ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Московский институт электроники и математики им. А. Н. Тихонова

Отчет по Лабороторной работе №6 по предмету «Языки Программирования»

Подготовил: студент группы СКБ221
Нугманов М. И.

Москва 2022

Иерархический список классов

Иерархия классов

Иерархия классов.

Алфавитный указатель классов

Классы

Классы с их кратким описанием.

Список файлов

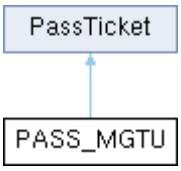
Файлы

Полный список файлов.

Классы

Класс PASS_MGTU

Производный класс, в котором реализовано генерирование билета согласно своим требованием
#include <classes.h>
Граф наследования:PASS_MGTU:



Открытые члены

- **PASS_MGTU** (int **sex**, int **year**, int **month**, int **day**)
- std::string **pass_generate** () override final
  *Метод класса **PASS_MGTU** для генерации билета*

Дополнительные унаследованные члены

Подробное описание

Производный класс, в котором реализовано генерирование билета согласно своим требованием

См. определение в файле **classes.h** строка **37**

Конструктор(ы)

PASS_MGTU::PASS_MGTU (int *sex*, int *year*, int *month*, int *day*)[inline]

См. определение в файле **classes.h** строка **39**

Методы

std::string PASS_MGTU::pass_generate ()[final], [override], [virtual]

Метод класса **PASS_MGTU** для генерации билета

Возвращает

passticket Сгенерированный билет

Замещает **PassTicket** (*стр.7*).

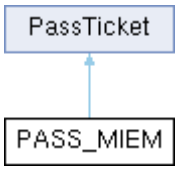
См. определение в файле **classes.cpp** строка **47**

Объявления и описания членов классов находятся в файлах:

- **classes.h**
- **classes.cpp**

Класс PASS_MIEM

Производный класс, в котором реализовано генерирование билета согласно своим требованием
#include <classes.h>
Граф наследования:PASS_MIEM:



Открытые члены

- **PASS_MIEM** (int **sex**, int **year**, int **month**, int **day**)
- std::string **pass_generate** () override final
  *Метод класса **PASS_MIEM** для генерации билета*

Дополнительные унаследованные члены

Подробное описание

Производный класс, в котором реализовано генерирование билета согласно своим требованием

См. определение в файле **classes.h** строка **24**

Конструктор(ы)

PASS_MIEM::PASS_MIEM (int *sex*, int *year*, int *month*, int *day*)[inline]

См. определение в файле **classes.h** строка **26**

Методы

std::string PASS_MIEM::pass_generate ()[final], [override], [virtual]

Метод класса **PASS_MIEM** для генерации билета

Возвращает

passticket Сгенерированный билет

Замещает **PassTicket** (*стр.7*).

См. определение в файле **classes.cpp** строка **19**

Объявления и описания членов классов находятся в файлах:

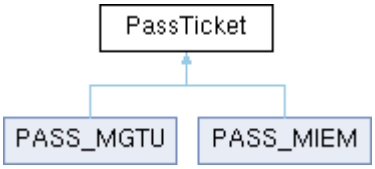
- **classes.h**
- **classes.cpp**

Класс PassTicket

Абстрактные класс, заключающий в себе функционал генерирования билета, который реализован в производных классах

#include <classes.h>

Граф наследования:PassTicket:



Открытые члены

- virtual std::string **pass_generate** ()=0

Защищенные данные

- int **sex**
- int **year**
- int **month**
- int **day**
- std::string **passticket** = ""

Подробное описание

Абстрактные класс, заключающий в себе функционал генерирования билета, который реализован в производных классах

См. определение в файле **classes.h** строка **11**

Методы

virtual std::string PassTicket::pass_generate ()[pure virtual]

Замещается в **PASS_MIEM** (*стр.6*) и **PASS_MGTU** (*стр.5*).

