

# СЕМЕСТРИАЛНА

# ДОМАШНА РАБОТА

# по

# Дисциплина „Базово програмиране“

# Тема: Beauty Contest

# 

# 

# 

# Изготвил: Иван Иванов Славов

# Проверил:

# Факултетен номер: 25621928

# Специалност: Изкуствен Интелект

# Курс: 1-ви курс

# Варна, 2025

Съдържание

# Директории Йерархия

## Директории

BeautyContest 4

logic **Error! Bookmark not defined.**

addition\_of\_contestants **Error! Bookmark not defined.**

addition\_of\_contestants\_menu.cpp 22

addition\_of\_contestants\_menu.h

categorisation **Error! Bookmark not defined.**

categorising.cpp 29

categorising.h

contestants\_manipulation **Error! Bookmark not defined.**

contestans\_sorters.cpp 33

contestans\_sorters.h

contestants\_generators.cpp 36

contestants\_generators.h

contestants\_manager.cpp 41

contestants\_manager.h

contestants\_printers.cpp 43

contestants\_printers.h

contestants\_searchers.cpp 48

contestants\_searchers.h

exit **Error! Bookmark not defined.**

exit\_logic.cpp 56

exit\_logic.h

file **Error! Bookmark not defined.**

file\_menu.cpp 59

file\_menu.h

search\_contestant **Error! Bookmark not defined.**

search\_show\_contestant\_menu.cpp 63

search\_show\_contestant\_menu.h

settings **Error! Bookmark not defined.**

settings\_menu\_manager.cpp 67

settings\_menu\_manager.h

show\_contestants **Error! Bookmark not defined.**

show\_contestants.cpp 70

show\_contestants.h

sorting **Error! Bookmark not defined.**

sorting\_menu.cpp 74

sorting\_menu.h

winner\_decision **Error! Bookmark not defined.**

winner\_decsision\_menu.cpp 78

winner\_decsision\_menu.h

utils **Error! Bookmark not defined.**

file\_manager **Error! Bookmark not defined.**

file\_manager.cpp 88

file\_manager.h

cin\_validators.cpp 81

cin\_validators.h

contestant.h 86

debug.cpp 87

debug.h

settings.h 92

utils.cpp 94

utils.h

visual\_enhancement.cpp 98

visual\_enhancement.h

beauty\_contest.cpp 7

beauty\_contest.h

block\_mangment.cpp 13

block\_mangment.h

# Класове Указател

## Класове Списък

Класове, структури, обединения и интерфейси с кратко описание:

**Contestants (A structure to hold all details for a single beauty contestant )** 4

# Файлове Списък

## Файлове Списък

Пълен списък с документирани файлове с кратко описание:

**BeautyContest/beauty\_contest.cpp (Main entry point for the Beauty Contest application )** 7

**BeautyContest/beauty\_contest.h**  12

**BeautyContest/block\_mangment.cpp (Implementation of text-based menu displays and title formatting )** 13

**BeautyContest/block\_mangment.h**  22

**BeautyContest/logic/addition\_of\_contestants/addition\_of\_contestants\_menu.cpp (Logic for adding new contestants (manual and random) )** 22

**BeautyContest/logic/addition\_of\_contestants/addition\_of\_contestants\_menu.h**  29

**BeautyContest/logic/categorisation/categorising.cpp (Implementation of age-based categorization logic )** 29

**BeautyContest/logic/categorisation/categorising.h**  32

**BeautyContest/logic/contestants\_manipulation/contestans\_sorters.cpp (Implementation of sorting algorithms (Bubble Sort) for contestants )** 33

**BeautyContest/logic/contestants\_manipulation/contestans\_sorters.h**  36

**BeautyContest/logic/contestants\_manipulation/contestants\_generators.cpp (Functions for generating random contestants and reading manual input )** 36

**BeautyContest/logic/contestants\_manipulation/contestants\_generators.h**  40

**BeautyContest/logic/contestants\_manipulation/contestants\_manager.cpp (Helper functions for swapping, duplicating, and resetting contestant data )** 41

**BeautyContest/logic/contestants\_manipulation/contestants\_manager.h**  43

**BeautyContest/logic/contestants\_manipulation/contestants\_printers.cpp (Functions responsible for printing contestant details to the console )** 43

**BeautyContest/logic/contestants\_manipulation/contestants\_printers.h**  47

**BeautyContest/logic/contestants\_manipulation/contestants\_searchers.cpp (Implementation of searching algorithms (by ID, Name, Age, etc.) )** 48

**BeautyContest/logic/contestants\_manipulation/contestants\_searchers.h**  56

**BeautyContest/logic/exit/exit\_logic.cpp (Handles the exit sequence and prompts to save unsaved changes )** 56

**BeautyContest/logic/exit/exit\_logic.h**  59

**BeautyContest/logic/file/file\_menu.cpp (Implementation of the File Manager menu interactions )** 59

**BeautyContest/logic/file/file\_menu.h**  63

**BeautyContest/logic/search\_contestant/search\_show\_contestant\_menu.cpp (Logic for the Search menu and displaying search results )** 63

**BeautyContest/logic/search\_contestant/search\_show\_contestant\_menu.h**  67

**BeautyContest/logic/settings/settings\_menu\_manager.cpp (Handles user interactions for changing program settings (Debug, AutoSave, etc.) )** 67

**BeautyContest/logic/settings/settings\_menu\_manager.h**  69

**BeautyContest/logic/show\_contestants/show\_contestants.cpp (Logic for displaying various lists of contestants (All, Winners, Categories) )** 70

**BeautyContest/logic/show\_contestants/show\_contestants.h**  73

**BeautyContest/logic/sorting/sorting\_menu.cpp (Implementation of the Sorting menu logic )** 74

**BeautyContest/logic/sorting/sorting\_menu.h**  77

**BeautyContest/logic/winner\_decision/winner\_decsision\_menu.cpp (Logic for calculating and displaying the winners )** 78

**BeautyContest/logic/winner\_decision/winner\_decsision\_menu.h**  80

**BeautyContest/utils/cin\_validators.cpp (Helper functions to validate user input from the console )** 81

**BeautyContest/utils/cin\_validators.h**  86

**BeautyContest/utils/contestant.h (Defines the Contestants structure and related data fields )** 86

**BeautyContest/utils/debug.cpp (Debugging utilities for printing array values and status messages )** 87

**BeautyContest/utils/debug.h**  88

**BeautyContest/utils/settings.h (Global configuration constants and settings flags for the application )** 92

**BeautyContest/utils/utils.cpp (General utility functions for string manipulation and array comparison )** 94

**BeautyContest/utils/utils.h**  97

**BeautyContest/utils/visual\_enhancement.cpp (Console visual helpers, such as clearing the screen )** 98

**BeautyContest/utils/visual\_enhancement.h**  99

**BeautyContest/utils/file\_manager/file\_manager.cpp (Low-level file I/O operations (saving and loading binary data) )** 88

**BeautyContest/utils/file\_manager/file\_manager.h**  91

# Директории Документация

## BeautyContest Директория Съдържание

### Директории

### директория utilsФайлове

* файл beauty\_contest.cpp

*Main entry point for the Beauty Contest application.*

* файл **beauty\_contest.h**
* файл block\_mangment.cpp

*Implementation of text-based menu displays and title formatting.*

* файл **block\_mangment.h**

### Подробно описание

Hallo this is my semestral homework assignment. Use whatever suits you.

# Класове Документация

## Contestants Структура Препратка

A structure to hold all details for a single beauty contestant.

#include <contestant.h>

### Общодостъпни атрибути

* int ID

*Unique identifier.*

* bool isObjectUsed = false

*Identifier for active places in the strcut array. True -> active, false -> inactive.*

* char name [MAXNAMECHARS]

*Contestant's full name.*

* int age = 0

*Contestant's age.*

* bool isWoman = false

*Identifier for gender. true -> woman, false -> man.*

* double hipCirc = 0.0

*Contestant's hip circumference.*

* double shoulderCirc = 0.0

*Contestant's shoulder circumference.*

* double calfCirc = 0.0

*Contestant's calf circumference.*

* double neckCirc = 0.0

*Contestant's neck circumference.*

* double points = 0.0

*Contestant's points.*

### Подробно описание

A structure to hold all details for a single beauty contestant.

Виж дефиницията във файла contestant.h ред 9.

### Член данни Документация

#### int Contestants::age = 0

Contestant's age.

Виж дефиницията във файла contestant.h ред 21.

#### double Contestants::calfCirc = 0.0

Contestant's calf circumference.

Виж дефиницията във файла contestant.h ред 33.

#### double Contestants::hipCirc = 0.0

Contestant's hip circumference.

Виж дефиницията във файла contestant.h ред 27.

#### int Contestants::ID

Unique identifier.

Виж дефиницията във файла contestant.h ред 12.

#### bool Contestants::isObjectUsed = false

Identifier for active places in the strcut array. True -> active, false -> inactive.

Виж дефиницията във файла contestant.h ред 15.

#### bool Contestants::isWoman = false

Identifier for gender. true -> woman, false -> man.

Виж дефиницията във файла contestant.h ред 24.

#### char Contestants::name[MAXNAMECHARS]

Contestant's full name.

Виж дефиницията във файла contestant.h ред 18.

#### double Contestants::neckCirc = 0.0

Contestant's neck circumference.

Виж дефиницията във файла contestant.h ред 36.

#### double Contestants::points = 0.0

Contestant's points.

Виж дефиницията във файла contestant.h ред 39.

#### double Contestants::shoulderCirc = 0.0

Contestant's shoulder circumference.

Виж дефиницията във файла contestant.h ред 30.

#### Документация за структура генериран от следният файл:

* BeautyContest/utils/contestant.h

# Файлове Документация

## BeautyContest/beauty\_contest.cpp Файл Справка

Main entry point for the Beauty Contest application.

#include "beauty\_contest.h"

### Функции

* int main ()

*The main entry point of the application.*

### Подробно описание

Main entry point for the Beauty Contest application.

This file contains the main loop, menu navigation, and program initialization.

Виж дефиницията във файла beauty\_contest.cpp.

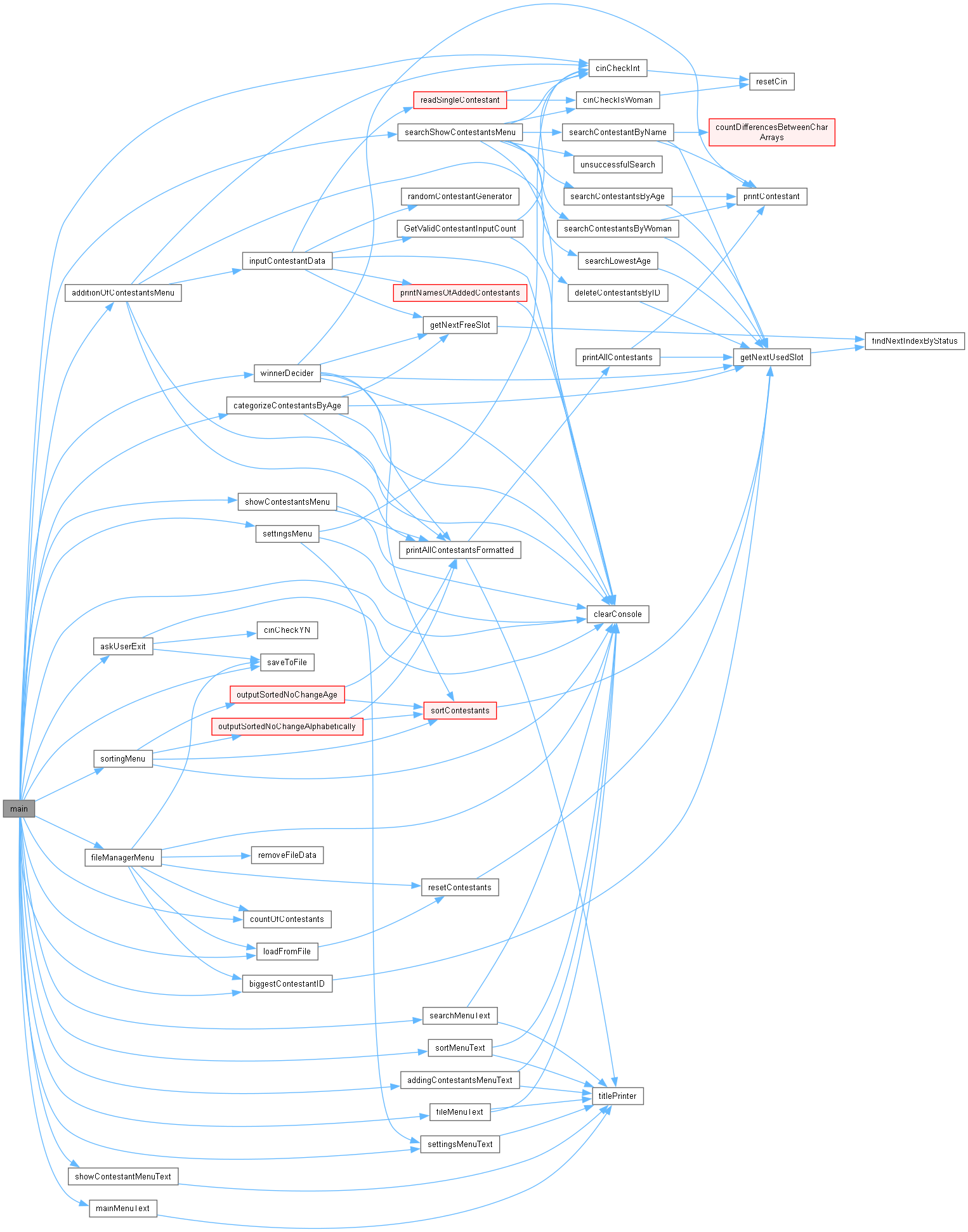
### Функции Документация

#### int main ()

The main entry point of the application.

Виж дефиницията във файла beauty\_contest.cpp ред 8.

Граф с извикванията за тази функция:



## beauty\_contest.cpp

Вижте документацията за този файл.

00001

00004

00005 #include "beauty\_contest.h"

00006

00008 int main() {

00009 clearConsole(CLEARCONSOLE);

00010 srand(time(0));

00011

00012 int leftSpaces = MAXCONTESTANTS;

00013 bool freeSpaces[MAXCONTESTANTS];

00014 int IDcounter = 1;

00015

00016 bool winnersDecided = false;

00017 bool madeCategories = false;

00018 bool savedChanages = true;

00019

00020

00021 Contestants contestant[MAXCONTESTANTS];

00022 if(loadFromFile(contestant, "zfiles/auto\_save.dat")){

00023 cout << "Successfully loaded form auto\_save.dat!" << endl;

00024

00025 IDcounter = biggestContestantID(contestant) + 1;

00026 deBugInfo("SYSTEM: IDcounter: " << IDcounter << endl);

00027

00028 leftSpaces = countOfContestants(contestant);

00029 deBugInfo("SYSTEM: leftSpaces: " << leftSpaces << endl);

00030 }

00031

00032

00033 Contestants category14\_16[MAXCONTESTANTS];

00034 Contestants category17\_19[MAXCONTESTANTS];

00035 Contestants category20\_22[MAXCONTESTANTS];

00036 Contestants category23\_25[MAXCONTESTANTS];

00037

00038 Contestants winners[MAXCONTESTANTS];

00039

00040 int menuChoice;

00041

00042

00043 bool running = true;

00044 do {

00045

00046 if(AUTOSAVE){

00047 if (saveToFile(contestant, "zfiles/auto\_save.dat")){

00048 cout << "Changes were auto saved!" << endl;

00049 savedChanages = true;

00050 }

00051 }

00052

00053 mainMenuText(winnersDecided, madeCategories);

00054 menuChoice = cinCheckInt("Type a number to chose your action: ");

00055

00056 switch (menuChoice) {

00057

00058 case 0:

00059 {

00060 clearConsole(CLEARCONSOLE);

00061

00062

00063 if (askUserExit(contestant, savedChanages)){

00064 running = false;

00065 }

00066

00067

00068 break;

00069 }

00070 case 1:

00071 {

00072 clearConsole(CLEARCONSOLE);

00073

00074

00075

00076 int leftSpacesBefore = leftSpaces;

00077 int leftSpacesAfter;

00078

00079 addingContestantsMenuText();

00080 menuChoice = cinCheckInt("Type a number to chose your action: ");

00081 cout << endl;

00082

00083

00084 additionOfContestantsMenu(contestant, leftSpaces, IDcounter, menuChoice);

00085

00086

00087 leftSpacesAfter = leftSpaces;

00088 bool contestantsWereChanged = leftSpacesBefore != leftSpacesAfter;

00089 if (contestantsWereChanged){

00090

00091 if(AUTOCATEGORIZE){

00092 categorizeContestantsByAge(contestant, category14\_16, category17\_19, category20\_22, category23\_25);

00093 madeCategories = true;

00094 } else {

00095 madeCategories = false;

00096 }

00097 savedChanages = false;

00098 winnersDecided = false;

00099

00100 }

00101 break;

00102 }

00103

00104 case 2:

00105 {

00106 clearConsole(CLEARCONSOLE);

00107

00108 bool showMore = madeCategories || winnersDecided;

00109 if (showMore){

00110 showContestantMenuText(winnersDecided, madeCategories);

00111 menuChoice = cinCheckInt("Type a number to chose your action: ");

00112 cout << endl;

00113 }

00114

00115 if (!showMore) menuChoice = 1;

00116 showContestantsMenu(contestant, winners, category14\_16, category17\_19, category20\_22, category23\_25, menuChoice, madeCategories, winnersDecided);

00117

00118 break;

00119 }

00120

00121 case 3:

00122 {

00123 clearConsole(CLEARCONSOLE);

00124

00125 searchMenuText();

00126 menuChoice = cinCheckInt("Type a number to chose your action: ");

00127 cout << endl;

00128

00129

00130 searchShowContestantsMenu(contestant, menuChoice);

00131

00132

00133 break;

00134 }

00135

00136 case 4:

00137 {

00138 clearConsole(CLEARCONSOLE);

00139

00140 sortMenuText();

00141 menuChoice = cinCheckInt("Type a number to chose your action: ");

00142 cout << endl;

00143

00144

00145 sortingMenu(contestant, menuChoice, savedChanages);

00146

00147

00148 bool changesWereMade = menuChoice == 1 || menuChoice == 2 || menuChoice == 3;

00149 if(changesWereMade){

00150 savedChanages = false;

00151 }

00152

00153 break;

00154 }

00155

00156 case 5:

00157 {

00158 clearConsole(CLEARCONSOLE);

00159

00160

00161 categorizeContestantsByAge(contestant, category14\_16, category17\_19, category20\_22, category23\_25);

00162

00163

00164 madeCategories = true;

00165

00166 break;

00167 }

00168

00169 case 6:

00170 {

00171 clearConsole(CLEARCONSOLE);

00172

00173

00174 winnerDecider(contestant, winners);

00175

00176

00177 winnersDecided = true;

00178 break;

00179 }

00180

00181 case 7:

00182 {

00183 clearConsole(CLEARCONSOLE);

00184

00185 fileMenuText();

00186 menuChoice = cinCheckInt("Type a number to chose your action: ");

00187 cout << endl;

00188

00189

00190 fileManagerMenu(contestant, menuChoice, savedChanages, madeCategories, winnersDecided, IDcounter, leftSpaces);

00191

00192

00193 break;

00194 }

00195

00196 case 8:

00197 {

00198 clearConsole(CLEARCONSOLE);

00199

00200 settingsMenuText();

00201 menuChoice = cinCheckInt("Type a number to chose your action: ");

00202 cout << endl;

00203

00204

00205 settingsMenu(menuChoice);

00206

00207

00208 break;

00209 }

00210

00211 default:

00212 {

00213 deBugInfo("ERROR: Expectednumber from 0 to 6 got: "<< menuChoice << endl);

00214 cout<< "Please enter a valid option!";

00215 break;

00216 }

00217 }

00218 } while (running);

00219

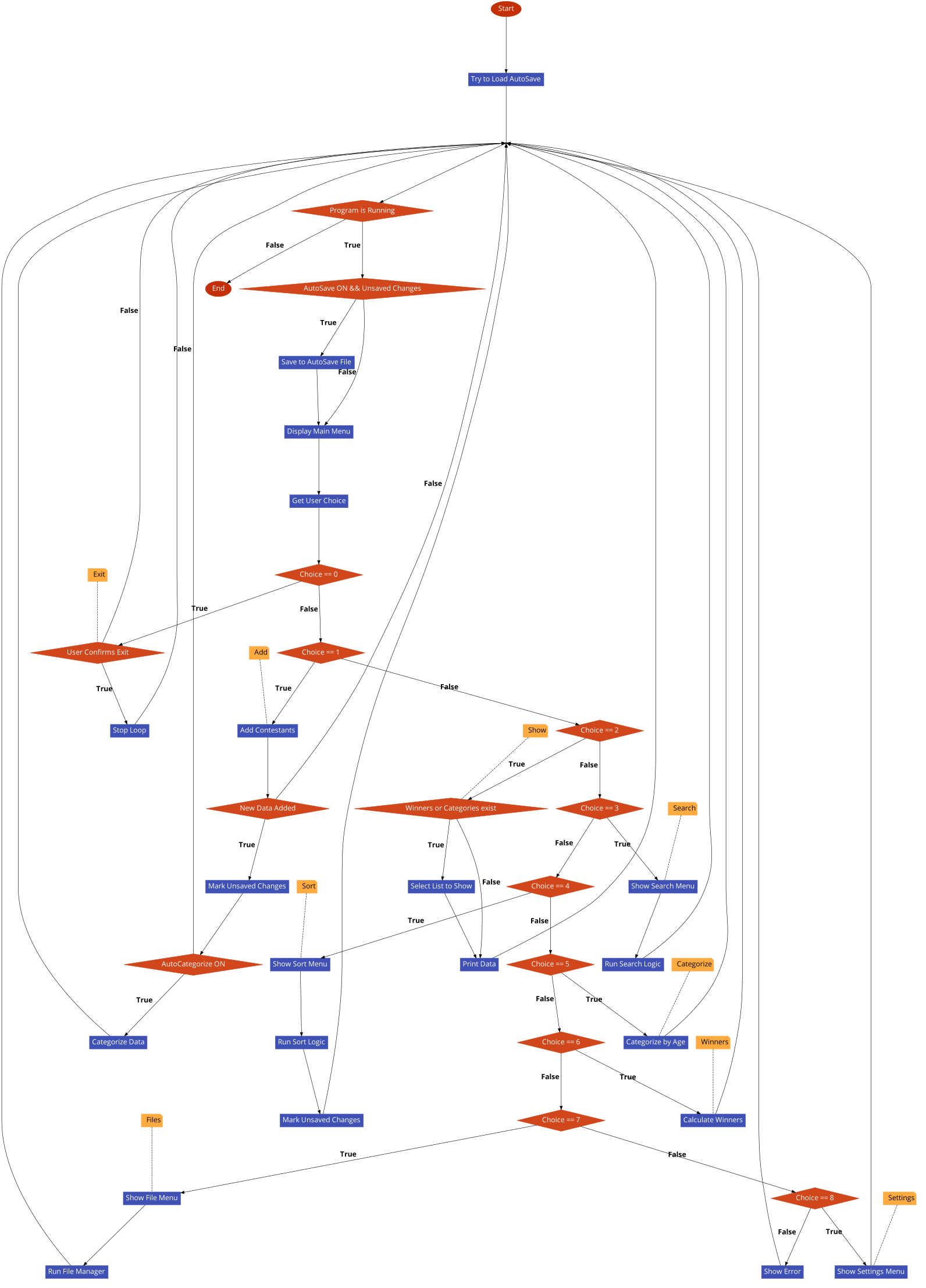
00220 return 0;

00221

00222 }

00223

00224

00225 

## beauty\_contest.h

00001 #pragma once

00002

00003 #include <iostream>

00004 #include <string>

00005 #include <algorithm>

00006 using namespace std;

00007

00008 #include "block\_mangment.h"

00009

00010 #include "logic/exit/exit\_logic.h"

00011 #include "logic/addition\_of\_contestants/addition\_of\_contestants\_menu.h"

00012 #include "logic/show\_contestants/show\_contestants.h"

00013 #include "logic/sorting/sorting\_menu.h"

00014 #include "logic/search\_contestant/search\_show\_contestant\_menu.h"

00015 #include "logic/contestants\_manipulation/contestants\_printers.h"

00016 #include "logic/categorisation/categorising.h"

00017 #include "logic/winner\_decision/winner\_decsision\_menu.h"

00018 #include "logic/file/file\_menu.h"

00019 #include "logic/settings/settings\_menu\_manager.h"

00020

00021 #include "utils/utils.h"

00022 #include "utils/debug.h"

00023 #include "utils/contestant.h"

00024 #include "utils/settings.h"

00025 #include "utils/visual\_enhancement.h"

00026 #include "utils/cin\_validators.h"

## BeautyContest/block\_mangment.cpp Файл Справка

Implementation of text-based menu displays and title formatting.

#include "block\_mangment.h"

### Функции

* void titlePrinter (string title, int width, char filler=' ')

*Helper for printing out beautiful filler around the title in the center.*

* void mainMenuText (bool winnersDecided, bool madeCategories)

*Outputs Main Menu.*

* void addingContestantsMenuText ()

*Outputs the addition of contestants submenu.*

* void showContestantMenuText (bool winnersDecided, bool madeCategories)

*Outputs the show submenu. Shows if different patterns depending on winners and categories.*

* void searchMenuText ()

*Outputs the search submenu.*

* void sortMenuText ()

*Outputs the sort submenu.*

* void fileMenuText ()

*Outputs the file submenu.*

* void settingsMenuText ()

*Outputs the file submenu. Shows the settings toggle state.*

* void mainMenuTextBASEEDITION ()
* void searchMenuTextBASEEDITION ()
* void winnersMenuText ()

*Outputs the winners submenu.*

### Подробно описание

Implementation of text-based menu displays and formatting.

Виж дефиницията във файла block\_mangment.cpp.

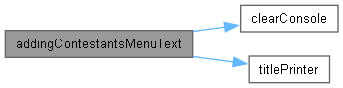
### Функции Документация

#### void addingContestantsMenuText ()

Outputs the addition of contestants submenu.

Виж дефиницията във файла block\_mangment.cpp ред 44.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:

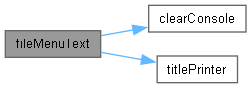


#### void fileMenuText ()

Outputs the file submenu.

Виж дефиницията във файла block\_mangment.cpp ред 100.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void mainMenuText (bool winnersDecided = false, bool madeCategories = false)

Outputs Main Menu.

##### Аргументи

|  |  |
| --- | --- |
| *winnersDecided* | If winners were decided |
| *madeCategories* | If categories were made |

Виж дефиницията във файла block\_mangment.cpp ред 9.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void mainMenuTextBASEEDITION ()

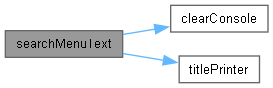
Виж дефиницията във файла block\_mangment.cpp ред 166.

#### void searchMenuText ()

Outputs the search submenu.

