AAU Scheduler

1.0

Generated by Doxygen 1.8.10

Wed Dec 9 2015 11:24:17

Contents

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

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

Chapter 3

Data Structure Documentation

3.1 Course Struct Reference

Data Fields

- char **name** [64]
- int **totLectures**
- int **numTeachers**
- **Teacher** $**$ **teachers**

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

- **structs.h**

3.2 Flags Struct Reference

Data Fields

- int **doubleBookingRoom**
- int **doubleBookingLecture**
- int **lectureOverflow**
- int **semesterOverflow**

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

- **structs.h**

3.3 Generations Struct Reference

Data Fields

- int **numLectures**
- **Lecture** $*$ **schedules** [GENERATION_SIZE]

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

- **structs.h**

3.4 Lecture Struct Reference

Data Fields

- int **day**
- int **period**
- **Room** ∗ **assignedRoom**
- **Course** ∗ **assignedCourse**
- int **fitness**
- **Flags flags**

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

- **structs.h**

3.5 OffTime Struct Reference

Data Fields

- int **day**
- int **periods** [MAX_PERIODS]

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

- **structs.h**

3.6 Room Struct Reference

Data Fields

- char **name** [32]
- int **seats**

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

- **structs.h**

3.7 SemesterData Struct Reference

Data Fields

- int **numWeeks**
- int **numRooms**
- **Room** ∗ **rooms**
- int **numTeachers**
- **Teacher** ∗ **teachers**
- int **numCourses**
- **Course** ∗ **courses**
- int **numSpecializations**
- **Specialization** ∗ **specializations**

- Generation ∗ **generation**

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

- **structs.h**

3.8 Specialization Struct Reference

Data Fields

- char **name** [32]
- int **numStudents**
- int **numCourses**
- **Course** ∗∗ **courses**

