Quiz Game Test_me

Generated by Doxygen 1.9.3

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 files with brief descriptions:

Chapter 3

Class Documentation

3.1 Player Class Reference

`#include <Player.h>`

Public Member Functions

- Player ()
- void Player_inputdata ()
- void Player_play ()

3.1.1 Detailed Description

A Player Class with several private variables and methods

Parameters

int	Total an integer tracting player score
char	username a character variable storing player username
Game_level	Select an enum type/ difficulty level selected by user
std::vector	<Question> Tasks a vector of questions read from file based on difficulty level selcted by player in Game_levl

Definition at line 17 of file Player.h.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Player()

```
Player::Player ( )
```

A constructor for the Question Class without parameters

Definition at line 8 of file Player.cpp.

3.1.3 Member Function Documentation

3.1.3.1 Player_inputdata()

```
void Player::Player_inputdata ( )
```

A public method that takes the input from the player, switches to the difficulty level, reads questions from a specific file per difficulty level and initials game play

Definition at line 13 of file Player.cpp.

3.1.3.2 Player_play()

```
void Player::Player_play ( )
```

A public method that displays all of the questions from a choosen category selected by the player read from a vector of questions

Definition at line 284 of file Player.cpp.

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

- Player.h
- Player.cpp

3.2 Question Class Reference

```
#include <Question.h>
```

Public Member Functions

- bool Evaluate_ans (std::string response, std::string _answer)
- Question (std::string Question_text, std::string A, std::string B, std::string C, std::string D, std::string Answer, int Mark)
- Question ()
- void Question_set (std::string Rquestion_text, std::string Ra, std::string Rb, std::string Rc, std::string Rd, std::string Ranswer, int Rmark)
- int Get_answer ()

3.2.1 Detailed Description

A Question Class with several private variables and methods

Parameters

std::string	question_text a string variable storing question text
std::string	a a string variable storing option a
std::string	b a string variable storing option b
std::string	c a string variable storing option c
std::string	d a string variable storing option d
std::string	answer a string variable storing the the correct option between a,b,c,d
int	mark an integer variable storing the mark

Definition at line 20 of file Question.h.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Question() [1/2]

```
Question::Question (
            std::string Question_text,
            std::string A,
            std::string B,
            std::string C,
            std::string D,
            std::string Answer,
            int Mark )
```

A constructor for the Question Class with given parameters

Parameters

std::string	Question_text a string variable storing question text
std::string	A a string variable storing option a
std::string	B a string variable storing option b
std::string	C a string variable storing option c
std::string	D a string variable storing option d
std::string	Answer a string variable storing the the correct option between a,b,c,d
int	Mark an integer variable storing the mark

Definition at line 11 of file Question.cpp.

3.2.2.2 Question() [2/2]

```
Question::Question ( )
```

A constructor for the Question Class without parameters

Definition at line 6 of file Question.cpp.

3.2.3 Member Function Documentation

3.2.3.1 Evaluate_ans()

```
bool Question::Evaluate_ans (
            std::string response,
            std::string _answer )  [inline]
```

A public method of the Question Class that evaluates the response from the player with the answer and returns a pass(1) or fail(0)

Parameters

std::string	response a string variable with the response from player
std::string	answer a string variable storing the the correct option between a,b,c,d for the question class

Returns

bool 0 failed question

bool 1 passed question

Definition at line 34 of file Question.h.

3.2.3.2 Get_answer()

```
int Question::Get_answer ( )
```

A public method that displays the question to player, receives the response, checks if the answer is correct and allocates score for the question

Returns

int score an integer variable storing the mark for the question based on evaluation from the bool Evaluate_ans() method.

Definition at line 30 of file Question.cpp.

3.2.3.3 Question_set()

```
void Question::Question_set (
            std::string Rquestion_text,
            std::string Ra,
            std::string Rb,
            std::string Rc,
            std::string Rd,
            std::string Ranswer,
            int Rmark )
```

A public method that intializes the variables and sets a question

Parameters

std::string	Rquestion_text a string variable storing question text
std::string	Ra a string variable storing option a
std::string	Rb a string variable storing option b
std::string	Rc a string variable storing option c
std::string	Rd a string variable storing option d
std::string	Ranswer a string variable storing the the correct option between a,b,c,d
int	Rmark an integer variable storing the mark

Definition at line 19 of file Question.cpp.

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

- Question.h
- Question.cpp

3.3 quiz_question Struct Reference

`#include <QuestionSet.h>`

Public Attributes

- Question q1
- Question q2
- Question q3
- Question q4
- Question q5
- Question q6
- Question q7
- Question q8
- Question q9
- Question q10

3.3.1 Detailed Description

A structure that stores a set of 10 questions for each player

Definition at line 3 of file QuestionSet.cpp.

3.3.2 Member Data Documentation

3.3.2.1 q1

```
Question quiz_question::q1
```

Definition at line 5 of file QuestionSet.cpp.

3.3.2.2 q10

```
Question quiz_question::q10
```

Definition at line 14 of file QuestionSet.cpp.

3.3.2.3 q2

```
Question quiz_question::q2
```

Definition at line 6 of file QuestionSet.cpp.

3.3.2.4 q3

```
Question quiz_question::q3
```

Definition at line 7 of file QuestionSet.cpp.

3.3.2.5 q4

```
Question quiz_question::q4
```

Definition at line 8 of file QuestionSet.cpp.

3.3.2.6 q5

```
Question quiz_question::q5
```

Definition at line 9 of file QuestionSet.cpp.

3.3.2.7 q6

```
Question quiz_question::q6
```

Definition at line 10 of file QuestionSet.cpp.