Виж дефиницията във файла block\_mangment.cpp ред 68.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void searchMenuTextBASEEDITION ()

Виж дефиницията във файла block\_mangment.cpp ред 186.

#### void settingsMenuText ()

Outputs the file submenu. Shows the settings toggle state.

Виж дефиницията във файла block\_mangment.cpp ред 116.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void showContestantMenuText (bool winnersDecided = true, bool madeCategories = true)

Outputs the show submenu. Shows if different patterns depending on winners and categories.

##### Аргументи

|  |  |
| --- | --- |
| *winnersDecided* | If winners were decided |
| *madeCategories* | If categories were made |

Виж дефиницията във файла block\_mangment.cpp ред 57.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:

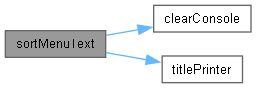


#### void sortMenuText ()

Outputs the sort submenu.

Виж дефиницията във файла block\_mangment.cpp ред 83.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void titlePrinter (string title, int width, char filler)

Helper for printing out beautiful filler around the title in the center.

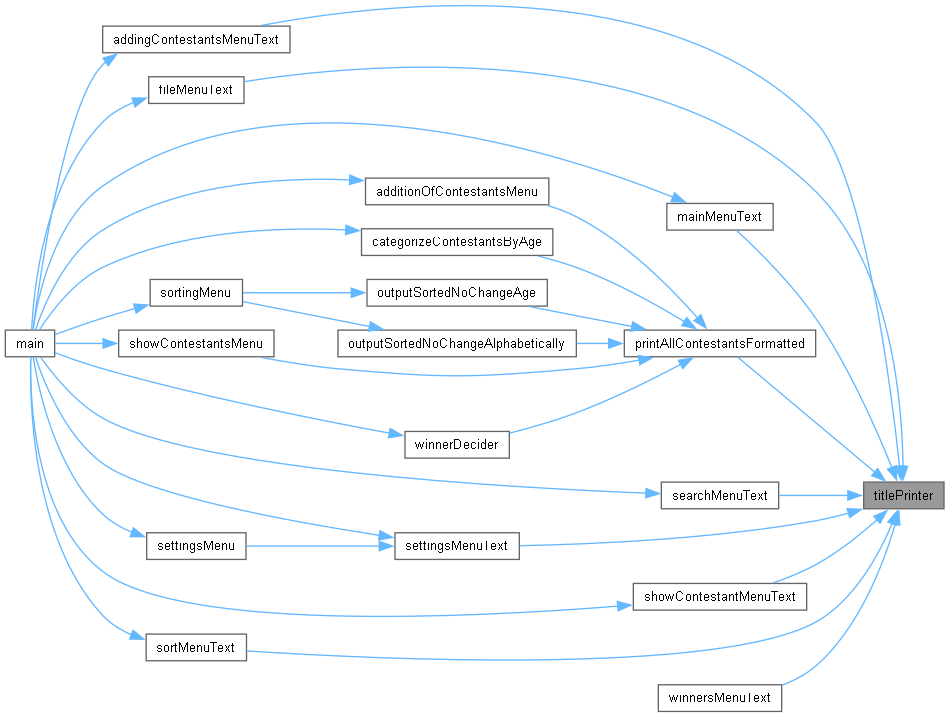
* + Puts the title in the center and fills the empty space with filler. Accounts when title cant centered and puts one extra filler at end. If title is too big: debug warning alarms and puts only one filler at both ends.

##### Аргументи

|  |  |
| --- | --- |
| *title* | The message you want to be seen. |
| *width* | What space are working with. |
| *filler* | The char that will be around the title. |

Виж дефиницията във файла block\_mangment.cpp ред 130.

Граф на извикванията за тази функция:

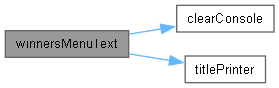


#### void winnersMenuText ()

Outputs the winners submenu.

Виж дефиницията във файла block\_mangment.cpp ред 198.

Граф с извикванията за тази функция:



## block\_mangment.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "block\_mangment.h"

00005

00006 void titlePrinter(string title, int width, char filler = ' ');

00007

00008

00009 void mainMenuText(bool winnersDecided, bool madeCategories) {

00010 cout << endl;

00011

00012 titlePrinter("BEAUTY CONTEST", TITLEWIDTH, '-');

00013 cout <<

00014 "1. Contestant\n"

00015 " 1.1 Manual adding\n"

00016 " 1.2 Random values\n"

00017 " 1.3 Remove contestant\n"

00018 << (winnersDecided || madeCategories ? "2. Show\n" : "2. Show all contestants\n")

00019 << (winnersDecided || madeCategories ? " 2.1 All contestants\n" : "")

00020 << (winnersDecided ? " 2.2 Winners\n" : "")

00021 << ( madeCategories ? " 2.3 Categories\n" : "") <<

00022 "3. Search\n"

00023 " 3.1 The lowest age\n"

00024 " 3.2 Age\n"

00025 " 3.3 Name\n"

00026 " 3.4 Gender\n"

00027 "4. Sort\n"

00028 " 4.1 Sort by Age ascending\n"

00029 " 4.2 Sort by Name\n"

00030 " 4.3 Sort by ID\n"

00031 " 4.4 Output sorted without change by age\n"

00032 " 4.5 Output sorted without change alphabetically\n"

00033 "5. Categorize contestants in age groups\n"

00034 "6. Decide winners\n"

00035 "7. File\n"

00036 " 7.1 Export\n"

00037 " 7.2 Import\n"

00038 " 7.3 Reset...\n"

00039 "8. Settings...\n"

00040 "0. Exit"

00041 << endl;

00042 }

00043

00044 void addingContestantsMenuText(){

00045 clearConsole();

00046 cout << endl;

00047

00048 titlePrinter("CONTESTANT", TITLEWIDTH, '-');

00049 cout <<

00050 " 1. Manual adding\n"

00051 " 2. Random values\n"

00052 " 3. Remove contestant\n"

00053 " 0. Back\n"

00054 << endl;

00055 }

00056

00057 void showContestantMenuText(bool winnersDecided, bool madeCategories){

00058 bool large = winnersDecided || madeCategories;

00059 if (large) titlePrinter("SHOW", TITLEWIDTH, '-');

00060 cout <<

00061 (large ? " 1. All contestants\n" : "")

00062 << (winnersDecided ? " 2. Winners\n" : "")

00063 << (madeCategories ? " 3. Categories\n" : "")

00064 << (large ? " 0. Back" : "")

00065 << endl;

00066 }

00067

00068 void searchMenuText(){

00069 clearConsole();

00070 cout << endl;

00071

00072 titlePrinter("SEARCH", TITLEWIDTH, '-');

00073 cout <<

00074 " 1. The lowest age\n"

00075 " 2. Age\n"

00076 " 3. Name\n"

00077 " 4. Gender\n"

00078 " 0. Back\n"

00079 << endl;

00080 }

00081

00082

00083 void sortMenuText(){

00084 clearConsole();

00085 cout << endl;

00086

00087 titlePrinter("SORT", TITLEWIDTH, '-');

00088 cout <<

00089 " 1. Sort by Age ascending\n"

00090 " 2. Sort by Name\n"

00091 " 3. Sort by ID\n"

00092 " 4. Output sorted without change by age\n"

00093 " 5. Output sorted without change alphabetically\n"

00094 " 0. Back\n"

00095 << endl;

00096 }

00097

00098

00099

00100 void fileMenuText(){

00101 clearConsole();

00102 cout << endl;

00103

00104 titlePrinter("FILE", TITLEWIDTH, '-');

00105 cout <<

00106 " 1. Export\n"

00107 " 2. Import\n"

00108 " 3. Reset auto\_save.dat\n"

00109 " 4. Reset contestants.dat\n"

00110 " 5. Reset Contestants struct\n"

00111 " 6. Reset EVERYTHING\n"

00112 " 0. Back\n"

00113 << endl;

00114 }

00115

00116 void settingsMenuText(){

00117

00118 titlePrinter("SETTINGS", TITLEWIDTH, '-');

00119 cout <<

00120 " 1. Debugmode = " << (DEBUGMODE ? "true" : "false") << "\n"

00121 " 2. Clear console = " << (CLEARCONSOLE ? "true" : "false") << "\n"

00122 " 3. Auto save = " << (AUTOSAVE ? "true" : "false") << "\n"

00123 " 4. Auto categorize = " << (AUTOCATEGORIZE ? "true" : "false") << "\n"

00124 " 0. Back\n"

00125 << endl;

00126 }

00127

00128

00129

00130 void titlePrinter(string title, int width, char filler){

00131

00132

00133 int sizeOfMessage = title.size();

00134 int fillerAroundMessage = (width - sizeOfMessage) / 2;

00135

00136 bool isTitleEven = sizeOfMessage % 2 == 0;

00137 bool isWidthEven = width % 2 == 0;

00138

00139 bool extraFillerAtEndNeeded = isTitleEven ^ isWidthEven;

00140

00141

00142 bool atLeastOneFiller = fillerAroundMessage < 1;

00143 if(atLeastOneFiller)

00144 {

00145 deBugInfo("WARNING: Message is too large with " << fillerAroundMessage << endl);

00146 fillerAroundMessage = 1;

00147 extraFillerAtEndNeeded = false;

00148 }

00149

00150

00151 //fillers at start

00152 for(int i = 0; i < fillerAroundMessage; i++){

00153 cout << filler;

00154 }

00155

00156 cout << title;

00157

00158 //fillers at end

00159 for(int i = 0; i < fillerAroundMessage + extraFillerAtEndNeeded; i++){

00160 cout << filler;

00161 }

00162 cout << endl;

00163 }

00164

00165 //Outdated

00166 void mainMenuTextBASEEDITION() {

00167 cout << endl;

00168

00169 titlePrinter("BEAUTY CONTEST", TITLEWIDTH, '-');

00170 cout <<

00171 "1. Add new contestant\n"

00172 " 1.1 Manual adding\n"

00173 " 1.2 Random values\n"

00174 "2. Show all contestants\n"

00175 "3. Search\n"

00176 " 3.1 The lowest age\n"

00177 " 3.2 Name\n"

00178 "4. Sort contestants by youngest to oldest (No output)\n"

00179 "5. File\n"

00180 " 5.1 Export\n"

00181 " 5.2 Import\n"

00182 "0. Exit"

00183 << endl;

00184 }

00185

00186 void searchMenuTextBASEEDITION(){

00187 clearConsole(CLEARCONSOLE);

00188 cout << endl;

00189

00190 titlePrinter("SEARCH", TITLEWIDTH, '-');

00191 cout <<

00192 " 1. Lowest age \n"

00193 " 2. Name\n"

00194 " 0. Back\n"

00195 << endl;

00196 }

00197

00198 void winnersMenuText(){

00199 clearConsole(CLEARCONSOLE);

00200 cout << endl;

00201

00202 titlePrinter("Decide winners", TITLEWIDTH, '-');

00203 cout <<

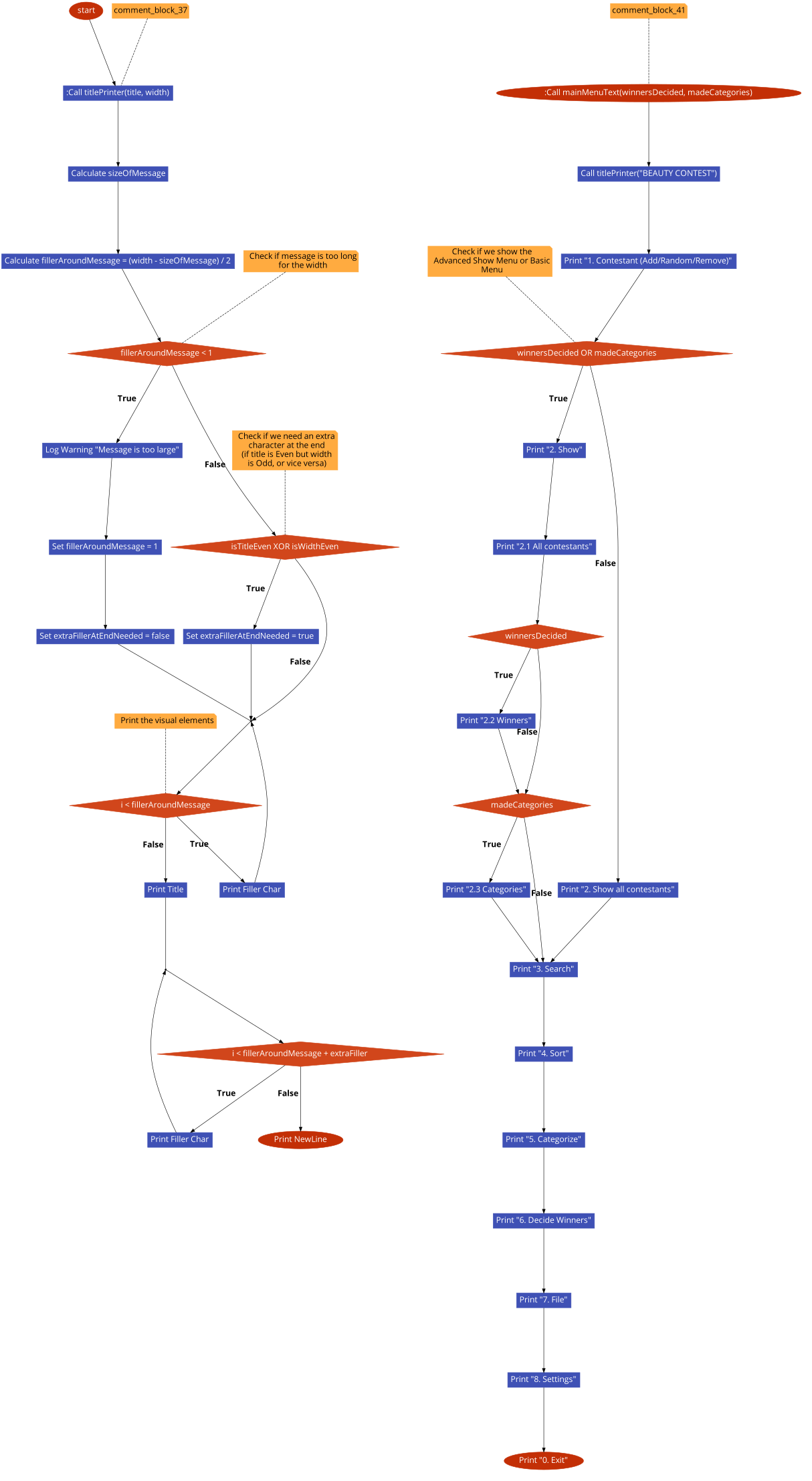
00204 " 1. Scores between 0.54 - 0.62\n"

00205 " 2. Output Scores > 0.63 ascending, then < 0.53\n"

00206 " 0. Back\n"

00207 << endl;

00208 }



## block\_mangment.h

00001 #pragma once

00002 #include <iostream>

00003 #include <string>

00004 #include <cmath>

00005

00006 #include "utils/settings.h"

00007 #include "utils/visual\_enhancement.h"

00008 #include "utils/debug.h"

00009

00010 using namespace std;

00011

00012

00013 //MAIN MENU

00014

00018 void mainMenuText(bool winnersDecided = false, bool madeCategories = false);

00019

00020

00021 //SUBMENUS

00022

00026 void showContestantMenuText(bool winnersDecided = true, bool madeCategories = true);

00027

00029 void searchMenuText();

00030

00032 void sortMenuText();

00033

00035 void winnersMenuText();

00036

00038 void addingContestantsMenuText();

00039

00041 void winnersMenuText();

00042

00044 void fileMenuText();

00045

00047 void settingsMenuText();

00048

00049

00050 //MENUS HELPER/S

00051

00060 void titlePrinter(string title, int width, char filler);

## BeautyContest/logic/addition\_of\_contestants/addition\_of\_contestants\_menu.cpp Файл Справка

Logic for adding new contestants (manual and random).

#include "addition\_of\_contestants\_menu.h"

### Функции

* int GetValidContestantInputCount (int &leftSpaces)

*Gets user to type a value within the left spaces for contestants.*

* void inputContestantData (int &leftSpaces, int &IDcounter, Contestants contestant[], bool random)

*Manager for random and single contestant initialisation.*

* void printNamesOfAddedContestants (string savedNames[])

*Output all the saved names to the user.*

* void additionOfContestantsMenu (Contestants contestant[], int &leftSpaces, int &IDcounter, int &menuChoice)

*Manges logic for contestnat menu, adding manually, randomly and removing.*

### Подробно описание

Logic for adding new contestants (manual and random).

Виж дефиницията във файла addition\_of\_contestants\_menu.cpp.

### Функции Документация

#### void additionOfContestantsMenu (Contestants contestant[], int & leftSpaces, int & IDcounter, int & menuChoice)

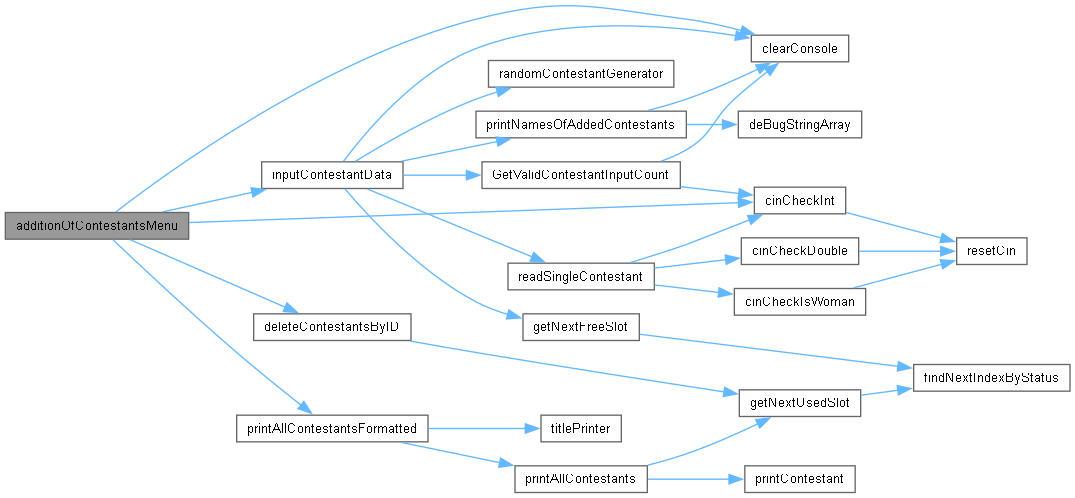
Manges logic for contestnat menu, adding manually, randomly and removing.

##### Аргументи

|  |  |
| --- | --- |
| *contestnt* | Contestants struct you want ot change |
| *leftSpaces* | Spaces left with tag isObjectUsed = false; |
| *IDcounter* | From what id to start adding |
| *menuChoice* | What the user choose to do |

Виж дефиницията във файла addition\_of\_contestants\_menu.cpp ред 24.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### int GetValidContestantInputCount (int & leftSpaces)

Gets user to type a value within the left spaces for contestants.

##### Аргументи

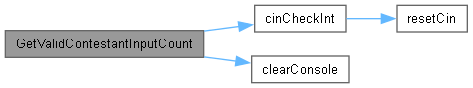
|  |  |
| --- | --- |
| *leftSpaces* | left space for use in the Contestants struct |

##### Връща

a valid number of contestants to add to the struct

Виж дефиницията във файла addition\_of\_contestants\_menu.cpp ред 74.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void inputContestantData (int & leftSpaces, int & IDcounter, Contestants contestant[], bool random)

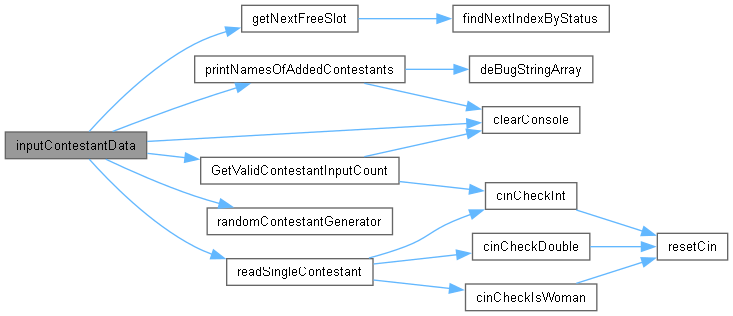
Manager for random and single contestant initialisation.

##### Аргументи

|  |  |
| --- | --- |
| *leftSpaces* | Left free spaces in COntestants struct |
| *IDcounter* | The last ishued ID |
| *contestant* | Array of all contestants |
| *random* | If you want manual addition of contestants or random values |

Виж дефиницията във файла addition\_of\_contestants\_menu.cpp ред 93.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void printNamesOfAddedContestants (string savedNames[])

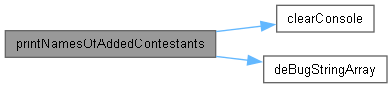
Output all the saved names to the user.

##### Аргументи

|  |  |
| --- | --- |
| *savedNames* | string array of all saved names |

Виж дефиницията във файла addition\_of\_contestants\_menu.cpp ред 132.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



## addition\_of\_contestants\_menu.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "addition\_of\_contestants\_menu.h"

00005

00006

00010 int GetValidContestantInputCount(int& leftSpaces);

00011

00017 void inputContestantData(int& leftSpaces, int& IDcounter, Contestants contestant[], bool random);

00018

00021 void printNamesOfAddedContestants(string savedNames[]);

00022

00023

00024 void additionOfContestantsMenu(Contestants contestant[], int& leftSpaces, int& IDcounter, int& menuChoice){

00025 clearConsole(CLEARCONSOLE);

00026

00027

00028 switch (menuChoice){

00029 case 0:

00030 {

00031 clearConsole();

00032 return;

00033 break;

00034 }

00035 case 1:

00036 {

00037 inputContestantData(leftSpaces, IDcounter, contestant, false);

00038 break;

00039 }

00040 case 2:

00041 {

00042 inputContestantData(leftSpaces, IDcounter, contestant, true);

00043 break;

00044 }

00045 case 3:

00046 {

00047 printAllContestantsFormatted(contestant, "ALL CONTESTANTS");

00048 cout << endl;

00049

00050 int idToRemove = 0;

00051 do{

00052 if (idToRemove != 0) {

00053 deleteContestantsByID(contestant, idToRemove);

00054 leftSpaces++;

00055 }

00056 idToRemove = cinCheckInt("Enter ID of contestants to remove or 0 to cancel: ");

00057 } while(idToRemove != 0);

00058

00059 clearConsole(CLEARCONSOLE);

00060 break;

00061 }

00062 default:

00063 {

00064 clearConsole();

00065 cout << "Invalid option!" << endl;

00066 deBugInfo("ERROR: expected from 0 - 3 got: " << menuChoice);

00067 break;

00068 }

00069

00070 }

00071 }

00072

00073

00074 int GetValidContestantInputCount(int& leftSpaces){

00075

00076 bool isValid = false;

00077 do{

00078 cout << "Spaces left: " << leftSpaces << endl;

00079

00080 int contestantsToAdd = cinCheckInt("How many contestants do you want to add: ");

00081

00082 isValid = contestantsToAdd > leftSpaces;

00083 if (isValid) {

00084 clearConsole(CLEARCONSOLE);

00085 cout << "Too many contestants!" << endl << endl;

00086 } else {

00087 return contestantsToAdd;

00088 }

00089 } while(isValid);

00090 return 0;

00091 }

00092

00093 void inputContestantData(int& leftSpaces, int& IDcounter, Contestants contestant[], bool random){

00094

00095 if (leftSpaces <= 0){

00096 cout << "No spaces left.\n Remove contestants to add more!\n";

00097 return;

00098 }

00099

00100 int contestantsToAdd = GetValidContestantInputCount(leftSpaces);

00101 leftSpaces -= contestantsToAdd;

00102

00103 string savedNames[MAXCONTESTANTS];

00104

00105

00106 deBugInfo("SYSTEM: size savedNames: " << size(savedNames) << endl);

00107

00108 for (int i = 0; contestantsToAdd > i; i++) {

00109

00110 int validIndex = getNextFreeSlot(contestant);

00111

00112 int newID = IDcounter++;

00113 if (random){

00114 randomContestantGenerator(contestant[validIndex], newID);

00115 } else {

00116 readSingleContestant(contestant[validIndex], newID);

00117 }

00118

00119 savedNames[i] = contestant[validIndex].name;

00120 }

00121

00122 if(savedNames[0] == ""){

00123 clearConsole(CLEARCONSOLE);

00124 cout << "No contestants added." << endl;

00125 deBugInfo("SYSTEM: contestantsToAdd: " << contestantsToAdd);

00126 } else {

00127 printNamesOfAddedContestants(savedNames);

00128 }

00129 }

00130

00131

00132 void printNamesOfAddedContestants(string savedNames[]){

00133

00134 clearConsole(CLEARCONSOLE);

00135

00136 deBugStringArray(savedNames, MAXCONTESTANTS);

00137 cout << "Contestant/s ";

00138

00139 bool foundEmptyIndex = false;

00140

00141 for(int i = 0; !foundEmptyIndex && MAXCONTESTANTS - 1 >= i; i++){

00142 string name = savedNames[i];

00143 foundEmptyIndex = (name == "");

00144

00145 if(i != 0 && !foundEmptyIndex) cout <<", ";

00146 cout << name;

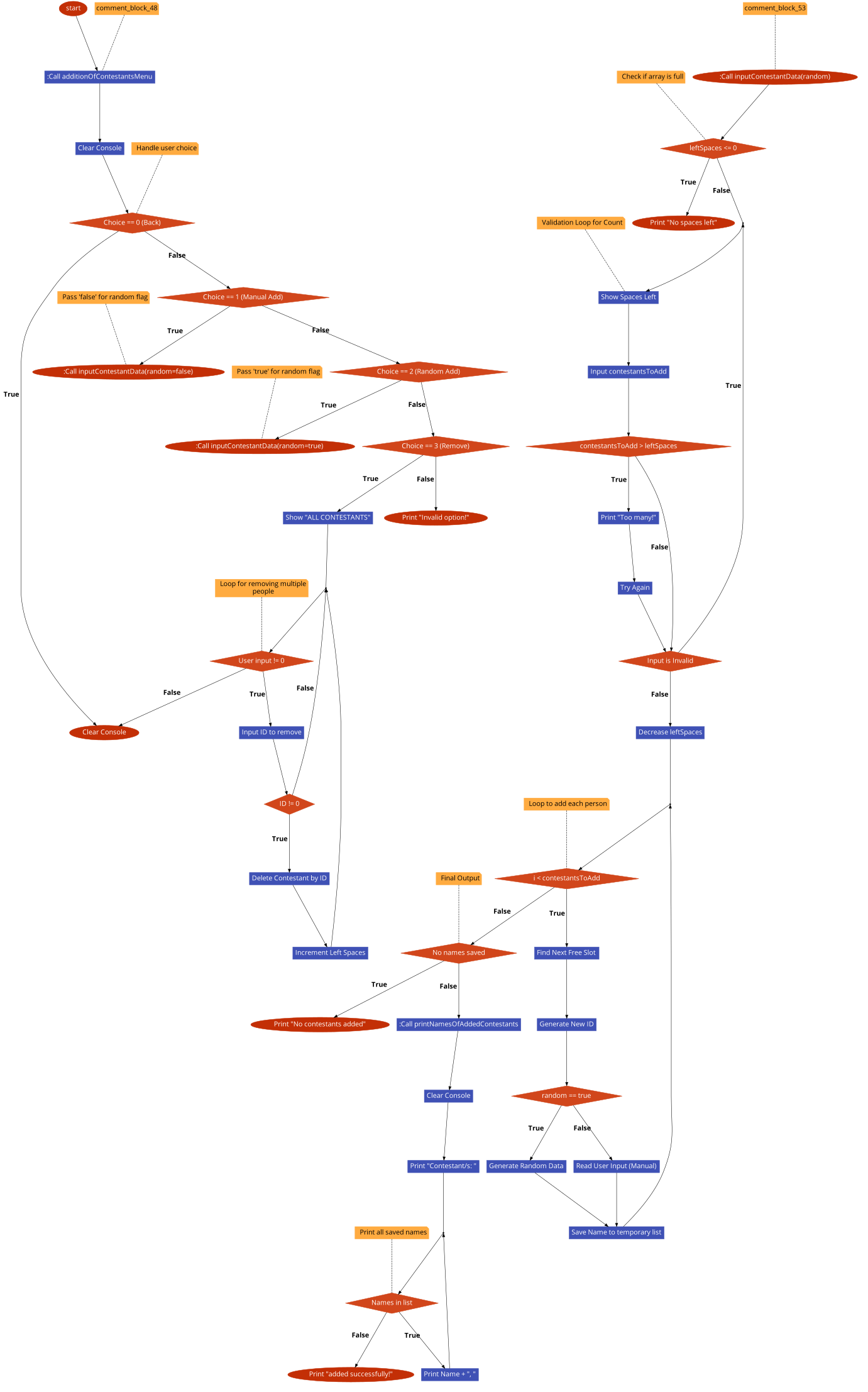
00147

00148 }

00149

00150 cout << " maybe added successfully!" << endl;

00151 }



## addition\_of\_contestants\_menu.h

00001 #pragma once

00002 #include <iostream>

00003 #include <string>

00004

00005 using namespace std;

00006

00007 #include "../contestants\_manipulation/contestants\_generators.h"

00008 #include "../contestants\_manipulation/contestants\_searchers.h"

00009

00010 #include "../../utils/visual\_enhancement.h"

00011 #include "../../utils/settings.h"

00012 #include "../../utils/contestant.h"

00013 #include "../../utils/cin\_validators.h"

00014 #include "../../utils/utils.h"

00015

00016

00022 void additionOfContestantsMenu(Contestants contestant[], int& leftSpaces, int& IDcounter, int& menuChoice);

00023

## BeautyContest/logic/categorisation/categorising.cpp Файл Справка

Implementation of age-based categorization logic.