Данные класса

int PassTicket::day[protected]

См. определение в файле **classes.h** строка **16**

int PassTicket::month[protected]

См. определение в файле **classes.h** строка **15**

std::string PassTicket::passticket = ""[protected]

См. определение в файле **classes.h** строка **17**

int PassTicket::sex[protected]

См. определение в файле **classes.h** строка **13**

int PassTicket::year[protected]

См. определение в файле **classes.h** строка **14**

Объявления и описания членов класса находятся в файле:

- **classes.h**

Файлы

Файл classes.cpp

Файл с описанием основных функций, используемых для реализации программы

```
#include <iostream>
#include <fstream>
#include <string>
#include <random>
#include <chrono>
#include <set>
#include "classes.h"
```

Функции

- std::string **GetPassTicket** (**PassTicket** &University)
  Фунция, обобщающая работу с объектами классов, производных от абстрактного

- bool **IsDigit** (char symbol)
- bool **IsUniversityName** ()
  Функция, проверяющая корректность ввода название университета

- int **GetNumberOfTickets** ()
  Функция, возвращающая количество билетов, которые необходимо сгенерировать

- int **GetSex** ()
  Функция, возвращающая пол владельца генерируемого билета

- int **GetYear** ()
  Функция, возвращающая год рождения владельца генерируемого билета

- int **GetMonth** ()
  Функция, возвращающая месяц рождения владельца генерируемого билета

- int **GetDay** ()
  Функция, возвращающая день рождения владельца генерируемого билета

- void **FillFieldsConsole** ()
  Функция ,заполняющая переменные с консоли, которые будут использованы для полей классов

- void **FillFieldsFile** (std::ifstream &FromFile)
  Функция, заполняющая переменные с файла, которые будут использованы для полей классов

- void **CheckDate** ()
  Функция, проверяющая корректность введеной даты

- void **CompletionToConsole** (std::string group[], int num_tickets)
  Функция, выписывающая результат на консоль

- void **CompletionToFile** (std::string group[], int num_tickets, const char *nametofile)
  Функция, выписывающая результаты в файл

- void **GetResultFromConsole** ()
  Функция, принимающая входные данные с консоли, и выписывающая результаты в консоль

- void **GetResultFromConsole** (const char *nametofile, bool is_name_file)
  Функция, принимающая входные данные с консоли, и выписывающая результаты в файл

- void **GetResultFromFile** (const char *namefromfile)
  Функция, принимающая входные даннные с файла, и выписывающая результаты на консоль

- void **GetResultFromFile** (const char *namefromfile, const char *nametofile, bool is_name_file)
  Функция, принимающая входные данные с файла, и выписывающая результаты в файл

- void **Help** ()
  Функция, выводящая инструкцию по использованию программы

Переменные

- std::string **stream**
  Константные переменные, служащие для заполнения полей классов

- std::string **name_univ**
- std::string **s_sex**
- std::string **s_year**
- std::string **s_month**
- std::string **s_day**
- int **sex**
- int **year**
- int **month**
- int **day**

Подробное описание

Файл с описанием основных функций, используемых для реализации программы

См. определение в файле **classes.cpp**

Функции

void CheckDate ()

Функция, проверяющая корректность введеной даты

См. определение в файле **classes.cpp** строка **254**

void CompletionToConsole (std::string *group*[], int *num_tickets*)

Функция, выписывающая результат на консоль

См. определение в файле **classes.cpp** строка **266**

void CompletionToFile (std::string *group*[], int *num_tickets*, const char * *nametofile*)

Функция, выписывающая результаты в файл

См. определение в файле **classes.cpp** строка **275**

void FillFieldsConsole ()

Функция ,заполняющая переменные с консоли, которые будут использованы для полей классов

См. определение в файле **classes.cpp** строка **226**

void FillFieldsFile (std::ifstream & *FromFile*)

Функция, заполняющая переменные с файла, которые будут использованы для полей классов

См. определение в файле **classes.cpp** строка **240**

int GetDay ()

Функция, возвращающая день рождения владельца генерируемого билета

См. определение в файле **classes.cpp** строка **198**

int GetMonth ()

Функция, возвращающая месяц рождения владельца генерируемого билета

См. определение в файле **classes.cpp** строка **170**

int GetNumberOfTickets ()

Функция, возвращающая количество билетов, которые необходимо сгенерировать

См. определение в файле **classes.cpp** строка **92**

std::string GetPassTicket (PassTicket & *University*)

Фунция, обобщающая работу с объектами классов, производных от абстрактного

Аргументы

University	Ссылка на объект класса, производного от абстрактного

Возвращает

University.pass_generate() Сгенерированный билет

См. определение в файле **classes.cpp** строка **77**

void GetResultFromConsole ()

