/getoptlong erstelltes Optionsparsing enthalten (Es kann Optionen geben, die nur als Schalter wirken (ein/aus), Optionen, die zwingend weitere Argumente benötigen oder Optionen, bei denen die weiteren Argumente optional sind). Können Optionen nicht geparst werden, so handelt es sich hierbei um übergebene Dateien.

4.2. Hilfestellung

Um das Programm benutzbar zu machen, soll eine Hilfestellung analog **cp --help** implementiert werden. Die Gestaltung ist Ihnen überlassen.

{ReqFunc01} Das Programm soll einen Hilfetext besitzen, welcher per Kommandozeilenparameter **-h** und **--help** aufrufbar ist und die Benutzung Ihres Programms aufzeigt
{ReqFunc02} Der Hilfetext soll sämtliche Kommandozeilenparameter Ihres Progamms und deren Benutzung erklären. Zugelassene Sprachen sind deutsch und englisch
{ReqFunc03} Um den Autor identifizieren zu können, soll (im JSON Format) eine Ausgabeblock erstellt werden: Der Namen der abgebenden Person, der Studiengang (TIT,TIK,TIM,TIS sind zulässig) und wenn möglich eine

Mail Adresse zur Kontaktaufnahme. Codepage ist **utf-8**. Der Schalter hierfür ist **--programmer-info**

Beispiel (Vollausgefüllt):

```
morse --programmer-info
{
    "firstname": "Thomas",
    "surname": "Staudacher",
    "branch_of_study": "TIS",
    "contact": "thomas_staudacher@yahoo.de"
}
```

Beispiel (**contact** bleibt ungenutzt):

```
morse --programmer-info
{
    "firstname": "Thomas",
    "surname": "Staudacher",
    "branch_of_study": "TIS",
    "contact": ""
}
```

4.3. Encodieren / Dekodieren

{ReqFunc04} Soll ein Text in Morsecode gewandelt werden, so wird der Schalter **-e, --encode** gewählt

{ReqFunc05} Default kann Encode eingestellt werden (wandeln von Text in Morse Code), um beim weglassen des Schalter **-e, --encode** eine Ausgabe zu erzeugen

{ReqFunc06} Soll ein Text aus Morsecode dekodiert werden, so wird der Schalter **-d, --decode** gewählt

{ReqFunc07} Ist sowohl der Schalter Encodieren **-e, --encode** wie auch Dokodieren **-d, --decode** gewählt, so soll das Programm mit einer Fehlermeldung abgebrochen werden

4.4. Texteingabe und Umleitung

Ohne Angabe eines Schalters werden die verbleibenden Übergaben (Option) als Datei ausgewertet und entsprechend verarbeitet.

{ReqFunc08} Ist die per Option übergebene Datei nicht vorhanden oder kann nicht geöffnet werden, dann soll eine Fehlermeldung ausgegeben werden (Text passend zum Ereignis)

{ReqFunc09} Dateien, die gewandelt werden sollen, können in allen beliebigen Ober-, Unterverzeichnissen, hinter Links oder in absoluten Pfaden abliegen (wenn die Zugriffsrechte stimmen)

{ReqFunc10} Wird per Pipe ein Text an das Programm übergeben, soll aus der Pipe dieser gelesen werden

Beispiel:

```
echo -n "Dieser Text wird per Pipe uebergeben" | morse -e
-.. .. . ... . .-.   - . -..- -   .-- .. .-. -..   .--. . .-.   .--. .. .--. .   ..- . -... .
.-. --. . -... . -.
```

4.5. Textausgabe und Umleitung

Schon in Anfangzeiten von *UNIX* hat sich für die Ausgabeumleitung der Option Schalter **-o, --out** herausgebildet. Aus Kompatibilitätsgründe soll er in diesem Programm mit eingebaut werden.

{ReqFunc11} Über die Schalter **-o, --out** soll der Text in eine Datei umgeleitet werden, in allen beliebigen Ober-, Unterverzeichnissen, hinter Links oder in absoluten Pfaden (wenn die Zugriffsrechte stimmen)

Beispiel:

```
morse quickbrownfox.txt -o quickbrownfox.morse.txt

cat quickbrownfox.morse.txt
- .... .   --.- ..- .. -.-. -.-   -... .-. --- .-- -.   ..-. --- -..-   .--- ..- -- .--. ...
--- ...- . .-.   - .... .   .-.. .- --.. -.--   -.. --- --. -.-.--
```

{ReqFunc12} Über die Schalter **-o, --out** an Dateien ausgegebene Texte können in allen beliebigen Ober-, Unterverzeichnissen, hinter Links oder in absoluten Pfaden abliegen

{ReqFunc13} Über eine Pipe sollen an ein nachgelagertes Programm die Ausgabe übertragen werden

Beispiel:

```
morse <<< Hallo | wc
      1       5      23
```

5. Zeichensatz für Morsecode

Folgende Morsecodetabelle DARC Morsecode wird zu Grunde gelegt. Die in Klammer stehende Satzzeichen werden nicht umgesetzt.

