Лабораторные работы №6-7

Создано системой Doxygen 1.9.5

Глава 1

Алфавитный указатель пространств имен

1.1 Пространства имен

Полный список документированных пространств имен.

Глава 2

Иерархический список классов

2.1 Иерархия классов

Иерархия классов.

Глава 3

Алфавитный указатель классов

3.1 Классы

Классы с их кратким описанием.

Глава 4

Список файлов

4.1 Файлы

Полный список документированных файлов.

Глава 5

Пространства имен

5.1 Пространство имен sex

Пространство имен, отвечающее за пол студентна. Используется в функциях для создания студ. билета.

Переменные

- int man = 0
- int woman = 1

5.1.1 Подробное описание

Пространство имен, отвечающее за пол студентна. Используется в функциях для создания студ. билета.

5.1.2 Переменные

5.1.2.1 man

int sex::man = 0

См. определение в файле main.cpp строка 10

5.1.2.2 woman

int sex::woman = 1

См. определение в файле main.cpp строка 11

Глава 6

Классы

6.1 Класс MGTUU

Является подклассом University (стр. 15). Отвечат за генерацию билета для МГТУУ.

#include <classes.h>

Открытые члены

- std::string generate (int, int, int, int) override

  Генерирует номер студ. билета МГТУУ на основе пола и даты рождения студента.

6.1.1 Подробное описание

Является подклассом University (стр. 15). Отвечат за генерацию билета для МГТУУ.

См. определение в файле classes.h строка 32

6.1.2 Конструктор(ы)

6.1.2.1 MGTUU()

MGTUU::MGTUU ( ) [inline]

См. определение в файле classes.h строка 38

6.1.3 Методы

6.1.3.1 generate()

```
std::string MGTUU::generate (
                int sex,
                int y,
                int m,
                int d )   [override], [virtual]
```

Генерирует номер студ. билета МГТУУ на основе пола и даты рождения студента.

Аргументы

in	sex	- пол (1 - женский, 0 - мужской)
in	y	- год рождения
in	m	- месяц рождения
in	d	- день рождения

Возвращает

ticket - номер студ. билета.

Первая цифра билета зависит от пола. Затем к ней добавляется дата рождения (г, м, д). После чего генерируется 5-изначное случайное число. Если число при делении на 10 дает нечетный остаток, то заменяется вторая цифра случайного числа, пока остаток не станет четным. После чего подбирается последняя цифра.

Замещает University (стр. 15).

См. определение в файле classes.cpp строка 15

Объявления и описания членов классов находятся в файлах:

- D:/c++/labs/lab_6_7/classes.h
- D:/c++/labs/lab_6_7/classes.cpp

6.2 Класс MIEM

Является подклассом University (стр. 15). Отвечат за генерацию билета для МИЭМ.

#include <classes.h>

Открытые члены

- std::string generate (int, int, int, int) override

  Генерирует номер студ. билета МИЭМ на основе пола и даты рождения студента.

6.2.1 Подробное описание

Является подклассом University (стр. 15). Отвечат за генерацию билета для МИЭМ.

См. определение в файле classes.h строка 19

6.2.2 Конструктор(ы)

6.2.2.1 MIEM()

MIEM::MIEM ( ) [inline]

См. определение в файле classes.h строка 25

6.2.3 Методы

6.2.3.1 generate()

std::string MIEM::generate (
int sex,
int y,
int m,
int d) [override], [virtual]

Генерирует номер студ. билета МИЭМ на основе пола и даты рождения студента.

Аргументы

in	sex	- пол (1 - женский, 0 - мужской)
in	y	- год рождения
in	m	- месяц рождения
in	d	- день рождения

Возвращает

ticket - номер студ. билета.

Первая цифра билета зависит от пола. Затем к ней добавляется дата рождения (г, м, д). После чего генерируется 4-хзначное случайное число. Если число при делении на 11 дает остаток 10, то заменяется третья цифра случайного числа, пока остаток равен 10. После чего подбирается последняя цифра.

