Design Document
„Ex3 – MP3 Database“

Ex3 DD
Einführung in die strukturierte Programmierung
WS 2010/11

Autoren: Ralph Ankele, 0931953
Robin Ankele, 0931951
Muesluem Atas, 0931946
Raphael Sommer,0931955

---

1) Überblick

Das Programm ist klein genug, um in ein einziges Source-File zu passen, dessen Name *ex3.c* sein wird. Das Programm wird aus 50 Funktionen bestehen:

- eine Funktion *CommandPrompt*, welche den Standartprompt esp> ausgibt.
- eine Funktion *checkCommand,* welche den Input des Users überprüft ob es ein Befehl ist.
- eine Funktion *checkInt,* welche überprüft ob ein eingegebener Wert eine Zahl ist.
- eine Funktion *checkDouble,* welche überprüft ob ein eingegebener Wert eine Fließkommazahl ist welche zwischen +5.0 und -5.0 liegt und eine Zahl ist.
- eine Funktion *checkForPoint*, welches eingelesene String auf kommazahl überprüft.
- eine Funktion *checkForAlp,* welches eingelesene String auf Zahlen überprüft.
- eine Funktion *checkFirstChar*, welches nullte stelle von eingelesene String auf plus, minus, kommastelle und Zahl überprüft.
- eine Funktion *biggestID,* welche die höchste ID sucht.
- eine Funktion *print*, welche den Abstand für die Funktion list() ausgibt.
- eine Funktion *newParameters*
- eine Funktion *error,* welche eine Errormessage ausgibt und den verwendeten Speicher freigibt
- eine Funktion *errortext,* welche einen errortext erstellt aus einen Anfangstext, dem Namen der Datenbankdatei und einem Endtext
- eine Funktion *charToInt*, welche ein Char in ein Int Umwandelt.
- eine Funktion *charToDouble*, welche ein Char in ein Double umwandelt.
- eine Funktion *intToChar*, welche ein Int in ein Char umwandelt.
- eine Funktion *doubleToChar*, welche ein Double in ein Char umwandelt.
- eine Funktion *getDatabaseName,* welche den Namen der Datenbankdatei ermittelt.
- eine Funktion *getString,* welche den User-Input einliest und in ein Char- Array speichert.
- eine Funktion *readConsoleChar*, welche eine Char einliest, überprüft und ausgibt.
- eine Funktion *new,* welche einen neuen Datenbankeintrag hinzufügt.
- eine Funktion *view,* welche einen Datenbankeintrag anzeigt
- eine Funktion *edit,* welche einen Datenbankeintrag editiert.
- eine Funktion *delete,* welche einen Datenbankeintrag löscht.

