CSVTableProject

Generated by Doxygen 1.11.0

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 Command Class Reference

Represents a user command.

`#include <Command.h>`

Public Member Functions

- **Command** ()
  Creates an empty command.
- void parse (string input)
  Cuts a line of user input into arguments and updates the arguments array.
- int getArgCount () const
  Returns the number of arguments stored.
- string & operator[ ] (int index)
  Overload of operator[] to access the argument at the given index.

3.1.1 Detailed Description

Represents a user command.

The command can have up to maxArguments arguments. The first argument is always the name of the command, and the others are its parameters. The arguments are stored in an array of strings.

3.1.2 Member Function Documentation

3.1.2.1 getArgCount()

```
int Command::getArgCount () const
```

Returns the number of arguments stored.

Returns

An integer - the number of arguments stored.

3.1.2.2 operator[]()

```
string & Command::operator[] (
            int index)
```

Overload of operator[] to access the argument at the given index.

Parameters

index	

Throws an exception if index is out of boundaries.

3.1.2.3 parse()

```
void Command::parse (
            string input)
```

Cuts a line of user input into arguments and updates the arguments array.

Parameters

input	User input.

The documentation for this class was generated from the following files:

- Command.h
- Command.cpp

3.2 CommandController Class Reference

Represents a command controller.

```
#include <CommandController.h>
```

Public Member Functions

- **CommandController** (const CommandController &other)=delete

  Copy constructor is deleted.
- CommandController & **operator=** (const CommandController &othewr)=delete

  Copy assignment operator is deleted.
- bool execute (string input)

  Executes the given user input.

Static Public Member Functions

- static CommandController * **Instance** ()

  Static function to access the single instance of the class.
- static void **Release** ()

  Releases the memory for the instance of the Singleton class.

3.2.1 Detailed Description

Represents a command controller.

Controlls the user inputs, parses them and edits the table. The class is a singleton.

3.2.2 Member Function Documentation

3.2.2.1 execute()

```
bool CommandController::execute (
            string input)
```

Executes the given user input.

Parameters

input	

Takes user input, parses it into a command and executes the command.

The documentation for this class was generated from the following files:

- CommandController.h
- CommandController.cpp

3.3 Table Class Reference

Represents a table.

```
#include <Table.h>
```

Public Member Functions

- **Table** ()=default

  Default constructor, creates an empty table.
- Table (vector< TableRow > rows, const TableRow &names)

  Creates a table with a given vector of rows and names.
- TableRow & operator[ ] (int index)

  Overloaded operator[].
- const TableRow & operator[ ] (int index) const

  Overloaded const operator[].
- int getSize () const

  Returns the amount of rows in the table.
- void addRow (TableRow row)

  Adds a new row to the table.
- bool copyRow (int index)

  Adds a new row to the table, duplicate to the one on the given index.

- void removeRow (int index)

  Removes the row at the given index.
- void removeColumn (int index)

  Removes the column at the given index.
- void removeColumn (string name)

  Removes the column with the given name.
- string findMin (int index) const

  Finds the minimal element in the column at the given index.
- string findMax (int index) const

  Finds the maximum element in the column at the given index.
- void copyMin ()

  Creates a new row with the minimal values for each column and adds it to the table.
- void copyMax ()

  Creates a new row with the maximum values for each column and adds it to the table.
- void copyFrequent ()

  Creates a new row with the most frequent values for each column and adds it to the table.
- bool swapRows (int first, int second)

  Swaps the positions of the given two rows.
- bool swapCols (int first, int second)

  Swaps the positions of the given two rows.
- bool permutate (string perm)

  Swaps the order of the columns according to the given permutation string.
- void **removeDupes** ()

  Removes the duplicate rows in the table.
- void sort (int index, bool order)

  Sorts the table based on the column at the given index and the given order.
- void sort (string name, bool order)

  Sorts the table based on the column with the given name and the given order.
- void filter (int index, string sign, string other)

  Removes the rows which do not follow the given condition at the given column index.
- void filter (string name, string sign, string other)

  Removes the rows which do not follow the given condition at the given column index.
- void addNames ()

  Removes the first row of the table and stores it in names.
- bool isEmpty () const

  Checks if the table is empty.

Friends

- ostream & operator<< (ostream &os, const Table &table)

  Overloading the operator << to print a Table.

3.3.1 Detailed Description

Represents a table.

The table is represented by a vector of TableRows. Can have a TableRow of names, which are treated as seperate from the table for methods, that work with the data in the table.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Table()

```
Table::Table (
            vector< TableRow > rows,
            const TableRow & names)
```

Creates a table with a given vector of rows and names.

