Министерство науки и высшего образования РФ

 Пензенский государственный университет Кафедра "Вычислительная техника"

Отчет по лабораторной работе №2

Работу выполнили студенты группы 23ВВВ3 бригады №9:

Полиневский Вадим

Проверили:

Кандидат т.н. Гудков А.А

Кандидат т.н. Евсеева И. Е

Пенза 2025

**Цель работы**: Изучить механизмы перегрузки операций и дружественных функций в языке C++

**Задаие :**Модифицировать программу из лабораторной работы № 2 следующим образом:

* В базовом разрабатываемом классе перегрузить операции равенства (==), неравенства (!=) и присваивания (=);
* 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.

**Описание практической части**:

Для выполнения программы было использовано два модуля: **lab\_1.cpp** с функцией main и файл заголовка **header.h.**

Листинг:

**lab\_1.cpp:**

#include <iostream>

#include <vector>

#include "header.h"

using namespace std;

int main (){

unsigned int count\_objects {3};

vector <Bus> objects;

cout << "lab\_2" << endl;

for (int i = 0; i< count\_objects; i++) {

if (i == 0) Set\_Values\_For\_ClassObject(1, objects, i);

if (i == 1) Set\_Values\_For\_ClassObject(2, objects, i);

if (i == 2) Set\_Values\_For\_ClassObject(3, objects, i);

}

//Find\_in\_vector(objects);

//lab2 added

Compraing\_ClassObjects(objects);

Print\_Full\_String(objects);

}

**header.h:**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Bus {

friend class Observer;

private:

string Name\_Driver;

unsigned int Number\_Path;

string Brand\_Bus;

unsigned int Year\_Beginning\_Start;

unsigned int Mileage;

public:

Bus (){

Name\_Driver = "No name";

Brand\_Bus = "No Brand";

Year\_Beginning\_Start = 2000;

}

Bus (string \_Name\_Driver, unsigned int \_Number\_Path, string \_Brand\_Bus, unsigned int \_Year\_Start, unsigned int \_Mileage) {

Name\_Driver = \_Name\_Driver;

Number\_Path = \_Number\_Path;

Brand\_Bus = \_Brand\_Bus;

Year\_Beginning\_Start = \_Year\_Start;

Mileage = \_Mileage;

}

Bus (const Bus& \_object\_BUS) {

Name\_Driver = \_object\_BUS.Name\_Driver;

Number\_Path = \_object\_BUS.Number\_Path;

Brand\_Bus = \_object\_BUS.Brand\_Bus;

Year\_Beginning\_Start = \_object\_BUS.Year\_Beginning\_Start;

Mileage = \_object\_BUS.Mileage;

}

~Bus(){

}

// added in lab2

bool operator == (const Bus & \_Bus) {

if (this->Brand\_Bus == \_Bus.Brand\_Bus) return true;

else return false;

}

bool operator != (const Bus & \_Bus) {

if (this->Number\_Path != \_Bus.Number\_Path) return true;

else return false;

}

Bus& operator = (const Bus & \_Bus) {

this->Name\_Driver = \_Bus.Name\_Driver;

this->Number\_Path =\_Bus.Number\_Path;

this->Brand\_Bus = \_Bus.Brand\_Bus;

this->Year\_Beginning\_Start = \_Bus.Year\_Beginning\_Start;

this->Mileage = \_Bus.Mileage;

return \*this;

}

//

void set\_Name\_Driver(string \_Name) {Name\_Driver = \_Name;}

void set\_Number\_Path(unsigned int \_Number\_Path){Number\_Path = \_Number\_Path;}

void set\_Brand\_Bus(string \_Brand\_Bus){Brand\_Bus = \_Brand\_Bus;}

void set\_Year\_Begginning\_Start(unsigned int \_Year\_Beginning\_Start){Year\_Beginning\_Start = \_Year\_Beginning\_Start;}

void set\_Mileage(unsigned int \_Mileage){Mileage = \_Mileage;}

string get\_Name\_Driver() {return Name\_Driver;}

unsigned int get\_Number\_Path(){return Number\_Path;}

string get\_Brand\_Bus(){return Brand\_Bus;}

unsigned int get\_Year\_Begginning\_Start(){return Year\_Beginning\_Start;}

unsigned int get\_Mileage(){return Mileage;}

};

