JeuDeTaquin

Generated by Doxygen 1.10.0

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 GenData Struct Reference 5

3.1.1 Detailed Description 5

3.1.2 Member Data Documentation 5

3.1.2.1 size 5

3.1.2.2 startingNum 5

3.2 GraphItem Struct Reference 6

3.2.1 Detailed Description 6

3.2.2 Member Data Documentation 6

3.2.2.1 Avg 6

3.2.2.2 currRange 6

3.2.2.3 currSum 6

3.2.2.4 X 7

3.2.2.5 Y 7

3.3 ProgressArgs Struct Reference 7

3.3.1 Detailed Description 7

3.3.2 Member Data Documentation 7

3.3.2.1 MaxProgressSum 7

3.3.2.2 progressArray 8

3.3.2.3 progressEntriesCount 8

3.3.2.4 ShouldCancel 8

3.4 RandomSet Struct Reference 8

3.4.1 Detailed Description 8

3.4.2 Member Data Documentation 8

3.4.2.1 set 8

3.4.2.2 setSize 9

3.5 SaveData Struct Reference 9

3.5.1 Detailed Description 9

3.5.2 Member Data Documentation 9

3.5.2.1 basePath 9

3.5.2.2 digitsCount 9

3.5.2.3 index 10

3.5.2.4 tableau 10

3.6 Tableau Struct Reference 10

3.6.1 Detailed Description 10

3.6.2 Member Data Documentation 11

3.6.2.1 numberOfRows 11

3.6.2.2 sizesOfRows 11
3.6.2.3 startingNr 11
3.6.2.4 tableau 11
3.7 ThreadArgs Struct Reference 11
3.7.1 Detailed Description 12
3.7.2 Member Data Documentation 12
3.7.2.1 end 12
3.7.2.2 func 12
3.7.2.3 inputArray 12
3.7.2.4 outputArray 12
3.7.2.5 progress 13
3.7.2.6 start 13
3.8 UserInput Struct Reference 13
3.8.1 Detailed Description 13
3.8.2 Member Data Documentation 13
3.8.2.1 bPrintTables 13
3.8.2.2 ImgOutputPath 14
3.8.2.3 InputPath 14
3.8.2.4 TableauCount 14
3.8.2.5 TableauSize 14
3.8.2.6 TablesOutputPath 14

4 File Documentation 15

4.1 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayAnalyze/ArrayAnalyze.c File Reference 15
4.1.1 Macro Definition Documentation 15
4.1.1.1 DOUBLE_DIGITS 15
4.1.1.2 MAGIC 16
4.1.2 Function Documentation 16
4.1.2.1 LoadTableauFromFile() 16
4.1.2.2 SolveTableau() 16
4.2 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayAnalyze/ArrayAnalyze.h File Reference 16
4.2.1 Function Documentation 17
4.2.1.1 LoadTableauFromFile() 17
4.2.1.2 SolveTableau() 17
4.3 ArrayAnalyze.h 17
4.4 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/RandomSetStruct.h File Reference 17
4.5 RandomSetStruct.h 18
4.6 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauGen.c File Reference 18
4.6.1 Macro Definition Documentation 18
4.6.1.1 DIGITS_OF_STARTING_NUMBERS 18
4.6.1.2 MAXDIGITS 19
4.6.2 Function Documentation 19

4.6.2.1 GenerateTableau() . . . 19
4.6.2.2 PrintRow() . . . 19
4.6.2.3 PrintTableau() . . . 19
4.6.2.4 SaveTableau() . . . 20
4.7 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauGen.h File Reference . . . 20
4.7.1 Function Documentation . . . 21
4.7.1.1 GenerateTableau() . . . 21
4.7.1.2 PrintRow() . . . 21
4.7.1.3 PrintTableau() . . . 21
4.7.1.4 SaveTableau() . . . 21
4.8 TableauGen.h . . . 22
4.9 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauStructure.c File Reference . . . 22
4.9.1 Function Documentation . . . 23
4.9.1.1 CalculateDelta() . . . 23
4.9.1.2 FindThe2ndMaxElement() . . . 23
4.9.1.3 GenerateRandomSet() . . . 24
4.9.1.4 GenerateStartingNumbers() . . . 24
4.9.1.5 ResizeRow() . . . 24
4.9.1.6 ResizeSizesArray() . . . 25
4.9.1.7 ResizeTableau() . . . 25
4.9.1.8 ThrowElementToRow() . . . 25
4.10 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauStructure.h File Reference . . . 26
4.10.1 Function Documentation . . . 26
4.10.1.1 CalculateDelta() . . . 26
4.10.1.2 FindThe2ndMaxElement() . . . 27
4.10.1.3 GenerateRandomSet() . . . 27
4.10.1.4 GenerateStartingNumbers() . . . 27
4.10.1.5 ResizeRow() . . . 28
4.10.1.6 ResizeSizesArray() . . . 28
4.10.1.7 ResizeTableau() . . . 28
4.10.1.8 ThrowElementToRow() . . . 29
4.11 TableauStructure.h . . . 29
4.12 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/Graph.c File Reference . . . 30
4.12.1 Macro Definition Documentation . . . 30
4.12.1.1 CHECK_BOUNDS2 . . . 30
4.12.1.2 OPTIMIZED_AVG . . . 31
4.12.2 Function Documentation . . . 31
4.12.2.1 Compaper() . . . 31
4.12.2.2 GenerateDB() . . . 31
4.12.2.3 GenerateGraph() . . . 31
4.12.2.4 IsSorted() . . . 32
4.12.2.5 SetAverages() . . . 32

4.12.2.6 SortIfNotSorted() 32
4.13 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/Graph.h File Reference 34
4.13.1 Function Documentation 34
4.13.1.1 Compaper() 34
4.13.1.2 GenerateDB() 35
4.13.1.3 GenerateGraph() 35
4.13.1.4 IsSorted() 35
4.13.1.5 SetAverages() 36
4.13.1.6 SortIfNotSorted() 36
4.14 Graph.h 36
4.15 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/GraphItem.h File Reference 37
4.16 GraphItem.h 37
4.17 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners.c File Reference 37
4.17.1 Function Documentation 38
4.17.1.1 AnalyzeTables() 38
4.17.1.2 GenerateTables() 38
4.17.1.3 GetTableauSize() 39
4.17.1.4 LoadTableaus() 39
4.17.1.5 PrintTables() 39
4.17.1.6 SaveTableaus() 40
4.18 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners.h File Reference 40
4.18.1 Function Documentation 40
4.18.1.1 AnalyzeTables() 40
4.18.1.2 GenerateTables() 41
4.18.1.3 GetTableauSize() 41
4.18.1.4 LoadTableaus() 41
4.18.1.5 PrintTables() 42
4.18.1.6 SaveTableaus() 42
4.19 BatchRunners.h 42
4.20 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Analyze.c File Reference 43
4.20.1 Function Documentation 43
4.20.1.1 AnalyzeTable_Thread() 43
4.20.1.2 AnalyzeTablesMultiThreaded() 43
4.21 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Analyze.h File Reference 44
4.21.1 Function Documentation 44
4.21.1.1 AnalyzeTable_Thread() 44
4.21.1.2 AnalyzeTablesMultiThreaded() 44
4.22 Analyze.h 45
4.23 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Generate.c File Reference 45
4.23.1 Function Documentation 45
4.23.1.1 GenerateTablesMultiThread() 45
4.23.1.2 GenerateTablesSingleThread() 46

4.23.1.3 GenTable_Thread() 46
4.24 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Generate.h File Reference 46
4.24.1 Function Documentation 46
4.24.1.1 GenerateTablesMultiThread() 46
4.24.1.2 GenerateTablesSingleThread() 47
4.24.1.3 GenTable_Thread() 47
4.25 Generate.h 47
4.26 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/GenerateData.h File Reference 48
4.27 GenerateData.h 48
4.28 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Load.c File Reference 48
4.28.1 Function Documentation 49
4.28.1.1 LoadTable_Thread() 49
4.28.1.2 LoadTableausMultiThread() 49
4.28.1.3 LoadTableausSingleThread() 49
4.29 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Load.h File Reference 49
4.29.1 Function Documentation 50
4.29.1.1 LoadTable_Thread() 50
4.29.1.2 LoadTableausMultiThread() 50
4.29.1.3 LoadTableausSingleThread() 51
4.30 Load.h 51
4.31 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Save.c File Reference 51
4.31.1 Function Documentation 52
4.31.1.1 SaveTable_Thread() 52
4.31.1.2 SaveTableausMultiThreaded() 52
4.31.1.3 SaveTableausSingleThread() 52
4.32 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Save.h File Reference 53
4.32.1 Function Documentation 53
4.32.1.1 SaveTable_Thread() 53
4.32.1.2 SaveTableausMultiThreaded() 54
4.32.1.3 SaveTableausSingleThread() 54
4.33 Save.h 54
4.34 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/SaveData.h File Reference 55
4.35 SaveData.h 55
4.36 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Clock.c File Reference 55
4.36.1 Function Documentation 55
4.36.1.1 GetCurrTimeMs() 55
4.37 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Clock.h File Reference 56
4.37.1 Function Documentation 56
4.37.1.1 GetCurrTimeMs() 56
4.38 Clock.h 56
4.39 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Exceptions.c File Reference 56
4.39.1 Function Documentation 56

4.39.1.1 ChangeConsoleColor() 56
4.39.1.2 Log() 57
4.40 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Exceptions.h File Reference 57
4.40.1 Macro Definition Documentation 58
4.40.1.1 LOG 58
4.40.1.2 LOG_ERROR 58
4.40.1.3 LOG_WARNING 58
4.40.2 Enumeration Type Documentation 58
4.40.2.1 LogType 58
4.40.3 Function Documentation 59
4.40.3.1 ChangeConsoleColor() 59
4.40.3.2 Log() 59
4.41 Exceptions.h 59
4.42 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/MultithreadHelper.c File Reference 60
4.42.1 Function Documentation 60
4.42.1.1 GetCoresCount() 60
4.42.1.2 RunBatch() 60
4.42.1.3 RunBatchThread() 61
4.42.1.4 RunProgressThread() 61
4.42.1.5 UpdateProgress() 61
4.43 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/MultithreadHelper.h File Reference 63
4.43.1 Function Documentation 63
4.43.1.1 GetCoresCount() 63
4.43.1.2 RunBatch() 64
4.43.1.3 RunBatchThread() 64
4.43.1.4 RunProgressThread() 64
4.43.1.5 UpdateProgress() 65
4.44 MultithreadHelper.h 65
4.45 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ProjectRequirements.h File Reference 66
4.46 ProjectRequirements.h 66
4.47 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/UserInput.c File Reference 66
4.47.1 Function Documentation 66
4.47.1.1 DrawUsage() 66
4.47.1.2 DrawVersion() 67
4.47.1.3 ReadUserInputFromArgs() 67
4.47.1.4 ReadUserInputFromPrompts() 67
4.47.1.5 ShouldUseExistingTables() 67
4.47.1.6 TakeUserInput() 68
4.47.1.7 ValidateUserInput() 68
4.48 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/UserInput.h File Reference 68
4.48.1 Function Documentation 69
4.48.1.1 DrawUsage() 69

4.48.1.2 DrawVersion() 69
4.48.1.3 ReadUserInputFromArgs() 69
4.48.1.4 ReadUserInputFromPrompts() 70
4.48.1.5 ShouldUseExistingTables() 70
4.48.1.6 TakeUserInput() 70
4.48.1.7 ValidateUserInput() 71
4.49 UserInput.h 71
4.50 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/UserInputStruct.h File Reference 71
4.51 UserInputStruct.h 72
4.52 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Version.h File Reference 72
4.52.1 Macro Definition Documentation 72
4.52.1.1 MAGIC 72
4.52.1.2 VERSION 72
4.53 Version.h 72
4.54 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/JeuDeTaquin.c File Reference 72
4.54.1 Macro Definition Documentation 73
4.54.1.1 MULTITHREAD 73
4.54.2 Function Documentation 73
4.54.2.1 main() 73
4.55 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/JeuDeTaquin.h File Reference 73
4.56 JeuDeTaquin.h 73

