Лабораторная работа №7

Реализует класс-генератор общего вида, используя шаблонный метод.

Объект *template_pattern_student_number_generator*

Класс, определяющий шаблонный метод. Содержит скелет алгоритма.

Функция get_sex возвращает требуемое значение пола для некоторого метода.

Функция BaseOperation составляет начало билета: пол + год + месяц + дата рождения.

Функция zfill дополняет строку str до длины n, приписывая слева нули.

Функция get_c генерирует часть билета, стоящую после даты рождения.

```
class template_pattern_student_number_generator
{
    public:
        std::string generator(int sex, int year, int month, int day) {
            std::string student_number;
            int newsex = this->get_sex(sex);
            student_number = this->BaseOperation(newsex);
            student_number += this->get_c(newsex, year, month, day);

            return student_number;

        }
    protected:
        int sex;
        int year;
        int month;
        int day;
        std::string BaseOperation(int newsex) {
            std::string personal_number;
            personal_number += std::to_string(newsex);
            personal_number += std::to_string(year);
            personal_number += zfill(std::to_string(month), 2);
            personal_number += zfill(std::to_string(day), 2);
            return personal_number;
        }

        std::string zfill(std::string str, int n) {
        std::string zeros(n - str.length(), '0');
        return zeros + str;
        }

        virtual std::string get_c(int newsex, int year, int month, int day) = 0;
        virtual int get_sex(int sex) = 0;
};
```

Объект *Miem*

Наследуется от объекта *template_pattern_student_number_generator* и переопределяет методы get_sex, get_c по заданным в условии правилам.

Метод get_random_number - генерирует случайное число, базируясь на переданной ему дате рождения.

```
class Miem : public template_pattern_student_number_generator {
    protected:
        int get_sex(int sex) {
```

```
        if (sex == 0) return 4;
        if (sex == 1) return 8;
        }

        int get_random_number(int year, int month, int day) {
            srand(std::stoi(std::to_string(year) + zfill(std::to_string(month), 2) + zfill(std::to_string(day), 2)));
            int num = rand() % 100000;
            return num;
        }

        std::string get_c(int newsex, int year, int month, int day) {
            int random_number = get_random_number(year, month, day);
            int c = sex + 2 * (year / 1000) + 3 *  ((year / 100) % 10) + 4 * ((year / 10) % 10) + 5 * (year % 10)
                    + 6 * (month / 10) + 7 * (month % 10) + 8 * (day / 10) + 9 * (day % 10)
                    + 10 * (random_number / 10000) + 11 * ((random_number / 1000) % 10) + 12 * ((random_number / 100) % 10)
                    + 13 * ((random_number / 10) % 10) + 14 * (random_number % 10);
            int i = 0;
            for (i; i <= 10; ++i) {
                if ((c + (i * 15)) % 11 == 0) break;
            }
            return zfill(std::to_string(random_number), 5) + std::to_string(i);
        }
};
```

Объект *Mgtu*

Аналогично объекту *Miem,* наследуется от объекта *template_pattern_student_number_generator* и переопределяет методы get_sex, get_c.

```
class Mgtu : public template_pattern_student_number_generator {
    protected:
        int get_sex(int sex) {
        if (sex == 0) return 1;
        if (sex == 1) return 2;
        }

        int get_random_number(int year, int month, int day) {
            srand(std::stoi(std::to_string(year) + zfill(std::to_string(month), 2) + zfill(std::to_string(day), 2)));
            int num = rand() % 10000;
            return num;
        }

        std::string get_c(int newsex, int year, int month, int day) {
            int random_number = get_random_number(year, month, day);
            int c = sex + 2 * (year / 1000) + 3 *  ((year / 100) % 10) + 4 * ((year / 10) % 10) + 5 * (year % 10)
                    + 6 * (month / 10) + 7 * (month % 10) + 8 * (day / 10) + 9 * (day % 10)
                    + 10 * ((random_number / 1000) % 10) + 11 * ((random_number / 100) % 10)
                    + 12 * ((random_number / 10) % 10) + 13 * (random_number % 10);
            int i = 0;
            for (int i = 0; i <= 10; ++i) {
                if ((c + (i * 15)) % 11 == 0) break;
            }
            return zfill(std::to_string(random_number), 4) + std::to_string(i);
        }
};
```