Automatiser dit studievalg

Generated by Doxygen 1.8.13

Tue Dec 10 2019 18:03:40

Contents

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 database Struct Reference 5

3.2 Database Struct Reference 5

3.2.1 Detailed Description 5

3.3 education Struct Reference 6

3.3.1 Detailed Description 6

3.4 location Struct Reference 6

3.5 profile Struct Reference 6

3.5.1 Detailed Description 7

3.6 qualification Struct Reference 7

3.7 subject Struct Reference 7

3.8 vector Struct Reference 7

4 File Documentation **9**

4.1 commands.c File Reference 9

4.2 commands.h File Reference 9

4.2.1 Detailed Description 11

4.2.2 Function Documentation 11

4.2.2.1 classNameStr() 11

4.2.2.2 convertScale() 11

4.2.2.3 deleteCmd() 12

4.2.2.4 evalCmd() 12

4.2.2.5 findCmd() 12

4.2.2.6 getEmptyIndex() 13

4.2.2.7 getIndex() 13

4.2.2.8 isQualified() 13

4.2.2.9 levelAsValue() 14

4.2.2.10 listCmd() 14

4.2.2.11 listIsFull() 14

4.2.2.12 recommendCmd() 14

4.2.2.13 saveCmd() 15

4.2.2.14 saveProfile() 15

4.2.2.15 searchCmd() 15

4.2.2.16 setImportantSubjects() 16

4.2.2.17 setOtherSubjects() 16

4.2.2.18 setProfileInterests() 16

4.2.2.19 setProfileLocation() 17

4.2.2.20 setProfileQualifications() 17

4.2.2.21 setSubjects() 17

4.2.2.22 surveyCmd() 17

4.2.2.23 validScaleValue() 18

4.3 constants.h File Reference 18

4.3.1 Detailed Description 19

4.4 database.h File Reference 19
4.4.1 Detailed Description 19
4.5 education.h File Reference 19
4.5.1 Detailed Description 20
4.5.2 Function Documentation 20
4.5.2.1 createArrayOfEducations() 20
4.5.2.2 createDefaultEducation() 20
4.6 parser.h File Reference 21
4.6.1 Detailed Description 22
4.6.2 Function Documentation 22
4.6.2.1 createArrayOfStrings() 22
4.6.2.2 findDatabaseLine() 22
4.6.2.3 parseDatabase() 22
4.6.2.4 parseDatabaseLine() 24
4.6.2.5 parseEduDesc() 24
4.6.2.6 parseEduNames() 24
4.6.2.7 parseEduRegion() 25
4.6.2.8 parseEduString() 25
4.6.2.9 parseInterestNames() 26
4.6.2.10 parseInterestValues() 26
4.6.2.11 parseNumOfEdu() 26
4.6.2.12 parseNumOfInterests() 26
4.6.2.13 parseReqGrade() 27
4.6.2.14 parseSubReq() 27
4.6.2.15 readReqString() 27
4.6.2.16 strToReg() 28
4.7 profile.h File Reference 28
4.7.1 Detailed Description 29
4.7.2 Function Documentation 29
4.7.2.1 createProfile() 29

4.7.2.2 freeProfile() 29
4.7.2.3 printProfile() 29
4.8 region.h File Reference 30
4.8.1 Detailed Description 30
4.8.2 Enumeration Type Documentation 30
4.8.2.1 region 30
4.9 subjects.h File Reference 30
4.9.1 Detailed Description 31
4.9.2 Function Documentation 31
4.9.2.1 charToLevel() 31
4.9.2.2 levelToChar() 32
4.9.2.3 stringToClass() 32
4.10 vector.h File Reference 32
4.10.1 Detailed Description 33
4.10.2 Function Documentation 33
4.10.2.1 addVector() 33
4.10.2.2 copyVector() 33
4.10.2.3 createVector() 34
4.10.2.4 dotProduct() 34
4.10.2.5 freeVector() 34
4.10.2.6 freeVectorM() 35
4.10.2.7 lengthOfVector() 35
4.10.2.8 normalizeVector() 35
4.10.2.9 printVector() 35
4.10.2.10 scaleVector() 36
4.10.2.11 subtractVector() 36

Index **37**

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

database 5
Database
A structure to store a database 5
education
Describes an education and all it requirements 6
location 6
profile
Describes a user 6
qualification 7
subject 7
vector 7

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

commands.c 9
commands.h
Contains functions related to command handling 9
constants.h
Contains symbolic constants used throughout the program 18
database.h
Contains elements relating to the database 19
education.h
Contains elements relating to educations 19
parser.h
Contains elements relating to parsing the database 21
profile.h
Contains elements relating to user profiles 28
region.h
Contains geographical elements 30
subjects.h
Contains code regarding subjects and qualifcations for educations 30
vector.h
Contains elements relating to vectors 32

Chapter 3

Class Documentation

3.1 database Struct Reference

Collaboration diagram for database:

Public Attributes

- int **amount_of_educations**
- struct education ∗ educations

    the amount of educations in the database
- int amount_of_interests

    an array of educations delimited by amount_of_educations
- char ∗∗ interest_string

    the amount of interests in the database

The documentation for this struct was generated from the following file:

- database.h

3.2 Database Struct Reference

A structure to store a database.

```
#include <database.h>
```

3.2.1 Detailed Description

A structure to store a database.