Index **75**

Chapter 1

Class Index

1.1 Class List

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

GenData
A tuple with information for generating each array 5
GraphItem
A struct to represent one point on the graph 6
ProgressArgs
Arguments to pass to a progress counter thread 7
RandomSet
Random Set which will be used in generating the **Tableau** (p. 10) 8
SaveData
A tuple with information for saving each array 9
The tables are represented in French notation, with rows numbered in that order 10
ThreadArgs
Arguments to pass to a worker thread for main threading 11
UserInput
Provides user's input data necessary to generate tables 13

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ **JeuDeTaquin.c** 72
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ **JeuDeTaquin.h** 73
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayAnalyze/ **ArrayAnalyze.c** 15
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayAnalyze/ **ArrayAnalyze.h** 16
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/ **RandomSetStruct.h** 17
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/ **TableauGen.c** 18
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/ **TableauGen.h** 20
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/ **TableauStructure.c** 22
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/ **TableauStructure.h** 26
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/ **Graph.c** 30
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/ **Graph.h** 34
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/ **GraphItem.h** 37
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **BatchRunners.c** 37
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **BatchRunners.h** 40
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **Clock.c** 55
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **Clock.h** 56
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **Exceptions.c** 56
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **Exceptions.h** 57
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **MultithreadHelper.c** 60
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **MultithreadHelper.h** 63
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **ProjectRequirements.h** 66
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **UserInput.c** 66
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **UserInput.h** 68
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **UserInputStruct.h** 71
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **Version.h** 72
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **Analyze.c** 43
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **Analyze.h** 44
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **Generate.c** 45
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **Generate.h** 46
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **GenerateData.h** 48
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **Load.c** 48
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **Load.h** 49
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **Save.c** 51
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **Save.h** 53
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **SaveData.h** 55

Chapter 3

Class Documentation

3.1 GenData Struct Reference

A tuple with information for generating each array.

```
#include <GenerateData.h>
```

Public Attributes

- float **startingNum**
  First number in an array.
- int **size**
  Amount of items in the array.

3.1.1 Detailed Description

A tuple with information for generating each array.

3.1.2 Member Data Documentation

3.1.2.1 size

```
int GenData::size
```

Amount of items in the array.

3.1.2.2 startingNum

```
float GenData::startingNum
```

First number in an array.

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

- S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **GenerateData.h**

3.2 GraphItem Struct Reference

A struct to represent one point on the graph.

```
#include <GraphItem.h>
```

Public Attributes

- float **X**
  X-axis index: starting num of the table.
- int **Y**
  Y-axis item: sum of column + row of the result.
- double **Avg**
  Moving average of Y. Should be filled from ***SetAverages()*** (p. *32) function.*
- int **currSum**
  Current sum of nearby neighbords. Used for generating moving average.
- int **currRange**
  Current range of the moving average. Gets smaller near borders.

3.2.1 Detailed Description

A struct to represent one point on the graph.

3.2.2 Member Data Documentation

3.2.2.1 Avg

```
double GraphItem::Avg
```

Moving average of Y. Should be filled from **SetAverages()** (p. 32) function.

3.2.2.2 currRange

```
int GraphItem::currRange
```

Current range of the moving average. Gets smaller near borders.

3.2.2.3 currSum

```
int GraphItem::currSum
```

Current sum of nearby neighbords. Used for generating moving average.

3.2.2.4 X

```
float GraphItem::X
```

X-axis index: starting num of the table.

3.2.2.5 Y

```
int GraphItem::Y
```

Y-axis item: sum of column + row of the result.

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

- S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/ **GraphItem.h**

3.3 ProgressArgs Struct Reference

Arguments to pass to a progress counter thread.

```
#include <MultithreadHelper.h>
```

Public Attributes

- int ∗∗ **progressArray**
  An array of pointers to progresses' of each worker.
- int **MaxProgressSum**
  Expected sum of all progresses.
- int **progressEntriesCount**
  Amount of workers.
- bool ∗ **ShouldCancel**
  Set to true once main work is done so that the counter thread can stop listening.

3.3.1 Detailed Description

Arguments to pass to a progress counter thread.

3.3.2 Member Data Documentation

3.3.2.1 MaxProgressSum

```
int ProgressArgs::MaxProgressSum
```

Expected sum of all progresses.

Returns

3.3.2.2 progressArray

```
int** ProgressArgs::progressArray
```

An array of pointers to progresses' of each worker.

3.3.2.3 progressEntriesCount

```
int ProgressArgs::progressEntriesCount
```

Amount of workers.

3.3.2.4 ShouldCancel

```
bool* ProgressArgs::ShouldCancel
```

Set to true once main work is done so that the counter thread can stop listening.

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

- S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **MultithreadHelper.h**

3.4 RandomSet Struct Reference

Random Set which will be used in generating the **Tableau** (p. 10).

```
#include <RandomSetStruct.h>
```

Public Attributes

- int **setSize**
- float ∗ **set**

3.4.1 Detailed Description

Random Set which will be used in generating the **Tableau** (p. 10).

3.4.2 Member Data Documentation

3.4.2.1 set

```
float* RandomSet::set
```

3.4.2.2 setSize

```
int RandomSet::setSize
```

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

- S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/ **RandomSetStruct.h**

3.5 SaveData Struct Reference

A tuple with information for saving each array.

```
#include <SaveData.h>
```

Public Attributes

- struct **Tableau** ∗ **tableau**

  A tableau.
- char ∗ **basePath**

  Path to base directory tableau will be saved in.
- int **index**

  ***Tableau** (p. 10) index which will be included in the file name.*
- int **digitsCount**

  Count of digits to include in the file name - to help with sorting.

3.5.1 Detailed Description

A tuple with information for saving each array.

3.5.2 Member Data Documentation

3.5.2.1 basePath

```
char* SaveData::basePath
```

Path to base directory tableau will be saved in.

3.5.2.2 digitsCount

```
int SaveData::digitsCount
```

Count of digits to include in the file name - to help with sorting.

3.5.2.3 index

```
int SaveData::index
```

Tableau (p. 10) index which will be included in the file name.

3.5.2.4 tableau

```
struct Tableau* SaveData::tableau
```

A tableau.

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

- S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/ **SaveData.h**

3.6 Tableau Struct Reference

Represents a Young **Tableau** (p. 10)

The tables are represented in French notation, with rows numbered in that order.

```
#include <TableauStructure.h>
```

Public Attributes

- float **startingNr**

  Number of first element in randomset of the tableau.
- int * **sizesOfRows**

  Sizes of each rows.
- int **numberOfRows**

  Number of rows.
- float ** **tableau**

  *2D Array, THE Young **Tableau** (p. 10), with rows numbered in the order of French notation:*
  _ 0 1 2
  3[]
  2[]
  1[][]
  0[][][]
  Therefore the cell[0][0] is in the down - left corner

3.6.1 Detailed Description

Represents a Young **Tableau** (p. 10)

The tables are represented in French notation, with rows numbered in that order.

3.6.2 Member Data Documentation

3.6.2.1 numberOfRows

```
int Tableau::numberOfRows
```

Number of rows.

3.6.2.2 sizesOfRows

```
int* Tableau::sizesOfRows
```

Sizes of each rows.

3.6.2.3 startingNr

```
float Tableau::startingNr
```

Number of first element in randomset of the tableau.

3.6.2.4 tableau

```
float** Tableau::tableau
```

2D Array, THE Young **Tableau** (p. 10), with rows numbered in the order of French notation:

_ 0 1 2

3[]

2[]

1[][]

0[][][]

Therefore the cell[0][0] is in the down - left corner

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

- S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/ **TableauStructure.h**

3.7 ThreadArgs Struct Reference

Arguments to pass to a worker thread for main threading.

```
#include <MultithreadHelper.h>
```

Public Attributes

- void ∗(∗ **func**)(void ∗)

  Function to call on each data item. Argument is an item from inputArray; return value is written to outputArray.
- void ∗∗ **inputArray**

  Array with input values. Set to NULL to skip.
- void ∗∗ **outputArray**

  Array for output values. Set to NULL to skip.
- int **start**

  Start index for this thread.
- int **end**

  End index for this thread.
- int ∗ **progress**

  Pointer to a progress counter which can be updated.

3.7.1 Detailed Description

Arguments to pass to a worker thread for main threading.

3.7.2 Member Data Documentation

3.7.2.1 end

```
int ThreadArgs::end
```

End index for this thread.

3.7.2.2 func

```
void *(* ThreadArgs::func) (void *)
```

Function to call on each data item. Argument is an item from inputArray; return value is written to outputArray.

3.7.2.3 inputArray

```
void** ThreadArgs::inputArray
```

Array with input values. Set to NULL to skip.

3.7.2.4 outputArray

```
void** ThreadArgs::outputArray
```

Array for output values. Set to NULL to skip.

3.7.2.5 progress

```
int* ThreadArgs::progress
```

Pointer to a progress counter which can be updated.

3.7.2.6 start

```
int ThreadArgs::start
```

Start index for this thread.

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

- S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **MultithreadHelper.h**

3.8 UserInput Struct Reference

Provides user's input data necessary to generate tables.

```
#include <UserInputStruct.h>
```

Public Attributes

- int **TableauSize**
  Amount of elements in each tableau.
- int **TableauCount**
  Amount of tableaus.
- char ∗ **InputPath**
  OPTIONAL: path to load tableaus from.
- char ∗ **TablesOutputPath**
  OPTIONAL: path to save the generated tables to.
- char ∗ **ImgOutputPath**
  OPTIONAL: path to save the generated image to.
- bool **bPrintTables**
  OPTIONAL: if true, print tables before analyzing them.

3.8.1 Detailed Description

Provides user's input data necessary to generate tables.

3.8.2 Member Data Documentation

3.8.2.1 bPrintTables

```
bool UserInput::bPrintTables
```

OPTIONAL: if true, print tables before analyzing them.

3.8.2.2 ImgOutputPath

```
char* UserInput::ImgOutputPath
```

OPTIONAL: path to save the generated image to.

3.8.2.3 InputPath

```
char* UserInput::InputPath
```

OPTIONAL: path to load tableaus from.

3.8.2.4 TableauCount

```
int UserInput::TableauCount
```

Amount of tableaus.

3.8.2.5 TableauSize

```
int UserInput::TableauSize
```

Amount of elements in each tableau.

3.8.2.6 TablesOutputPath

```
char* UserInput::TablesOutputPath
```

OPTIONAL: path to save the generated tables to.

