Индивидуальное задание по ООП «создание класса»
Выполнил: Смирнов Евгений, 3530904/00006
Преподаватель: Киреев Сергей Петрович

1. Общая постановка задачи

1. Разработать класс «Фильм», содержащий информацию:
   - оригинальное название,
   - фамилия и инициалы режиссёра,
   - год выхода фильма,
   - фамилия и инициалы сценариста,
   - жанр фильма
   - время длительности фильма

2. Для разрабатываемого класса необходимо выполнить:
1. Описание полей класса.
2. Конструктор без параметров, конструктор с параметрами, конструкторы копирования, деструктор.
3. Перегрузку бинарного оператора + как метод класса.
4. Перегрузку бинарного оператора == через дружественную функцию.
5. Перегрузку оператора> отношения как метод класса.
6. Перегрузку оператора< отношения через дружественную функцию.
7. Перегрузку оператора присваивания.
8. Перегрузку префиксного инкремента «++» и «--» как метод класса.
9. Перегрузку постфиксного инкремента «++» и «--» через дружественную функцию.
10. Перегрузку операторов << и >> через дружественные функции.
11. Создать массив из объектов разработанного класса. Ввести в массив данные из текстового файла. Имя файла со входными данными должны вводиться из потока cin.
12. Определить последний вышедший фильм. Для сравнения использовать перегруженный оператор отношения, поиск элемента выполнить в отдельной процедуре.
13. Выполнить сортировку массива методом простых вставок по году выхода фильма. Для сортировки написать и использовать шаблон функции. Результат вывести в файл в виде таблицы, сделать заголовки колонок таблицы.
14. Создать список жанров фильмов. Результат вывести в файл в виде таблицы, сделать заголовки колонок таблицы
15. Предусмотреть обработку и инициализацию исключительных ситуаций, связанных, например, с проверкой значения полей перед инициализацией и присваиванием

Пользователь вводит название и путь файла – строку, из которой считывается в массив фильмы.
Пользователь вводит одно число от 0 до 6, где каждая цифра это способ вызвать следующее действие:

0- Выход из программы
1- Вывод в консоль и в файл списка фильмов
2- Вывод в консоль и в файл списка фильма отсортированного по году
3- Вывод в файл таблицы фильмов по жанрам
4- Узнать самый новый фильм
5- Узнать самый старый фильм
6- Увидеть работу перегрузок и методов класса фильм по заданию

2. Спецификация и требования

Класс «Фильм»:
Поля:
1. оригинальное название,
2. фамилия и инициалы режиссёра,
3. год выхода фильма,
4. фамилия и инициалы сценариста,
5. жанр фильма
6. время длительности фильма

Оригинальное название – строка, имеющая формат «Название Фильма» или «Title Of Film» Первая буква каждого слова обязательно должна быть заглавной.
В названии не разрешается использовать значки «_+=-)(;.,:%№?[]<>»!»

Фамилия и инициалы режиссёра – строка, имеющая формат «Иванов И. И.» или «Ivanov I. I.»
Формат записи должен строго соблюдаться, используя пробелы и точки как в примере выше.
В ФИО не разрешается использовать значки «_+=-)(;,:%№?[]<>»!»

Год выхода фильма – целое положительное число, большее 1894 и меньшее 2100.

Фамилия и инициалы сценариста – строка, имеющая формат «Иванов И. И.» или «Ivanov I. I.»
Формат записи должен строго соблюдаться, используя пробелы и точки как в примере выше.
В ФИО не разрешается использовать значки «_+=-)(;,:%№?[]<>»!»

Жанр фильма – строка состоящая из одного слова следующего формата «комедия» или «comedy». Все буквы нижнего регистра. В жанре не разрешается использовать значки «_+=-)(;.,:%№?[]<>»!»

Время длительности фильма (в минутах) – целое положительное число большее 10 и меньшее 1000

Примечание:
Все строки являются членами класса **myString**.
При несоответствии входных данных вышеуказанным форматам выбрасывается исключение класса **Exception**.

Класс myString:
Поля:

1. Указатель на массив элементов char[] не более 99 символов;
2. Длина строки – целое число

Для класса «MtStr» написаны следующие методы и перегружены следующие операторы:

- int getLenght() – возвращает длину строки;
- [](int i) – возвращает (i + 1)-ый символ строки; если i меньше 0 или больше длины строки, то выбрасывается исключение EXCEPTION;
- =(const myString& variable) – присваивает значение правой строки (типа myString&) левой строке;
- ==(const MyStr&) – сравнивает 2 строки MyStr, возвращает true, если длина строк одинакова и все символы тоже одинаковы, иначе возвращает false;
- friend std::istream& operator>>(std::istream& in, MyStr& ob) – считывает из потока ввода строку длинной не более 99!!! символов до символа перевода строки;
- friend std::ostream& operator<<(std::ostream& out, const MyStr& ob) – выводит строку в поток;
- void pushBack(char& temp) – добаляет в конец строки символ и увеличивает длину строки на 1

Класс Exception:

- Поля: строка messege_;
- Методы: getMessege() – возввращает сообщение о данной ошибке;

Номер ошибки	вывод
1	Surname is a big letter!
2	Forgot '.' or ' ' in initials!
3	Such short Name. It is impossible!
4	initials are big letters!
5	Incorrect fullname! Error in Register!
6	Incorrect genre. More 2 letters!
7	Every letter of title is a big letter!
8	Error in the title!
9	A year of film premiere could not be ealier then 1895!
10	File could not be opened!
11	So long String!
12	Out of bounds of string array!
13	Error! in genre illegal symbols or uppercase
14	Year is a number!
15	String is not more than 100 symbols!
16	Выход за границу массива строки класса май стринг
17	Массив состоит из более чем 0 элементов
18	Выход за границу массива при вызове[]
19	Минуты - целое число
20	Длительность в фильме не может быть больше 1000 минут и меньше 10 минут!