//added in lab2

class Observer{

public:

string Get\_Info(const Bus& \_Bus){return "" + \_Bus.Name\_Driver + ", " + to\_string(\_Bus.Number\_Path) + ", " + \_Bus.Brand\_Bus + ", " + to\_string(\_Bus.Year\_Beginning\_Start) + ", " + to\_string(\_Bus.Mileage);}

};

//

void Print\_Vulue(Bus temp\_object){

cout << temp\_object.get\_Name\_Driver() << ", "<< temp\_object.get\_Number\_Path()<< ", "<< temp\_object.get\_Brand\_Bus()<< ", "<< temp\_object.get\_Year\_Begginning\_Start() <<", "<< temp\_object.get\_Mileage() << endl;

}

void Enter\_Value(string \*Name, string \*Brand, unsigned int\* Year, unsigned int \*Number, unsigned int \*Milage) {

cout << "ENTER VALUES: " ;

cout << "enter Name\_Driver: ";

cin >> \*Name;

cout << "enter Number\_Path: ";

cin >> \*Number;

cout << "enter Brand\_Bus: ";

cin >> \*Brand;

cout << "enter Year\_Beginning\_Start: ";

cin >> \*Year;

cout << "enter Mileage: ";

cin >> \*Milage;

}

void Set\_Values\_For\_ClassObject(int value, vector<Bus>& Vector, int ptr\_Vector) {

string Name, Brand;

unsigned int Number, Milage , Year;

Enter\_Value(&Name, &Brand, &Year, &Number, &Milage);

switch (value)

{

case 1:

{

Bus object;

object.set\_Name\_Driver(Name);

object.set\_Number\_Path(Number);

object.set\_Brand\_Bus(Brand);

object.set\_Year\_Begginning\_Start(Year);

object.set\_Mileage(Milage);

Vector.push\_back(object);

break;

}

case 2:

{

Bus object(Name, Number, Brand, Year, Milage);

Vector.push\_back(object);

break;

}

case 3:

{

Bus object = Vector[ptr\_Vector - 1];

Vector.push\_back(object);

break;

}

default:

break;

}

}

void Sorting\_by\_Number\_Path(unsigned int number\_path, Bus temp\_object){

cout << "Sortig by number path: " << number\_path << endl;

if (number\_path == temp\_object.get\_Number\_Path()) Print\_Vulue(temp\_object);

cout << "--------" << endl;

}

void Sorting\_by\_Year(Bus temp\_object){

unsigned int year\_now = 2025;

cout << "Sortig by bus using more 10 year:" << endl;

if ((year\_now - temp\_object.get\_Year\_Begginning\_Start()) >= 10) Print\_Vulue(temp\_object);

cout << "--------"<< endl;

}

void Sorting\_by\_Mileage(Bus temp\_object){

cout << "Sorting by mileage" << endl;

if (temp\_object.get\_Mileage() >= 10000) Print\_Vulue(temp\_object);

cout << "--------" << endl;

}

void Find\_in\_vector(vector <Bus> temp\_object){

unsigned int temp\_number\_path;

cout << "Enter number path for find: ";

cin >> temp\_number\_path;

for (int i = 0; i< temp\_object.size(); i++) {

Sorting\_by\_Number\_Path(temp\_number\_path ,temp\_object[i]);

}

for (int i = 0; i < temp\_object.size(); i++) {

Sorting\_by\_Year(temp\_object[i]);

}

for (int i =0; i < temp\_object.size(); i++) {

Sorting\_by\_Mileage(temp\_object[i]);