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

- S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ **UserInputStruct.h**

Chapter 4

File Documentation

4.1 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayAnalyze/ArrayAnalyze.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "../ArrayGen/TableauStructure.h"
#include "ArrayAnalyze.h"
#include "../Helpers/ProjectRequirements.h"
#include "..\Helpers\Version.h"
#include "..\Helpers\Exceptions.h"
```

Macros

- #define **MAGIC** "TAB"
- #define **DOUBLE_DIGITS** 18

Functions

- struct **Tableau** ∗ **LoadTableauFromFile** (char ∗filePath)

  Loads tableau from file.
- int **SolveTableau** (struct **Tableau** ∗tableau)

  *Solves Young **Tableau** (p. 10).*

4.1.1 Macro Definition Documentation

4.1.1.1 DOUBLE_DIGITS

```
#define DOUBLE_DIGITS 18
```

4.1.1.2 MAGIC

```
#define MAGIC "TAB"
```

4.1.2 Function Documentation

4.1.2.1 LoadTableauFromFile()

```
struct  Tableau * LoadTableauFromFile (
             char * filePath )
```

Loads tableau from file.

Parameters

filePath	Path to file containing tableau

Returns

Loaded tableau

4.1.2.2 SolveTableau()

```
int SolveTableau (
             struct  Tableau * tableau )
```

Solves Young **Tableau** (p. 10).

Parameters

tableau	**Tableau** (p. 10) to solve

Returns

The sum of row and column on which the game ended

4.2 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayAnalyze/ArrayAnalyze.h File Reference

```
#include "../ArrayGen/TableauStructure.h"
```

Functions

- struct **Tableau** ∗ **LoadTableauFromFile** (char ∗filePath)
  Loads tableau from file.
- int **SolveTableau** (struct **Tableau** ∗tableau)
  *Solves Young **Tableau** (p. 10).*

4.2.1 Function Documentation

4.2.1.1 LoadTableauFromFile()

```
struct  Tableau * LoadTableauFromFile (
             char * filePath )
```

Loads tableau from file.

Parameters

filePath	Path to file containing tableau

Returns

Loaded tableau

4.2.1.2 SolveTableau()

```
int SolveTableau (
             struct  Tableau * tableau )
```

Solves Young **Tableau** (p. 10).

Parameters

tableau	**Tableau** (p. 10) to solve

Returns

The sum of row and column on which the game ended

4.3 ArrayAnalyze.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 #include "../ArrayGen/TableauStructure.h"
00004
00010 struct Tableau* LoadTableauFromFile(char* filePath);
00011
00017 int SolveTableau(struct Tableau* tableau);
```

4.4 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/RandomSetStruct.h File Reference

Classes

- struct **RandomSet**

  Random Set which will be used in generating the ***Tableau*** *(p. 10).*

4.5 RandomSetStruct.h

Go to the documentation of this file.

```
#pragma once
struct RandomSet
{
    int setSize; //size of set of randomly generated numbers, the size is given as an input
    float* set; // set of randomly generated numbers, those will be used to generate a tabeleux
};
```

4.6 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauGen.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "TableauStructure.h"
#include "RandomSetStruct.h"
#include <math.h>
#include <stdbool.h>
#include <Windows.h>
#include "..\Helpers\Version.h"
#include "..\Helpers\Exceptions.h"
```

Macros

- #define **MAXDIGITS** 10
- #define **DIGITS_OF_STARTING_NUMBERS** 2

Functions

- void **SaveTableau** (struct **Tableau** tab, char path[ ])

  Saves one tab to a file
  FILE STRUCTURE:
  1st line: Magic number
  2nd line: Version
  3rd line: ***Tableau*** (p. *10) starting num*
  4th line: Rows count
  5th line: Length of the longest row
  rest: tableau
- void **PrintRow** (float row[ ], int size)

  Prints single row.
- void **PrintTableau** (struct **Tableau** tab)

  Prints all the tableau.
- struct **Tableau** ∗ **GenerateTableau** (double startingNum, int setSize)

  Main generating process.

4.6.1 Macro Definition Documentation

4.6.1.1 DIGITS_OF_STARTING_NUMBERS

```
#define DIGITS_OF_STARTING_NUMBERS 2
```

4.6.1.2 MAXDIGITS

```
#define MAXDIGITS 10
```

4.6.2 Function Documentation

4.6.2.1 GenerateTableau()

```
struct Tableau * GenerateTableau (
            double startingNum,
            int setSize )
```

Main generating process.

Parameters

startingNum	Starting number in the set
setSize	Size of the set which will be used in generating the tableau

Returns

Generated Young **Tableau** (p. 10)

4.6.2.2 PrintRow()

```
void PrintRow (
            float row[],
            int size )
```

Prints single row.

Parameters

row	Row to print
size	Size of the row

4.6.2.3 PrintTableau()

```
void PrintTableau (
            struct Tableau tab )
```

Prints all the tableau.

Parameters

tab	**Tableau** (p. 10) to print

4.6.2.4 SaveTableau()

```
void SaveTableau (
             struct  Tableau tab,
             char path[ ] )
```

Saves one tab to a file

FILE STRUCTURE:

1st line: Magic number

2nd line: Version

3rd line: **Tableau** (p. 10) starting num

4th line: Rows count

5th line: Length of the longest row

rest: tableau

Parameters

tab	**Tableau** (p. 10) to save
path	Path to save the file in

4.7 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauGen.h File Reference

Functions

- void **SaveTableau** (struct **Tableau** tab, char path[ ])

  Saves one tab to a file
  FILE STRUCTURE:
  1st line: Magic number
  2nd line: Version
  *3rd line: **Tableau** (p. 10) starting num*
  4th line: Rows count
  5th line: Length of the longest row
  rest: tableau
- void **PrintRow** (float row[ ], int size)

  Prints single row.
- void **PrintTableau** (struct **Tableau** tab)

  Prints all the tableau.
- struct **Tableau** ∗ **GenerateTableau** (double startingNum, int setSize)

  Main generating process.

4.7.1 Function Documentation

4.7.1.1 GenerateTableau()

```
struct  Tableau * GenerateTableau (
             double startingNum,
             int setSize )
```

Main generating process.

Parameters

startingNum	Starting number in the set
setSize	Size of the set which will be used in generating the tableau

Returns

Generated Young **Tableau** (p. 10)

4.7.1.2 PrintRow()

```
void PrintRow (
             float row[],
             int size )
```

Prints single row.

Parameters

row	Row to print
size	Size of the row

4.7.1.3 PrintTableau()

```
void PrintTableau (
             struct  Tableau tab )
```

Prints all the tableau.

Parameters

tab	**Tableau** (p. 10) to print

4.7.1.4 SaveTableau()

```
void SaveTableau (
struct Tableau tab,
char path[] )
```

Saves one tab to a file

FILE STRUCTURE:

1st line: Magic number

2nd line: Version

3rd line: **Tableau** (p. 10) starting num

4th line: Rows count

5th line: Length of the longest row

rest: tableau

Parameters

tab	**Tableau** (p. 10) to save
path	Path to save the file in

4.8 TableauGen.h

Go to the documentation of this file.

```
#pragma once
void SaveTableau(struct Tableau tab, char path[]);

void PrintRow(float row[], int size);

void PrintTableau(struct Tableau tab);

struct Tableau* GenerateTableau(double startingNum, int setSize);
```

4.9 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauStructure.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include "../Helpers/ProjectRequirements.h"
```

Functions

- float ∗ **ResizeRow** (float ∗row, int ∗size)

  Resizes a single row.
- int ∗ **ResizeSizesArray** (int ∗sizes, int currentRowsCounter)

  Resizes an array which holds the sizes.

- float ∗∗ **ResizeTableau** (float ∗∗tableau, int ∗numberOfRows)

  *Resizes 2D array called **Tableau** (p. 10).*
- void **GenerateRandomSet** (float set[ ], int size)

  Generates random set.
- float **CalculateDelta** (int howMany)

  Calculates difference between n numbers in range (0,1)
- float ∗ **GenerateStartingNumbers** (float delta, int howManyNumbers)

  Generates starting numbers from Delta, the starting numbers used in creating the graph.
- float ∗ **FindThe2ndMaxElement** (float ∗row, float newElement, int ∗rowSize)

  Finds the 2nd max element in a row.
- float ∗ **ThrowElementToRow** (float ∗row, float element, int ∗rowSize, float ∗elementToThrowOut)

  In the tableau generating process, it throws the 2nd max element to the next row.

4.9.1 Function Documentation

4.9.1.1 CalculateDelta()

```
float CalculateDelta (
            int howMany )
```

Calculates difference between n numbers in range (0,1)

Parameters

howMany	How many number should there be in the range of (0,1)

Returns

The difference variable

4.9.1.2 FindThe2ndMaxElement()

```
float * FindThe2ndMaxElement (
            float * row,
            float newElement,
            int * rowSize )
```

Finds the 2nd max element in a row.

Parameters

row	Row where we need to find the element
newElement	Element to put in the place of the old
rowSize	Size of the row

Returns

Updated row

4.9.1.3 GenerateRandomSet()

```
void GenerateRandomSet (
             float set[],
             int size )
```

Generates random set.

Parameters

set	Array which contains the set
size	Size of set

4.9.1.4 GenerateStartingNumbers()

```
float * GenerateStartingNumbers (
             float delta,
             int howManyNumbers )
```

Generates starting numbers from Delta, the starting numbers used in creating the graph.

Parameters

delta	Difference between numbers calculated in CalculateDelta function
howManyNumbers	How many numbers to generate

Returns

Generated array of the generated numbers

4.9.1.5 ResizeRow()

```
float * ResizeRow (
             float * row,
             int * size )
```

Resizes a single row.

Parameters

row	A row, an array to resize
size	Pointer to the size variable of the row

Returns

Resized row

4.9.1.6 ResizeSizesArray()

```
int * ResizeSizesArray (
             int * sizes,
             int currentRowsCounter )
```

Resizes an array which holds the sizes.

Parameters

sizes	Array which will be resized
currentRowsCounter	Size of that array

Returns

Resized array

4.9.1.7 ResizeTableau()

```
float ** ResizeTableau (
             float ** tableau,
             int * numberOfRows )
```

Resizes 2D array called **Tableau** (p. 10).

Parameters

tableau	The 2D array tableau, an element of the struct **Tableau** (p. 10)
numberOfRows	number of rows

Returns

2D array of floats which will be the new tableau

4.9.1.8 ThrowElementToRow()

```
float * ThrowElementToRow (
             float * row,
             float element,
             int * rowSize,
             float * elementToThrowOut )
```

In the tableau generating process, it throws the 2nd max element to the next row.

Parameters

row	Row where we throw the element
element	Element to throw to the current row
rowSize	Size of the row
elementToThrowOut	Pointer to an element to throw to the next row

Returns

4.10 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauStructure.h File Reference

Classes

- struct **Tableau**

  Represents a Young ***Tableau*** *(p. 10)*
  The tables are represented in French notation, with rows numbered in that order.

Functions

- float ∗ **ResizeRow** (float ∗row, int ∗size)

  Resizes a single row.
- int ∗ **ResizeSizesArray** (int ∗sizes, int currentRowsCounter)

  Resizes an array which holds the sizes.
- float ∗∗ **ResizeTableau** (float ∗∗tableau, int ∗numberOfRows)

  Resizes 2D array called ***Tableau*** *(p. 10).*
- void **GenerateRandomSet** (float set[ ], int size)

  Generates random set.
- float **CalculateDelta** (int howMany)

  Calculates difference between n numbers in range (0,1)
- float ∗ **GenerateStartingNumbers** (float delta, int howManyNumbers)

  Generates starting numbers from Delta, the starting numbers used in creating the graph.
- float ∗ **FindThe2ndMaxElement** (float ∗row, float newElement, int ∗rowSize)

  Finds the 2nd max element in a row.
- float ∗ **ThrowElementToRow** (float ∗row, float element, int ∗rowSize, float ∗elementToThrowOut)

  In the tableau generating process, it throws the 2nd max element to the next row.

4.10.1 Function Documentation

4.10.1.1 CalculateDelta()

```
float CalculateDelta (
            int howMany )
```

Calculates difference between n numbers in range (0,1)

Parameters

howMany	How many number should there be in the range of (0,1)

Returns

The difference variable

4.10.1.2 FindThe2ndMaxElement()

```
float * FindThe2ndMaxElement (
             float * row,
             float newElement,
             int * rowSize )
```

Finds the 2nd max element in a row.

Parameters

row	Row where we need to find the element
newElement	Element to put in the place of the old
rowSize	Size of the row

Returns

Updated row

4.10.1.3 GenerateRandomSet()

```
void GenerateRandomSet (
             float set[],
             int size )
```

Generates random set.

Parameters

set	Array which contains the set
size	Size of set

4.10.1.4 GenerateStartingNumbers()

```
float * GenerateStartingNumbers (
             float delta,
             int howManyNumbers )
```

Generates starting numbers from Delta, the starting numbers used in creating the graph.

Parameters

delta	Difference between numbers calculated in CalculateDelta function
howManyNumbers	How many numbers to generate

Returns

Generated array of the generated numbers

4.10.1.5 ResizeRow()

```
float * ResizeRow (
            float * row,
            int * size )
```

Resizes a single row.

Parameters

row	A row, an array to resize
size	Pointer to the size variable of the row

Returns

Resized row

4.10.1.6 ResizeSizesArray()

```
int * ResizeSizesArray (
            int * sizes,
            int currentRowsCounter )
```

Resizes an array which holds the sizes.

Parameters

sizes	Array which will be resized
currentRowsCounter	Size of that array

Returns

Resized array

4.10.1.7 ResizeTableau()

```
float ** ResizeTableau (
            float ** tableau,
            int * numberOfRows )
```

Resizes 2D array called **Tableau** (p. 10).

