

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №5**

**з дисципліни «Безпека програм та даних»**

**на тему: «Реалізація дискреційної моделі політики безпеки»**

Виконав: ???

Прийняв: ???

Дніпро, 2023

**Тема**: Реалізація дискреційної моделі політики безпеки

**Мета**: Ознайомитися з проблемами реалізації політик безпеки в комп'ютерних систем на прикладі дискреційної моделі.

**Завдання**:

Отримати дані про кількість користувачів і об'єктів комп'ютерної системи з таблиці, відповідно до варіанта. Реалізувати програмний модуль, що створює матрицю доступу користувачів до об'єктів комп'ютерної системи.

|  |  |  |
| --- | --- | --- |
| Варіант | Кількість суб’єктів доступу (користувачів) | Кількість суб’єктів доступу |
| 11 | 5 | 3 |

**Короткі теоретичні відомості:**

Під політикою безпеки розуміють набір норм, правил і практичних прийомів, що регулюють управління, захист і розподіл цінної інформації. Політика безпеки задає механізми управління доступу до об'єкту, визначає як дозволені, так і заборонені доступи.

Політика безпеки реалізується за допомогою адміністративно-організаційних заходів, фізичних і программно-технічних засобів і визначає архітектуру системи захисту. Для конкретної організації політика безпеки повинна носити індивідуальний характер і залежати від конкретної технології обробки інформації і використовуваних програмних і технічних засобів.

Політика безпеки визначається способом управління доступом, який задає порядок доступу до об'єктів системи. Розрізняють два основних види політики безпеки: виборчу і повноважну.

Виборча політика безпеки заснована на виборчому способі управління доступом. Виборче (або діскреціонное) управління доступом характеризується заданим адміністратором безліччю дозволених відносин доступу (наприклад, у вигляді трійок об'єкт - суб'єкт - тип доступу). Зазвичай для опису властивостей виборчого управління доступом застосовують математичну модель на основі матриці доступу.

**Текст програми:**

Файл New\_file.h:

#pragma once

#include <string>

#include <map>

#include"User.h"

class New\_file

{

private:

std::string name;

std::map<std::string, bool[3]> access\_rights;//{чтение, запись, исполнение}

public:

New\_file(std::string name) :name(name) {};

void change\_rights(User& new\_user, bool access\_rights[3]);//{чтение, запись, исполнение}

bool\* get\_access\_rights(User& new\_user);

std::string get\_name() { return name; }

};

Файл New\_file.cpp:

#include "New\_file.h"

void New\_file::change\_rights(User& new\_user, bool access\_rights[3])//{чтение, запись, исполнение}

{

New\_file::access\_rights[new\_user][0] = access\_rights[0];

New\_file::access\_rights[new\_user][1] = access\_rights[1];

New\_file::access\_rights[new\_user][2] = access\_rights[2];

}

bool\* New\_file::get\_access\_rights(User& new\_user)

{

return New\_file::access\_rights[new\_user];

}

Файл User.h:

#pragma once

#include <string>

struct User

{

std::string name;

User(std::string name) :name(name) {};

operator std::string()

{

return name;

}

};

Файл Source.cpp:

#include <iostream>

#include "User.h"

#include "New\_file.h"

int main()

{

setlocale(LC\_ALL, "Rus");

User\* users [5];

users[0] = new User("Peter");

users[1] = new User("Oleg");

users[2] = new User("Boris");

users[3] = new User("Vasiliy");

users[4] = new User("Guest");

New\_file\* new\_files[3];

new\_files[0] = new New\_file("Books");

new\_files[1] = new New\_file("Films");

new\_files[2] = new New\_file("Games");

bool full[] = { true,true,true };//полные права

bool rw[] = { true,true,false };//чтение и запись

bool empty[] = { false,false,false };//пустые права

bool r[] = { true,false,false };//чтение

bool e[] = { false,false,true };//исполнение

//устанавливаем права доступа

//для Петра

new\_files[0]->change\_rights(\*users[0], full);

new\_files[1]->change\_rights(\*users[0], full);

new\_files[2]->change\_rights(\*users[0], full);