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

- **structs.h**

3.9 Teacher Struct Reference

Data Fields

- char **name** [32]
- int **numOffTimes**
- **OffTime** ∗ **offTimes**

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

- **structs.h**

Chapter 4

File Documentation

4.1 config_reader.c File Reference

This script is responsible for reading the data file.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "structs.h"
#include "config_reader.h"
#include "defs.h"
```

Macros

- #define **BUFFER_SIZE** 512

Functions

- int **read_config** (char ∗fileName, **SemesterData** ∗sd)

    Initial function for the config reader.
- void **handle_line** (char ∗line, **SemesterData** ∗sd)

    This function handles the lines from the main config reader function.
- int **read_int** (char ∗line, int ∗position, int ∗out)

    Reads an int from a string and adds the amount of digits to position.
- int **read_multiple_words** (char ∗line, int ∗position, char ∗out)

    Reads an entire string between two apostrophes.
- void **add_teacher** (**SemesterData** ∗sd, char ∗name, int numOffTimes, **OffTime** ∗offTimes)

    Adds a teacher to the teachers array.
- void **add_room** (**SemesterData** ∗sd, char ∗name, int seats)

    Adds a room to the rooms array.
- void **add_course** (**SemesterData** ∗sd, char ∗name, int totLectures, int numTeachers, **Teacher** ∗∗teachers)

    Adds a course to the courses array.
- void **add_specialization** (**SemesterData** ∗sd, char ∗name, int numStudents, int numCourses, **Course** ∗∗courses)

    Adds a specialization to the specializations array.

4.1.1 Detailed Description

This script is responsible for reading the data file.

4.1.2 Function Documentation

4.1.2.1 void add_course (SemesterData ∗ *sd*, char ∗ *name*, int *totLectures*, int *numTeachers*, Teacher ∗∗ *teachers*)

Adds a course to the courses array.

Parameters

`in`	*sd*	The courses array is part of the **SemesterData** (p. 6) struct
`in`	*name*	The name of the course
`in`	*totLectures*	The total amount of lectures in the course
`in`	*numTeachers*	The amount of teachers assigned to the course
`in`	*teachers*	The array of teachers assigned

Allocates the memory needed and updates relevant variables and values

4.1.2.2 void add_room (SemesterData ∗ *sd*, char ∗ *name*, int *seats*)

Adds a room to the rooms array.

Parameters

`in`	*sd*	The rooms array is part of the **SemesterData** (p. 6) struct
`in`	*name*	The name of the room
`in`	*seats*	The amount of seats available in the room

Allocates the memory needed and updates relevant variables and values

4.1.2.3 void add_specialization (SemesterData ∗ *sd*, char ∗ *name*, int *numStudents*, int *numCourses*, Course ∗∗ *courses*)

Adds a specialization to the specializations array.

Parameters

`in`	*sd*	The specialization array is part of the **SemesterData** (p. 6) struct
`in`	*name*	The name of the specialization
`in`	*numStudents*	The total amount of students in the specialization
`in`	*numCourses*	The amount of courses assigned to the specialization
`in`	*courses*	The array of courses assigned

Allocates the memory needed and updates relevant variables and values

4.1.2.4 void add_teacher (SemesterData ∗ *sd*, char ∗ *name*, int *numOffTimes*, OffTime ∗ *offTimes*)

Adds a teacher to the teachers array.

Parameters

`in`	*sd*	The teachers array is part of the **SemesterData** (p. 6) struct
`in`	*name*	The name of the teacher
`in`	*numOffTimes*	The amount of off times

in	*offTimes*	An array of offTimes

Allocates the memory needed and updates relevant variables and values

4.1.2.5 void handle_line (char ∗ *line*, SemesterData ∗ *sd*)

This function handles the lines from the main config reader function.

Parameters

in	*line*	This line is given by the config_reader function
in	*sd*	**SemesterData** (p. 6) is a link to the structs the function needs

This function goes through the line and checks it for commands and parameters. Essentially it works like a parser

4.1.2.6 int read_config (char ∗ *fileName*, SemesterData ∗ *sd*)

Initial function for the config reader.

Parameters

in	*fileName*	The name of the file to read from
in	*sd*	**SemesterData** (p. 6) is a link to our structs that we will need for this function

Returns

Returns 1 or 0 depending weather the function succeded or failed

The function reads the file line by line and formats them to the format we need for further processing, then sends it to handle_line

4.1.2.7 int read_int (char ∗ *line*, int ∗ *position*, int ∗ *out*)

Reads an int from a string and adds the amount of digits to position.

Parameters

in	*line*	The string to read
in	*position*	Current position in the string
out	*out*	A pointer to an int where the final number will be stored

Returns

Returns weather the function has failed or succeded

The function goes through the string (line) until there are no more characters. It then converts the content of the string to int and outputs it to the out variable

4.1.2.8 int read_multiple_words (char ∗ *line*, int ∗ *position*, char ∗ *out*)

Reads an entire string between two apostrophes.

Parameters

in	*line*	The string to read from

in	*position*	The current position in the string
out	*out*	The output string

Returns

Returns weather the function succeded or not

