|  |  |
| --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования «Полоцкий государственный университет» | |
|  | Факультет информационных технологий  Кафедра технологий программирования |
| Лабораторная работа №5 по курсу «Объектно-ориентированные технологии программирования и стандарты проектирования»  «Перегрузка операторов»  Вариант 6 | |
| Выполнил | Студент гр. 21-ИТ-1  Катушёнок И.В. |
| Проверил | Ассистент  Хирьянов И.Д. |
| Полоцк, 2023г. | |

**Цель работы:** Научиться перегружать бинарные и унарные операторы для пользовательского класса, применять перегруженные операторы для операций над объектами классов.

**Ход работы**

**Задание:**

1. Дополнить и при необходимости модифицировать приложение, разработанное согласно варианта лабораторной работы №4.

2. Описать в основном классе оператор + для добавления объекта одного из классов-наследников во внутренний массив.

3. Описать в основном классе операторы ++ префиксный и постфиксный для добавления объектов по умолчанию в массив (по умолчанию считаем 0-й объект в массиве).

4. Описать в основном классе оператор [] для доступа к элементу массива по индексу.

5. Описать глобальный оператор << для вывода данных на экран.

6. Описать оператор =.

7. В классах-наследниках описать операторы сравнения , ==. Сравнение производить по любому атрибуту базового класса или самого класса.

8. В основном теле программы дополнить основное меню, чтобы можно было продемонстрировать применение всех операторов, включая сравнение пар объектов классов-наследников.

9. Реализовать все необходимые проверки на ввод значений.

10.Сделать выводы

**Вариант**

Варианты распределяются аналогично предыдущим лабораторным работам.

Листинг 1.1 – Вспомогательный класс для проверки корректности ввода

class CorrectInput

{

public static bool IsOnlyLetters(string str)

{

if (str == "") return false;

if (str == " ") return false;

foreach (char c in str)

{

if (Char.IsNumber(c)) return false;

if (c == '-') return false;

if (c == '.') return false;

if (c == ',') return false;

if (c == ';') return false;

if (c == '/') return false;

if (c == ':') return false;

if (c == '@') return false;

if (c == '#') return false;

if (c == '$') return false;

if (c == '%') return false;

if (c == '^') return false;

if (c == '&') return false;

if (c == '\*') return false;

}

return true;

}

public static bool InRange(int leftRange, int rightRange, int number)

{

if (number >= leftRange && number <= rightRange) return true;

return false;

}

}

Листинг 1.2 – реализация метода Main

class Program

{

static void Main()

{

Catalog catalog = new Catalog();

bool MenuCycle = true;

while (MenuCycle)

{

Console.WriteLine("\tSelect operation\n" +

"[1] - add\n" +

"[2] - info all\n" +

"[3] - delete\n" +

"[4] - edit\n" +

"[5] - search film\n" +

"[6] - sort\n" +

"[7] - operators");

int.TryParse(Console.ReadLine(), out int choise);

switch (choise)

{

case 1:

Console.WriteLine("\tSelect film to add\n" +

"[1] - favorite\n" +

"[2] - blocked");

if(int.TryParse(Console.ReadLine(), out int filmChoise))

switch(filmChoise)

{

case 1:

catalog.AddFavoriteFilm();

break;

case 2:

catalog.AddBlockedFilm();

break;

}

break;

case 2:

if (catalog.isEmpty())

{

Console.WriteLine("Catalog empty");

break;

}

catalog.SeeAll();

break;

case 3:

catalog.RemoveFromCatalog();

break;

case 4:

catalog.EditFilm();

break;

case 5:

catalog.SearchFilm();

break;

case 6:

catalog.SortFilms();

break;

case 7:

catalog.UseOperators(catalog);

break;

case -1:

MenuCycle = false;

break;

}

Console.WriteLine("========================");

}

Console.ReadKey();

}

}

Листинг 1.3 – реализация основного класса:

class Catalog

{

private List<Film> filmList = new List<Film>();

public int FilmsCount

=> filmList.Count;

public static Catalog operator++ (Catalog catalog)

{

if (catalog.filmList.Count != 0)

catalog.filmList.Add(catalog.filmList[0]);

else

Console.WriteLine("operator ++ cannot be use, list is empty");

return catalog;

}

public static Catalog operator +(Catalog catalog, Film film)

{

catalog.filmList.Add(film);

return catalog;

}

public void UseOperators(Catalog catalog)