Функция, принимающая входные данные с консоли, и выписывающая результаты в консоль

См. определение в файле **classes.cpp** строка **285**

void GetResultFromConsole (const char * *nametofile*, bool *is_name_file*)

Функция, принимающая входные данные с консоли, и выписывающая результаты в файл

См. определение в файле **classes.cpp** строка **318**

void GetResultFromFile (const char * *namefromfile*)

Функция, принимающая входные данные с файла, и выписывающая результаты на консоль

См. определение в файле **classes.cpp** строка **368**

void GetResultFromFile (const char * *namefromfile*, const char * *nametofile*, bool *is_name_file*)

Функция, принимающая входные данные с файла, и выписывающая результаты в файл

См. определение в файле **classes.cpp** строка **404**

int GetSex ()

Функция, возвращающая пол владельца генерируемого билета

См. определение в файле **classes.cpp** строка **121**

int GetYear ()

Функция, возвращающая год рождения владельца генерируемого билета

См. определение в файле **classes.cpp** строка **142**

void Help ()

Функция, выводящая инструкцию по использованию программы

См. определение в файле **classes.cpp** строка **480**

bool IsDigit (char *symbol*)

См. определение в файле **classes.cpp** строка **78**

bool IsUniversityName ()

Функция, проверяющая корректность ввода название университета

См. определение в файле **classes.cpp** строка **85**

Переменные

int day

См. определение в файле **classes.cpp** строка **13**

int month

См. определение в файле **classes.cpp** строка **13**

std::string name_univ

См. определение в файле **classes.cpp** строка **12**

std::string s_day

См. определение в файле **classes.cpp** строка **12**

std::string s_month

См. определение в файле **classes.cpp** строка **12**

std::string s_sex

См. определение в файле **classes.cpp** строка **12**

std::string s_year

См. определение в файле **classes.cpp** строка **12**

int sex

См. определение в файле **classes.cpp** строка **13**

std::string stream

Константные переменные, служащие для заполнения полей классов

См. определение в файле **classes.cpp** строка **12**

int year

См. определение в файле **classes.cpp** строка **13**

classes.cpp

См. документацию.00001

```
00004 #include <iostream>
00005 #include <fstream>
00006 #include <string>
00007 #include <random>
00008 #include <chrono>
00009 #include <set>
00010 #include "classes.h"
00012 std::string stream, name_univ, s_sex, s_year, s_month, s_day;
00013 int sex, year, month, day;
00014
00019 std::string PASS_MIEM::pass_generate() {
00020     int date = year * 10000 + month * 100 + day;
00021     passticket += std::to_string(sex) + std::to_string(date);
00022     std::mt19937 randomizer(date + std::chrono::steady_clock::now().time_since_epoch().count());
00023     std::uniform_int_distribution<int> NNNNN(10000, 99999);
00024     passticket += std::to_string(NNNNN(randomizer));
00025     int sum = 0;
00026     for(int i = 0; i < passticket.size(); i++) {
00027         sum += (passticket[i] - '0') * (i + 1);
00028     }
00029     while(sum % 11 == 4) {
00030         sum -= (passticket[9] - '0') * 10;
00031         int new_digit = (passticket[9] - '0' + 1) % 10;
00032         passticket[9] = new_digit + '0';
00033         sum += new_digit * 10;
00034     }
00035     for(int i = 0; i < 10; i++) {
00036         if((sum + i * 15) % 11 == 0) {
00037             passticket += std::to_string(i);
00038             break;
00039         }
00040     }
00041     return passticket;
00042 }
00047 std::string PASS_MGTU::pass_generate() {
00048     int date = year * 10000 + month * 100 + day;
00049     passticket += std::to_string(sex) + std::to_string(date);
00050     std::mt19937 randomizer(date + std::chrono::steady_clock::now().time_since_epoch().count());
00051     std::uniform_int_distribution<int> NNNN(1000, 9999);
00052     passticket += std::to_string(NNNN(randomizer));
00053     int sum = 0;
00054     for(int i = 0; i < passticket.size(); i++) {
00055         sum += (passticket[i] - '0') * (i + 1);
00056     }
00057     while((sum % 10) % 2 != 0) {
00058         sum -= (passticket[10] - '0') * 11;
00059         int new_digit = (passticket[10] - '0' + 1) % 10;
00060         passticket[10] = new_digit + '0';
00061         sum += new_digit * 11;
00062     }
00063
00064     for(int i = 0; i < 10; i++) {
00065         if((sum + i * 14) % 10 == 0) {
00066             passticket += std::to_string(i);
00067             break;
00068         }
00069     }
00070     return passticket;
00071 }
00077 std::string GetPassTicket(PassTicket &University) { return University.pass_generate(); }
00078 bool IsDigit(char symbol) {
00079     if(symbol >= '0' && symbol <= '9') return true;
00080     return false;
00081 }
00085 bool IsUniversityName() {
00086     if(name_univ == "MIEM" || name_univ == "MGTU") return true;
00087     return false;
00088 }
00092 int GetNumberOfTickets() {
00093     int result = 0;
00094     if(stream.size()) {
00095         for(int i = 0; i < stream.size(); i++) {
00096             if(!IsDigit(stream[i])) {
00097                 std::cerr << "ERROR: invalid symbols in natural number of pass tickets input!" << std::endl;
00098                 exit(1);
00099             }
00100             else {