The documentation for this struct was generated from the following file:

- database.h

3.3 education Struct Reference

Describes an education and all it requirements.

`#include <education.h>`

Collaboration diagram for education:

Public Attributes

- char ∗ **name**
- char ∗ **description**
- char ∗ **link**
- enum region **region**
- double **required_grade**
- struct vector **interests**
- struct qualification **required_qualifications**

3.3.1 Detailed Description

Describes an education and all it requirements.

A structure, which contains amount_of_educations educations.

This structure defines an education and all the details about the education.

The documentation for this struct was generated from the following file:

- education.h

3.4 location Struct Reference

Public Attributes

- enum region **region**
- double **region_importance**

The documentation for this struct was generated from the following file:

- region.h

3.5 profile Struct Reference

Describes a user.

`#include <profile.h>`

Collaboration diagram for profile:

Public Attributes

- struct vector **interests**
- struct vector **adjustment_vector**
- char **name** [MAX_NAME_LENGTH]
- struct qualification **qualifications**
- double **average**
- struct location **location**
- char **saved_educations** [EDUCATION_LIST_LENGTH][MAX_EDU_NAME_LENGTH]
- int **last_recommended**
- char **recommended_educations** [EDUCATION_LIST_LENGTH][MAX_EDU_NAME_LENGTH]

3.5.1 Detailed Description

Describes a user.

This structure defines the profile of a user and all the details about the user

The documentation for this struct was generated from the following file:

- profile.h

3.6 qualification Struct Reference

Collaboration diagram for qualification:

Public Attributes

- int **amount_of_subjects**
- struct subject ∗ subjects

    the amount of subjects in qualifications

The documentation for this struct was generated from the following file:

- subjects.h

3.7 subject Struct Reference

Public Attributes

- enum level level

    the name of the subject

The documentation for this struct was generated from the following file:

- subjects.h

3.8 vector Struct Reference

Public Attributes

- double ∗ **array**
- int **size**

The documentation for this struct was generated from the following file:

- vector.h

Chapter 4

File Documentation

4.1 commands.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <unistd.h>
#include "profile.h"
#include "education.h"
#include "subjects.h"
#include "vector.h"
#include "commands.h"
#include "constants.h"
```
Include dependency graph for commands.c:

4.2 commands.h File Reference

Contains functions related to command handling.

```
#include "profile.h"
#include "education.h"
#include "subjects.h"
#include "vector.h"
#include "database.h"
```
Include dependency graph for commands.h: This graph shows which files directly or indirectly include this file:

Functions

- void menuCmd (void)

    Prints all the possible commands the user can use.
- void surveyCmd (struct profile ∗user, const struct database ∗db)

    Tests the current user for name, location, interests, qualifications and average grade.
- void setProfileLocation (struct profile ∗user)

    Sets the region of choice in user.

- double convertScale (int initial_value)

    Returns the converted value.
- int validScaleValue (int value, int interval_start, int interval_end)

    Returns a value between interval_start and interval_end.
- int getValidInteger (void)

    Returns a valid integer given through the terminal.
- void setProfileInterests (struct profile ∗user, const struct database ∗db)

    Saves all interests to user as a converted value (see convertScale)
- void setProfileQualifications (struct profile ∗user)

    Saves all the users qualifications as given by the terminal.
- void setSubjects (struct profile ∗user)

    Sets all qualifications in user to match the enum class.
- void setImportantSubjects (struct profile ∗user)

    Saves all the qualifications for the important subjects.
- const char ∗ classNameStr (enum class class)

    Returns the name as a string of a class given as an enum class.
- enum level levelAsValue (char c)

    Returns the enum value of a level given as a character.
- void setOtherSubjects (struct profile ∗user, int start, int end)

    Saves all the levels of the other subjects (not the important ones)
- void **chooseFromList** (struct profile ∗user, int interval_start, int interval_end)
- double getValidDouble (void)

    Returns a valid double entered in the terminal.
- void evalCmd (struct profile ∗user, struct education ∗current_education, int arg)

    Changes the adjustment vector for the user to approach the current education.
- struct education findCmd (char ∗arg, const struct database ∗db)

    Finds and prints out the education with the exact name given as and argument.
- void searchCmd (char ∗arg, const struct database ∗db)

    Finds and prints out the educations whose name contains the given argument.
- struct education recommendCmd (struct profile ∗user, const struct database ∗database)

    Goes trough the available educations and compares them to the user: Both their interests, qualifications and location are considered.
- int isQualified (struct profile user, struct education education)

    Checks if the user has the subject levels required by the education.
- const char ∗ **getRegionName** (enum region r)
- void **printEducation** (struct education)
- void saveCmd (struct profile ∗user, struct education ∗current_education)

    Saves the given education to a list in the profile struct.
- int getIndex (char edu_array[EDUCATION_LIST_LENGTH][MAX_EDU_NAME_LENGTH], struct education target)

    Returns the index of the given target in the array.
- int getEmptyIndex (char edu_array[EDUCATION_LIST_LENGTH][MAX_EDU_NAME_LENGTH])

    Returns an index with an empty string in the given array.
- int listIsFull (int i)

    A logical statement that returns a boolean value.
- void clearBuffer (void)

    Empties the buffer for standard input.
- void listCmd (const struct profile ∗user)

    Prints out the names of all the saved educations.
- void deleteCmd (struct profile ∗user, int deleted_entry)

    Removes the name of the education at the given index.

- void saveProfile (struct profile user)

  Saves a file with the information collected about the user.
- int **checkProfile** (const char name[ ])
- struct profile **loadProfile** (char ∗name, int number_of_interests)

4.2.1 Detailed Description

Contains functions related to command handling.

Contains all of the functions used for handling commands, such as those relating to verifying input and the functions that act on receiving a command.

4.2.2 Function Documentation

4.2.2.1 classNameStr()