#include "categorising.h"

### Функции

* void categorizeContestantsByAge (Contestants contestant[], Contestants category14\_16[], Contestants category17\_19[], Contestants category20\_22[], Contestants category23\_25[])

*Categorizes every contestant.*

### Подробно описание

Implementation of age-based categorization logic.

Виж дефиницията във файла categorising.cpp.

### Функции Документация

#### void categorizeContestantsByAge (Contestants contestant[], Contestants category14\_16[], Contestants category17\_19[], Contestants category20\_22[], Contestants category23\_25[])

Categorizes every contestant.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | From where to get the contestants |
| *category14\_16* | First age group |
| *category17\_19* | Secund age group |
| *category20\_22* | Third age group |
| *category23\_25* | Forth age group |

Виж дефиницията във файла categorising.cpp ред 6.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



## categorising.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "categorising.h"

00005

00006 void categorizeContestantsByAge(Contestants contestant[], Contestants category14\_16[], Contestants category17\_19[], Contestants category20\_22[], Contestants category23\_25[]){

00007 for(int i = getNextUsedSlot(contestant, 0); i != -1; i = getNextUsedSlot(contestant, i + 1)){

00008 deBugInfo("SYSTEM: i: " << i << endl);

00009

00010 Contestants cont = contestant[i];

00011 int contAge = cont.age;

00012

00013 int validIndexToPut;

00014 //LATER: make this with Contestants temp; use dupe function to make logic more readble

00015 if (contAge >= 14 && contAge <= 16){

00016 validIndexToPut = getNextFreeSlot(category14\_16);

00017 category14\_16[validIndexToPut] = cont;

00018 deBugInfo("SYSTEM: Put cont " << cont.name << " at category14\_16" << endl);

00019

00020 } else if(contAge >= 17 && contAge <= 19){

00021 validIndexToPut = getNextFreeSlot(category17\_19);

00022 category17\_19[validIndexToPut] = cont;

00023 deBugInfo("SYSTEM: Put cont " << cont.name << " at category17\_19" << endl);

00024

00025 } else if(contAge >= 20 && contAge <= 22){

00026 validIndexToPut = getNextFreeSlot(category20\_22);

00027 category20\_22[validIndexToPut] = cont;

00028 deBugInfo("SYSTEM: Put cont " << cont.name << " at category20\_22" << endl);

00029

00030 } else if(contAge >= 23 && contAge <= 25){

00031 validIndexToPut = getNextFreeSlot(category23\_25);

00032 category23\_25[validIndexToPut] = cont;

00033 deBugInfo("SYSTEM: Put cont " << cont.name << " at category23\_25" << endl);

00034

00035 } else {

00036 deBugInfo("WARNING - categorizeContestantsByAge: Contestant is too old. Skipping. Age: " << contAge << endl);

00037 }

00038 }

00039

00040 clearConsole();

00041 printAllContestantsFormatted(category14\_16, "Category 14/16 years");

00042

00043 printAllContestantsFormatted(category17\_19, "Category 17/19 years");

00044

00045 printAllContestantsFormatted(category20\_22, "Category 20/22 years");

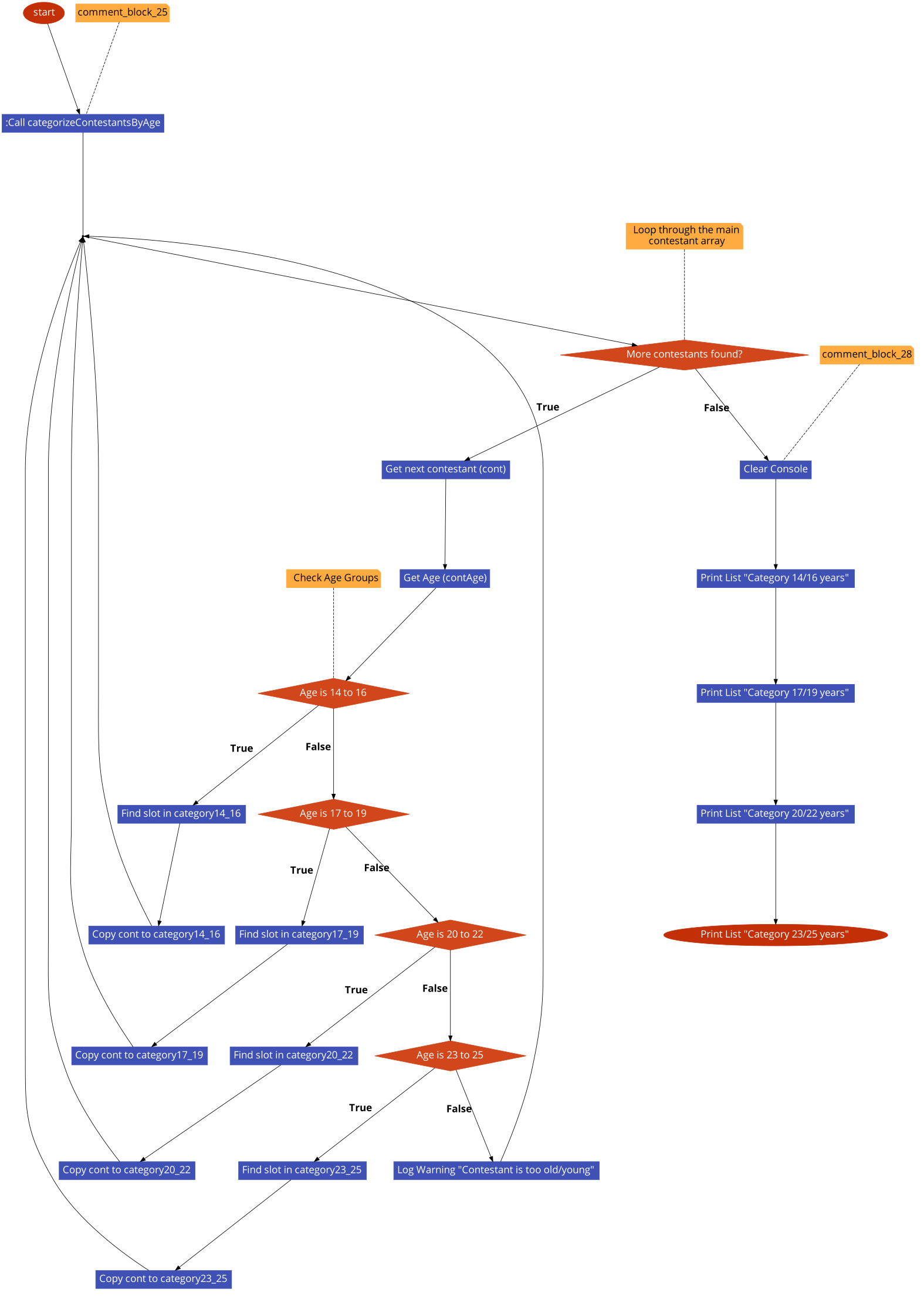
00046

00047 printAllContestantsFormatted(category23\_25, "Category 23/25 years");

00048

00049 }

00050



## categorising.h

00001 #pragma once

00002 #include <iostream>

00003 // #include <>

00004

00005 using namespace std;

00006

00007 #include "../../utils/contestant.h"

00008 #include "../../utils/debug.h"

00009 #include "../../utils/settings.h"

00010

00011 #include "../../block\_mangment.h"

00012

00013 #include "../contestants\_manipulation/contestants\_manager.h"

00014 #include "../contestants\_manipulation/contestants\_searchers.h"

00015

00022 void categorizeContestantsByAge(Contestants contestant[], Contestants category14\_16[], Contestants category17\_19[], Contestants category20\_22[], Contestants category23\_25[]);

## BeautyContest/logic/contestants\_manipulation/contestans\_sorters.cpp Файл Справка

Implementation of sorting algorithms (Bubble Sort) for contestants.

#include "contestans\_sorters.h"

### Функции

* void sortContestants (Contestants contestant[], SortMode mode)

*Manages the logic for the sorting menu.*

### Подробно описание

Implementation of sorting algorithms (Bubble Sort) for contestants.

Виж дефиницията във файла contestans\_sorters.cpp.

### Функции Документация

#### void sortContestants (Contestants contestant[], SortMode mode)

Manages the logic for the sorting menu.

##### Аргументи

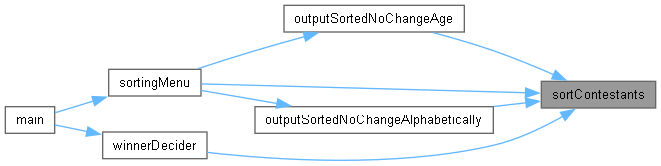
|  |  |
| --- | --- |
| *contestant* | The contestants to sort |
| *mode* | SORT\_BY\_ID, SORT\_BY\_NAME, SORT\_BY\_AGE\_ASC, SORT\_BY\_POINTS\_ASC, SORT\_BY\_POINTS\_DESC |

Виж дефиницията във файла contestans\_sorters.cpp ред 6.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



## contestans\_sorters.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "contestans\_sorters.h"

00005

00006 void sortContestants(Contestants contestant[], SortMode mode) {

00007 bool swapped = true;

00008

00009 while (swapped) {

00010 swapped = false;

00011

00012 int validIndex = getNextUsedSlot(contestant);

00013

00014 while (validIndex != -1) {

00015

00016 int nextValidIndex = getNextUsedSlot(contestant, validIndex + 1);

00017

00018 if (nextValidIndex == -1) {

00019 break;

00020 }

00021

00022 bool shouldSwap = false;

00023

00024 switch (mode) {

00025 case SORT\_BY\_ID:

00026 shouldSwap = contestant[validIndex].ID > contestant[nextValidIndex].ID;

00027 break;

00028 case SORT\_BY\_NAME:

00029 shouldSwap = strcmp(contestant[validIndex].name, contestant[nextValidIndex].name) > 0;

00030 break;

00031 case SORT\_BY\_AGE\_ASC:

00032 shouldSwap = contestant[validIndex].age > contestant[nextValidIndex].age;

00033 break;

00034 case SORT\_BY\_POINTS\_DESC:

00035 shouldSwap = contestant[validIndex].points < contestant[nextValidIndex].points;

00036 break;

00037 case SORT\_BY\_POINTS\_ASC:

00038 shouldSwap = contestant[validIndex].points > contestant[nextValidIndex].points;

00039 break;

00040 }

00041

00042 if (shouldSwap) {

00043 swapContestantPlaces(contestant, validIndex, nextValidIndex);

00044 swapped = true;

00045 }

00046

00047 validIndex = nextValidIndex;

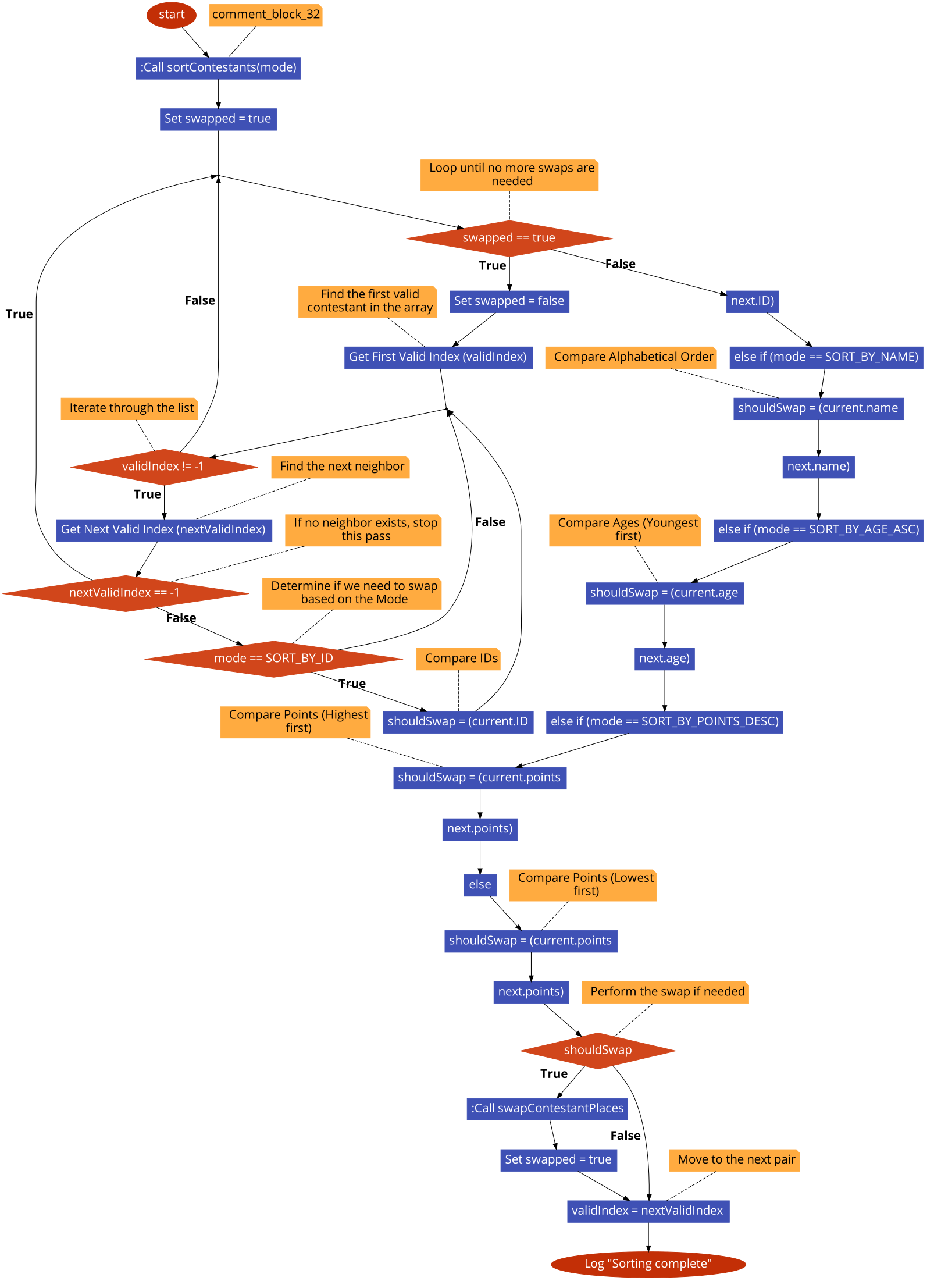
00048 }

00049 }

00050

00051 deBugInfo("SYSTEM: Sorting complete." << endl);

00052 }



## contestans\_sorters.h

00001 #pragma once

00002 #include <cstring>

00003 #include <string>

00004

00005 using namespace std;

00006

00007 #include "contestants\_searchers.h"

00008 #include "contestants\_manager.h"

00009

00010 #include "../../utils/contestant.h"

00011 #include "../../utils/settings.h"

00012 #include "../../utils/debug.h"

00013

00014 enum SortMode {

00015 SORT\_BY\_ID = 0,

00016 SORT\_BY\_NAME = 1,

00017 SORT\_BY\_AGE\_ASC = 2,

00018 SORT\_BY\_POINTS\_ASC = 4,

00019 SORT\_BY\_POINTS\_DESC = 3

00020 };

00021

00025 void sortContestants(Contestants contestant[], SortMode mode);

00026

## BeautyContest/logic/contestants\_manipulation/contestants\_generators.cpp Файл Справка

Functions for generating random contestants and reading manual input.

#include "contestants\_generators.h"

### Функции

* void randomContestantGenerator (Contestants &contestant, const int &IDcounter)

*Generates a random contestant values.*

* void readSingleContestant (Contestants &contestant, const int &IDcounter)

*Puts infor from user to contestant.*

### Подробно описание

Functions for generating random contestants and reading manual input.

Виж дефиницията във файла contestants\_generators.cpp.

### Функции Документация

#### void randomContestantGenerator (Contestants & contestant, const int & ID)

Generates a random contestant values.

* + Has female and Male names that are picked randomly according to gender. Other parameters as hipCirc, shoulderCirc, calfCirc, neckCirc are picked between ranges according to gender.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | The array of all contestants. |
| *IDcounter* | The ID the contestant should get. |

Виж дефиницията във файла contestants\_generators.cpp ред 6.

Граф на извикванията за тази функция:



#### void readSingleContestant (Contestants & contestant, const int & IDcounter)

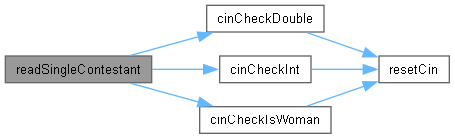
Puts infor from user to contestant.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | the contestant that needs manipulation |
| *IDcounter* | the last id issued |

Виж дефиницията във файла contestants\_generators.cpp ред 82.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



## contestants\_generators.cpp

Вижте документацията за този файл.

00001

00003

00004 #include"contestants\_generators.h"

00005

00006 void randomContestantGenerator (Contestants& contestant, const int& IDcounter)

00007 {

00008

00009

00010 int ageVariation = 11;

00011 int ageOffSet = 15;

00012

00013 string femaleNames[60] = {

00014 "Mary", "Patricia", "Linda", "Barbara", "Elizabeth", "Jennifer", "Maria", "Susan", "Margaret", "Dorothy",

00015 "Lisa", "Nancy", "Karen", "Betty", "Helen", "Sandra", "Donna", "Carol", "Ruth", "Sharon",

00016 "Michelle", "Laura", "Sarah", "Kimberly", "Deborah", "Jessica", "Shirley", "Cynthia", "Angela", "Melissa",

00017 "Brenda", "Amy", "Anna", "Rebecca", "Virginia", "Kathleen", "Pamela", "Martha", "Debra", "Amanda",

00018 "Stephanie", "Carolyn", "Christine", "Marie", "Janet", "Catherine", "Frances", "Ann", "Joyce", "Diane",

00019 "Alice", "Julie", "Heather", "Teresa", "Doris", "Gloria", "Evelyn", "Jean", "Cheryl", "Mildred"

00020 };

00021

00022 string maleNames[60] = {

00023 "James", "John", "Robert", "Michael", "William", "David", "Richard", "Joseph", "Thomas", "Charles",

00024 "Christopher", "Daniel", "Matthew", "Anthony", "Donald", "Mark", "Paul", "Steven", "Andrew", "Kenneth",

00025 "Joshua", "Kevin", "Brian", "George", "Edward", "Ronald", "Timothy", "Jason", "Jeffrey", "Ryan",

00026 "Jacob", "Gary", "Nicholas", "Eric", "Jonathan", "Stephen", "Larry", "Justin", "Scott", "Brandon",

00027 "Benjamin", "Samuel", "Frank", "Gregory", "Raymond", "Alexander", "Patrick", "Jack", "Dennis", "Jerry",

00028 "Tyler", "Aaron", "Jose", "Adam", "Henry", "Nathan", "Douglas", "Zachary", "Peter", "Kyle"

00029 };

00030

00031 bool isObjectUsed = rand() % 2;

00032 char name[MAXNAMECHARS];

00033 int age = rand() % ageVariation + ageOffSet;

00034 bool isWoman = rand() % 2;

00035 double hipCirc;

00036 double shoulderCirc;

00037 double calfCirc;

00038 double neckCirc;

00039

00040 const int OFFSET = 12; // so they can enter in all the winner brackets

00041

00042

00043 // Measurement,Women (Range),Men (Range)

00044 // Hip Circ (H),90 - 105 cm,90 - 105 cm

00045 // Shoulder (S)95 - 110 cm,110 - 125 cm

00046 // Calf Circ (C),32 - 40 cm,35 - 42 cm

00047 // Neck Circ (N),30 - 35 cm,37 - 42 cm

00048 // thank you gemini

00049

00050 string tempName;

00051 if (isWoman){

00052 tempName = femaleNames[rand() % 60];

00053 hipCirc = (rand() % 1500) / 100.0 + 90.0 + OFFSET;

00054 shoulderCirc = (rand() % 1500) / 100.0 + 95.0;

00055 calfCirc = (rand() % 800) / 100.0 + 32.0 ;

00056 neckCirc = (rand() % 500) / 100.0 + 30.0;

00057 } else {

00058 tempName = maleNames[rand() % 60];

00059 hipCirc = (rand() % 1500) / 100.0 + 90.0 + OFFSET;

00060 shoulderCirc = (rand() % 1500) / 100.0 + 110.0;

00061 calfCirc = (rand() % 700) / 100.0 + 35.0 ;

00062 neckCirc = (rand() % 500) / 100.0 + 37.0;

00063 }

00064

00065

00066

00067 contestant.ID = IDcounter;

00068 contestant.isObjectUsed = true;

00069

00070 strncpy(contestant.name, tempName.c\_str(), MAXNAMECHARS - 1);

00071 contestant.name[tempName.length()] = '\0';

00072

00073 contestant.age = age;

00074 contestant.isWoman = isWoman;

00075 contestant.hipCirc = hipCirc;

00076 contestant.shoulderCirc = shoulderCirc;

00077 contestant.calfCirc = calfCirc;

00078 contestant.neckCirc = neckCirc;

00079 contestant.points = hipCirc / (shoulderCirc + neckCirc + calfCirc);

00080 }

00081

00082 void readSingleContestant(Contestants& contestant, const int& IDcounter){

00083

00084 contestant.ID = IDcounter;

00085 contestant.isObjectUsed = true;

00086

00087 cout << "Enter name: ";

00088 cin.getline(contestant.name, MAXNAMECHARS);

00089

00090 contestant.age = cinCheckInt("Enter age: ");

00091

00092 contestant.isWoman = cinCheckIsWoman("Enter gender (m/f): ");

00093

00094 contestant.hipCirc = cinCheckDouble("Enter hip circumference (cm): ");

00095

00096 contestant.shoulderCirc = cinCheckDouble("Enter shoulder circumference (cm): ");

00097

00098 contestant.neckCirc = cinCheckDouble("Enter neck circumference (cm): ");

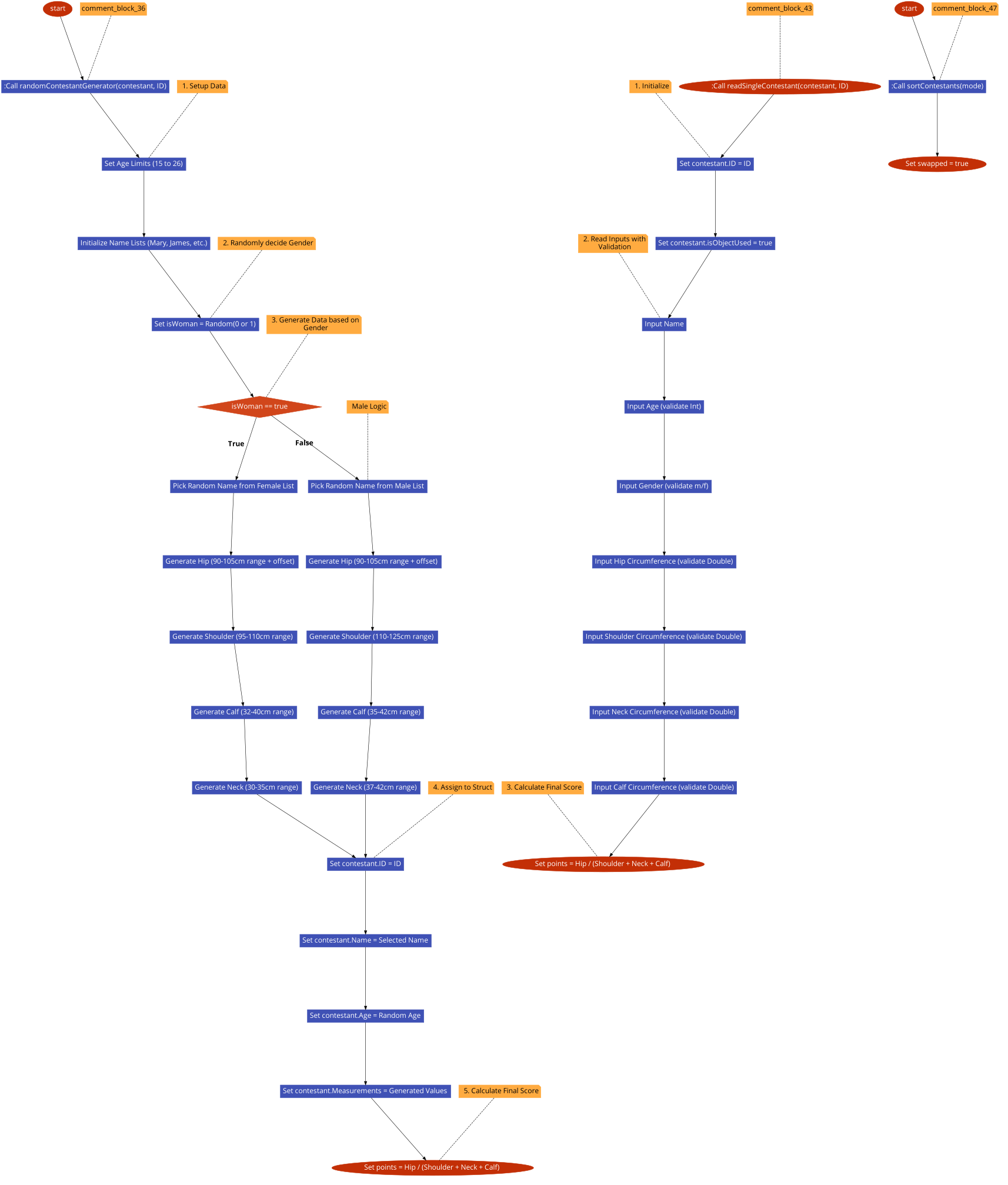
00099

00100 contestant.calfCirc = cinCheckDouble("Enter calf circumference (cm): ");

00101

00102 contestant.points = contestant.hipCirc / (contestant.shoulderCirc + contestant.neckCirc + contestant.calfCirc);

00103 }



## contestants\_generators.h

00001 #pragma once

00002 #include <iostream>

00003 #include <string>

00004 #include <cstring>

00005

00006 #include "../../utils/contestant.h"

00007 #include "../../utils/debug.h"

00008 #include "../../utils/settings.h"

00009 #include "../../utils/cin\_validators.h"

00010

00011 using namespace std;

00012

00013

00020 void randomContestantGenerator (Contestants& contestant, const int& ID);

00021

00022

00026 void readSingleContestant(Contestants& contestant, const int& IDcounter);

## BeautyContest/logic/contestants\_manipulation/contestants\_manager.cpp Файл Справка

Helper functions for swapping, duplicating, and resetting contestant data.

