Министерство образования Республики Беларусь

Учреждения образования

“Брестский государственный технический университет”

Кафедра интеллектуально-информационных технологий

ОТЧЁТ

По лабораторной работе №4

**Функции-друзья классов. Перегрузка операторов классов**

Выполнил:

студент 2 курса

группы ИИ-23

Макаревич Н. Р.

Проверил:

Монтик Н. С.

Брест-2023

ЦЕЛЬ РАБОТЫ

1. Изучение использования friend-функций для доступа к классам извне.

2. Изучение особенностей перегрузки операторов в пользовательских классах.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить правила использования friend-функций.

2. Изучить правила и способы перегрузки операторов.

3. Реализовать в индивидуальных заданиях по разработке классов перегрузку операторов обоими способами.

**КОД ПРОГРАММЫ:**

#include <iostream>

#include <vector>

using namespace std;

class Arif {

private:

int a;

int b;

int c;

public:

Arif() { }

Arif(const Arif& temp) {

a = temp.a;

b = temp.b;

c = temp.c;

}

Arif(int a, int b, int c) {

this->a = a;

this->b = b;

this->c = c;

}

int getA() {

return a;

}

int getB() {

return b;

}

int getC() {

return c;

}

void setA(int x) {

a = x;

}

void setB(int x) {

b = x;

}

void setC(int x) {

c = x;

}

void printAll() {

cout << a << " " << b << " " << c << endl;

}

Arif operator + (const Arif& temp) {

return Arif{

a + temp.a,

b + temp.b,

c + temp.c

};

}

Arif operator = (const Arif& temp) {

a = temp.a;

b = temp.b;

c = temp.c;

return \*this;

}

Arif operator ++ () {

++a;

++b;

++c;

return \*this;

}

Arif operator ++(int) {

++a;

++b;

++c;

return \*this;

}

void AddUp(int i) {

a += i;

}

void AddUp(int i, int j) {

a += i;

b += j;

}

void AddUp(int i, int j, int k) {

a += i;

b += j;

c += k;

}

};

class Film {

private:

string name;

vector <string> actors;

double length;

friend class Season;

public:

Film() { }

Film(string name, vector <string> actors, int length) {

this->name = name;

this->length = length;

this->actors = actors;

}

Film(const Film& temp) {

this->name = temp.name;

this->length = temp.length;

this->actors = temp.actors;

}

void setLength(double temp) {

length = temp;

}

void setName(string temp) {

name = temp;

}

void setActors(vector <string> temp) {

actors = temp;

}

double getLength() {

return length;

}

string getName() {

return name;

}

friend void showAll(const Film& film);

friend vector <string> getActors(Film& film);

Film operator = (const Film& temp) {

name = temp.name;

length = temp.length;

actors = temp.actors;

return \*this;

}

Film operator + (const Film& temp) {

length += temp.length;

return \*this;

}

Film operator --() {

--length;

return \*this;

}

void AddActors(vector <string> newActors) {

for (int i = 0; i < newActors.size(); i++) {

actors.push\_back(newActors[i]);

}

}

void AddActors(string\* newActors, int size) {

for (int i = 0; i < size; i++) {

actors.push\_back(newActors[i]);

}

}

void AddActors(string actor) {

actors.push\_back(actor);

}

};

class Season {

private:

Film\* series;

string name;

int size;

string& nameLink = name;

void seriesInitialization(Film\* tempSeries) {

if (size > 0) {

series = new Film[size];

for (int i = 0; i < size; i++)

series[i] = tempSeries[i];

}

else cout << "ERROR: size is unknown" << endl;

}

public:

Season() {

name = "UNNAMED";

size = 0;

}

Season(string tempName, int tempSize, Film\* tempSeries) {

name = tempName;

size = tempSize;

seriesInitialization(tempSeries);

}

Season(Season& temp) {

name = temp.name;

size = temp.size;

seriesInitialization(temp.series);

}

string getEpName(int num) {

if (num >= size)

return "ERROR";

return series[num].getName();

}

Film getEpizode(int num) { return series[num]; }

void copySeries(Film\*& temp, int& tempSize) {

for (int i = 0; i < size; i++) {

temp[i] = series[i];

}

if (tempSize > size)

tempSize = size;

}

void addEpizode(Film newEp) {

Film\* temp = new Film[size];

copySeries(temp, size);

size++;

series = new Film[size];

for (int i = 0; i < size - 1; i++)

series[i] = temp[i];

series[size - 1] = newEp;

}

};

void showAll(const Film& film) {

cout << film.name << " " << film.length << endl;

for (int i = 0; i < film.actors.size(); i++)

cout << film.actors[i] << " ";

cout << endl;

}

vector <string> getActors(Film& film) {

return film.actors;

}

int main()

{

Arif nums(1, 1, 1);

nums.printAll();

nums++; //перегрузка ++

nums.printAll();

++nums;

nums.printAll();

cout << endl;

vector <string> temp1 = { "1", "2" };

Film newSeason[2];

newSeason[0] = Film("Batman", temp1, 1.23);

newSeason[1] = Film("Spider Man", temp1, 2.03);