```
const char * classNameStr (
             enum class class  )
```

Returns the name as a string of a class given as an enum class.

Parameters

class	The enum value the name should return for

4.2.2.2 convertScale()

```
double convertScale (
             int v )
```

Returns the converted value.

Parameters

v	The value to be converted

Returns

A double value between -1 and 1 given that the input is between 0 and 10

4.2.2.3 deleteCmd()

```
void deleteCmd (
            struct profile * user,
            int deleted_entry )
```

Removes the name of the education at the given index.

Parameters

user	The profile struct for the user

4.2.2.4 evalCmd()

```
void evalCmd (
            struct profile * user,
            struct education * current_education,
            int arg )
```

Changes the adjustment vector for the user to approach the current education.

The distance of the change is determined by the argument

Parameters

current_education	The education currently being displayed
user	The profile struct whose adjustment vector is changed
arg	The user input argument for how much to change the adjustment vector

4.2.2.5 findCmd()

```
struct education findCmd (
            char * arg,
            const struct database * db )
```

Finds and prints out the education with the exact name given as and argument.

Parameters

arg	The argument string which should be the name of an education
database	The database in which all educations are stored

Returns

A struct for the education found

4.2.2.6 getEmptyIndex()

```
int getEmptyIndex (
            char edu_array[EDUCATION_LIST_LENGTH][MAX_EDU_NAME_LENGTH] )
```

Returns an index with an empty string in the given array.

Parameters

edu_array	An array of strings in which the empty string should be found

4.2.2.7 getIndex()

```
int getIndex (
            char edu_array[EDUCATION_LIST_LENGTH][MAX_EDU_NAME_LENGTH],
            struct education target )
```

Returns the index of the given target in the array.

Parameters

edu_array	An array of strings
target	An education whose name is to be found in the array

4.2.2.8 isQualified()

```
int isQualified (
            struct profile user,
            struct education education  )
```

Checks if the user has the subject levels required by the education.

Parameters

user	The profile struct whose quailification is checked
education	The education struct with the requirements

Returns

0 if the user does not have the required levels and 1 if the user does

4.2.2.9 levelAsValue()

```
enum level levelAsValue (
            char c )
```

Returns the enum value of a level given as a character.

Parameters

c	The level as a character to be converted to enum level

4.2.2.10 listCmd()

```
void listCmd (
            const struct profile * user )
```

Prints out the names of all the saved educations.

Parameters

user	The profile struct for the user

4.2.2.11 listIsFull()

```
int listIsFull (
            int i )
```

A logical statement that returns a boolean value.

Parameters

i	The index of an array of education structs 1 if the index is -1 and 0 otherwise

4.2.2.12 recommendCmd()

```
struct education recommendCmd (
            struct profile * user,
            const struct database * database )
```

Goes trough the available educations and compares them to the user: Both their interests, qualifications and location are considered.

Parameters

user	The profile struct which is compared
database	The database containing the educations

Returns

A struct for the recommended education.

4.2.2.13 saveCmd()

```
void saveCmd (
            struct profile * user,
            struct education * current_education )
```

Saves the given education to a list in the profile struct.

Parameters

current_education	A pointer to an education
user	The profile struct of the user in which the education is saved

4.2.2.14 saveProfile()

```
void saveProfile (
            struct profile user )
```

Saves a file with the information collected about the user.

Parameters

user	The profile struct for the user

4.2.2.15 searchCmd()