3.3.2.8 q7

```
Question quiz_question::q7
```

Definition at line 11 of file QuestionSet.cpp.

3.3.2.9 q8

```
Question quiz_question::q8
```

Definition at line 12 of file QuestionSet.cpp.

3.3.2.10 q9

```
Question quiz_question::q9
```

Definition at line 13 of file QuestionSet.cpp.

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

- QuestionSet.cpp
- QuestionSet.h

Chapter 4

File Documentation

4.1 DifficultyLevel.cpp File Reference

```
#include "Enum.h"
```

Functions

- Game_level convert (char difficulty)

4.1.1 Function Documentation

4.1.1.1 convert()

```
Game_level convert (
            char difficulty )
```

The function reads the char input(difficulty) from player and allocates the correct enum difficulty type

Parameters

std,↩ :	char difficulty variable entered by player to select difficulty level

Returns

operation r returns an enum type with a specific difficulty selected by player

Definition at line 3 of file DifficultyLevel.cpp.

4.2 DifficultyLevel.cpp

Go to the documentation of this file.

```
00001 #include "Enum.h"
00002
00003 Game_level convert(char difficulty)
00004 {
00005     Game_level r = Game_level::simple;
00006
00007
00008     if (difficulty == '1') { r = Game_level::simple; }
00009     if (difficulty == '2') { r = Game_level::medium; }
00010     if (difficulty == '3') { r = Game_level::hard; }
00011     if (difficulty == '0' or difficulty > '3') { r = Game_level::other; }
00012     return r;
00013 }
```

4.3 DifficultyLevel.h File Reference

```
#include "Enum.h"
```

Macros

- #define DIFFICULTYLEVL_H

Functions

- Game_level convert (char difficulty)

4.3.1 Macro Definition Documentation

4.3.1.1 DIFFICULTYLEVL_H

```
#define DIFFICULTYLEVL_H
```

Definition at line 3 of file DifficultyLevel.h.

4.3.2 Function Documentation

4.3.2.1 convert()

```
Game_level convert (
            char difficulty )
```

The function reads the char input(difficulty) from player and allocates the correct enum difficulty type

Parameters

std,↩ :	char difficulty variable entered by player to select difficulty level

Returns

operation r returns an enum type with a specific difficulty selected by player

Definition at line 3 of file DifficultyLevel.cpp.

4.4 DifficultyLevel.h

Go to the documentation of this file.

```
00001 #pragma
00002 #ifndef DIFFICULTYLEVEL_H
00003 #define DIFFICULTYLEVL_H
00004
00005 #include"Enum.h"
00006
00007
00013 Game_level convert(char difficulty);
00014
00015 #endif
00016
```

4.5 Enum.cpp File Reference

Enumerations

- enum class Game_level {
  simple = 1 , medium , hard , other ,
  simple = 1 , medium , hard , other }

4.5.1 Enumeration Type Documentation

4.5.1.1 Game_level

```
enum class Game_level  [strong]
```

Enumerator

simple	
medium	
hard	
other	
simple	
medium	
hard	
other	

Definition at line 2 of file Enum.cpp.

4.6 Enum.cpp

Go to the documentation of this file.

```
00001
00002 enum class Game_level { simple = 1, medium, hard, other };
```

4.7 Enum.h File Reference

Macros

- #define ENUM_H

Enumerations

- enum class Game_level {
  simple = 1 , medium , hard , other ,
  simple = 1 , medium , hard , other }

4.7.1 Macro Definition Documentation

4.7.1.1 ENUM_H

```
#define ENUM_H
```

Definition at line 3 of file Enum.h.

4.7.2 Enumeration Type Documentation

4.7.2.1 Game_level

```
enum class Game_level  [strong]
```

An enum of 4 difficulty levels for each player

Enumerator

simple	
medium	
hard	
other	
simple	
medium	
hard	

Definition at line 9 of file Enum.h.

4.8 Enum.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #ifndef ENUM_H
00003 #define ENUM_H
00004
00009 enum class Game_level { simple = 1, medium, hard, other };
00010
00011 #endif // !ENUM.H
```

4.9 Player.cpp File Reference

```
#include <iostream>
#include "Player.h"
#include "QuestionSet.h"
#include "DifficultyLevel.h"
#include "ReadingFile.h"
```

4.10 Player.cpp

Go to the documentation of this file.

```
00001 #include <iostream>
00002 #include "Player.h"
00003 #include "QuestionSet.h"
00004 #include "DifficultyLevel.h"
00005 #include "ReadingFile.h"
00006
00007
00008 Player::Player()
00009 {
00010     Player_inputdata();
00011 }
00012
00013 void Player::Player_inputdata()
00014 {
00015     char difficulty;
00016     std::string file;
00017     std::vector< std::vector<std::string» questionsfromfile;
00018     quiz_question Set;
00019
00020     std::cout « "Welcome to TEST-ME quiz game \n" « '\n';
00021
00022     std::cout « "Please select Quiz difficulty level : \n type 1 for simple \n 2 for average \n 3 for
       hard \n 4 for expert" « '\n';
00023     std::cin » difficulty;
00024
00025     std::cout « "please Type in your user name" « '\n';
00026
00027     std::cin » username;
00028
00029
00030     Game_level Select = convert(difficulty);
00031     switch (Select)
00032     {
00033
00034     case Game_level::simple:
00035
00036         std::cout « "\n Simple Quiz Game mode activated \n";
00037
00038         file = "question_simple.txt";
00039         questionsfromfile = read_questions(file);
00040
00041
```