{

Console.WriteLine("\tSelect operation\n" +

"[1] - + (add to film list)\n" +

"[2] - ++ (add to film list film with id 0)\n" +

"[3] - == (compare by film's parametrs)\n" +

"[4] - < (compare by film's rating)\n" +

"[5] - > (compare by film's rating)\n" +

"[6] - != (compare by film's parametrs)\n" +

"[7] - >= (compare by film's rating)\n" +

"[8] - <= (compare by film's rating)\n" +

"[9] - all compares");

int.TryParse(Console.ReadLine(), out int select);

switch (select)

{

case 1:

Console.WriteLine("\tWhat add\n" +

"[1] - new favorite film\n" +

"[2] - new blocked film\n" +

"[3] - film from film list");

int.TryParse(Console.ReadLine(), out int select2);

switch (select2)

{

case 1:

catalog += new FavoriteFilm();

break;

case 2:

catalog += new BlockedFilm();

break;

case 3:

if (filmList.Count == 0)

{

Console.WriteLine("Add film from film list cannot be use, film list is empty");

break;

}

Console.WriteLine("Enter film id");

int.TryParse(Console.ReadLine(), out int filmId);

if (CorrectInput.InRange(0, filmList.Count - 1, filmId)) catalog += filmList[filmId];

else Console.WriteLine("Incorrect film number");

break;

}

break;

case 2:

catalog++;

break;

case 3:

Console.Write("Enter film fisrt id : ");

int.TryParse(Console.ReadLine(), out int firstId);

Console.Write("Enter film second id : ");

int.TryParse(Console.ReadLine(), out int secondId);

if (CorrectInput.InRange(0, filmList.Count - 1, firstId) && CorrectInput.InRange(0, filmList.Count - 1, secondId))

{

if (filmList[firstId] == filmList[secondId])

Console.WriteLine("Films are equal");

else

Console.WriteLine("Films are different");

}

else

Console.WriteLine("Incorrect film number");

break;

case 4:

Console.Write("Enter film fisrt id : ");

int.TryParse(Console.ReadLine(), out firstId);

Console.Write("Enter film second id : ");

int.TryParse(Console.ReadLine(), out secondId);

if (CorrectInput.InRange(0, filmList.Count - 1, firstId) && CorrectInput.InRange(0, filmList.Count - 1, secondId))

{

if (filmList[firstId] < filmList[secondId])

Console.WriteLine("First film raiting less than second");

else

Console.WriteLine("First film raiting more than second");

}

else

Console.WriteLine("Incorrect film number");

break;

case 5:

Console.WriteLine("Enter film fisrt id");

int.TryParse(Console.ReadLine(), out firstId);

Console.WriteLine("Enter film second id");

int.TryParse(Console.ReadLine(), out secondId);

if (CorrectInput.InRange(0, filmList.Count - 1, firstId) && CorrectInput.InRange(0, filmList.Count - 1, secondId))

{

if (filmList[firstId] > filmList[secondId])

Console.WriteLine("First film raiting more than second");

else

Console.WriteLine("First film raiting less than second");

}

else

Console.WriteLine("Incorrect film numebr");

break;

case 6:

Console.WriteLine("Enter film fisrt id");

int.TryParse(Console.ReadLine(), out firstId);

Console.WriteLine("Enter film second id");

int.TryParse(Console.ReadLine(), out secondId);

if (CorrectInput.InRange(0, filmList.Count - 1, firstId) && CorrectInput.InRange(0, filmList.Count - 1, secondId))

{

if (filmList[firstId] != filmList[secondId]) Console.WriteLine("Films are different");

else Console.WriteLine("Films are equal");

}

else Console.WriteLine("Incorrect film number");

break;

case 7:

Console.WriteLine("Enter film fisrt id");

int.TryParse(Console.ReadLine(), out firstId);

Console.WriteLine("Enter film second id");

int.TryParse(Console.ReadLine(), out secondId);

if (CorrectInput.InRange(0, filmList.Count - 1, firstId) && CorrectInput.InRange(0, filmList.Count - 1, secondId))

{

if (filmList[firstId] >= filmList[secondId])

Console.WriteLine("First film raiting more than second or equal");

else

Console.WriteLine("First film raiting less than second");

}

else

Console.WriteLine("Incorrect film numebr");

break;

case 8:

Console.WriteLine("Enter film fisrt id");

int.TryParse(Console.ReadLine(), out firstId);