}

}

// FUNCTIONS FOR OPERATOR OVERLORD

void Compraing\_ClassObjects(vector <Bus> &temp\_objects){

if (temp\_objects[temp\_objects.size() - 1] == temp\_objects[temp\_objects.size() - 2]) cout << "the last two objects-BRAND\_BUS are equel\n---------" << endl;

else cout << "the last two objects-BRAND\_BUS are not equel\n---------" << endl;

if (temp\_objects[temp\_objects.size() - 1] != temp\_objects[temp\_objects.size() - 2]) cout << "the last two objects\_PATH are not equel\n---------" << endl;

else cout << "the last two objects\_PATH are equel\n---------" << endl;

temp\_objects[temp\_objects.size() -1] = temp\_objects[temp\_objects.size() - 3];

}

void Print\_Full\_String(vector <Bus> temp\_objects){

Observer TEMP;

for (int i = 0; i < temp\_objects.size(); i++) {

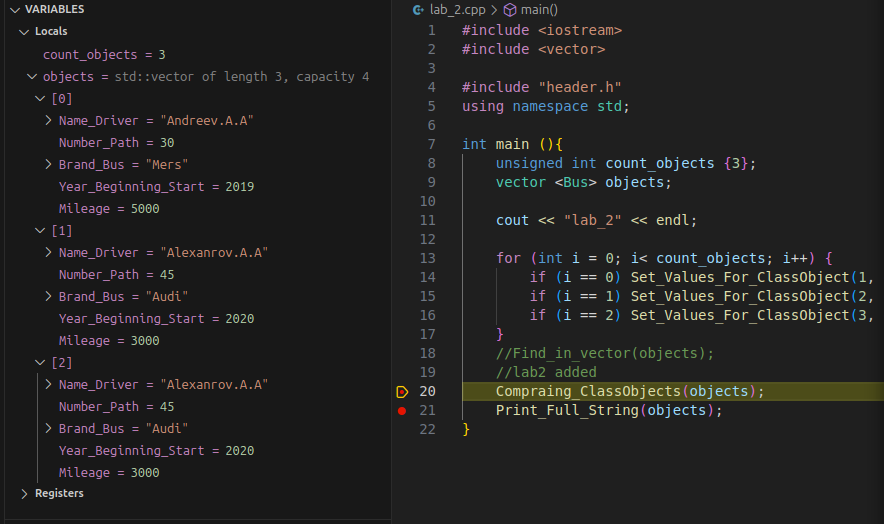
cout << TEMP.Get\_Info(temp\_objects[i])<< "\n ----------"<< endl;

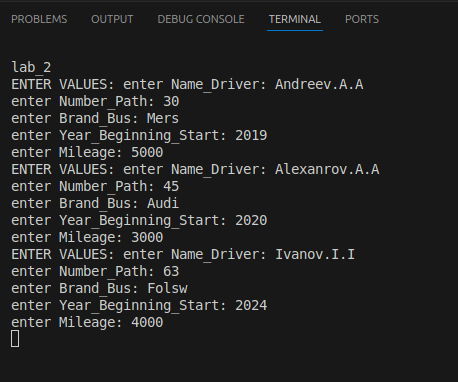
}

}

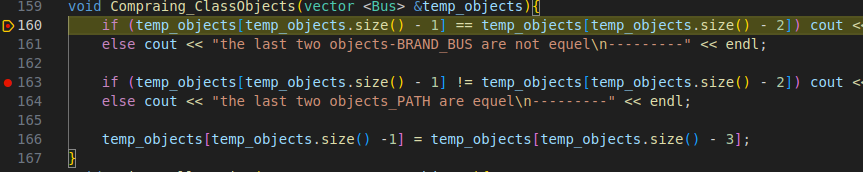
**Отладка:**

Значения полей объектов в векторе после пользовательского ввода, по логике идентичному lab1, до вызова основных функций lab2:

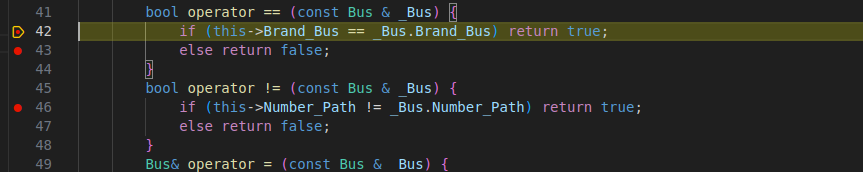


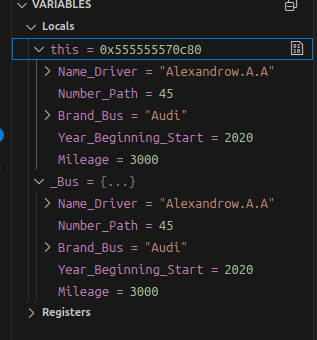


Вызов функции Compraing\_ClassObjects, сравнение по BRAND.

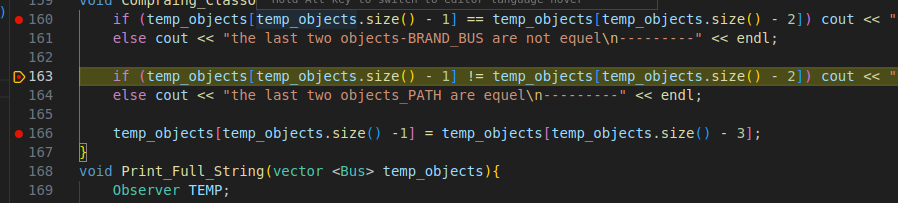


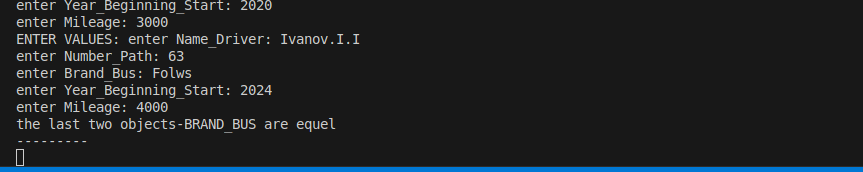
Следкющий шаг, оператор ==. Переменные



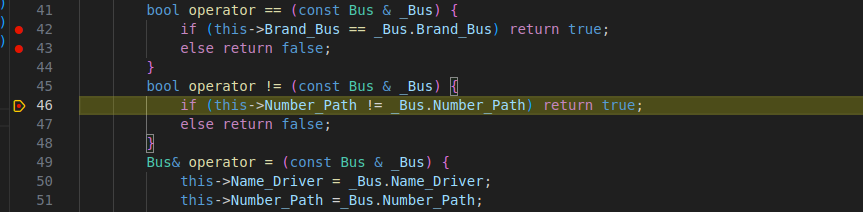


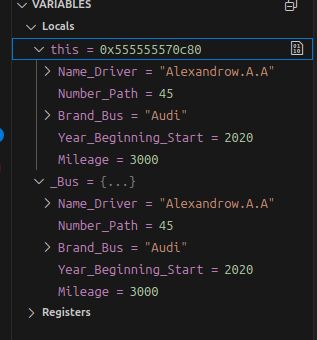
Следкющий шаг. Terminal:



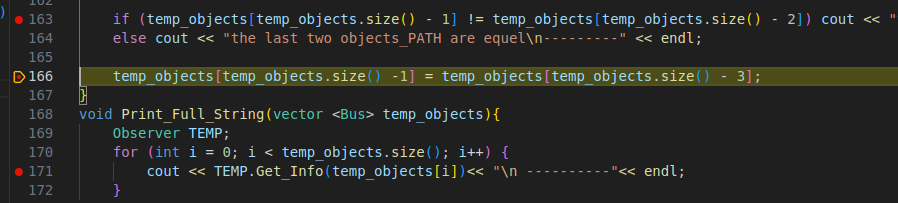


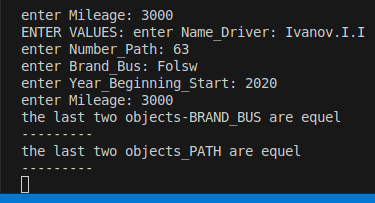
Следующий шаг, оператор !=. Переменные

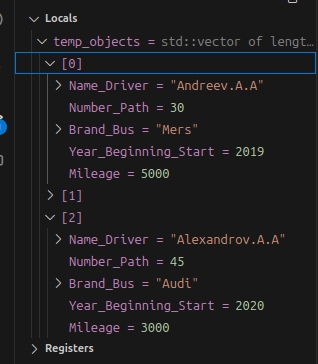




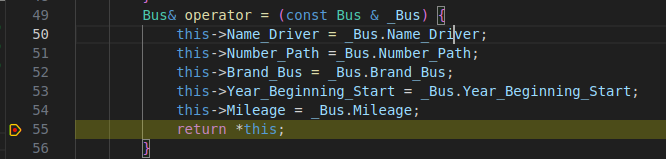
Следующий шаг. Консоль. Данные до перегрузки оператора =:

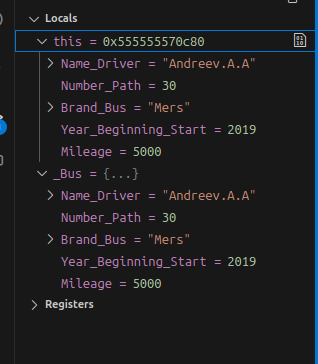




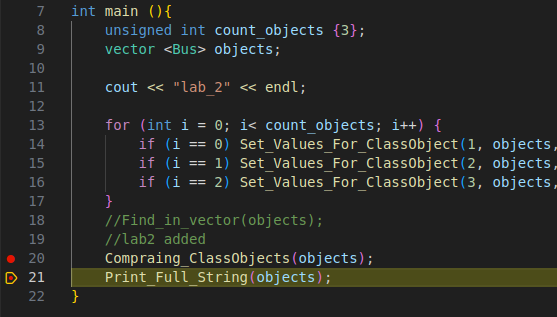


Следующий шаг. Перегрузка оператора =.

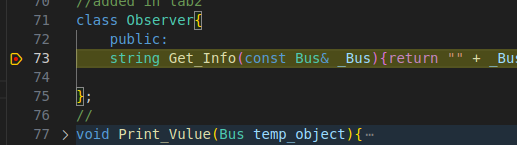




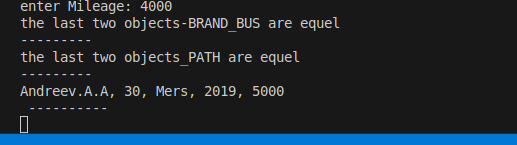
Следующий шаг. Main функция:



Следующий шаг. Дружественный класс. Функция вывода:



Следующий шаг. Консоль:



**Terminal-Log:**

lab\_2

ENTER VALUES: enter Name\_Driver: Andreev.A.A

enter Number\_Path: 30

enter Brand\_Bus: Mers

enter Year\_Beginning\_Start: 2019

enter Mileage: 5000

ENTER VALUES: enter Name\_Driver: Alexandrov.A.A

enter Number\_Path: 45

enter Brand\_Bus: Audi

enter Year\_Beginning\_Start: 2020

enter Mileage: 3000

ENTER VALUES: enter Name\_Driver: Ivanov.I.I

enter Number\_Path: 63

enter Brand\_Bus: Folsw

enter Year\_Beginning\_Start: 2024

enter Mileage: 4000

the last two objects-BRAND\_BUS are equel

---------

the last two objects\_PATH are equel

---------

Andreev.A.A, 30, Mers, 2019, 5000

----------

Alexandrov.A.A, 45, Audi, 2020, 3000

----------

Andreev.A.A, 30, Mers, 2019, 5000