Season\* season1 = new Season("Introduction", 2, newSeason);

Film epizode3 = newSeason[0];

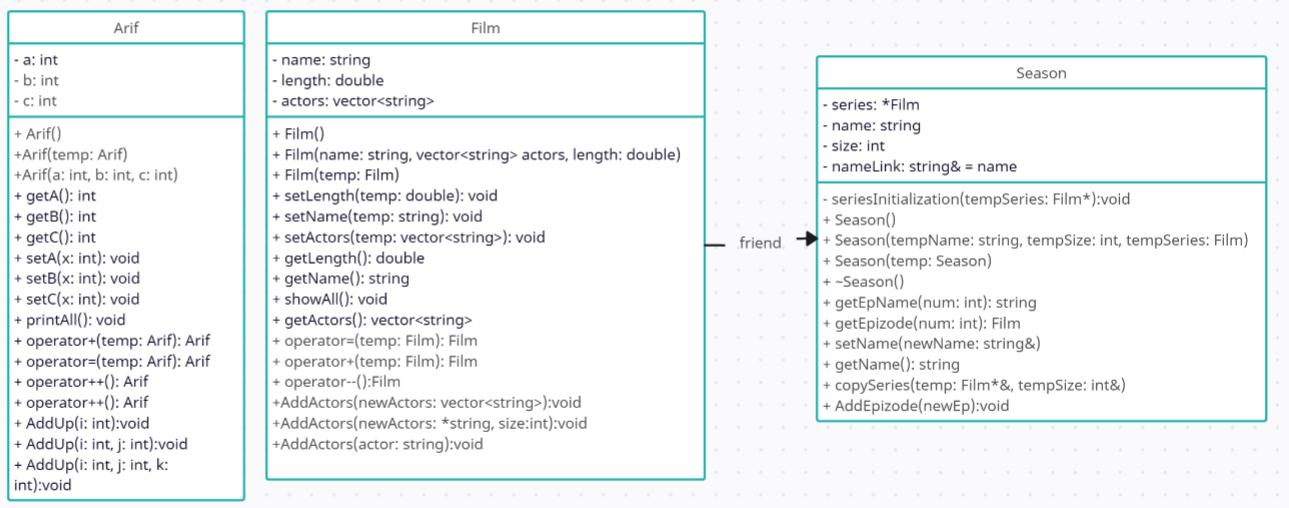
epizode3.AddActors("3");

epizode3.setName("Hulk");

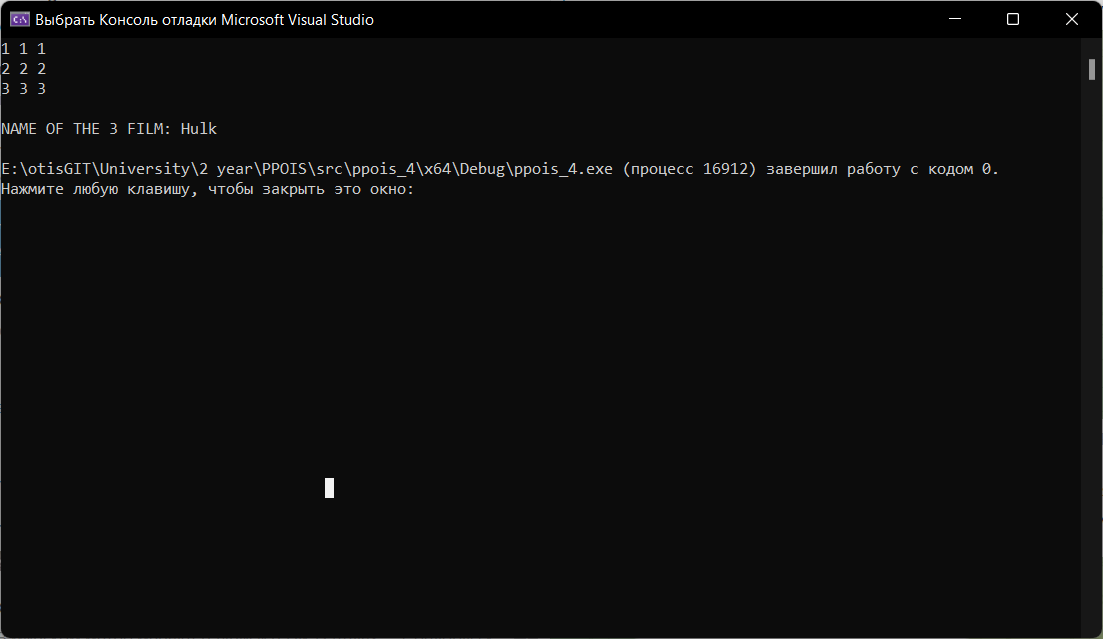
season1->addEpizode(epizode3);

cout << "NAME OF THE 3 FILM: " << season1->getEpName(2) << endl; //friend метод

return 0;

}****

Вывод:



Изучил использования friend-функций для доступа к классам извне.