```
00042         for (int row = 0; row < questionsfromfile.size(); row++)
00043         {
00044             for (int col = 0; col < questionsfromfile[0].size(); col++)
00045             {
00046                 if (row == 0)
00047                 {
00048 
00049                     Set.q1.Question_set(questionsfromfile[0][0], questionsfromfile[0][1],
      questionsfromfile[0][2], questionsfromfile[0][3], questionsfromfile[0][4], questionsfromfile[0][5],
      stoi(questionsfromfile[0][6]));
00050 
00051                 }
00052                 if (row == 1)
00053                 {
00054 
00055                     Set.q2.Question_set(questionsfromfile[1][0], questionsfromfile[1][1],
      questionsfromfile[1][2], questionsfromfile[1][3], questionsfromfile[1][4], questionsfromfile[1][5],
      stoi(questionsfromfile[1][6]));
00056 
00057                 }
00058                 if (row == 2)
00059                 {
00060 
00061                     Set.q3.Question_set(questionsfromfile[2][0], questionsfromfile[2][1],
      questionsfromfile[2][2], questionsfromfile[2][3], questionsfromfile[2][4], questionsfromfile[2][5],
      stoi(questionsfromfile[2][6]));
00062 
00063 
00064                 }
00065                 if (row == 3)
00066                 {
00067 
00068                     Set.q4.Question_set(questionsfromfile[3][0], questionsfromfile[3][1],
      questionsfromfile[3][2], questionsfromfile[3][3], questionsfromfile[3][4], questionsfromfile[3][5],
      stoi(questionsfromfile[3][6]));
00069 
00070                 }
00071                 if (row == 4)
00072                 {
00073                     Set.q5.Question_set(questionsfromfile[4][0], questionsfromfile[4][1],
      questionsfromfile[4][2], questionsfromfile[4][3], questionsfromfile[4][4], questionsfromfile[4][5],
      stoi(questionsfromfile[4][6]));
00074 
00075 
00076                 }
00077                 if (row == 5)
00078                 {
00079 
00080                     Set.q6.Question_set(questionsfromfile[5][0], questionsfromfile[5][1],
      questionsfromfile[5][2], questionsfromfile[5][3], questionsfromfile[5][4], questionsfromfile[5][5],
      stoi(questionsfromfile[5][6]));
00081 
00082 
00083                 }
00084                 if (row == 6)
00085                 {
00086                     Set.q7.Question_set(questionsfromfile[6][0], questionsfromfile[6][1],
      questionsfromfile[6][2], questionsfromfile[6][3], questionsfromfile[6][4], questionsfromfile[6][5],
      stoi(questionsfromfile[6][6]));
00087 
00088                 }
00089                 if (row == 7)
00090                 {
00091                     Set.q8.Question_set(questionsfromfile[7][0], questionsfromfile[7][1],
      questionsfromfile[7][2], questionsfromfile[7][3], questionsfromfile[7][4], questionsfromfile[7][5],
      stoi(questionsfromfile[7][6]));
00092 
00093                 }
00094                 if (row == 8)
00095                 {
00096 
00097                     Set.q9.Question_set(questionsfromfile[8][0], questionsfromfile[8][1],
      questionsfromfile[8][2], questionsfromfile[8][3], questionsfromfile[8][4], questionsfromfile[8][5],
      stoi(questionsfromfile[8][6]));
00098 
00099                 }
00100                 if (row == 9)
00101                 {
00102                     Set.q10.Question_set(questionsfromfile[9][0], questionsfromfile[9][1],
      questionsfromfile[9][2], questionsfromfile[9][3], questionsfromfile[9][4], questionsfromfile[9][5],
      stoi(questionsfromfile[9][6]));
00103 
00104                 }
00105 
00106             }
00107         }
00108 
```

```
00109         Tasks = { Set.q1, Set.q2,Set.q3,Set.q4, Set.q5, Set.q6, Set.q7, Set.q8, Set.q9, Set.q10 };
00110
00111         Player_play();
00112
00113         break;
00114     case Game_level::medium:
00115
00116         std::cout « "\n Medium Quiz Game mode activated \n";
00117
00118         file = "question_medium.txt";
00119         questionsfromfile = read_questions(file);
00120
00121         for (int row = 0; row < questionsfromfile.size(); row++)
00122         {
00123             for (int col = 0; col < questionsfromfile[0].size(); col++)
00124             {
00125                 if (row == 0)
00126                 {
00127
00128                     Set.q1.Question_set(questionsfromfile[0][0], questionsfromfile[0][1],
      questionsfromfile[0][2], questionsfromfile[0][3], questionsfromfile[0][4], questionsfromfile[0][5],
      stoi(questionsfromfile[0][6]));
00129
00130                 }
00131                 if (row == 1)
00132                 {
00133
00134                     Set.q2.Question_set(questionsfromfile[1][0], questionsfromfile[1][1],
      questionsfromfile[1][2], questionsfromfile[1][3], questionsfromfile[1][4], questionsfromfile[1][5],
      stoi(questionsfromfile[1][6]));
00135
00136                 }
00137                 if (row == 2)
00138                 {
00139
00140                     Set.q3.Question_set(questionsfromfile[2][0], questionsfromfile[2][1],
      questionsfromfile[2][2], questionsfromfile[2][3], questionsfromfile[2][4], questionsfromfile[2][5],
      stoi(questionsfromfile[2][6]));
00141
00142
00143                 }
00144                 if (row == 3)
00145                 {
00146
00147                     Set.q4.Question_set(questionsfromfile[3][0], questionsfromfile[3][1],
      questionsfromfile[3][2], questionsfromfile[3][3], questionsfromfile[3][4], questionsfromfile[3][5],
      stoi(questionsfromfile[3][6]));
00148
00149                 }
00150                 if (row == 4)
00151                 {
00152                     Set.q5.Question_set(questionsfromfile[4][0], questionsfromfile[4][1],
      questionsfromfile[4][2], questionsfromfile[4][3], questionsfromfile[4][4], questionsfromfile[4][5],
      stoi(questionsfromfile[4][6]));
00153
00154
00155                 }
00156                 if (row == 5)
00157                 {
00158
00159                     Set.q6.Question_set(questionsfromfile[5][0], questionsfromfile[5][1],
      questionsfromfile[5][2], questionsfromfile[5][3], questionsfromfile[5][4], questionsfromfile[5][5],
      stoi(questionsfromfile[5][6]));
00160
00161
00162                 }
00163                 if (row == 6)
00164                 {
00165                     Set.q7.Question_set(questionsfromfile[6][0], questionsfromfile[6][1],
      questionsfromfile[6][2], questionsfromfile[6][3], questionsfromfile[6][4], questionsfromfile[6][5],
      stoi(questionsfromfile[6][6]));
00166
00167                 }
00168                 if (row == 7)
00169                 {
00170                     Set.q8.Question_set(questionsfromfile[7][0], questionsfromfile[7][1],
      questionsfromfile[7][2], questionsfromfile[7][3], questionsfromfile[7][4], questionsfromfile[7][5],
      stoi(questionsfromfile[7][6]));
00171
00172                 }
00173                 if (row == 8)
00174                 {
00175
00176                     Set.q9.Question_set(questionsfromfile[8][0], questionsfromfile[8][1],
      questionsfromfile[8][2], questionsfromfile[8][3], questionsfromfile[8][4], questionsfromfile[8][5],
      stoi(questionsfromfile[8][6]));
00177
```

