Отчет по лабораторным работам №6-7

Обработка флагов

Флаги --tofile и --fromfile указывают на параметры ввода и вывода. Для их обработки я добавила в функцию main параметры agc и **argv. Сравнением строк определяется положение флагов. В случае, когда за флагом идет название файла, оно запоминается в переменную char* Учтена возможность введения двух флагов одновременно, а также некорректный ввод флагов. Алгоритм одинаков для введения флагов по отдельности и одновременно.

Основной алгоритм

Чтобы выполнить поставленную задачу мне потребовалось реализовать абстрактный класс *Student_id* реализация всех его чисто виртуальных методов содержать в себе наследники этого класса *MIEM* и *MGTUU*. В наследниках переопределены методы:

virtual std::string set_sex(std::string)=0; //устанавливает пол
virtual std::string set_number(int)=0;// генерирует псевдослучайное число
virtual std::string set_id(std::string sex, int y, int m, int d)=0;// создает номер билета

Для реализации метода set_number я использовала генератор случайных чисел mt19937, а чтобы задать диапазон значений – uniform_int_distribution

Реализован класс-генератор общего вида TemplateGenerator, который содержит в себе метод templateGenerator, который возвращает указатель на объект класса *Student_id* в соответствии с названием вуза. А метод set_id уже возвращает студенческий билет.

Makefie

Содержит в себе *clean* для быстрой очистки результатов сборки проекта и *distclean* для удаления файлов txt

Вывод

Я устала, а также узнала о псевдослучайных числах, абстрактных и виртуальных классах и паттернах.

Makefie

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

main.o : main.cpp
	g++ -c main.cpp -o main.o
clean :
	rm out *.o
distclean :
	rm out *.o test.txt
```

Main.cpp

```
#include<iostream>
#include<fstream>
#include<string>
#include <ctime>
#include<random>
class Student_id{
    protected:
    std::string sexx;
```

```
    int date;
    std::string number;
    std::string id;
    public:
    virtual std::string set_sex(std::string)=0;
    virtual std::string set_number(int)=0;
    virtual std::string set_id(std::string sex, int y, int m, int d)=0;
};
class MIEM: public Student_id{
    public:
    std::string set_sex(std::string s) override{
        s=="man"?sexx="8":sexx="4";
        return sexx;
    }
    std::string set_number(int d)override{
        std::mt19937 gen(d + time(0));
        std::uniform_int_distribution<> values(10000, 99999);
        number= std::to_string(values(gen));
        date=d;
        return number;

    }
    std::string set_id(std::string sex1, int y, int m, int d)override{
        set_sex(sex1);
        set_number(y);
        id=sexx+std::to_string(y)+std::to_string(m)+std::to_string(d)+number;
        int sum=0;
        for(int i=0;i<id.length();i++){
            sum+=(id[i]-'0')*(i+1);
        }
        int c;
        for(int i=0; i<10;i++){
            sum+=i*15;
            if(sum%11==0){
                c=i;
                break;
            }
            else{
                sum-=i*15;
            }
        }

        id+=std::to_string(c);
        return id;
    }

};
class MGTUU: public Student_id{
    public:
    std::string set_sex(std::string s) override{
        s=="man"?sexx="1":sexx="2";
        return sexx;
    }
    std::string set_number(int d)override{
        std::mt19937 gen(d + time(0));
        std::uniform_int_distribution<> values(1000, 9999);
        number= std::to_string(values(gen));
        date=d;
        return number;

    }
   std::string set_id(std::string sex1, int y, int m, int d)override{
        set_sex(sex1);
        set_number(y);
        id=sexx+std::to_string(y)+std::to_string(m)+std::to_string(d)+number;
        int sum=0;
```

```
        for(int i=0;i<id.length();i++){
            sum+=(id[i]-'0')*(i+1);
        }
        int c;
        for(int i=0; i<10;i++){
            sum+=i*14;
            if(sum%10==0){
                c=i;
                break;
            }
            else{
                sum-=i*15;
            }
        }