Замещает University (стр. 15).

См. определение в файле classes.cpp строка 76

Объявления и описания членов классов находятся в файлах:

- D:/c++/labs/lab_6_7/classes.h
- D:/c++/labs/lab_6_7/classes.cpp

6.3 Класс TicketGenerator

Вспомогательный класс, для создания студ. билетов.

#include <classes.h>

Открытые члены

- University * generator (std::string)

  Возвращает объект класса MIEM (стр. 12) или MGTUU (стр. 11), в зависимости от переданного аргумента.

6.3.1 Подробное описание

Вспомогательный класс, для создания студ. билетов.

См. определение в файле classes.h строка 46

6.3.2 Конструктор(ы)

6.3.2.1 TicketGenerator()

```
TicketGenerator::TicketGenerator ( )  [inline]
```

См. определение в файле classes.h строка 48

6.3.3 Методы

6.3.3.1 generator()

```
University * TicketGenerator::generator (
            std::string name )
```

Возвращает объект класса MIEM (стр. 12) или MGTUU (стр. 11), в зависимости от переданного аргумента.

Аргументы

in	name	- название университета

Возвращает

university - объект класса MIEM (стр. 12) или MGTUU (стр. 11)

См. определение в файле classes.cpp строка 129

Объявления и описания членов классов находятся в файлах:

- D:/c++/labs/lab_6_7/classes.h
- D:/c++/labs/lab_6_7/classes.cpp

6.4 Класс University

Абстрактный класс. Является суперклассом для классов MIEM (стр. 12) и MGTUU (стр. 11).

#include <classes.h>

Открытые члены

- virtual std::string generate (int, int, int, int)=0

6.4.1 Подробное описание

Абстрактный класс. Является суперклассом для классов MIEM (стр. 12) и MGTUU (стр. 11).

См. определение в файле classes.h строка 11

6.4.2 Методы

6.4.2.1 generate()

```
virtual std::string University::generate (
                int ,
                int ,
                int ,
                int ) [pure virtual]
```

Замещается в MIEM (стр. 13) и MGTUU (стр. 12).

Объявления и описания членов класса находятся в файле:

- D:/c++/labs/lab_6_7/classes.h

Глава 7

Файлы

7.1 classes.cpp

```
00001 #include "classes.h"
00002
00015 std::string MGTUU::generate(int sex, int y, int m, int d) {
00016     std::string ticket = "";
00017     if (!sex) {
00018         ticket += std::to_string(man);
00019         sum += man;
00020     }
00021     else {
00022         ticket += std::to_string(woman);
00023         sum += woman;
00024     }
00025
00026     ticket += std::to_string(y * 10000 + m * 100 + d);
00027     for (int i = 1; i < ticket.size(); ++i) {
00028         sum += (ticket[i] - '0') * (i + 1);
00029     }
00030
00031     int date = y + m + d;
00032
00033     std::mt19937 gen(date);
00034     std::uniform_int_distribution<> distr(1000, 9999);
00035     std::string id = std::to_string(distr(gen));
00036
00037     for (int i = 0; i < id.size(); ++i)
00038         sum += (id[i] - '0') * (i + 10);
00039
00040     while ((sum % mult) % 2 == 1) {
00041         sum -= id[1] * 11;
00042         int new_d = (id[1] - '0' + 1) % 10;
00043         char new_dc = new_d + '0';
00044         std::cout <<"d " << new_d << ' ' << new_dc << std::endl;
00045         id = {id[0], new_dc, id[2], id[3]};
00046         sum += id[1] * 11;
00047         std::cout << "s " << sum << ' ' << sum % 10 << std::endl;
00048     }
00049
00050     ticket += id;
00051
00052     int ost = sum % mult;
00053     for (int i = 0; i < 10; i++) {
00054         if ((ost + (i * 14) % mult) % mult == 0) {
00055             ost = 0;
00056             ticket += std::to_string(i);
00057             break;
00058         }
00059     }
00060
00061     return ticket;
00062 }
00063
00076 std::string MIEM::generate(int sex, int y, int m, int d) {
00077     std::string ticket = "";
00078     if (!sex) {
00079         ticket += std::to_string(man);
00080         sum += man;
00081     }
00082     else {
```

```
00083         ticket += std::to_string(woman);
00084         sum += woman;
00085     }
00086
00087     ticket += std::to_string(y * 10000 + m * 100 + d);
00088     for (int i = 1; i < ticket.size(); ++i) {
00089         sum += (ticket[i] - '0') * (i + 1);
00090     }
00091
00092     int date = y + m + d;
00093
00094     std::mt19937 gen(date);
00095     std::uniform_int_distribution<> distr(10000, 99999);
00096     std::string id = std::to_string(distr(gen));
00097
00098     for (int i = 0; i < id.size(); ++i)
00099         sum += (id[i] - '0') * (i + 10);
00100
00101     while (sum % mult == 4) {
00102         sum -= id[2] * 12;
00103         int new_d = (id[2] - '0' + 1) % 10;
00104         char new_dc = new_d + '0';
00105         id = {id[0], id[1], new_dc, id[3], id[4]};
00106         sum += id[2] * 12;
00107     }
00108
00109     ticket += id;
00110
00111     int ost = sum % mult;
00112     for (int i = 0; i < 10; i++) {
00113         if ((ost + (i * 15) % mult) % mult == 0) {
00114             ost = 0;
00115             ticket += std::to_string(i);
00116             break;
00117         }
00118     }
00119
00120     return ticket;
00121 }
00122
00123
00129 University* TicketGenerator::generator(std::string name) {
00130     if (name == "MIEM") {
00131         MIEM* university = new MIEM;
00132         return university;
00133     }
00134     MGTUU* university = new MGTUU;
00135     return university;
00136 }
```