```
00178                 }
00179                 if (row == 9)
00180                 {
00181                     Set.q10.Question_set(questionsfromfile[9][0], questionsfromfile[9][1],
      questionsfromfile[9][2], questionsfromfile[9][3], questionsfromfile[9][4], questionsfromfile[9][5],
      stoi(questionsfromfile[9][6]));
00182 
00183                 }
00184 
00185             }
00186         }
00187 
00188         Tasks = { Set.q1, Set.q2,Set.q3,Set.q4, Set.q5, Set.q6, Set.q7, Set.q8, Set.q9, Set.q10 };
00189 
00190         Player_play();
00191 
00192         break;
00193 
00194     case Game_level::hard:
00195 
00196         std::cout << "Hard Quiz Game mode activated" << '\n';
00197 
00198         file = "question_hard.txt";
00199         questionsfromfile = read_questions(file);
00200 
00201         for (int row = 0; row < questionsfromfile.size(); row++)
00202         {
00203             for (int col = 0; col < questionsfromfile[0].size(); col++)
00204             {
00205                 if (row == 0)
00206                 {
00207 
00208                     Set.q1.Question_set(questionsfromfile[0][0], questionsfromfile[0][1],
      questionsfromfile[0][2], questionsfromfile[0][3], questionsfromfile[0][4], questionsfromfile[0][5],
      stoi(questionsfromfile[0][6]));
00209 
00210                 }
00211                 if (row == 1)
00212                 {
00213 
00214                     Set.q2.Question_set(questionsfromfile[1][0], questionsfromfile[1][1],
      questionsfromfile[1][2], questionsfromfile[1][3], questionsfromfile[1][4], questionsfromfile[1][5],
      stoi(questionsfromfile[1][6]));
00215 
00216                 }
00217                 if (row == 2)
00218                 {
00219 
00220                     Set.q3.Question_set(questionsfromfile[2][0], questionsfromfile[2][1],
      questionsfromfile[2][2], questionsfromfile[2][3], questionsfromfile[2][4], questionsfromfile[2][5],
      stoi(questionsfromfile[2][6]));
00221 
00222 
00223                 }
00224                 if (row == 3)
00225                 {
00226 
00227                     Set.q4.Question_set(questionsfromfile[3][0], questionsfromfile[3][1],
      questionsfromfile[3][2], questionsfromfile[3][3], questionsfromfile[3][4], questionsfromfile[3][5],
      stoi(questionsfromfile[3][6]));
00228 
00229                 }
00230                 if (row == 4)
00231                 {
00232                     Set.q5.Question_set(questionsfromfile[4][0], questionsfromfile[4][1],
      questionsfromfile[4][2], questionsfromfile[4][3], questionsfromfile[4][4], questionsfromfile[4][5],
      stoi(questionsfromfile[4][6]));
00233 
00234 
00235                 }
00236                 if (row == 5)
00237                 {
00238 
00239                     Set.q6.Question_set(questionsfromfile[5][0], questionsfromfile[5][1],
      questionsfromfile[5][2], questionsfromfile[5][3], questionsfromfile[5][4], questionsfromfile[5][5],
      stoi(questionsfromfile[5][6]));
00240 
00241 
00242                 }
00243                 if (row == 6)
00244                 {
00245                     Set.q7.Question_set(questionsfromfile[6][0], questionsfromfile[6][1],
      questionsfromfile[6][2], questionsfromfile[6][3], questionsfromfile[6][4], questionsfromfile[6][5],
      stoi(questionsfromfile[6][6]));
00246 
00247                 }
00248                 if (row == 7)
```