```
void searchCmd (
            char * arg,
            const struct database * db )
```

Finds and prints out the educations whose name contains the given argument.

Parameters

arg	The argument string which should be contained in the name of an education
database	The database in which all educations are stored.

4.2.2.16 setImportantSubjects()

```
void setImportantSubjects (
            struct profile * user )
```

Saves all the qualifications for the important subjects.

Parameters

user	The profile struct where the subjects are saved to

4.2.2.17 setOtherSubjects()

```
void setOtherSubjects (
            struct profile * user,
            int start,
            int end )
```

Saves all the levels of the other subjects (not the important ones)

Parameters

user	The profile struct where the qualifications are to be saved
start	The start of the subjects to be asked for
end	The ens of the subjects to be asked for

4.2.2.18 setProfileInterests()

```
void setProfileInterests (
            struct profile * user,
            const struct database * db )
```

Saves all interests to user as a converted value (see convertScale)

Parameters

user	The profile struct where the interests are saved to
db	The database struct where information about all interests are saved as a pointer

4.2.2.19 setProfileLocation()

```
void setProfileLocation (
             struct profile * user )
```

Sets the region of choice in user.

Saves the interest in studying in this location

Parameters

user	The profile struct where the information about location should be saved

4.2.2.20 setProfileQualifications()

```
void setProfileQualifications (
             struct profile * user )
```

Saves all the users qualifications as given by the terminal.

Parameters

user	The profile struct where the qualifications are saved to

4.2.2.21 setSubjects()

```
void setSubjects (
             struct profile * user )
```

Sets all qualifications in user to match the enum class.

Parameters

user	The profile struct where the subjects are saved to

4.2.2.22 surveyCmd()

```
void surveyCmd (
             struct profile * user,
             const struct database * db )
```

Tests the current user for name, location, interests, qualifications and average grade.

Parameters

user	The profile struct where all survey results are saved
db	The database where information of interests and subjects are as a pointer

4.2.2.23 validScaleValue()

```
int validScaleValue (
            int value,
            int interval_start,
            int interval_end )
```

Returns a value between interval_start and interval_end.

If the given value outside the interval it will return the value inside the interval closest the value. The interval_start must be less than the interval_end

Parameters

value	The value to check within the scale
interval_start	The start value of the scale
interval_end	The end value the scale

4.3 constants.h File Reference

Contains symbolic constants used throughout the program.

This graph shows which files directly or indirectly include this file:

Macros

- #define **VERSION** "1.0.1"
- #define **NUMBER_OF_REGIONS** 5
- #define **IMPORTANT_SUBJECTS** 5
- #define **OTHER_SUBJECTS** 11
- #define **LANGUAGE_SUBJECTS** 11
- #define **USELESS_SUBJECTS** 2
- #define **TOTAL_SUBJECTS** (IMPORTANT_SUBJECTS + OTHER_SUBJECTS + LANGUAGE_SUBJEC↵ TS)
- #define **MAX_NAME_LENGTH** 20
- #define **MAX_FILE_NAME_LENGTH** MAX_NAME_LENGTH + 12
- #define **EDUCATION_LIST_LENGTH** 10
- #define **MAX_EDU_NAME_LENGTH** 40
- #define **MAX_COMMAND_LENGTH** 10

- #define **MAX_INPUT_LENGTH** (MAX_COMMAND_LENGTH + 100)
- #define **NOT_IN_LIST** -1
- #define **NO_EMPTY_INDEX** -1
- #define **FIELD_SIZE** 25
- #define **ADJUSTMENT_CONSTANT** 0.1
- #define **STRING_MAX_LENGTH** 10000
- #define **TABS** ' '
- #define **NOT_FOUND_STRING** " "
- #define **EDU_MAX_SUBJECTS** 10
- #define **DATABASE_PATH** "./bin/data/database.txt"

4.3.1 Detailed Description

Contains symbolic constants used throughout the program.

This header-file contains all of the symbolic constants used throughout the entire program, such as those relating to the number of regions, the max lenght of strings or constants used for string formatting.

4.4 database.h File Reference

Contains elements relating to the database.

```
#include "education.h"
```
Include dependency graph for database.h: This graph shows which files directly or indirectly include this file:

Classes

- struct database

Functions

- void **freeDatabase** (struct database ∗)
- struct database ∗ **createDatabase** (char ∗)
- struct education ∗ **findEducation** (char ∗, struct database ∗)

4.4.1 Detailed Description

Contains elements relating to the database.

Contains the database struct and functions for creating, freeing and finding educations.

4.5 education.h File Reference

Contains elements relating to educations.