This function reads from the line string and outputs everything between two apostrophes to the output string

4.2 config_reader.h File Reference

This file contains prototypes required by the **config_reader.c** (p. 9) script.

Functions

- int **read_config** (char ∗fileName, **SemesterData** ∗data)

  Initial function for the config reader.
- void **handle_line** (char ∗line, **SemesterData** ∗data)

  This function handles the lines from the main config reader function.
- int **read_int** (char ∗line, int ∗position, int ∗out)

  Reads an int from a string and adds the amount of digits to position.
- int **read_multiple_words** (char ∗line, int ∗position, char ∗out)

  Reads an entire string between two apostrophes.
- void **add_teacher** (**SemesterData** ∗sd, char ∗name, int numOffTimes, **OffTime** ∗offTimes)

  Adds a teacher to the teachers array.
- void **add_room** (**SemesterData** ∗sd, char ∗name, int seats)

  Adds a room to the rooms array.
- void **add_course** (**SemesterData** ∗sd, char ∗name, int totLectures, int numTeachers, **Teacher** ∗∗teachers)

  Adds a course to the courses array.
- void **add_specialization** (**SemesterData** ∗sd, char ∗name, int numStudents, int numCourses, **Course** ∗∗courses)

  Adds a specialization to the specializations array.

4.2.1 Detailed Description

This file contains prototypes required by the **config_reader.c** (p. 9) script.

4.2.2 Function Documentation

4.2.2.1 void add_course (SemesterData ∗ *sd,* char ∗ *name,* int *totLectures,* int *numTeachers,* Teacher ∗∗ *teachers*)

Adds a course to the courses array.

Parameters

in	*sd*	The courses array is part of the **SemesterData** (p. 6) struct
in	*name*	The name of the course

in	*totLectures*	The total amount of lectures in the course
in	*numTeachers*	The amount of teachers assigned to the course
in	*teachers*	The array of teachers assigned

Allocates the memory needed and updates relevant variables and values

4.2.2.2 void add_room (SemesterData ∗ *sd*, char ∗ *name*, int *seats*)

Adds a room to the rooms array.

Parameters

in	*sd*	The rooms array is part of the **SemesterData** (p. 6) struct
in	*name*	The name of the room
in	*seats*	The amount of seats available in the room

Allocates the memory needed and updates relevant variables and values

4.2.2.3 void add_specialization (SemesterData ∗ *sd*, char ∗ *name*, int *numStudents*, int *numCourses*, Course ∗∗ *courses*)

Adds a specialization to the specializations array.

Parameters

in	*sd*	The specialization array is part of the **SemesterData** (p. 6) struct
in	*name*	The name of the specialization
in	*numStudents*	The total amount of students in the specialization
in	*numCourses*	The amount of courses assigned to the specialization
in	*courses*	The array of courses assigned

Allocates the memory needed and updates relevant variables and values

4.2.2.4 void add_teacher (SemesterData ∗ *sd*, char ∗ *name*, int *numOffTimes*, OffTime ∗ *offTimes*)

Adds a teacher to the teachers array.

Parameters

in	*sd*	The teachers array is part of the **SemesterData** (p. 6) struct
in	*name*	The name of the teacher
in	*numOffTimes*	The amount of off times
in	*offTimes*	An array of offTimes

Allocates the memory needed and updates relevant variables and values

4.2.2.5 void handle_line (char ∗ *line*, SemesterData ∗ *sd*)

This function handles the lines from the main config reader function.

Parameters

in	*line*	This line is given by the config_reader function
in	*sd*	**SemesterData** (p. 6) is a link to the structs the function needs

This function goes through the line and checks it for commands and parameters. Essentially it works like a parser

4.2.2.6 int read_config (char ∗ *fileName*, SemesterData ∗ *sd*)

Initial function for the config reader.

Parameters

in	*fileName*	The name of the file to read from
in	*sd*	**SemesterData** (p. 6) is a link to our structs that we will need for this function

Returns

Returns 1 or 0 depending weather the function succeded or failed

The function reads the file line by line and formats them to the format we need for further processing, then sends it to handle_line

4.2.2.7 int read_int (char ∗ *line,* int ∗ *position,* int ∗ *out*)

Reads an int from a string and adds the amount of digits to position.

Parameters

in	*line*	The string to read
in	*position*	Current position in the string
out	*out*	A pointer to an int where the final number will be stored

Returns

Returns weather the function has failed or succeded

The function goes through the string (line) until there are no more characters. It then converts the content of the string to int and outputs it to the out variable

4.2.2.8 int read_multiple_words (char ∗ *line,* int ∗ *position,* char ∗ *out*)

Reads an entire string between two apostrophes.

Parameters

in	*line*	The string to read from
in	*position*	The current position in the string
out	*out*	The output string

Returns

Returns weather the function succeded or not

This function reads from the line string and outputs everything between two apostrophes to the output string

4.3 data_test.c File Reference

This script contains the functions responsible for performing tests on the generations.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "structs.h"
#include "data_utility.h"
#include "defs.h"
#include "data_test.h"
```

