Отчет о лабораторной работе номер 6

1. class interface

Описание класса:

Виртуальный класс-интерфейс

Методы класса:

Виртуальная функция std::string generateStudentCard.

Входные данные:

Принимает на вход пол человека, год рождения,месяц рождения,день рождения.

Описание функции:

Будет определена в классах-наследниках

```
class interface{
        public:
                virtual std::string generateStudentCard(std::string sex,std::string birthYear,std::string birthMonth,std::string birthDay )=0;

};
```

2. class miem : public interface

Описание класса:

Класс для создания номер студенческого билета МИЭМа.

Методы класса:

Функция std::string generateStudentCard(...) override final

Входные данные:

Те же, что и в классе-родителе

Описание функции:

1. Вначале получаем соответствующую полу человека цифру, проверяя правильность написания пола(поддерживается либо “man”,либо “woman”). Добавляем эту цифру в поток строк. В программе используется следующий метод получения конечной строки: записываем в нужной последовательности необходимые данные в соответствии с заданием в поток строк, потом с помощью метода .str() класса std::stringstream получаем конечную строку в нужном формате.

2. Получаем соответствующую дату рождения, проверяя правильность введенных дат. Записываем ее в поток

3. Получаем рандомное 5ти значное число с помощью rand(),используя в качестве ключа дату рождения.Записываем это число в поток.

4. Подбираем последнюю цифру таким образом, чтобы сумма цифр студ. билета вычисляемая в соотвествии с заданием была кратна 11. Записываем эту цифру в поток.

5. Преобразуем поток в строковую переменную, с помощью метода описанного в пункте 1. Описания этой функции.

```
class miem : public interface{
        public:
                std::string generateStudentCard(std::string sex,std::string birthYear,std::string
birthMonth,std::string birthDay ) override final{
                        std::stringstream res;

                        if(sex.length()==3||sex.length()==5){
                                if(sex=="man"){
                                        res << "8";
                                }
                                else{
                                        res << "4";
                                }
                        }
                        else {
                                std::cerr << "WRONG SEX" << std::endl;
                                exit(1);
                        }

                        std::string birthDat;
                        std::stringstream birthDate;
                        if(birthYear.length()==4){
                                birthDate << birthYear;
                        }
                        else{
                                std::cerr << "WRONG BIRTHYEAR FORMAT" << std::endl;
                                exit(1);
                        }


        if(birthMonth.length()==2&&(std::stoi(birthMonth)<=12&&std::stoi(birthMonth)>=1)){
                                birthDate << birthMonth;
                        }
                        else{
                                std::cerr << "WRONG BIRTHMONTH FORMAT" << std::endl;
                                exit(1);
                        }
```

```
                if(birthDay.length()==2&&(std::stoi(birthDay)<=31&&std::stoi(birthDay)>=1)){
                        birthDate << birthDay;
                }
                else{
                        std::cerr << "WRONG BIRTHDAY FORMAT" << std::endl;
                        exit(1);
                }
                birthDat = birthDate.str();
                res << birthDat;

                int randomNumber;
                srand(std::stoi(birthDat));
                randomNumber = rand()%(99999-10000+1)+10000;
                res << std::to_string(randomNumber);

                std::string calcC = res.str();
                int Multi = 0;
                for(int i=0;i<calcC.length();i++){
                        Multi = Multi + (i+1)*std::stoi(calcC.substr(i,1));

                }
                for(int i=0;i<10;i++){
                        if((Multi+i*15)%11==0){
                                res << std::to_string(i);
                        }
                }

                std::string result = res.str();
                return result;
        }
};
```

3. Класс mgtu не будет описан, ввиду того, что алгоритм и структура класса с классом МИЭМа практически индентична

4. std::string fromFileFunc(miem* miemObjects,mgtu* mgtuObjects,int number)

   Входные данные:

   Массивы хранящие объекты класса миэм и мгтуу.(размерность массивов 1, в самой первой версии предполагалось, что будет храниться больше объектов.В данном виде программа не поддерживает хранения множества массивов (>1) при вводе из файлы)

   Описание алгоритма:

   1. Считывается строка из файла. Далее обрабатывается отдельно для двух классов. Строка разделяется на параметры для последующего вызова функции-генератора. Возвращается значение из функции-генератора.

```
std::string fromFileFunc(miem* miemObjects,mgtu* mgtuObjects,int number){
```

```
        std::ifstream file("fromFileExample.txt"); //Format example: 0123456789
        std::string line;                                   //      mgtu woman 1999 11 21
        int mgtuPos = 0;                                    //      mgtu man 2005 01 07
        int miemPos = 0;
        if(file.is_open()){
                getline(file,line);
        }
        else{
                std::cerr << "FILE CANNOT BE OPENED" << std::endl;
                exit(1);
        }

        if(line.substr(0,4)=="mgtu"){
                if(line.substr(5,3)=="man"){
                        return
mgtuObjects[mgtuPos].generateStudentCard(line.substr(5,3),line.substr(9,4),line.substr(14,2),line.substr(17,
2));
                }
                else return
mgtuObjects[mgtuPos].generateStudentCard(line.substr(5,5),line.substr(11,4),line.substr(16,2),line.substr(19
,2));
        }
        else if(line.substr(0,4)=="miem"){
                if(line.substr(5,3)=="man"){
                        return
miemObjects[miemPos].generateStudentCard(line.substr(5,3),line.substr(9,4),line.substr(14,2),line.substr(17,
2));
                }
                else return
miemObjects[miemPos].generateStudentCard(line.substr(5,5),line.substr(11,4),line.substr(16,2),line.substr(1
9,2));
        }
        else{
                std::cerr << "WRONG FROMFILE INPUT" << std::endl;
                exit(1);
        }
}
```

Main

Описание алгоритма:

Проверяем наличие флагов при запуске программы. Создаем массивы для хранения объектов в случае считывания из файлов. Учитывая значения флагов, выводим результат в консоль, либо в файл. Ввод из файла, либо через редактирование кода. Ввод через консоль не предусмотрен.

```
int main(int argc, char** argv){
	bool toFile = false;
	bool fromFile = false;

	int number = 1;
	miem* miemObjects = new miem[number];
	mgtu* mgtuObjects = new mgtu[number];

	for(int i=0;i<argc;i++){
		if(strcmp(argv[i],"--toFile")==0) toFile = true;
		if(strcmp(argv[i],"--fromFile")==0) {
			fromFile = true;

		}
	}

	//if(fromFile) fromFileFunc(miemObjects,mgtuObjects,number);

	if(toFile&&fromFile==true){
		std::ofstream file("toFile.txt");
		file << fromFileFunc(miemObjects,mgtuObjects,number);
	}
	if(toFile==false&&fromFile==true){
		std::cout << fromFileFunc(miemObjects,mgtuObjects,number) << std::endl;
	}

	/*
	miem miemObject;
	miemObject.generateStudentCard("man","2005","01","07");
	std::cout << miemObject.data << std::endl;

	mgtu mgtuObject;
	mgtuObject.generateStudentCard("woman","1984","10","07");
	std::cout << mgtuObject.data << std::endl;
	*/
	delete[] miemObjects;
	delete[] mgtuObjects;
	return 0;
}
```

Лабораторная работа 7

В классе-шаблоне TemplateStudentCard в методе TemplateGenerator описан алгоритм составления номера студенческого билета. Вначале вызываем функцию, возвращающую определенную цифру в зависимости от пола человека (эта функция определяется в наследнике). Далее составляем дату

рождения, в данном случае алгоритм составления даты один и тот же для обоих видов студенческих билетов, поэтому определяем эту функцию в самом классе-шаблоне. Далее считаем случайное число (функция определяется в классах наследниках, т.к. для 1ого требуется 5значное число, а для 2ого 4ех значное).

