Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Кафедра компьютерной безопасности

Отчёт

к лабораторным работам № 6-7

по дисциплине

«Языки программирования»

Работу

выполнила

студентка

группы СКБ221

В. А. Кинаш

Москва 2022

Постановка задачи

Требуется сгенерировать студенческий билет вида `SYYYYMMDDNC` для заданного университета, пола и даты рождения. S – цифра, отвечающая за пол, YYYY – год рождения учащегося, MM – месяц рождения, DD - число, N - псевдослучайное число, подобранное так, чтобы оно было уникально для даты рождения, C - это цифра, подобранная так, чтобы сумма, состоящая из цифр студбилета, умноженных на их позицию в номере слева направо, начиная с 1, делилась на число k (разное для разного вуза).

Необходимо реализовать виртуальный класс-интерфейс для генераторов номеров студбилетов, от которого необходимо отнаследоваться два раза: МИЭМовским и ЭмГэТУУшным классом-генератором.

Код программы должен содержать ввод с консоли флага --tofile и --fromfile. Первый говорит нам, что ввод будет задан в заданный пользователем в командной строке файл (в случае отсутствия файла создать его), а второй говорит о считывании из файла заданного пользователем. Предусмотреть возможность исполнения обоих флагов. Учесть возможность некорректного ввода и обработать исключения.

Так же необходимо реализовать класс-генератор общего вида, используя паттерн Шаблонный Метод.

Создать Makefile с возможностью сборки, clean и distclean.

Алгоритм выполнения задачи

1) В Функции `main (int argc, char** argv)` в начале происходит обработка флагов, в которой мы определяем, откуда производится ввод, и куда производить вывод (основываясь на возможном наличие флагов `--fromfile, --tofile`). Если передано больше аргументов, или после флага не указано название файла, то возвращается `EXIT_FAILURE`. Для работы с файлами был подключен заголовочный файл `<fstream>`.

2) Если указан флаг `--fromfile`, создается объект класса `ifstream`, который связывается с файлом, из которого будет производиться чтение, с помощью метода `open(filename)`. После чего последовательно происходит чтение из файла входных данных: вуз студента, его пол и дата рождения в переменные `string uni, sex; int dd, mm, yy` соответственно.

3) Если этого флага не указано, то происходит считывание данных из терминала в такие же переменные.

4) Далее происходит обработка случаев, когда входные данные указаны некорректно: неправильно введен пол или вуз или введена несуществующая дата.

5) С помощью класса-генератора общего вида `template_pattern_student_number_generator` с помощью метода `generator(std::string name_of_university)` генерируется объект класса либо `MIEM`, либо `MGTUU`, которые в свою очередь с помощью метода `gen(std::string sex, int dd, int mm, int yy)` возвращают нужный студенческий билет.

6) Классы `MIEM` и `MGTUU` унаследованы от абстрактного класса `University`, в котором присутствует виртуальный метод генерации `gen`, переопределенный в дочерних классах, так как для каждого из вузов генерация билета происходит немного по-разному.

7) Метод `gen` в каждом из дочерних классов:

   a. Переводит по определенному правилу пол студента в переменную `int sexx`.
   b. С помощью функции `string id_gen(int sexx, int dd, int mm, int yy)`

генерируется первая часть студенческого билета вида `SYYYYMMDDN`.

c. Псевдослучайное число `N` гарнируется отдельно для каждого из двух классов с помощью функции `string random_miem(int date_of_birth)` и `string random_mgtuu(int date_of_birth)` с помощью генератора случайных чисел `std::mt19937` и `std::uniform_int_distribution`, позволяющего генерировать случайные числа в заданном диапазоне.

d. Подбирает такое число, чтобы сумма, состоящая из цифр студбилета, умноженных на их позицию в номере слева направо, начиная с 1, делилась на число, уникальное для каждого из классов.

8) Если указан флаг `--tofile`, создается объект класса `ofstream`, который связывается с файлом, в которой будет производиться вывод, с помощью метода `open(filename)`. После чего в нужный файл записывается полученный студенческий билет.