```
00249                 {
00250                     Set.q8.Question_set(questionsfromfile[7][0], questionsfromfile[7][1],
      questionsfromfile[7][2], questionsfromfile[7][3], questionsfromfile[7][4], questionsfromfile[7][5],
      stoi(questionsfromfile[7][6]));
00251
00252                 }
00253                 if (row == 8)
00254                 {
00255
00256                     Set.q9.Question_set(questionsfromfile[8][0], questionsfromfile[8][1],
      questionsfromfile[8][2], questionsfromfile[8][3], questionsfromfile[8][4], questionsfromfile[8][5],
      stoi(questionsfromfile[8][6]));
00257
00258                 }
00259                 if (row == 9)
00260                 {
00261                     Set.q10.Question_set(questionsfromfile[9][0], questionsfromfile[9][1],
      questionsfromfile[9][2], questionsfromfile[9][3], questionsfromfile[9][4], questionsfromfile[9][5],
      stoi(questionsfromfile[9][6]));
00262
00263                 }
00264
00265             }
00266         }
00267
00268         Tasks = { Set.q1, Set.q2,Set.q3,Set.q4, Set.q5, Set.q6, Set.q7, Set.q8, Set.q9, Set.q10 };
00269
00270         Player_play();
00271
00272         break;
00273
00274     case Game_level::other:
00275
00276         std::cout « " The difficulty level you selected is not available, please start over and select
      difficulty level: 1 = simple \n 2 for average\n 3 for hard\n 4 for expert \n" « "Good-bye and try
      again soon";
00277
00278         break;
00279
00280     };
00281
00282 }
00283
00284 void Player::Player_play()
00285 {
00286     std::cout « "\n Your Quiz game has been generated begin!!!" « '\n';
00287
00288     for (int t=0 ; t < Tasks.size(); t++)
00289     {
00290         Total += Tasks[t].Get_answer();
00291     }
00292
00293     std::cout « username « "  your total game score is : " « Total « "/ 100 \n";
00294 };
```

4.11 Player.h File Reference

```
#include <vector>
#include "Question.h"
#include "DifficultyLevel.h"
```

Classes

- class Player

Macros

- #define PLAYER_H

4.11.1 Macro Definition Documentation

4.11.1.1 PLAYER_H

```
#define PLAYER_H
```

Definition at line 3 of file Player.h.

4.12 Player.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #ifndef PLAYER_H
00003 #define PLAYER_H
00004
00005 #include <vector>
00006 #include "Question.h"
00007 #include"DifficultyLevel.h"
00008
00009
00017 class Player
00018 {
00019     int Total = 0;
00020     char username;
00021     Game_level Select;
00022     std::vector <Question> Tasks;
00023
00024
00025 public:
00026
00027
00029     Player();
00030
00032     void Player_inputdata();
00033
00035     void Player_play();
00036 };
00037
00038 #endif
```

4.13 Question.cpp File Reference

```
#include <iostream>
#include "Question.h"
```

Functions

- bool Evaluate_ans (std::string response, std::string _answer)

4.13.1 Function Documentation

4.13.1.1 Evaluate_ans()

```
bool Evaluate_ans (
            std::string response,
            std::string _answer )
```

Definition at line 17 of file Question.cpp.

4.14 Question.cpp

Go to the documentation of this file.

```
00001 #include <iostream>
00002 #include "Question.h"
00003
00004 //Question constructor
00005
00006 Question::Question()
00007 {
00008
00009 }
00010
00011 Question::Question(std::string Question_text, std::string  A, std::string  B, std::string  C,
       std::string D, std::string Answer, int Mark)
00012 {
00013     Question_set(Question_text, A, B, C, D, Answer, Mark);
00014 }
00015
00016 //Question memeber function
00017 bool Evaluate_ans(std::string response, std::string _answer) { if (response == _answer) { return 1; }
       else { return 0; } }
00018
00019 void Question::Question_set(std::string Rquestion_text, std::string Ra, std::string Rb, std::string
       Rc, std::string Rd, std::string Ranswer, int Rmark)
00020 {
00021     question_text = Rquestion_text;
00022     a = Ra;
00023     b = Rb;
00024     c = Rc;
00025     d = Rd;
00026     answer = Ranswer;
00027     mark = Rmark;
00028 }
00029
00030 int Question::Get_answer() {
00031
00032
00033     int score = 0;
00034     std::string response = "";
00035
00036
00037     std::cout « "\n";
00038     std::cout « "Question : " « question_text « '\n';
00039     std::cout « "a. " « a « '\n';
00040     std::cout « "b. " « b « '\n';
00041     std::cout « "c. " « c « '\n';
00042     std::cout « "d. " « d « '\n';
00043
00044     std::cout « " Select an answer from the options (a,b,c,d)" « "\n";
00045
00046     std::cin » response;
00047
00048     std::cout « "confirm answer" « "\n";
00049     std::cin » response;
00050
00051
00052     bool validate = Evaluate_ans(response, answer);
00053
00054     if (Evaluate_ans(response, answer) == true)
00055     {
00056         score += mark;
00057     }
00058     else
00059     {
00060         score = 0;
00061     }
00062     return score;
00063 }
```

4.15 Question.h File Reference

```
#include <string>
```

Classes

- class Question

Macros

- #define QUESTION_H

4.15.1 Macro Definition Documentation

4.15.1.1 QUESTION_H

```
#define QUESTION_H
```

Definition at line 3 of file Question.h.