Класс myVector:
-- класс для любого типа данных (шаблон)

Поля
int number_;//количествово элементов в массиве
T* data_;//указатель на массив
Методы
myVector()\\ указатель на nulllptr и количество = 0
myVector(const int number)\\ конструктор с параметром
myVector(const myVector& variable)\\ конст копирования
~myVector()\\ обязательное удаление указателя

void pushBack(const T& variable)\\ добавляет в конец элемент и увеличивает кол-во массива

T& operator[] (const int index) \\ возвращает значение по индексу

template<typename T>
friend std::ostream& operator<< (std::ostream& out, myVector<T>& varible); //вывод

template<typename T>
friend std::istream& operator>> (std::istream& in, myVector<T>& variable); //ввод

int getNumber()- возвращает кол-во элементов в массиве

Все ошибки обрабатываются классом Excepton

Класс Matrix:
-- класс для любого типа данных (шаблон)

поля
```
T** M; // матрица
int m_; // количество строк
int n_; // количество столбцов
```

методы
```
// конструктор 0 0 nullptr
MATRIX()

// конструктор с параметрами
MATRIX(int m, int n)

// Конструктор копирования
MATRIX(const MATRIX& _M)

//getter – вытаскивает элемент  типа T из матрицы
T GetMij(int i, int j)

// метод выводящий длину самой длинной строки в массиве – используется для вывода
int GetMaxLength()

// вместо [] ( без перешрузки ) используется метод для этого– устанавливает в матрицу новое значение
void SetMij(int i, int j, T value)

// метод, выводящий матрицу
void Print()

// оператор перегрузка присваивания
MATRIX operator=(const MATRIX& _M)

// Деструктор → ОБЯЗАТЕЛЬНО ! Удалять нужно указательни на все строки и на указатель первый
~MATRIX()
```

3. Тест план

Номер	Спецификация	Входные данные	Вывод
1	1.1	Гарри поттер И Тайная Комната Крис К. Л. 2002 Стив К. Л. фентези 100	Every first letter of title is a big letter
2	1.1	ГарриЗ Поттер И Тайная Комната Крис К. Л. 2002 Стив К. Л. фэнтези 100	Error in the title
3	1.1	Гарри Поттер И* Тайная Комната Крис К. Л. 2002 Стив К. Л. фэнтези 100	Error in the title
4	1.2	Home Alone gosnel R. J. 1997 Hughes J. H. comedy 100	Surname is a big letter
5	1.2	Home Alone Gosnel r. j. 1997 Hughes J. H. comedy 100	initials are big letters!
6	1.2	Home Alone Go5snel R. J. 1997 Hughes J. H. comedy 100	Incorrect fullname. Error in Register
7	1.2	Home Alone Gosnel R.) J. 1997 Hughes J. H. comedy 100	Incorrect fullname. Error in Register

8	1.2	Home Alone Go R. J. 1997 Hughes J. H. comedy 100	Such short Name. It is impossible!
9	1.3	Taxi Gerard K. R. -2000 Luc B. S. comedy 100	A year of film premiere could not be ealier then 1895
10	1.3	Taxi Gerard K. R. Boo Luc B. S. comedy 100	Year is a number!
11	1.4	Сияние Кубрик С. Т. 1980 гинг С. Т. ужасы 100	Surname is a big letter
12	1.4	Сияние Кубрик С. Т. 1980 Кинг r. . ужасы 100	initials are big letters!
13	1.4	Сияние Кубрик С. Т. 1980 Ки__нг С. Т. ужасы 100	Incorrect fullname. Error in Register
14	1.4	Сияние Кубрик С. Т. 1980 КИнг С. Т. ужасы 100	Incorrect fullname. Error in Register
15	1.4	Сияние Кубрик С. Т. 1980 Ки. ужасы 100	Such short Name. It is impossible!

16	1.4	Дом Странных Детей Кубрик С Т 2015 Кинг С. Т. ужасы 100	Forgot '.' or ' ' in initials
17	1.4	Дом Странных Детей Кубрик С. Т. 2015 КингС.Т. ужасы 100	Forgot '.' or ' ' in initials
18	1.5	Доктор Стрейнджлав Кубрик С. Т. 1964 Кубрик С. Т. ко 100	Incorrect ganre. More 2 letters!
19	1.5	Доктор Стрейнджлав Кубрик С. Т. 1964 Кубрик С. Т. комедия 100	Error! in genre illegal symbols or uppercase
20	1.5	Доктор Стрейнджлав Кубрик С. Т. 1964 Кубрик С. Т. КОМЕДИЯ 100	Error! in genre illegal symbols or uppercase
21	1.5	Доктор Стрейнджлав Кубрик С. Т. 1964 Кубрик С. Т. коме+дия 100	Error! in genre illegal symbols or uppercase
22	1.6	Доктор Стрейнджлавмуин Кпавпекупцуемпамеыкпркк кукцерфпцкцрк Кубрик С. Т. 1964 Кубрик С. Т. коме+дия 100	So long String! Maximum 100 symbols
23	1.7	Гарри Поттер И Тайная Комната Крис К. Л. 2002 Стив К. Л. фэнтази 6	Длительность в фильме не может быть больше 1000 минут и меньше 10 минут

25	1.8	T4h3jy54mu65,rj -- Если файл не будет найден --	File could not opened

Результат выполнения выбора команды пользователя

1

Название	Режиссер	Премьера	Сценарист	Жанр	Продолжительность
Гарри Поттер И Тайная Комната	Крис К. Л.	2002	Стив К. Л.	фэнтази	96
Home Alone	Gosling R. J.	1997	Hughes J. H.	comedy	60
Taxi	Gerard K. R.	2000	Luc B. S.	comedy	45
Shrek	Andrew A. D.	2001	Ted E. L.	cartoon	34
Звериполис	Байрон Х. М.	2016	Джаред Б. Ш.	мультфильм	56
Гарри Поттер И Философский Камень	Крис К. Л.	2001	Стив К. Л.	фэнтези	120
Гарри Поттер И Узник Аскабана	Крис К. Л.	2004	Стив К. Л.	фэнтези	140
Гарри Поттер И Кубок Огня	Крис К. Л.	2005	Стив К. Л.	фэнтези	180
Гарри Поттер И Орден Феникса	Крис К. Л.	2007	Стив К. Л.	фэнтези	200
Доктор Стрейнджлав	Кубрик С. Т.	1964	Кубрик С. Т.	комедия	12
Сияние	Кубрик С. Т.	1980	Кинг С. Т.	ужасы	110
Холоп	Шипенко К. Л.	2019	Грацевич Д. А.	комедия	90
Сияние Начало	Кубрик С. Т.	1980	Кинг С. Т.	ужасы	99
Дом Странных Детей	Кубрик С. Т.	1980	Кинг С. Т.	ужасы	78
Гарри Поттер И Принц Полукровка	Крис К. Л.	2009	Стив К. Л.	фэнтези	890
Frozen	Kris B. K.	2013	Jenefir L. L.	cartoon	102
Toy Story	Bowel L. M.	1995	Lower L. M.	cartoon	100