Parameters

tableau	The 2D array tableau, an element of the struct **Tableau** (p. 10)
numberOfRows	number of rows

Returns

2D array of floats which will be the new tableau

4.10.1.8 ThrowElementToRow()

```
float * ThrowElementToRow (
             float * row,
             float element,
             int * rowSize,
             float * elementToThrowOut )
```

In the tableau generating process, it throws the 2nd max element to the next row.

Parameters

row	Row where we throw the element
element	Element to throw to the current row
rowSize	Size of the row
elementToThrowOut	Pointer to an element to throw to the next row

Returns

4.11 TableauStructure.h

Go to the documentation of this file.

```
#pragma once

struct Tableau {
    float startingNr;

    int* sizesOfRows;

    int numberOfRows;

    float** tableau;

};

/*MASSIVE WARNING
For a while ill be using these 3 resize functions in that way:
function returns pointer to some array
i copy that array itd
why?
because i want to move on and ill repair it later*/

float* ResizeRow(float* row, int* size);

int* ResizeSizesArray(int* sizes, int currentRowsCounter);

float** ResizeTableau(float** tableau, int* numberOfRows);
```

```

void GenerateRandomSet(float set[], int size);

float CalculateDelta(int howMany);

float* GenerateStartingNumbers(float delta, int howManyNumbers);

float* FindThe2ndMaxElement(float* row, float newElement, int* rowSize);

float* ThrowElementToRow(float* row, float element, int* rowSize, float* elementToThrowOut);


```

4.12 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/Graph.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <stdbool.h>
#include <assert.h>
#include "Graph.h"
#include "GraphItem.h"
#include "../Helpers/Exceptions.h"
#include "../Helpers/Clock.h"
```

Macros

- #define **OPTIMIZED_AVG**
- #define **CHECK_BOUNDS2**

Functions

- void **SetAverages** (struct **GraphItem** ∗∗arr, int n)

  PRIVATE: add moving-averages to a set of GraphItems.
- void **SortIfNotSorted** (struct **GraphItem** ∗∗arr, int n)

  Sort an array of GraphItems by their X-axis value if they are not sorted.
- int **Compaper** (struct **GraphItem** ∗a, struct **GraphItem** ∗b)

  Compares the X-axis values of GraphItems.
- bool **IsSorted** (struct **GraphItem** ∗∗arr, int n)

  Checks whether all GraphItems are sorted ascending by their X-axis value.
- char ∗ **GenerateDB** (struct **GraphItem** ∗∗arr, int n)

  Generate a GNUPlot database of GraphItems.
- char ∗ **GenerateGraph** (struct **GraphItem** ∗∗arr, int n, char ∗imgPath, int tableSize)

  Generate a GNUPlot graph.

4.12.1 Macro Definition Documentation

4.12.1.1 CHECK_BOUNDS2

```
#define CHECK_BOUNDS2
```

4.12.1.2 OPTIMIZED_AVG

```
#define OPTIMIZED_AVG
```

4.12.2 Function Documentation

4.12.2.1 Compaper()

```
int Compaper (
            struct  GraphItem * a,
            struct  GraphItem * b )
```

Compares the X-axis values of GraphItems.

Parameters

a	First **GraphItem** (p. 6) to compare
b	Second **GraphItem** (p. 6) to compare

Returns

Standard comparison rules (a->X - b->X)

4.12.2.2 GenerateDB()

```
char * GenerateDB (
            struct  GraphItem ** arr,
            int n )
```

Generate a GNUPlot database of GraphItems.

Parameters

arr	An array of GraphItem*-s
n	count

Returns

Name of a generated database file

4.12.2.3 GenerateGraph()

```
char * GenerateGraph (
            struct  GraphItem ** arr,
            int n,
            char * imgPath,
            int tableSize )
```

Generate a GNUPlot graph.

Parameters

arr	An array of GraphItem∗-s
n	count
imgPath	Path where the resulting image will be saved
tableSize	Amount of items that existed in each table

Returns

Path of generated image

4.12.2.4 IsSorted()

```
bool IsSorted (
             struct  GraphItem ** arr,
             int n )
```

Checks whether all GraphItems are sorted ascending by their X-axis value.

Parameters

arr	An array of GraphItem∗-s
n	count

Returns

Whether all GraphItems are sorted ascending by their X-axis value

4.12.2.5 SetAverages()

```
void SetAverages (
             struct  GraphItem ** arr,
             int n )
```

PRIVATE: add moving-averages to a set of GraphItems.

Parameters

arr	An array of GraphItem∗-s
n	count

4.12.2.6 SortIfNotSorted()

```
void SortIfNotSorted (
             struct  GraphItem ** arr,
             int n )
```

Sort an array of GraphItems by their X-axis value if they are not sorted.

Parameters

arr	An array of GraphItem∗-s
n	count

4.13 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/Graph.h File Reference

```
#include "GraphItem.h"
```

Functions

- void **SetAverages** (struct **GraphItem** ∗∗arr, int n)

  PRIVATE: add moving-averages to a set of GraphItems.
- char ∗ **GenerateDB** (struct **GraphItem** ∗∗arr, int n)

  Generate a GNUPlot database of GraphItems.
- char ∗ **GenerateGraph** (struct **GraphItem** ∗∗arr, int n, char ∗imgPath, int tableSize)

  Generate a GNUPlot graph.
- void **SortIfNotSorted** (struct **GraphItem** ∗∗arr, int n)

  Sort an array of GraphItems by their X-axis value if they are not sorted.
- int **Compaper** (struct **GraphItem** ∗a, struct **GraphItem** ∗b)

  Compares the X-axis values of GraphItems.
- bool **IsSorted** (struct **GraphItem** ∗∗arr, int n)

  Checks whether all GraphItems are sorted ascending by their X-axis value.

4.13.1 Function Documentation

4.13.1.1 Compaper()

```
int Compaper (
             struct  GraphItem * a,
             struct  GraphItem * b )
```

Compares the X-axis values of GraphItems.

Parameters

a	First **GraphItem** (p. 6) to compare
b	Second **GraphItem** (p. 6) to compare

Returns

Standard comparison rules (a->X - b->X)

4.13.1.2 GenerateDB()

```
char * GenerateDB (
            struct  GraphItem ** arr,
            int n )
```

Generate a GNUPlot database of GraphItems.

Parameters

arr	An array of GraphItem∗-s
n	count

Returns

Name of a generated database file

4.13.1.3 GenerateGraph()

```
char * GenerateGraph (
            struct  GraphItem ** arr,
            int n,
            char * imgPath,
            int tableSize )
```

Generate a GNUPlot graph.

Parameters

arr	An array of GraphItem∗-s
n	count
imgPath	Path where the resulting image will be saved
tableSize	Amount of items that existed in each table

Returns

Path of generated image

4.13.1.4 IsSorted()

```
bool IsSorted (
            struct  GraphItem ** arr,
            int n )
```

Checks whether all GraphItems are sorted ascending by their X-axis value.

Parameters

arr	An array of GraphItem∗-s
n	count

Returns

Whether all GraphItems are sorted ascending by their X-axis value

4.13.1.5 SetAverages()

```
void SetAverages (
            struct  GraphItem ** arr,
            int n )
```

PRIVATE: add moving-averages to a set of GraphItems.

Parameters

arr	An array of GraphItem∗-s
n	count

4.13.1.6 SortIfNotSorted()

```
void SortIfNotSorted (
            struct  GraphItem ** arr,
            int n )
```

Sort an array of GraphItems by their X-axis value if they are not sorted.

Parameters

arr	An array of GraphItem∗-s
n	count

4.14 Graph.h

Go to the documentation of this file.

```
#pragma once

#include "GraphItem.h"

void SetAverages(struct GraphItem** arr, int n);

char* GenerateDB(struct GraphItem** arr, int n);

char* GenerateGraph(struct GraphItem** arr, int n, char* imgPath, int tableSize);

void SortIfNotSorted(struct GraphItem** arr, int n);

int Compaper(struct GraphItem* a, struct GraphItem* b);

bool IsSorted(struct GraphItem** arr, int n);
```

4.15 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/GraphItem.h File Reference

Classes

- struct **GraphItem**

  A struct to represent one point on the graph.

4.16 GraphItem.h

Go to the documentation of this file.

```
#pragma once

struct GraphItem {
    float X;

    int Y;

    double Avg;

    int currSum;

    int currRange;
};
```

4.17 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners.c File Reference

```
#include "BatchRunners.h"
#include <stdlib.h>
#include "../Graph/GraphItem.h"
#include "../ArrayGen/TableauStructure.h"
#include "../ArrayGen/TableauGen.h"
#include "BatchRunners/Generate.h"
#include "BatchRunners/Analyze.h"
#include <stdbool.h>
#include "../ArrayAnalyze/ArrayAnalyze.h"
#include "Windows.h"
#include "Exceptions.h"
#include "stdio.h"
#include "Clock.h"
#include "BatchRunners/Save.h"
#include "BatchRunners/Load.h"
```

Functions

- struct **Tableau** ∗∗ **GenerateTables** (int size, int count)

  Generate tables to an array in memory.
- void **SaveTableaus** (char ∗path, struct **Tableau** ∗∗arr, int n)

  Save an array of tableaus to files.
- struct **Tableau** ∗∗ **LoadTableaus** (char ∗path, int ∗n)

  Load all tables from a directory.

- void **PrintTables** (struct **Tableau** ∗∗tableaus, int n)

  Print all tables to standard output.
- char ∗ **AnalyzeTables** (char ∗imgPath, struct **Tableau** ∗∗tableaus, int n, int tableSize)

  Analyze all tables from an array and output a GNUPLOT graph.
- int **GetTableauSize** (struct **Tableau** ∗t)

  Get count of all items in a tableau.

4.17.1 Function Documentation

4.17.1.1 AnalyzeTables()

```
char * AnalyzeTables (
            char * imgPath,
            struct  Tableau ** tableaus,
            int n,
            int tableSize )
```

Analyze all tables from an array and output a GNUPLOT graph.

Parameters

imgPath	Path for saving the results image
tableaus	Array of tables
n	Count of tables
tableSize	Amount of items that existed in each table

Returns

Path to an image containing the images

4.17.1.2 GenerateTables()

```
struct  Tableau ** GenerateTables (
            int size,
            int count )
```

Generate tables to an array in memory.

Parameters

size	Amount of items in each table
count	Amount of tables

Returns

An array of generated tables

4.17.1.3 GetTableauSize()

```
int GetTableauSize (
            struct  Tableau * t )
```

Get count of all items in a tableau.

Parameters

t	A tableau to count items fof

Returns

Count of all items in the tableau

4.17.1.4 LoadTableaus()

```
struct  Tableau ** LoadTableaus (
            char * path,
            int * n )
```

Load all tables from a directory.

Parameters

path	Directory to load the tables from
n	RETURNS: count of items

Returns

An array of loaded tables

4.17.1.5 PrintTables()

```
void PrintTables (
            struct  Tableau ** tableaus,
            int n )
```

Print all tables to standard output.

Parameters

tableaus	Array of tables
n	Count of tables

4.17.1.6 SaveTableaus()

```
void SaveTableaus (
             char * path,
             struct  Tableau ** arr,
             int n )
```

Save an array of tableaus to files.

Parameters

path	Directory to save tableaus in
arr	Array of tableaus
n	Count of tables

4.18 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners.h File Reference

```
#include "../ArrayGen/TableauStructure.h"
```

Functions

- struct **Tableau** ∗∗ **GenerateTables** (int size, int count)
  Generate tables to an array in memory.
- char ∗ **AnalyzeTables** (char ∗imgPath, struct **Tableau** ∗∗tableaus, int n, int tableSize)
  Analyze all tables from an array and output a GNUPLOT graph.
- void **SaveTableaus** (char ∗path, struct **Tableau** ∗∗arr, int n)
  Save an array of tableaus to files.
- struct **Tableau** ∗∗ **LoadTableaus** (char ∗path, int ∗n)
  Load all tables from a directory.
- void **PrintTables** (struct **Tableau** ∗∗tableaus, int n)
  Print all tables to standard output.
- int **GetTableauSize** (struct **Tableau** ∗t)
  Get count of all items in a tableau.

4.18.1 Function Documentation

4.18.1.1 AnalyzeTables()

```
char * AnalyzeTables (
             char * imgPath,
             struct  Tableau ** tableaus,
             int n,
             int tableSize )
```

Analyze all tables from an array and output a GNUPLOT graph.

Parameters

imgPath	Path for saving the results image
tableaus	Array of tables
n	Count of tables
tableSize	Amount of items that existed in each table

Returns

Path to an image containing the images

4.18.1.2 GenerateTables()

```
struct  Tableau ** GenerateTables (
            int size,
            int count )
```

Generate tables to an array in memory.

Parameters

size	Amount of items in each table
count	Amount of tables

Returns

An array of generated tables

4.18.1.3 GetTableauSize()

```
int GetTableauSize (
            struct  Tableau * t )
```

Get count of all items in a tableau.

Parameters

t	A tableau to count items fof

Returns

Count of all items in the tableau

4.18.1.4 LoadTableaus()

```
struct  Tableau ** LoadTableaus (
            char * path,
            int * n )
```

Load all tables from a directory.

Parameters

path	Directory to load the tables from
n	RETURNS: count of items

Returns

An array of loaded tables

4.18.1.5 PrintTables()

```
void PrintTables (
             struct  Tableau ** tableaus,
             int n )
```

Print all tables to standard output.

Parameters

tableaus	Array of tables
n	Count of tables

4.18.1.6 SaveTableaus()

```
void SaveTableaus (
             char * path,
             struct  Tableau ** arr,
             int n )
```

Save an array of tableaus to files.

Parameters

path	Directory to save tableaus in
arr	Array of tableaus
n	Count of tables

4.19 BatchRunners.h

Go to the documentation of this file.