//для Олега

new\_files[0]->change\_rights(\*users[1], full);

new\_files[1]->change\_rights(\*users[1], rw);

new\_files[2]->change\_rights(\*users[1], empty);

//для Бориса

new\_files[0]->change\_rights(\*users[2], empty);

new\_files[1]->change\_rights(\*users[2], r);

new\_files[2]->change\_rights(\*users[2], empty);

//для Василия

new\_files[0]->change\_rights(\*users[3], full);

new\_files[1]->change\_rights(\*users[3], full);

new\_files[2]->change\_rights(\*users[3], full);

//для Гостя

new\_files[0]->change\_rights(\*users[4], empty);

new\_files[1]->change\_rights(\*users[4], empty);

new\_files[2]->change\_rights(\*users[4], empty);

std::string name;

User\* current\_user=users[4];

std::cout << "Здравствуйте, введите имя пользавателя:" << std::endl;

std::cin >> name;

for (User\* i : users)

{

if (i->name == name)

{

current\_user = i;

break;

}

}

std::cout << "Вы вошли как " << current\_user->name<<std::endl;

std::cout << "Ваши права: " << std::endl;

for (New\_file\* i : new\_files)

{

auto temporary\_rights = i->get\_access\_rights(\*current\_user);

std::cout << i->get\_name() << ": " << (

(temporary\_rights[0] && temporary\_rights[1] && temporary\_rights[2]) ? "полные права" : (

(!temporary\_rights[0] && !temporary\_rights[1] && !temporary\_rights[2]) ? "нет прав" : (

std::string(((temporary\_rights[0]) ? "чтение" : "")) + " " +

std::string(((temporary\_rights[1]) ? "запись" : "")) + " " +

std::string(((temporary\_rights[2]) ? "исполнение" : "")) + " ")

)

) << std::endl;

}

std::string next\_step = "выход";

std::string object = " ";

while (true)

{

std::cout << "Что Вы хотите сделать?" << std::endl;

std::cin >> next\_step;

int counter = ((next\_step == "read") ? 0 :( (next\_step == "write") ? 1 :( (next\_step == "execute") ? 2 :((next\_step == "change")?3: 4))));

if (counter == 4)

{

break;

}

std::cout << "Над каким объектом Вы хотите это сделать?" << std::endl;

std::cin >> object;

New\_file\* current\_file=nullptr;

for (New\_file\* i : new\_files)

{

if (i->get\_name() == object)

{

current\_file = i;

break;

}

}

if (current\_file == nullptr)

{

std::cout << "Невозможно найти данный файл" << std::endl;

continue;

}

int a = 0;

switch (counter)

{

case 0:

if (current\_file->get\_access\_rights(\*current\_user)[0])

std::cout << "Операция прошла успешно" << std::endl;

else

std::cout << "У Вас нет прав для чтения" << std::endl;

break;

case 1:

if (current\_file->get\_access\_rights(\*current\_user)[1])

std::cout << "Операция прошла успешно" << std::endl;

else

std::cout << "У Вас нет прав для записи" << std::endl;

break;

case 2:

if (current\_file->get\_access\_rights(\*current\_user)[2])

std::cout << "Операция прошла успешно" << std::endl;

else

std::cout << "У Вас нет прав для исполнения" << std::endl;

break;

case 3:

std::cout << "Какие будут новые права у данного файла?" << std::endl;

std::cin >> a;

if (a == 1)

current\_file->change\_rights(\*current\_user, full);

else

current\_file->change\_rights(\*current\_user, empty);

std::cout << "Новые права: " << std::endl;

for (New\_file\* i : new\_files)

{

auto temporary\_rights = i->get\_access\_rights(\*current\_user);

std::cout << i->get\_name() << ": " << (

(temporary\_rights[0] && temporary\_rights[1] && temporary\_rights[2]) ? "полные права" : (

(!temporary\_rights[0] && !temporary\_rights[1] && !temporary\_rights[2]) ? "нет прав" : (

std::string(((temporary\_rights[0]) ? "чтение" : "")) + " " +

std::string(((temporary\_rights[1]) ? "запись" : "")) + " " +

std::string(((temporary\_rights[2]) ? "исполнение" : "")) + " ")