- eine Funktion *list,* welche alle Datenbankeiträge anzeigt.
- eine Funktion *export,* welche die Datenbank in ein HTML-file exportiert.
- eine Funktion *quit,* welche das Programm beendet.
- eine Funktion *writeDatabase,* welche überprüft ob schon ein Eintrag vorhanden ist und dann die Funktion *newEntry* aufruft.
- eine Funktion *sortDatabase*, welche die einzelnen Einträge nach id sortiert.
- eine Funktion *readDatabase*, welche einen Eintrag aus einer doppelt verketteten Liste ausliest.
- *eine Funktion searchFirstEntry,* welche die erste Doppeltverkettete Liste, d.h. die Liste welche die kleinste ID hat findet.
- eine Funktion *deleteDatabase,* welche einen Eintrag aus einer doppelt verketteten Liste löscht.
- eine Funktion *newDatabase,* welche eine neue doppelt verkettete Liste erstellt.
- eine Funktion *newEntry,* welche einen Eintrag in eine neue doppelt verkettete Liste einfügt.
- eine Funktion *deleteEntry,* welche einen Eintrag aus einer doppelt verketteten Liste löscht.
- eine Funktion *writeDatabaseFile,* welche die Einträge aus einer doppelt verketteten Liste in das db-file schreibt.
- eine Funktion *readDatabaseFile,* welche überprüft ob ein db-file Fehlerfrei ist und dann den Inhalt des db-files in die doppelt verkettete Liste speichert.
- eine Funktion *writeHtmlFile,* welche die Einträge aus einer doppelt verketteten Liste in ein HTML-file schreibt.
- eine Funktion *freeDatabase,* welche den Speicher in einer doppelt verketteten Liste frei gibt.
- eine Funktion *freeMemory,* welche Speicher frei gibt.
- eine Funktion *memoryExtension,* welche den Speicher einer Variable erhöht. Mit einer zusätzlich eingebauten Fehleranalyse.
- eine Funktion *printMp3*, welche die Datenbank ausgibt.
- eine Funktion *readfile*, welche aus dem db-File einliest.
- eine Funktion *checkMagic*, welche überprüft ob EPDB am Anfang des db-files steht.
- eine Funktion *getDouble*, welche einen double aus dem db-file liest.
- eine Funktion *getInteger*, welche einen integer aus dem db-file liest.
- eine Funktion *getCString*, welche einen cstring aus dem db-file liest.
- eine Funktion *getHeader*, welche die einzelnen Werte im Header zusammenfügt.
- eine Funktion *getContent,* welche die einzelnen Werte im Content zusammenfügt
- eine Funktion *getContentLength,* welche die Länge des Content berechnet
- eine Funktion *main,* welche den Namen der Datenbank aus dem Kommandozeilenparameter ausliest und dann eine doppelt verkettete Liste erstellt. Dann wird in einer Endlosscheife überprüft ob ein Command eingegeben wird. Diese Schleife kann nur durch die Eingabe des Befehls quit beendet werde, womit dann auch da Programm endet.

2) Module / Funktion

Es folgt eine detailliertere Beschreibung der einzelnen Funktionen:

- **function MP3Database *CommandPrompt(int *sum_of_entries, MP3Database *database, MP3Database *recent_value, int *run, char* filename);**
  Die Funktion liefert nichts zurück. In der Funktion wird für die Eingabe char* Input erstellt, Input aus der Funktion getString() gelesen und dann wird checkCommand Funktion aufgerufen, um zu überprüfen ob die Eingabe ein Befehl ist.

- **function MP3Database *checkCommand(char *input);**

  Die Funktion liefert nichts zurück. In der Funktion wird der vom User eingegeben String überprüft ob es ein Befehl ist. Sollte es ein Befehl sein wird dieser aufgerufen. Sollte kein Befehl eingegeben werden so wird ein Error(„*error: unknown command*“) ausgegeben und der Standartprompt erneut ausgegeben. Danach wird erneut eingelesen und wider überprüft ob ein Befehl eingegeben wurde.

- **function int checkInt(char *input , int *check_double);**

  Die Funktion liefert einen Integer zurück. In der Funktion wird der zu überprüfende Wert an jeder Stelle überprüft ob es eine Zahl ist. Dies wird mit der Funktion is_digit() gemacht. Wenn die Zahl ein Integer ist wird „0“ returned. Ist die Zahl kein Integer wird ein Error ausgegeben und „1“ returned.

- **function int checkDouble(char *input);**

  Die Funktion liefert einen Integer zurück. In der Funktion wird überprüft ob die Zahl eine Fließkommazahl ist, welche zwischen +5.0 und -5.0 liegt. Zusätzlich wird die Funktion *checkInt* aufgerufen und check_double auf „1“ gesetzt damit der eingegebene Wert überprüft wird ob es eine Zahl ist jedoch kein Error ausgeben wird. Sollte der eingegebene Wert keine Fließkommazahl sein wird „1“ returned und ein Error ausgegeben. Sollte der eingegebene Wert eine Fließkommazahl sein so wird „0“ returned.