> {ReqFunc14} Die Buchstaben **A-Z** sollen in Morsecode wandelbar sein

Beispiel:

```
morse <<< "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
.- -... -.-. -.. . ..-. --. .... .. .--- -.- .-.. -- -. --- .--. --.- .-. ... - ..- ...- .--
-..- -.-- --..
```

> {ReqFunc15} Die Buchstaben **a-z** sollen in Morsecode wandelbar sein

Beispiel:

```
morse <<< "abcdefghijklmnopqrstuvwxyz"
.- -... -.-. -.. . ..-. --. .... .. .--- -.- .-.. -- -. --- .--. --.- .-. ... - ..- ...- .--
-..- -.-- --..
```

{ReqFunc16} Der Mosecode für die Buchstaben **A-Z,a-z** soll in Großbuchstaben-Klartext rückwandelbar sein

Beispiel:

```
morse -d <<< ".- -... -.-. -.. . ..-. --. .... .. .--- -.- .-.. -- -. --- .--. --.- .-. ... -
..- ...- .-- -..- -.-- --.."
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

{ReqFunc17} Die Zahlen **0-9** sollen in Morsecode wandelbar sein

Beispiel:

```
morse <<< "0123456789"
----- .---- ..--- ...-- ....- ..... -.... --... ---.. ----.
```

{ReqFunc18} Der Mosecode für die Buchstaben **0-9** soll in Klartext rückwandelbar sein

Beispiel:

```
morse -d <<< "----- .---- ..--- ...-- ....- ..... -.... --... ---.. ----."
0123456789
```

{ReqFunc19} Die Satzzeichen **.,:;?** (Punkt, Komma, Doppelpunkt, Strichpunkt, Fragezeichen) sollen in Morsecode wandelbar sein

Beispiel:

```
morse <<< ".,:;?"
.-.-.- --..-- ---... -.-.-. ..--..
```

{ReqFunc20} Der Mosecode für die Satzzeichen **.,:;?** (Punkt, Komma, Doppelpunkt, Strichpunkt, Fragezeichen) soll in Klartext rückwandelbar sein

Beispiel:

```
morse -d <<< ".-.-.- --..-- ---... -.-.-. ..--.."
.,:;?
```

{ReqFunc21} Die mathematischen Symbole **=-+** (Gleich, Minus, Plus) sollen in Morsecode wandelbar sein

Beispiel:

```
morse <<< "=-+"
-...- -....- .-.-.
```

{ReqFunc22} Der Mosecode für die mathematischen Symbole **=-+** (Gleich, Minus, Plus) soll in Klartext rückwandelbar sein

Beispiel:

```
morse -d <<< "-...- -....- .-.-."
=-+
```

{ReqFunc23} Die Formatier Symbole **_()/@** (Unterstrich, Klammer auf/zu, Schrägstrich, Klammeraffe) sollen in Morsecode wandelbar sein

Beispiel:

```
morse <<< "_()/@"
..--.- -.--. -.--.- -..-. .--.-.
```

{ReqFunc24} Der Mosecode für die Formatier Symbole **_()/@** (Unterstrich, Klammer auf/zu, Schrägstrich, Klammeraffe) soll in Klartext rückwandelbar sein

Beispiel:

```
morse -d <<< "..--.- -.--. -.--.- -..-. .--.-."
_()/@
```

6. Formatierung

6.1. Ausgabe

{ReqFunc25} Nicht vorhandene Buchstaben (Zeichen) sind beim Encodieren als ***** (Stern) aus zu geben

Dekodierte Buchstaben (Zeichen) sind in Großbuchstaben aus zu geben. Dazu das Requirement für Encodierung **a-z** beachten.

Beispiel ($-Zeichen ist ein Zeichen lang. €-Zeichen ist utf-8, daher drei Sterne weil drei Zeichen 0xe282ac):