00101                 result = result * 10 + (stream[i] - '0');
00102             }
00103         }
00104     }
00105     else {
00106         std::cerr << "ERROR: number of pass tickets not entered!" << std::endl;
00107         exit(1);
00108     }
00109     if(result) {
00110         stream = "";
00111         return result;
00112     }
00113     else {
00114         std::cout << "ERROR: You must enter a natural number of pass tickets!" << std::endl;
00115         exit(1);
00116     }
00117 }
00121 int GetSex() {
```

```
00122       if(s_sex.size()) {
00123           if(s_sex == "man") {
00124               return 1;
00125           }
00126           else if(s_sex == "woman") {
00127               return 0;
00128           }
00129           else {
00130               std::cerr << "ERROR: invalid symbols in sex!" << std::endl;
00131               exit(1);
00132           }
00133       }
00134       else {
00135           std::cerr << "ERROR: sex not entered!" << std::endl;
00136           exit(1);
00137       }
00138 }
00142 int GetYear() {
00143       int result = 0;
00144       if(s_year.size()) {
00145           for(int i = 0; i < s_year.size(); i++) {
00146               if(!IsDigit(s_year[i])) {
00147                   std::cerr << "ERROR: invalid symbols in year of birth!" << std::endl;
00148                   exit(1);
00149               }
00150               else {
00151                   result = result * 10 + (s_year[i] - '0');
00152               }
00153           }
00154       }
00155       else {
00156           std::cerr << "ERROR: year of birth not entered!" << std::endl;
00157           exit(1);
00158       }
00159       if(result >= 1900 && result <= 2005) {
00160           return result;
00161       }
00162       else {
00163           std::cerr << "ERROR: year of birth must be in range 1900 to 2005" << std::endl;
00164           exit(1);
00165       }
00166 }
00170 int GetMonth() {
00171       int result = 0;
00172       if(s_month.size()) {
00173           for(int i = 0; i < s_month.size(); i++) {
00174               if(!IsDigit(s_month[i])) {
00175                   std::cerr << "ERROR: invalid symbols in month of birth!" << std::endl;
00176                   exit(1);
00177               }
00178               else {
00179                   result = result * 10 + (s_month[i] - '0');
00180               }
00181           }
00182       }
00183       else {
00184           std::cerr << "ERROR: month of birth not entered!" << std::endl;
00185           exit(1);
00186       }
00187       if(result >= 1 && result <= 12) {
00188           return result;
00189       }
00190       else {
00191           std::cerr << "ERROR: month of birth must be in range 1 to 12!" << std::endl;
00192           exit(1);
00193       }
00194 }
00198 int GetDay() {
00199       int result = 0;
00200       if(s_day.size()) {
00201           for(int i = 0; i < s_day.size(); i++) {
00202               if(!IsDigit(s_day[i])) {
00203                   std::cout << "ERROR: invalid symbols in day of birth!" << std::endl;
00204                   exit(1);
00205               }
00206               else {
00207                   result = result * 10 + (s_day[i] - '0');
00208               }
00209           }
00210       }
00211       else {
00212           std::cerr << "ERROR: day of birth not entered!" << std::endl;
00213           exit(1);
00214       }
00215       if(result >= 1 && result <= 31) {
00216           return result;
00217       }
00218       else {
00219           std::cerr << "ERROR: day of birth must be in range 1 to 31" << std::endl;
00220           exit(1);
00221       }
00222 }
00226 void FillFieldsConsole() {
00227       std::cin >> name_univ >> s_sex >> s_year >> s_month >> s_day;
00228       if(!IsUniversityName()) {
00229           std::cerr << "ERROR: wrong University name!" << std::endl;
00230           exit(1);
00231       }
00232       sex = GetSex();
00233       year = GetYear();
00234       month = GetMonth();
```

```
00235       day = GetDay();
00236 }
00240 void FillFieldsFile(std::ifstream &FromFile) {
00241       FromFile >> name_univ >> s_sex >> s_year >> s_month >> s_day;
00242       if(!IsUniversityName()) {
00243             std::cerr << "ERROR: wrong University name!" << std::endl;
00244             exit(1);
00245       }
00246       sex = GetSex();
00247       year = GetYear();
00248       month = GetMonth();
00249       day = GetDay();
00250 }
00254 void CheckDate() {
00255       int calendar[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
00256       if(!(year % 4))
00257             calendar[1] = 29;
00258       if(day > calendar[month - 1] || day < 1) {
00259             std::cerr << "ERROR: day in this month and year must be in range 1 to " << calendar[month - 1] << std::endl;
00260             exit(1);
00261       }
00262 }
00266 void CompletionToConsole(std::string group[], int num_tickets) {
00267       std::cout << "\nRESULT:\n" << std::endl;
00268       for(int i = 0; i < num_tickets; i++) {
00269             std::cout << i + 1 << ") " << group[i] << std::endl;
00270       }
00271 }
00275 void CompletionToFile(std::string group[], int num_tickets, const char * nametofile) {
00276       std::ofstream ToFile(nametofile);