)

) << std::endl;

}

break;

default:

break;

}

}

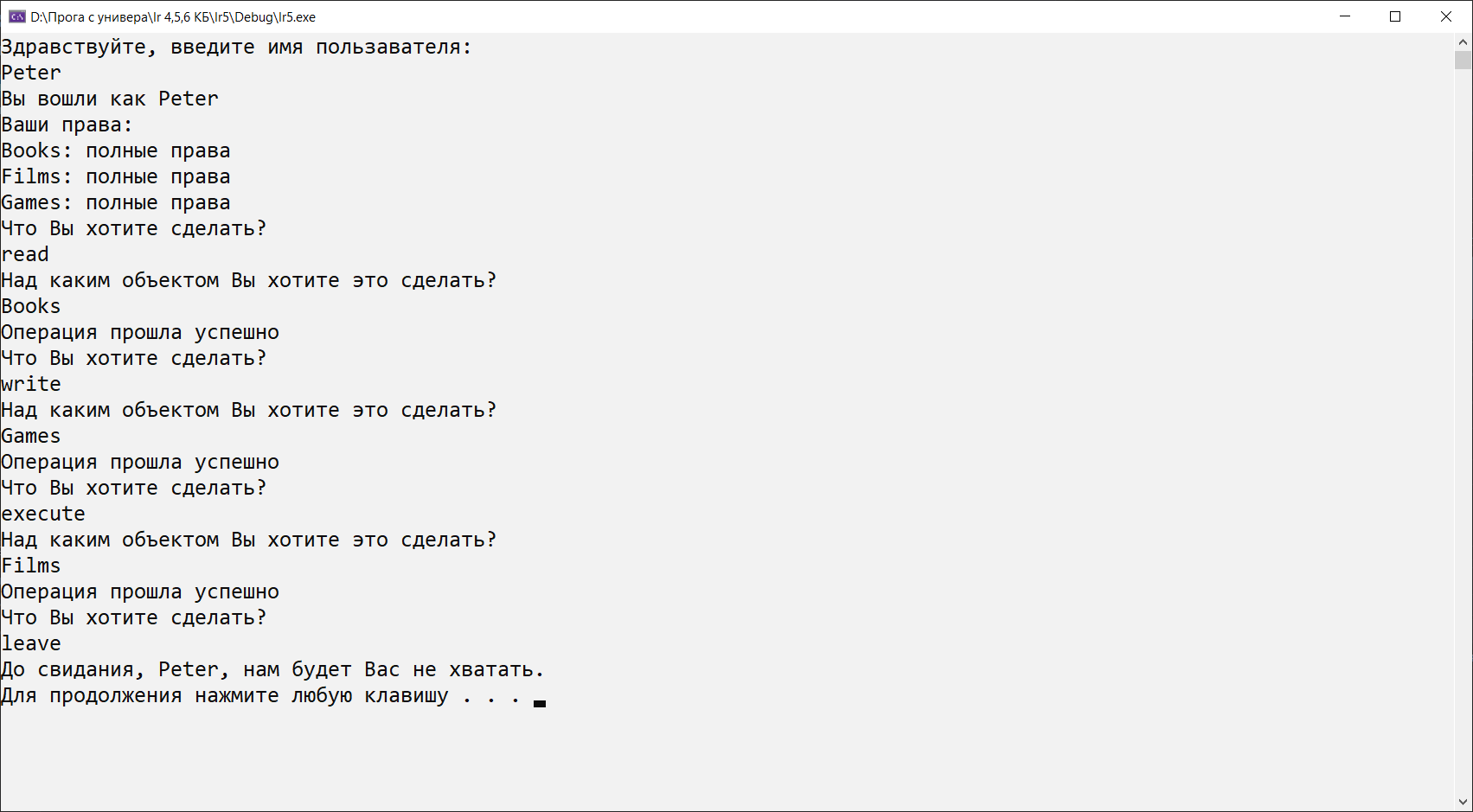
std::cout << "До свидания, " << current\_user->name << ", нам будет Вас не хватать." << std::endl;