Functions

- int **test_lecture_capacity** (**SemesterData** ∗sd, **Lecture** ∗lect)
  Test how well the lecture fits into the assigned room.
- int **test_teacher_availability** (**SemesterData** ∗sd, **Lecture** ∗lect)
  Test whether the teacher has an offtime on a given date.
- int **test_doublebooking** (**SemesterData** ∗sd, **Lecture** ∗lect)
  Test for doublebooking.
- int **test_weekly_distribution** (**SemesterData** ∗sd, **Lecture** ∗lect)
  Test distribution on a weekly basis to ensure an even workload.
- int **test_semester_distribution** (**SemesterData** ∗sd, **Lecture** ∗lect)
  Tests the semester distribution.
- int **test_semester_distribution_inner** (**SemesterData** ∗sd, **Lecture** ∗lect, **Specialization** ∗sp)
  Test how the lecture fits into the semester distribution.

4.3.1 Detailed Description

This script contains the functions responsible for performing tests on the generations.

4.3.2 Function Documentation

4.3.2.1 int test_doublebooking (SemesterData ∗ *sd,* Lecture ∗ *lect*)

Test for doublebooking.

Parameters

in	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function
in	*lect*	Pointer to lecture to test

Returns

Returns the severity of the test. Aka. Fitness

Performs tests for both room and lecture doublebooking

4.3.2.2 int test_lecture_capacity (SemesterData ∗ *sd,* Lecture ∗ *lect*)

Test how well the lecture fits into the assigned room.

Parameters

in	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function
in	*lect*	Pointer to lecture to test

Returns

Returns the severity of the test. Aka. Fitness

This function checks the capacity of the room and the amount of students on the lecture and determins the penalty in fitness by comparing the two.

4.3.2.3 int test_semester_distribution (SemesterData ∗ *sd,* Lecture ∗ *lect*)

Tests the semester distribution.

Parameters

`in`	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function
`in`	*lect*	Pointer to lecture to test

Returns

Returns the severity of the test. Aka. Fitness

Makes a call to the inner test function for every specialization on the specified lecture

4.3.2.4 int test_semester_distribution_inner (**SemesterData** ∗ *sd,* **Lecture** ∗ *lect,* **Specialization** ∗ *sp*)

Test how the lecture fits into the semester distribution.

Parameters

`in`	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function
`in`	*lect*	Pointer to lecture to test
`in`	*sp*	??????????????????????????????????

Returns

Returns the severity of the test. Aka. Fitness

Details

4.3.2.5 int test_teacher_availability (**SemesterData** ∗ *sd,* **Lecture** ∗ *lect*)

Test whether the teacher has an offtime on a given date.

Parameters

`in`	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function
`in`	*lect*	Pointer to lecture to test

Returns

Returns the severity of the test. Aka. Fitness

Also test whether the teacher is already assigned to a lecture on the same date

4.3.2.6 int test_weekly_distribution (**SemesterData** ∗ *sd,* **Lecture** ∗ *lect*)

Test distribution on a weekly basis to ensure an even workload.

Parameters

`in`	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function

`in`	*lect*	Pointer to lecture to test

Returns

Returns the severity of the test. Aka. Fitness

Details

4.4 data_test.h File Reference

This file contains prototypes required by the **data_test.c** (p. 14) script.

Functions

- int **test_lecture_capacity** (**SemesterData** ∗sd, **Lecture** ∗lect)

  Test how well the lecture fits into the assigned room.
- int **test_overlap** (**SemesterData** ∗sd, **Lecture** ∗lect)
- int **test_teacher_availability** (**SemesterData** ∗sd, **Lecture** ∗lect)

  Test whether the teacher has an offtime on a given date.
- int **test_doublebooking** (**SemesterData** ∗sd, **Lecture** ∗lect)

  Test for doublebooking.
- int **test_weekly_distribution** (**SemesterData** ∗sd, **Lecture** ∗lect)

  Test distribution on a weekly basis to ensure an even workload.
- int **test_semester_distribution** (**SemesterData** ∗sd, **Lecture** ∗lect)

  Tests the semester distribution.
- int **test_semester_distribution_inner** (**SemesterData** ∗sd, **Lecture** ∗lect, **Specialization** ∗sp)

  Test how the lecture fits into the semester distribution.

4.4.1 Detailed Description

This file contains prototypes required by the **data_test.c** (p. 14) script.

4.4.2 Function Documentation

4.4.2.1 int test_doublebooking (SemesterData ∗ *sd*, Lecture ∗ *lect*)

Test for doublebooking.

Parameters

`in`	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function
`in`	*lect*	Pointer to lecture to test

Returns

Returns the severity of the test. Aka. Fitness

Performs tests for both room and lecture doublebooking

4.4.2.2 int test_lecture_capacity (SemesterData ∗ *sd*, Lecture ∗ *lect*)

Test how well the lecture fits into the assigned room.