```
echo -n "Kurzer $, langer €" | morse
-.- ..- .-. --.. . .-.   * --..--   .-.. .- -. --. . .-.   * * *
```

6.2. Buchstaben-/Worttrennung

Um Ihr Ergebnis zu kontrollieren, kann die Seite Morse-Code kodieren und dekodieren verwendet werden. Dazu muß aber der Wordspacer analog dieser Internetseite angepasst werden.

{ReqFunc26} Buchstaben werden durch ein **SP** (SPACE) getrennt

{ReqFunc27} Wörter werden durch **SPSPSP** (3*SPACE) getrennt

{ReqFunc28} Newline und Carriage Return werden ignoriert (Whitespaces)

{ReqOptFunc01} Über die Schalter **--slash-wordspacer** soll zwischen Wörter der Space **SP,/,SP** eingefügt werden. Der Buchstabentrenner verändert sich nicht

Beispiel:

```
echo -n "Hallo Welt" | morse --slash-wordspacer
.... .- .-.. .-.. --- / .-- . .-.. -
```

{ReqOptFunc02} Der Schalter **--slash-wordspacer** darf nur in Verbindung mit Encode verfügbar sein. Im Fehlerfall muß eine Warnung ausgegeben und das Programm beendet werden

7. Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen sind umzusetzen. Im Gegensatz zu funktionalen Anforderungen, sind nichtfunktionale Anforderungen nicht direkt an Features des Programms gebunden, sondern beschreiben die Umgebung und das Tooling.

7.1. Tools und Build Umgebungen

7.1.1. Unterstütze Plattformen und Compiler

Um die Software auf den Systemen der DHBW Ravensburg Aussenstelle Friedrichshafen erstellen zu können, ist zwingend erforderlich einen Compiler zu verwenden, der die gleichen Kommandozeilenparameter versteht und frei von Lizenzen ist. Im Zweifelsfall erfolgt die Abnahme der Software zur Benotung in der DHBW Ravensburg Aussenstelle Friedrichshafen im Labor.
Sollten Zusatzlibraries zugelassen sein, so sind diese im Anhang aufgeführt.

{ReqNonFunc01} Die Software soll mit GNU-C/C++ für UNIX und WSL Linux compilierbar sein
{ReqOptFunc03} Die Software soll mit MinGW64 compilierbar sein
{ReqNonFunc02} Die Software soll unter UNIX und WSL Linux ausführbar sein
{ReqOptFunc04} Die Software soll unter Windows ausführbar sein

7.2. Erstellung der ausführbaren Datei

Um die Software frei von graphischen Entwicklungsumgebungen erstellen zu können, wird ein Makefile benötigt. Dieses kann dann unter make (UNIX) erstellt werden. Da der Umfang dieses von Hand zu erstellen den Umfang des Programmentwurfs überschreiten würde, wird hier auf CMake als Erstellungstool für das Makefile verwiesen

{ReqNonFunc03} Das Erstellen der Software soll mit CMake erfolgen
{ReqNonFunc04} Der Name des ausführbaren Programm ist **morse**

Anmerkung: Ein händisch erstelltes Makefile ist nicht zugelassen

7.3. Dokumentation des Programmentwurf

Um den Dozenten Th. Staudacher die Chance zu geben, die Struktur und Funktionalität des Programmentwurf zu überblicken, ist es wichtig, dass die Dokumentation des Codes im doxygen-Format erfolgt. Durch gute Dokumentation werden Rückfragen im Fehlerfall minimiert und es können ggf. auch besser Teilpunkte vergeben werden. Das Dokumentieren kann in deutscher oder englischer Sprache erfolgen.

{ReqOptFunc05} Die Dokumentation des Codes soll mit der Syntax von doxygen erfolgen

{ReqOptFunc06} Die Dokumentation des Codes mit doxygen soll jede Klasse, Methode (Funktion) und Variable abdecken

7.4. Einlesen der Optionen

Um eine einheitliche Basis zu schaffen und das Programm portabel zu halten, wird zum Parsen der Optionen die getopt(_long) Implementierung verwendet. Das implementieren anderer Libraries oder eigengeschriebene Anwendungen ist nicht gestattet.