4.16 Question.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #ifndef QUESTION_H
00003 #define QUESTION_H
00004
00005 #include <string>
00006 using namespace std;
00007
00008
00020 class Question
00021 {
00022 private:
00023     std::string question_text, a, b, c, d, answer;
00024     int mark=0;
00025
00026 public:
00034     bool Evaluate_ans(std::string response, std::string _answer) { if (response == _answer) { return
       1; } else { return 0; } };
00035
00046     Question(std::string Question_text, std::string  A, std::string  B, std::string  C, std::string D,
      std::string Answer, int Mark);
00047
00049     Question();
00050
00061     void Question_set(std::string Rquestion_text, std::string Ra, std::string Rb, std::string Rc,
      std::string Rd, std::string Ranswer, int Rmark);
00062
00067     int Get_answer();
00068 };
00069
00070 #endif
00071
00072
```

4.17 QuestionSet.cpp File Reference

```
#include "Question.h"
```

Classes

- struct quiz_question

4.18 QuestionSet.cpp

Go to the documentation of this file.

```
00001 #include"Question.h"
00002
00003 struct quiz_question
00004 {
00005     Question q1;
00006     Question q2;
00007     Question q3;
00008     Question q4;
00009     Question q5;
00010     Question q6;
00011     Question q7;
00012     Question q8;
00013     Question q9;
00014     Question q10;
00015 };
```

4.19 QuestionSet.h File Reference

```
#include "QuestionSet.h"
```

Classes

- struct quiz_question

Macros

- #define STRUCT_H

4.19.1 Macro Definition Documentation

4.19.1.1 STRUCT_H

```
#define STRUCT_H
```

Definition at line 3 of file QuestionSet.h.

4.20 QuestionSet.h

Go to the documentation of this file.

```
00001 #pragma once
00002 #ifndef STRUCT_H
00003 #define STRUCT_H
00004
00005 #include"QuestionSet.h"
00006
00007
00011 struct quiz_question
00012 {
00013     Question q1;
00014     Question q2;
00015     Question q3;
00016     Question q4;
00017     Question q5;
00018     Question q6;
00019     Question q7;
00020     Question q8;
00021     Question q9;
00022     Question q10;
00023 };
00024
00025 #endif // !STRUCT.H
00026
```

4.21 Quiz.cpp File Reference

```
#include <iostream>
#include "StringManipulators.h"
#include "ReadingFile.h"
#include "Player.h"
```

Functions

- int main ()

4.21.1 Function Documentation

4.21.1.1 main()

```
int main ( )
```

Definition at line 11 of file Quiz.cpp.

4.22 Quiz.cpp

Go to the documentation of this file.

```
00001 // Quiz.cpp : This file contains the 'main' function. Program execution begins and ends there.
00002 //
00003
00004 #include <iostream>
00005
00006 #include "StringManipulators.h"
00007 #include "ReadingFile.h"
00008 #include "Player.h"
00009
00010
00011 int main()
00012
00013 {
00014
00015     Player A;
00016     Player B;
00017
00018     return 0;
00019 }
```

4.23 ReadingFile.cpp File Reference

```
#include <vector>
#include <string>
#include <string.h>
#include <iostream>
#include <random>
#include <chrono>
#include <sstream>
#include <fstream>
#include <algorithm>
#include <cstddef>
#include <cstdlib>
#include "StringManipulators.h"
#include "ReadingFile.h"
#include "Question.h"
```

Functions

- std::string values (int Number, int c)
- void create_questions ()
- std::vector< std::vector< std::string > > read_questions (std::string filename)
- std::vector< Question > shuffle (std::vector< Question > Tasks)
- void debug_questions (std::vector< std::vector< std::string > >vect)

4.23.1 Function Documentation

4.23.1.1 create_questions()

```
void create_questions ( )
```

It generates and creates nrows of question each quetion with a fixed ncol of property

Definition at line 73 of file ReadingFile.cpp.

4.23.1.2 debug_questions()

```
void debug_questions (
            std::vector< std::vector< std::string > > vect )
```

The function debugs a matrix showing of a set of questions and all its variables on the console

Parameters

std::vector<std::vector<string>>	vect vector of vector double printed to the output

Definition at line 151 of file ReadingFile.cpp.

4.23.1.3 read_questions()

```
std::vector< std::vector< std::string > > read_questions (
            std::string filename )
```

The function reads a matrix of random number from a file

Parameters

std::string	file_name a string variable storing the file name from which data is read

Returns

std::vector< std::vector<std::string>> data returns a vector of vector matrix containing the read matrix

Definition at line 110 of file ReadingFile.cpp.

4.23.1.4 shuffle()

```
std::vector< Question > shuffle (
            std::vector< Question > Tasks )
```

The function shuffles a matrix of questions and returns a shuffled version

Parameters

std::vector<std::vector<string>>	Tasks vector of vector to be shuffled

Returns

std::vector<std::vector<string>> Tasks new vector of vector after shuffling

Definition at line 141 of file ReadingFile.cpp.

4.23.1.5 values()

```
std::string values (
            int NumberofQuestion,
            int QuestionParameters )
```

This function helps allocate per question the variables/properities(c) for each question.

Parameters

int	NumberofQuestion number of rows/questions to be entered into the matrix container and saved to file (r where r1 = question 1 and properties, r2 =question 2 and properties.....)
int	QuestionParameters number of columns/options per question (where (c1 = quetion text),(c2-c5 = options [a, b, c, d]), the answer(c6) and the marks(c7))

Definition at line 21 of file ReadingFile.cpp.

4.24 ReadingFile.cpp

Go to the documentation of this file.

