Automatiser dit studievalg

Generated by Doxygen 1.8.13

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 database Struct Reference

Collaboration diagram for database:

Public Attributes

- int **amount_of_educations**
- struct education * educations
- int amount_of_interests
- char ** interest_string

3.1.1 Member Data Documentation

3.1.1.1 amount_of_interests

```
int database::amount_of_interests
```

an array of educations delimited by amount_of_educations

3.1.1.2 educations

```
struct education* database::educations
```

the amount of educations in the database

3.1.1.3 interest_string

`char** database::interest_string`

the amount of interests in the database

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

- database.h

3.2 education Struct Reference

Describes an education and all it requirements.

`#include </home/xomnez/aau/1.semester/p1_ads/P1/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**

3.2.1 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

3.3 location Struct Reference

Public Attributes

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

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

- region.h

3.4 profile Struct Reference

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]

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

- profile.h

3.5 qualification Struct Reference

Collaboration diagram for qualification:

Public Attributes

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

3.5.1 Member Data Documentation

3.5.1.1 subjects

```
struct subject* qualification::subjects
```

the amount of subjects in qualifications

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

- subjects.h

3.6 subject Struct Reference

Public Attributes

- enum level level

3.6.1 Member Data Documentation

3.6.1.1 level

```
enum level subject::level
```

the name of the subject

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

- subjects.h

3.7 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.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:

4.2 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 **DATABASE_PATH** "./bin/data/database.txt"

4.2.1 Detailed Description

Contains symbolic constants used throughout the program.

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

4.3 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 *)
  Free a databases memories.
- struct database * createDatabase (char *)
  Create a Database object.
- struct education * findEducation (char *, struct database *)
  Finds an education in a database and returns a pointer to the education.
- void searchDatabaseForEducation (char *, struct database *, struct education **, int *)
  Finds all educations with the matching search_word.

4.3.1 Detailed Description

Contains elements relating to the database.

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

4.3.2 Function Documentation

4.3.2.1 createDatabase()

```
struct database* createDatabase (
            char * database_file )
```

Create a Database object.

Parameters

database_file	is the databasefile, which will be read into a database object

Returns

struct database*

4.3.2.2 findEducation()

```
struct education* findEducation (
            char * key,
            struct database * database )
```

Finds an education in a database and returns a pointer to the education.

Parameters

database	is the database, which will be searched

Returns

struct education*

4.3.2.3 freeDatabase()

```
void freeDatabase (
            struct database * database )
```

Free a databases memories.

Parameters

database	

4.3.2.4 searchDatabaseForEducation()

```
void searchDatabaseForEducation (
            char * search_word,
            struct database * database,
```

```
            struct education ** array,
            int * size_of_array )
```

Finds all educations with the matching search_word.

Parameters

search_word	The name of the education you are searching for

Returns

struct educationArray*

4.4 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)
- void freeEducation (struct education *)

4.4.1 Detailed Description

Contains elements relating to educations.

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

4.4.2 Function Documentation

4.4.2.1 freeEducation()

```
void freeEducation (
            struct education * education )
```

Parameters

education	

4.5 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)
- void **parseDatabaseLine** (const char key[ ], struct database *database, FILE *filereader)
- void **findDatabaseLine** (const char key[ ], FILE *filereader, char *current_line)
- int parseNumOfEdu (FILE *filereader)

  Returns the number of educations from database file.
- int **parseNumOfInterests** (FILE *filereader)
- void **parseEduNames** (struct database *database, char current_line[ ])
- void **parseEduDesc** (struct database *database, char current_line[ ])
- void **parseEduLink** (struct database *database, char current_line[ ])
- void **parseEduRegion** (struct database *database, char current_line[ ])
- void **parseSubReq** (struct education *education, int number_of_educations, FILE *filereader, char current_line[ ])
- void **parseReqGrade** (struct database *database, char current_line[ ])
- void **parseInterestValues** (struct database *database, FILE *filereader)
- void **parseInterestNames** (struct database *database, FILE *filereader)
- 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.
- int **sseek** (char *, char)
- void **readReqString** (struct qualification *, char *, int)
- enum region strToReg (char *region_string)

  Converts a string into an enum region.

4.5.1 Detailed Description

Contains elements relating to parsing the database.

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

4.5.2 Function Documentation

4.5.2.1 parseDatabase()

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

Parameters

database	
filereader	

4.5.2.2 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

4.5.2.3 parseNumOfEdu()

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

Returns the number of educations from database file.

Parameters

filereader	The file to read from

4.5.2.4 strToReg()

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

Converts a string into an enum region.

Parameters

region_string	The string to convert

4.6 profile.h File Reference

Contains elements relating to profile.

```
#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

Functions

- void **printProfile** (struct profile p)
- void **freeProfile** (struct profile p)
- void **freeQualifications** (struct qualification q)
- struct qualification **createQualifications** (int number_of_qualifications)
- struct profile **createProfile** (int number_of_interests)

4.6.1 Detailed Description

Contains elements relating to profile.

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

4.7 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.

4.7.1 Detailed Description

Contains geographical elements.

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

4.7.2 Enumeration Type Documentation

4.7.2.1 region

`enum region`

Describes a region.

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

4.8 serialize.h File Reference

Save and load profile data.

`#include "profile.h"`
Include dependency graph for serialize.h:

Macros

- #define **SAVE_FILE** "data/save.data"

Functions

- int **saveProfile** (struct profile *)
- struct profile * **loadProfile** ()

4.8.1 Detailed Description

Save and load profile data.

Author

Version

0.1

Date

2019-11-27

Copyright

Copyright (c) 2019

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

- void freeSubject (struct subject ∗)
  free a subject and its members
- void freeQualification (struct qualification ∗)
  free a qualification and its members
- enum **stringToClass** (char ∗)
- enum level **charToLevel** (char ch)
- char **levelToChar** (enum level l)

4.9.1 Detailed Description

Contains code regarding subjects and qualifcations for educations.

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

4.9.2 Function Documentation

4.9.2.1 freeQualification()

```
void freeQualification (
            struct qualification * qualification )
```

free a qualification and its members

Parameters

qualification	the qualification to be freed

4.9.2.2 freeSubject()

```
void freeSubject (
            struct subject * subject )
```

free a subject and its members

Parameters

subject	the subject to be freed

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