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 CuString Struct Reference

Public Attributes

- int **length**
- int **size**
- char * **buffer**

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

- CuTest.h

3.2 CuSuite Struct Reference

Collaboration diagram for CuSuite:

Public Attributes

- int **count**
- CuTest * **list** [MAX_TEST_CASES]
- int **failCount**

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

- CuTest.h

3.3 CuTest Struct Reference

Collaboration diagram for CuTest:

Public Attributes

- char * **name**
- TestFunction **function**
- int **failed**
- int **ran**
- const char * **message**
- jmp_buf * **jumpBuf**

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

- CuTest.h

3.4 database Struct Reference

Collaboration diagram for database:

Public Attributes

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

3.4.1 Member Data Documentation

3.4.1.1 amount_of_interests

```
int database::amount_of_interests
```

an array of educations delimited by amount_of_educations

3.4.1.2 educations

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

the amount of educations in the database

3.4.1.3 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

3.5 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

3.5.1 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.

3.5.2 Member Data Documentation

3.5.2.1 description

`char* education::description`

The name of an education

3.5.2.2 interests

`struct vector education::interests`

The minimum grade required for entry

3.5.2.3 link

`char* education::link`

The description of an education

3.5.2.4 region

```
enum region education::region
```

A link to the educations website

3.5.2.5 required_grade

```
double education::required_grade
```

The region where the education is found

3.5.2.6 required_qualifications

```
struct qualification education::required_qualifications
```

The amount each interest is associated with this education

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

- education.h

3.6 location Struct Reference

Public Attributes

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

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

- region.h

3.7 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

3.8 qualification Struct Reference

Collaboration diagram for qualification:

Public Attributes

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

3.8.1 Member Data Documentation

3.8.1.1 subjects

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

the amount of subjects in qualifications

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

- subjects.h

3.9 subject Struct Reference

Public Attributes

- enum level level

3.9.1 Member Data Documentation

3.9.1.1 level

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

the name of the subject

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

- subjects.h

3.10 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 serialize.h File Reference

Save and load profile data.

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

4.2 vector.h File Reference

Library which contains a variety of functions 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)

4.2.1 Detailed Description

Library which contains a variety of functions relating to vectors.

4.2.2 Function Documentation

4.2.2.1 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

4.2.2.2 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

4.2.2.3 createVector()

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

creates a vector on the heap and outputs it

Parameters

size	The number of elements in the vector

4.2.2.4 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

4.2.2.5 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

4.2.2.6 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

4.2.2.7 lengthOfVector()

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

Calculates and returns the length of the given vector.

Parameters

v	The vector whose length is found

4.2.2.8 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

4.2.2.9 printVector()

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

Prints a vector.

Parameters

v	The vector that is printed

4.2.2.10 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

4.2.2.11 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