```
#pragma once
#include "../ArrayGen/TableauStructure.h"

struct Tableau** GenerateTables(int size, int count);

char* AnalyzeTables(char* imgPath, struct Tableau** tableaus, int n, int tableSize);
```

```

void SaveTableaus(char* path, struct Tableau** arr, int n);

struct Tableau** LoadTableaus(char* path, int* n);

void PrintTables(struct Tableau** tableaus, int n);

int GetTableauSize(struct Tableau* t);
```

4.20 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/↩ Analyze.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include "../../ArrayGen/TableauStructure.h"
#include "GenerateData.h"
#include "../MultithreadHelper.h"
#include "../../Graph/GraphItem.h"
#include "../../Graph/Graph.h"
#include "../Exceptions.h"
#include "../../ArrayAnalyze/ArrayAnalyze.h"
#include "Analyze.h"
```

Functions

- char ∗ **AnalyzeTablesMultiThreaded** (char ∗imgPath, struct **Tableau** ∗∗tableaus, int n, int tableSize)

  PRIVATE: relays table analysis to a multithreaded system.
- static void ∗ **AnalyzeTable_Thread** (void ∗input)

4.20.1 Function Documentation

4.20.1.1 AnalyzeTable_Thread()

```
static void * AnalyzeTable_Thread (
            void * input )  [static]
```

4.20.1.2 AnalyzeTablesMultiThreaded()

```
char * AnalyzeTablesMultiThreaded (
            char * imgPath,
            struct Tableau ** tableaus,
            int n,
            int tableSize )
```

PRIVATE: relays table analysis to a multithreaded system.

Parameters

imgPath	Path for saving the results image
tableaus	Array of tables
n	Count of tables
tableSize	Amount of items that existed in each table

Returns

Path to an image containing the images

4.21 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/↩ Analyze.h File Reference

Functions

- char ∗ **AnalyzeTablesMultiThreaded** (char ∗imgPath, struct **Tableau** ∗∗tableaus, int n, int tableSize)

  PRIVATE: relays table analysis to a multithreaded system.
- static void ∗ **AnalyzeTable_Thread** (void ∗input)

  PRIVATE: a function to be ran on every input item, for multithreaded process.

4.21.1 Function Documentation

4.21.1.1 AnalyzeTable_Thread()

```
static void * AnalyzeTable_Thread (
            void * input )  [static]
```

PRIVATE: a function to be ran on every input item, for multithreaded process.

Parameters

input	Pointer to a struct Tableau∗

Returns

Pointer to a new struct **GraphItem** (p. 6), filled with X and Y values.

4.21.1.2 AnalyzeTablesMultiThreaded()

```
char * AnalyzeTablesMultiThreaded (
            char * imgPath,
            struct  Tableau ** tableaus,
            int n,
            int tableSize )
```

PRIVATE: relays table analysis to a multithreaded system.

Parameters

imgPath	Path for saving the results image
tableaus	Array of tables
n	Count of tables
tableSize	Amount of items that existed in each table

Returns

Path to an image containing the images

4.22 Analyze.h

Go to the documentation of this file.

```
#pragma once

char* AnalyzeTablesMultiThreaded(char* imgPath, struct Tableau** tableaus, int n, int tableSize);

static void* AnalyzeTable_Thread(void* input);
```

4.23 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/↩ Generate.c File Reference

```
#include <stdlib.h>
#include "../../ArrayGen/TableauStructure.h"
#include "../../ArrayGen/TableauGen.h"
#include <stdbool.h>
#include "GenerateData.h"
#include "../MultithreadHelper.h"
#include "Generate.h"
```

Functions

- struct **Tableau** ∗∗ **GenerateTablesSingleThread** (int size, int count)

  PRIVATE: relays table analysis to a singlethreaded system.
- struct **Tableau** ∗∗ **GenerateTablesMultiThread** (int size, int count)

  PRIVATE: relays table analysis to a multithreaded system.
- static void ∗ **GenTable_Thread** (void ∗input)

4.23.1 Function Documentation

4.23.1.1 GenerateTablesMultiThread()

```
struct Tableau ** GenerateTablesMultiThread (
        int size,
        int count )
```

PRIVATE: relays table analysis to a multithreaded system.

Parameters

size	Amount of items in each table
count	Amount of tables

Returns

An array of generated tables

4.23.1.2 GenerateTablesSingleThread()

```
struct  Tableau ** GenerateTablesSingleThread (
             int size,
             int count )
```

PRIVATE: relays table analysis to a singlethreaded system.

Parameters

size	Amount of items in each table
count	Amount of tables

Returns

An array of generated tables

4.23.1.3 GenTable_Thread()

```
static void * GenTable_Thread (
             void * input )  [static]
```

4.24 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/↩ Generate.h File Reference

Functions

- struct **Tableau** ∗∗ **GenerateTablesSingleThread** (int size, int count)
  PRIVATE: relays table analysis to a singlethreaded system.
- struct **Tableau** ∗∗ **GenerateTablesMultiThread** (int size, int count)
  PRIVATE: relays table analysis to a multithreaded system.
- static void ∗ **GenTable_Thread** (void ∗input)
  PRIVATE: a function to be ran on every input item, for multithreaded process.

4.24.1 Function Documentation

4.24.1.1 GenerateTablesMultiThread()

```
struct  Tableau ** GenerateTablesMultiThread (
             int size,
             int count )
```

PRIVATE: relays table analysis to a multithreaded system.

Parameters

size	Amount of items in each table
count	Amount of tables

Returns

An array of generated tables

4.24.1.2 GenerateTablesSingleThread()

```
struct  Tableau ** GenerateTablesSingleThread (
             int size,
             int count )
```

PRIVATE: relays table analysis to a singlethreaded system.

Parameters

size	Amount of items in each table
count	Amount of tables

Returns

An array of generated tables

4.24.1.3 GenTable_Thread()

```
static void * GenTable_Thread (
             void * input )  [static]
```

PRIVATE: a function to be ran on every input item, for multithreaded process.

Parameters

input	A struct GenData*, which is a tuple(startingNum, size)

Returns

Pointer to a generated struct Tableau*

4.25 Generate.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00009 struct Tableau** GenerateTablesSingleThread(int size, int count);
```

```
struct Tableau** GenerateTablesMultiThread(int size, int count);

static void* GenTable_Thread(void* input);
```

4.26 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/↩ GenerateData.h File Reference

Classes

- struct **GenData**

  A tuple with information for generating each array.

4.27 GenerateData.h

Go to the documentation of this file.

```
#pragma once

struct GenData {
    float startingNum;

    int size;
};
```

4.28 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Load.c File Reference

```
#include <stdlib.h>
#include "../../Graph/GraphItem.h"
#include "../../ArrayGen/TableauStructure.h"
#include "../../ArrayGen/TableauGen.h"
#include <stdbool.h>
#include "../../ArrayAnalyze/ArrayAnalyze.h"
#include "Windows.h"
#include "../Exceptions.h"
#include "stdio.h"
#include "../Clock.h"
#include "../MultithreadHelper.h"
#include "Load.h"
```

Functions

- struct **Tableau** ∗∗ **LoadTableausSingleThread** (char ∗path, int ∗n)

  PRIVATE: relay saving to a single threaded system.
- struct **Tableau** ∗∗ **LoadTableausMultiThread** (char ∗path, int ∗n)

  PRIVATE: relay saving to a multi threaded system.
- static void ∗ **LoadTable_Thread** (void ∗input)

4.28.1 Function Documentation

4.28.1.1 LoadTable_Thread()

```
static void * LoadTable_Thread (
            void * input )  [static]
```

4.28.1.2 LoadTableausMultiThread()

```
struct  Tableau ** LoadTableausMultiThread (
             char * path,
             int * n )
```

PRIVATE: relay saving to a multi threaded system.

Parameters

path	Directory to load the tables from
n	RETURNS: count of items

Returns

An array of loaded tables

4.28.1.3 LoadTableausSingleThread()

```
struct  Tableau ** LoadTableausSingleThread (
             char * path,
             int * n )
```

PRIVATE: relay saving to a single threaded system.

Parameters

path	Directory to load the tables from
n	RETURNS: count of items

Returns

An array of loaded tables

4.29 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Load.h File Reference

```
#include <stdlib.h>
#include "../../Graph/GraphItem.h"
```

```
#include "../../ArrayGen/TableauStructure.h"
#include "../../ArrayGen/TableauGen.h"
#include <stdbool.h>
#include "../../ArrayAnalyze/ArrayAnalyze.h"
#include "Windows.h"
#include "../Exceptions.h"
#include "stdio.h"
#include "../Clock.h"
#include "SaveData.h"
```

Functions

- struct **Tableau** ∗∗ **LoadTableausSingleThread** (char ∗path, int ∗n)
  PRIVATE: relay saving to a single threaded system.
- struct **Tableau** ∗∗ **LoadTableausMultiThread** (char ∗path, int ∗n)
  PRIVATE: relay saving to a multi threaded system.
- static void ∗ **LoadTable_Thread** (void ∗input)
  PRIVATE: a function to be ran on every input item, for multithreaded process.

4.29.1 Function Documentation

4.29.1.1 LoadTable_Thread()

```
static void * LoadTable_Thread (
            void * input )  [static]
```

PRIVATE: a function to be ran on every input item, for multithreaded process.

Parameters

input	Pointer to a file path

Returns

Pointer to a loaded struct **Tableau** (p. 10)

4.29.1.2 LoadTableausMultiThread()

```
struct  Tableau ** LoadTableausMultiThread (
            char * path,
            int * n )
```

PRIVATE: relay saving to a multi threaded system.

Parameters

path	Directory to load the tables from
n	RETURNS: count of items

Returns

An array of loaded tables

4.29.1.3 LoadTableausSingleThread()

```
struct  Tableau ** LoadTableausSingleThread (
            char * path,
            int * n )
```

PRIVATE: relay saving to a single threaded system.

Parameters

path	Directory to load the tables from
n	RETURNS: count of items

Returns

An array of loaded tables

4.30 Load.h

Go to the documentation of this file.