2

Название	Режиссер	Премьера	Сценарист	Жанр	Продолжительность
Холоп	Шипенко К. Л.	2019	Грацевич Д. А.	комедия	90
Звериполис	Байрон Х. М.	2016	Джаред Б. Ш.	мультфильм	56
Frozen	Kris B. K.	2013	Jenefir L. L.	cartoon	102
Гарри Поттер И Принц Полукровка	Крис К. Л.	2009	Стив К. Л.	фэнтези	890
Гарри Поттер И Орден Феникса	Крис К. Л.	2007	Стив К. Л.	фэнтези	200
Гарри Поттер И Кубок Огня	Крис К. Л.	2005	Стив К. Л.	фэнтези	180
Гарри Поттер И Узник Аскабана	Крис К. Л.	2004	Стив К. Л.	фэнтези	140
Гарри Поттер И Тайная Комната	Крис К. Л.	2002	Стив К. Л.	фэнтази	96
Shrek	Andrew A. D.	2001	Ted E. L.	cartoon	34
Гарри Поттер И Философский Камень	Крис К. Л.	2001	Стив К. Л.	фэнтези	120
Taxi	Gerard K. R.	2000	Luc B. S.	comedy	45
Home Alone	Gosling R. J.	1997	Hughes J. H.	comedy	60
Toy Story	Bowel L. M.	1995	Lower L. M.	cartoon	100
Сияние	Кубрик С. Т.	1980	Кинг С. Т.	ужасы	110
Сияние Начало	Кубрик С. Т.	1980	Кинг С. Т.	ужасы	99
Дом Странных Детей	Кубрик С. Т.	1980	Кинг С. Т.	ужасы	78
Доктор Стрейнджлав	Кубрик С. Т.	1964	Кубрик С. Т.	комедия	12

3

LIST OF FILMS GENRES

мультфильм	фэнтази	фэнтези	comedy	cartoon	ужасы
Звериполис	Гарри Поттер И Тайная Комната	Гарри Поттер И Принц Полукровка	Taxi	Frozen	Сияние
		Гарри Поттер И Орден Феникса	Home Alone	Shrek	Сияние Начало
		Гарри Поттер И Кубок Огня		Toy Story	Дом Странных Детей
		Гарри Поттер И Узник Аскабана			
		Гарри Поттер И Философский Камень			

