Функции main получает введенные пользователем флаги --fromfile и --tofile
и в зависимости от флага считывает
данные из выбранного файла и записывает результат в выбранный файл.
Порядок флагов не имеет значения. Если не было
введено флагов, то ввод и вывод осуществляется в коммандной строке.
Программа использует посимвольном вводе
из файла или из консоли и считывании из файла.
random_miem и random_mgtuu генерирует псевдослучайное число.
id_generate генерирует весь id.
poss_days проверяет возможное кол-во дней в месяце.
template_pattern_student_number_generator вызывает метод generator и
генерирует объекты MIEM и MGTUU.
generate возврпщпет нужный студ билет.

```
#include <iostream>
#include <string>
#include <cstring>
#include <fstream>
#include <map>
#include <random>
#include <time.h>

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

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

class MIEM : public Interface
{
public:
    MIEM(){}
    std::string generate(std::string, int, int, int) override;
};

class MGTUU : public Interface
{
public:
    MGTUU(){}
    std::string generate(std::string, int, int, int) override;
};

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

```
};

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

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


string id_generate(int sex, int d, int m, int y)
{
    string id = "";
    int date = y * 10000 + m * 100 + d;
    id += std::to_string(date);
    if (sex == 1 || sex == 2)
        id += random_mgtuu(date);
    else
        id += random_miem(date);

    return id;
}


string MIEM::generate(string sex, int d, int m, int y)
{
    int sexx;
    if (sex == "man") sexx = 8;
    else sexx = 4;
    string id = id_generate(sexx, d, m, y);
    int sum = 0;
    for (int i = 0; i < id.size(); ++i)
    {
        sum += (i + 1) * (id[i] - '0');
    }
    for (int c = 0; c <= 9; ++c)
    {
        if ((sum + 15 * c) % 11 == 0)
        {
            id += std::to_string(c);
            break;
        }
    }
    return id;
}

string MGTUU::generate(string sex, int d, int m, int y)
```

```
{
    int sexx;
    if (sex == "man") sexx = 2;
    else sexx = 1;
    string id = id_generate(sexx, d, m, y);
    int sum = 0;
    for (int i = 0; i < id.size(); ++i)
    {
        sum += (i + 1) * (id[i] - '0');
    }
    for (int c = 0; c <= 9; ++c)
    {
        if ((sum + 14 * c) % 10 == 0)
        {
            id += std::to_string(c);
            break;
        }
    }
    return id;
}

Interface*
template_pattern_student_number_generator::generator(std::string name)
{
    if (name == "miem")
    {
        MIEM* univer = new MIEM;
        return univer;
    }
    MGTUU* univer = new MGTUU;
    return univer;
}

int poss_days(int i)
{
    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* fromF = "0";
    const char* toF = "0";

    if (argc % 2 == 0 || argc > 5)
    {
        std::cout << "Incorrect flags";
        return EXIT_FAILURE;
    }
```

```
    else if (argc == 3)
    {
        if (std::strcmp(argv[1], "--fromfile") == 0)
        {
            fromF = argv[2];
        }
        else if (std::strcmp(argv[1], "--tofile") == 0)
        {
            toF = argv[2];
        }
        else
        {
            std::cout << "Incorrect flags";
            return EXIT_FAILURE;
        }
    }
    else if (argc == 5)
    {
        if (std::strcmp(argv[1], "--fromfile") == 0 and
std::strcmp(argv[3], "--tofile") == 0)
        {
            fromF = argv[2];
            toF = argv[4];
        }
        else if (std::strcmp(argv[1], "--tofile") == 0 and
std::strcmp(argv[3], "--fromfile") == 0 )
        {
            fromF = argv[4];
            toF = argv[2];
        }
        else
        {
            std::cout << "Incorrect format for flags";
            return EXIT_FAILURE;
        }
    }
    std::map <int, int> days_in_month;
    for (int i = 1; i <= 12; ++i)
    {
        days_in_month[i] = poss_days(i);
    }
    string univer, sex;
    int y, m, d;
    if (strcmp(fromF, "0") != 0)
    {
        std::ifstream fin;
        fin.open(fromF);
        fin >> univer >> sex >> y >> m >> d;
        if (!((univer == "miem" || univer == "mgtuu") && (sex == "man" ||
sex == "woman")
        && y >= 0 && (m <= 12 || m >= 1) && (d >= 0 && d <=
days_in_month[m])))
        {
            std::cout << "Incorrect format";
```

```
            return EXIT_FAILURE;
        }
        cout << univer << ' ' << sex << ' ' << y << ' ' << m << ' ' << d
<<'\n';
    }
    else
    {
        cout << "Enter the university name (miem or mgtuu)\n";
        cin >> univer;
        if (!(univer == "miem" || univer == "mgtuu"))
        {
            std::cout << "Incorrect iniversity";
            return EXIT_FAILURE;
        }

        cout << "Enter sex:\n";
        cin >> sex;
        if (!(sex == "man" || sex == "woman"))
        {
            std::cout << "Incorrect sex";
            return EXIT_FAILURE;
        }

        cout << "Enter the year of birth\n";
        cin >> y;
        if (y < 0)
        {
            std::cout << "Incorrect year";
            return EXIT_FAILURE;
        }
        cout << "Enter the month of birth\n";
        cin >> m;
        if (m > 12 || m < 1)
        {
            std::cout << "Incorrect month";
            return EXIT_FAILURE;
        }

        cout << "Enter the day of birth\n";
        cin >> d;
        if (d < 0 || d > days_in_month[m])
        {
            std::cout << "Incorrect day";
            return EXIT_FAILURE;
        }
    }
    template_pattern_student_number_generator templateGenerator;
    std::string SB = templateGenerator.generator(univer)->generate(sex,
d, m, y);
    if (strcmp(toF, "0") == 0)
    {
        cout << SB << '\n';
    }
    else
```

```
    {
        std::ofstream fout;
        fout.open(toF);
        fout << SB << '\n';
    }
    return 0;
}
```