- **function int checkForPoint(char *input);**

  Die Funktion liefert einen Integer zurück. In der Funktion wird zuerst ein Zähler für durchgehen aller eingelesene Zeichens und ein anderer Zähler für Kommazeichen Anzahl auf null initialisiert. Dann wird mit einer for schleife alle eingelesene Zeichens auf Kommazeichen überprüft, falls ein Kommazahl eingelesen wird, so wird der Kommazeichen Zähler um eins erhöht. Falls der der eingelesene String kein oder mehr als eins Kommazeichen Besitz, dann wird ein Fehlermeldung ausgegeben und es wird solange wiederholt, bis was Richtiges eingegeben wird.

- **function int checkForAlp(char *input);**

  Die Funktion liefert einen integer zurück. Hier wird ein Zähler mit eins initialisiert. Mit Hilfe einer For Schleife wird die eingelesene Input Zeichen für Zeichen kontrolliert. Falls die erste bis Letzte eingelesene Zeichen von String kein Zahl und kein Kommazeichen ist dann wird dann wird ein Fehlermeldung ausgegeben. Hier wird auch solange wiederholt, bis ein gültiges Zeichen eingegeben wird.

- **int checkFirstChar(char *input);**

  Die Funktion liefert einen Integer zurück. Der erste Zeichen von der eingelesen Zeichen wird überprüft. Wenn das Erste Zeichen von der eingelesenen String kein Minuszeichen, kein Pluszeichen, kein Kommazeichen und kein Zahl ist, dann wird ein Fehlermeldung ausgegeben.

- **function int biggestID(int *sum_of_entries, MP3Database *database);**

  Die Funktion liefert einen Integer zurück. Hier wird mit Hilfe einer for schleife alle Einträge durchgegangen, um den Eintrag mit größte ID zu finden. Der variable biggest ID wird am Anfang auf 0 gesetzt und falls die neue ID größer als der alte ID ist, dann wird die neue ID als biggestID gewählt.

- **function char *print(char *printing_text, int biggest_value, int id, char *title,char *artist);**

  Die Funktion liefert einen Char pointer zurück. Hier wird die stellen der ID mit Leerzeichen auf die jeweils größte vorhandene Zahl normiert. Wenn zum Beispiel der größte ID 325 ist, dann wird die abstand auf 3 normiert. Die Anzahl der Stellen werden mit log10() Funktion bestimmt und für diese Funktion wird Library math.h benötigt. Hier wird zuerst überprüft, ob id 0 ist. Falls id 0 ist, dann wird die Normierung in Abhängigkeit von biggestvalue bestimmt, ansonsten wird die Normierung von biggest Value – die Normierung von id bestimmt.

- **function void error(int errorcode, char *errortext, char *mem, int *errorflag);**

  Diese Funktion liefert nichts zurück. Die Funktion übergibt einem Pointer einen Errorwert mit welchen das Hauptprogramm abbrechen kann. Zusätzlich wird eine Errormessage ausgegeben welche dem Pointer errortext übergeben wurde. Die Funktion löscht auch noch den allozierten Speicher und gibt diesen frei.

- **function char* errortext(char *front_text, char *filename, char *end_text);**

  Die Funktion liefert einen Char Pointer auf einen Errortext zurück. In der Funktion wird der Errortext erstellt. Der Errortext wird zusammengesetzt aus dem

Anfangstext "*error: dbfile*“ sowie dem Filenamen „*[filename]*“ und der Endung „*corrupt\n*"

- **function double charToDouble(char *input);**

  Die Funktion liefert ein double zurück. Hier wird ein Character mit hilfe von atof Funktion in einem Double (float) umgewandelt.

- **function char *intToChar(int input);**

  Die Funktion liefert ein char pointer zurück. Hier wird ein Integer in einem Char umgewandelt. Nachher wird ein Speicher angefordert und auf die initialisierte char pointer buffer gespeichert und input mit einem sprintf ausgegeben (sprintf von eine Konsole in Variable).

- **function char *doubleToChar(double input);**

  Die Funktion liefert ein char pointer zurück. Hier wird ein Double in einem Char umgewandelt. Nachher wird ein Speicher angefordert und auf die initialisierte char pointer buffer gespeichert und input mit einem sprintf ausgegeben.