```
00001 #include <vector>
00002 #include<string>
00003 #include <string.h>
00004 #include <iostream>
00005 #include <vector>
00006 #include <random>
00007 #include <chrono>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <algorithm>
00011 #include <string>
00012 #include <cstddef>
00013 #include <cstdlib>
00014
00015 #include "StringManipulators.h"
00016 #include "ReadingFile.h"
00017 #include "Question.h"
00018
00019
00020
00021 std::string values(int Number, int c)
00022 {
00023     std::string value = " ";
00024     char temp = '"';
00025     char delim = ',';
00026     std::string line = " ";
00027
```

```
00028
00029         for (int C = 0; C < c; C++)
00030         {
00031
00032             if (C == 0)
00033             {
00034                 std::cout « "Please enter data for Question" « Number « '\n';
00035                 std::cout « "Type in Question text: \n"; cin » value;
00036                 line = temp + value + temp + delim;
00037             }
00038             if (C == 1)
00039             {
00040                 std::cout « "enter answer for option A : \n"; cin » value;
00041                 line += temp + value + temp + delim;
00042             }
00043             if (C == 2)
00044             {
00045                 std::cout « "enter answer for option B : \n"; cin » value;
00046                 line += temp + value + temp + delim;
00047             }
00048             if (C == 3)
00049             {
00050                 std::cout « "enter answer for option C : \n"; cin » value;
00051                 line += temp + value + temp + delim;
00052             }
00053             if (C == 4)
00054             {
00055                 std::cout « "enter answer for option D : \n"; cin » value;
00056                 line += temp + value + temp + delim ;
00057             }
00058             if (C == 5)
00059             {
00060                 std::cout « "Which is the answer is the correct option, select from option A,B,C,D :
    \n"; cin » value;
00061                 line += temp + value + delim;
00062             }
00063             if (C == 6)
00064             {
00065                 std::cout « "enter the mark obtained for answering correctly : \n"; cin » value;
00066                 line += temp + value + temp + delim + ';' + '\n';
00067             }
00068         }
00069
00070     return line;
00071 }
00072
00073 void create_questions()
00074 {
00075     std::string file_name = "";
00076     int NumberofQuestion = 0;
00077     int QuestionParameters = 7;
00078
00079     std::cout « "Please how many questions do you want printed/ saved  to the file ?\n";
00080     cin » NumberofQuestion;
00081     std::cout « "Please enter a file name to be created for questions to be printed / saved ";
00082     getline(cin, file_name );
00083     getline(cin, file_name);
00084     file_name += ".txt";
00085
00086
00087
00088
00089
00090     std::ofstream file(file_name); //ofstream + open
00091     if (file) //is it open?
00092     {
00093         int Qcounter = 0;
00094         std::string input = "";
00095
00096         for (int r = 0; r < NumberofQuestion; r++)
00097         {
00098             Qcounter = r + 1;
00099             file « '#' « Qcounter « '\n';
00100
00101                 input = values(r, QuestionParameters);
00102                 file « input ;
00103
00104         }
00105         file.close();
00106         std::cout « NumberofQuestion « " questions has been printed  to : the file " « file_name «
    '\n';
00107     }
00108 }
00109
00110 std::vector< std::vector<std::string» read_questions(std::string filename)
00111 {
00112     char question_start = '#';
```

```
00113     char question_end = ';';
00114     char delim = ',';
00115
00116     std::vector< std::vector<std::string» data;
00117     std::vector<std::string> row;
00118     std::ifstream in(filename);
00119
00120     for (std::string line; getline(in, line);)
00121     {
00122         line = Remove_question_no(line, question_start);
00123         if (line.empty()) continue;
00124
00125         line = trim(line);
00126         line = Replace(line, delim);
00127
00128         std::stringstream ss(line);
00129
00130         for (std::string d; ss » d; )
00131         {
00132             d = trim(d);
00133             row.push_back(d);
00134         }
00135
00136         data.push_back(row);
00137     }
00138     return data;
00139 }
00140
00141 std::vector<Question> shuffle(std::vector<Question> Tasks)
00142 {
00143     std::vector<Question> shuffled_pack;
00144     unsigned seed = std::chrono::system_clock::now().time_since_epoch().count();
00145     std::default_random_engine s(seed);
00146     std::shuffle(Tasks.begin(), Tasks.end(), s);
00147
00148     return Tasks;
00149 }
00150
00151 void debug_questions(std::vector< std::vector<std::string»vect)
00152 {
00153     for (auto i = 0; i < vect.size(); i++)
00154     {
00155         for (auto j = 0; j < vect[0].size(); j++)
00156         {
00157             std::cout « vect[i][j] « " ";
00158
00159         }
00160         std::cout « std::endl;
00161     }
00162 }
00163
00164
00165
```

4.25 ReadingFile.h File Reference

```
#include <vector>
#include "Question.h"
```

Functions

- std::string values (int NumberofQuestion, int QuestionParameters)
- void create_questions ()
- std::vector< std::vector< std::string > > read_questions (std::string filename)
- std::vector< Question > shuffle (std::vector< Question > Tasks)
- void debug_questions (std::vector< std::vector< std::string > >vect)

4.25.1 Function Documentation

4.25.1.1 create_questions()

```
void create_questions ( )
```

It generates and creates nrows of question each quetion with a fixed ncol of property

Definition at line 73 of file ReadingFile.cpp.

4.25.1.2 debug_questions()

```
void debug_questions (
            std::vector< std::vector< std::string > > vect )
```

The function debugs a matrix showing of a set of questions and all its variables on the console

Parameters

std::vector<std::vector<string>>	vect vector of vector double printed to the output

Definition at line 151 of file ReadingFile.cpp.

4.25.1.3 read_questions()

```
std::vector< std::vector< std::string > > read_questions (
            std::string filename )
```

The function reads a matrix of random number from a file

Parameters

std::string	file_name a string variable storing the file name from which data is read

Returns

std::vector< std::vector<std::string>> data returns a vector of vector matrix containing the read matrix

Definition at line 110 of file ReadingFile.cpp.

4.25.1.4 shuffle()