4

Холоп	Шипенко К. Л.	2019	Грацевич Д. А.	комедия	90

5

Доктор Стрейнджлав	Кубрик С. Т.	1964	Кубрик С. Т.	комедия	12

КОД ПРОГРАММЫ

MAIN

```
#include <iostream>
#include <fstream>

#include "Matrix.h"
#include "MyVector.h"
#include "Film.h"
#include "Exception.h"
#include "MyString.h"


Exception ERROR;

const myString OUT = "Нажмите 1, чтобы вывести данные в консоль и в файл \nНажмите 2,чтобы вывести данные в консоль и файл отсортированных по году фильмов \nНажмите 3, чтобы вывести в файл таблицу фильмов по жанрам\nНажмите 4, чтобы узнать самый новый фильм \nНажмите 5, чтобы узнать самый старый фильм\nНажмите 6, чтобы увидеть работу перегрузок и тд.\nЧтобы выйти, нажмите 0";

template<class A>
void SwapFilms(A& year1, A& year2);

template <class C>
void SortFilms(myVector<C>& cinema);

template <class B>
void OutputToFile(myVector<B>& cinema);

template <class E>
void OutputToConsole(myVector<E>& cinema);

template <class S>
void MakeAndOutListGenre(myVector<S>& cinema);

Film getNewestFilm(myVector<Film> cinema);
Film getOldestFilm(myVector<Film> cinema);
void inputFromFile(myVector <Film>& cinema);


int main()
{
	setlocale(LC_ALL, "RUS");
	myVector<Film> cinema;
	inputFromFile(cinema);
	int choiser;

	std::cout << OUT << std::endl << std::endl;

	while (std::cin >> choiser)
	{
		if (choiser == 1)
		{
			OutputToConsole(cinema);
			OutputToFile(cinema);
		}
		if (choiser == 2)
		{
			SortFilms(cinema);
			OutputToConsole(cinema);
			OutputToFile(cinema);
```

```
		}
		if (choiser == 3)
		{
			MakeAndOutListGenre(cinema);
		}
		if (choiser == 4)
		{
			Film newest = getNewestFilm(cinema);
			std::cout << newest;;
		}
		if (choiser == 5)
		{
			Film oldest = getOldestFilm(cinema);
			std::cout << oldest;
		}
		if (choiser == 6)
		{
			std::cout << "Работа перегрузки операторов" << "\n";

			std::cout << "Вывод фильма 1 <<" << "\n";
			Film f1 = cinema[1];
			std::cout << f1 << "\n";

			std::cout << "Вывод фильма 2 <<" << "\n";
			Film f2 = cinema[2];
			std::cout << f2 << "\n";

			std::cout << "Сложение(+) фильмов 1 и 2- общее время длительности" <<
"\n";
			int time = f1 + f2;
			std::cout << "общее время:" << time << "\n";

			std::cout<<" премьера фильма(getyear) - " << f1.getYear() << "\n";
			std::cout << "Фильм 3 , созданные без параметров:" << "\n";
			Film  t;
			std::cout << t << "\n";
			std::cout << "Присваение фильму 3 полей фильма 1 и использование
постфикса ++" << "\n";
			t = f1++;
			std::cout << "год фильма 3 - " << t.getYear() << "\n";
			std::cout << "год фильма 1- " << f1.getYear() << "\n";

			std::cout << "Присваение фильму 3 полей фильма 2 и использование
префикса --" << "\n";
			t = --f2;
			std::cout << "год фильма 3- " << t.getYear() << "\n";
			std::cout << "год фильма 2- " << f2.getYear() << "\n";

			myString name = "Петров П. П.";
			std::cout << "изменить параметр директор у фильма 1 на петров через
set" << "\n";
			f1.setDirector(name);
			std::cout << f1 << "\n";
		}
		if (choiser == 0)
		{
			break;
		}

		std::cout << std::endl << OUT << std::endl << std::endl;
	}
}
// swaper
template <class A>
```

```
void SwapFilms(A& year1, A& year2)
{
	A temp = year1;
	year1 = year2;
	year2 = temp;
}


//  вывод в файл массива фильмов
template <class B>
void OutputToFile(myVector<B>& cinema )
{
	std::ofstream fileOut;

	try
	{
		fileOut.open("Output.txt");
		if (!fileOut) {
			ERROR = 10;
			throw ERROR;
		}
	}
	catch (Exception error)
	{
		error.showException();
	}

	for (int i = 0; i < 6; i++) {
		fileOut << ' ';
	}
	fileOut << "   Название";
	for (int i = 0; i < 34; i++) {
		fileOut << ' ';
	}
	fileOut << "Режиссер";
	for (int i = 0; i < 21; i++) {
		fileOut << ' ';
	}
	fileOut << "Премьера";
	for (int i = 0; i < 14; i++) {
		fileOut << ' ';
	}
	fileOut << "Сценарист";
	for (int i = 0; i < 25; i++) {
		fileOut << ' ';
	}
	fileOut << "Жанр";
	for (int i = 0; i < 10; i++) {
		fileOut << ' ';
	}
	fileOut << "Продолжительность";
	fileOut << std::endl;
	fileOut << "-------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------" << std::endl;
	for (int i = 0; i < cinema.getNumber(); i++)
	{
		fileOut << cinema[i];
	}
	fileOut.close();
}


// вывод в консоль массива фильмов
```

```
template <class E>
void OutputToConsole(myVector <E>& cinema)
{
	std::cout << std::endl;

		for (int i = 0; i < 6; i++) {
			std::cout << ' ';
		}
		std::cout << "   Название";
		for (int i = 0; i < 34; i++) {
			std::cout << ' ';
		}
		std::cout << "Режиссер";
		for (int i = 0; i < 21; i++) {
			std::cout << ' ';
		}
		std::cout << "Премьера";
		for (int i = 0; i < 14; i++) {
			std::cout << ' ';
		}
		std::cout <<  "Сценарист";
		for (int i = 0; i < 25; i++) {
			std::cout << ' ';
		}
		std::cout << "Жанр";
		for (int i = 0; i < 10; i++) {
			std::cout << ' ';
		}
		std::cout << "Продолжительность";
		std::cout << std::endl;
	std::cout << "-------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
-------------------------------------------------------------" << std::endl;

	for (int i = 0; i < cinema.getNumber(); i++)
	{
		std::cout << cinema[i];
	}
}


// поиск новейшего
Film getNewestFilm(myVector <Film> cinema)
{
	static Film temp;
	int size = cinema.getNumber();
	for (int i = 0; i < size; i++)
	{
		if (temp < cinema[i])
		{
			temp = cinema[i];
		}
	}
	return temp;
}


// поиск старейшего
Film getOldestFilm(myVector <Film> cinema){
	static Film temp;
	temp.setYear(10000);
	int size = cinema.getNumber();
	for (int i = 0; i < size; i++)
	{
```

```
			if (temp > cinema[i])
			{
				temp = cinema[i];
			}
		}
		return temp;
} //


// ввод массива из файла
void inputFromFile(myVector<Film>& cinema) {
		setlocale(LC_ALL, "RUS");
		std::ifstream fileIn;
		std::string nameOfInputFile;
		std::cout << "Write name of file to read films" << std::endl;
		std::cin >> nameOfInputFile;
		try {
			fileIn.open("Films.txt");
			if (!fileIn) {
				ERROR = 10;
				throw ERROR;
			}
		}
		catch (Exception error)
		{
			error.showException();
			fileIn.close();
			exit(-1);
		}
		try {
			while (!fileIn.eof()) {
				Film f;
				fileIn >> f;
				cinema.pushBack(f);
			}
		}
		catch (Exception error)
		{
			error.showException();
		}
		fileIn.close();
}



// сортировка массива по возрастанию
template <class C>
void SortFilms(myVector<C>& cinema) {
		int L = cinema.getNumber();
		for (int i = 0; i < L; i++) {
			for (int j = 0; j < i; j++) {
				if (cinema[i] > cinema[j]) SwapFilms(cinema[i], cinema[j]);
			}
		}
}


// составление списка по жанрам
template <class S>
void MakeAndOutListGenre(myVector<S>& cinema) {

		myString FALSE = "false";
		myVector<myString> dataOfGenre;  // жанры с мусором
```

```
	// сбор всех данных жанров
	for (int i = 0; i < cinema.getNumber(); i++) {
		Film temf = cinema[i];
		myString temp = temf.getGenre();
		dataOfGenre.pushBack(temp);
	}

	// обработка жанров
	for (int i = 0; i < dataOfGenre.getNumber(); i++) {
		for (int j = i + 1; j < dataOfGenre.getNumber(); j++) {
			if (dataOfGenre[i] == dataOfGenre[j]) {
				dataOfGenre[i] = FALSE;
			}
		}
	}

	myVector <myString> ListOfGenres; // жанры без мусора
	int X = 0; // кол-во СТОЛБЦОВ

	// запись неповторяющихся жанров
	for (int i = 0; i < dataOfGenre.getNumber(); i++) {
		if (!(dataOfGenre[i] == FALSE)) {
			X = X + 1;
			ListOfGenres.pushBack(dataOfGenre[i]);
		}
	}

	MATRIX <myString> list(cinema.getNumber(), X); // матрица где в строке название фильма, столбец - жанр

	// заполнение матрицы false
	for (int j = 0; j < cinema.getNumber(); j++) {
		for (int i = 0; i < ListOfGenres.getNumber(); i++) {
			list.SetMij(j, i, FALSE);
		}
	}

	// заполнение заголовков столбцов матрицы
	for (int i = 0; i < X; i++) {
		list.SetMij(0, i, ListOfGenres[i]);
	}

	// заполнение матрицы с точным попадание фильма в свой жанр, но с мусором
	for (int j = 0; j < cinema.getNumber(); j++) {
		Film temp = cinema[j];
		for (int i = 0; i < ListOfGenres.getNumber(); i++) {
			if (j != 0) {
				if (list.GetMij(0, i) == temp.getGenre()) {
					list.SetMij(j, i, temp.getTitle());
				}
			}
			else {
				if (list.GetMij(0, i) == temp.getGenre()) {
					list.SetMij(1, i, temp.getTitle());
				}
			}

		}


	}

	std::cout << std::endl << std::endl;
	// заполнение матрицы с точным попадание фильма в свой жанр
```

```
	bool flag = false;
	for (int j = 0; j < ListOfGenres.getNumber(); j++) {
		for (int i = 1; i < cinema.getNumber() ; i++) {
			if (list.GetMij(i, j) == FALSE ) {
				flag = false;
				for (int k = i; k < cinema.getNumber(); k++) {
					if ( (!(list.GetMij(k, j) == FALSE)) && (flag == false) )
{
						list.SetMij(i, j, list.GetMij(k, j));
						list.SetMij(k, j, FALSE);
						flag = true ;
					}
				}
			}
		}
	}

	list.Print();
}
```