00277       ToFile << "RESULT:\n" << std::endl;
00278       for(int i = 0; i < num_tickets; i++) {
00279             ToFile << i + 1 << ") " << group[i] << std::endl;
00280       }
00281 }
00285 void GetResultFromConsole() {
00286       std::cout << "Enter the number of pass tickets: ";
00287       std::cin >> stream;
00288       std::cout << "\n";
00289       int num_tickets = GetNumberOfTickets();
00290       std::string group[num_tickets];
00291       std::set <std::string> unique;
00292       for(int i = 0; i < num_tickets; i++) {
00293             std::cout << i+1 << ") ";
00294             FillFieldsConsole();
00295             CheckDate();
00296             if(name_univ == "MIEM") {
00297                   PASS_MIEM person(sex, year, month, day);
00298                   std::string temp = GetPassTicket(person);
00299                   while(unique.find(temp) != unique.end())
00300                         temp = GetPassTicket(person);
00301                   group[i] = temp;
00302                   unique.insert(temp);
00303             }
00304             else if(name_univ == "MGTU") {
00305                   PASS_MGTU person(sex, year, month, day);
00306                   std::string temp = GetPassTicket(person);
00307                   while(unique.find(temp) != unique.end())
00308                         temp = GetPassTicket(person);
00309                   group[i] = temp;
00310                   unique.insert(temp);
00311             }
00312       }
00313       CompletionToConsole(group, num_tickets);
00314 }
00318 void GetResultFromConsole(const char * nametofile, bool is_name_file) {
00319       std::cout << "Enter the number of pass tickets: ";
00320       std::cin >> stream;
00321       std::cout << "\n";
00322       int num_tickets = GetNumberOfTickets();
00323       std::string group[num_tickets];
00324       std::set <std::string> unique;
00325       for(int i = 0; i < num_tickets; i++) {
00326             std::cout << i+1 << ") ";
00327             FillFieldsConsole();
00328             CheckDate();
00329             if(name_univ == "MIEM") {
00330                   PASS_MIEM person(sex, year, month, day);
00331                   std::string temp = GetPassTicket(person);
00332                   while(unique.find(temp) != unique.end())
00333                         temp = GetPassTicket(person);
00334                   group[i] = temp;
00335                   unique.insert(temp);
00336             }
00337             else if(name_univ == "MGTU") {
00338                   PASS_MGTU person(sex, year, month, day);
00339                   std::string temp = GetPassTicket(person);
00340                   while(unique.find(temp) != unique.end())
00341                         temp = GetPassTicket(person);
00342                   group[i] = temp;
00343                   unique.insert(temp);
00344             }
00345       }
00346
00347       if(is_name_file) {
00348             std::ifstream check(nametofile);
00349             if(check.is_open()) {
00350                   check.close();
00351                   CompletionToFile(group, num_tickets, nametofile);
00352                   std::cout << "\nCOMMAND COMPLETED, CHECK THE FILE " << nametofile << std::endl;
00353             }
```

```
00354           else {
00355               check.close();
00356               std::cerr << "ERROR: output file nor found, You must create it or enter right!" << std::endl;
00357               exit(1);
00358           }
00359       }
00360       else {
00361           CompletionToFile(group, num_tickets, nametofile);
00362           std::cout << "\nCOMMAND COMPLETED, CHECK THE FILE " << nametofile << std::endl;
00363       }
00364 }
00368 void GetResultFromFile(const char * namefromfile) {
00369       std::ifstream FromFile(namefromfile);
00370       if(FromFile.is_open()) {
00371           int counter = 0;
00372           std::set <std::string> unique;
00373           while(!FromFile.eof()) {
00374               counter++;
00375               FillFieldsFile(FromFile);
00376               CheckDate();
00377               if(name_univ   == "MIEM") {
00378                   PASS_MIEM person(sex, year, month, day);
00379                   std::string temp = GetPassTicket(person);
00380                   while(unique.find(temp) != unique.end())
00381                       temp = GetPassTicket(person);
00382                   unique.insert(temp);
00383                   std::cout << counter << ") " << temp << std::endl;
00384               }
00385               else if(name_univ == "MGTU") {
00386                   PASS_MGTU person(sex, year, month, day);
00387                   std::string temp = GetPassTicket(person);
00388                   while(unique.find(temp) != unique.end())
00389                       temp = GetPassTicket(person);
00390                   unique.insert(temp);
00391                   std::cout << counter << ") " << temp << std::endl;
00392               }
00393           }
00394       }
00395       else {
00396           FromFile.close();
00397           std::cerr << "ERROR: input file not found, You must create it or enter right!" << std::endl;
00398           exit(1);
00399       }
00400 }
00404 void GetResultFromFile(const char * namefromfile, const char * nametofile, bool is_name_file) {