Console.WriteLine("Enter film second id");

int.TryParse(Console.ReadLine(), out secondId);

if (CorrectInput.InRange(0, filmList.Count - 1, firstId) && CorrectInput.InRange(0, filmList.Count - 1, secondId))

{

if (filmList[firstId] <= filmList[secondId])

Console.WriteLine("First film raiting less than second or equal");

else

Console.WriteLine("First film raiting more than second");

}

else

Console.WriteLine("Incorrect film number");

break;

case 9:

Console.Write("Enter film fisrt id : ");

int.TryParse(Console.ReadLine(), out firstId);

Console.Write("Enter film second id : ");

int.TryParse(Console.ReadLine(), out secondId);

if (CorrectInput.InRange(0, filmList.Count - 1, firstId) && CorrectInput.InRange(0, filmList.Count - 1, secondId))

{

Console.Write("< - ");

if (filmList[firstId] < filmList[secondId])

Console.WriteLine("True");

else

Console.WriteLine("False");

Console.Write("> - ");

if (filmList[firstId] > filmList[secondId])

Console.WriteLine("True");

else

Console.WriteLine("False");

Console.Write("== - ");

if (filmList[firstId] == filmList[secondId])

Console.WriteLine("True");

else

Console.WriteLine("False");

Console.Write("!= - ");

if (filmList[firstId] != filmList[secondId])

Console.WriteLine("True");

else

Console.WriteLine("False");

Console.Write("<= - ");

if (filmList[firstId] <= filmList[secondId])

Console.WriteLine("True");

else

Console.WriteLine("False");

Console.Write(">= - ");

if (filmList[firstId] >= filmList[secondId])

Console.WriteLine("True");

else

Console.WriteLine("False");

}

break;

}

}

public void AddBlockedFilm()

{

Console.WriteLine("\tWhich constructor use?\n" +

"[1 - deafult\n" +

"[2] - with parametrs\n" +

"[3] - copy");

if (int.TryParse(Console.ReadLine(), out int filmChoise))

switch (filmChoise)

{

case 1:

filmList.Add(new BlockedFilm());

break;

case 2:

filmList.Add(new BlockedFilm(Film.EnterName(), Film.EnterYear(), Film.EnterType(), Film.EnterRate(), Film.EnterCountry(), BlockedFilm.EnterDescription()));

break;

case 3:

if (isEmpty())

{

Console.WriteLine("catalog empty");

break;

}

Console.WriteLine("Enter film number");

int.TryParse(Console.ReadLine(), out int filmPosition);

if (CorrectInput.InRange(0, filmList.Count - 1, filmPosition))

{

if (filmList[filmPosition] is BlockedFilm)

{

filmList.Add(new BlockedFilm((BlockedFilm)filmList[filmPosition]));

}

else filmList.Add(new FavoriteFilm((FavoriteFilm)filmList[filmPosition]));

}

else

{

Console.WriteLine("Incorrect film number");

break;

}

break;

}

}

public void AddFavoriteFilm()

{

Console.WriteLine("\tWhich constructor use?\n" +

"[1 - deafult\n[2] - with parametrs\n[3] - copy");

if (int.TryParse(Console.ReadLine(), out int filmChoise))

switch (filmChoise)

{

case 1:

filmList.Add(new FavoriteFilm());

break;

case 2:

filmList.Add(new FavoriteFilm(Film.EnterName(), Film.EnterYear(), Film.EnterType(), Film.EnterRate(), Film.EnterCountry(), FavoriteFilm.EnterDescription()));

break;

case 3:

if (filmList.Count == 0)

{

Console.WriteLine("catalog empty");

break;

}

Console.WriteLine("Enter film number");

int.TryParse(Console.ReadLine(), out int filmPosition);

if (CorrectInput.InRange(0, filmList.Count - 1, filmPosition))

{

if (filmList[filmPosition] is FavoriteFilm)

{

filmList.Add(new FavoriteFilm((FavoriteFilm)filmList[filmPosition]));

}

else filmList.Add(new BlockedFilm((BlockedFilm)filmList[filmPosition]));

}

else

{

Console.WriteLine("Incorrect film number");

break;

}

break;

}

}

public void EditFilm()

{

if (filmList.Count == 0)

{

Console.WriteLine("catalog empty");

return;

}

Console.WriteLine("Enter film number");

int.TryParse(Console.ReadLine(), out int filmPosition);

if (CorrectInput.InRange(0, filmList.Count - 1, filmPosition)) filmList[filmPosition].Edit();

else Console.WriteLine("Incorrect input");