Film.h

```
#ifndef FILM_H_INCLUDED
#define FILM_H_INCLUDED

#include "MyString.h"

class Film {
private:
	// поля

	myString title_;	// название фильма
	myString director_;	// фио режиссер
	int year_;		// год премьеры
	myString scriptwriter_; // фио сценариста
	myString genre_;	// жанр фильма
	int minutes_ ;		// минуты

public:

	Film();

	  // конструктор без параметров
	Film(myString title, myString director,int year, myString scriptwriter, myString genre, int minutes);		// конструткор с параметрами
	Film(const Film& ob);
		// конструктор копирования
	~Film();
		// деструктор

	// методы

	// перегрузщики
	bool operator== (const Film& movie);		// сравнивает фильмы по году
	Film& operator= (const Film& movie);		// присваение
	bool operator> (const Film& movie);		// сравнивает фильмы по году
	bool operator< (const Film& movie);		// сравнивает фильмы по году
	Film& operator++();		// увеличивает год фильма на один префикс
	Film& operator--();		// уменьшает год фильма на один префикс

	// chekkers -- проверка на правельный формат
```

```
        void checkTitle() const;
        void checkDirector() const;
        void checkYear() const;
        void checkScriptwriter() const;
        void checkGenre() const;
        void checkMinutes() const;

        // getters
        myString& getTitle();
        myString& getDirector();
        int& getYear();
        myString& getScriptwriter();
        myString& getGenre();
        int& getMinutes();

        // setters
        void setTitle(myString title);
        void setDirector(myString director);
        void setYear(int year);
        void setScriptwriter(myString scriptwriter);
        void setGenre(myString genre);
        void setMinutes(int minutes);


        // дружественные функции
        friend int operator+ (const Film& f1, const Film& f2);                                  // создает новый
фильм из двух
        friend bool operator> (const Film& f1, const Film& f2);                                     // оператор
отношений сравнения
        friend bool operator< (const Film& f1, const Film& f2);                                     //
оператор отношений сравнения
        friend Film operator++ (Film& movie, int);          // постфикс                             //
увеличивает год фильма на один постфикс
        friend Film operator-- (Film& movie, int);                  // уменьшает год фильма на один постфикс
        friend std::istream& operator>>(std::istream& in, Film &movie);                         // >> ввод
        friend std::ostream& operator<<(std::ostream& out, Film &movie);            // << вывод
};

#endif
```

Film.cpp

```
#include "Film.h"
#include "Exception.h"
#include "MyString.h"
#include <iostream>


Exception ERROR_SPELLING;


// конструкторы и деструкторы
Film::Film()
{
        title_ = "Прибытие Поезда На Вокзал Города Ла Сьота";
        director_ = "Луи Л. Ю.";
        year_ = 1895;
        scriptwriter_ = "Огюст Л. Ю.";
        genre_ = "документальныи";
        minutes_ = 100;
}
Film::Film(myString title, myString director, int year, myString scriptwriter, myString
genre, int minutes)
{
        title_ = title;
```

```
	director_ = director;
	year_ = year;
	scriptwriter_ = scriptwriter;
	genre_ = genre;
	minutes_ = minutes;
}
Film::Film(const Film& ob)
{
	this->title_ = ob.title_;
	this->director_ = ob.director_;
	this->year_ = ob.year_;
	this->scriptwriter_ = ob.scriptwriter_;
	this->genre_ = ob.genre_;
	this->minutes_ = ob.minutes_;
}
Film::~Film()
{
	//std::cout << "Destructor " << std::endl;
}

// перезагрузчики сравнения года
bool Film::operator==(const Film& movie)
{
	return this->year_ == movie.year_;
}
bool Film::operator>(const Film& movie)
{
	return this->year_ > movie.year_;
}
bool Film::operator<(const Film& movie)
{
	return this->year_ < movie.year_;
}

//префикс как метод
Film& Film::operator++()
{
	this->year_++;
	return *this;
}
Film& Film::operator--()
{

	this->year_--;
	return *this;
}

// постфиксы друж функция
Film operator++(Film& movie, int)
{
	Film temp = movie;
	movie.year_++;
	return temp;
}
Film operator--(Film& movie, int)
{
	Film temp = movie;
	movie.year_--;
	return temp;
}

// присваение
Film& Film::operator=(const Film& movie)
{
```

```
	this->title_ = movie.title_;
	this->director_ = movie.director_;
	this->year_ = movie.year_;
	this->scriptwriter_ = movie.scriptwriter_;
	this->genre_ = movie.genre_;
	this->minutes_ = movie.minutes_;
	return *this;
}

// проверки
void Film::checkTitle() const
{
	int flag = false;
	int L = title_.getLength();
	if (L < 2) {
		ERROR_SPELLING = 8;
		throw ERROR_SPELLING;
	}
	if ((int(title_[0]) < -64 || int(title_[0]) > -33) && (int(title_[0]) < 65 ||
int(title_[0]) > 90))  // Sm Ip
	{
		ERROR_SPELLING = 7;
		throw ERROR_SPELLING;
	}
	for (int i = 1; i < L; i++) {
		if (int(title_[i]) != 32) {
			if (!flag) {
				if ((int(title_[i]) < -32 || int(title_[i]) > -1) &&
(int(title_[i]) < 97 || int(title_[i]) > 122))
				{
					ERROR_SPELLING = 8;
					throw ERROR_SPELLING;

				}
			}
			else {
				flag = false;
				if ((int(title_[i]) < -64 || int(title_[i]) > -33) &&
(int(title_[i]) < 65 || int(title_[i]) > 90))
				{
					ERROR_SPELLING = 8;
					throw ERROR_SPELLING;
				}
			}
		}
		else {
			flag = true;
			if ((int(title_[i + 1]) < -64 || int(title_[i + 1]) > -33) &&
(int(title_[i + 1]) < 65 || int(title_[i + 1]) > 90))
			{
				ERROR_SPELLING = 7;
				throw ERROR_SPELLING;
			}
		}
	}
}
void Film::checkDirector() const
{
	int L = director_.getLength() - 6;
	int i;
	if (L<3){
		ERROR_SPELLING = 3;
		throw ERROR_SPELLING;