Далее считаем последнее число (функция определяется в наследнике). И затем получаем номер студенческого билета. В целом алгоритмы функций такие же, как и в лр 6, только разделены на отдельные функции.

```
class TemplateStudentCard{
        public:
                std::string TemplateGenerator(std::string sex,std::string birthYear,std::string
birthMonth,std::string birthDay) const{
                        std::stringstream almostResultTemp;
                        std::string amlostResult;
                        std::string result;
                        SomeMethod1();
                        almostResultTemp << generateSex(sex);
                        std::string birthDate = generateBirthDate(birthYear,birthMonth,birthDay);
                        almostResultTemp << birthDate;
                        almostResultTemp << randomNumber(birthDate);
                        SomeMethod2();
                        std::string almostResult = almostResultTemp.str();
                        almostResultTemp << lastNumberRes(almostResult);
                        result = almostResultTemp.str();
                        return result;
                }
        protected:
                std::string generateBirthDate(std::string birthYear,std::string birthMonth,std::string
birthDay) const {
                        std::string birthDat;
                        std::stringstream birthDate;
                        if(birthYear.length()==4){
                                birthDate << birthYear;
                        }
                        else{
                                std::cerr << "WRONG BIRTHYEAR FORMAT" << std::endl;
                                exit(1);
                        }


        if(birthMonth.length()==2&&(std::stoi(birthMonth)<=12&&std::stoi(birthMonth)>=1)){
                                birthDate << birthMonth;
                        }
                        else{
                                std::cerr << "WRONG BIRTHMONTH FORMAT" << std::endl;
                                exit(1);
```

```
                }

                if(birthDay.length()==2&&(std::stoi(birthDay)<=31&&std::stoi(birthDay)>=1)){
                        birthDate << birthDay;
                }
                else{
                        std::cerr << "WRONG BIRTHDAY FORMAT" << std::endl;
                        exit(1);
                }
                birthDat = birthDate.str();
                return birthDat;
        }

        virtual std::string generateSex(std::string sex) const =0;
        virtual std::string randomNumber(std::string birthDate) const = 0;
        virtual std::string lastNumberRes(std::string alsmostRes) const = 0;

        virtual void SomeMethod1()const{}
        virtual void SomeMethod2()const{}
};
class Mgtuu : public TemplateStudentCard{
        protected:
                std::string generateSex(std::string sex) const override{
                        std::string res;
                        if(sex.length()==3||sex.length()==5){
                                if(sex=="man"){
                                        res = "2";
                                }
                                else{
                                        res = "1";
                                }
                        }
                        return res;
                }
                std::string randomNumber(std::string birthDate) const override{
                        std::string res;
                        int randomNumber;
                        srand(std::stoi(birthDate));
                        randomNumber = (rand()+1000)%10000;
                        res = std::to_string(randomNumber);
                        return res;
                }
                std::string lastNumberRes(std::string calcC) const override{
                        int Multi = 0;
```

```
                std::string res;
                for(int i=calcC.length()-1;i>=1;i--){
                        Multi = Multi + (i+1)*std::stoi(calcC.substr(i,1));
                }
                bool check = false;
                for(int i=0;i<10;i++){
                        if((Multi+i*1)%10==0){
                                check = true;
                                res = std::to_string(i);
                                break;
                        }
                }
                if(check){
                        return res;
                }
                else {
                        std::cerr << "SOMETHING WENT WRONG WHILE CALCULATING LAST NUMBER." << std::endl;
                        exit(1);
                }
        }
};

class Miem : public TemplateStudentCard{
        protected:
                std::string generateSex(std::string sex) const override{
                        std::string res;
                        if(sex.length()==3||sex.length()==5){
                                if(sex=="man"){
                                        res = "8";
                                }
                                else{
                                        res = "4";
                                }
                        }
                        return res;
                }
                std::string randomNumber(std::string birthDate) const override{
                        std::string res;
                        int randomNumber;
                        srand(std::stoi(birthDate));
                        randomNumber = (rand()%90000)+10000;
                        res = std::to_string(randomNumber);
                        return res;
```

```
        }
        std::string lastNumberRes(std::string calcC) const override{
                int Multi = 0;
                std::string res;
                for(int i=0;i<calcC.length();i++){
                        Multi = Multi + (i+1)*std::stoi(calcC.substr(i,1));
                }
                bool check = false;
                for(int i=0;i<10;i++){
                        if((Multi+i*15)%11==0){
                                check = true;
                                res = std::to_string(i);
                                break;
                        }
                }
                if(check){
                        return res;
                }
                else {
                        std::cerr << "SOMETHING WENT WRONG WHILE CALCULATING LAST
NUMBER." << std::endl;
                        exit(1);
                }
        }
};
```