```
#include "region.h"
#include "subjects.h"
#include "vector.h"
```
Include dependency graph for education.h: This graph shows which files directly or indirectly include this file:

Classes

- struct education

  Describes an education and all it requirements.

Functions

- struct education createDefaultEducation (int amount_of_interests, int amount_of_subjects)

  Assigns default values to the fields of the education struct.
- struct education ∗ createArrayOfEducations (int amount_of_educations)

  Allocate memory for an array of educations and return a pointer to it.
- void **freeEducation** (struct education ∗)

4.5.1 Detailed Description

Contains elements relating to educations.

This file contains the education struct and the function that creates educations.

4.5.2 Function Documentation

4.5.2.1 createArrayOfEducations()

```
struct education * createArrayOfEducations (
              int amount_of_educations )
```

Allocate memory for an array of educations and return a pointer to it.

Parameters

amount_of_educations	The amount of educations to be stored in the array

4.5.2.2 createDefaultEducation()

```
struct education createDefaultEducation (
              int amount_of_interests,
              int amount_of_subjects )
```

Assigns default values to the fields of the education struct.

Parameters

amount_of_interests	The number of interests the education should hold
amount_of_subjects	The number of subjects the education should hold

4.6 parser.h File Reference

Contains elements relating to parsing the database.

```
#include <stdio.h>
#include <stdlib.h>
#include "database.h"
#include "region.h"
```
Include dependency graph for parser.h:

Functions

- void parseDatabase (struct database ∗database, FILE ∗filereader)

  Parse the database file and set all values in the database.
- void parseDatabaseLine (const char key[ ], struct database ∗database, FILE ∗filereader)

  Parse the line containing key and return into database.
- void findDatabaseLine (const char key[ ], FILE ∗filereader, char ∗current_line)

  Search the database until the first word of a line matches with key.
- int parseNumOfEdu (FILE ∗filereader)

  Returns the number of educations from database file.
- int parseNumOfInterests (FILE ∗filereader)

  Parse/count the number of intersts in the database file and return as int.
- void parseEduNames (int amount_of_educations, struct education ∗educations, char current_line[ ])

  Parses the name for each education.
- void parseEduDesc (int amount_of_educations, struct education ∗educations, char current_line[ ])

  Parses the description for each education.
- void **parseEduLink** (int amount_of_educations, struct education ∗educations, char current_line[ ])
- void parseEduRegion (int amount_of_educations, struct education ∗educations, char current_line[ ])

  Parses the region for each education.
- void parseSubReq (int amount_of_educations, struct education ∗educations, char current_line[ ])

  Parses the subject requirements for each education.
- void parseReqGrade (int amount_of_educations, struct education ∗educations, char current_line[ ])

  Parses the required average grade for each education.
- void parseInterestNames (struct database ∗database, FILE ∗filereader)

  Parse the names of each interest and return to the database.
- void parseInterestValues (int amount_of_interests, int amount_of_educations, struct education ∗educations, FILE ∗filereader)

  Parse the values for each interest in all educations and return into educations.
- char ∗ parseEduString (char ∗current_line, int amount_of_educations, int offset)

  Scans the current line + i until TABS or newline.
- char ∗∗ createArrayOfStrings (int amount_of_strings)

  Allocate memory for an array of strings and return a pointer to it.
- int **sseek** (char ∗, char)
- void readReqString (struct qualification ∗, char ∗, int)

  Read a requiremnt from a string.
- enum region strToReg (char ∗region_string)

  Converts a string into an enum region and return an enum region.

4.6.1 Detailed Description

Contains elements relating to parsing the database.

<Detailed esription="" here>="">

4.6.2 Function Documentation

4.6.2.1 createArrayOfStrings()

```
char ** createArrayOfStrings (
             int amount_of_strings )
```

Allocate memory for an array of strings and return a pointer to it.

Parameters

amount_of_strings	The amount of strings to be stored in the array

4.6.2.2 findDatabaseLine()

```
void findDatabaseLine (
             const char key[],
             FILE * filereader,
             char * current_line )
```

Search the database until the first word of a line matches with key.

Return the line through current_line. If line does not exist, return NOT_FOUND_STRING.

Parameters

key	The term to search for
filereader	The database file
current_line	Return through this parameter

4.6.2.3 parseDatabase()

```
void parseDatabase (
             struct database * database,
             FILE * filereader )
```

Parse the database file and set all values in the database.

Parameters

database	The database to modify
filereader	The database file

4.6.2.4 parseDatabaseLine()

```
void parseDatabaseLine (
             const char key[],
             struct database * database,
             FILE * filereader )
```

Parse the line containing key and return into database.

Parameters

key	The relevant line to parse
database	The database
filereader	The database file

4.6.2.5 parseEduDesc()

```
void parseEduDesc (
             int amount_of_educations,
             struct education * educations,
             char current_line[] )
```

Parses the description for each education.

Parses the "read further" link for each education.

Parameters

educations	An array of educations
amount_of_educations	The amount of educations
current_line	The line to parse education names

4.6.2.6 parseEduNames()

```
void parseEduNames (
             int amount_of_educations,
             struct education * educations,
             char current_line[ ] )
```

Parses the name for each education.

Parameters

educations	An array of educations
amount_of_educations	The amount of educations
current_line	The line to parse education names

4.6.2.7 parseEduRegion()

