Министерство науки и высшего образования Российской Федерации  
Пензенский государственный университет  
Кафедра вычислительная техника

**ОТЧËТ**  
по лабораторной работе №2  
по курсу «Объектно-ориентированное программирование»  
на тему «Механизмы перегрузки операций и дружественных функций в языке»

Выполнили студенты группы 22ВВП1:  
Беляев Д.

Захаров А.

Приняли:  
Евсеева Ю.И.  
Гудков А. А.

Пенза 2024

**Название**

Механизмы перегрузки операций и дружественных функций в языке

**Цель работы**

Изучить механизмы перегрузки операций и дружественных функций в языке

**Лабораторное задание**

1. В базовом разрабатываемом классе перегрузить операции равенства (==), неравенства (!=) и присваивания (=);
2. Разработать новый класс Observer, который реализует дружественный метод GetInfo по отношению к базовому классу. Метод GetInfo должен выводить в формате целостной строки сведения из private-полей базового класса

**Метод решения задачи**

Разработали классы согласно заданию, сделали перегрузки методов, и добавили дружественный метод

**Листинг**

**main.cpp**

#include <iostream>

#include <vector>

#include "Patient.h"

const int LIST\_SIZE = 3;

Patient\* CreatePatient();

void FindByDisease(int diseaseNumber, std::vector<Patient\*> ptList);

void FindByCardInRange(int rangeMin, int rangeMax, std::vector<Patient\*> ptList);

int main()

{

std::vector<Patient\*> ptList;

for (size\_t i = 0; i < LIST\_SIZE; i++)

{

ptList.push\_back(CreatePatient());

std::cout << "\n";

}

/\*int disNumb = 0;

std::cout << "Insert disease number: ";

std::cin >> disNumb;

FindByDisease(disNumb, ptList);

int cardNumbMin = 0;

int cardNumbMax = 0;

std::cout << "Insert card range number min and max: ";

std::cin >> cardNumbMin >> cardNumbMax;

FindByCardInRange(cardNumbMin, cardNumbMax, ptList);\*/

std::cout << "Before assignment: " << ptList[0]->GetFio() << std::endl;

ptList[0] = ptList[1];

std::cout << "After assignment: " << ptList[0]->GetFio() << std::endl;

bool equal = ptList[0] == ptList[1];

bool notEqual = ptList[0] != ptList[1];

std::cout << ptList[0]->GetFio() << " == " << ptList[1]->GetFio() << " = " << equal << std::endl;

std::cout << ptList[0]->GetFio() << " != " << ptList[1]->GetFio() << " = " << notEqual << std::endl;

Observer obs;

obs.GetInfo(\*ptList[0]);

return 0;

}

Patient\* CreatePatient()

{

std::string name;

std::string secName;

std::string otch;

int cardNumb;

int diseaseNumb;

std::cout << "Create Patient\n";

std::cout << "Insert name: ";

std::cin >> name;

std::cout << "Insert second name: ";

std::cin >> secName;

std::cout << "Insert otch: ";

std::cin >> otch;

std::cout << "Insert card number: ";

std::cin >> cardNumb;

std::cout << "Insert disease number: ";

std::cin >> diseaseNumb;

return new Patient(name, secName, otch, cardNumb, diseaseNumb);

}

void FindByCardInRange(int rangeMin, int rangeMax, std::vector<Patient\*> ptList)

{

std::cout << "Patient with card in range(" << rangeMin << ", " << rangeMax << "):\n";

for (size\_t i = 0; i < LIST\_SIZE; i++)

{

int cardNumb = ptList[i]->GetCardNumb();

if (cardNumb >= rangeMin && cardNumb <= rangeMax)

std::cout << ptList[i]->GetFio() << "\n";

}

}

void FindByDisease(int diseaseNumber, std::vector<Patient\*> ptList)

{

std::cout << "Patient with disease " << diseaseNumber << ":\n";

for (size\_t i = 0; i < LIST\_SIZE; i++)

{

if (ptList[i]->GetDiseaseNumb() != diseaseNumber)

continue;

std::cout << ptList[i]->GetFio() << "\n";

}

std::cout << "\n";

}

**patient.h**

#ifndef PATIENT

#define PATIENT

#include <string>

class Patient;

class Observer

{

public:

void GetInfo(Patient& patient);

};

class Patient

{

public:

friend void Observer::GetInfo(Patient& patient);

Patient(std::string name, std::string secName, std::string otch, int cardNumb, int diseaseNumb);

~Patient();

// Getter

std::string GetFio();

int GetCardNumb();

int GetDiseaseNumb();

// Setter

void SetFio(std::string name, std::string secName, std::string otch);

void SetCardNumb(int cardNumb);

void SetDiseaseNumb(int diseaseNumb);