9) Если этого флага не указано, то происходит вывод билета на экран.

10) Для удобства сборки проекта был реализован `makefile` с возможностью сборки, `clean` и `distclean`.

Код проекта

main.cpp

```
#include <iostream>
#include <string>
#include <fstream>
#include <map>
#include "uni.h"

using std::cin;
using std::cout;
using std::string;

bool IsLeap(int year){
    return (year % 400 == 0) || ((year % 4 == 0) &&
(year % 100 != 0));
}

int days(int i, int y) {
    if (i == 1 || i == 3 || i == 5 || i == 7 || i ==
8 || i == 10 || i == 12)
        return 31;
    if (i == 4 || i == 6 || i == 9 || i == 11)
        return 30;
    return 28;
}

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

```
    const char* from_file = "-1";
    const char* to_file = "-1";

    if (argc % 2 == 0 || argc > 5) {
        std::cerr << "Incorrect format for flags";
        return EXIT_FAILURE;
    }
    else if (argc == 3) {
        if (std::strcmp(argv[1], "--fromfile") == 0){
            from_file = argv[2];
        }
        else if (std::strcmp(argv[1], "--tofile") ==
0){
            to_file = argv[2]; //если флаг вывода в
файл
        }
        else {
            std::cerr << "Incorrect format for
flags";
            return EXIT_FAILURE;
        }
    }
    else if (argc == 5) {
        if (std::strcmp(argv[1], "--fromfile") == 0
and std::strcmp(argv[3], "--tofile") == 0){
            from_file = argv[2];
            to_file = argv[4];
        }
        else if (std::strcmp(argv[1], "--tofile") ==
0 and std::strcmp(argv[3], "--fromfile") == 0 ){
            from_file = argv[4];
            to_file = argv[2];
```

```
        }
        else {
            std::cerr << "Incorrect format for
flags";
            return EXIT_FAILURE;
        }
    }

    std::map <int, int> days_in_months;
    for (int i = 1; i <= 12; ++i) {
        days_in_months[i] = days(i, 2022);
    }

    string uni, sex;
    int yy, mm, dd;

    if (strcmp(from_file, "-1") != 0) { //из файла
        std::ifstream fin;
        fin.open(from_file);

        fin >> uni >> sex >> yy >> mm >> dd;
        if (IsLeap(yy)
            days_in_months[2] = 29;

        if (!((uni == "miem" || uni == "mgtuu") &&
(sex == "man" || sex == "woman")
        && yy >= 0 && (mm <= 12 || mm >= 1) && (dd >=
0 && dd <= days_in_months[mm]))) {
            std::cerr << "Incorrect format";
            return EXIT_FAILURE;
        }
```

```
        cout << uni << ' ' << sex << ' ' << yy << ' '
<< mm << ' ' << dd <<'\n';


    }
    else {
        cout << "-----Please enter the university:
miem or mgtuu-----\n";
        cin >> uni;
        if (!(uni == "miem" || uni == "mgtuu")) {
            std::cerr << "Incorrect format for the
university";
            return EXIT_FAILURE;
        }


        cout << "-----Please enter your sex: man or
woman-----\n";
        cin >> sex;
        if (!(sex == "man" || sex == "woman")) {
            std::cerr << "Incorrect format for the
sex";
            return EXIT_FAILURE;
        }


        cout << "-----Enter the year of your birth---
--\n";
        cin >> yy;
        if (yy < 0) {
            std::cerr << "Incorrect format for the
year";
            return EXIT_FAILURE;
        }
        if (IsLeap(yy))
            days_in_months[2] = 29;
```