```
void parseEduRegion (
            int amount_of_educations,
            struct education * educations,
            char current_line[ ] )
```

Parses the region for each education.

Parameters

educations	An array of educations
amount_of_educations	The amount of educations
current_line	The line to parse regions from

4.6.2.8 parseEduString()

```
char * parseEduString (
            char * current_line,
            int amount_of_educations,
            int offset )
```

Scans the current line + i until TABS or newline.

Saves the scanned string and returns a pointer to it.

Parameters

current_line	The line to scan
amount_of_educations	The amount of educations in database
offset	The offset to decide how many chars to skip in current_line

4.6.2.9 parseInterestNames()

```
void parseInterestNames (
             struct database * database,
             FILE * filereader )
```

Parse the names of each interest and return to the database.

Parameters

database	The database
filereader	The database file

4.6.2.10 parseInterestValues()

```
void parseInterestValues (
             int amount_of_interests,
             int amount_of_educations,
             struct education * educations,
             FILE * filereader )
```

Parse the values for each interest in all educations and return into educations.

Parameters

amount_of_interests	The amount of interests
amount_of_educations	The amount of educations
educations	The array of educations
filereader	The database file

4.6.2.11 parseNumOfEdu()

```
int parseNumOfEdu (
             FILE * filereader )
```

Returns the number of educations from database file.

Parameters

filereader	The file to read from

4.6.2.12 parseNumOfInterests()

```
int parseNumOfInterests (
            FILE * filereader )
```

Parse/count the number of intersts in the database file and return as int.

Parameters

filereader	The database file

4.6.2.13 parseReqGrade()

```
void parseReqGrade (
            int amount_of_educations,
            struct education * educations,
            char current_line[ ] )
```

Parses the required average grade for each education.

Parameters

educations	An array of educations
amount_of_educations	The amount of educations
current_line	The line to parse the required average grade from

4.6.2.14 parseSubReq()

```
void parseSubReq (
            int amount_of_educations,
            struct education * educations,
            char current_line[ ] )
```

Parses the subject requirements for each education.

Parameters

educations	An array of educations
amount_of_educations	The amount of educations
current_line	The line to parse subject requirements from

4.6.2.15 readReqString()

```
void readReqString (
            struct qualification * qualification,
            char * string,
            int education_location )
```

Read a requiremnt from a string.

Parameters

qualification	The qualification structure, where the read input is stored.
string	The string in which the requirements exists.
education_location	Which colomn is the educations requirements in.

4.6.2.16 strToReg()

```
int strToReg (
            char * region_string )
```

Converts a string into an enum region and return an enum region.

Parameters

region_string	The string to convert

4.7 profile.h File Reference

Contains elements relating to user profiles.

```
#include "vector.h"
#include "subjects.h"
#include "region.h"
#include "education.h"
#include "constants.h"
```

Include dependency graph for profile.h: This graph shows which files directly or indirectly include this file:

Classes

- struct profile

  Describes a user.

Functions

- struct profile createProfile (int number_of_interests)

  Allocates memory for each of the fields in the profile struct.
- void freeProfile (struct profile p)

  Frees the allocated memory for the given profile.
- void printProfile (struct profile p)

  Prints information stored in the given profil.

4.7.1 Detailed Description

Contains elements relating to user profiles.

Contains the profile struct and the functions for creating, printing and deallocating user profiles.

4.7.2 Function Documentation

4.7.2.1 createProfile()

```
struct profile createProfile (
             int number_of_interests )
```

Allocates memory for each of the fields in the profile struct.

Parameters

number_of_interests	The number of interests allocated

4.7.2.2 freeProfile()

```
void freeProfile (
             struct profile p )
```

Frees the allocated memory for the given profile.

Parameters

p	The profile struct which is freed

4.7.2.3 printProfile()

```
void printProfile (
             struct profile p )
```

Prints information stored in the given profil.

Parameters

p	The profile struct which is printed

4.8 region.h File Reference

Contains geographical elements.

This graph shows which files directly or indirectly include this file:

Classes

- struct location

Enumerations

- enum region {
  NORTH_JUTLAND = 0, **CENTRAL_JUTLAND**, **SOUTHERN_DENMARK**, **ZEALAND**,
  CAPITAL_REGION }

  Describes a region.

4.8.1 Detailed Description

Contains geographical elements.

This file contains the enums for different regions and the struct that symbolises a location.

4.8.2 Enumeration Type Documentation

4.8.2.1 region