#include "contestants\_manager.h"

### Функции

* void swapContestantPlaces (Contestants contestant[], int FIndex, int SIndex)

*Swaps contestant places with given indexes.*

* void dupeContestantInfo (Contestants from[], Contestants to[])

*Duplicates info from one contestants struct to other.*

* void resetContestants (Contestants contestant[])

*Resets everything in contestants struct.*

### Подробно описание

Helper functions for swapping, duplicating, and resetting contestant data.

Виж дефиницията във файла contestants\_manager.cpp.

### Функции Документация

#### void dupeContestantInfo (Contestants from[], Contestants to[])

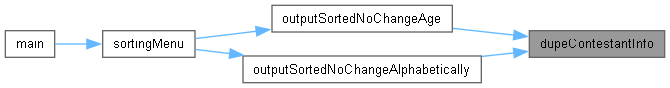
Duplicates info from one contestants struct to other.

##### Аргументи

|  |  |
| --- | --- |
| *from* | where you get info to dupe |
| *to* | location to dupe on |

Виж дефиницията във файла contestants\_manager.cpp ред 14.

Граф на извикванията за тази функция:



#### void resetContestants (Contestants contestant[])

Resets everything in contestants struct.

##### Аргументи

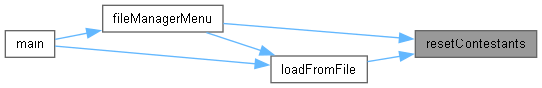
|  |  |
| --- | --- |
| *contestant* | the struct you want to reset |

Виж дефиницията във файла contestants\_manager.cpp ред 20.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void swapContestantPlaces (Contestants contestant[], int FIndex, int SIndex)

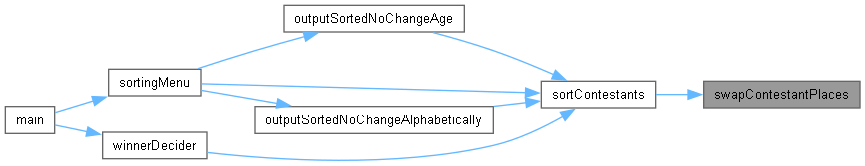
Swaps contestant places with given indexes.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | Where you want the swaping to be done |
| *Findex* | First contestant to swap |
| *Sindex* | Second contestant to swap |

Виж дефиницията във файла contestants\_manager.cpp ред 7.

Граф на извикванията за тази функция:



## contestants\_manager.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "contestants\_manager.h"

00005

00006

00007 void swapContestantPlaces(Contestants contestant[], int FIndex, int SIndex){

00008 deBugInfo("SYSTEM: Swapping indexes " << FIndex << " and " << SIndex << endl);

00009 Contestants placeHolder = contestant[FIndex];

00010 contestant[FIndex] = contestant[SIndex];

00011 contestant[SIndex] = placeHolder;

00012 }

00013

00014 void dupeContestantInfo (Contestants from[], Contestants to[]){

00015 for(int i = 0; i < MAXCONTESTANTS; i++){

00016 to[i] = from[i];

00017 }

00018 }

00019

00020 void resetContestants (Contestants contestant[]){

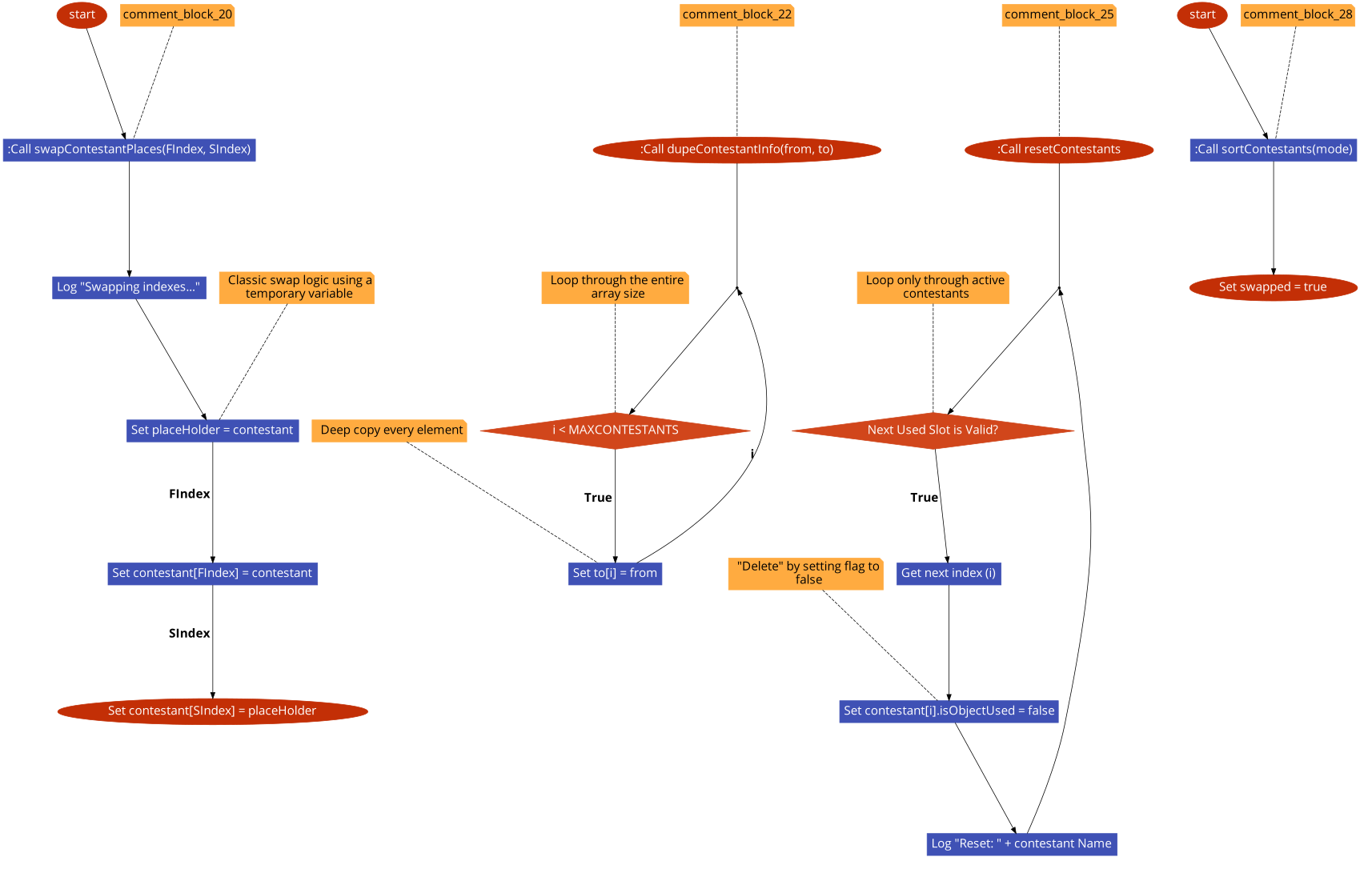
00021 for(int i = getNextUsedSlot(contestant, 0); i != -1; i = getNextUsedSlot(contestant, i + 1)){

00022 contestant[i].isObjectUsed = false;

00023 deBugInfo("SYSTEM: Rested: " << contestant[i].name << endl);

00024 }

00025 }



## contestants\_manager.h

00001 #pragma once

00002

00003 #include "contestants\_searchers.h"

00004

00005 #include "../../utils/contestant.h"

00006 #include "../../utils/debug.h"

00007 using namespace std;

00008

00009

00014 void swapContestantPlaces(Contestants contestant[], int FIndex, int SIndex);

00015

00019 void dupeContestantInfo (Contestants from[], Contestants to[]);

00020

00023 void resetContestants (Contestants contestant[]);

## BeautyContest/logic/contestants\_manipulation/contestants\_printers.cpp Файл Справка

Functions responsible for printing contestant details to the console.

#include "contestants\_printers.h"

### Функции

* void printContestant (Contestants contestant[], int index)

*Prints out contestant in one single line(has debug that shows the index).*

* void printAllContestants (Contestants contestant[])

*Prints out all contestants with isUsed = 1.*

* void printAllContestantsFormatted (Contestants contestant[], string message)

*Prints all contestnats but little bit more formatted.*

### Подробно описание

Functions responsible for printing contestant details to the console.

Виж дефиницията във файла contestants\_printers.cpp.

### Функции Документация

#### void printAllContestants (Contestants contestant[])

Prints out all contestants with isUsed = 1.

##### Аргументи

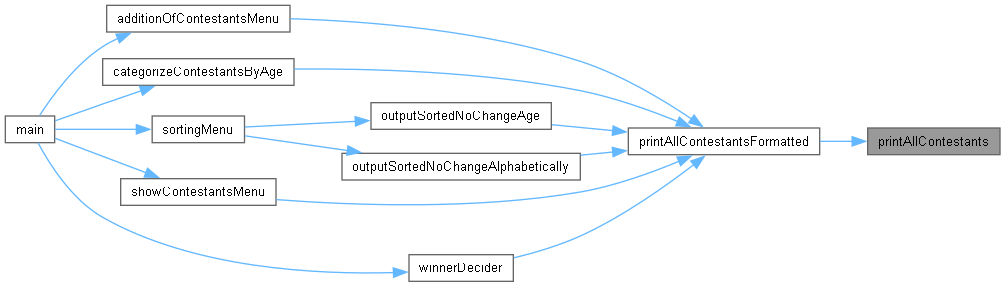
|  |  |
| --- | --- |
| *contestant* | Array of contestnats |

Виж дефиницията във файла contestants\_printers.cpp ред 21.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void printAllContestantsFormatted (Contestants contestant[], string title)

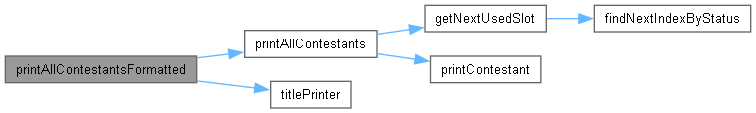
Prints all contestnats but little bit more formatted.

##### Аргументи

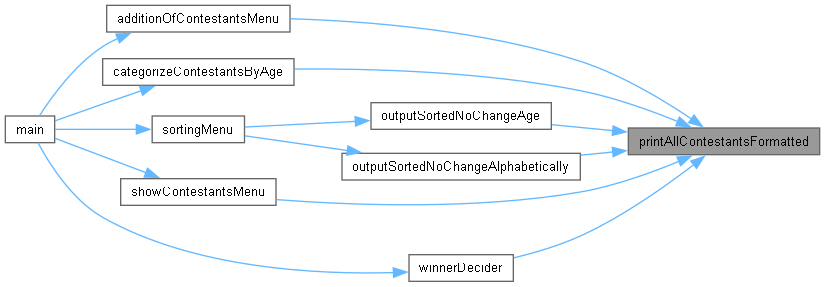
|  |  |
| --- | --- |
| *contestant* | Contestant array |
| *title* | Title of the table |

Виж дефиницията във файла contestants\_printers.cpp ред 42.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void printContestant (Contestants contestant[], int index)

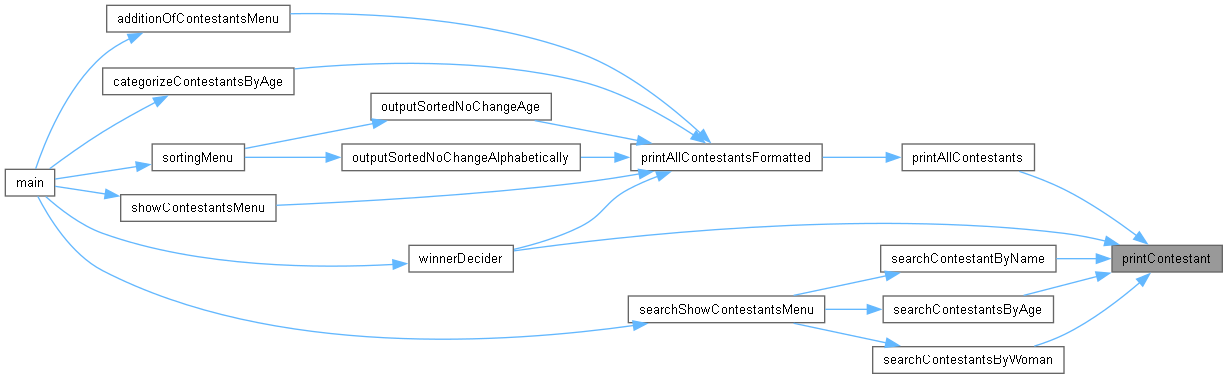
Prints out contestant in one single line(has debug that shows the index).

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | The array of all contestants. |
| *index* | The index of the contestant.(if deBug == 1 prints out the index too) |

Виж дефиницията във файла contestants\_printers.cpp ред 6.

Граф на извикванията за тази функция:



## contestants\_printers.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "contestants\_printers.h"

00005

00006 void printContestant(Contestants contestant[], int index){

00007

00008 deBugInfo("Index: " << index << " ");

00009 cout << "ID: " << contestant[index].ID << " ";

00010 cout << "Name: " << contestant[index].name << " ";

00011 cout << "Age: " << contestant[index].age << " ";

00012 cout << "Gender: " << ((contestant[index].isWoman) ? "Woman" : "Man") << " ";

00013 cout << "HipCirc: " << contestant[index].hipCirc << " ";

00014 cout << "ShoulderCirc: " << contestant[index].shoulderCirc << " ";

00015 cout << "CalfCirc: " << contestant[index].calfCirc << " ";

00016 cout << "NeckCirc: " << contestant[index].neckCirc << " ";

00017 cout << "Points: " << contestant[index].points << " ";

00018 cout << endl;

00019 }

00020

00021 void printAllContestants(Contestants contestant[]){

00022 bool hasContestants = false;

00023

00024 for(int i = 0; i < MAXCONTESTANTS; i++){

00025

00026 int validIndex = getNextUsedSlot(contestant, i);

00027 i = validIndex;

00028

00029

00030 if (validIndex == -1) {

00031 cout << endl;

00032

00033 if(!hasContestants) cout << "### No contestants ###\n\n";

00034

00035 break;

00036 }

00037 hasContestants = true;

00038 printContestant(contestant, validIndex);

00039 }

00040 }

00041

00042 void printAllContestantsFormatted(Contestants contestant[], string message){

00043 titlePrinter(message, TITLEWIDTH, '-');

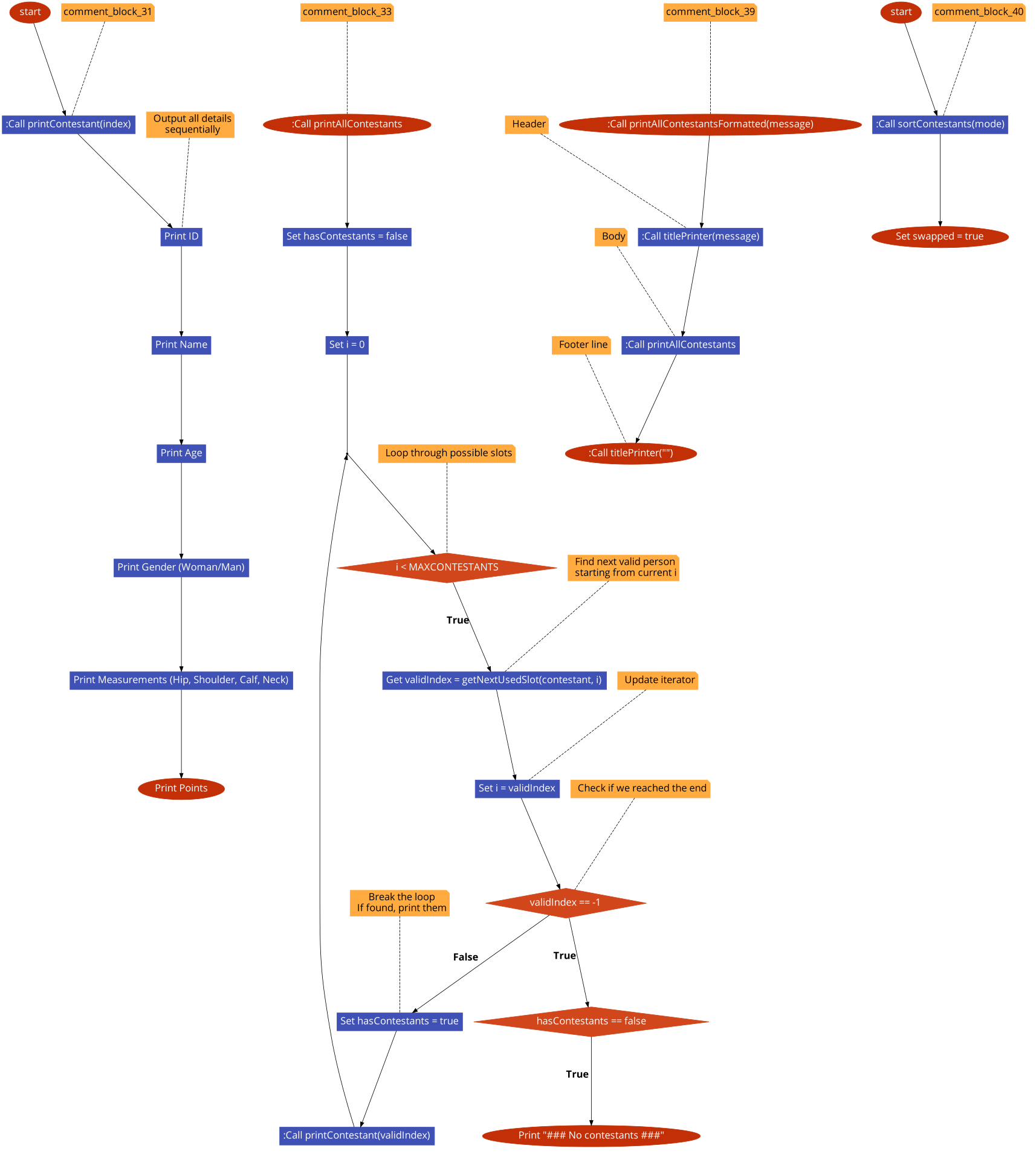
00044 printAllContestants(contestant);

00045 titlePrinter("", TITLEWIDTH, '-');

00046 cout << endl;

00047

00048 }



## contestants\_printers.h

00001 #pragma once

00002

00003

00004 #include "contestants\_searchers.h"

00005

00006 #include "../../block\_mangment.h"

00007

00008 #include "../../utils/contestant.h"

00009 #include "../../utils/debug.h"

00010

00011 using namespace std;

00012

00013

00017 void printContestant(Contestants contestant[], int index);

00018

00021 void printAllContestants(Contestants contestant[]);

00022

00026 void printAllContestantsFormatted(Contestants contestant[], string title);

## BeautyContest/logic/contestants\_manipulation/contestants\_searchers.cpp Файл Справка

Implementation of searching algorithms (by ID, Name, Age, etc.).

#include "contestants\_searchers.h"

### Функции

* int findNextIndexByStatus (Contestants contestant[], int searchingFor=0, int afterIndex=0)

*Outputs the first contestant it sees depending on if you want the contestant spot to be free or used.*

* int getNextFreeSlot (Contestants contestant[], int startIdx)

*Wrapper Function to find next free slot in contestants starting from startIdx.*

* int getNextUsedSlot (Contestants contestant[], int startIdx)

*Wrapper Function to find next used slot in contestants starting from startIdx.*

* bool searchContestantsByAge (Contestants contestant[], const int &age)

*Prints out contestants with given age and if it found someone.*

* bool searchContestantsByWoman (Contestants contestant[], const bool &isWoman)

*Prints out contestants with given gender.*

* int biggestContestantID (Contestants contestant[])

*Gives out the contestant with largest ID.*

* int countOfContestants (Contestants contestant[])

*Gives how many contestants are with isObjectUsed = true.*

* bool deleteContestantsByID (Contestants contestant[], const int &ID)

*Prints out.*

* bool searchContestantByName (Contestants contestant[], char nameToSearch[], int sizeToSearch, string didYouMeanNames[])

*Searches contestnats by name and gives possible names.*

* int searchLowestAge (Contestants contestant[])

*Finds lowest age in Contestants struct.*

### Подробно описание

Implementation of searching algorithms (by ID, Name, Age, etc.).

Виж дефиницията във файла contestants\_searchers.cpp.

### Функции Документация

#### int biggestContestantID (Contestants contestant[])

Gives out the contestant with largest ID.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | The array of all contestants. |

##### Връща

biggestID

Виж дефиницията във файла contestants\_searchers.cpp ред 75.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### int countOfContestants (Contestants contestant[])

Gives how many contestants are with isObjectUsed = true.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | Where you want to search |

##### Връща

Count of active contestants

Виж дефиницията във файла contestants\_searchers.cpp ред 90.

Граф на извикванията за тази функция:



#### bool deleteContestantsByID (Contestants contestant[], const int & ID)

Prints out.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | The array of all contestants. |
| *ID* | ID to delete |

##### Връща

If it was successful

Виж дефиницията във файла contestants\_searchers.cpp ред 105.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### int findNextIndexByStatus (Contestants contestant[], int searchingFor = 0, int afterIndex = 0)

Outputs the first contestant it sees depending on if you want the contestant spot to be free or used.

* + It searches after a certain index so after putting it in a cycle it doesnt output the same contestant when no match found after the index outputs -1 which has to be detected with if statement.

##### Аргументи

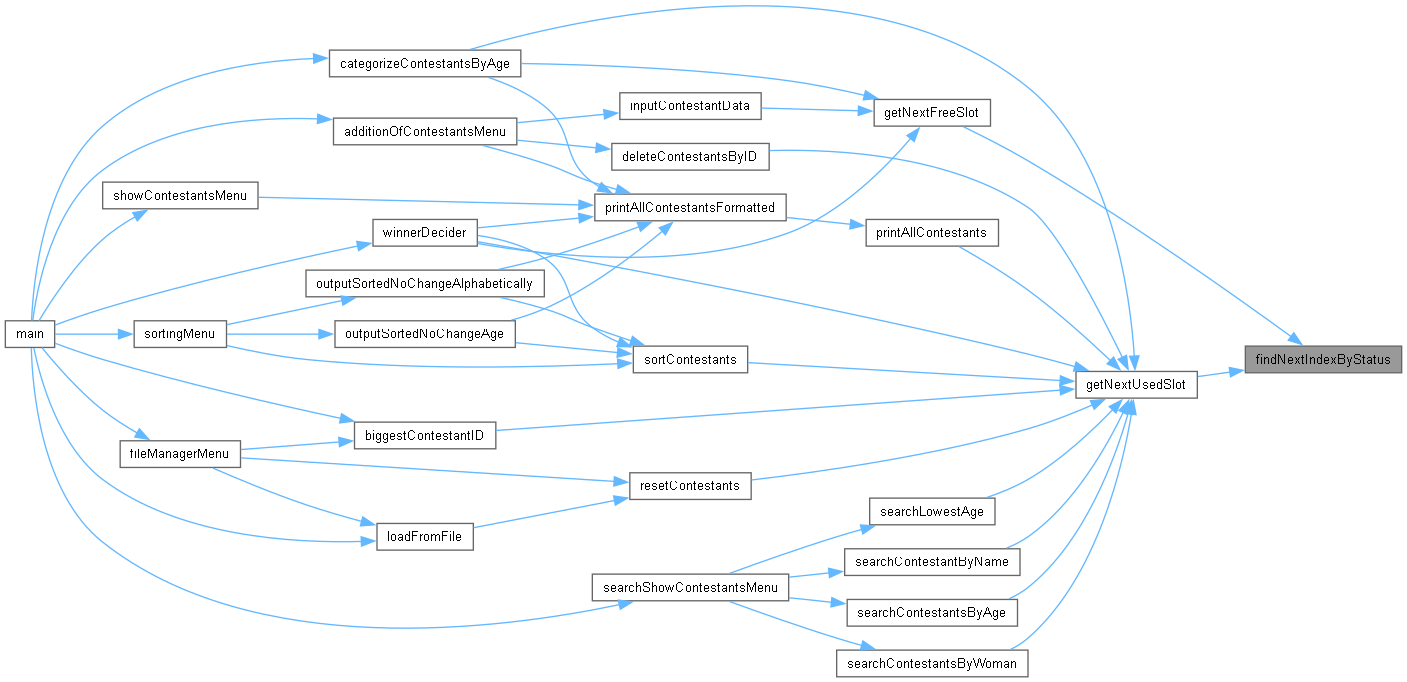
|  |  |
| --- | --- |
| *contestant* | The array of all contestants. |
| *searchingFor* | The value you want isObjectUsed to match. |
| *afterIndex* | The index to start searching from (to avoid duplicates). |

##### Връща

int The index of the next matching slot, or -1 if full.

Виж дефиницията във файла contestants\_searchers.cpp ред 15.

Граф на извикванията за тази функция:



#### int getNextFreeSlot (Contestants contestant[], int startIdx = 0)

Wrapper Function to find next free slot in contestants starting from startIdx.