```
#pragma once
#include <stdlib.h>
#include "../../Graph/GraphItem.h"
#include "../../ArrayGen/TableauStructure.h"
#include "../../ArrayGen/TableauGen.h"
#include <stdbool.h>
#include "../../ArrayAnalyze/ArrayAnalyze.h"
#include "Windows.h"
#include "../Exceptions.h"
#include "stdio.h"
#include "../Clock.h"
#include "SaveData.h"

struct Tableau** LoadTableausSingleThread(char* path, int* n);

struct Tableau** LoadTableausMultiThread(char* path, int* n);

static void* LoadTable_Thread(void* input);
```

4.31 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Save.c File Reference

```
#include <stdlib.h>
#include "../../Graph/GraphItem.h"
#include "../../ArrayGen/TableauStructure.h"
#include "../../ArrayGen/TableauGen.h"
#include <stdbool.h>
#include "../../ArrayAnalyze/ArrayAnalyze.h"
#include "Windows.h"
#include "../Exceptions.h"
#include "stdio.h"
```

```
#include "../Clock.h"
#include "SaveData.h"
#include "../MultithreadHelper.h"
#include "Save.h"
#include "Math.h"
```

Functions

- void **SaveTableausSingleThread** (char ∗path, struct **Tableau** ∗∗arr, int n)

  PRIVATE: relay saving to a single threaded system.
- void **SaveTableausMultiThreaded** (char ∗path, struct **Tableau** ∗∗arr, int n)

  PRIVATE: relays table saving to a multithreaded system.
- static void ∗ **SaveTable_Thread** (void ∗input)

4.31.1 Function Documentation

4.31.1.1 SaveTable_Thread()

```
static void * SaveTable_Thread (
             void * input )  [static]
```

4.31.1.2 SaveTableausMultiThreaded()

```
void SaveTableausMultiThreaded (
             char * path,
             struct  Tableau ** arr,
             int n )
```

PRIVATE: relays table saving to a multithreaded system.

Parameters

path	Directory to save tableaus in
arr	Array of tableaus
n	Count of tables

4.31.1.3 SaveTableausSingleThread()

```
void SaveTableausSingleThread (
             char * path,
             struct  Tableau ** arr,
             int n )
```

PRIVATE: relay saving to a single threaded system.

Parameters

path	Directory to save tableaus in
arr	Array of tableaus
n	Count of tables

4.32 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Save.h File Reference

```
#include <stdlib.h>
#include "../../Graph/GraphItem.h"
#include "../../ArrayGen/TableauStructure.h"
#include "../../ArrayGen/TableauGen.h"
#include <stdbool.h>
#include "../../ArrayAnalyze/ArrayAnalyze.h"
#include "Windows.h"
#include "../Exceptions.h"
#include "stdio.h"
#include "../Clock.h"
#include "SaveData.h"
```

Functions

- void **SaveTableausSingleThread** (char ∗path, struct **Tableau** ∗∗arr, int n)

  PRIVATE: relay saving to a single threaded system.
- void **SaveTableausMultiThreaded** (char ∗path, struct **Tableau** ∗∗arr, int n)

  PRIVATE: relays table saving to a multithreaded system.
- static void ∗ **SaveTable_Thread** (void ∗input)

  PRIVATE: a function to be ran on every input item, for multithreaded process.

4.32.1 Function Documentation

4.32.1.1 SaveTable_Thread()

```
static void * SaveTable_Thread (
            void * input )  [static]
```

PRIVATE: a function to be ran on every input item, for multithreaded process.

Parameters

input	Pointer to a struct SaveData∗

Returns

Nothing (always returns NULL)

4.32.1.2 SaveTableausMultiThreaded()

```
void SaveTableausMultiThreaded (
            char * path,
            struct  Tableau ** arr,
            int n )
```

PRIVATE: relays table saving to a multithreaded system.

Parameters

path	Directory to save tableaus in
arr	Array of tableaus
n	Count of tables

4.32.1.3 SaveTableausSingleThread()

```
void SaveTableausSingleThread (
            char * path,
            struct  Tableau ** arr,
            int n )
```

PRIVATE: relay saving to a single threaded system.

Parameters

path	Directory to save tableaus in
arr	Array of tableaus
n	Count of tables

4.33 Save.h

Go to the documentation of this file.

```
#pragma once
#include <stdlib.h>
#include "../../Graph/GraphItem.h"
#include "../../ArrayGen/TableauStructure.h"
#include "../../ArrayGen/TableauGen.h"
#include <stdbool.h>
#include "../../ArrayAnalyze/ArrayAnalyze.h"
#include "Windows.h"
#include "../Exceptions.h"
#include "stdio.h"
#include "../Clock.h"
#include "SaveData.h"

void SaveTableausSingleThread(char* path, struct Tableau** arr, int n);

void SaveTableausMultiThreaded(char* path, struct Tableau** arr, int n);

static void* SaveTable_Thread(void* input);
```

4.34 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Save↩ Data.h File Reference

```
#include "../../ArrayGen/TableauStructure.h"
```

Classes

- struct **SaveData**

  A tuple with information for saving each array.

4.35 SaveData.h

Go to the documentation of this file.

```
#pragma once
#include "../../ArrayGen/TableauStructure.h"

struct SaveData {
    struct Tableau* tableau;

    char* basePath;

    int index;

    int digitsCount;
};
```

4.36 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Clock.c File Reference

```
#include <sys/timeb.h>
```

Functions

- long **GetCurrTimeMs** ()

  Get current system time in miliseconds.

4.36.1 Function Documentation

4.36.1.1 GetCurrTimeMs()

```
long GetCurrTimeMs ( )
```

Get current system time in miliseconds.

Returns

Current system time in miliseconds.

4.37 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Clock.h File Reference

Functions

- long **GetCurrTimeMs** ()

  Get current system time in miliseconds.

4.37.1 Function Documentation

4.37.1.1 GetCurrTimeMs()

```
long GetCurrTimeMs ( )
```

Get current system time in miliseconds.

Returns

Current system time in miliseconds.

4.38 Clock.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00007 long GetCurrTimeMs();
```

4.39 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Exceptions.c File Reference

```
#include "Exceptions.h"
#include <stdio.h>
```

Functions

- void **ChangeConsoleColor** (enum **LogType** type)

  Change color of the standard output.
- void **Log** (const char ∗msg, enum **LogType** type, int line, const char ∗file)

  Log a message to standard output.

4.39.1 Function Documentation

4.39.1.1 ChangeConsoleColor()

```
void ChangeConsoleColor (
            enum LogType type )
```

Change color of the standard output.

Parameters

type	Type of log to take the color from

4.39.1.2 Log()

```
void Log (
            const char * msg,
            enum  LogType type,
            int line,
            const char * file )
```

Log a message to standard output.

Parameters

msg	Message
type	Type of message
line	Source line of code where the log was thrown
file	Path to source file where the log was thrown

4.40 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Exceptions.h File Reference

```
#include <windows.h>
```

Macros

- #define **LOG**(msg) **Log**(msg, **LOGTYPE_OK**, __LINE__, __FILE__)
  Log a standard message to standard output.
- #define **LOG_ERROR**(msg) **Log**(msg, **LOGTYPE_ERROR**, __LINE__, __FILE__)
  Log a critical error to standard output. It will be shown in red.
- #define **LOG_WARNING**(msg) **Log**(msg, **LOGTYPE_WARNING**, __LINE__, __FILE__)
  Log a non-critical warning to standard output. It will be shown in orange.

Enumerations

- enum **LogType** { **LOGTYPE_OK** , **LOGTYPE_WARNING** , **LOGTYPE_ERROR** }
  Character of the log call.

Functions

- void **ChangeConsoleColor** (enum **LogType** type)
  Change color of the standard output.
- void **Log** (const char ∗msg, enum **LogType** type, int line, const char ∗file)
  Log a message to standard output.

4.40.1 Macro Definition Documentation

4.40.1.1 LOG

```
#define LOG(
              msg )  Log(msg,  LOGTYPE_OK, __LINE__, __FILE__)
```

Log a standard message to standard output.

Parameters

msg	Message

4.40.1.2 LOG_ERROR

```
#define LOG_ERROR(
              msg )  Log(msg,  LOGTYPE_ERROR, __LINE__, __FILE__)
```

Log a critical error to standard output. It will be shown in red.

Parameters

msg	Message

4.40.1.3 LOG_WARNING

```
#define LOG_WARNING(
              msg )  Log(msg,  LOGTYPE_WARNING, __LINE__, __FILE__)
```

Log a non-critical warning to standard output. It will be shown in orange.

Parameters

msg	Message

4.40.2 Enumeration Type Documentation

4.40.2.1 LogType

```
enum  LogType
```

Character of the log call.

Enumerator

LOGTYPE_OK	LogType: normal info.
LOGTYPE_WARNING	LogType: non-critical warning. Shows in orange.
LOGTYPE_ERROR	LogType: critical error. Shows in red.

4.40.3 Function Documentation

4.40.3.1 ChangeConsoleColor()

```
void ChangeConsoleColor (
             enum  LogType type )
```

Change color of the standard output.

Parameters

type	Type of log to take the color from

4.40.3.2 Log()

```
void Log (
             const char * msg,
             enum  LogType type,
             int line,
             const char * file )
```

Log a message to standard output.

Parameters

msg	Message
type	Type of message
line	Source line of code where the log was thrown
file	Path to source file where the log was thrown

4.41 Exceptions.h

Go to the documentation of this file.

```
#pragma once
#include <windows.h>

enum LogType {
    LOGTYPE_OK,

    LOGTYPE_WARNING,

    LOGTYPE_ERROR
};

void ChangeConsoleColor(enum LogType type);

void Log(const char* msg, enum LogType type, int line, const char* file);

#define LOG(msg) Log(msg, LOGTYPE_OK, __LINE__, __FILE__)

#define LOG_ERROR(msg) Log(msg, LOGTYPE_ERROR, __LINE__, __FILE__)

#define LOG_WARNING(msg) Log(msg, LOGTYPE_WARNING, __LINE__, __FILE__)
```

4.42 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/MultithreadHelper.c File Reference

```
#include <math.h>
#include "MultithreadHelper.h"
#include <stdbool.h>
#include <Windows.h>
#include "Exceptions.h"
#include "Clock.h"
```

Functions

- int **GetCoresCount** ()

  PRIVATE FUNCTION: Get amount of logical cores in the system.
- void **RunBatch** (void ∗(∗func)(void ∗), void ∗∗inputArray, void ∗∗outputArray, int n)

  Run a given function on a data set with automatical split to threads.
- int **RunBatchThread** (struct **ThreadArgs** ∗args)

  PRIVATE FUNCTION: Main function for each thread.
- HANDLE ∗ **RunProgressThread** (int ∗∗progressArray, int progressCount, int MaxProgressSum, bool ∗b↩ Finished)

  Spawn a progress thread, which will periodically count and print current progress of all workers to standard output.
- int **UpdateProgress** (struct **ProgressArgs** ∗args)

  PRIVATE: Main function for progress counter thread.

4.42.1 Function Documentation

4.42.1.1 GetCoresCount()

```
int GetCoresCount ( )
```

PRIVATE FUNCTION: Get amount of logical cores in the system.

Returns

Amount of logical cores in the system

4.42.1.2 RunBatch()

```
void RunBatch (
            void *(*)(void *) func,
            void ** inputArray,
            void ** outputArray,
            int n )
```

Run a given function on a data set with automatical split to threads.

Parameters

func	Function to call on each data item
inputArray	Input array. Elements from this array will be passed as arg to the func. Set to NULL to skip.
outputArray	Output array. Return value from the func will be written to this array. Set to NULL to skip.
n	Amount of items in array

4.42.1.3 RunBatchThread()

```
int RunBatchThread (
            struct  ThreadArgs * args )
```

PRIVATE FUNCTION: Main function for each thread.

Parameters

args	Thread arguments

Returns

Return code of the thread

4.42.1.4 RunProgressThread()

```
HANDLE * RunProgressThread (
            int ** progressArray,
            int progressCount,
            int MaxProgressSum,
            bool * bFinished )
```

Spawn a progress thread, which will periodically count and print current progress of all workers to standard output.

Parameters

progressArray	An array of pointers to progresses' of each worker
progressCount	Amount of workers in prev array
MaxProgressSum	Expected sum of all progresses
bFinished	Pointer to a bool which will change once the task is finished and monitoring should be ceased

Returns

Handle to spawned thread

4.42.1.5 UpdateProgress()

```
int UpdateProgress (
            struct  ProgressArgs * args )
```

PRIVATE: Main function for progress counter thread.

Parameters

progressArray	An array with arguments for the thread

Returns

Return code of the thread

4.43 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/MultithreadHelper.h File Reference

```
#include <stdbool.h>
#include <windows.h>
```

Classes

- struct **ThreadArgs**

  Arguments to pass to a worker thread for main threading.
- struct **ProgressArgs**

  Arguments to pass to a progress counter thread.