```
enum region
```

Describes a region.

This enum descripes a region AKA it descripes a location in denmark.

4.9 subjects.h File Reference

Contains code regarding subjects and qualifcations for educations.

This graph shows which files directly or indirectly include this file:

Classes

- struct subject
- struct qualification

Enumerations

- enum {
  MATHEMATICS, **CHEMISTRY**, **BIOLOGY**, **PHYSICS**,
  ENGLISH, **BIOTECHNOLOGY**, **GEOSCIENCE**, **HISTORY**,
  IDEA_HISTORY, **INFORMATICS**, **INTERNATIONAL_ECONOMICS**, **COMMUNICATION_AND_IT**,
  RELIGION, **SOCIALSTUDIES**, **BUSINESS_ECONOMICS**, **CONTEMPORARY_HISTORY**,
  FRENCH, **SPANISH**, **GERMAN**, **CHINESE**,
  ARABIC, **GREEK**, **ITALIAN**, **JAPANESE**,
  LATIN, **PORTUGESE**, **RUSSIAN**, **NONE**,
  DANISH }
- enum **level** { **Z**, **C**, **B**, **A** }

Functions

- struct qualification **createQualifications** (int number_of_ualifications)
- void **freeQualifications** (struct qualification ∗)
- enum stringToClass (char ∗)

  Returns the enum class associated with the given string.
- enum level charToLevel (char ch)

  Returns the enum level associated with the given char.
- char levelToChar (enum level l)

  Returns the character associated with the given enum level.

4.9.1 Detailed Description

Contains code regarding subjects and qualifcations for educations.

Contains the enums for different classes and their levels. Also includes the subject and qualification structs and some related functions

4.9.2 Function Documentation

4.9.2.1 charToLevel()

```
enum level charToLevel (
             char ch )
```

Returns the enum level associated with the given char.

Parameters

ch	The character which is converted into an enum level

4.9.2.2 levelToChar()

```
enum char levelToChar (
            enum level l )
```

Returns the character associated with the given enum level.

Parameters

l	The enum level which is converted into a character

4.9.2.3 stringToClass()

```
enum class stringToClass (
            char * string )  [strong]
```

Returns the enum class associated with the given string.

Parameters

string	The string which is converted into an enum class

4.10 vector.h File Reference

Contains elements relating to vectors.

This graph shows which files directly or indirectly include this file:

Classes

- struct vector

Functions

- struct vector createVector (int size)

  creates a vector on the heap and outputs it
- struct vector copyVector (struct vector v)

  Copies the the inputted vector into vector copy and returns this.
- struct vector addVector (struct vector v1, struct vector v2)

  Adds two vectors together and outputs the sum as a vector.
- struct vector subtractVector (struct vector v1, struct vector v2)

  Subtracts the second vector from the first vector and returns the result as a vector.
- struct vector scaleVector (struct vector v, double scale)

  Multiplies the given vector's array values by the value inputted as scale, then outputs the result as a vector.

- struct vector normalizeVector (struct vector v)

    Normalises a vector via scaling it by one over it's length, then returns the normalized vector.
- double lengthOfVector (struct vector v)

    Calculates and returns the length of the given vector.
- double dotProduct (struct vector v1, struct vector v2)

    Calculates and returns the dot product of two vectors.
- void printVector (struct vector v)

    Prints a vector.
- void freeVector (struct vector v)

    frees the dynamically allocated array on the heap
- void freeVectorM (int num,...)

    Frees a variable number of struct vectors using free(Vector)

4.10.1 Detailed Description

Contains elements relating to vectors.

This file contains the vector struct and various functions used to create, manipulate or free vectors.

4.10.2 Function Documentation

4.10.2.1 addVector()

```
struct vector addVector (
              struct vector v1,
              struct vector v2 )
```

Adds two vectors together and outputs the sum as a vector.

Parameters

v1	The first vector struct: v1.array[] is a vector, v1.size number of elements in the vector
v2	The second vector struct: v2.array[] is a vector

4.10.2.2 copyVector()

```
struct vector copyVector (
              struct vector v )
```

Copies the the inputted vector into vector copy and returns this.

Parameters

v	The input vector that is copied

4.10.2.3 createVector()

```
struct vector createVector (
            int size )
```

creates a vector on the heap and outputs it

Parameters

size	The number of elements in the vector

4.10.2.4 dotProduct()

```
double dotProduct (
            struct vector v1,
            struct vector v2 )
```

Calculates and returns the dot product of two vectors.

Parameters

v1	The first vector to be used for dot product calculation
v2	The second vector to be used for dot product calculation

4.10.2.5 freeVector()

```
void freeVector (
            struct vector v )
```

frees the dynamically allocated array on the heap

Parameters

v	The vector struct containing the array on the heap

4.10.2.6 freeVectorM()

```
void freeVectorM (
             int num,
              ...  )
```

Frees a variable number of struct vectors using free(Vector)

Parameters

num	The number of arguments (vectors) that should be freed

4.10.2.7 lengthOfVector()

```
double lengthOfVector (
             struct vector v )
```

Calculates and returns the length of the given vector.

Parameters

v	The vector whose length is found

4.10.2.8 normalizeVector()

```
struct vector normalizeVector (
             struct vector v )
```

Normalises a vector via scaling it by one over it's length, then returns the normalized vector.

Parameters

v	The vector which is to be normalized

4.10.2.9 printVector()