```
std::vector< Question > shuffle (
            std::vector< Question > Tasks )
```

The function shuffles a matrix of questions and returns a shuffled version

Parameters

std::vector<std::vector<string>>	Tasks vector of vector to be shuffled

Returns

std::vector<std::vector<string>> Tasks new vector of vector after shuffling

Definition at line 141 of file ReadingFile.cpp.

4.25.1.5 values()

```
std::string values (
            int NumberofQuestion,
            int QuestionParameters )
```

This function helps allocate per question the variables/properities(c) for each question.

Parameters

int	NumberofQuestion number of rows/questions to be entered into the matrix container and saved to file (r where r1 = question 1 and properties, r2 =question 2 and properties.....)
int	QuestionParameters number of columns/options per question (where (c1 = quetion text),(c2-c5 = options [a, b, c, d]), the answer(c6) and the marks(c7))

Definition at line 21 of file ReadingFile.cpp.

4.26 ReadingFile.h

Go to the documentation of this file.

```
00001 #ifndef READINGFILE_H
00002 #define READINGFILE_H
00003
00004 #include <vector>
00005 #include "Question.h"
00006
00007
00008
00013 std::string values(int  NumberofQuestion, int  QuestionParameters);
00014
00016 void create_questions();
00017
00023 std::vector< std::vector<std::string» read_questions(std::string filename);
00024
00025
00031 std::vector<Question> shuffle(std::vector<Question> Tasks);
00032
00037 void debug_questions(std::vector< std::vector<std::string»vect);
00038
00039 #endif
```

4.27 StringManipulators.cpp File Reference

```
#include <string>
#include <algorithm>
#include "StringManipulators.h"
```

Functions

- std::string Remove_question_no (std::string &line, char &question_start)
- std::string Replace (std::string &line, char &delim)
- std::string trim (const std::string s)

4.27.1 Function Documentation

4.27.1.1 Remove_question_no()

```
std::string Remove_question_no (
            std::string & line,
            char & question_start )
```

The function evaluates a line and extraxts line if the variable in the memory location @hash is encountered

Parameters

std::string	&line ampersand to address of line to be evaluated
char	& question_start variable of character that triggers an extraction of line

Returns

std::string Line returns a string with substring from the line

Definition at line 6 of file StringManipulators.cpp.

4.27.1.2 Replace()

```
std::string Replace (
            std::string & line,
            char & delim )
```

The function evaluates a line and replaces delimeters or separators with white space

Parameters

std::string&	line ampersand to address of line to be evaluated
char&	delim ampersand to address of with chararacter of delimiter to be evaluate

Returns

std::string Line returns a string with the edited line

Definition at line 16 of file StringManipulators.cpp.

4.27.1.3 trim()

```
std::string trim (
            const std::string s )
```

The function evaluates a line and returns a string of words without - "" or ; used as boundary marks within a question

Parameters

std::string	s string of texts to be trimmed

Returns

std::string s returns a string with the edited line

Definition at line 22 of file StringManipulators.cpp.

4.28 StringManipulators.cpp

Go to the documentation of this file.

```
00001 #include <string>
00002 #include <algorithm>
00003
00004 #include "StringManipulators.h"
00005
00006 std::string Remove_question_no(std::string& line, char& question_start)
00007 {
00008     const auto pos = line.find(question_start);
00009     if (pos != std::string::npos)
00010     {
00011         line = line.substr(0, pos);
00012     }
00013     return line;
00014 }
00015
00016 std::string Replace(std::string& line, char& delim)
00017 {
00018     replace(line.begin(), line.end(), delim, ' ');
00019     return line;
00020 }
00021
00022 std::string trim(const std::string s)
00023 {
00024     std::size_t first = s.find_first_not_of("\"");
00025     std::size_t last = s.find_last_not_of("\";\\");
00026     return s.substr(first, (last - first + 1));
00027 }
00028
00029
```

4.29 StringManipulators.h File Reference

Functions

- std::string Remove_question_no (std::string &line, char &question_start)
- std::string Replace (std::string &line, char &delim)
- std::string trim (const std::string s)

4.29.1 Function Documentation

4.29.1.1 Remove_question_no()

```
std::string Remove_question_no (
            std::string & line,
            char & question_start )
```

The function evaluates a line and extraxts line if the variable in the memory location @hash is encountered

Parameters

std::string	&line ampersand to address of line to be evaluated
char	& question_start variable of character that triggers an extraction of line

Returns

std::string Line returns a string with substring from the line

Definition at line 6 of file StringManipulators.cpp.

4.29.1.2 Replace()

```
std::string Replace (
            std::string & line,
            char & delim )
```

The function evaluates a line and replaces delimeters or separators with white space

Parameters

std::string&	line ampersand to address of line to be evaluated
char&	delim ampersand to address of with chararacter of delimiter to be evaluate

Returns

std::string Line returns a string with the edited line

Definition at line 16 of file StringManipulators.cpp.

4.29.1.3 trim()

```
std::string trim (
            const std::string s )
```

The function evaluates a line and returns a string of words without - "" or ; used as boundary marks within a question

Parameters

std::string	s string of texts to be trimmed

Returns

std::string s returns a string with the edited line

Definition at line 22 of file StringManipulators.cpp.

4.30 StringManipulators.h

Go to the documentation of this file.

```
00001 #ifndef STRINGMANIPULATORS_H
00002 #define STRINGMANIPULATORS_H
00003
00004
00005
00012 std::string Remove_question_no(std::string& line, char& question_start);
00013
00020 std::string Replace(std::string& line, char& delim);
00021
00027 std::string trim(const std::string s);
00028
00029
00030 #endif
00031
00032
00033
```

Index