Parameters

rows	
names	

3.3.3 Member Function Documentation

3.3.3.1 addNames()

```
void Table::addNames ()
```

Removes the first row of the table and stores it in names.

If the table is empty or already has names, does nothing.

3.3.3.2 addRow()

```
void Table::addRow (
            TableRow row)
```

Adds a new row to the table.

Parameters

row	The row to be added.

3.3.3.3 copyFrequent()

```
void Table::copyFrequent ()
```

Creates a new row with the most frequent values for each column and adds it to the table.

If there are multiple most frequent values, uses the smallest of them according to the compare function.

3.3.3.4 copyMax()

```
void Table::copyMax ()
```

Creates a new row with the maximum values for each column and adds it to the table.

Uses findMax. Does nothing if the table has no columns.

3.3.3.5 copyMin()

```
void Table::copyMin ()
```

Creates a new row with the minimal values for each column and adds it to the table.

Uses findMin. Does nothing if the table has no columns.

3.3.3.6 copyRow()

```
bool Table::copyRow (
            int index)
```

Adds a new row to the table, duplicate to the one on the given index.

Parameters

index	

Returns

TRUE If the idex is inside the boundaries.

FALSE otherwise.

3.3.3.7 filter() [1/2]

```
void Table::filter (
            int index,
            string sign,
            string other)
```

Removes the rows which do not follow the given condition at the given column index.

Parameters

index	The index of the column to be checked.
sign	The sign of the comparison. Can be "<", ">", "<=", ">=", "==", "!=".
other	The string to compare to.

Compares every element at the given column in the table with the other element. If the condition does not match, removes the row.

3.3.3.8 filter() [2/2]

```
void Table::filter (
            string name,
            string sign,
            string other)
```

Removes the rows which do not follow the given condition at the given column index.

Parameters

name	The name of the column to be checked.
sign	The sign of the comparison. Can be "<", ">", "<=", ">=", "==", "!=".
other	The string to compare to.

Compares every element at the given column in the table with the other element. If the condition does not match, removes the row.

3.3.3.9 findMax()

```
string Table::findMax (
            int index) const
```

Finds the maximum element in the column at the given index.

Parameters

index	

Returns

Empty string if the index is out of boundaries. Otherwise it returns the maximum string in the given column.

Uses a special compare function.

3.3.3.10 findMin()

```
string Table::findMin (
            int index) const
```

Finds the minimal element in the column at the given index.

Parameters

index	

Returns

Empty string if the index is out of boundaries. Otherwise it returns the minimal string in the given column.

Uses a special compare function.

3.3.3.11 getSize()

```
int Table::getSize () const
```

Returns the amount of rows in the table.

Returns

Returs an integer - the size of the vector of rows.

3.3.3.12 isEmpty()

```
bool Table::isEmpty () const
```

Checks if the table is empty.

Returns

TRUE If the table has 0 rows.

FALSE Otherwise.

3.3.3.13 operator[]() [1/2]

```
TableRow & Table::operator[] (
            int index)
```

Overloaded operator[].

Parameters

index	The index to access.

Returns

Returns a reference to the TableRow in position index if position is in boundaries.

If index is out of boundaries, throws exception.

3.3.3.14 operator[]() [2/2]

```
const TableRow & Table::operator[] (
            int index) const
```

Overloaded const operator[].

Parameters

index	The index to access.

Returns

Returns a constant reference to the TableRow in position index if position is in boundaries.

If index is out of boundaries, throws exception.

3.3.3.15 permutate()

```
bool Table::permutate (
            string perm)
```

Swaps the order of the columns according to the given permutation string.

Parameters

perm	Permutation string, contains only unique numbers from 1 to the number of rows.

Returns

TRUE If the permutation is successful.

FALSE If the permutation string is wrong.

3.3.3.16 removeColumn() [1/2]

```
void Table::removeColumn (
            int index)
```

Removes the column at the given index.

Parameters

index	

Removes the element at the given index for each row in the column. Does nothing if index is more than the columns in the table.

3.3.3.17 removeColumn() [2/2]

```
void Table::removeColumn (
            string name)
```

Removes the column with the given name.

Parameters

name	

Attempts to find the index of the column with the given name. If multiple are found, removes the last one found. If none are found, does nothing.

3.3.3.18 removeRow()

```
void Table::removeRow (
            int index)
```

Removes the row at the given index.

Parameters

index	

Does nothing if index is out of boundaries.

3.3.3.19 sort() [1/2]

```
void Table::sort (
            int index,
            bool order)
```

Sorts the table based on the column at the given index and the given order.

Parameters

index	
order	If TRUE, sorts in descending order, if FALSE sorts in ascending order.