}

public void RemoveFromCatalog()

{

if (isEmpty())

{

Console.WriteLine("Catalog empty");

return;

}

Console.Write("Enter film number: ");

int.TryParse(Console.ReadLine(), out int position);

if (CorrectInput.InRange(0, filmList.Count - 1, position))

{

filmList.RemoveAt(position);

GC.Collect();

}

else

{

Console.WriteLine("Film with this number not found");

return;

}

}

public void GetInfoFromPosition(int filmPosition) { filmList[filmPosition].SeeInfo(); }

public void SearchFilm()

{

if (isEmpty())

{

Console.Write("Catalog empty");

return;

}

Console.WriteLine("How you want to search?");

Console.WriteLine("[1] - Name\n[2] - Year\n[3] - Type\n[4] - Rating\n[5] - Country.");

int.TryParse(Console.ReadLine(), out int choise);

switch (choise)

{

case 1:

SearchName();

break;

case 2:

SearchYear();

break;

case 3:

SearchType();

break;

case 4:

SeatchRate();

break;

case 5:

SearchCounrty();

break;

}

}

public void SearchCounrty()

{

Console.WriteLine("Enter country");

string Country = Console.ReadLine();

for (int i = 0; i < filmList.Count; i++)

{

if (filmList[i].GetCountry() == Country)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

}

}

}

public void SearchType()

{

Console.WriteLine("Enter type");

string Type = Console.ReadLine();

for (int i = 0; i < filmList.Count; i++)

{

if (filmList[i].GetType() == Type)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

}

}

}

public void SearchName()

{

Console.WriteLine("Enter name");

string Name = Console.ReadLine();

for (int i = 0; i < filmList.Count; i++)

{

if (filmList[i].GetName().IndexOf(Name) != -1)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

}

}

}

public void SearchYear()

{

Console.WriteLine("Enter year");

int.TryParse(Console.ReadLine(), out int year);

Console.WriteLine("[1] - >\n[2] - <");

int.TryParse(Console.ReadLine(), out int choise);

for (int i = 0; i < filmList.Count; i++)

{

if (filmList[i].GetYear() > year && choise == 1)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

}

if (filmList[i].GetYear() < year && choise == 2)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

}

}

}

public void SeatchRate()

{

Console.WriteLine("Enter search rating(0-100)");

int rate;

while(true)

{

int.TryParse(Console.ReadLine(), out rate);

if (rate < 100 && rate > 0) break;

Console.WriteLine("Incorrect input, try again");

}

Console.WriteLine("[1] - <\n[2] - >");

int chose = Convert.ToInt32(Console.ReadLine());

foreach (Film film in filmList)

{

if(film.GetRate() > rate && chose == 1) film.SeeInfo();

if(film.GetRate() > rate && chose == 2) film.SeeInfo();

}

}

public void SeeAll()

{

Console.WriteLine("========================");

for (int i = 0; i < filmList.Count; i++)

{

Console.Write(i + ".");

filmList[i].SeeInfo();

Console.WriteLine(" ");

}

}

public bool isEmpty()

{

if (filmList.Count == 0) return true;

else return false;

}

public void SortFilms()

{

int choise = 0;

Console.WriteLine("How you want to sort?");

Console.WriteLine("[1] - Year\n[2] - Rating.");

try

{

choise = Convert.ToInt32(Console.ReadLine());

}

catch

{

Console.WriteLine("Incorrect input");

return;

}

switch (choise)

{

case 1:

for (int i = 0; i < filmList.Count; i++)

for (int j = 0; j < filmList.Count; j++)

{

if (filmList[i].GetYear() < filmList[j].GetYear())

{

Film temp = filmList[i];

filmList[i] = filmList[j];

filmList[j] = temp;

}

}

break;

case 2:

for (int i = 0; i < filmList.Count; i++)

for (int j = 0; j < filmList.Count; j++)

{

if (filmList[i].GetRate() < filmList[j].GetRate())

{

Film temp = filmList[i];

filmList[i] = filmList[j];

filmList[j] = temp;

}

}

break;

}

}

}

Листинг 1.4 – реализация дополнительного класса:

abstract class Film

{

protected string \_name;

protected int \_year;

protected int \_rate;

protected string \_type;

protected string \_country;

public Film()

{

Console.WriteLine("Constructor without parametrs was called");

SetName(EnterName());

SetYear(EnterYear());

SetType(EnterType());

SetRate(EnterRate());

SetCountry(EnterCountry());