	}
```

```
	if ((int(director_[0]) < -64 || int(director_[0]) > -33) && (int(director_[0]) < 65
|| int(director_[0]) > 90)) {
		ERROR_SPELLING = 1;
		throw ERROR_SPELLING;
	}
	for (i = 1; i < L; i++) {

		if ((int(director_[i]) < -32 || int(director_[i]) > -1) && (int(director_[i])
< 97 || int(director_[i]) > 122)) {
			ERROR_SPELLING = 5;
			throw ERROR_SPELLING;
		}
	}
	if (int(director_[i])!=32 || int(director_[i+3]) != 32){
		ERROR_SPELLING = 2;
		throw ERROR_SPELLING;
	}
	i++;
	if ((int(director_[i]) < -64 || int(director_[i]) > -33) && (int(director_[i]) < 65
|| int(director_[i]) > 90)) {
		ERROR_SPELLING = 4;
		throw ERROR_SPELLING;
	}
	i++;
	if (int(director_[i]) != 46 || int(director_[i + 3]) != 46) {
		ERROR_SPELLING = 2;
		throw ERROR_SPELLING;
	}
}
void Film::checkYear() const
{
	if (this-> year_ < 1895) {
		ERROR_SPELLING = 9;
		throw ERROR_SPELLING;
	}
}
void Film::checkScriptwriter() const
{
	int L = scriptwriter_.getLength() - 6;
	int i;
	if (L < 2) {
		ERROR_SPELLING = 3;
		throw ERROR_SPELLING;
	}
	if ((int(scriptwriter_[0]) < -64 || int(scriptwriter_[0]) > -33) &&
(int(scriptwriter_[0]) < 65 || int(scriptwriter_[0]) > 90)) {
		ERROR_SPELLING = 1;
		throw ERROR_SPELLING;
	}
	for (i = 1; i < L; i++) {

		if ((int(scriptwriter_[i]) < -32 || int(scriptwriter_[i]) > -1) &&
(int(scriptwriter_[i]) < 97 || int(scriptwriter_[i]) > 122)) {
			ERROR_SPELLING = 5;
			throw ERROR_SPELLING;
		}
	}
	if (int(scriptwriter_[i]) != 32 || int(scriptwriter_[i + 3]) != 32) {
		ERROR_SPELLING = 2;
		throw ERROR_SPELLING;
	}
	i++;
	if ((int(scriptwriter_[0]) < -64 || int(scriptwriter_[0]) > -33) &&
(int(scriptwriter_[0]) < 65 || int(scriptwriter_[0]) > 90)) {
```

```
                ERROR_SPELLING = 4;
                throw ERROR_SPELLING;
        }
        i++;
        if (int(scriptwriter_[i]) != 46 || int(scriptwriter_[i + 3]) != 46) {
                ERROR_SPELLING = 2;
                throw ERROR_SPELLING;
        }
}
void Film::checkGenre() const
{
        int L = genre_.getLength();
        if (L < 4) {
                ERROR_SPELLING = 6;
                throw ERROR_SPELLING;
        }

        for (int i = 0; i < L; i++) {
                if ((int(genre_[i]) < -32 || int(genre_[i]) > -1) && (int(genre_[i]) < 97 ||
int(genre_[i]) > 122)) {
                        ERROR_SPELLING = 13;
                        throw ERROR_SPELLING;
                }
        }
}
void Film::checkMinutes() const
{
        if (!int(minutes_)) {
                ERROR_SPELLING = 19;
                throw ERROR_SPELLING;
        }
        if (this->minutes_ < 10 || this->minutes_ >999) {
                ERROR_SPELLING = 20;
                throw ERROR_SPELLING;
        }

}

// getters
myString& Film::getTitle()
{
        return title_;
}
myString& Film::getDirector()
{
        return director_;
}
int& Film::getYear()
{
        return year_;
}
myString& Film::getScriptwriter()
{
         return scriptwriter_;
}
myString& Film::getGenre()
{
        return genre_;
}
int& Film::getMinutes()
{
        return minutes_;
}
```

```
// setters
void Film::setTitle(myString title)
{
	this->title_ = title;
	this->checkTitle();
}
void Film::setDirector(myString director)
{
	this->director_ = director;
	this->checkDirector();
}
void Film::setYear(int year)
{
	this->year_ = year;
	this->checkYear();

}
void Film::setScriptwriter(myString scriptwriter)
{
	this->scriptwriter_ = scriptwriter;
	this->checkYear();
}
void Film::setGenre(myString genre)
{
	this->genre_ = genre;
	this->checkYear();
}
void Film::setMinutes(int minutes)
{
	this->minutes_ = minutes;
	this->checkMinutes();
}

int operator+(const Film& f1, const Film& f2)
{
	int time = 0;
	time = f1.minutes_ + f2.minutes_;
	return time;
}

bool operator>(const Film& f1, const Film& f2)
{
	return f1.year_ > f2.year_;
}
bool operator<(const Film& f1, const Film& f2)
{
	return f1.year_ < f2.year_;
}


std::istream& operator>>(std::istream& in, Film& movie)
{
	myString help1;
	myString help2;
	myString help3;
	myString help4;
	char h;
	bool flag = false;

	in.get(h);
	if (h == '\n') {
		flag = true;
		h = ' ';
	}
```

```
while (h != '\n') {
	if (flag) {
		in.get(h);
		flag = false;
	}
	else {
		help1.pushBack(h);
		in.get(h);
	}
}

movie.title_ = help1;
movie.checkTitle();

in.get(h);

if (h == '\n') {
	flag = true;
}
while (h != '\n') {
	if (flag) {
		in.get(h);
		flag = false;
	}
	else {
		help2.pushBack(h);
		in.get(h);
	}
}
movie.director_ = help2;
movie.checkDirector();

in >> movie.year_;
if (!int(movie.year_)) {
	ERROR_SPELLING = 14;
	throw ERROR_SPELLING;
}
movie.checkYear();

in.get(h);
if (h == '\n') {
	flag = true;
	h = ' ';
}

while (h != '\n') {
	if (flag) {
		in.get(h);
		flag = false;
	}
	else {
		help3.pushBack(h);
		in.get(h);
	}
}
movie.scriptwriter_ = help3;
movie.checkScriptwriter();

in >> movie.genre_;
movie.checkGenre();

in >> movie.minutes_;
movie.checkMinutes();
```

```
	in.get(h);
	return in;
}
std::ostream& operator<<(std::ostream& out, Film& movie)
{
	for (int i = 0; i < 3; i++) {
		out << ' ';
	}
	out << movie.title_;
	for (int i = 0; i < 43 - movie.title_.getLength(); i++) {
		out << ' ';
	}
	out << movie.director_;
	for (int i = 0; i < 34 - movie.director_.getLength(); i++) {
		out << ' ';
	}
	out << movie.year_;
	for (int i = 0; i < 18; i++) {
		out << ' ';
	}
	out << movie.scriptwriter_;
	for (int i = 0; i < 34 - movie.scriptwriter_.getLength(); i++) {
		out << ' ';
	}
	out << movie.genre_;
	for (int i = 0; i < 18 - movie.genre_.getLength(); i++) {
		out << ' ';
	}
	out << movie.minutes_ << "\n";

	return out;
}
```