If index is outside of boundaries, does nothing.

3.3.3.20 sort() [2/2]

```
void Table::sort (
            string name,
            bool order)
```

Sorts the table based on the column with the given name and the given order.

Parameters

name	
order	If TRUE, sorts in descending order, if FALSE, sorts in ascending order.

If name is not found, does nothing.

3.3.3.21 swapCols()

```
bool Table::swapCols (
            int first,
            int second)
```

Swaps the positions of the given two rows.

Parameters

first	
second	

Returns

TRUE If the index is inside of the boundaries and the swap is successful.

FALSE If the index is outside of the boundaries.

3.3.3.22 swapRows()

```
bool Table::swapRows (
            int first,
            int second)
```

Swaps the positions of the given two rows.

Parameters

first	
second	

Returns

TRUE If the index is inside of the boundaries and the swap is successful.

FALSE If the index is outside of the boundaries.

3.3.4 Friends And Related Symbol Documentation

3.3.4.1 operator<<

```
ostream & operator<< (
            ostream & os,
            const Table & table)  [friend]
```

Overloading the operator << to print a Table.

Parameters

os	The stream to output to.
table	The given Table.

Returns

The output stream by reference.

The documentation for this class was generated from the following files:

- Table.h
- Table.cpp

3.4 TableRow Class Reference

Represents a row in the table.

```
#include <TableRow.h>
```

Public Member Functions

- **TableRow** ()=default

  Default constructor, using vector's and string's default constructors.
- TableRow (vector< string > data)

  Creates a new row with the given data.
- string & operator[ ] (int index)

  Overloaded operator[].
- const string & operator[ ] (int index) const

  Overloaded const operator[].
- void removeElement (int index)

  Removes an element at the given position.
- void parseFromFile (string data)

  Loads a row from a string.
- int getSize () const

  Returns the size of the vector.
- void addElement (string newElement)

  Adds a new element to the end of the row.
- bool swap (int first, int second)

  Swaps the positions of the elements at index first and second.
- void **reset** ()

  Resets the row to an empty one.
- bool isEmpty () const

  Checks if the row is empty.

Friends

- ostream & operator<< (ostream &os, const TableRow &row)

  Overloading the operator << to print a TableRow.
- bool operator== (const TableRow &lhs, const TableRow &rhs)

  Overloading the operator == to compare two TableRows.

3.4.1 Detailed Description

Represents a row in the table.

The row is represented as a vector of strings.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 TableRow()

```
TableRow::TableRow (
            vector< string > data)
```

Creates a new row with the given data.

Parameters

data	The data for the row.

3.4.3 Member Function Documentation

3.4.3.1 addElement()

```
void TableRow::addElement (
            string newElement)
```

Adds a new element to the end of the row.

Parameters

newElement	The new element to be added.

3.4.3.2 getSize()

```
int TableRow::getSize () const
```

Returns the size of the vector.

Returns

Returns an integer - the size of the vector.

3.4.3.3 isEmpty()

```
bool TableRow::isEmpty () const
```

Checks if the row is empty.

Returns

TRUE if the row is empty.

FALSE if the row is not empty.

3.4.3.4 operator[]() [1/2]

```
string & TableRow::operator[] (
            int index)
```

Overloaded operator[].

Parameters

index	The index to access.

Returns

Returns a reference to the string in position index if position is in boundaries.

If index is out of boundaries, throws exception.

3.4.3.5 operator[]() [2/2]

```
const string & TableRow::operator[] (
            int index) const
```

Overloaded const operator[].

Parameters

index	The index to access.

Returns

Returns a constant reference to the string in position index if position is in boundaries.

If index is out of boundaries, throws exception.

3.4.3.6 parseFromFile()

```
void TableRow::parseFromFile (
            string data)
```

Loads a row from a string.

Parameters

data	Given string to load from.

Cuts the string in pieces with delimiter ' ' and loads a row with the given pieces.

3.4.3.7 removeElement()

```
void TableRow::removeElement (
            int index)
```

Removes an element at the given position.

Parameters

index	The index to remove.

If index is out of boundaries, doesn't do anything.

3.4.3.8 swap()

```
bool TableRow::swap (
            int first,
            int second)
```

Swaps the positions of the elements at index first and second.

Parameters

first	The index of the first element.
second	The index of the second element.

Returns

TRUE if the indexes are correct and the swap is successful.

FALSE if the indexes are out of boundaries.

3.4.4 Friends And Related Symbol Documentation

3.4.4.1 operator<<

```
ostream & operator<< (
            ostream & os,
            const TableRow & row)  [friend]
```

Overloading the operator << to print a TableRow.