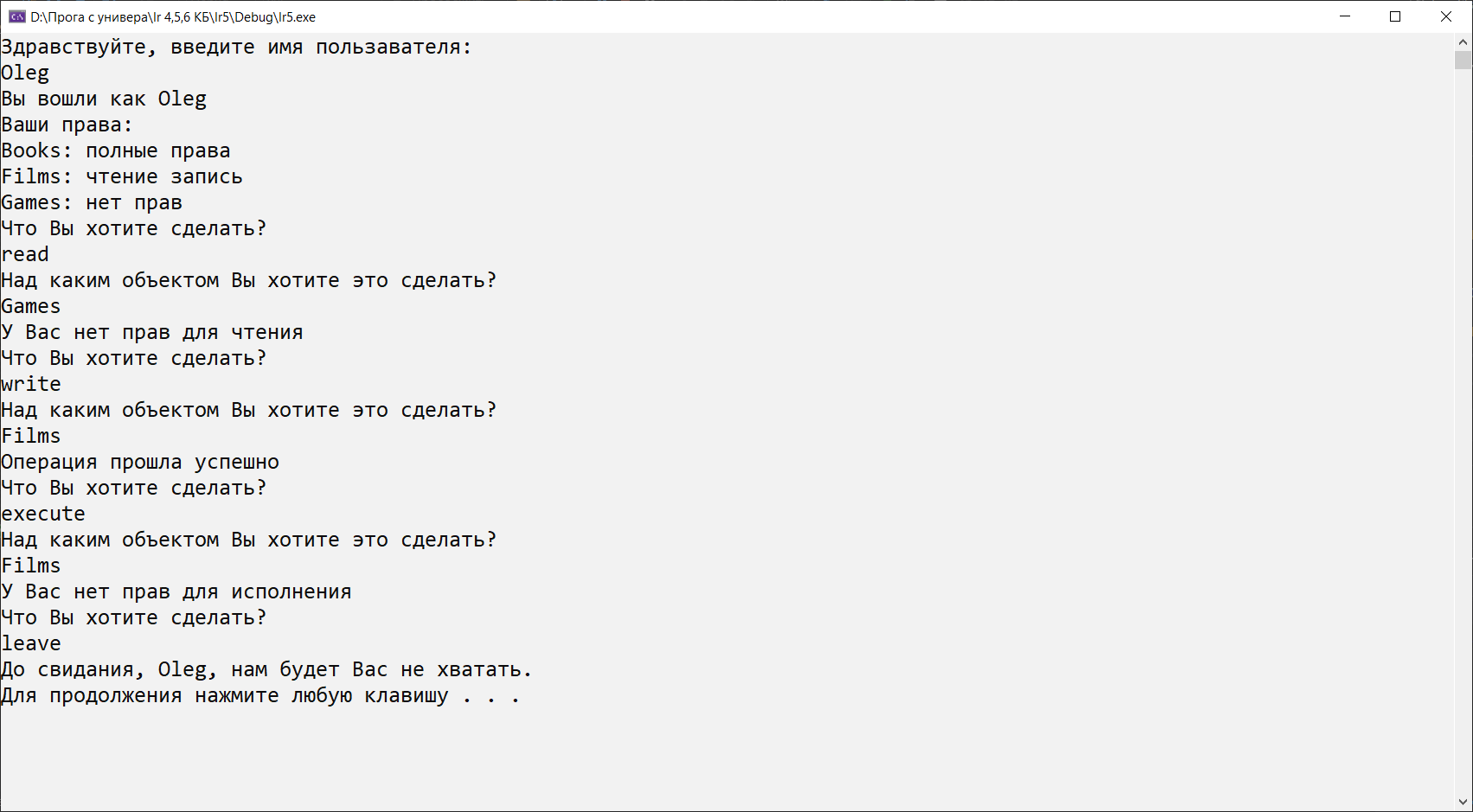
system("pause");