Parameters

in	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function
in	*lect*	Pointer to lecture to test

Returns

Returns the severity of the test. Aka. Fitness

This function checks the capacity of the room and the amount of students on the lecture and determins the penalty in fitness by comparing the two.

4.4.2.3 int test_semester_distribution (SemesterData ∗ *sd*, Lecture ∗ *lect*)

Tests the semester distribution.

Parameters

in	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function
in	*lect*	Pointer to lecture to test

Returns

Returns the severity of the test. Aka. Fitness

Makes a call to the inner test function for every specialization on the specified lecture

4.4.2.4 int test_semester_distribution_inner (SemesterData ∗ *sd*, Lecture ∗ *lect*, Specialization ∗ *sp*)

Test how the lecture fits into the semester distribution.

Parameters

in	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function
in	*lect*	Pointer to lecture to test
in	*sp*	??????????????????????????????????

Returns

Returns the severity of the test. Aka. Fitness

Details

4.4.2.5 int test_teacher_availability (SemesterData ∗ *sd*, Lecture ∗ *lect*)

Test whether the teacher has an offtime on a given date.

Parameters

in	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function

`in`	*lect*	Pointer to lecture to test

Returns

Returns the severity of the test. Aka. Fitness

Also test whether the teacher is already assigned to a lecture on the same date

4.4.2.6 int test_weekly_distribution (**SemesterData** ∗ *sd,* **Lecture** ∗ *lect*)

Test distribution on a weekly basis to ensure an even workload.

Parameters

`in`	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function
`in`	*lect*	Pointer to lecture to test

Returns

Returns the severity of the test. Aka. Fitness

Details

4.5 data_utility.h File Reference

This file contains prototypes required by the data_utility.c script.

```
#include "structs.h"
```

Functions

- void **reset_lecture_flags** (**SemesterData** ∗sd)
- void **add_lecture** (**SemesterData** ∗sd, int lectIndex, int day, int period, int roomId, int courseId)
- int **teacher_has_offtime** (**SemesterData** ∗sd, **Teacher** ∗teacher, int dayId, int periodId)
- int **specialization_has_lecture** (**Specialization** ∗sp, **Lecture** ∗lect)
- int **get_students_on_course** (**SemesterData** ∗sd, **Course** ∗course)
- int **get_amount_of_lectures** (**SemesterData** ∗sd)
- int **get_specializations_for_course** (**SemesterData** ∗sd, **Course** ∗course, **Specialization** ∗∗∗specs)
- const char ∗ **get_name_of_period** (int periodId)
- const char ∗ **get_name_of_day** (int dayId)

4.5.1 Detailed Description

This file contains prototypes required by the data_utility.c script.

4.6 defs.h File Reference

This file contains the defines required by the program.

Macros

- #define **MAX_PERIODS** 2
- #define **DAYS_PER_WEEK** 5
- #define **GENERATION_SIZE** 100
- #define **ERROR_OUT_OF_MEMORY** 1
- #define **ERROR_ARRAY_BOUNDS_EXCEEDED** 2

4.6.1 Detailed Description

This file contains the defines required by the program.