7.2 classes.h

```
00001 #ifndef UNIVERSITY
00002 #define UNIVERSITY
00003
00004 #include <iostream>
00005 #include <string>
00006 #include <random>
00007
00011 class University {
00012 public:
00013     virtual std::string generate(int, int, int, int) = 0;
00014 };
00015
00019 class MIEM: public University {
00020     int man   =  8;
00021     int woman =  4;
00022     int sum   =  0;
00023     int mult  = 11;
00024 public:
00025     MIEM() {}
00026     std::string generate(int, int, int, int) override;
00027 };
00028
00032 class MGTUU: public University {
00033     int man   =  2;
00034     int woman =  1;
00035     int sum   =  0;
00036     int mult  = 10;
00037 public:
00038     MGTUU() {}
00039     std::string generate(int, int, int, int) override;
00040 };
00041
```

```
00042
00046 class TicketGenerator {
00047 public:
00048     TicketGenerator() {};
00049     University* generator(std::string);
00050 };
00051
00052 #endif
```

7.3 main.cpp

```
00001 #include "classes.h"
00002 #include <algorithm>
00003 #include <fstream>
00004 #include <cstring>
00005
00009 namespace sex{
00010     int man = 0;
00011     int woman = 1;
00012 };
00013
00014 std::string university; //< Название университета.
00015 std::string vsex;       //< Пол студента.
00016 int year;  //< Год рождения.
00017 int month; //< Месяц рождения.
00018 int day;   //< День рождения.
00019
00026 void usage() {
00027     std::cout « std::endl;
00028     std::cout « "USAGE: ./prog [--tofile <file_name>] [--fromfile <file_name>]" « std::endl;
00029     std::cout « std::endl;
00030     std::cout « "------------------------Flags meaning----------------------" « std::endl;
00031     std::cout « "--tofile   ------> write output data in file"                 « std::endl;
00032     std::cout « "--fromfile ------> read input data from file"                   « std::endl;
00033     std::cout « std::endl;
00034 }
00035
00041 char to_lower(char c) { return std::tolower(c); }
00042
00048 char to_upper(char c) { return std::toupper(c); }
00049
00055 int check_input() {
00056     if (university != "MIEM" && university != "MGTUU") {
00057         std::cerr « "##### Wrong name of university #####\n";
00058         return 1;
00059     }
00060
00061     if (vsex != "man" && vsex != "woman") {
00062         std::cerr « "##### Wrong sex #####\n";
00063         return 1;
00064     }
00065
00066     if (year < 1900 || year > 5000) {
00067         std::cerr « "##### Wrong year of birthday #####\n";
00068         return 1;
00069     }
00070     if (month < 1 || month > 12) {
00071         std::cerr « "##### Wrong month of birthday #####\n";
00072         return 1;
00073     }
00074     if (day < 1 || day > 31) {
00075         std::cerr « "##### Wrong day of birthday #####\n";
00076         return 1;
00077     }
00078
00079     return 0;
00080 }
00081
00087 int read() {
00088     std::cin » university;
00089     std::transform(university.begin(), university.end(), university.begin(), to_upper);
00090     std::cin » vsex;
00091     std::transform(vsex.begin(), vsex.end(), vsex.begin(), to_lower);