// Method overloading

bool operator == (const Patient& pat) const { return \_cardNumb == pat.\_cardNumb; }

bool operator != (const Patient& pat) const { return \_cardNumb != pat.\_cardNumb; }

Patient& operator = (const Patient& pat)

{

this->SetFio(pat.\_name, pat.\_secName, pat.\_otch);

this->SetCardNumb(pat.\_cardNumb);

this->SetDiseaseNumb(pat.\_diseaseNumb);

return \*this;

}

private:

std::string \_name;

std::string \_secName;

std::string \_otch;

int \_cardNumb;

int \_diseaseNumb;

};

#endif // !PATIENT

**patient.cpp**

#include <iostream>

#include "Patient.h"

Patient::Patient(std::string name, std::string secName, std::string otch, int cardNumb, int diseaseNumb)

{

SetFio(name, secName, otch);

\_cardNumb = cardNumb;

\_diseaseNumb = diseaseNumb;

std::cout << "Patient " << GetFio() << " with card: " << GetCardNumb() << " with disease " << GetDiseaseNumb() << " created" << std::endl;

}

Patient::~Patient()

{

std::cout << "Patient deleted" << std::endl;

}

#pragma region Getter

int Patient::GetCardNumb() { return \_cardNumb; }

int Patient::GetDiseaseNumb() { return \_diseaseNumb; }

std::string Patient::GetFio() { return "FIO: " + \_name + "\_" + \_secName + "\_" + \_otch; }

#pragma endregion

#pragma region Setter

void Patient::SetFio(std::string name, std::string secName, std::string otch)

{

\_name = name;

\_secName = secName;

\_otch = otch;

}

void Patient::SetCardNumb(int cardNumb) { \_cardNumb = cardNumb; }

void Patient::SetDiseaseNumb(int diseaseNumb) { \_diseaseNumb = diseaseNumb; }

#pragma endregion

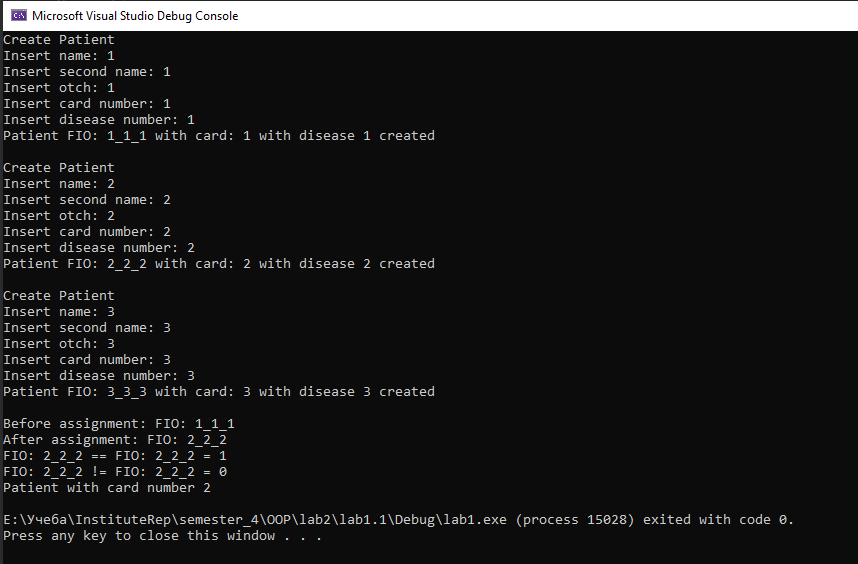
void Observer::GetInfo(Patient& patient)

{

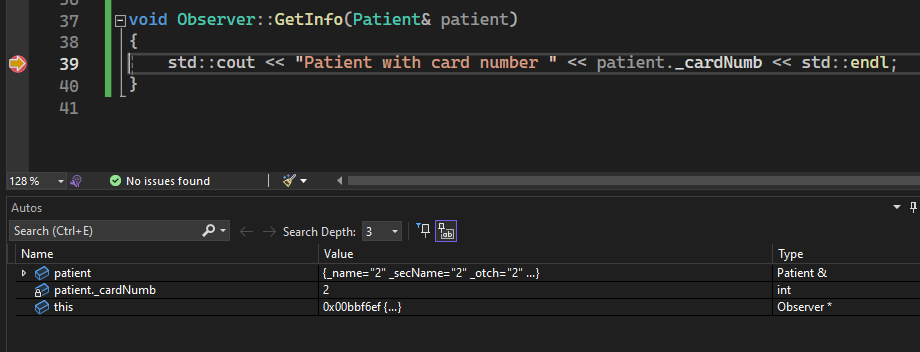
std::cout << "Patient with card number " << patient.\_cardNumb << std::endl;

}

**Результат работы программы**

****

**Протокол трассировки программы**

****

**Вывод**

Мы изучили механизмы перегрузки операций и дружественных функций в языке