4.7 html_output.c File Reference

The html output script is responsible for the html schedules that is being generated.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdarg.h>
#include "defs.h"
#include "structs.h"
#include "data_utility.h"
```

Macros

- #define **WEEK_WIDTH** 10.0f
- #define **TABLE_WIDTH** 100.0f

Functions

- void **print_file_header** (FILE ∗f, char ∗pageTitle)

  Prints the file header.
- void **print_footer** (FILE ∗f)

  Prints the file footer.
- void **begin_print_table** (FILE ∗f, int cellspacing)

  Initiates a table.
- void **end_print_table** (FILE ∗f)

  Ends a table.
- void **print_row_header** (FILE ∗f, double width, const char ∗str,...)

  Prints a header for a row.
- void **print_title** (FILE ∗f, const char ∗title)

  Prints a shedule title.
- void **begin_print_data** (FILE ∗f, const char ∗str)

  Brief.
- void **end_print_data** (FILE ∗f)

  Ends the data print.
- void **begin_print_row** (FILE ∗f, const char ∗backgroundColor)

  Prints the rows of lectures.
- void **end_print_row** (FILE ∗f)

Ends the row print.

- void **print_period** (**SemesterData** ∗sd, **Specialization** ∗sp, FILE ∗f, int periodId, int weekNumber)

    Prints a period to the schedule.

- void **print_schedule_to_file** (**SemesterData** ∗sd, **Specialization** ∗sp, const char ∗fileName)

    Prints a schedule for a specific specialization to a file.

4.7.1 Detailed Description

The html output script is responsible for the html schedules that is being generated.

4.7.2 Function Documentation

4.7.2.1 void begin_print_data (FILE ∗ *f*, const char ∗ *str*)

Brief.

Parameters

`in`	*f*	The file in which the schedule is being generated
`in`	*str*	The data to be printed

This function is printing the provided data from str into the file f

4.7.2.2 void begin_print_row (FILE ∗ *f*, const char ∗ *backgroundColor*)

Prints the rows of lectures.

Parameters

`in`	*f*	The file in which the schedule is being generated
`in`	*backgroundColor*	The color of the row

This function initiates rows with a given color

4.7.2.3 void begin_print_table (FILE ∗ *f*, int *cellspacing*)

Initiates a table.

Parameters

`in`	*f*	The file in which the schedule is being generated
`in`	*cellspacing*	The spacing between the cells in the table

This function is laying the foundation for a html table

4.7.2.4 void end_print_data (FILE ∗ *f*)

Ends the data print.

Parameters

`in`	*f*	The file in which the schedule is being generated

Adds the ending tag for the data

4.7.2.5 void end_print_row (FILE ∗ *f*)

Ends the row print.

Parameters

`in`	*f*	The file in which the schedule is being generated

This function adds the ending tag for the row

4.7.2.6 void end_print_table (FILE ∗ *f*)

Ends a table.

Parameters

`in`	*f*	The file in which the schedule is being generated

This function is adding the end table tag for a html table

4.7.2.7 void print_file_header (FILE ∗ *f*, char ∗ *pageTitle*)

Prints the file header.

Parameters

`in`	*f*	The file in which the schedule is being generated
`in`	*pageTitle*	The name of the page

This function is responsible for the header of the file

4.7.2.8 void print_footer (FILE ∗ *f*)

Prints the file footer.

Parameters

`in`	*f*	The file in which the schedule is being generated

This function is responsible for the footer of the file

4.7.2.9 void print_period (**SemesterData** ∗ *sd*, **Specialization** ∗ *sp*, FILE ∗ *f*, int *periodId*, int *weekNumber*)

Prints a period to the schedule.

Parameters

`in`	*sd*	**SemesterData** (p. 6) contains the required information about the data that should be printed
`in`	*sp*	**Specialization** (p. 7) contains information about the specialization the schedule is generated for
`in`	*f*	The file in which the schedule is being generated
`in`	*periodId*	?????????????
`in`	*weekNumber*	The number of the current week

This function adds a period to the schedule and formats it as needed

4.7.2.10 void print_row_header (FILE ∗ *f*, double *width*, const char ∗ *str*, ...)

Prints a header for a row.

Parameters

in	*f*	The file in which the schedule is being generated
in	*width*	The width of the row
in	*str*	The name of the row
in	*...*	??????????????

This function creates a row with the given width and name

4.7.2.11 void print_schedule_to_file (SemesterData ∗ *sd*, Specialization ∗ *sp*, const char ∗ *fileName*)

Prints a schedule for a specific specialization to a file.

Parameters

in	*sd*	**SemesterData** (p. 6) contains the required information about the data that should be printed
in	*sp*	**Specialization** (p. 7) contains information about the specialization the schedule is generated for
in	*fileName*	The name of the file in which the schedule should be generated

The final step of the schedule creation

4.7.2.12 void print_title (FILE ∗ *f*, const char ∗ *title*)

Prints a shedule title.

Parameters

in	*f*	The file in which the schedule is being generated
in	*title*	The title

This defines a title for the schedule. An example could be "Schedule for Robotics"

4.8 html_output.h File Reference

This file contains prototypes required by the **html_output.c** (p. 20) script.

Functions

- void **print_schedule_to_file** (**SemesterData** ∗sd, **Specialization** ∗sp, const char ∗fileName)
  Prints a schedule for a specific specialization to a file.

4.8.1 Detailed Description

This file contains prototypes required by the **html_output.c** (p. 20) script.

4.8.2 Function Documentation

4.8.2.1 void print_schedule_to_file (SemesterData ∗ *sd*, Specialization ∗ *sp*, const char ∗ *fileName*)

Prints a schedule for a specific specialization to a file.

Parameters

in	*sd*	**SemesterData** (p. 6) contains the required information about the data that should be printed
in	*sp*	**Specialization** (p. 7) contains information about the specialization the schedule is generated for
in	*fileName*	The name of the file in which the schedule should be generated

The final step of the schedule creation

4.9 scheduler.c File Reference

The main script of the program, the magic starts here.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "structs.h"
#include "scheduler.h"
#include "config_reader.h"
#include "data_utility.h"
#include "data_test.h"
#include "defs.h"
#include "html_output.h"
```