```
        cout << "-----Enter the month of your birth-----\n";
        cin >> mm;
        if (mm > 12 || mm < 1) {
            std::cerr << "Incorrect format for the month";
            return EXIT_FAILURE;
        }

        cout << "-----Enter the day of your birth-------\n";
        cin >> dd;
        if (dd < 0 || dd > days_in_months[mm]) {
            std::cerr << "Incorrect format for the day";
            return EXIT_FAILURE;
        }
    }
    template_pattern_student_number_generator templateGenerator;
    std::string SB = templateGenerator.generator(uni)->gen(sex, dd, mm, yy);

    if (strcmp(to_file, "-1") == 0){
        cout << SB << '\n';
    }
    else {
        std::ofstream fout;
        fout.open(to_file);
        fout << SB << '\n';
    }
```

```
    return 0;
}
```

uni.h

```
 #include <iostream>

class University {
public:
    std::string sex;
    University() {}
    virtual std::string gen(std::string, int,
int, int) = 0;
};

class MIEM : public University {
public:
    MIEM(){}
    std::string gen(std::string, int, int, int)
override;
};

class MGTUU : public University {
public:
    MGTUU(){}
    std::string gen(std::string, int, int, int)
override;
};

std::string first_gen(int, int, int, int);

class template_pattern_student_number_generator {
public:
    template_pattern_student_number_generator()
{};
    University* generator(std::string);
};
```

uni.cpp

```
#include <iostream>
#include "uni.h"
#include <random>
#include <time.h>

using std::string;

string random_miem(int date_of_birth) {
    std::mt19937 gen(date_of_birth + time(0));
    std::uniform_int_distribution<> values(10000,
99999);
    return std::to_string(values(gen));
}

string random_mgtuu(int date_of_birth) {
    std::mt19937 gen(date_of_birth + time(0));
    std::uniform_int_distribution<> values(1000,
9999);
    return std::to_string(values(gen));
}

string id_gen(int sex, int dd, int mm, int yy) {
    string id = "111111111";
    id[0] = '0' + sex;
    for (int i = 4; i >= 1; --i) {
        id[i] = '0' + yy % 10;
        yy /= 10;
    }
    id[5] = '0' + mm / 10;
    id[6] = '0' + mm % 10;
    id[7] = '0' + dd / 10;
    id[8] = '0' + dd % 10;

    int date_of_birth = yy * 10000 + mm * 100 + dd;
    if (sex == 1 || sex == 2)
        id += random_mgtuu(date_of_birth);
    else
        id += random_miem(date_of_birth);
```

```
    return id;
}


string MIEM::gen(string sex, int dd, int mm, int yy){
    int sexx = (sex == "man" ? 8 : 4);
    string id = id_gen(sexx, dd, mm, yy);

    int sum = 0;
    for (int i = 0; i < id.size(); ++i) {
        sum += (i + 1) * (id[i] - '0');
    }
    int i = 0;
    for (; i <= 9; ++i) {
        if ((sum + 15 * i) % 11 == 0){break;}
    }
    id += std::to_string(i);


    return id;
}

string MGTUU::gen(string sex, int dd, int mm, int
yy){
    int sexx = (sex == "man" ? 2 : 1);
    string id = id_gen(sexx, dd, mm, yy);

    int sum = 0;
    for (int i = 0; i < id.size(); ++i) {
        sum += (i + 1) * (id[i] - '0');
    }
    int i = 0;
    for (; i <= 9; ++i) {
        if ((sum + 14 * i) % 10 == 0){break;}
    }
    id += std::to_string(i);

    return id;
}
```

```
University*
template_pattern_student_number_generator::generator(
std::string name) {
    if (name == "miem") {
        MIEM* uni = new MIEM;
        return uni;
    }

    MGTUU* uni = new MGTUU;
    return uni;
}
```

makefile

```
out : main.o uni.o
    g++ main.o uni.o -o out

main.o : main.cpp
    g++ -c main.cpp -o main.o

fun.o : uni.cpp
    g++ -c uni.cpp -o uni.o

clean :
    rm out *.o

distclean :
    rm out *.o *.txt
```

Вывод

В ходе данной лабораторной работы я изучила новые способы генерации псевдослучайных чисел в c++, потренировалась в реализации полиморфизма, чисто абстрактных классов и познакомилась с паттерном «Шаблонный метод».