- **function char *getDatabasename(char **parameter);**

  Die Funktion gibt einen Char Pointer auf den Namen des Datenbank-files zurück. In der Funktion wird aus dem Komanndozeilenpaarameter ausgelesenen Eingabe der Datenbankname extrahiert. Zusätzlich wird noch überprüft ob das Programm richtig gestartet wurde(falls mehr als ein Kommandozeilenparameter eingegeben wurde oder das Programm nicht mit "-o=" beginnt). In diesem Fall wird da Programm mit Returnwert „-1“ beendet und ein Error("usage: ex3 -o=dbfile\n") wird ausgegeben.

- **function char *getString(int *strlenght, int *errorflag);**

  Die Funktion liefert einen Char Pointer auf den Inhalt des vom User eingegebenen Input zurück. Der eingelesene Text wird auf ein Char-Array gespeichert für welches dynamisch Speicher angefordert wird. Der Text wird solange eingelesen bis das Ende des Char(mit Abfrage ob ,\0’) festgestellt wird. Sollte kein dynamischer Speicher mehr vorhanden sein wird eine Errormeldung *„Out of Memory“* ausgegeben und die Funktion gibt nichts zurück, setzt jedoch einen errorflag pointer auf -2.

- **function char *readConsoleChar(char *name, int *abort);**

  Die Funktion liefert einen char pointer zurück. Hier wird zuerst mit einer if gefragt, ob count gesetzt ist, wenn ja, dann wird nicht zurück geliefert. Solange etwas

eigegeben wird, wird die entsprechende ausgabe gemacht (id, title, artist,..), input aus dem getstring gelesen und input wird zurück gegeben.

- **function MP3Database *new(int *sum_of_entries, MP3Database *database);**

  Die Funktion liefert nichts zurück. In der Funktion wird ein neuer Datensatz angelegt. Dieser fragt folgende Parameter ab: "id? " "title? " "artist? " "file? " "rating? " "comment? ". Nach dem einlesen der Werte werden diese der Funktion writeDatabase übergeben.

- **function void view(int *sum_of_entries, MP3Database *database);**

  Die Funktion liefert nichts zurück. Die Funktion liest die id ein und gibt für diesen Datensatz den Inhalt aus. Nachdem dem einlesen der id wird diese der Funktion readDatabase übergeben welche dann den Datensatz ausgibt.

- **function void edit(int *sum_of_entries, MP3Database *database, MP3Database *recent_value);**

  Die Funktion liefert nichts zurück. In dieser Funktion wird ein Datensatz editiert. Es wird zuerst die Funktion view() aufgerufen mit welcher ein Datensatz angezeigt wird. Danach kann man die Werte neu eingeben. Sollte kein Wert eingegeben werden so bleibt der alte Wert erhalten. Die neue eingegeben Werte werden der Funktion writeDatabase übergeben.

- **function void delete(int *sum_of_entries, MP3Database *database);**

  Die Funktion liefert nichts zurück. Diese Funktion löschte einen Datensatz unwiderruflich. Dabei wird zuerst die id des zu löschenden Datensatzes abgefragt und danach der Funktion deleteDatabase übergeben in welcher der Datensatz gelöscht wird.

- **function void list(int *sum_of_entries, MP3Database *database);**

  Die Funktion liefert nichts zurück. Diese Funktion listet die vorhandenen Datensätze untereinander auf. Es wird nach id, titel und intepret aufgelistet. Wobei die id nach der größten vorkommenden Zahl zu normieren ist. Die Funktion ruft die Funktion readDatabase auf welche die Datensätze auflistet.

- **function void export(int *sum_of_entries, MP3Database *database);**

  Die Funktion liefert nichts zurück. Die Funktion fragt den Namen für die zu exportierende Datei ab. Danach wir die Funktion xxx aufgerufen welche den Inhalt der Datenbank in ein HTML file speichert.

- **function void quit(int *run, char *filename, int *sum_of_entries, MP3Database *database);**

  Die Funktion liefert nichts zurück. Wenn die Funktion aufgerufen wird, wird die Variable run im Hauptprogramm auf „0“ gesetzt und das Programm damit beendet.

