Отчет к Лабораторной работе 6-7.

Хедер class.hpp

В хедере используем заголовочные файлы <iostream> <map>, <cstdlib>, <fstream>, <cstring>.

Student_Ticket_Generator – виртуальный класс-интерфейс, описывающий структуру выполнения генерации студенческого билета, который содержит 2 чисто виртуальных метода Generate_random_number и generate и 2 метода, которые одинаковы для всех классов-наследников. Метод Summurise складывает все цифры в студбилете, а метод Random генерирует псевдорандомное число в диапазоне от min до max.

```
class Student_Ticket_Generator {
public:
    int Summurise(const std::string &str)  {
        int sum = 0;
        for (size_t i = 0; i < str.size(); ++i) {
            char c = str[i];
            sum += ((c - '0') * (i+1));
        }
        return sum;
    }

    int Random(int min, int max) {
        return min + std::rand() % (max - min);
    }

    virtual int
    Generate_random_number(std::string &year, std::string &month, std::string
&day,
                           std::map <std::string, std::vector<int>>
&randomNumbers) = 0;

    virtual std::string
    generate(std::string &sex, std::string &year, std::string &month,
std::string &day,
             std::map <std::string, std::vector<int>> &randomNumbers) = 0;
};
```

class MGTU – наследник класса Student_Ticket_Generator. Он переопределяет методы базового класса(полиморфизм). Также при переопределении дописывается модификатор override, который позволяет компилятору следить за правильностью переопределения метода.
Метод Generate_random_number генерирует четырехзначное псевдорандомное число так, чтобы ни у какого другого студента МГТУ с такой же датой рождения не было такого же. Такую проверку выполним с помощью словаря с ключом string, по которому лежит vector с уже сгенерированными числами.

```
    int
    Generate_random_number(std::string &year, std::string &month, std::string
&day,
                           std::map <std::string, std::vector<int>>
&randomNumbers) override {
        std::string s = (year) + (month) + (day);
        int number = Random(1000, 10000);
        if (randomNumbers.find(s) != randomNumbers.end()) {
            for (size_t i = 0; i < randomNumbers.find(s)->second.size(); ++i)
{
```

```
                if (randomNumbers[s][i] == number) {
                    number = Random(1000, 10000);
                } else {
                    randomNumbers[s].push_back(number);
                }
            }
        } else {
            randomNumbers[s] = {};
            randomNumbers[s].push_back(number);
        }
        return number;
    }
```

Метод generate выполняет генерацию последней цифры студбилета в соответствии с условием, определяет пол студента и возвращает весь его номер в виде строки.

```
    std::string
    generate(std::string &sex, std::string &year, std::string &month,
std::string &day,
             std::map <std::string, std::vector<int>> &randomNumbers)
override {
        std::string gender;
        if (sex == "man") {
            gender = "2";
        } else if (sex == "woman") {
            gender = "1";
        }
        int number = Generate_random_number(year, month, day, randomNumbers);
        std::string s =
                gender + (year) + (month) + (day) + std::to_string(number);
        int Sum = Summurise(s);
        for (int i = 0; i < 10; ++i) {
            if ((Sum + i) % 11 == 0) {
                s = s + std::to_string(i);
                break;
            }
        }
        return s;
    }
};
```

Класс MIEM – такой же наследник класса Student_Ticket_Generator, но в нем методы базового класса переопределены в соответствии с другими правилами(см. условие).

Создаем структуру Generator, в которой узнаем информацию об университете и создаем экземпляр соответствующего класса.

```
struct Generator {
    Student_Ticket_Generator* generator(std::string university) {
        if (university == "MGTU") {
            MGTU *a = new MGTU;
            return a;
        } else if (university == "MIEM") {
            MIEM *a = new MIEM;
            return a;
        }
        return nullptr;
    }
};
```

Файл class.cpp.

Имеем две функции. Первая – CorrectInformation преобразует информацию в вид, который нужен для представления студбилета.

```
void CorrectInformation(std::string &year, std::string &month, std::string
&day) {
    while (year.size() < 4) {
        year = "0" + year;
    }
    if (month.size() == 1) {
        month = "0" + month;
    }
    if (day.size() == 1) {
        day = "0" + day;
    }
}
```

Вторая – CheckInformation, проверяет корректность введенной информации о студенте.

```
bool
CheckInformation(std::string &university, std::string &sex, std::string
&year, std::string &month, std::string &day) {
    if ((university != "MIEM" && university != "MGTU") || (sex != "man" &&
sex != "woman") || std::stoi(year) > 9999 ||
        std::stoi(month) < 1 || std::stoi(month) > 12 ||
        std::stoi(day) < 1 ||
        std::stoi(day) > 31) {
        return false;
    }
    return true;
}
```

Файл main.cpp.

Здесь идет основная проверка на введенные флаги. В случае ошибки в веденных флагах, программа завершается с ошибкой

```
if (argc < 2 || argc > 5) {
    std::cerr << "Incorrect enter!" << std::endl;
    return -1;
}
```

В случае некорректно введенной информации, программа завершается

```
if (!CheckInformation(university, sex, year, month, day)) {
    std::cerr << "Wrong information, check your enter, please" << std::endl;
    return -2;
}
```

При обработке каждого флага и выполнении программы в соответствии с правилом, которое устанавливает конкретный флаг, выполняется проверка на корректность введенных данных, корректировка этих данных и генерация студбилета в зависимости от университета.

```
std::ofstream out("F");
while (std::cin >> university >> sex >> year >> month >> day) {
    if (!CheckInformation(university, sex, year, month, day)) {
        std::cerr << "Wrong information, check your enter, please" <<
std::endl;
        return -2;
    }
    CorrectInformation(year, month, day);
```

```
    if (university == "MIEM") {
        out << templateGenerator.generator(university)->generate(sex, year,
month, day, randomNumbers1) << std::endl;
    } else if (university == "MGTU") {
        out << templateGenerator.generator(university)->generate(sex, year,
month, day, randomNumbers2) << std::endl;
    }
}
```