Functions

- int **GetCoresCount** ()

  PRIVATE FUNCTION: Get amount of logical cores in the system.
- void **RunBatch** (void ∗(∗func)(void ∗), void ∗∗inputArray, void ∗∗outputArray, int n)

  Run a given function on a data set with automatical split to threads.
- int **RunBatchThread** (struct **ThreadArgs** ∗args)

  PRIVATE FUNCTION: Main function for each thread.
- HANDLE ∗ **RunProgressThread** (int ∗∗progressArray, int progressCount, int MaxProgressSum, bool ∗b↩ Finished)

  Spawn a progress thread, which will periodically count and print current progress of all workers to standard output.
- int **UpdateProgress** (struct **ProgressArgs** ∗args)

  PRIVATE: Main function for progress counter thread.

4.43.1 Function Documentation

4.43.1.1 GetCoresCount()

```
int GetCoresCount ( )
```

PRIVATE FUNCTION: Get amount of logical cores in the system.

Returns

Amount of logical cores in the system

4.43.1.2 RunBatch()

```
void RunBatch (
            void *(*)(void *) func,
            void ** inputArray,
            void ** outputArray,
            int n )
```

Run a given function on a data set with automatical split to threads.

Parameters

func	Function to call on each data item
inputArray	Input array. Elements from this array will be passed as arg to the func. Set to NULL to skip.
outputArray	Output array. Return value from the func will be written to this array. Set to NULL to skip.
n	Amount of items in array

4.43.1.3 RunBatchThread()

```
int RunBatchThread (
            struct ThreadArgs * args )
```

PRIVATE FUNCTION: Main function for each thread.

Parameters

args	Thread arguments

Returns

Return code of the thread

4.43.1.4 RunProgressThread()

```
HANDLE * RunProgressThread (
            int ** progressArray,
            int progressCount,
            int MaxProgressSum,
            bool * bFinished )
```

Spawn a progress thread, which will periodically count and print current progress of all workers to standard output.

Parameters

progressArray	An array of pointers to progresses' of each worker
progressCount	Amount of workers in prev array
MaxProgressSum	Expected sum of all progresses
bFinished	Pointer to a bool which will change once the task is finished and monitoring should be ceased

Returns

Handle to spawned thread

4.43.1.5 UpdateProgress()

```
int UpdateProgress (
            struct  ProgressArgs * args )
```

PRIVATE: Main function for progress counter thread.

Parameters

progressArray	An array with arguments for the thread

Returns

Return code of the thread

4.44 MultithreadHelper.h

Go to the documentation of this file.

```
#pragma once
#include <stdbool.h>
#include <windows.h>

struct ThreadArgs {
    void* (*func)(void*);

    void** inputArray;

    void** outputArray;

    int start;

    int end;

    int* progress;
};

struct ProgressArgs {
    int** progressArray;

    int MaxProgressSum;

    int progressEntriesCount;

    bool* ShouldCancel;
};

int GetCoresCount();

void RunBatch(void* (*func)(void*), void** inputArray, void** outputArray, int n);

int RunBatchThread(struct ThreadArgs* args);

HANDLE* RunProgressThread(int** progressArray, int progressCount, int MaxProgressSum, bool*
      bFinished);

int UpdateProgress(struct ProgressArgs* args);
```

4.45 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ProjectRequirements.h File Reference

4.46 ProjectRequirements.h

Go to the documentation of this file.

```
#pragma once

// A switch between optimized (with real life sense) andquestionable
// function implementations, as the latter were forced by some of uni’s project requirements
// CURRENTLY AFFECTS:
// ArrayAnalyze/ArrayAnalyze.c -> SolveTableau (in-place vs recursion)
// ArrayGen/TableauStructure.c -> FindThe2ndMaxElement (linear search vs qsort)

// #define UNOPTIMAL_PROJECT_REQUIREMENTS
```

4.47 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/UserInput.c File Reference

```
#include <stdbool.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "UserInputStruct.h"
#include "UserInput.h"
#include "Exceptions.h"
#include "ProjectRequirements.h"
```

Functions

- bool **TakeUserInput** (struct **UserInput** ∗returnInput, int argc, char ∗argv[ ])

  Obtain input from the user; automatically either reads content of args list, or asks the user for them if they are missing.
- bool **ReadUserInputFromArgs** (struct **UserInput** ∗returnInput, int argc, char ∗argv[ ])

  Obtain user input from provided command line args.
- bool **ValidateUserInput** (struct **UserInput** ∗input)

  Validivies user input.
- bool **ReadUserInputFromPrompts** (struct **UserInput** ∗returnInput)

  Obtain input by interactively ask the user to provide it.
- bool **ShouldUseExistingTables** (struct **UserInput** input)

  True if a path to existing tables which should be used was passed.
- void **DrawUsage** (void)

  Print explanations of command line args to standard output.
- void **DrawVersion** (void)

  Print version info to standard output.

4.47.1 Function Documentation

4.47.1.1 DrawUsage()

```
void DrawUsage (
            void  )
```

Print explanations of command line args to standard output.

4.47.1.2 DrawVersion()

```
void DrawVersion (
             void  )
```

Print version info to standard output.

4.47.1.3 ReadUserInputFromArgs()

```
bool ReadUserInputFromArgs (
             struct  UserInput * returnInput,
             int argc,
             char * argv[] )
```

Obtain user input from provided command line args.

Parameters

returnInput	A pointer to returnInput where the filled input will be put
argc	Command line args count
argv	Command line args values

Returns

Whether the input was taken sucessfully

4.47.1.4 ReadUserInputFromPrompts()

```
bool ReadUserInputFromPrompts (
             struct  UserInput * returnInput )
```

Obtain input by interactively ask the user to provide it.

Parameters

returnInput	A pointer to returnInput where the filled input will be put

Returns

Whether the input was taken sucessfully

4.47.1.5 ShouldUseExistingTables()

```
bool ShouldUseExistingTables (
             struct  UserInput input )
```

True if a path to existing tables which should be used was passed.

Parameters

input	User input

Returns

4.47.1.6 TakeUserInput()

```
bool TakeUserInput (
            struct  UserInput * returnInput,
            int argc,
            char * argv[] )
```

Obtain input from the user; automatically either reads content of args list, or asks the user for them if they are missing.

Parameters

returnInput	A pointer to returnInput where the filled input will be put
argc	Command line args count
argv	Command line args values

Returns

Whether the input was taken sucessfully

4.47.1.7 ValidateUserInput()

```
bool ValidateUserInput (
            struct  UserInput * input )
```

Validivies user input.

Parameters

input	User input

Returns

Whether the user input is valid

4.48 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/UserInput.h File Reference

```
#include <stdbool.h>
```

```
#include "UserInputStruct.h"
```

Functions

- bool **TakeUserInput** (struct **UserInput** ∗returnInput, int argc, char ∗argv[ ])

  Obtain input from the user; automatically either reads content of args list, or asks the user for them if they are missing.
- bool **ReadUserInputFromArgs** (struct **UserInput** ∗returnInput, int argc, char ∗argv[ ])

  Obtain user input from provided command line args.
- bool **ValidateUserInput** (struct **UserInput** ∗input)

  Validivies user input.
- bool **ReadUserInputFromPrompts** (struct **UserInput** ∗returnInput)

  Obtain input by interactively ask the user to provide it.
- bool **ShouldUseExistingTables** (struct **UserInput** input)

  True if a path to existing tables which should be used was passed.
- void **DrawUsage** (void)

  Print explanations of command line args to standard output.
- void **DrawVersion** (void)

  Print version info to standard output.

4.48.1 Function Documentation

4.48.1.1 DrawUsage()

```
void DrawUsage (
             void  )
```

Print explanations of command line args to standard output.

4.48.1.2 DrawVersion()

```
void DrawVersion (
             void  )
```

Print version info to standard output.

4.48.1.3 ReadUserInputFromArgs()

```
bool ReadUserInputFromArgs (
             struct  UserInput * returnInput,
             int argc,
             char * argv[] )
```

Obtain user input from provided command line args.

Parameters

returnInput	A pointer to returnInput where the filled input will be put
argc	Command line args count
argv	Command line args values

Returns

Whether the input was taken sucessfully

4.48.1.4 ReadUserInputFromPrompts()

```
bool ReadUserInputFromPrompts (
            struct  UserInput * returnInput )
```

Obtain input by interactively ask the user to provide it.

Parameters

returnInput	A pointer to returnInput where the filled input will be put

Returns

Whether the input was taken sucessfully

4.48.1.5 ShouldUseExistingTables()

```
bool ShouldUseExistingTables (
            struct  UserInput input )
```

True if a path to existing tables which should be used was passed.

Parameters

input	User input

Returns

4.48.1.6 TakeUserInput()

```
bool TakeUserInput (
            struct  UserInput * returnInput,
            int argc,
            char * argv[] )
```

Obtain input from the user; automatically either reads content of args list, or asks the user for them if they are missing.

Parameters

returnInput	A pointer to returnInput where the filled input will be put
argc	Command line args count
argv	Command line args values

Returns

Whether the input was taken sucessfully

4.48.1.7 ValidateUserInput()

```
bool ValidateUserInput (
            struct  UserInput * input )
```

Validivies user input.

Parameters

input	User input

Returns

Whether the user input is valid

4.49 UserInput.h

Go to the documentation of this file.

```
#pragma once
#include <stdbool.h>
#include "UserInputStruct.h"

bool TakeUserInput(struct UserInput* returnInput, int argc, char* argv[]);

bool ReadUserInputFromArgs(struct UserInput* returnInput, int argc, char* argv[]);

bool ValidateUserInput(struct UserInput* input);

bool ReadUserInputFromPrompts(struct UserInput* returnInput);

bool ShouldUseExistingTables(struct UserInput input);

void DrawUsage(void);

void DrawVersion(void);
```

4.50 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/UserInputStruct.h File Reference

```
#include <stdbool.h>
```

Classes

- struct **UserInput**

  Provides user's input data necessary to generate tables.

4.51 UserInputStruct.h

Go to the documentation of this file.

```
#pragma once
#include <stdbool.h>

struct UserInput {
    int TableauSize;

    int TableauCount;

    char* InputPath;

    char* TablesOutputPath;

    char* ImgOutputPath;

    bool bPrintTables;
};
```

4.52 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Version.h File Reference

Macros

- #define **VERSION** "v1.0"
- #define **MAGIC** "TAB"

4.52.1 Macro Definition Documentation

4.52.1.1 MAGIC

```
#define MAGIC "TAB"
```

4.52.1.2 VERSION

```
#define VERSION "v1.0"
```

4.53 Version.h

Go to the documentation of this file.

```
#define VERSION "v1.0"
#define MAGIC "TAB"
```

4.54 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/JeuDeTaquin.c File Reference

```
#include "JeuDeTaquin.h"
#include "Helpers/UserInputStruct.h"
#include "Helpers/UserInput.h"
#include "Helpers/BatchRunners.h"
#include <stdlib.h>
#include <stdio.h>
```

Macros

- #define **MULTITHREAD**

Functions

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

4.54.1 Macro Definition Documentation

4.54.1.1 MULTITHREAD

```
#define MULTITHREAD
```

4.54.2 Function Documentation

4.54.2.1 main()

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

4.55 S:/Uni/C/JeuDeTaquin/JeuDeTaquin/JeuDeTaquin.h File Reference

4.56 JeuDeTaquin.h

Go to the documentation of this file.

```
// JeuDeTaquin.h : Include file for standard system include files,
// or project specific include files.

#pragma once
```