- **function MP3Database *writeDatabase(int existing_value, int id, Header *new_header, Content *new_content, int *sum_of_entries, MP3Database *database);**

  Die Funktion gibt einen Pointer auf einen Datentyp Mp3Database zurück. Die Funktion überprüft ob schon ein Eintrag mit derselben id vorhanden ist. Wenn einer vorhanden ist wird dieser upgedated und zurückgegeben. Sollte noch keiner vorhanden sein so wird die Funktion *newEntry* aufgerufen. Die Anzahl der Einträge wird erhöht und der neue Eintrag zurückgegeben.

- **function void sortDatabase(int id, Header *new_header, Content *new_content, int *sum_of_entries, MP3Database *database);**

  Die Funktion gibt einen Pointer auf einen Datentyp Mp3Database zurück. In der Funktion wird mit Hilfe einer For Schleife, alle eintrage durchgegangen und die ID sortiert. Dann werden alle ID aufsteigend sortiert.

- **function MP3Database *readDatabase(int id, int *sum_of_entries, MP3Database *database);**

  Die Funktion gibt einen Pointer auf einen Datentyp MP3Database zurück. In der Funktion wird die ganze doppelt verkettete Liste nach der eingegeben id durchsucht und der Eintrag mit der gesuchten id wird zurückgegeben.

- **function MP3Database *searchFirstEntry(int *sum_of_entries, MP3Database *database);**

  Die Funktion gibt einen Pointer auf einen Datentyp MP3Database zurück. In der Funktion wird die kleinste ID gesucht, damit man weißt wo der Doppeltverkettete Liste anfängt. Hier wird zuerst wieder mit eine for Schleife alle Einträge durchgegangen, Falls der smallest_id größer als neu ID ist, dann neue ID als smallest ID gespeichert.

- **function void deleteDatabase(int id, int *sum_of_entries, MP3Database *database, char *filename);**

Diese Funktion liefert nichts zurück. In der Funktion wird die ganze doppelt verkettete Liste nach der eingegeben id durchsucht und sollte sie gefunden werden wird die Funktion deleteEntry aufgerufen welche den Eintrag löscht. Zusätzlich wird die Summe der Einträge um eins reduziert.

- **function MP3Database *newDatabase();**

  Die Funktion gibt einen Pointer auf einen Datentyp MP3Database zurück. In dieser Funktion wird eine neue verkettete Liste erstellt. Dabei zeigt ein Pointer immer auf den nächsten Wert sowie ein Pointer immer auf den vorhergehenden Wert.

- **function MP3Database *newEntry(MP3Database *database, Header *header, Content *new_content);**

  Die Funktion gibt einen Pointer auf einen Datentyp MP3Database zurück. In dieser Funktion wird eine neuer Eintrag in eine verkettete Liste hinzugefügt. Dabei werden zuerst die Pointer, welche auf den follgenden und vorherigen Eintrag zeigen, aufgelöst und nachdem der neue Eintrag hinzugefügt wurde wieder gesetzt. Jedoch nun mit dem neuen Eintrag in der Liste.

- **void deleteEntry(MP3Database *database);**

  Diese Funktion liefert nichts zurück. In der Funktion wird ein Eintrag in der Liste gelöscht. Dabei werden wieder die Pointer welche auf den follgenden und nächsten Eintrag in der Liste stehen aufgelöst. Dann werden die Pointer neu gesetzt ohne dem zu löschenden Eintrag. Am Ende wird der Speicher des zu löschenden Eintrages wieder freigegeben.

- **function void writeDatabaseFile(char *filename, char *content, int *sum_of_entries);**

  Diese Funktion liefert nichts zurück. Diese Funktion schreibt die Einträge in der doppelt verketteten Liste in das db-file.

- **void readDatabaseFile(char *filename, int *sum_of_entries, MP3Database *database)**

  Diese Funktion liefert nichts zurück. In dieser Funktion wird überprüft ob das db-file korrekt ist. Danach wird in dieser Funktion der Inhalt des db-files in die doppelt verkettete Liste eingefügt.