00405       std::ifstream FromFile(namefromfile);
00406       if(FromFile.is_open()) {
00407           if(is_name_file) {
00408               std::ifstream check(nametofile);
00409               if(check.is_open()) {
00410                   check.close();
00411                   std::ofstream ToFile(nametofile);
00412                   int counter = 0;
00413                   std::set <std::string> unique;
00414                   while(!FromFile.eof()) {
00415                       FillFieldsFile(FromFile);
00416                       CheckDate();
00417                       if(name_univ == "MIEM") {
00418                           PASS_MIEM person(sex, year, month, day);
00419                           std::string temp = GetPassTicket(person);
00420                           while(unique.find(temp) != unique.end())
00421                               temp = GetPassTicket(person);
00422                           unique.insert(temp);
00423                           ToFile << ++counter << ") " << temp << std::endl;
00424                       }
00425                       else if(name_univ == "MGTU") {
00426                           PASS_MGTU person(sex, year, month, day);
00427                           std::string temp = GetPassTicket(person);
00428                           while(unique.find(temp) != unique.end())
00429                               temp = GetPassTicket(person);
00430                           unique.insert(temp);
00431                           ToFile << ++counter << ") " << temp << std::endl;
00432                       }
00433                   }
00434               }
00435               else {
00436                   check.close();
00437                   FromFile.close();
00438                   std::cerr << "ERROR: output file not found, You must create it or enter right!" << std::endl;
00439                   exit(1);
00440               }
00441           }
00442           else {
00443               std::ofstream ToFile(nametofile);
00444               int counter = 0;
00445               std::set <std::string> unique;
00446               while(!FromFile.eof()) {
00447                   FillFieldsFile(FromFile);
00448                   CheckDate();
00449                   if(name_univ == "MIEM") {
00450                       PASS_MIEM person(sex, year, month, day);
00451                       std::string temp = GetPassTicket(person);
00452                       while(unique.find(temp) != unique.end())
00453                           temp = GetPassTicket(person);
00454                       unique.insert(temp);
00455                       ToFile << ++counter << ") " << temp << std::endl;
00456                   }
00457                   else if(name_univ == "MGTU") {
00458                       PASS_MGTU person(sex, year, month, day);
00459                       std::string temp = GetPassTicket(person);
00460                       while(unique.find(temp) != unique.end())
```

```
                temp = GetPassTicket(person);
            unique.insert(temp);
            ToFile << ++counter << ") " << temp << std::endl;
        }
      }
      ToFile.close();
      FromFile.close();
      std::cout << "COMMAND COMPLETED, CHECK THE FILE " << nametofile << std::endl;
    }
  }
  else {
    FromFile.close();
    std::cerr << "ERROR: input file not found, You must create it or enter right!" << std::endl;
    exit(1);
  }
}
void Help() {
  std::cout << "\n";
  std::cout << "###########################################################_INSTRUCTION_##############################################################" << std::endl;
  std::cout << "You must enter:   [./<program_name>] [*] "                                                        << std::endl;
  std::cout << "------------------------------------------------------------------------------------------------------------------------------------" << std::endl;
  std::cout << "Instead of the [*], You must enter the following options:"                                       << std::endl;
  std::cout << "   1) --help .......................................(output of instructions for working with the program)"          << std::endl;
  std::cout << "   2) --default ....................................(console input and console output)"                            << std::endl;
  std::cout << "   3) --tofile .....................................(console input and default file 'DefaultFile.txt' output)"     << std::endl;
  std::cout << "   4) --tofile <filename> ..........................(console input and user-selected file output)"                << std::endl;
  std::cout << "   5) --fromfile <filename> ........................(user-selected file input and console output)"                << std::endl;
  std::cout << "   6) --fromfile <filename> --tofile ...............(user-selected file input and default file 'DefaultFile.txt' output)" << std::endl;
  std::cout << "   7) --fromfile <filename1> --tofile <filename2> ...(user-selected file input and user-selected file output, neccesary different files!)" << std::endl;
  std::cout << "####################################################################################################################################" << std::endl;
  std::cout << "\n";
}
```