myString.h

```
#ifndef MyString_H_INCLUDED
#define MyString_H_INCLUDED

#include <ostream>

class myString {
private:

	unsigned int size_;
	char* string_;

public:

	myString();  // конструктор без параметров
	myString(unsigned int size); // конструктор с параметром длина
	myString(const char* string); // конструктор копирования от char
	myString(const myString& object); // конструктор копирования этого же класса
	~myString(); // деструктор


	bool operator== (myString& secondWord); // оператор сравнения
	char& operator[] (const int index) const; // перезагрузка для индексов
	myString& operator= (const myString& variable); // перезагрузка для присваивания

	void pushBack(char& temp);
	int getLength() const; // метод получения длины


	myString& operator+ (const myString& anotherString) const;
```

```
	friend std::ostream& operator<<(std::ostream& out, const myString& variable); // вывод
	friend std::istream& operator>>(std::istream& in, myString& variable); // ввод


};

#endif
```

myString.cpp

```
#include "MyString.h"
#include "Exception.h"
#include <iostream>

Exception ERROR_IN_STRING;

myString::myString()
{
	string_ = new char[1];
	string_[0] = '\0';
	size_ = 0;
}
myString::myString(unsigned int size)
{
	if (size>99|| size <0) {
		ERROR_IN_STRING = 15;
		throw ERROR_IN_STRING;
	}
	string_ = new char[ size + 1];
	string_[size] = '\0';
	size_ = size;
}
myString::myString(const char* string)
{
	int size = 0;
	while (string[size] != '\0')
	{
		size++;
	}

	this->string_ = new char[size + 1];
	this->size_ = size;

	for (int i = 0; i < size; i++)
	{
		string_[i] = string[i];
	}

	this->string_[size] = '\0';
}
myString::myString(const myString& object)
{
	int size = object.getLength();
	this->string_ = new char[size + 1];
	this->size_ = object.size_;
	for (int i = 0; i < size + 1; i++)
	{
		this->string_[i] = object.string_[i];
	}
}
myString::~myString()
{
	delete[] this->string_;
```

```
}


void myString::pushBack(char& temp)
{

	char* variable = new char[this->size_ + 1];

	if (this->size_ > 97) {
		ERROR_IN_STRING = 16;
		throw ERROR_IN_STRING;
	}

	variable[this->size_] = temp;

	for (int i = 0; i < this->size_; i++)
	{
		variable[i] = this->string_[i];
	}
	if (this->string_ == nullptr) {

	}
	else {
		delete[] this->string_;
	}
	this->size_ = this->size_ + 2;
	this->string_ = new char[this->size_];

	for (int i = 0; i < size_ - 1; i++)
	{
		this->string_[i] = variable[i];
	}

	this->string_[size_ - 1] = '\0';
	this->size_--;

	delete[] variable;
}
int myString::getLength() const  //возвращает длину строки без \0
{
	return size_;
}

myString& myString::operator+(const myString& anotherString) const
{
		myString newString = new char[this->size_ + anotherString.size_];

		for (unsigned int i = 0; i < this->size_ + anotherString.size_; i++)
		{

			for (unsigned int j = 0; j < this->size_; i++, j++)
			{
				newString.string_[i] = this->string_[j];
			}

			for (unsigned int j = 0; j < anotherString.size_; i++, j++)
			{
				newString.string_[i] = anotherString.string_[j];
			}
		}

		newString.string_[this->size_ + anotherString.size_] = '\0';

		return newString;
	}
```

```
myString& myString::operator= (const myString& variable)
{
	if (this->string_ == nullptr) {
		delete[] this->string_;
	}
	if (variable.size_ > 99 || variable.size_ < 0) {
		ERROR_IN_STRING = 15;
		throw ERROR_IN_STRING;
	}
	this->string_ = new char[variable.size_ + 1];
	this->size_ = variable.size_;
	for (int i = 0; i < this->size_; i++)
	{
		this->string_[i] = variable.string_[i];
	}
	return *this;
}
bool myString::operator==(myString& secondWord)
{
	if (this->getLength() == secondWord.getLength())
	{
		for (int i = 0; i < this->getLength(); i++)
		{
			if ((*this)[i] != secondWord[i])
			{
				return false;
			}
		}
	}
	else
	{
		return false;
	}
	return true;
}


char& myString::operator[] (const int index) const
{
	if (index >= this->size_ || index < 0)
	{
		ERROR_IN_STRING = 16;
		throw ERROR_IN_STRING;
	}
return this->string_[index];
}

std::ostream& operator<<(std::ostream& out, const myString& variable)
{
	for (int i = 0; i < variable.size_; i++)
	{
		out << variable.string_[i];
	}
	return out;
}




std::istream& operator>>(std::istream& in, myString& variable)
{
	delete[] variable.string_;

	variable.size_ = 0;
```

```
	variable.string_ = new char[variable.size_];
	char temp = '\0';
	in.get(temp);
	int i = 0;
	if (temp == ' ')
	{
		in.get(temp);
	}
	while (temp != ' ' && temp != '\n')
		{
			variable.pushBack(temp);
			temp = '\n';
			in.get(temp);
	}
	return in;

}
```

myVector.h

```
#include <iostream>
#include "MyString.h"
#include "Exception.h"

template <class T>
class myVector
{
private:
	int number_;//количествово элементов
	T* data_;//указатель на массив
public:
	myVector();
	myVector(const int number);
	myVector(const myVector& variable);
	~myVector();

	void pushBack(const T& variable);

	T& operator[] (const int index);

	template<typename T>
	friend std::ostream& operator<< (std::ostream& out, myVector<T>& varible);

	template<typename T>
	friend std::istream& operator>> (std::istream& in, myVector<T>& variable);

	int getNumber()
	{
		return number_;
	}

};

Exception ERROR_IN_VECTOR;

template <class T>
class myVector;

template <class T> // конструткор без параметров
myVector<T>::myVector() {
	number_ = 0;
	data_ = nullptr;
}
```

```
template <class T> // конструткор с параемтром
myVector<T>::myVector(const int number)
{
	if (number <= 0)
	{
		ERROR_IN_VECTOR = 17;
		throw ERROR_IN_VECTOR;
	}
	this->number_ = number;
	data_ = new T[number]{ T() };
}

template <class T> // деструктор
myVector<T>::~myVector()
{
	delete[] this->data_;
}

// конструктор копирования
template <class T>
myVector<T>::myVector(const myVector& variable)
{
	int size = variable.number_;
	this->data_ = new T[size];
	for (int i = 0; i < size; i++)
	{
		this->data_[i] = variable.data_[i];
	}
	this->number_ = variable.number_;
}

// перегрузка
template <class T>
T& myVector<T>::operator[] (const int index)
{
	if (index < 0 || index >= number_)
	{
		ERROR_IN_VECTOR = 18;
		throw ERROR_IN_VECTOR;
	}
	return this->data_[index];
}

//метод
template <class T>
inline void myVector<T>::pushBack(const T& variable)
{
	T* temp = data_;
	this->number_++;
	data_ = new T[number_];
	for (int i = 0; i < number_ - 1; i++)
	{
		data_[i] = temp[i];
	}
	delete[] temp;
	data_[number_ - 1] = variable;
}

// вывод
template <class T>
std::ostream& operator<< (std::ostream& out, myVector<T>& variable)
{
	int  size = variable.number_;
	for (int i = 0; i < size; i++)
	{
```

```
                out << variable.data_[i] << "\t";
        }
        return out;
}

// ввод
template <class T>
std::istream& operator>> (std::istream& in, myVector<T>& variable)
{
        int  size = variable.number_;
        for (int i = 0; i < size; i++)
        {
                in >> variable.data_[i];
        }
        return in;
}
```