```
void printVector (
             struct vector v )
```

Prints a vector.

Parameters

v	The vector that is printed

4.10.2.10 scaleVector()

```
struct vector scaleVector (
            struct vector v,
            double scale )
```

Multiplies the given vector's array values by the value inputted as scale, then outputs the result as a vector.

Parameters

v	The vector that should be up- or downscaled
scale	The value that the vector should be scaled by

4.10.2.11 subtractVector()

```
struct vector subtractVector (
            struct vector v1,
            struct vector v2 )
```

Subtracts the second vector from the first vector and returns the result as a vector.

Parameters

v1	The vector that should be subtracted from
v2	The vector that is used for subtraction

Index

addVector
    vector.h, 33

charToLevel
    subjects.h, 31
classNameStr
    commands.h, 11
commands.c, 9
commands.h, 9
    classNameStr, 11
    convertScale, 11
    deleteCmd, 11
    evalCmd, 12
    findCmd, 12
    getEmptyIndex, 13
    getIndex, 13
    isQualified, 13
    levelAsValue, 13
    listCmd, 14
    listIsFull, 14
    recommendCmd, 14
    saveCmd, 15
    saveProfile, 15
    searchCmd, 15
    setImportantSubjects, 16
    setOtherSubjects, 16
    setProfileInterests, 16
    setProfileLocation, 17
    setProfileQualifications, 17
    setSubjects, 17
    surveyCmd, 17
    validScaleValue, 18
constants.h, 18
convertScale
    commands.h, 11
copyVector
    vector.h, 33
createArrayOfEducations
    education.h, 20
createArrayOfStrings
    parser.h, 22
createDefaultEducation
    education.h, 20
createProfile
    profile.h, 29
createVector
    vector.h, 34

Database, 5
database, 5
database.h, 19
deleteCmd
    commands.h, 11
dotProduct
    vector.h, 34

education, 6
education.h, 19
    createArrayOfEducations, 20
    createDefaultEducation, 20
evalCmd
    commands.h, 12

findCmd
    commands.h, 12
findDatabaseLine
    parser.h, 22
freeProfile
    profile.h, 29
freeVector
    vector.h, 34
freeVectorM
    vector.h, 34

getEmptyIndex
    commands.h, 13
getIndex
    commands.h, 13

isQualified
    commands.h, 13

lengthOfVector
    vector.h, 35
levelAsValue
    commands.h, 13
levelToChar
    subjects.h, 31
listCmd
    commands.h, 14
listIsFull
    commands.h, 14
location, 6

normalizeVector
    vector.h, 35

parseDatabase
    parser.h, 22
parseDatabaseLine
    parser.h, 24

parseEduDesc
    parser.h, 24
parseEduNames
    parser.h, 24
parseEduRegion
    parser.h, 25
parseEduString
    parser.h, 25
parseInterestNames
    parser.h, 25
parseInterestValues
    parser.h, 26
parseNumOfEdu
    parser.h, 26
parseNumOfInterests
    parser.h, 26
parseReqGrade
    parser.h, 27
parseSubReq
    parser.h, 27
parser.h, 21
    createArrayOfStrings, 22
    findDatabaseLine, 22
    parseDatabase, 22
    parseDatabaseLine, 24
    parseEduDesc, 24
    parseEduNames, 24
    parseEduRegion, 25
    parseEduString, 25
    parseInterestNames, 25
    parseInterestValues, 26
    parseNumOfEdu, 26
    parseNumOfInterests, 26
    parseReqGrade, 27
    parseSubReq, 27
    readReqString, 27
    strToReg, 28
printProfile
    profile.h, 29
printVector
    vector.h, 35
profile, 6
profile.h, 28
    createProfile, 29
    freeProfile, 29
    printProfile, 29

qualification, 7

readReqString
    parser.h, 27
recommendCmd
    commands.h, 14
region
    region.h, 30
region.h, 30
    region, 30

saveCmd
    commands.h, 15
saveProfile
    commands.h, 15
scaleVector
    vector.h, 36
searchCmd
    commands.h, 15
setImportantSubjects
    commands.h, 16
setOtherSubjects
    commands.h, 16
setProfileInterests
    commands.h, 16
setProfileLocation
    commands.h, 17
setProfileQualifications
    commands.h, 17
setSubjects
    commands.h, 17
strToReg
    parser.h, 28
stringToClass
    subjects.h, 32
subject, 7
subjects.h, 30
    charToLevel, 31
    levelToChar, 31
    stringToClass, 32
subtractVector
    vector.h, 36
surveyCmd
    commands.h, 17

validScaleValue
    commands.h, 18
vector, 7
vector.h, 32
    addVector, 33
    copyVector, 33
    createVector, 34
    dotProduct, 34
    freeVector, 34
    freeVectorM, 34
    lengthOfVector, 35
    normalizeVector, 35
    printVector, 35
    scaleVector, 36
    subtractVector, 36