- **function void writeHtmlFile(char *filename, char *content);**

  Diese Funktion liefert nichts zurück. In dieser Methode werden die Einträge der doppelt verketteten Liste in ein HTML-file geschrieben.

- **function void freeDatabase(MP3Database *database);**

  Diese Funktion liefert nichts zurück. In dieser Funktion wird der Speicher in einer doppelt verketteten Liste wieder freigegeben wenn ein Eintrag gelöscht wird.

- **function void freeMemory(char* mem);**

  Diese Funktion liefert nichts zurück. In dieser Funktion wird angefordertet Speicher wieder freigegeben.

- **function char *memoryExtension(char *extendvariable, int *size, int *errorflag);**

  Diese Funktion liefert einen Char-Pointer zurück. In der Funktion wird der Speicher eines Strings dynamisch erweitert. Der String wird danach wider zurückgegeben.

- **function void printMp3(MP3Database mp3);**

  Diese Funktion liefert nichts zurück. In der Funktion wird der Header und der Content eines Database Eintrages ausgegeben.

- **function char* readFile();**

  Diese Funktion liefert einen Char-Pointer zurück. In der Funktion wird aus dem db-file eingelesen und der Inhalt auf einen String gespeichert. Dieser String wird dann zurückgegeben.

- **function int checkMagic(char* content);**

  Diese Funktion liefert einen Integer zurück. In der Funktion wird überprüft ob an den ersten 4 Stellen des db-files die Buchstaben EPDB stehen. Sollte dies der Fall sein wird 1 returned. Sollte das File nicht mit der Buchstabenkombination beginnen wird 0 returned.

- **function double getDouble(char* content, int start);**

  Diese Funktion liefert einen Double Wert zurück. In der Funktion wird aus dem db-file ein double Wert ausgelesen. Dieser Wert wird dann zurückgegeben.

- **function unsigned int getInteger(char* content, int start);**

Diese Funktion liefert einen Integer Wert zurück. In der Funktion wird aus dem db-file ein integer Wert ausgelesen. Dieser Wert wird dann zurückgegeben.

- **function char* getCString(char* content, int* start);**

  Diese Funktion liefert einen CString Wert zurück. In der Funktion wird aus dem db-file ein CString Wert ausgelesen. Dieser Wert wird dann zurückgegeben.

- **function void getHeader(char* content, Header* header, int* str_show);**

  Diese Funktion liefert nichts zurück. In der Funktion werden die einzelnen Werte für den Header zusammengefügt.

- **function void getContent(char* content, Content* cont, int* str_show);**

  Diese Funktion liefert nichts zurück. In der Funktion werden die einzelnen Werte für den Content zusammengefügt.

- **function unsigned int getContentLength(Content* content)**

  Diese Funktion liefert die Länge des Contents zurück. Es werden mit Hilfe von strlen alle Längen berechnet.

- **int main(int argc, char *argv[])**

  Die main liest den Datenbasis Namen ein. Danach liest sie das DatenbasisFile in die doppelt verkettete Liste ein. Danach wird in die While Schleife gesprungen, wo gewartet wird bis der User eine Eingabe macht. Diese wird dann über die Funktion commandPrompt eingelesen und bearbeitet. Wenn run auf 0 gesetzt wird wird die Schleife verlassen, die doppelt verkettete Liste gefreed und danach das Programm beendet.
  Das Programm wird mit folgenden return Werten beendet:

- „0“: Das Programm ist Fehlerfrei abgelaufen.
- „-1“: Wenn mehr als ein Parameter eingegeben wird oder der Parameter beginnt nicht mit „–o=“.
- „-2“: Es ist keine dynamischer Speicher mehr verfügbar.
- „-3“: Wenn aus der Datenbankdatei dbfile nicht gelesen werden kann
- „-4“: Wenn die Datei dbfile corrupt ist.
- „-5“: Wenn auf die Datenbankdatei dbfile nicht geschrieben werden kann.