Лабораторная работа №6

Алгоритм программы

В зависимости от переданных флагов, программа считывает данные из файла или из консоли. После проверки ввода данных, в зависимости от ввода создается экземпляр класса *Miem* или *Mgtu*, от него вызывается функция *generate*, генерирующая номер студенческого билета.

Код

Файл main: функция *main*

В данном блоке считываем данные: флаги и ввод, из файла или из консоли. Проверяем ввод на ошибки и создаем экземпляр класса, от которого вызываем функцию. После чего записываем результат выполнения в файл/выводим в консоль и закрываем файл.

```
int main(int argc, char* argv[]) {
    char* input_path;
    char* output_path;
    bool flags[2] = {false,  // --fromfile
                     false}; // --tofile
    int input_path_index;
    int output_path_index;
    if (argc > 1) {
        for (int i = 1; i < argc; ++i) {
            if (strcmp("--fromfile", argv[i]) == 0) {
                input_path = argv[i + 1];
                flags[0] = true;
                ++i;
            }
            if (strcmp("--tofile", argv[i]) == 0) {
                output_path = argv[i + 1];
                flags[1] = true;
                ++i;
            }
        }
    }

    std::ifstream input(input_path);
    std::ofstream output(output_path);

    if (!input && flags[0]) {
        std::cerr << "Wrong input file name. Try again." << std::endl;
        return EXIT_FAILURE;
    }

    int uni;
    if (flags[0]) input >> uni;
    else {
        std::cout << "Select university (0 for miem, 1 for mgtu): ";
        std::cin >> uni;
    }
    if (uni != 0 && uni != 1) {
        std::cerr << "Wrong university name. Try again." << std::endl;
        return EXIT_FAILURE;
    }
```

```
    <...>

    if (flags[0]) input.close();

    if (uni == 0) {
        Miem id(sex, year, month, day);
        if (flags[1]) {
        output << id.generator();
        output.close();
        }
        else std::cout << id.generator() << std::endl;
    } else {
        Mgtu id(sex, year, month, day);
        if (flags[1]) {
        output << id.generator();
        output.close();
        }
        else std::cout << id.generator() << std::endl;
    }

    return EXIT_SUCCESS;
}
```

Файл objects: функция *zfill*

Аналог функции zfill из Python: дополняет строку str до длины n, приписывая слева нули.

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

Файл objects: функция *get_random_number*

Функция принимает на вход верхнюю границу, до которой генерировать число, год, месяц и день. Число, вида $\overline{abcdefgh}$, где $\overline{abcd} = year$, $\overline{ef} = month$, $\overline{gh} = day$ - база для генерации числа.

```
int get_random_number(int max, 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() % max;
  return num;
}
```

Файл objects: объект *StudentID*

Объект StudentID - виртуальный класс, от которого в последствии наследуются другие классы.

```
class StudentID
{
    protected:
        int sex;
        int year;
        int month;
        int day;
    public:
```

```
        virtual const std::string generator() = 0;
        virtual const int get_c(int sex, int year, int month, int day, int random_number) = 0;
};
```

Файл objects: объект *Miem*

При инициализации принимает на вход пол пользователя и дату его рождения. На основании этих данных, по заданным правилам генерирует номер студенческого билета.

```
class Miem : virtual StudentID
{
    public:
        Miem(int sex, int year, int month, int day) {
      this -> sex = sex;
      this -> year = year;
            this -> month = month;
            this -> day = day;
    };

    const int get_c(int sex, int year, int month, int day, int random_number) override {
        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 i;
    };

    const std::string generator() override {
        std::string personal_number = "";
        int newsex;
        if (sex == 0) newsex = 4;
        if (sex == 1) newsex = 8;
        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);
        int random_number = get_random_number(100000, year, month, day);
        personal_number += zfill(std::to_string(random_number), 5);
        int c = get_c(newsex, year, month, day, random_number);
        personal_number += std::to_string(c);

        return personal_number;
    };
};
```

Файл objects: объект *Mgtu*

Аналогичен классу *Miem*, генерирует номер студенческого билета по заданным правилам.

```
class Mgtu : virtual StudentID
{
    public:
        Mgtu(int sex, int year, int month, int day) {
      this -> sex = sex;
      this -> year = year;
            this -> month = month;
            this -> day = day;
```

```
    };

    const int get_c(int sex, int year, int month, int day, int random_number) override {
        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 i;
    };

    const std::string generator() override {
        std::string personal_number = "";
        int newsex;
        if (sex == 0) newsex = 1;
        if (sex == 1) newsex = 2;
        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);
        int random_number = get_random_number(10000, year, month, day);
        personal_number += zfill(std::to_string(random_number), 4);
        int c = get_c(newsex, year, month, day, random_number);
        personal_number += std::to_string(c);

        return personal_number;
    };
};
```