Parameters

os	The stream to output to.
row	The given TableRow.

Returns

The output stream by reference.

3.4.4.2 operator==

```
bool operator== (
            const TableRow & lhs,
            const TableRow & rhs)  [friend]
```

Overloading the operator == to compare two TableRows.

Parameters

lhs	First TableRow.
rhs	Second TableRow.

Returns

TRUE if they are identical.

FALSE if they are not identical.

The documentation for this class was generated from the following files:

- TableRow.h
- TableRow.cpp

Chapter 4

File Documentation

4.1 Command.h

```
00001 #include <string>
00002 #include <cstring>
00003 #include <iostream>
00004 #include <stdexcept>
00005
00006 using namespace std;
00007
00008 const int maxArguments = 10;
00009
00016 class Command
00017 {
00018 private:
00019     string arguments[maxArguments];
00020     int argCount;
00021
00022 public:
00023
00027     Command();
00028
00033     void parse(string input);
00034
00039     int getArgCount() const;
00040
00047     string& operator[](int index);
00048 };
```

4.2 CommandController.h

```
00001 #pragma once
00002 #include "Command.h"
00003 #include "Table.h"
00004 #include <fstream>
00005 #include <string>
00006
00007 using namespace std;
00008
00015 class CommandController
00016 {
00017 private:
00018     Table table;
00019     Table undoTable;
00020     Command command;
00021     string filePath;
00022     bool hasChanged = false;
00023     bool hasNames = false;
00024     static CommandController* pInstance;
00025
00029     CommandController() {}
00030
00039     bool loadFromFile(const char* filePath);
00040
00046     bool saveToFile();
00047
00054     bool saveToFile(const char* filePath);
```

```
00055
00056 public:
00057
00061     CommandController(const CommandController& other) = delete;
00062
00066     CommandController& operator=(const CommandController& othewr) = delete;
00067
00071     static CommandController* Instance();
00072
00079     bool execute(string input);
00080
00084     static void Release();
00085 };
```

4.3 Table.h

```
00001 #pragma once
00002 #include "TableRow.h"
00003
00010 class Table
00011 {
00012 private:
00013     TableRow names;
00014     vector<TableRow> rows;
00015
00016 public:
00017
00021     Table() = default;
00022
00028     Table(vector<TableRow> rows, const TableRow& names);
00029
00037     TableRow& operator[](int index);
00038
00046     const TableRow& operator[](int index)const;
00047
00052     int getSize() const;
00053
00058     void addRow(TableRow row);
00059
00066     bool copyRow(int index);
00067
00074     void removeRow(int index);
00075
00082     void removeColumn(int index);
00083
00090     void removeColumn(string name);
00091
00099     string findMin(int index)const;
00100
00108     string findMax(int index)const;
00109
00115     void copyMin();
00116
00122     void copyMax();
00123
00129     void copyFrequent();
00130
00138     bool swapRows(int first, int second);
00139
00147     bool swapCols(int first, int second);
00148
00155     bool permutate(string perm);
00156
00160     void removeDupes();
00161
00169     void sort(int index, bool order);
00170
00178     void sort(string name, bool order);
00179
00188     void filter(int index, string sign, string other);
00189
00198     void filter(string name, string sign, string other);
00199
00205     void addNames();
00206
00212     bool isEmpty()const;
00213
00214     friend ostream& operator<<(ostream& os, const Table& table);
00215 };
00216
00223 ostream& operator<<(ostream& os, const Table& table);
00224
00231 bool checkPerm(string perm);
```

```
00232
00243 int compare(string lhs, string rhs);
```

4.4 TableRow.h

```
00001 #pragma once
00002 #include <string>
00003 #include <vector>
00004 #include <iostream>
00005 using namespace std;
00006
00013 class TableRow
00014 {
00015 private:
00016     vector<string> data;
00017
00018 public:
00022     TableRow() = default;
00023
00028     TableRow(vector<string> data);
00029
00037     string& operator[](int index);
00038
00046     const string& operator[](int index)const;
00047
00054     void removeElement(int index);
00055
00062     void parseFromFile(string data);
00063
00068     int getSize()const;
00069
00074     void addElement(string newElement);
00075
00083     bool swap(int first, int second);
00084
00088     void reset();
00089
00095     bool isEmpty()const;
00096
00097     friend ostream& operator<<(ostream& os, const TableRow& row);
00098     friend bool operator==(const TableRow& lhs, const TableRow& rhs);
00099 };
00100
00107 ostream& operator<<(ostream& os, const TableRow& row);
00108
00116 bool operator==(const TableRow& lhs, const TableRow& rhs);
```

Index