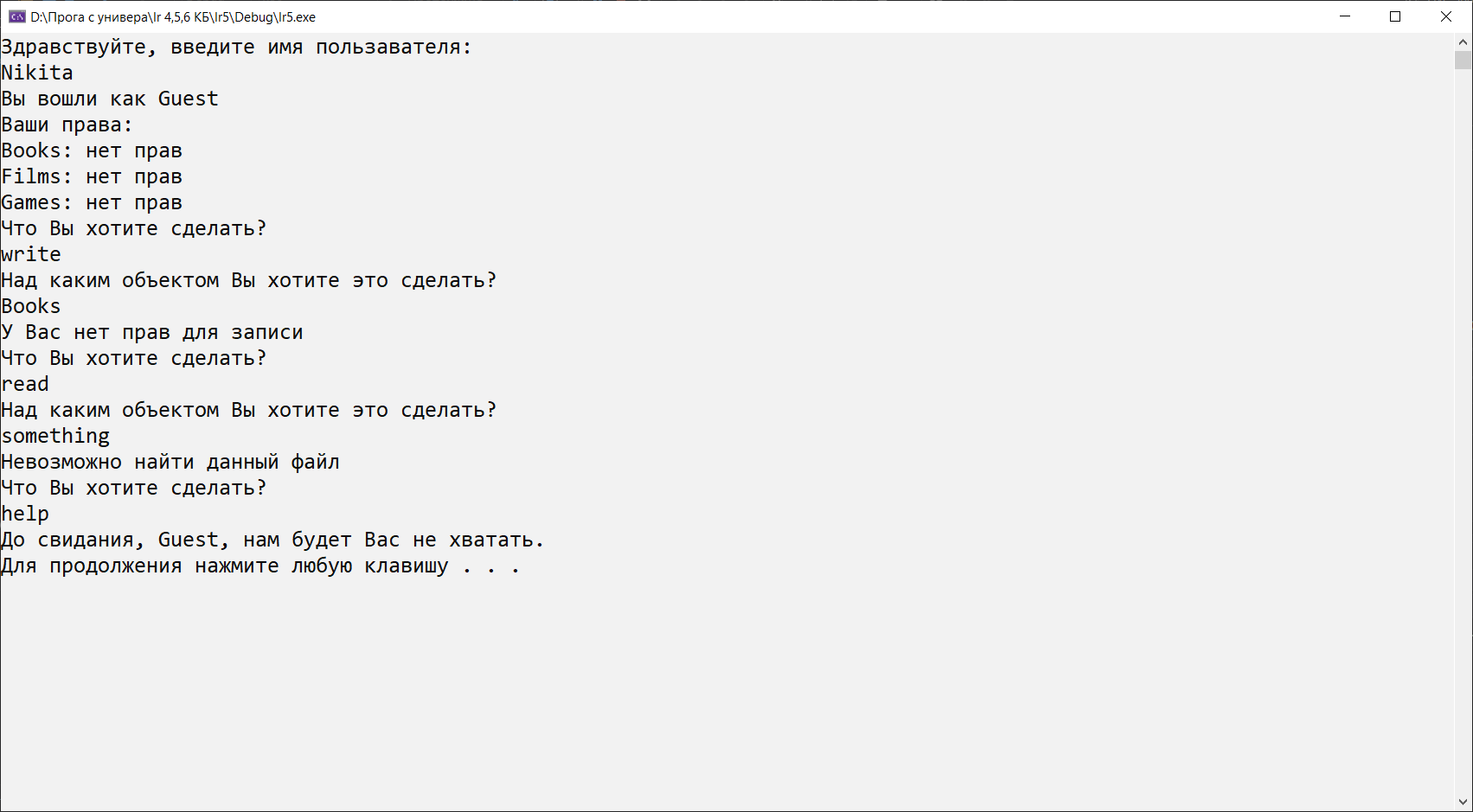
return 0;

}

**Результати виконання програми:**

1) Перший запуск програми 

2) Другий запуск програми 

3) Третій запуск програми 

**Висновок**: в ході виконання лабораторної роботи я ознайомився з проблемами реалізації політик безпеки в комп'ютерних систем на прикладі дискреційної моделі.