##### Аргументи

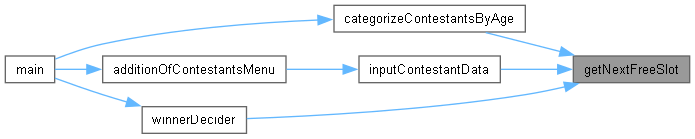
|  |  |
| --- | --- |
| *contestant* | The array of all contestants. |
| *startIdx* | The index to start searching from. |

Виж дефиницията във файла contestants\_searchers.cpp ред 32.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### int getNextUsedSlot (Contestants contestant[], int startIdx = 0)

Wrapper Function to find next used slot in contestants starting from startIdx.

##### Аргументи

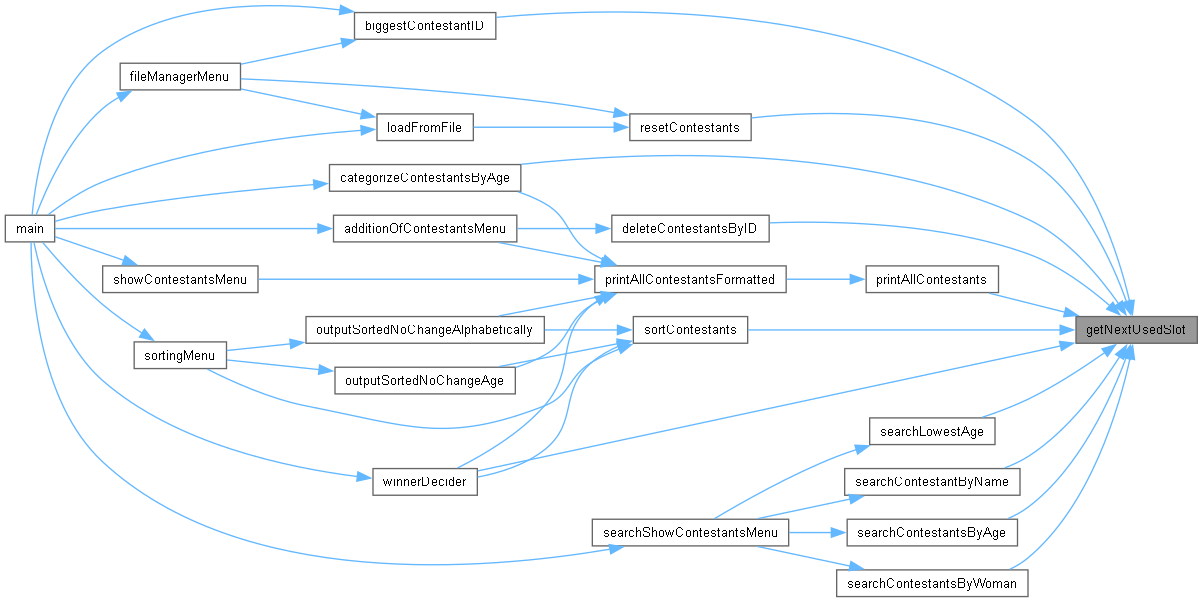
|  |  |
| --- | --- |
| *contestant* | The array of all contestants. |
| *startIdx* | The index to start searching from. |

Виж дефиницията във файла contestants\_searchers.cpp ред 37.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### bool searchContestantByName (Contestants contestant[], char nameToSearch[], int sizeToSearch, string didYouMeanNames[])

Searches contestnats by name and gives possible names.

##### Аргументи

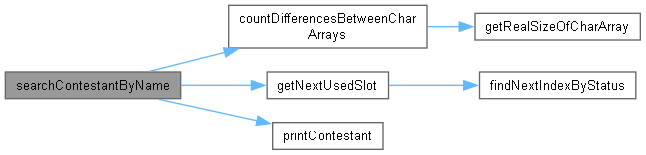
|  |  |
| --- | --- |
| *contestant* | The array of all contestants. |
| *nameToSearch* | what name to search |
| *sizeToSearch* | size of name |
| *didYouMeanNames* | possible names |

##### Връща

if person was found

Виж дефиницията във файла contestants\_searchers.cpp ред 124.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### bool searchContestantsByAge (Contestants contestant[], const int & age)

Prints out contestants with given age and if it found someone.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | The array of all contestants. |
| *age* | age to search for |

##### Връща

if it found a match

Виж дефиницията във файла contestants\_searchers.cpp ред 42.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### bool searchContestantsByWoman (Contestants contestant[], const bool & isWoman)

Prints out contestants with given gender.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | The array of all contestants. |
| *isWoman* | this is the given gender |

##### Връща

if it found a match

Виж дефиницията във файла contestants\_searchers.cpp ред 63.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### int searchLowestAge (Contestants contestant[])

Finds lowest age in Contestants struct.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | Array of all contestants |

##### Връща

If it found a match

Виж дефиницията във файла contestants\_searchers.cpp ред 154.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



## contestants\_searchers.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "contestants\_searchers.h"

00005

00015 int findNextIndexByStatus(Contestants contestant[], int searchingFor = 0, int afterIndex = 0) {

00016

00017 deBugInfo("SYSTEM - FNIBS: was lunched!" << endl);

00018 // deBugInfo("SYSTEM:FNIBS Starting search at index: " << afterIndex << endl);

00019 for (int n = afterIndex; n < MAXCONTESTANTS; n++) {

00020 int cont = contestant[n].isObjectUsed;

00021 if (cont == searchingFor) {

00022 // deBugInfo("SYSTEM:FNIBS Found match at index: "<< n << endl);

00023 return n;

00024 }

00025 }

00026 // deBugInfo("SYSTEM:FNIBS No spaces with " << searchingFor << endl);

00027 return -1;

00028 }

00029

00030

00031

00032 int getNextFreeSlot(Contestants contestant[], int startIdx) {

00033 return findNextIndexByStatus(contestant, 0, startIdx);

00034 }

00035

00036

00037 int getNextUsedSlot(Contestants contestant[], int startIdx) {

00038 return findNextIndexByStatus(contestant, 1, startIdx);

00039 }

00040

00041

00042 bool searchContestantsByAge(Contestants contestant[],const int& age){

00043

00044 bool foundPerson = false;

00045 int validIndex = getNextUsedSlot(contestant);

00046 for(int i = 0; i < MAXCONTESTANTS && validIndex != -1; i++){

00047

00048 deBugInfo("SYSTEM: i = "<< i << endl);

00049 deBugInfo("SYSTEM: age found: " << age << endl);

00050

00051 validIndex = getNextUsedSlot(contestant, i);

00052

00053 i = validIndex;

00054 if (contestant[validIndex].age == age){

00055 printContestant(contestant, validIndex);

00056 foundPerson = true;

00057 }

00058 }

00059 return foundPerson;

00060 }

00061

00062

00063 bool searchContestantsByWoman(Contestants contestant[],const bool& isWoman){

00064

00065 bool foundPerson = false;

00066 for(int i = getNextUsedSlot(contestant); i < MAXCONTESTANTS && i != -1; i = getNextUsedSlot(contestant, i + 1)){

00067

00068 if (contestant[i].isWoman == isWoman){

00069 printContestant(contestant, i);

00070 foundPerson = true;

00071 }

00072 }

00073 return foundPerson;

00074 }

00075 int biggestContestantID (Contestants contestant[]){

00076

00077 int biggestID = 0;

00078

00079 for(int i = getNextUsedSlot(contestant); i < MAXCONTESTANTS && i != -1; i = getNextUsedSlot(contestant, i + 1)){

00080 deBugInfo("SYSTEM - biggestContestantID: I: " << i << endl);

00081 if (contestant[i].ID > biggestID){

00082 biggestID = contestant[i].ID;

00083 deBugInfo("SYSTEM: biggestID: " << biggestID << " | I: " << i << endl);

00084

00085 }

00086 }

00087 return biggestID;

00088 }

00089

00090 int countOfContestants(Contestants contestant[]){

00091 int contestantCount = 0;

00092

00093 for(int i = 0; i < MAXCONTESTANTS; i++){

00094

00095 if (!contestant[i].isObjectUsed){

00096 contestantCount = contestantCount + 1;

00097 deBugInfo("SYSTEM: contestantCount got incremented!" << endl);

00098

00099 }

00100 deBugInfo("SYSTEM: contestantCount: " << contestantCount << endl;)

00101 }

00102 return contestantCount;

00103 }

00104

00105 bool deleteContestantsByID(Contestants contestant[],const int& ID){

00106

00107 bool foundPerson = false;

00108 int validIndex = getNextUsedSlot(contestant);

00109 for(int i = validIndex; i < MAXCONTESTANTS && validIndex != -1; i++){

00110

00111 validIndex = getNextUsedSlot(contestant, i);

00112

00113 i = validIndex;

00114 if (contestant[validIndex].ID == ID){

00115 cout << "Contestant " << contestant[validIndex].name << " successfully delted!" << endl;

00116 contestant[validIndex].isObjectUsed = false;

00117 foundPerson = true;

00118 }

00119 }

00120 return foundPerson;

00121 }

00122

00123

00124 bool searchContestantByName(Contestants contestant[], char nameToSearch[], int sizeToSearch, string didYouMeanNames[]){

00125 bool foundPerson = false;

00126 int validIndex = getNextUsedSlot(contestant);

00127 int didYouMean\_ValidIndex = 0;

00128 for(int i = validIndex; i < MAXCONTESTANTS && validIndex != -1; i++ )

00129 {

00130

00131 char contName[MAXNAMECHARS];

00132 strcpy(contName, contestant[i].name);

00133

00134 int sizeFirst = sizeof(contName) / sizeof(contName[0]);

00135 int sizeSecond = sizeToSearch / sizeof(nameToSearch[0]);

00136 int differences = countDifferencesBetweenCharArrays(contName, nameToSearch, sizeFirst, sizeSecond);

00137

00138

00139 if(differences == 0){

00140 printContestant(contestant, i);

00141 foundPerson = true;

00142 } else if (differences == 1 && !foundPerson){

00143 didYouMeanNames[didYouMean\_ValidIndex] = contestant[i].name;

00144 didYouMean\_ValidIndex++;

00145 }

00146

00147 validIndex = getNextUsedSlot(contestant, i);

00148 i = validIndex;

00149 deBugInfo("Next validIndex: " << validIndex << endl);

00150 }

00151 return foundPerson;

00152 }

00153

00154 int searchLowestAge (Contestants contestant[]){

00155

00156 int validIndex = getNextUsedSlot(contestant);

00157 int lowestAge = contestant[validIndex].age;

00158 for(int i = 1; i < MAXCONTESTANTS; i++){

00159

00160 validIndex = getNextUsedSlot(contestant, i);

00161 i = validIndex;

00162

00163

00164 if (validIndex == -1) {

00165 cout << endl;

00166 break;

00167 }

00168

00169

00170 int contestantAge = contestant[validIndex].age;

00171 if(contestantAge <= lowestAge){

00172 lowestAge = contestantAge;

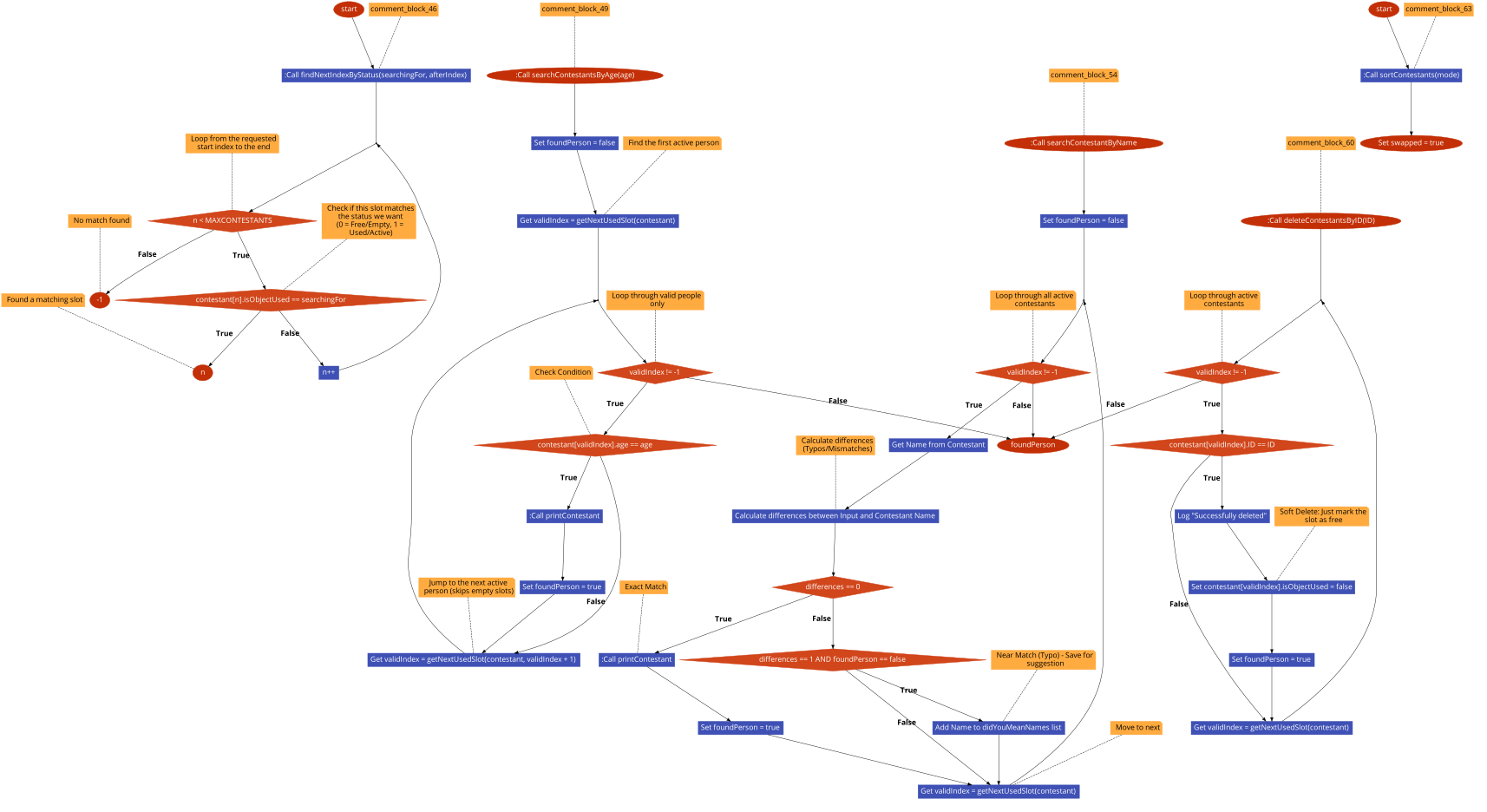
00173 }

00174

00175 }

00176 return lowestAge;

00177 }



## contestants\_searchers.h

00001 #pragma once

00002 #include <string>

00003 #include <cstring>

00004

00005 #include "contestants\_printers.h"

00006

00007 #include "../../utils/contestant.h"

00008 #include "../../utils/debug.h"

00009 #include "../../utils/settings.h"

00010 #include "../../utils/utils.h"

00011 using namespace std;

00012

00013

00014

00018 int getNextFreeSlot(Contestants contestant[], int startIdx = 0);

00019

00023 int getNextUsedSlot(Contestants contestant[], int startIdx = 0);

00024

00029 bool deleteContestantsByID(Contestants contestant[],const int& ID);

00030

00034 int biggestContestantID (Contestants contestant[]);

00035

00039 int countOfContestants(Contestants contestant[]);

00040

00045 bool searchContestantsByAge(Contestants contestant[],const int& age);

00046

00051 bool searchContestantsByWoman(Contestants contestant[],const bool& isWoman);

00052

00059 bool searchContestantByName(Contestants contestant[], char nameToSearch[], int sizeToSearch, string didYouMeanNames[]);

00060

00064 int searchLowestAge (Contestants contestant[]);

00065

00066

## BeautyContest/logic/exit/exit\_logic.cpp Файл Справка

Handles the exit sequence and prompts to save unsaved changes.

#include "exit\_logic.h"

### Функции

* char cinCheckYN (string message)

*Checks if user typed yes , no or cancel gets the first char to validate and is case insensitive.*

* bool askUserExit (Contestants contestant[], bool savedChanages)

*Logic for exit menu Ask if you want tos ave unsaved changes.*

### Подробно описание

Handles the exit sequence and prompts to save unsaved changes.

Виж дефиницията във файла exit\_logic.cpp.

### Функции Документация

#### bool askUserExit (Contestants contestant[], bool savedChanages)

Logic for exit menu Ask if you want tos ave unsaved changes.

##### Аргументи

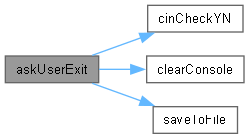
|  |  |
| --- | --- |
| *contestant* | What you want to save |
| *savedChanages* | Whether changes were saved or not |

##### Връща

Whether to stop the program or to continue it

Виж дефиницията във файла exit\_logic.cpp ред 9.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### char cinCheckYN (string message)

Checks if user typed yes , no or cancel gets the first char to validate and is case insensitive.

##### Аргументи

|  |  |
| --- | --- |
| *message* | What message you want to output |

##### Връща

what is the persons answer

Виж дефиницията във файла exit\_logic.cpp ред 46.

Граф на извикванията за тази функция:



## exit\_logic.cpp

Вижте документацията за този файл.

00001 #include "exit\_logic.h"

00002

00006 char cinCheckYN(string message);

00007

00008

00009 bool askUserExit(Contestants contestant[], bool savedChanages){

00010

00011 bool exiting = false;

00012 if (savedChanages){

00013

00014 exiting = (cinCheckYN("Do you want to exit? yes / no") == 'y' ? true : false);

00015

00016 } else {

00017

00018 cout << "You have unsaved changes!" << endl;

00019 char answer = cinCheckYN("Do you want to save them in files/auto\_save.dat? yes / no / Cancel");

00020

00021 if (answer == 'y'){

00022 clearConsole(CLEARCONSOLE);

00023 if (saveToFile(contestant, "zfiles/auto\_save.dat")){

00024 cout << "Data saved successfully!" << endl;

00025 exiting = true;

00026 } else {

00027 cout << "Couldn't save!" << endl;

00028 exiting = (cinCheckYN("Do you want to exit? yes / no") == 'y' ? true : false);

00029 }

00030 } else if(answer == 'n'){

00031 exiting = true;

00032

00033 } else if(answer == 'c'){

00034 exiting = false;

00035 }

00036 }

00037

00038 if (exiting){

00039

00040 cout << "Goodbye!" << endl;

00041 }

00042 return exiting;

00043 }

00044

00045

00046 char cinCheckYN(string message){

00047 string input;

00048 char answer;

00049

00050 do{

00051 cout << message << endl;

00052 getline(cin, input);

00053 answer = tolower(input[0]);

00054

00055 if (answer == 'y') { return 'y'; }

00056 else if (answer == 'n') { return 'n'; }

00057 else if (answer == 'c') { return 'c'; }

00058 else { cout << "Invalid input! Try Again!"; }

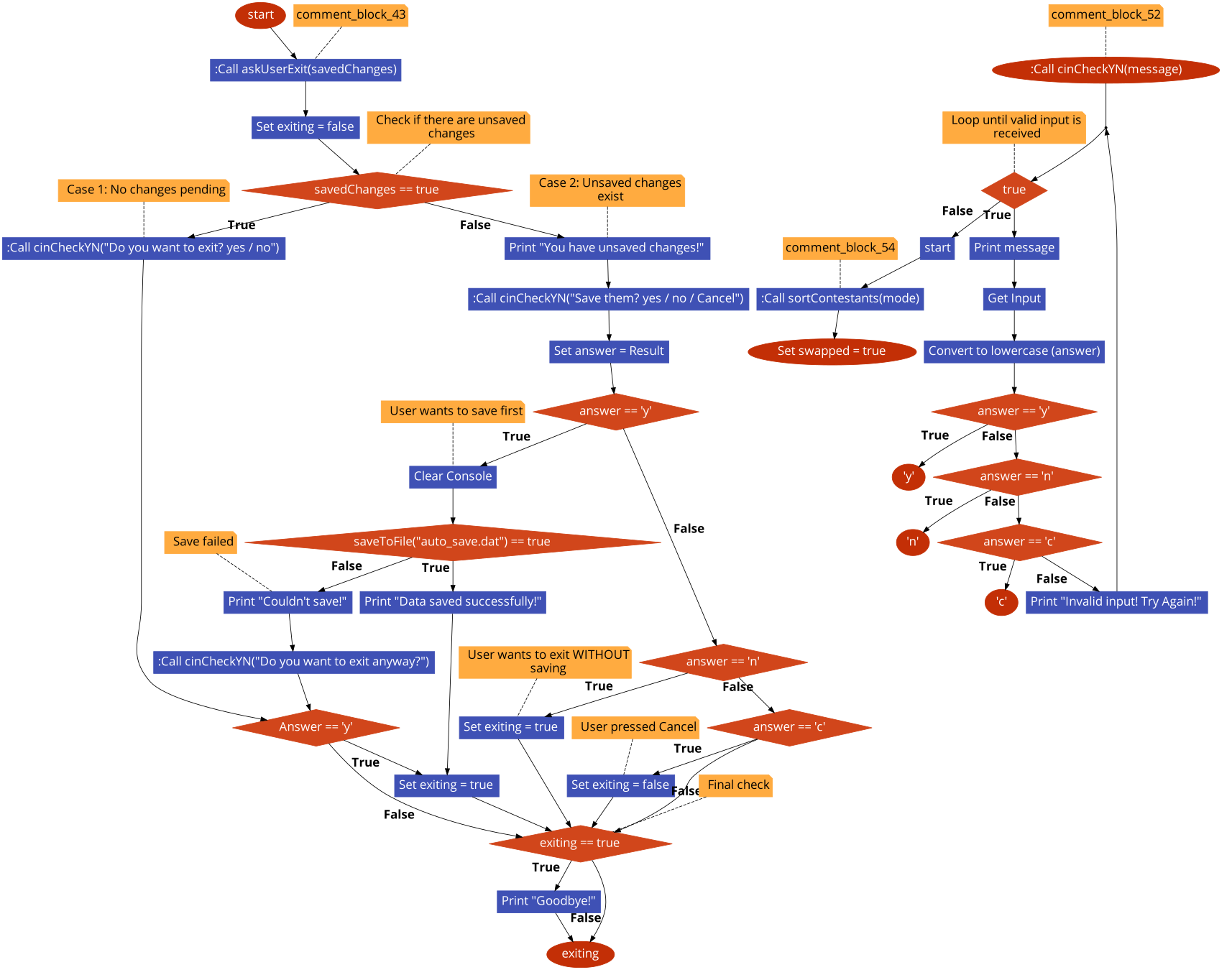
00059

00060 } while (answer != '0');

00061 return 'c';

00062 }

00063



## exit\_logic.h

00001

00003

00004 #pragma once

00005 #include <iostream>

00006 #include <string>

00007

00008 #include "../../utils/visual\_enhancement.h"

00009 #include "../../utils/debug.h"

00010 #include "../../utils/settings.h"

00011 #include "../../utils/contestant.h"

00012 #include "../../utils/file\_manager/file\_manager.h"

00013

00014 using namespace std;

00015

00020 bool askUserExit(Contestants contestant[], bool savedChanages);

## BeautyContest/logic/file/file\_menu.cpp Файл Справка

Implementation of the File Manager menu interactions.

#include "file\_menu.h"

### Функции

* void fileManagerMenu (Contestants contestant[], int &menuChoice, bool &savedChanages, bool &madeCategories, bool &winnersDecided, int &IDcounter, int &leftSpaces)

*Logic for managing the file menu.*

### Подробно описание

Implementation of the File Manager menu interactions.

Виж дефиницията във файла file\_menu.cpp.