Functions

- int **compare_time** (const void ∗a, const void ∗b)

  A function to compare times of two lectures.
- int **generate_next** (**SemesterData** ∗sd)

  Brief.
- int **main** (void)

  The main function.
- void **generate_initial_schedule** (**SemesterData** ∗sd)

  Generate a 'dumb' schedule (array of lectures)
- void **free_all** (**SemesterData** ∗sd)

  *Free all memory associated with the **SemesterData** (p. 6) struct.*

4.9.1 Detailed Description

The main script of the program, the magic starts here.

4.9.2 Function Documentation

4.9.2.1 int compare_time (const void ∗ *a*, const void ∗ *b*)

A function to compare times of two lectures.

Parameters

in	*a*	**Lecture** (p. 6) one
in	*b*	**Lecture** (p. 6) two

Returns

Returns a value to qsort in order to determin the way to order the lectures array

This function is run by qsort. The function starts by comparing the two lectures by day. If they have the same day, they are compared by period

4.9.2.2 void free_all (SemesterData * *sd*)

Free all memory associated with the **SemesterData** (p. 6) struct.

Parameters

in	*sd*	The **SemesterData** (p. 6) struct

Run by main. Dynamically allocated arrays inside the structs are also freed.

4.9.2.3 void generate_initial_schedule (SemesterData * *sd*)

Generate a 'dumb' schedule (array of lectures)

Parameters

in	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function

Run by main. The only fulfilled requirement is the amount of lectures per course

4.9.2.4 int generate_next (SemesterData * *sd*)

Brief.

Parameters

in	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function

Returns

Returns the combined fitness of ?

Run by main

4.9.2.5 int main (void)

The main function.

Returns

Returns a value based on how the program exited

Run by the OS

4.10 scheduler.h File Reference

This file contains prototypes required by the **scheduler.c** (p. 24) script.

Functions

- void **free_all** (**SemesterData** ∗sd)

  Free all memory associated with the ***SemesterData*** *(p. 6) struct.*
- void **generate_initial_schedule** (**SemesterData** ∗sd)

  Generate a 'dumb' schedule (array of lectures)

4.10.1 Detailed Description

This file contains prototypes required by the **scheduler.c** (p. 24) script.

4.10.2 Function Documentation

4.10.2.1 void free_all (SemesterData ∗ *sd*)

Free all memory associated with the **SemesterData** (p. 6) struct.

Parameters

in	*sd*	The **SemesterData** (p. 6) struct

Run by main. Dynamically allocated arrays inside the structs are also freed.

4.10.2.2 void generate_initial_schedule (SemesterData ∗ *sd*)

Generate a 'dumb' schedule (array of lectures)

Parameters

in	*sd*	**SemesterData** (p. 6) contains all the information about the structs needed for this function

Run by main. The only fulfilled requirement is the amount of lectures per course

4.11 structs.h File Reference

The header file containing all the structs required by the program.

```
#include "defs.h"
```

Data Structures

- struct **Room**
- struct **OffTime**
- struct **Teacher**
- struct **Course**
- struct **Specialization**
- struct **Flags**
- struct **Lecture**
- struct **SemesterData**
- struct **Generations**

4.11.1 Detailed Description

The header file containing all the structs required by the program.

Index