        id+=std::to_string(c);
        return id;
    }


};
class TemplateGenerator {
    public:
        Student_id* templateGenerator(std::string name) {
            if (name == "MIEM") {
                MIEM* miem_id = new MIEM;
                return miem_id;
            } else if (name == "MGTUU") {
                MGTUU* mgtuu_id = new MGTUU;
                return mgtuu_id;
            }
            exit(1);
        }
};
int main(int argc, char **argv){
    std::string uni,sex;
    int day, month, year;
    FILE *fp;
    bool tofile=false;
    bool fromfile=false;
    char *name1;
    char *name2;
    bool testfile=false;
    bool tofilename=false;
    if(argc==1){
        std::cerr<<"Nothing was read\n";
            return EXIT_FAILURE;
    }
    if(argc>5){
        std::cerr<<"Too many flags\n";
            return EXIT_FAILURE;
    }
    else{
        for (int i=1;i<argc;++i){
            if(std::strcmp(argv[i],"--tofile")==0){
                tofile=true;
                if((!argv[i+1]) || (std::strcmp(argv[i],"--fromfile")==0)){
                    std::ofstream oFile("test.txt");
                    testfile=true;
                }
                else{
                    name1=argv[i+1];

                }
            }
            if(std::strcmp(argv[i],"--fromfile")==0){
```

```
                fromfile=true;
                if((!argv[i+1]) || (std::strcmp(argv[i],"--tofile"))==0){
                    std::cerr<<"File name not specified\n";
                    return EXIT_FAILURE;
                }
                else{
                    name2=argv[i+1];
                }

            }

        }
    if(!tofile and !fromfile){
        std::cerr<<"Wrong flags\n";
    }
    }


    if(tofile && !fromfile){
        std::cout<<"Enter the name of the university in capital letters (MGTUU or
MIEM)\n";
        std::cin>>uni;
        if(uni!="MGTUU"&& uni!="MIEM"){
            std::cout<<"Wrong name\n";
            return EXIT_FAILURE;
        }
        std::cout<<"Enter your sex (man or woman)\n";
            std::cin>>sex;
            if(sex!="man"&& sex!="woman"){
            std::cout<<"Wrong sex\n";
            return EXIT_FAILURE;
        }
        std::cout<<"Enter your day of birth\n";
        std::cin>>day;
        if(day<0 || day>31){
            std::cout<<"Wrong day\n";
            return EXIT_FAILURE;
        }
        std::cout<<"Enter your month of birth\n";
        std::cin>>month;
         if(month<0 || month>12){
            std::cout<<"Wrong month\n";
            return EXIT_FAILURE;
        }
        std::cout<<"Enter your year of birth\n";
        std::cin>>year;
         if(year<1900 || year>2022){
            std::cout<<"Wrong year\n";
            return EXIT_FAILURE;
        }
        if(testfile){
            std::ofstream out;
            out.open("test.txt");
            if(out.is_open()){
                TemplateGenerator gen;
                std::string ticket=gen.templateGenerator(uni)-
>set_id(sex,year,month,day);
                out<<ticket<<'\n';
            }

            out.close();
            std::cout<<"Answer in file: test.txt"<<std::endl;
        }
        else{
            std::fstream output(name1);
            fp=fopen(name1,"w");
```

```
            if(output.is_open()){
                TemplateGenerator gen;
                std::string ticket=gen.templateGenerator(uni)-
>set_id(sex,year,month,day);
                output<<ticket<<'\n';
            }
            output.close();
            std::cout<<"Answer in file: "<<name1<<std::endl;
        }

    }
    if(fromfile&&!tofile){
        std::fstream input(name2);
        fp=fopen(name2,"r");
        input>>uni>>year>>month>>day;

       TemplateGenerator gen;
        std::string ticket=gen.templateGenerator(uni)->set_id(sex,year,month,day);
        std::cout<<ticket<<'\n';

    }
    if(fromfile && tofile){
        if(testfile){
            std::ofstream out;
            out.open("test.txt");
            std::fstream input(name2);
            fp=fopen(name2,"r");
            input>>uni>>year>>month>>day;
            if(out.is_open()){
               TemplateGenerator gen;
                std::string ticket=gen.templateGenerator(uni)-
>set_id(sex,year,month,day);
                out<<ticket<<'\n';
            }
            out.close();
            std::cout<<"Answer in file: test.txt"<<std::endl;
        }
        else{
            std::fstream input(name2);
            fp=fopen(name2,"r");
            input>>uni>>year>>month>>day;
             std::fstream output(name1);
            fp=fopen(name1,"w");
            if(output.is_open()){
               TemplateGenerator gen;
         std::string ticket=gen.templateGenerator(uni)->set_id(sex,year,month,day);
                output<<ticket<<'\n';
            }
            output.close();
            std::cout<<"Answer in file: "<<name1<<std::endl
        }
    }


}
```