### Функции Документация

#### void fileManagerMenu (Contestants contestant[], int & menuChoice, bool & savedChanages, bool & madeCategories, bool & winnersDecided, int & IDcounter, int & leftSpaces)

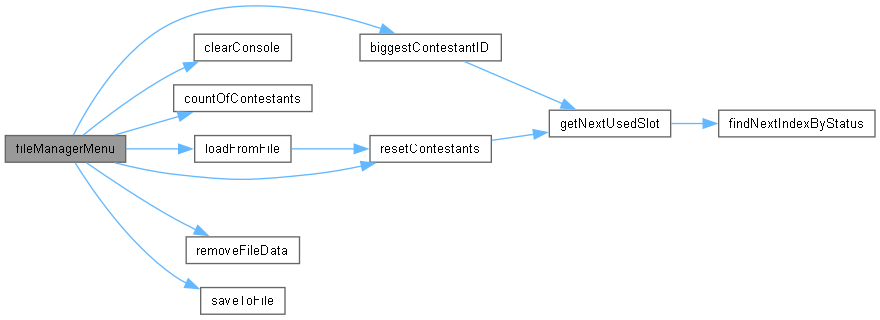
Logic for managing the file menu.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | What you want to manipulate |
| *menuChoice* | What was the users wish |

Виж дефиницията във файла file\_menu.cpp ред 6.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



## file\_menu.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "file\_menu.h"

00005

00006 void fileManagerMenu(Contestants contestant[], int& menuChoice, bool& savedChanages, bool& madeCategories, bool& winnersDecided, int& IDcounter, int& leftSpaces){

00007

00008 clearConsole(CLEARCONSOLE);

00009

00010 switch(menuChoice){

00011 case 0:

00012 {

00013 return;

00014 break;

00015 }

00016 case 1:

00017 {

00018 if (saveToFile(contestant,"zfiles/contestants.dat")){

00019 cout << "Data saved successfully!" << endl;

00020

00021 savedChanages = true;

00022 }

00023

00024 break;

00025 }

00026 case 2:

00027 {

00028 if (loadFromFile(contestant, "zfiles/contestants.dat")){

00029 cout << "Data loaded successfully!" << endl;

00030

00031 IDcounter = biggestContestantID(contestant) + 1;

00032 deBugInfo("SYSTEM: IDcounter: " << IDcounter << endl);

00033

00034 leftSpaces = countOfContestants(contestant);

00035 deBugInfo("SYSTEM: leftSpaces: " << leftSpaces << endl);

00036

00037 madeCategories = false;

00038 winnersDecided = false;

00039 savedChanages = true;

00040 }

00041

00042 break;

00043 }

00044 case 3:

00045 {

00046 if (removeFileData("zfiles/auto\_save.dat")){

00047 cout << "Scraped auto\_save.dat successfully!" << endl;

00048

00049 savedChanages = false;

00050 }

00051

00052 break;

00053 }

00054 case 4:

00055 {

00056 if (removeFileData("zfiles/contestants.dat")){

00057 cout << "Scraped contestants.dat successfully!" << endl;

00058

00059 savedChanages = false;

00060 }

00061

00062 break;

00063 }

00064 case 5:

00065 {

00066 resetContestants(contestant);

00067 cout << "Scraped all contestants!" << endl;

00068

00069 leftSpaces = MAXCONTESTANTS;

00070 madeCategories = false;

00071 winnersDecided = false;

00072 savedChanages = false;

00073

00074 break;

00075 }

00076 case 6:

00077 {

00078 if (removeFileData("zfiles/contestants.dat")){

00079 cout << "Scraped contestants.dat successfully!" << endl;

00080 }

00081

00082 if (removeFileData("zfiles/auto\_save.dat")){

00083 cout << "Scraped auto\_save.dat successfully!" << endl;

00084 }

00085

00086 resetContestants(contestant);

00087 cout << "Scraped all contestants!" << endl;

00088

00089 leftSpaces = MAXCONTESTANTS;

00090 madeCategories = false;

00091 winnersDecided = false;

00092 savedChanages = false;

00093 IDcounter = 1;

00094

00095 break;

00096 }

00097 default:

00098 {

00099 clearConsole(CLEARCONSOLE);

00100 cout << "Invalid option!" << endl;

00101 deBugInfo("ERROR: expected from 0 - 4 got: " << menuChoice);

00102

00103 break;

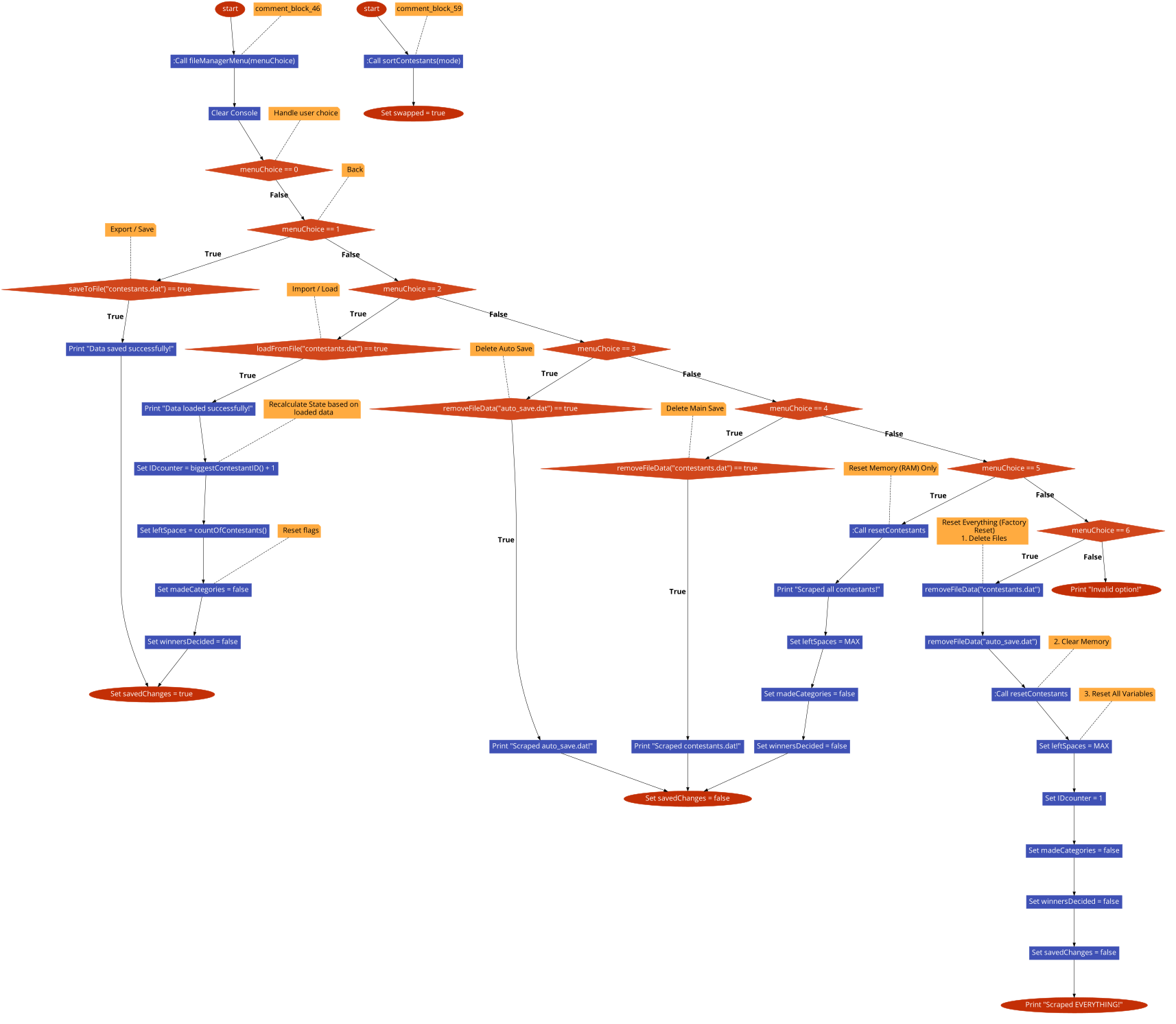
00104 }

00105 }

00106

00107

00108 }



## file\_menu.h

00001 #pragma once

00002 #include <iostream>

00003

00004 using namespace std;

00005

00006 #include "../../utils/file\_manager/file\_manager.h"

00007 #include "../../utils/visual\_enhancement.h"

00008 #include "../../utils/debug.h"

00009 #include "../contestants\_manipulation/contestants\_manager.h"

00010

00011

00015 void fileManagerMenu(Contestants contestant[], int& menuChoice, bool& savedChanages, bool& madeCategories, bool& winnersDecided, int& IDcounter, int& leftSpaces);

## BeautyContest/logic/search\_contestant/search\_show\_contestant\_menu.cpp Файл Справка

Logic for the Search menu and displaying search results.

#include "search\_show\_contestant\_menu.h"

### Функции

* void unsuccessfulSearch (bool event, string startingMessage, char s[], string didYouMeanNames[])

*Did you mean functions if name is close to one from the contestants list.*

* void unsuccessfulSearch (bool event, string startingMessage, string num)

*Simpler unsuccessful messenger function.*

* void searchShowContestantsMenu (Contestants contestant[], int &menuChoice)

*Logic for contestants menu. Has a did you mean feature when searching for names.*

### Подробно описание

Logic for the Search menu and displaying search results.

Виж дефиницията във файла search\_show\_contestant\_menu.cpp.

### Функции Документация

#### void searchShowContestantsMenu (Contestants contestant[], int & menuChoice)

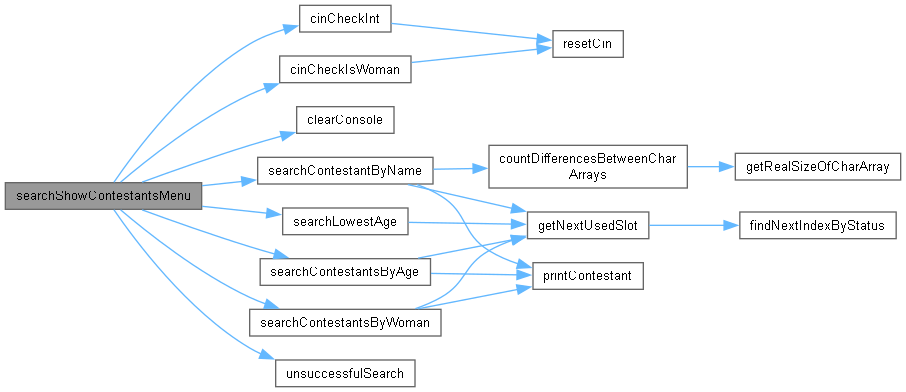
Logic for contestants menu. Has a did you mean feature when searching for names.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | Where you want to search |
| *menuChoice* | What user wants to search |

Виж дефиницията във файла search\_show\_contestant\_menu.cpp ред 20.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void unsuccessfulSearch (bool event, string startingMessage, char s[], string didYouMeanNames[])

Did you mean functions if name is close to one from the contestants list.

##### Аргументи

|  |  |
| --- | --- |
| *event* | If the search was successful or not |
| *startingMessage* | Whit what you want to start your unsuccessful search message |
| *s* | What name user searched for |
| *didYouMeanNames* | List of possible names user maybe wanted to type |

Виж дефиницията във файла search\_show\_contestant\_menu.cpp ред 90.

Граф на извикванията за тази функция:



#### void unsuccessfulSearch (bool event, string startingMessage, string num)

Simpler unsuccessful messenger function.

##### Аргументи

|  |  |
| --- | --- |
| *event* | Weather the search was successful or not |
| *startingMessage* | How you want to start the unsuccessfulness |
| *num* | Parameter user searched for |

Виж дефиницията във файла search\_show\_contestant\_menu.cpp ред 112.

## search\_show\_contestant\_menu.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "search\_show\_contestant\_menu.h"

00005

00006

00012 void unsuccessfulSearch(bool event, string startingMessage, char s[], string didYouMeanNames[]);

00013

00018 void unsuccessfulSearch(bool event, string startingMessage, string num);

00019

00020 void searchShowContestantsMenu(Contestants contestant[], int& menuChoice){

00021 switch(menuChoice){

00022 case 0:

00023 {

00024 clearConsole();

00025 return;

00026 break;

00027 }

00028 case 1:

00029 {

00030 clearConsole(CLEARCONSOLE);

00031

00032 cout << "Contestant/s with lowest age: " << endl;

00033

00034 int lowestAge = searchLowestAge(contestant);

00035 bool foundPerson = searchContestantsByAge(contestant, lowestAge);

00036

00037 break;

00038 }

00039 case 2:

00040 {

00041 clearConsole(CLEARCONSOLE);

00042

00043 int age;

00044 age = cinCheckInt("Enter age of contestant: ");

00045

00046 bool foundPerson = searchContestantsByAge(contestant, age);

00047

00048 unsuccessfulSearch(!foundPerson, "Person with age ", to\_string(age));

00049 break;

00050 }

00051 case 3:

00052 {

00053

00054 clearConsole(CLEARCONSOLE);

00055

00056

00057 cout << "Enter name of contestant: ";

00058

00059 char nameToSearch[MAXNAMECHARS];

00060 cin >> nameToSearch;

00061

00062 string didYouMeanNames[MAXCONTESTANTS];

00063

00064 bool foundPerson = searchContestantByName(contestant, nameToSearch, sizeof(nameToSearch), didYouMeanNames);

00065

00066 unsuccessfulSearch(!foundPerson, "Person with name ", nameToSearch , didYouMeanNames);

00067 break;

00068 }

00069 case 4:

00070 {

00071 clearConsole(CLEARCONSOLE);

00072 bool gender = cinCheckIsWoman("Enter gender to search for (m/f): ");

00073

00074 bool foundPerson = searchContestantsByWoman(contestant, gender);

00075

00076 unsuccessfulSearch(!foundPerson, "Person with gender ", gender ? "f" : "m");

00077 break;

00078 }

00079 default:

00080 {

00081 clearConsole();

00082 cout << "Invalid option!" << endl;

00083 deBugInfo("ERROR: expected from 0 - 4 got: " << menuChoice);

00084 break;

00085 }

00086 }

00087 }

00088

00089

00090 void unsuccessfulSearch(bool event, string startingMessage, char s[], string didYouMeanNames[]){

00091 if (event){

00092 cout << startingMessage << s << " has not been found.\n";

00093

00094 //DID you mean logic

00095 bool haveSuggestions = didYouMeanNames[0] != "";

00096 if(haveSuggestions){

00097 cout << "Did you mean ";

00098 bool nextIsEmpty = false;

00099 for(int i = 0; !nextIsEmpty && MAXCONTESTANTS - 1 >= i; i++){

00100 string name = didYouMeanNames[i];

00101 nextIsEmpty = (name == "");

00102

00103 if(i != 0 && !nextIsEmpty) cout <<" or ";

00104 cout << name;

00105

00106 }

00107 cout << ".\n";

00108 }

00109 }

00110 }

00111

00112 void unsuccessfulSearch(bool event, string startingMessage, string num){

00113 if (event){

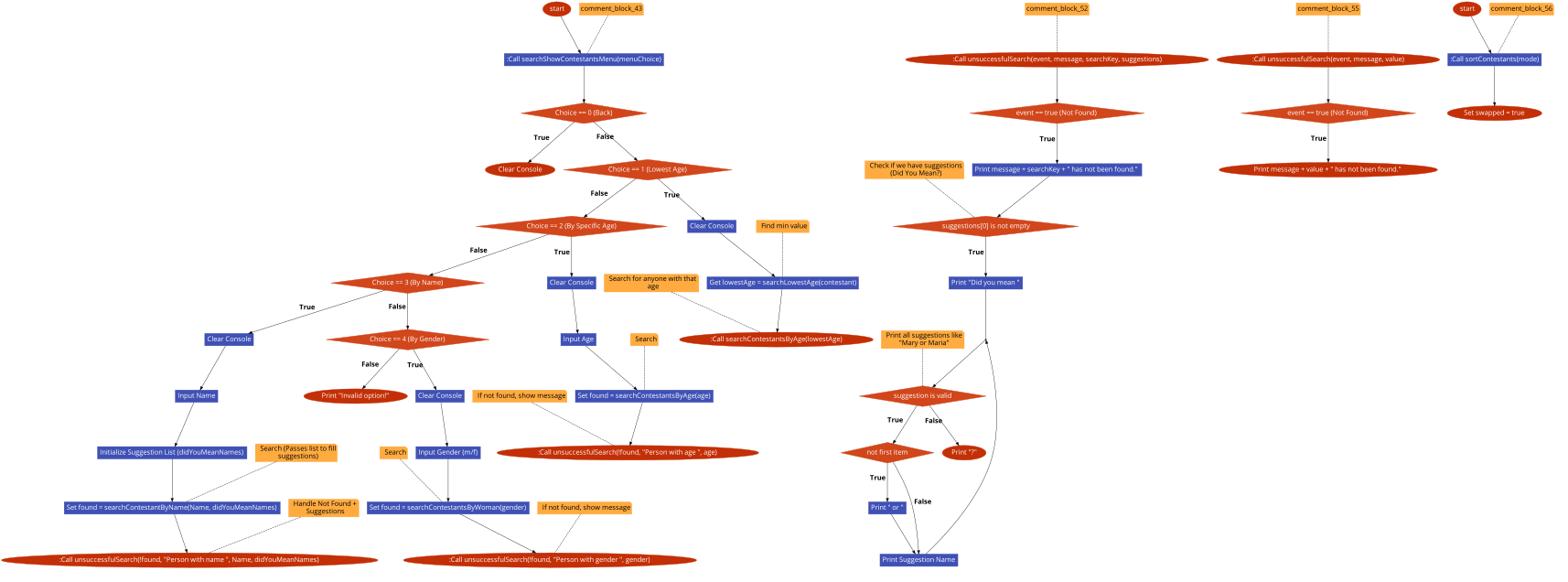
00114 cout << startingMessage << num << " has not been found.\n";

00115 }

00116 }

00117

00118



## search\_show\_contestant\_menu.h

00001 #pragma once

00002 #include <iostream>

00003 #include <string>

00004 #include <cstring>

00005 #include <cmath>

00006

00007 using namespace std;

00008

00009 #include "../contestants\_manipulation/contestants\_searchers.h"

00010

00011 #include "../../utils/visual\_enhancement.h"

00012 #include "../../utils/settings.h"

00013 #include "../../utils/contestant.h"

00014 #include "../../utils/cin\_validators.h"

00015 #include "../../utils/utils.h"

00016

00020 void searchShowContestantsMenu(Contestants contestant[], int& menuChoice);

## BeautyContest/logic/settings/settings\_menu\_manager.cpp Файл Справка

Handles user interactions for changing program settings (Debug, AutoSave, etc.).

#include "settings\_menu\_manager.h"

### Функции

* void settingsMenu (int menuChoice)

*Logic for changing the settings.*

### Подробно описание

Handles user interactions for changing program settings (Debug, AutoSave, etc.).

Виж дефиницията във файла settings\_menu\_manager.cpp.

### Функции Документация

#### void settingsMenu (int menuChoice)

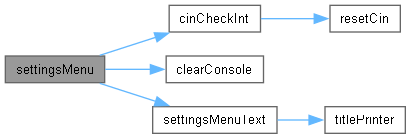
Logic for changing the settings.

##### Аргументи

|  |  |
| --- | --- |
| *menuChoice* | Which settings the user wants to change |

Виж дефиницията във файла settings\_menu\_manager.cpp ред 6.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



## settings\_menu\_manager.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "settings\_menu\_manager.h"

00005

00006 void settingsMenu (int menuChoice){

00007

00008 while (true){

00009

00010 switch (menuChoice){

00011 case 0:

00012 {

00013 clearConsole();

00014

00015 return;

00016 break;

00017 }

00018 case 1:

00019 {

00020 clearConsole();

00021 DEBUGMODE = !DEBUGMODE;

00022

00023 if(DEBUGMODE){

00024 cout << "DebugMode on!\n";

00025 } else {

00026 cout << "DebugMode off!\n";

00027 }

00028

00029 break;

00030 }

00031 case 2:

00032 {

00033 clearConsole();

00034 CLEARCONSOLE = !CLEARCONSOLE;

00035

00036 if(CLEARCONSOLE){

00037 cout << "Clear console on!\n";

00038 } else {

00039 cout << "Clear console off!\n";

00040 }

00041

00042 break;

00043 }

00044 case 3:

00045 {

00046 clearConsole();

00047 AUTOSAVE = !AUTOSAVE;

00048

00049 if(AUTOSAVE){

00050 cout << "Auto save on!\n";

00051 } else {

00052 cout << "Auto save off!\n";

00053 }

00054

00055 break;

00056 clearConsole();

00057 }

00058 case 4:

00059 {

00060 clearConsole();

00061 AUTOCATEGORIZE = !AUTOCATEGORIZE;

00062

00063 if(AUTOCATEGORIZE){

00064 cout << "Auto categorize on!\n";

00065 } else {

00066 cout << "Auto categorize off!\n";

00067 }

00068

00069 break;

00070 }

00071 default:

00072 {

00073 clearConsole();

00074 cout << "Invalid option!" << endl;

00075 deBugInfo("ERROR: expected from 0 - 6 got: " << menuChoice);

00076 break;

00077 }

00078 }

00079 settingsMenuText();

00080 menuChoice = cinCheckInt("Type a number to chose your action: ");

00081 }

00082

00083 }

## settings\_menu\_manager.h

00001 #pragma once

00002 #include <iostream>

00003

00004 using namespace std;

00005

00006 #include "../../utils/settings.h"

00007 #include "../../utils/debug.h"

00008 #include "../../utils/visual\_enhancement.h"

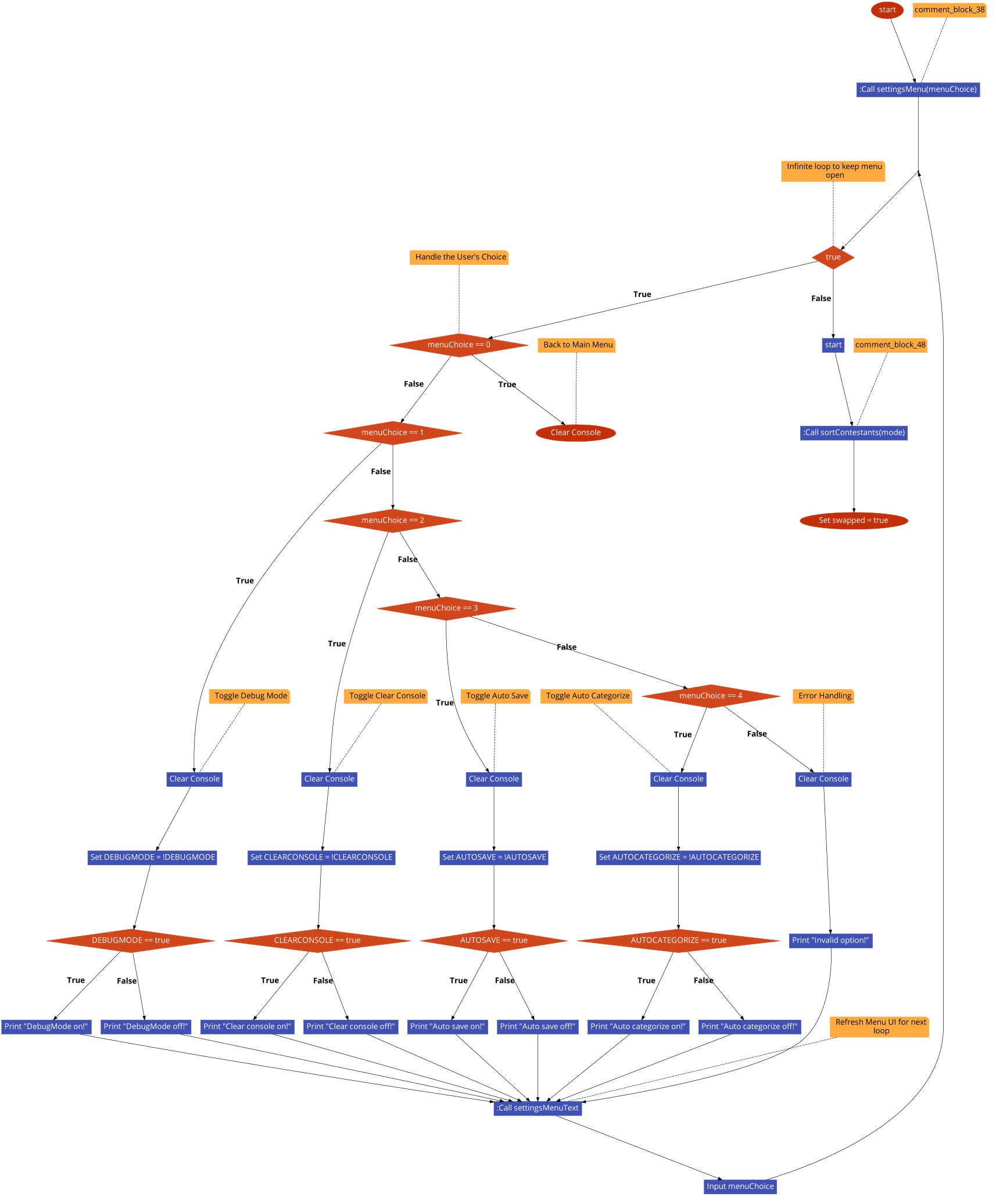
00009 #include "../../utils/cin\_validators.h"

00010 #include "../../block\_mangment.h"

00011

00014 void settingsMenu (int menuChoice);

00015



## BeautyContest/logic/show\_contestants/show\_contestants.cpp Файл Справка

Logic for displaying various lists of contestants (All, Winners, Categories).

#include "show\_contestants.h"

### Функции

* void showContestantsMenu (Contestants contestant[], Contestants winners[], Contestants category14\_16[], Contestants category17\_19[], Contestants category20\_22[], Contestants category23\_25[], int menuChoice, bool winnersDecided, bool madeCategories)

*Manages logic for the printing the contestants. Outputs a message if category or winners weren't defined yet.*

### Подробно описание

Logic for displaying various lists of contestants (All, Winners, Categories).

Виж дефиницията във файла show\_contestants.cpp.

### Функции Документация

#### void showContestantsMenu (Contestants contestant[], Contestants winners[], Contestants category14\_16[], Contestants category17\_19[], Contestants category20\_22[], Contestants category23\_25[], int menuChoice, bool winnersDecided, bool madeCategories)

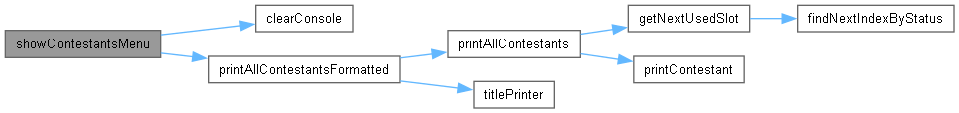
Manages logic for the printing the contestants. Outputs a message if category or winners weren't defined yet.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | Contestants to be printed |
| *winners* | Winners to be printed |
| *category14\_16* | First age group to be printed |
| *category17\_19* | Secund age group to be printed |
| *category20\_22* | Third age group to be printed |
| *category23\_25* | Forth age group to be printed |
| *menuChoice* | What user wants to print |
| *winnersDecided* | Are winners decided to be printed |
| *madeCategories* | Are categories made to be printed |

Виж дефиницията във файла show\_contestants.cpp ред 6.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



## show\_contestants.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "show\_contestants.h"

00005

00006 void showContestantsMenu(Contestants contestant[], Contestants winners[], Contestants category14\_16[], Contestants category17\_19[], Contestants category20\_22[], Contestants category23\_25[], int menuChoice, bool winnersDecided, bool madeCategories){

00007

00008 switch(menuChoice){

00009 case 0:

00010 {

00011 clearConsole();

00012 return;

00013 break;

00014 }

00015 case 1:

00016 {

00017 clearConsole();

00018 printAllContestantsFormatted(contestant, "ALL CONTESTANTS");

00019 break;

00020 }

00021 case 2:

00022 {

00023 clearConsole();

00024 if (!madeCategories){

00025 cout << "Winners have not been decided yet!\n";

00026 break;

00027 }

00028 printAllContestantsFormatted(winners, "WINNERS");

00029 break;

00030 }

00031 case 3:

00032 {

00033 clearConsole();

00034 if (!winnersDecided){

00035 cout << "Categories have not been made yet!\n";

00036 break;

00037 }

00038

00039 printAllContestantsFormatted(category14\_16, "Category 14/16 years");

00040

00041 printAllContestantsFormatted(category17\_19, "Category 17/19 years");

00042

00043 printAllContestantsFormatted(category20\_22, "Category 20/22 years");

00044

00045 printAllContestantsFormatted(category23\_25, "Category 23/25 years");

00046 break;

00047

00048 }

00049 default:

00050 {

00051 clearConsole();

00052 cout << "Invalid option!" << endl;

00053 deBugInfo("ERROR: expected from 0 - 3 got: " << menuChoice);

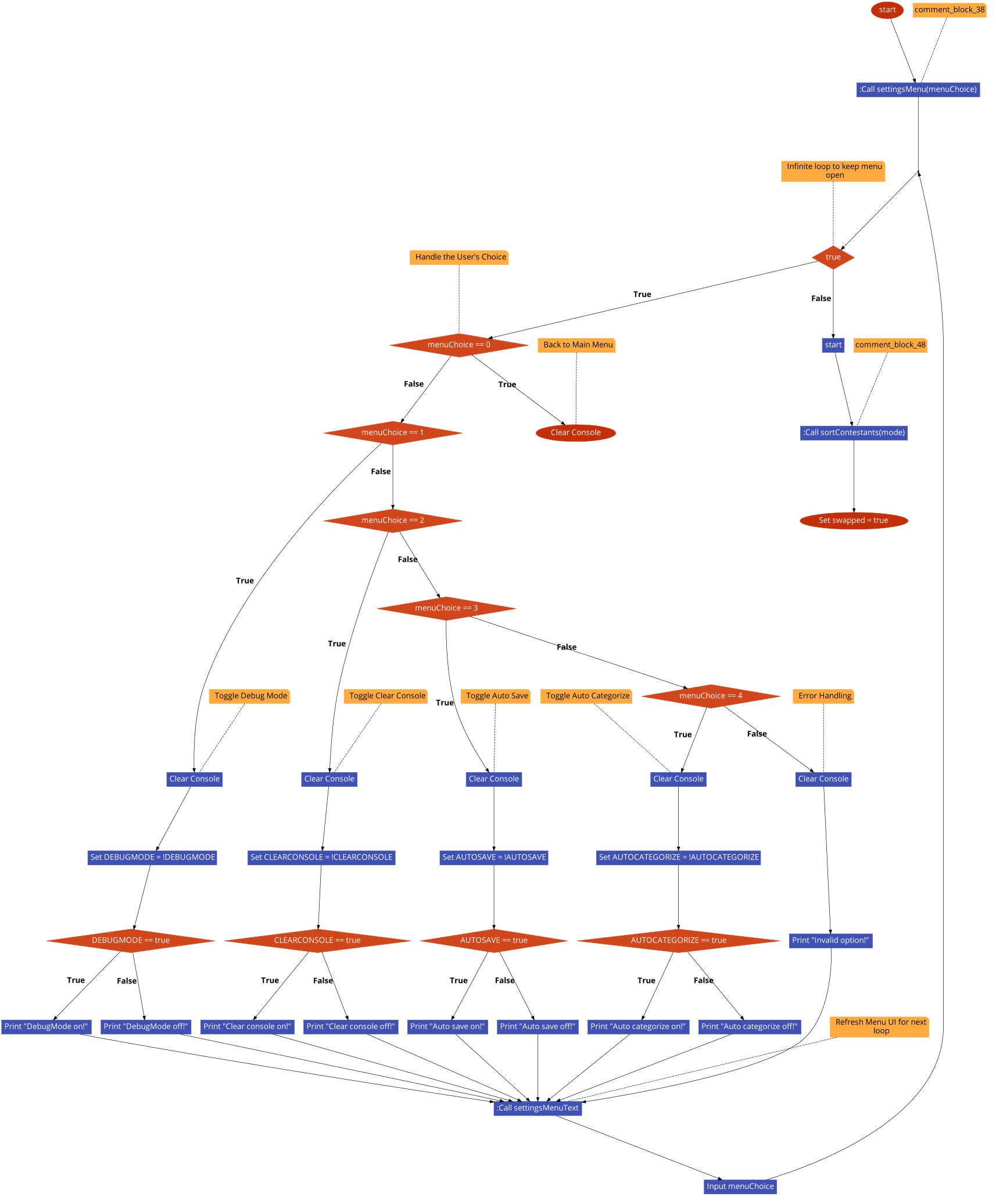
00054 break;

00055 }

00056 }

00057

00058 }



## show\_contestants.h

00001 #pragma once

00002 #include <iostream>

00003 #include <string>

00004

00005 #include "../contestants\_manipulation/contestants\_printers.h"

00006

00007 #include "../../block\_mangment.h"

00008

00009 #include "../../utils/settings.h"

00010 #include "../../utils/contestant.h"

00011

00022 void showContestantsMenu(Contestants contestant[], Contestants winners[], Contestants category14\_16[], Contestants category17\_19[], Contestants category20\_22[], Contestants category23\_25[], int menuChoice, bool winnersDecided, bool madeCategories);

## BeautyContest/logic/sorting/sorting\_menu.cpp Файл Справка

Implementation of the Sorting menu logic.