Index

Analyze.c
    AnalyzeTable_Thread, 43
    AnalyzeTablesMultiThreaded, 43
Analyze.h
    AnalyzeTable_Thread, 44
    AnalyzeTablesMultiThreaded, 44
AnalyzeTable_Thread
    Analyze.c, 43
    Analyze.h, 44
AnalyzeTables
    BatchRunners.c, 38
    BatchRunners.h, 40
AnalyzeTablesMultiThreaded
    Analyze.c, 43
    Analyze.h, 44
ArrayAnalyze.c
    DOUBLE_DIGITS, 15
    LoadTableauFromFile, 16
    MAGIC, 15
    SolveTableau, 16
ArrayAnalyze.h
    LoadTableauFromFile, 17
    SolveTableau, 17
Avg
    GraphItem, 6

basePath
    SaveData, 9
BatchRunners.c
    AnalyzeTables, 38
    GenerateTables, 38
    GetTableauSize, 38
    LoadTableaus, 39
    PrintTables, 39
    SaveTableaus, 39
BatchRunners.h
    AnalyzeTables, 40
    GenerateTables, 41
    GetTableauSize, 41
    LoadTableaus, 41
    PrintTables, 42
    SaveTableaus, 42
bPrintTables
    UserInput, 13

CalculateDelta
    TableauStructure.c, 23
    TableauStructure.h, 26
ChangeConsoleColor
    Exceptions.c, 56
    Exceptions.h, 59
CHECK_BOUNDS2
    Graph.c, 30
Clock.c
    GetCurrTimeMs, 55
Clock.h
    GetCurrTimeMs, 56
Compaper
    Graph.c, 31
    Graph.h, 34
currRange
    GraphItem, 6
currSum
    GraphItem, 6

DIGITS_OF_STARTING_NUMBERS
    TableauGen.c, 18
digitsCount
    SaveData, 9
DOUBLE_DIGITS
    ArrayAnalyze.c, 15
DrawUsage
    UserInput.c, 66
    UserInput.h, 69
DrawVersion
    UserInput.c, 66
    UserInput.h, 69

end
    ThreadArgs, 12
Exceptions.c
    ChangeConsoleColor, 56
    Log, 57
Exceptions.h
    ChangeConsoleColor, 59
    LOG, 58
    Log, 59
    LOG_ERROR, 58
    LOG_WARNING, 58
    LogType, 58
    LOGTYPE_ERROR, 58
    LOGTYPE_OK, 58
    LOGTYPE_WARNING, 58

FindThe2ndMaxElement
    TableauStructure.c, 23
    TableauStructure.h, 27
func
    ThreadArgs, 12

GenData, 5

size, 5
startingNum, 5
Generate.c
GenerateTablesMultiThread, 45
GenerateTablesSingleThread, 46
GenTable_Thread, 46
Generate.h
GenerateTablesMultiThread, 46
GenerateTablesSingleThread, 47
GenTable_Thread, 47
GenerateDB
Graph.c, 31
Graph.h, 34
GenerateGraph
Graph.c, 31
Graph.h, 35
GenerateRandomSet
TableauStructure.c, 23
TableauStructure.h, 27
GenerateStartingNumbers
TableauStructure.c, 24
TableauStructure.h, 27
GenerateTableau
TableauGen.c, 19
TableauGen.h, 21
GenerateTables
BatchRunners.c, 38
BatchRunners.h, 41
GenerateTablesMultiThread
Generate.c, 45
Generate.h, 46
GenerateTablesSingleThread
Generate.c, 46
Generate.h, 47
GenTable_Thread
Generate.c, 46
Generate.h, 47
GetCoresCount
MultithreadHelper.c, 60
MultithreadHelper.h, 63
GetCurrTimeMs
Clock.c, 55
Clock.h, 56
GetTableauSize
BatchRunners.c, 38
BatchRunners.h, 41
Graph.c
CHECK_BOUNDS2, 30
Compaper, 31
GenerateDB, 31
GenerateGraph, 31
IsSorted, 32
OPTIMIZED_AVG, 30
SetAverages, 32
SortIfNotSorted, 32
Graph.h
Compaper, 34
GenerateDB, 34
GenerateGraph, 35
IsSorted, 35
SetAverages, 36
SortIfNotSorted, 36
GraphItem, 6
Avg, 6
currRange, 6
currSum, 6
X, 6
Y, 7

ImgOutputPath
UserInput, 13
index
SaveData, 9
inputArray
ThreadArgs, 12
InputPath
UserInput, 14
IsSorted
Graph.c, 32
Graph.h, 35

JeuDeTaquin.c
main, 73
MULTITHREAD, 73

Load.c
LoadTable_Thread, 49
LoadTableausMultiThread, 49
LoadTableausSingleThread, 49
Load.h
LoadTable_Thread, 50
LoadTableausMultiThread, 50
LoadTableausSingleThread, 51
LoadTable_Thread
Load.c, 49
Load.h, 50
LoadTableauFromFile
ArrayAnalyze.c, 16
ArrayAnalyze.h, 17
LoadTableaus
BatchRunners.c, 39
BatchRunners.h, 41
LoadTableausMultiThread
Load.c, 49
Load.h, 50
LoadTableausSingleThread
Load.c, 49
Load.h, 51
LOG
Exceptions.h, 58
Log
Exceptions.c, 57
Exceptions.h, 59
LOG_ERROR
Exceptions.h, 58
LOG_WARNING
Exceptions.h, 58

LogType
    Exceptions.h, 58
LOGTYPE_ERROR
    Exceptions.h, 58
LOGTYPE_OK
    Exceptions.h, 58
LOGTYPE_WARNING
    Exceptions.h, 58

MAGIC
    ArrayAnalyze.c, 15
    Version.h, 72
main
    JeuDeTaquin.c, 73
MAXDIGITS
    TableauGen.c, 18
MaxProgressSum
    ProgressArgs, 7
MULTITHREAD
    JeuDeTaquin.c, 73
MultithreadHelper.c
    GetCoresCount, 60
    RunBatch, 60
    RunBatchThread, 61
    RunProgressThread, 61
    UpdateProgress, 61
MultithreadHelper.h
    GetCoresCount, 63
    RunBatch, 63
    RunBatchThread, 64
    RunProgressThread, 64
    UpdateProgress, 65

numberOfRows
    Tableau, 11

OPTIMIZED_AVG
    Graph.c, 30
outputArray
    ThreadArgs, 12

PrintRow
    TableauGen.c, 19
    TableauGen.h, 21
PrintTableau
    TableauGen.c, 19
    TableauGen.h, 21
PrintTables
    BatchRunners.c, 39
    BatchRunners.h, 42
progress
    ThreadArgs, 12
ProgressArgs, 7
    MaxProgressSum, 7
    progressArray, 7
    progressEntriesCount, 8
    ShouldCancel, 8
progressArray
    ProgressArgs, 7
progressEntriesCount
    ProgressArgs, 8

RandomSet, 8
    set, 8
    setSize, 8
ReadUserInputFromArgs
    UserInput.c, 67
    UserInput.h, 69
ReadUserInputFromPrompts
    UserInput.c, 67
    UserInput.h, 70
ResizeRow
    TableauStructure.c, 24
    TableauStructure.h, 28
ResizeSizesArray
    TableauStructure.c, 24
    TableauStructure.h, 28
ResizeTableau
    TableauStructure.c, 25
    TableauStructure.h, 28
RunBatch
    MultithreadHelper.c, 60
    MultithreadHelper.h, 63
RunBatchThread
    MultithreadHelper.c, 61
    MultithreadHelper.h, 64
RunProgressThread
    MultithreadHelper.c, 61
    MultithreadHelper.h, 64

S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayAnalyze/ArrayAnalyze.c, 15
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayAnalyze/ArrayAnalyze.h, 16, 17
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/RandomSetStruct.h, 17, 18
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauGen.c, 18
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauGen.h, 20, 22
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauStructure.c, 22
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/ArrayGen/TableauStructure.h, 26, 29
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/Graph.c, 30
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/Graph.h, 34, 36
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Graph/GraphItem.h, 37
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners.c, 37
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners.h, 40, 42
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Analyze.c, 43
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Analyze.h, 44, 45

S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/GenerateData.c,
45
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/GenerateData.h,
46, 47
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/GenerateDataMulti.c,
48
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Load.c,
48
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Load.h,
49, 51
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Save.c,
51
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/Save.h,
53, 54
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/BatchRunners/SaveData.h,
55
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Clock.c,
55
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Clock.h,
56
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Exceptions.c,
56
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Exceptions.h,
57, 59
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/MultithreadHelpers.c,
60
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/MultithreadHelpers.h,
63, 65
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/ProjectRequirements.h,
66
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/UserInput.c,
66
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/UserInput.h,
68, 71
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/UserInputStruct.h,
71, 72
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/Helpers/Version.h,
72
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/JeuDeTaquin.c, 72
S:/Uni/C/JeuDeTaquin/JeuDeTaquin/JeuDeTaquin.h, 73
Save.c
    SaveTable_Thread, 52
    SaveTableausMultiThreaded, 52
    SaveTableausSingleThread, 52
Save.h
    SaveTable_Thread, 53
    SaveTableausMultiThreaded, 53
    SaveTableausSingleThread, 54
SaveData, 9
    basePath, 9
    digitsCount, 9
    index, 9
    tableau, 10
SaveTable_Thread
    Save.c, 52
    Save.h, 53
SaveTableau
    TableauGen.c, 19
    TableauGen.h, 21
SaveTableaus
    BatchRunners.c, 39
    BatchRunners.h, 42
SaveTableausMultiThreaded
    Save.c, 52
    Save.h, 53
SaveTableausSingleThread
    Save.c, 52
    Save.h, 54
seed
    RandomSet, 8
setAverages
    Graph.c, 32
    Graph.h, 36
setSize
    RandomSet, 8
ShouldCancel
    ProgressArgs, 8
ShouldUseExistingTables
    UserInput.c, 67
    UserInput.h, 70
size
    GenData, 5
sizesOfRows
    Tableau, 11
SolveTableau
    ArrayAnalyze.c, 16
    ArrayAnalyze.h, 17
SortIfNotSorted
    Graph.c, 32
    Graph.h, 36
start
    ThreadArgs, 13
startingNr
    Tableau, 11
startingNum
    GenData, 5

Tableau, 10
    numberOfRows, 11
    sizesOfRows, 11
    startingNr, 11
    tableau, 11
tableau
    SaveData, 10
    Tableau, 11
TableauCount
    UserInput, 14
TableauGen.c
    DIGITS_OF_STARTING_NUMBERS, 18
    GenerateTableau, 19
    MAXDIGITS, 18
    PrintRow, 19
    PrintTableau, 19
    SaveTableau, 19
TableauGen.h
    GenerateTableau, 21
    PrintRow, 21

PrintTableau, 21
SaveTableau, 21
TableauSize
UserInput, 14
TableauStructure.c
CalculateDelta, 23
FindThe2ndMaxElement, 23
GenerateRandomSet, 23
GenerateStartingNumbers, 24
ResizeRow, 24
ResizeSizesArray, 24
ResizeTableau, 25
ThrowElementToRow, 25
TableauStructure.h
CalculateDelta, 26
FindThe2ndMaxElement, 27
GenerateRandomSet, 27
GenerateStartingNumbers, 27
ResizeRow, 28
ResizeSizesArray, 28
ResizeTableau, 28
ThrowElementToRow, 29
TablesOutputPath
UserInput, 14
TakeUserInput
UserInput.c, 68
UserInput.h, 70
ThreadArgs, 11
end, 12
func, 12
inputArray, 12
outputArray, 12
progress, 12
start, 13
ThrowElementToRow
TableauStructure.c, 25
TableauStructure.h, 29

UpdateProgress
MultithreadHelper.c, 61
MultithreadHelper.h, 65
UserInput, 13
bPrintTables, 13
ImgOutputPath, 13
InputPath, 14
TableauCount, 14
TableauSize, 14
TablesOutputPath, 14
UserInput.c
DrawUsage, 66
DrawVersion, 66
ReadUserInputFromArgs, 67
ReadUserInputFromPrompts, 67
ShouldUseExistingTables, 67
TakeUserInput, 68
ValidateUserInput, 68
UserInput.h
DrawUsage, 69
DrawVersion, 69
ReadUserInputFromArgs, 69
ReadUserInputFromPrompts, 70
ShouldUseExistingTables, 70
TakeUserInput, 70
ValidateUserInput, 71

ValidateUserInput
UserInput.c, 68
UserInput.h, 71
VERSION
Version.h, 72
Version.h
MAGIC, 72
VERSION, 72

X
GraphItem, 6

Y
GraphItem, 7