Файл classes.h

Заголовочный файл, в которой обьявлены используемые классы и фукнции
#include <string>
#include <list>

Классы

- class **PassTicket**
  Абстрактные класс, заключающий в себе функционал генерирования билета, который реализован в производных классах
- class **PASS_MIEM**
  Производный класс, в котором реализовано генерирование билета согласно своим требованием
- class **PASS_MGTU**
  Производный класс, в котором реализовано генерирование билета согласно своим требованием

Функции

- void **Help** ()
  Функция, выводящая инструкцию по использованию программы

- void **GetResultFromConsole** ()
  Функция, принимающая входные данные с консоли, и выписывающая результаты в консоль

- void **GetResultFromConsole** (const char *nametofile, bool is_name_file)
  Функция, принимающая входные данные с консоли, и выписывающая результаты в файл

- void **GetResultFromFile** (const char *namefromfile)
  Функция, принимающая входные данные с файла, и выписывающая результаты на консоль

- void **GetResultFromFile** (const char *namefromfile, const char *nametofile, bool is_name_file)
  Функция, принимающая входные данные с файла, и выписывающая результаты в файл

Подробное описание

Заголовочный файл, в которой обьявлены используемые классы и фукнции

См. определение в файле **classes.h**

Функции

void GetResultFromConsole ()

Функция, принимающая входные данные с консоли, и выписывающая результаты в консоль

См. определение в файле **classes.cpp** строка **285**

void GetResultFromConsole (const char * ***nametofile*****, bool** ***is_name_file*****)**

Функция, принимающая входные данные с консоли, и выписывающая результаты в файл

См. определение в файле **classes.cpp** строка **318**

void GetResultFromFile (const char * ***namefromfile*****)**

Функция, принимающая входные данные с файла, и выписывающая результаты на консоль

См. определение в файле **classes.cpp** строка **368**

void GetResultFromFile (const char * ***namefromfile*****,** **const char *** ***nametofile*****,** **bool** ***is_name_file*****)**

Функция, принимающая входные данные с файла, и выписывающая результаты в файл

См. определение в файле **classes.cpp** строка **404**

void Help ()