#include "sorting\_menu.h"

### Функции

* void outputSortedNoChangeAlphabetically (Contestants contestant[])

*Sorts contestants by name descending without changing the original.*

* void outputSortedNoChangeAge (Contestants contestant[])

*Sorts contestants by age ascending without changing the original.*

* void sortingMenu (Contestants contestant[], int &menuChoice, bool &madeChanges)

*Manages the logic for the sorting menu.*

### Подробно описание

Implementation of the Sorting menu logic.

Виж дефиницията във файла sorting\_menu.cpp.

### Функции Документация

#### void outputSortedNoChangeAge (Contestants contestant[])

Sorts contestants by age ascending without changing the original.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | Contestants to be outputted |

Виж дефиницията във файла sorting\_menu.cpp ред 77.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void outputSortedNoChangeAlphabetically (Contestants contestant[])

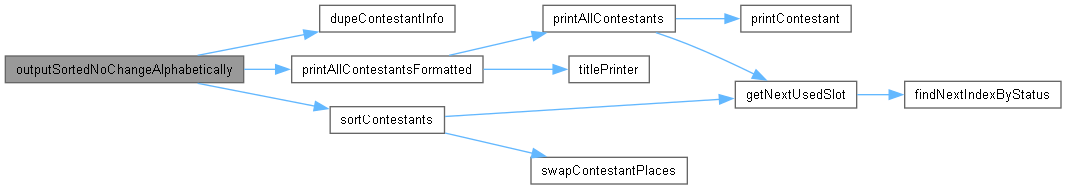
Sorts contestants by name descending without changing the original.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | Contestants to be outputted |

Виж дефиницията във файла sorting\_menu.cpp ред 67.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### void sortingMenu (Contestants contestant[], int & menuChoice, bool & madeChanges)

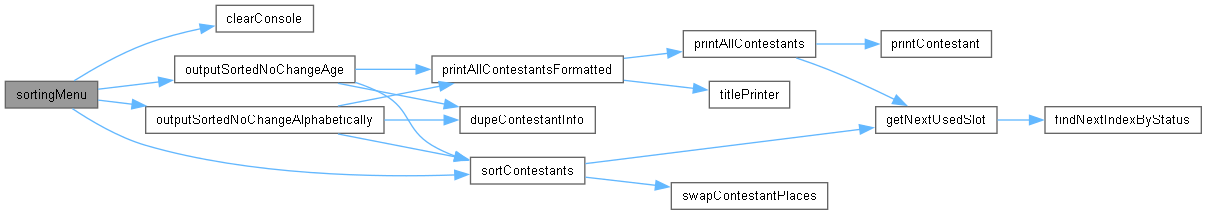
Manages the logic for the sorting menu.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | Contestants to sort |
| *menuChoice* | what user wants to sort |
| *madeChanges* | If user made any change to the inputed contestants |

Виж дефиницията във файла sorting\_menu.cpp ред 16.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



## sorting\_menu.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "sorting\_menu.h"

00005

00008 void outputSortedNoChangeAlphabetically(Contestants contestant[]);

00009

00012 void outputSortedNoChangeAge(Contestants contestant[]);

00013

00014

00015

00016 void sortingMenu (Contestants contestant[], int& menuChoice, bool& madeChanges){

00017

00018 clearConsole(CLEARCONSOLE);

00019

00020 switch (menuChoice){

00021 case 0:

00022 {

00023 return;

00024 break;

00025 }

00026 case 1:

00027 {

00028 sortContestants(contestant,SORT\_BY\_AGE\_ASC);

00029 cout << "Contestants sorted successfully!\n";

00030 madeChanges = false;

00031 break;

00032 }

00033 case 2:

00034 {

00035 sortContestants(contestant, SORT\_BY\_NAME);

00036 madeChanges = false;

00037 break;

00038 }

00039 case 3:

00040 {

00041 sortContestants(contestant, SORT\_BY\_ID);

00042 madeChanges = false;

00043 break;

00044 }

00045 case 4:

00046 {

00047 outputSortedNoChangeAge(contestant);

00048 break;

00049 }

00050 case 5:

00051 {

00052 outputSortedNoChangeAlphabetically(contestant);

00053 break;

00054 }

00055 default:

00056 {

00057

00058 cout << "Invalid option!" << endl;

00059 deBugInfo("ERROR: expected from 0 - 5 got: " << menuChoice);

00060 break;

00061 }

00062 }

00063 }

00064

00065

00066

00067 void outputSortedNoChangeAlphabetically(Contestants contestant[]){

00068 Contestants alphSortedCont[MAXCONTESTANTS];

00069 dupeContestantInfo(contestant, alphSortedCont);

00070

00071 sortContestants(alphSortedCont, SORT\_BY\_NAME);

00072

00073 printAllContestantsFormatted(alphSortedCont, "CONTESTANTS SORTED BY NAME");

00074 }

00075

00076

00077 void outputSortedNoChangeAge(Contestants contestant[]){

00078 Contestants alphSortedCont[MAXCONTESTANTS];

00079 dupeContestantInfo(contestant, alphSortedCont);

00080

00081 sortContestants(alphSortedCont, SORT\_BY\_AGE\_ASC);

00082

00083 printAllContestantsFormatted(alphSortedCont,"CONTESTANTS SORTED BY AGE");

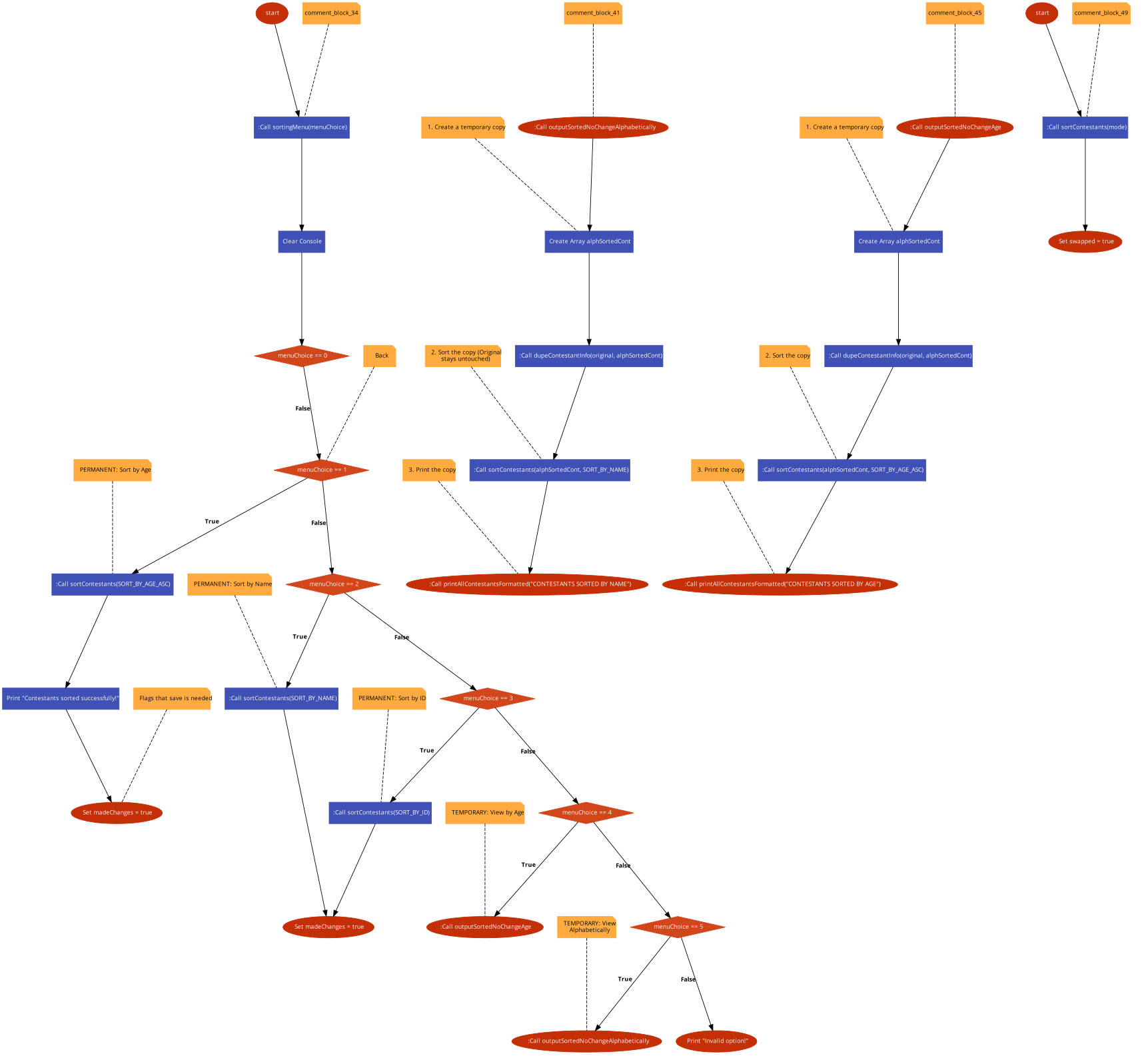
00084 }

00085

00086

00087

00088



## sorting\_menu.h

00001 #pragma once

00002 #include <iostream>

00003 #include <string>

00004

00005

00006 #include "../contestants\_manipulation/contestants\_printers.h"

00007 #include "../contestants\_manipulation/contestants\_searchers.h"

00008 #include "../contestants\_manipulation/contestants\_manager.h"

00009 #include "../contestants\_manipulation/contestans\_sorters.h"

00010

00011

00012

00013 #include "../../block\_mangment.h"

00014

00015 #include "../../utils/settings.h"

00016 #include "../../utils/contestant.h"

00017

00018 using namespace std;

00019

00024 void sortingMenu (Contestants contestant[], int& menuChoice, bool& madeChanges);

00025

00026

## BeautyContest/logic/winner\_decision/winner\_decsision\_menu.cpp Файл Справка

Logic for calculating and displaying the winners.

#include "winner\_decsision\_menu.h"

### Функции

* void winnerDecider (Contestants contestant[], Contestants winners[])

*Deciding who are the winners losers and real winners.*

### Подробно описание

Logic for calculating and displaying the winners.

Виж дефиницията във файла winner\_decsision\_menu.cpp.

### Функции Документация

#### void winnerDecider (Contestants contestant[], Contestants winners[])

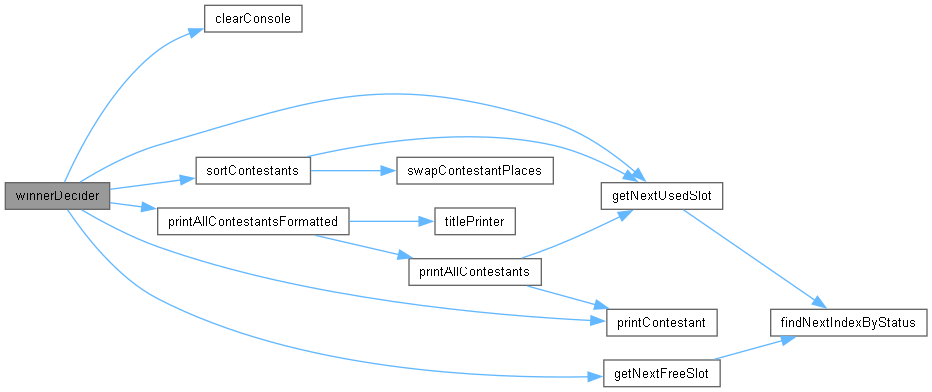
Deciding who are the winners losers and real winners.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | Contestants to decide form |
| *winners* | Where winners are saved for latter use |

Виж дефиницията във файла winner\_decsision\_menu.cpp ред 6.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



## winner\_decsision\_menu.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "winner\_decsision\_menu.h"

00005

00006 void winnerDecider (Contestants contestant[], Contestants winners[]){

00007

00008 Contestants tooGoodContestants[MAXCONTESTANTS];

00009 Contestants tooBadContestants[MAXCONTESTANTS];

00010 Contestants outSiders[MAXCONTESTANTS];

00011

00012 for(int i = getNextUsedSlot(contestant, 0); i != -1; i = getNextUsedSlot(contestant, i + 1)){

00013

00014 double contPoints = contestant[i].points;

00015 deBugInfo("SYSTEM: Contestant with points: " << contPoints);

00016 if (contPoints >= 0.54 && contPoints <= 0.62){

00017 winners[getNextFreeSlot(winners)] = contestant[i];

00018 deBugInfo("SYSTEM: Putting contestant into winners: ");

00019

00020 } else if(contPoints < 0.54){

00021 tooBadContestants[getNextFreeSlot(tooBadContestants)] = contestant[i];

00022 deBugInfo("SYSTEM: Putting contestant into tooBadContestants: ");

00023

00024 } else if(contPoints > 0.62){

00025 tooGoodContestants[getNextFreeSlot(tooGoodContestants)] = contestant[i];

00026 deBugInfo("SYSTEM: Putting contestant into tooGoodContestants: " << endl);

00027

00028 } else {

00029 outSiders[getNextFreeSlot(outSiders)] = contestant[i];

00030 deBugInfo("SYSTEM: Putting contestant into outSiders: ");

00031

00032

00033 }

00034 printContestant(contestant, i); //DEBUG

00035 cout << endl;

00036

00037 }

00038

00039 clearConsole();

00040

00041 sortContestants(winners, SORT\_BY\_POINTS\_DESC);

00042 sortContestants(tooBadContestants, SORT\_BY\_POINTS\_ASC);

00043 sortContestants(tooGoodContestants, SORT\_BY\_POINTS\_ASC);

00044

00045 printAllContestantsFormatted(winners, "WINNERS");

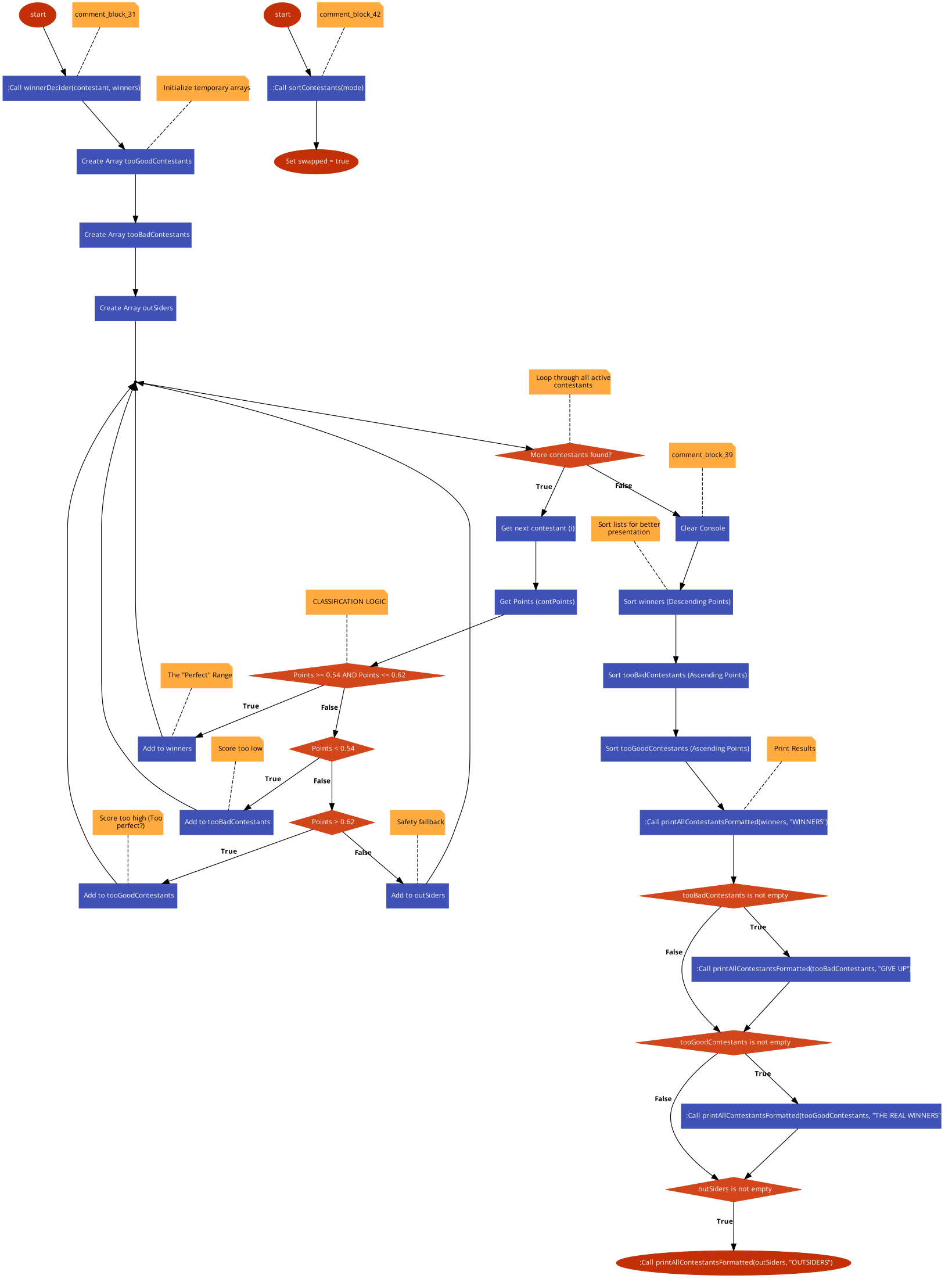
00046 if(getNextUsedSlot(tooBadContestants) != -1) printAllContestantsFormatted(tooBadContestants, "GIVE UP");

00047 if(getNextUsedSlot(tooGoodContestants) != -1) printAllContestantsFormatted(tooGoodContestants, "THE REAL WINNERS");

00048 if(getNextUsedSlot(outSiders) != -1) printAllContestantsFormatted(outSiders, "OUTSIDERS");

00049

00050 }



## winner\_decsision\_menu.h

00001 #pragma once

00002 #include <iostream>

00003

00004 using namespace std;

00005

00006 #include "../contestants\_manipulation/contestants\_printers.h"

00007 #include "../contestants\_manipulation/contestans\_sorters.h"

00008

00009

00010 #include "../../utils/cin\_validators.h"

00011 #include "../../utils/debug.h"

00012 #include "../../utils/contestant.h"

00013

00017 void winnerDecider (Contestants contestant[], Contestants winners[]);

00018

00019

## BeautyContest/utils/cin\_validators.cpp Файл Справка

Helper functions to validate user input from the console.

#include "cin\_validators.h"

### Функции

* void resetCin ()

*Resets cin of any junk till new line.*

* int cinCheckInt (const string &message)

*Check if user gave number, if not asks again for input.*

* double cinCheckDouble (const string &message)

*Check if user gave number, if not asks again for input.*

* bool cinCheckIsWoman (const string &message)

*Checks if you user wrote m or f.*

### Подробно описание

Helper functions to validate user input from the console.

Виж дефиницията във файла cin\_validators.cpp.

### Функции Документация

#### double cinCheckDouble (const string & message)

Check if user gave number, if not asks again for input.

##### Аргументи

|  |  |
| --- | --- |
| *message* | What to ask the user to input |

##### Връща

Double value of users input

Виж дефиницията във файла cin\_validators.cpp ред 28.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### int cinCheckInt (const string & message)

Check if user gave number, if not asks again for input.

##### Аргументи

|  |  |
| --- | --- |
| *message* | What to ask the user to input |

##### Връща

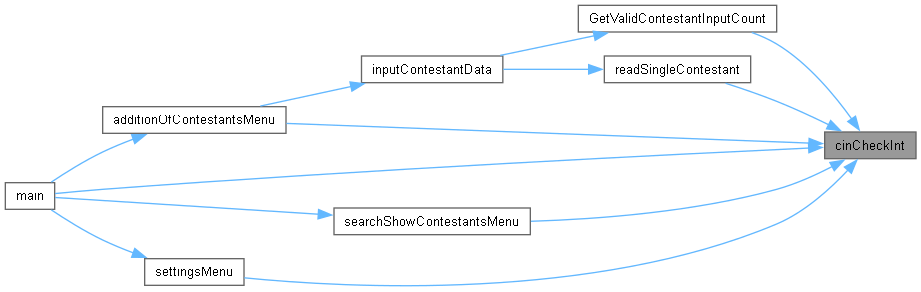
Integer value of users input

Виж дефиницията във файла cin\_validators.cpp ред 12.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### bool cinCheckIsWoman (const string & message)

Checks if you user wrote m or f.

##### Аргументи

|  |  |
| --- | --- |
| *message* | Message making user input data |

##### Връща

Bool value of what user inputted m - false, f - true

Виж дефиницията във файла cin\_validators.cpp ред 43.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:

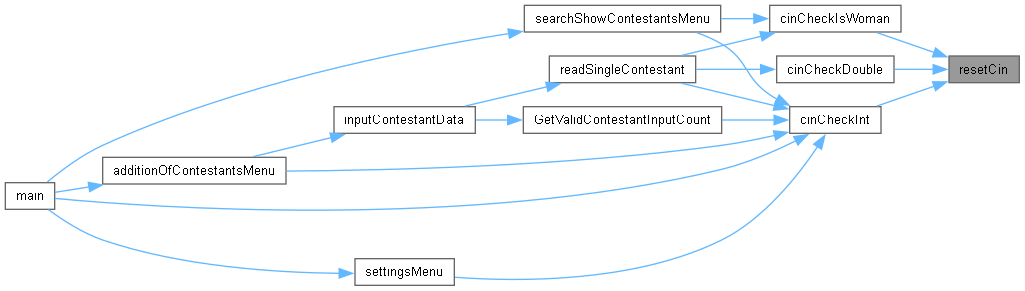


#### void resetCin ()

Resets cin of any junk till new line.

Виж дефиницията във файла cin\_validators.cpp ред 6.