}

public Film(string name, int year, string type, int rate, string country)

{

Console.WriteLine("Constructor with parametrs was called");

\_name = name;

\_year = year;

\_type = type;

\_rate = rate;

\_country = country;

}

public Film(Film film)

{

\_name = film.\_name;

\_year = film.\_year;

\_type = film.\_type;

\_rate = film.\_rate;

\_country = film.\_country;

Console.WriteLine("Copy constructor was called");

}

~Film()

{

Console.WriteLine("Destructor was called");

}

public static bool operator ==(Film film1, Film film2)

{

if (film1.GetName() == film2.GetName() &&

film1.GetType() == film2.GetType() &&

film1.GetYear() == film2.GetYear() &&

film1.GetCountry() == film2.GetCountry() &&

film1.GetRate() == film2.GetRate())

return true;

else

return false;

}

public static bool operator !=(Film film1, Film film2)

{

if (film1.GetName() == film2.GetName() &&

film1.GetType() == film2.GetType() &&

film1.GetYear() == film2.GetYear() &&

film1.GetCountry() == film2.GetCountry() &&

film1.GetRate() == film2.GetRate())

return false;

else

return true;

}

public static bool operator >(Film film1, Film film2)

{

if (film1.GetRate() > film2.GetRate())

return true;

else

return false;

}

public static bool operator <(Film film1, Film film2)

{

if (film1.GetRate() < film2.GetRate()) return true;

else return false;

}

public static bool operator >=(Film film1, Film film2)

{

if (film1.GetRate() >= film2.GetRate())

return true;

else

return false;

}

public static bool operator <=(Film film1, Film film2)

{

if (film1.GetRate() <= film2.GetRate())

return true;

else

return false;

}

public string GetName() => \_name;

public int GetYear() => \_year;

public int GetRate() => \_rate;

public string GetType() => \_type;

public string GetCountry() => \_country;

public void SetName(string name) { \_name = name; }

public void SetYear(int year) { \_year = year; }

public void SetType(string type) {\_type = type; }

public void SetRate(int rate) { \_rate = rate; }

public void SetCountry(string country) { \_country = country; }

public static string EnterName()

{

Console.Write("Enter film name: ");

while(true)

{

string name = Console.ReadLine();

if (name != "")

{

return name;

}

}

}

public static int EnterYear()

{

while (true)

{

Console.Write("Enter film year: ");

int.TryParse(Console.ReadLine(), out int year);

if (CorrectInput.InRange(1927, 2023, year))

{

return year;

}

Console.WriteLine("Incorrect input, try again");

}

}

public static string EnterType()

{

while (true)

{

Console.Write("Enter film type: ");

string type = Console.ReadLine();

if (CorrectInput.IsOnlyLetters(type))

{

return type;

}

Console.WriteLine("Incorrect input, try again");

}

}

public static int EnterRate()

{

while (true)

{

Console.Write("Enter film rating(0-100): ");

bool check = int.TryParse(Console.ReadLine(), out int rate);

if (CorrectInput.InRange(0, 100, rate) && check)

{

return rate;

}

Console.WriteLine("Incorrect input, try again");

}

}

public static string EnterCountry()

{

while (true)

{

Console.Write("Enter film country: ");

string country = Console.ReadLine();

if (CorrectInput.IsOnlyLetters(country))

{

return country;

}

Console.WriteLine("Incorrect input, try again");

}

}

public void Edit()

{

Console.WriteLine("What do you want to change?");

Console.WriteLine("[1] - Name\n" +

"[2] - Year\n" +

"[3] - Type\n" +

"[4] - Rating\n" +

"[5] - Country.");

int edit = Convert.ToInt32(Console.ReadLine());

switch (edit)

{

case 1:

SetName(EnterName());

break;

case 2:

SetYear(EnterYear());

break;

case 3:

SetType(EnterType());

break;

case 4:

SetRate(EnterRate());

break;

case 5:

SetCountry(EnterCountry());

break;

}

}

public abstract void SeeInfo();

}

}

Листинг 1.5 – Перегрузка оператора +

public static Catalog operator +(Catalog catalog, Film film)

{

catalog.filmList.Add(film);

return catalog;

}

Листинг 1.6 – Перегрузка оператора ++

public static Catalog operator++ (Catalog catalog)

{

if (catalog.filmList.Count != 0)

catalog.filmList.Add(catalog.filmList[0]);

else

Console.WriteLine("operator ++ cannot be use, list is empty");

return catalog;