Matrix.h

```
#pragma once

#include <fstream>
#include <iostream>
#include "MyString.h"
#include "Exception.h"

Exception ERROR_IN_MATRIX;
// шаблонный класс Матрица для моего типа данных !!!!!!!!!!!!!!!!!!!!

template <typename T>
class MATRIX
{
private:
    T** M; // матрица
    int m_; // количество строк
    int n_; // количество столбцов

public:
    //конструктор
    MATRIX()
    {
        n_ = m_ = 0;
        M = nullptr; //
    }

    //конструктор с параметрами
    MATRIX(int m, int n)
    {
        if (m < 0 || n < 0) {
            ERROR_IN_MATRIX = 17;
            throw ERROR_IN_MATRIX;
        }
        m_ = m;
        n_ = n;

        // Выделить память для матрицы
        // Выделить память для массива указателей
        M = (T**) new T * [m_];

        // Выделить память для каждого указателя
        for (int i = 0; i < m_; i++)
            M[i] = (T*)new T[n_];
```

```
        // массив состоит из пробелов
        for (int i = 0; i < m_; i++)
            for (int j = 0; j < n_; j++)
                M[i][j] = " ";
    }

    //Конструктор копирования
    MATRIX(const MATRIX& _M)
    {
        m_ = _M.m_;
        n_ = _M.n_;

        M = (T**) new T * [m_];

        for (int i = 0; i < m_; i++)
            M[i] = (T*) new T[n_];

        for (int i = 0; i < m_; i++)
            for (int j = 0; j < n_; j++)
                M[i][j] = _M.M[i][j];
    }

    //getter
    T GetMij(int i, int j)
    {
        if ((m_ > -1) && (n_ > -1))
            return M[i][j];
        else
            return " ";
    }

    //метод выводящий длину самой длинной строки в массиве
    int GetMaxLength() {
        myString maxstring;
        myString l;
        myString r;
        for (int i = 0; i < m_-1; i++) {
            for (int j = 0; j < n_ ; j++) {
                l = GetMij(i, j);
                r = GetMij(i+1, j);
                if (l.getLength() > r.getLength() && l.getLength() > maxstring.getLength())
{
                    maxstring = l;
                }
                if (!(l.getLength() > r.getLength()) && r.getLength() >
maxstring.getLength()) {
                    maxstring = r;
                }
            }
        }
        return maxstring.getLength();
    }

    //вместо []
    void SetMij(int i, int j, T value)
    {
        if ((i < 0) || (i >= m_))
            return;
        if ((j < 0) || (j >= n_))
            return;
        M[i][j] = value;
    }
```

```
    //метод, выводящий матрицу
    void Print()
    {
        std::ofstream f("Output.txt");
        int N = this->GetMaxLength();
        myString probel;
        char p = ' ';
        for (int i = 0; i < N; i++) {
            probel.pushBack(p);
        }

        myString temp;
        myString str;
        myString FALSE = "false";

        int i = 0;
        int j = 0;
        f << "                                                                    LIST OF FILMS
GENRES                                                                  " << std::endl;
        f << std::endl;
        for (int i = 0; i < m_; i++)
        {
            for (int j = 0; j < n_; j++) {
                temp = M[i][j];
                if (i == 1 && j == 0) {
                    f << "-------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------" << std::endl;
                    f << "-------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------" << std::endl;
                }
                if (i == 0) {
                    if (j == 0) {
                        f << M[i][j];
                        str = M[i][j];
                    }
                    else {
                        for (int i = 0; i < N + 2 - str.getLength(); i++)
                        {
                            f << ' ';
                        }
                        f  << M[i][j];
                        str = M[i][j];
                    }
                }
                else {
                    if (temp == FALSE) {
                        if (j == 0) {
                            str = " ";
                            f << " ";
                        }
                        else {
                            for (int i = 0; i < N + 2 - str.getLength(); i++)
                            {
                                f << ' ';
                            }
                            f <<" ";
                            str = " ";
                        }

                    }
```

```
                else {
                    if (j == 0) {
                        f << M[i][j];
                        str = M[i][j];
                    }
                    else {
                        for (int i = 0; i < N + 2 - str.getLength(); i++)
                        {
                            f << ' ';
                        }
                        f << M[i][j];
                        str = M[i][j];
                    }
                }
            }
        }
        f << std::endl;
    }
    f.close();
}

//оператор присваивания
MATRIX operator=(const MATRIX& _M)
{
    if (n_ > 0)
    {
        // нужно освободить память, выделенную ранее для объекта *this
        for (int i = 0; i < m_; i++)
            delete[] M[i];
    }

    if (m_ > 0)
    {
        delete[] M;
    }

    m_ = _M.m;
    n_ = _M.n;
    // так как мы удалили прошлые массивы указателей M штук то создаем заново
    M = (T**) new T * [m_];
    for (int i = 0; i < m_; i++)
        M[i] = (T*) new T[n_];


    for (int i = 0; i < m_; i++)
        for (int j = 0; j < n_; j++)
            M[i][j]= _M.M[i][j];
    return *this;
}

//Деструктор
~MATRIX()
{
    // перестраховка
    if (n_ > 0)
    {
        for (int i = 0; i < m_; i++)
            delete[] M[i];
    }
    // перестраховка
    if (m_ > 0)
        delete[] M;
}
```

};**Exception.**h

```
#ifndef EXCEPTION_H_INCLUDED
#define EXCEPTION_H_INCLUDED
class Exception {
private:
	int indexError_;
public:
	void operator=(int temp);
	void showException();
};
#endif
```

Exception.cpp

```
#include "Exception.h"
#include <iostream>

void Exception::showException()
{
	switch (indexError_)
	{
	case 1:
		std::cout << "Surname is a big letter!\n";
		break;
	case 2:
		std::cout << "Forgot '.' or ' ' in initials!\n";
		break;
	case 3:
		std::cout << "Such short Name. It is impossible!\n";
		break;
	case 4:
		std::cout << "initials are big letters!\n";
		break;
	case 5:
		std::cout << "Incorrect fullname. Error in Register!\n";
		break;
	case 6:
		std::cout << "Incorrect ganre. More 2 letters!\n";
		break;
	case 7:
		std::cout << "Every letter of title is a big letter!\n";
		break;
	case 8:
		std::cout << "Error in the title!\n";
		break;
	case 9:
		std::cout << "A year of film premiere could not be ealier then 1895!\n";
		break;
	case 10:
		std::cout << "File could not opened!\n";
		break;
	case 11:
		std::cout << "So long String!\n";
		break;
	case 12:
		std::cout << "Out of bounds of string array!\n";
		break;
	case 13:
		std::cout << "Error!in genre illegal symbols or uppercase\n";
		break;
	case 14:
		std::cout << "Yaer is a number!\n";
		break;
```

```
		case 15:
			std::cout << "String is not more then 100 symbols\n!";
			break;
		case 16:
			std::cout << "Выход за границу массива строки класса май стринг\n!";
			break;
		case 17:
			std::cout << "Массив состоит из более чем 0 элементов \n!";
			break;
		case 18:
			std::cout << "Выход за границу массива при вызове[] \n!";
			break;
		case 19:
			std::cout << " Минуты - целое число \n!";
			break;
		case 20:
			std::cout << " Длительность в фильме не может быть больше 1000 минут и меньше 10 минут \n!";
		}
}
void Exception::operator=(int temp)
{
		indexError_ = temp;
}
```