Граф на извикванията за тази функция:



## cin\_validators.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "cin\_validators.h"

00005

00006 void resetCin (){

00007 cin.clear(); // fix the stream

00008 cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // yeet the garbage

00009 }

00010

00011

00012 int cinCheckInt(const string& message) {

00013 int value;

00014 while (true) {

00015 cout << message;

00016

00017 if (cin >> value) {

00018 resetCin();

00019 return value;

00020 } else {

00021 resetCin();

00022 cout << "Invalid input. Try again." << endl;

00023 }

00024 }

00025 }

00026

00027

00028 double cinCheckDouble(const string& message) {

00029 double value;

00030 while (true) {

00031 cout << message;

00032

00033 if (cin >> value) {

00034 resetCin();

00035 return value;

00036 } else {

00037 resetCin();

00038 cout << "Invalid input. Try again." << endl;

00039 }

00040 }

00041 }

00042

00043 bool cinCheckIsWoman(const string& message){

00044

00045 while(true) {

00046 cout << message;

00047

00048 char input = ' ';

00049 cin >> input;

00050

00051 deBugInfo("SYSTEM: input: " << input << " | ");

00052 char gender = tolower(input);

00053 deBugInfo("Gender: " << gender << endl);

00054

00055

00056 if (gender == 'm') {

00057 resetCin();

00058 return false;

00059 } else if (gender == 'f') {

00060 resetCin();

00061 return true;

00062 } else {

00063 cout << "Invalid gender. Try again!" << endl;

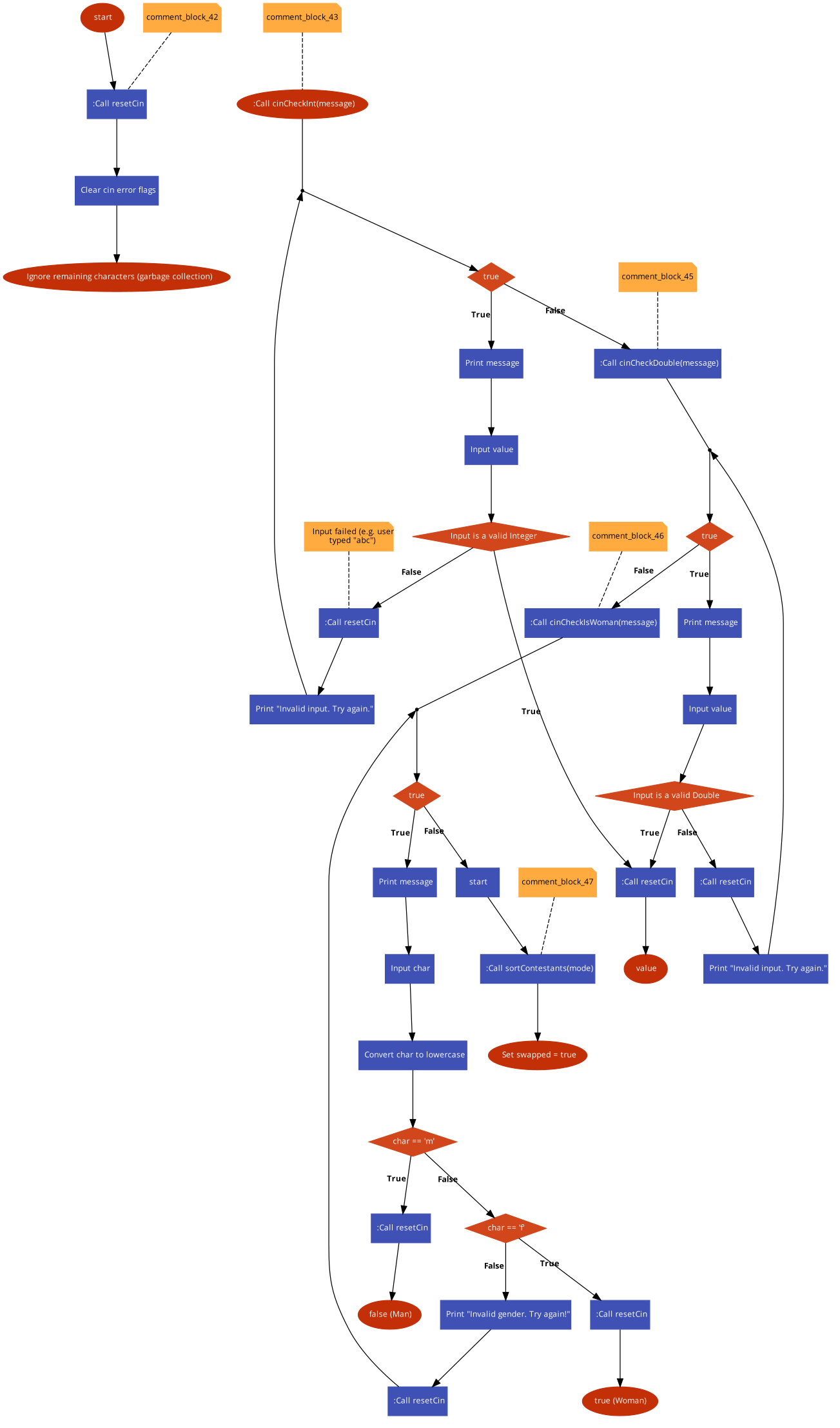
00064 resetCin();

00065 deBugInfo("ERROR: cin broke. Restarted" << endl);

00066 }

00067 }

00068 }



## cin\_validators.h

00001 #pragma once

00002 #include <iostream>

00003 #include <string>

00004 #include <limits>

00005

00006 #include "settings.h"

00007 #include "debug.h"

00008

00009

00011 void resetCin();

00012

00016 int cinCheckInt(const string& message);

00017

00021 double cinCheckDouble(const string& message);

00022

00026 bool cinCheckIsWoman(const string& message);

## BeautyContest/utils/contestant.h Файл Справка

Defines the Contestants structure and related data fields.

#include <string>

#include "settings.h"

### Класове

struct Contestants*A structure to hold all details for a single beauty contestant.*

### Подробно описание

Defines the Contestants structure and related data fields.

Виж дефиницията във файла contestant.h.

## contestant.h

Вижте документацията за този файл.

00001

00003

00004 #pragma once

00005 #include <string>

00006 #include "settings.h"

00007

00009 struct Contestants {

00010

00012 int ID;

00013

00015 bool isObjectUsed = false;

00016

00018 char name[MAXNAMECHARS];

00019

00021 int age = 0;

00022

00024 bool isWoman = false;

00025

00027 double hipCirc = 0.0;

00028

00030 double shoulderCirc = 0.0;

00031

00033 double calfCirc = 0.0;

00034

00036 double neckCirc = 0.0;

00037

00039 double points = 0.0;

00040 };

## BeautyContest/utils/debug.cpp Файл Справка

Debugging utilities for printing array values and status messages.

#include "debug.h"

### Функции

* void deBugStringArray (string s[], int size)

*Debugger that prints full array of strings.*

### Подробно описание

Debugging utilities for printing array values and status messages.

Виж дефиницията във файла debug.cpp.

### Функции Документация

#### void deBugStringArray (string s[], int size)

Debugger that prints full array of strings.

##### Аргументи

|  |  |
| --- | --- |
| *s* | What assay you want to print |
| *size* | The number of string in the array |

Виж дефиницията във файла debug.cpp ред 7.

Граф на извикванията за тази функция:



## debug.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "debug.h"

00005

00006

00007 void deBugStringArray (string s[],int size){

00008

00009 deBugInfo("SYSTEM: array values: ");

00010 for(int i = 0; i < size; i++)

00011 deBugInfo(i << " [" << s[i] << "], ");

00012 deBugInfo(endl);

00013 }

## debug.h

00001 #pragma once

00002 #include <iostream>

00003 #include <string>

00004

00005 #include "settings.h"

00006

00007 using namespace std;

00008

00010 #define deBugInfo(message) if (DEBUGMODE) { cout << message; }

00011

00015 void deBugStringArray (string s[], int size);

## BeautyContest/utils/file\_manager/file\_manager.cpp Файл Справка

Low-level file I/O operations (saving and loading binary data).

#include "file\_manager.h"

### Функции

* bool saveToFile (Contestants contestant[], string filePath)

*Saves file of contestnats.*

* bool loadFromFile (Contestants contestant[], string filePath)

*Loads file of contestnats.*

* bool removeFileData (string filePath)

*Remove all the data in a file.*

### Подробно описание

Low-level file I/O operations (saving and loading binary data).

Виж дефиницията във файла file\_manager.cpp.

### Функции Документация

#### bool loadFromFile (Contestants contestant[], string filePath)

Loads file of contestnats.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | Contestants to load |

##### Връща

If it was successful

Виж дефиницията във файла file\_manager.cpp ред 26.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### bool removeFileData (string filePath)

Remove all the data in a file.

##### Аргументи

|  |  |
| --- | --- |
| *filePath* | File you want to scrape |

##### Връща

If it was successful

Виж дефиницията във файла file\_manager.cpp ред 49.

Граф на извикванията за тази функция:



#### bool saveToFile (Contestants contestant[], string filePath)

Saves file of contestnats.

##### Аргументи

|  |  |
| --- | --- |
| *contestant* | Contestants to save |

##### Връща

If it was successful

Виж дефиницията във файла file\_manager.cpp ред 7.

Граф на извикванията за тази функция:



## file\_manager.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "file\_manager.h"

00005

00006

00007 bool saveToFile(Contestants contestant[], string filePath) {

00008 // Open file for writing (out) in binary mode (binary)

00009 ofstream file(filePath, ios::out | ios::binary);

00010

00011 if (!file) {

00012 cout << "Error opening file! " << filePath << endl;

00013 return false;

00014 }

00015

00016 for (int i = 0; i < MAXCONTESTANTS; i++) {

00017 if (contestant[i].isObjectUsed) {

00018

00019 file.write((char\*)&contestant[i], sizeof(Contestants));

00020 }

00021 }

00022 file.close();

00023 return true;

00024 }

00025

00026 bool loadFromFile(Contestants contestant[], string filePath) {

00027 ifstream file(filePath, ios::in | ios::binary);

00028

00029 if (!file)

00030 {

00031 cout << "Can't load! No such file: " << filePath << endl;

00032 return false;

00033 }

00034 resetContestants(contestant);

00035

00036 Contestants temp;

00037 int index = 0;

00038

00039 while (file.read((char\*)&temp, sizeof(Contestants))) {

00040 if (index < MAXCONTESTANTS) {

00041 contestant[index] = temp;

00042 index++;

00043 }

00044 }

00045 file.close();

00046 return true;

00047 }

00048

00049 bool removeFileData(string filePath){

00050 ofstream file(filePath, ios::out | ios::trunc | ios::binary);

00051

00052 if(!file){

00053 cout << "Couldn't delete file data of: " << filePath << endl;

00054 return false;

00055 }

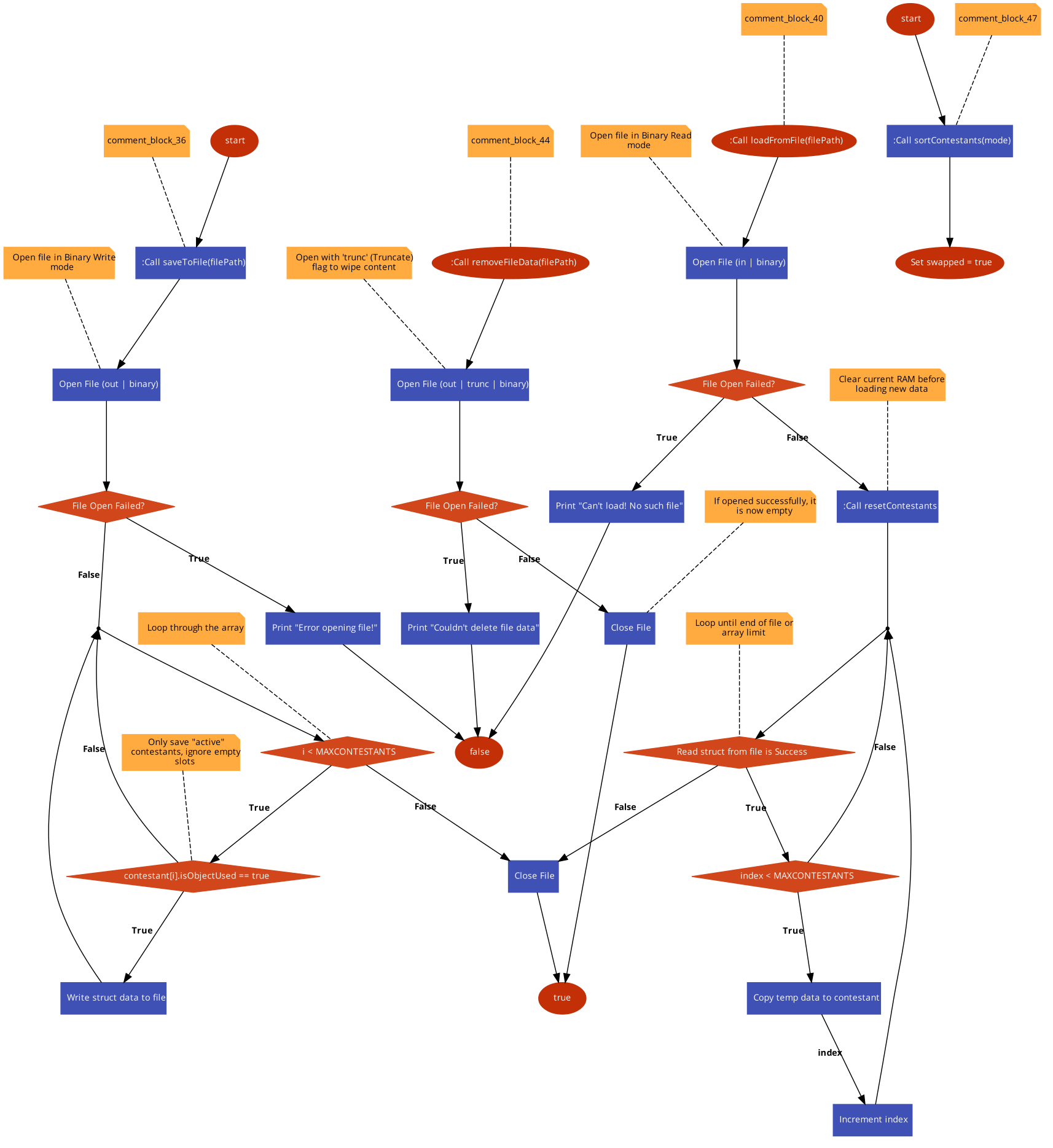
00056

00057 file.close();

00058 return true;

00059 }

00060



## file\_manager.h

00001 #pragma once

00002 #include <iostream>

00003 #include <fstream>

00004 #include <cstring>

00005 #include <iomanip>

00006

00007 using namespace std;

00008

00009 #include "../../logic/contestants\_manipulation/contestants\_manager.h"

00010 #include "../../utils/settings.h"

00011 #include "../../utils/contestant.h"

00012

00013

00017 bool saveToFile(Contestants contestant[], string filePath);

00018

00022 bool loadFromFile(Contestants contestant[], string filePath);

00023

00027 bool removeFileData(string filePath);

00028

00029

## BeautyContest/utils/settings.h Файл Справка

Global configuration constants and settings flags for the application.

### Променливи

* constexpr unsigned int MAXCONTESTANTS = 30

*The max contestants user can manipulate.*

* constexpr unsigned int MAXNAMECHARS = 100

*Limit how long can be a name.*

* constexpr unsigned int TITLEWIDTH = 65

*How long the titles are.*

* bool DEBUGMODE = false

*Switch for Debuging.*

* bool CLEARCONSOLE = true

*Switch for clear console feature.*

* bool AUTOSAVE = true

*Switch for autosave feature.*

* bool AUTOCATEGORIZE = false

*Switch for auto categorisation after adding new contestant.*

### Подробно описание

Global configuration constants and settings flags for the application.

Виж дефиницията във файла settings.h.

### Променливи Документация

#### bool AUTOCATEGORIZE = false[inline]

Switch for auto categorisation after adding new contestant.

Виж дефиницията във файла settings.h ред 26.

#### bool AUTOSAVE = true[inline]

Switch for autosave feature.

Виж дефиницията във файла settings.h ред 23.

#### bool CLEARCONSOLE = true[inline]

Switch for clear console feature.

Виж дефиницията във файла settings.h ред 20.

#### bool DEBUGMODE = false[inline]

Switch for Debuging.

Виж дефиницията във файла settings.h ред 17.

#### unsigned int MAXCONTESTANTS = 30[constexpr]

The max contestants user can manipulate.

Виж дефиницията във файла settings.h ред 7.

#### unsigned int MAXNAMECHARS = 100[constexpr]

Limit how long can be a name.

Виж дефиницията във файла settings.h ред 10.

#### unsigned int TITLEWIDTH = 65[constexpr]

How long the titles are.

Виж дефиницията във файла settings.h ред 13.

## settings.h

Вижте документацията за този файл.

00001

00003

00004 #pragma once

00005

00007 constexpr unsigned int MAXCONTESTANTS = 30;

00008

00010 constexpr unsigned int MAXNAMECHARS = 100;

00011

00013 constexpr unsigned int TITLEWIDTH = 65;

00014

00015

00017 inline bool DEBUGMODE = false;

00018

00020 inline bool CLEARCONSOLE = true;

00021

00023 inline bool AUTOSAVE = true;

00024

00026 inline bool AUTOCATEGORIZE = false;

## BeautyContest/utils/utils.cpp Файл Справка

General utility functions for string manipulation and array comparison.

#include "utils.h"

### Функции

* int getRealSizeOfCharArray (char arr[], int sizeArr, char end)

*Number of chars in array before a terminator.*

* int countDifferencesBetweenCharArrays (char first[], char second[], int sizeFirst, int sizeSecond)

*Gives differences between two char array. Has protection against putting two unequal sized arrays.*

* string toLowerString (string s)

*Makes all capital chars of a string into small.*

### Подробно описание

General utility functions for string manipulation and array comparison.

Виж дефиницията във файла utils.cpp.

### Функции Документация

#### int countDifferencesBetweenCharArrays (char first[], char second[], int sizeFirst, int sizeSecond)

Gives differences between two char array. Has protection against putting two unequal sized arrays.

##### Аргументи

|  |  |
| --- | --- |
| *first* | The first array |
| *second* | The second array |
| *sizeFirst* | The size of the first array |
| *sizeSecond* | The size of the second array |

##### Връща

Count of differences between two equal size arrays

Виж дефиницията във файла utils.cpp ред 18.

Граф с извикванията за тази функция:



Граф на извикванията за тази функция:



#### int getRealSizeOfCharArray (char arr[], int sizeArr, char end = '\0')

Number of chars in array before a terminator.

##### Аргументи

|  |  |
| --- | --- |
| *arr* | char array you what to search in |
| *sizeArr* | size fo set array |
| *end* | where to stop counting |

##### Връща

real size of array

Виж дефиницията във файла utils.cpp ред 7.

Граф на извикванията за тази функция:



#### string toLowerString (string s)

Makes all capital chars of a string into small.

##### Аргументи

|  |  |
| --- | --- |
| *s* | String you want to lower |

##### Връща

The lowered string

Виж дефиницията във файла utils.cpp ред 48.

## utils.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "utils.h"

00005

00006

00007 int getRealSizeOfCharArray(char arr[], int sizeArr, char end){

00008 int realSize = 0;

00009 bool endOfArray = 0;

00010 for(int i = 0; i < sizeArr - 1 && !endOfArray; i++){

00011 realSize++;

00012 endOfArray = arr[i + 1] == end;

00013 }

00014 return realSize;

00015 }

00016

00017

00018 int countDifferencesBetweenCharArrays(char first[], char second[], int sizeFirst, int sizeSecond){

00019

00020 int realSizeFirst = getRealSizeOfCharArray(first, sizeFirst);

00021 int realSizeSecond = getRealSizeOfCharArray(second, sizeSecond);

00022

00023 int lastToCheck = max(realSizeFirst, realSizeSecond);

00024 int minSize = min(sizeFirst, sizeSecond);

00025 if(lastToCheck > minSize){

00026 lastToCheck = minSize;

00027 deBugInfo("WARNING - countDifferencesBetweenCharArrays: Size of array is too small!" << endl);

00028 }

00029

00030 deBugInfo("SYSTEM: lastToCheck: " << lastToCheck << endl);

00031

00032 int differences = 0;

00033

00034 for(int i = 0; i < lastToCheck ; i++ ){

00035 bool different = tolower(first[i]) != tolower(second[i]);

00036 if(different) differences++;

00037 deBugInfo("F: " << first[i] << " S: " << second[i] << " D? " << different << endl);

00038 //deBugInfo("SYSTEM: Comparing " << tolower(first[i]) << " and " << tolower(second[i]) << " | " << "nowDiff: " << differences);

00039 }

00040 deBugInfo("Total differences: " << differences <<" | Compared: " << first << " and " << second << endl << endl);

00041 return differences;

00042 }

00043

00044

00048 string toLowerString(string s){

00049 string newS = "";

00050 int sizeOfString = s.length();

00051

00052 for(int i = 0; i < sizeOfString; i++){

00053 char letter = s[i];

00054

00055 if( letter >= 'A' && letter <= 'Z'){

00056 newS.append(1,letter + 32);

00057

00058 } else {

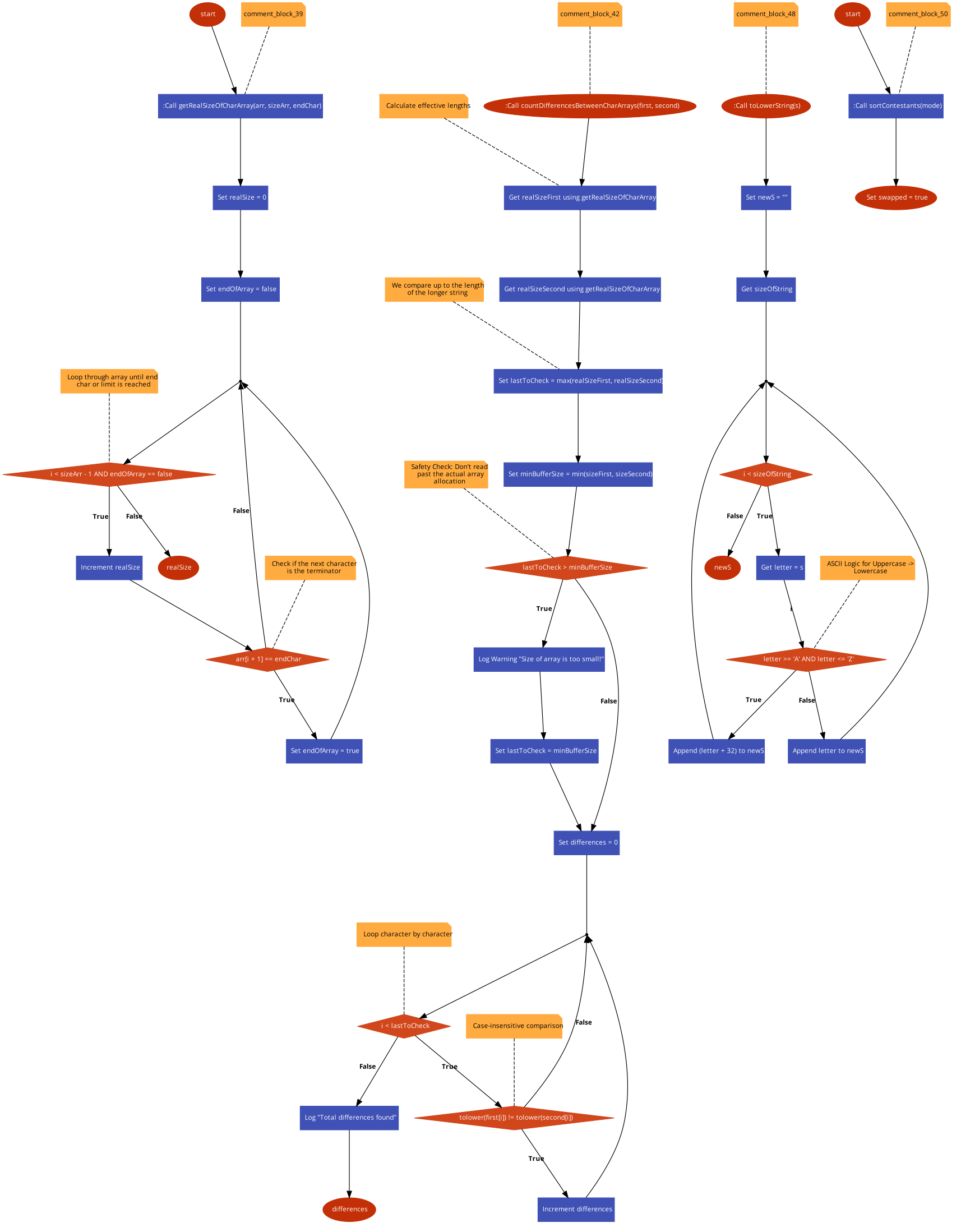
00059 newS.append(1,letter);

00060 }

00061 }

00062 return newS;

00063 }



## utils.h

00001 #pragma once

00002 #include <string>

00003

00004 #include "debug.h"

00005

00006 using namespace std;

00007

00008

00014 int getRealSizeOfCharArray(char arr[], int sizeArr, char end = '\0');

00015

00023 int countDifferencesBetweenCharArrays(char first[], char second[], int sizeFirst, int sizeSecond);

00024

00025

## BeautyContest/utils/visual\_enhancement.cpp Файл Справка

Console visual helpers, such as clearing the screen.

#include "visual\_enhancement.h"

### Функции

* void clearConsole (bool clear)

*Clears console.*

### Подробно описание

Console visual helpers, such as clearing the screen.

Виж дефиницията във файла visual\_enhancement.cpp.

### Функции Документация

#### void clearConsole (bool clear = true)

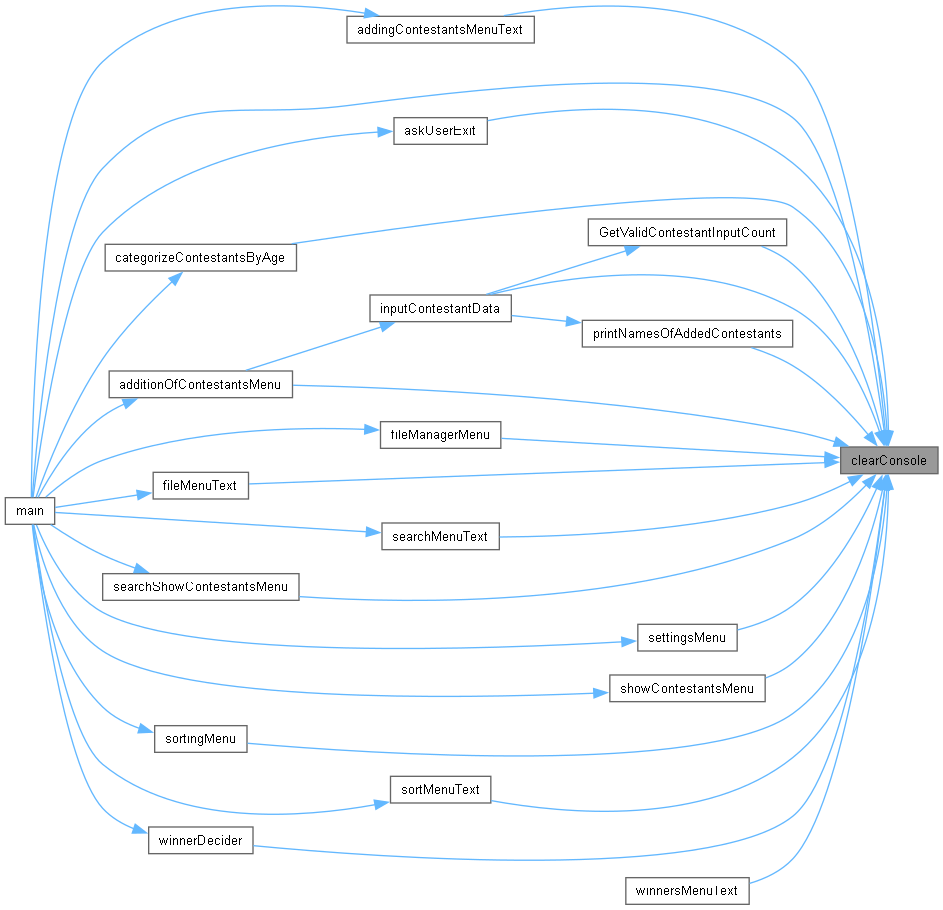
Clears console.

##### Аргументи

|  |  |
| --- | --- |
| *clear* | Whether to clear console or not |

Виж дефиницията във файла visual\_enhancement.cpp ред 6.

Граф на извикванията за тази функция:



## visual\_enhancement.cpp

Вижте документацията за този файл.

00001

00003

00004 #include "visual\_enhancement.h"

00005

00006 void clearConsole(bool clear) { if (clear) system("cls"); }

## visual\_enhancement.h

00001 #pragma once

00002 #include <cstdlib>

00003

00004 using namespace std;

00005

00006

00009 void clearConsole(bool clear = true);

# Азбучен указател

INDEX