00092     std::cin » year » month » day;
00093
00094     if (check_input()) return 1;
00095     return 0;
00096 }
00097
00104 int read(std::string filename) {
00105     std::ifstream file (filename);
00106     if (!file.is_open()) {
00107         std::cerr « "##### Wrong name of file #####\n";
00108         return 1;
```

```
00109     }
00110
00111     std::getline(file, university);
00112     std::transform(university.begin(), university.end(), university.begin(), to_upper);
00113     std::getline(file, vsex);
00114     std::transform(vsex.begin(), vsex.end(), vsex.begin(), to_lower);
00115     file » year » month » day;
00116
00117     file.close();
00118
00119     if (check_input()) return 1;
00120     return 0;
00121 }
00122
00127 void print(std::string t) { std::cout « "Ticket is " « t « std::endl; }
00128
00134 void print(std::string t, std::string filename) {
00135     std::ofstream file (filename);
00136     file « "Ticket is " « t « '\n';
00137     file.close();
00138 }
00139
00147 std::string generate_ticket() {
00148     TicketGenerator ticket_generator;
00149     std::string ticket;
00150     if (vsex == "man")
00151         ticket = ticket_generator.generator(university)->generate(sex::man, year, month, day);
00152     else
00153         ticket = ticket_generator.generator(university)->generate(sex::woman, year, month, day);
00154     return ticket;
00155 }
00156
00171 int main(int argc, char ** argv) {
00172     std::string ticket;
00173     switch (argc) {
00174     case 1:
00175         if (read()) exit(1);
00176         ticket = generate_ticket();
00177         print(ticket);
00178         break;
00179     case 3:
00180         if (!strcmp(argv[1], "--fromfile")) {
00181             if (read(argv[2])) exit(1);
00182             ticket = generate_ticket();
00183             print(ticket);
00184         }
00185         else if (!strcmp(argv[1], "--tofile")) {
00186             if (read()) exit(1);
00187             ticket = generate_ticket();
00188             print(ticket, argv[2]);
00189         }
00190         else {
00191             std::cerr « "##### Wrong flags #####\n";
00192             usage();
00193             exit(1);
00194         }
00195         break;
00196     case 5:
00197         if (!strcmp(argv[1], "--tofile") && !strcmp(argv[3], "--fromfile")) {
00198             if (read(argv[4])) exit(1);
00199             ticket = generate_ticket();
00200             print(ticket, argv[2]);
00201         }
00202         else if  (!strcmp(argv[3], "--tofile") && !strcmp(argv[1], "--fromfile")) {
00203             if (read(argv[2])) exit(1);
00204             ticket = generate_ticket();
00205             print(ticket, argv[4]);
00206         }
00207         else {
00208             std::cerr « "##### Wrong flags #####\n";
00209             usage();
00210             exit(1);
00211         }
00212         break;
00213     default:
00214         std::cerr « "##### Wrong flags #####\n";
00215         usage();
00216         exit(1);
00217         break;
00218     }
00219
00220     return 0;
00221 }
```