{ReqNonFunc05} Das Parsen der Optionen soll über die LibC Funktionalität getopt(...) und getopt_long(...) aus getopt.h erfolgen

7.5. Dateien und Verzeichnisstruktur

Um eine saubere Trennung zwischen Implementierung und Deklarierung zu bekommen, ist eine Auftrennung in zwei Verzeichnisse nötig. Unterverzeichnisse in dieser Auftrennung sind ebenfalls möglich.

{ReqOptFunc07} Die Header- und Source-Dateien ihres Programms sollen sich in verschiedenen Unterverzeichnissen befinden
{ReqOptFunc08} Das Inkludieren der Header-Dateien soll nicht im aktuellen Verzeichnis erfolgen, sondern im Inlude-Pfad des Compilers
{ReqOptFunc09} Das Inkludieren von Source-Dateien(*.c / *.cpp) in Source-Dateien ist nicht erlaubt
{ReqOptFunc10} Die erstellten Header Dateien sollen gegen Mehrfacheinbindung geschützt werden und der Schutz soll kompatibel zum Compiler MinGW (Windows) und GNU-C (Unix) sein

8. Anhang

8.1. Zusatzlibraries

Zugelassen als Zusatzlibrary/Programme sind

- gcc/g++ Umfang der Libraries
- astyle zur Formatierung das Codes
- boost in alle Versionen
- cmake als Build System
- doxygen zur Dokumenation das Codes

8.2. Versionen

Version	Änderung
0.8	Initialversion
0.9	Requirements die von Optional abhängig sind wurden auch auf Optional umgeschalten. Satzbau wurde z.T. geändert
1.0	--programmer-info Ausgabe verändert, Rückwandlung Buchstaben A-Z,a-z in Großbuchstaben, nichtwandelbare Zeichen Musterausgabe angepasst

8.3. Übersicht Requirements

8.3.1. Anzahl Requirements

Diese Prüfung enthält 28 Funktionale, 5 Non-Funktionale und 10 optionale Requirements.
Die Punktevergabe findet auf Basis des Erfüllungsgrads der funktionalen und non-funktionalen Requirements statt.
In Summe hat die Prüfung 33 Punkte.
Die Prüfung zählt zu 1/3 ein (1/3 Python, 1/3 Java).

8.3.2. Tabelle der Requirements

ReqID	Punkte	Requirement
ReqFunc01	1	Das Programm soll einen Hilfetext besitzen, welcher per Kommandozeilenparameter **-h** und **--help** aufrufbar ist und die Benutzung Ihres Programms aufzeigt
ReqFunc02	1	Der Hilfetext soll sämtliche Kommandozeilenparameter Ihres Progamms und deren Benutzung erklären. Zugelassene Sprachen sind deutsch und englisch
ReqFunc03	1	Um den Autor identifizieren zu können, soll (im JSON Format) eine Ausgabeblock erstellt werden: Der Namen der abgebenden Person, der Studiengang (TIT,TIK,TIM,TIS sind zulässig) und wenn möglich eine Mail Adresse zur Kontaktaufnahme. Codepage ist **utf-8**. Der Schalter hierfür ist **--programmer-info**

ReqID	Punkte	Requirement
ReqFunc04	1	Soll ein Text in Morsecode gewandelt werden, so wird der Schalter **-e, --encode** gewählt
ReqFunc05	1	Default kann Encode eingestellt werden (wandeln von Text in Morse Code), um beim weglassen des Schalter **-e, --encode** eine Ausgabe zu erzeugen
ReqFunc06	1	Soll ein Text aus Morsecode dekodiert werden, so wird der Schalter **-d, --decode** gewählt
ReqFunc07	1	Ist sowohl der Schalter Encodieren **-e, --encode** wie auch Dokodieren **-d, --decode** gewählt, so soll das Programm mit einer Fehlermeldung abgebrochen werden
ReqFunc08	1	Ist die per Option übergebene Datei nicht vorhanden oder kann nicht geöffnet werden, dann soll eine Fehlermeldung ausgegeben werden (Text passend zum Ereignis)
ReqFunc09	1	Dateien, die gewandelt werden sollen, können in allen beliebigen Ober-, Unterverzeichnissen, hinter Links oder in absoluten Pfaden abliegen (wenn die Zugriffsrechte stimmen)
ReqFunc10	1	Wird per Pipe ein Text an das Programm übergeben, soll aus der Pipe dieser gelesen werden