}

Листинг 1.7 – Перегрузка операторов >,<,==,!=,>=,<=

public static bool operator ==(Film film1, Film film2)

{

if (film1.GetName() == film2.GetName() &&

film1.GetType() == film2.GetType() &&

film1.GetYear() == film2.GetYear() &&

film1.GetCountry() == film2.GetCountry() &&

film1.GetRate() == film2.GetRate())

return true;

else

return false;

}

public static bool operator !=(Film film1, Film film2)

{

if (film1.GetName() == film2.GetName() &&

film1.GetType() == film2.GetType() &&

film1.GetYear() == film2.GetYear() &&

film1.GetCountry() == film2.GetCountry() &&

film1.GetRate() == film2.GetRate())

return false;

else

return true;

}

public static bool operator >(Film film1, Film film2)

{

if (film1.GetRate() > film2.GetRate())

return true;

else

return false;

}

public static bool operator <(Film film1, Film film2)

{

if (film1.GetRate() < film2.GetRate()) return true;

else return false;

}

public static bool operator >=(Film film1, Film film2)

{

if (film1.GetRate() >= film2.GetRate())

return true;

else

return false;

}

public static bool operator <=(Film film1, Film film2)

{

if (film1.GetRate() <= film2.GetRate())

return true;

else

return false;

}

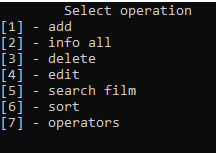


Рисунок 1. Начальное меню

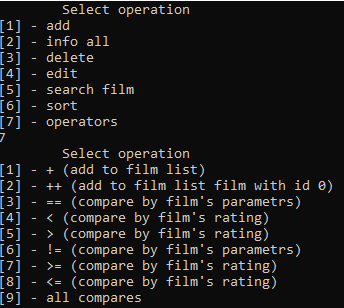


Рисунок 2. Меню перегруженных операторов

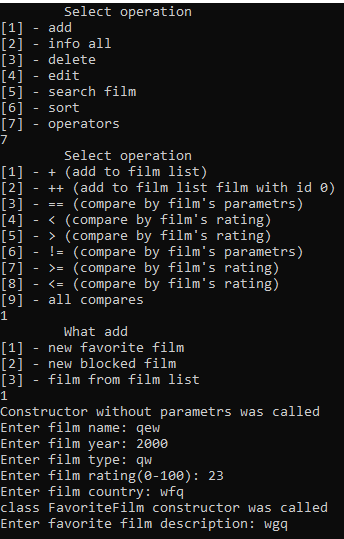


Рисунок 3. Оператор +

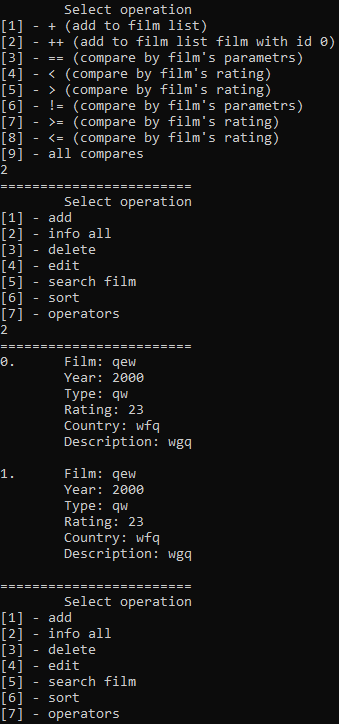


Рисунок 4. Оператор ++

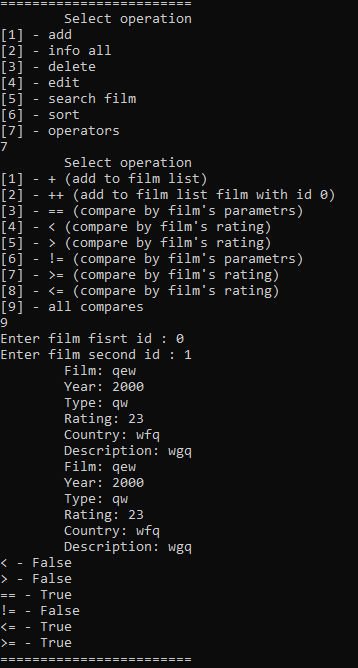


Рисунок 5. Операторы сравнения

**Вывод:** В ходе данной лабораторной работы было изучена перегрузка бинарных и унарных операторов для пользовательского класса, применение перегруженных операторов для операций над объектами классов.