Функция, выводящая инструкцию по использованию программы

См. определение в файле **classes.cpp** строка **480**

classes.h

См. документацию.00001

```
00004 #ifndef CLASSES
00005 #define CLASSES
00006 #include <string>
00007 #include <list>
00011 class PassTicket {
00012 protected:
00013       int sex;
00014       int year;
00015       int month;
00016       int day;
00017       std::string passticket = "";
00018 public:
00019       virtual std::string pass_generate() = 0;
00020 };
00024 class PASS_MIEM : public PassTicket {
00025 public:
00026       PASS_MIEM(int sex, int year, int month, int day) {
00027             this->sex     = (sex == 1) ? 8 : 4;
00028             this->year    = year;
00029             this->month = month;
00030             this->day     = day;
00031       }
00032       std::string pass_generate() override final;
00033 };
00037 class PASS_MGTU : public PassTicket {
00038 public:
00039       PASS_MGTU(int sex, int year, int month, int day) {
00040             this->sex = sex;
00041             this->year = year;
00042             this->month = month;
00043             this->day = day;
00044       }
00045       std::string pass_generate() override final;
00046 };
00050 void Help();
00054 void GetResultFromConsole();
00058 void GetResultFromConsole(const char * nametofile, bool is_name_file);
00062 void GetResultFromFile(const char * namefromfile);
00066 void GetResultFromFile(const char * namefromfile, const char * nametofile, bool is_name_file);
00067 #endif
```

Файл main.cpp

Главный файл, в котором реализован выбор функций программы пользователем
#include <iostream>
#include <cstring>
#include "classes.h"

Функции

- int **main** (int argc, char **argv)

Подробное описание

Главный файл, в котором реализован выбор функций программы пользователем

См. определение в файле **main.cpp**

Функции

int main (int ***argc*****, char ****** ***argv*****)**

См. определение в файле **main.cpp** строка **8**

main.cpp

См. документацию.

```
00001
00004 #include <iostream>
00005 #include <cstring>
00006 #include "classes.h"
00007
00008 int main(int argc, char ** argv) {
00009     const char * flags[5] = {"--help", "--default", "--tofile", "--fromfile", "DefaultOutput.txt"};
00010
00011     if(argc == 1) {
00012         std::cerr << "ERROR: flag not found!" << std::endl;
00013         std::cerr << "You can enter '--help' for instruction output" << std::endl;
00014         exit(1);
00015     }
00016     else if(argc == 2) {
00017         if(!strcmp(argv[1], flags[0]))
00018             Help();
00019         else if(!strcmp(argv[1], flags[1]))
00020             GetResultFromConsole();
00021         else if(!strcmp(argv[1], flags[2]))
00022             GetResultFromConsole(flags[4], false);
00023         else {
00024             std::cerr << "ERROR: invalid input!" << std::endl;
00025             std::cerr << "You can enter '--help' for instruction output" << std::endl;
00026             exit(1);
00027         }
00028     }
00029     else if(argc == 3) {
00030         if(!strcmp(argv[1],flags[2]))
00031             GetResultFromConsole(argv[2], true);
00032         else if(!strcmp(argv[1], flags[3]))
00033             GetResultFromFile(argv[2]);
00034         else {
00035             std::cerr << "ERROR: invalid input!" << std::endl;
00036             exit(1);
00037         }
00038     }
00039     else if(argc == 4) {
00040         if(!strcmp(argv[1], flags[3]) && !strcmp(argv[3], flags[2])) {
00041             GetResultFromFile(argv[2], flags[4], false);
00042         }
00043         else {
00044             std::cerr << "ERROR: invalid input!" << std::endl;
00045             exit(1);
00046         }
00047     }
00048     else if(argc == 5) {
00049         if(!strcmp(argv[1], flags[3]) && !strcmp(argv[3], flags[2])) {
00050             GetResultFromFile(argv[2], argv[4], true);
00051         }
00052         else {
00053             std::cerr << "ERROR: invalid input!" << std::endl;
00054             exit(1);
00055         }
00056     }
00057     else {
00058         std::cerr << "ERROR: invalid input!" << std::endl;
00059         exit(1);
00060     }
00061
00062
00063
00064
00065
00066     return 0;
00067 }
```

Алфавитный указатель

INDEX