ReqFunc11	1	Über die Schalter **-o, --out** soll der Text in eine Datei umgeleitet werden, in allen beliebigen Ober-, Unterverzeichnissen, hinter Links oder in absoluten Pfaden (wenn die Zugriffsrechte stimmen)
ReqFunc12	1	Über die Schalter **-o, --out** an Dateien ausgegebene Texte können in allen beliebigen Ober-, Unterverzeichnissen, hinter Links oder in absoluten Pfaden abliegen
ReqFunc13	1	Über eine Pipe sollen an ein nachgelagertes Programm die Ausgabe übertragen werden
ReqFunc14	1	Die Buchstaben **A-Z** sollen in Morsecode wandelbar sein
ReqFunc15	1	Die Buchstaben **a-z** sollen in Morsecode wandelbar sein

ReqID	Punkte	Requirement
ReqFunc16	1	Der Mosecode für die Buchstaben **A-Z,a-z** soll in Großbuchstaben-Klartext rückwandelbar sein
ReqFunc17	1	Die Zahlen **0-9** sollen in Morsecode wandelbar sein
ReqFunc18	1	Der Mosecode für die Buchstaben **0-9** soll in Klartext rückwandelbar sein
ReqFunc19	1	Die Satzzeichen **.,:;?** (Punkt, Komma, Doppelpunkt, Strichpunkt, Fragezeichen) sollen in Morsecode wandelbar sein
ReqFunc20	1	Der Mosecode für die Satzzeichen **.,:;?** (Punkt, Komma, Doppelpunkt, Strichpunkt, Fragezeichen) soll in Klartext rückwandelbar sein
ReqFunc21	1	Die mathematischen Symbole **=-+** (Gleich, Minus, Plus) sollen in Morsecode wandelbar sein
ReqFunc22	1	Der Mosecode für die mathematischen Symbole **=-+** (Gleich, Minus, Plus) soll in Klartext rückwandelbar sein
ReqFunc23	1	Die Formatier Symbole **_()/@** (Unterstrich, Klammer auf/zu, Schrägstrich, Klammeraffe) sollen in Morsecode wandelbar sein
ReqFunc24	1	Der Mosecode für die Formatier Symbole **_()/@** (Unterstrich, Klammer auf/zu, Schrägstrich, Klammeraffe) soll in Klartext rückwandelbar sein
ReqFunc25	1	Nicht vorhandene Buchstaben (Zeichen) sind beim Encodieren als ***** (Stern) aus zu geben
ReqFunc26	1	Buchstaben werden durch ein **SP** (SPACE) getrennt
ReqFunc27	1	Wörter werden durch **SPSPSP** (3*SPACE) getrennt
ReqFunc28	1	Newline und Carriage Return werden ignoriert (Whitespaces)
ReqNonFunc01	1	Die Software soll mit GNU-C/C++ für UNIX und WSL Linux compilierbar sein
ReqNonFunc02	1	Die Software soll unter UNIX und WSL Linux ausführbar sein
ReqNonFunc03	1	Das Erstellen der Software soll mit CMake erfolgen

ReqID	Punkte	Requirement
ReqNonFunc04	1	Der Name des ausführbaren Programm ist **morse**
ReqNonFunc05	1	Das Parsen der Optionen soll über die LibC Funktionalität getopt(...) und getopt_long(...) aus getopt.h erfolgen
ReqOptFunc01	0	Über die Schalter **--slash-wordspacer** soll zwischen Wörter der Space **SP,/,SP** eingefügt werden. Der Buchstabentrenner verändert sich nicht
ReqOptFunc02	0	Der Schalter **--slash-wordspacer** darf nur in Verbindung mit Encode verfügbar sein. Im Fehlerfall muß eine Warnung ausgegeben und das Programm beendet werden
ReqOptFunc03	0	Die Software soll mit MinGW64 compilierbar sein
ReqOptFunc04	0	Die Software soll unter Windows ausführbar sein
ReqOptFunc05	0	Die Dokumentation des Codes soll mit der Syntax von doxygen erfolgen
ReqOptFunc06	0	Die Dokumentation des Codes mit doxygen soll jede Klasse, Methode (Funktion) und Variable abdecken
ReqOptFunc07	0	Die Header- und Source-Dateien ihres Programms sollen sich in verschiedenen Unterverzeichnissen befinden
ReqOptFunc08	0	Das Inkludieren der Header-Dateien soll nicht im aktuellen Verzeichnis erfolgen, sondern im Inlude-Pfad des Compilers
ReqOptFunc09	0	Das Inkludieren von Source-Dateien(*.c / *.cpp) in Source-Dateien ist nicht erlaubt
ReqOptFunc10	0	Die erstellten Header Dateien sollen gegen Mehrfacheinbindung geschützt werden und der Schutz soll kompatibel zum Compiler MinGW (Windows) und GNU-C (Unix) sein