Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №2

по дисциплине: «Объектно ориентированное программирование»

на тему: «Изучить механизмы перегрузки операций и дружественных функций в языке C++»

Выполнили:

студенты группы 22ВВВ2

Ипполитов И. Д.

Хоссейни Нежад С. А. С. М.

Приняли:

Евсеева Ю.И.

Гудков А.А.

Пенза, 2024

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

1. Получить индивидуальное задание у преподавателя.
2. Написать программу, выполняющую действия, согласно заданию.
3. Данные в программе должны вводиться с клавиатуры
4. Программу требуется отладить и протестировать на различных наборах данных
5. Продемонстрировать работу программы на экране.
6. Сдать теоретическую часть по отчету преподавателю.

**Задания**

Модифицировать программу из лабораторной работы № 1 следующим образом:

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.

**Листинг**

#include <iostream>

#include <string>

#include <windows.h>

#include <vector>

using namespace std;

class Train;

class Observer {

private:

string destination;

int number;

int time;

int mainCount;

int kupeCount;

int plackartCount;

public:

Observer(string destination, int number, int time, int mainCount, int kupeCount, int plackartCount)

{

this->destination = destination;

this->number = number;

this->time = time;

this->mainCount = mainCount;

this->kupeCount = kupeCount;

this->plackartCount = plackartCount;

}

Observer(const Observer& t)

{

this->destination = t.destination;

this->number = t.number;

this->time = t.time;

this->mainCount = t.mainCount;

this->kupeCount = t.kupeCount;

this->plackartCount = t.plackartCount;

}

~Observer()

{

}

void GetInfo(const Train&);

};

class Train {

friend void Observer::GetInfo(const Train&);

private:

string destination;

int number;

int time;

int mainCount;

int kupeCount;

int plackartCount;

public:

Train(string destination, int number, int time, int mainCount, int kupeCount, int plackartCount)

{

this->destination = destination;

this->number = number;

this->time = time;

this->mainCount = mainCount;

this->kupeCount = kupeCount;

this->plackartCount = plackartCount;

}

Train(const Train& t)

{

this->destination = t.destination;

this->number = t.number;

this->time = t.time;

this->mainCount = t.mainCount;

this->kupeCount = t.kupeCount;

this->plackartCount = t.plackartCount;

}

~Train()

{

}

bool operator == (const Train& train) const {

return mainCount == train.mainCount;

}

bool operator != (const Train& train) const {

return mainCount != train.mainCount;

}

Train& operator++ () {

number += 1;

return \*this;

}

Train operator++ (int) {

Train copy{ \*this };

++(\*this);

return copy;

}

Train& operator-- () {

number -= 1;

return \*this;

}

Train operator-- (int) {

Train copy{ \*this };

--(\*this);

return copy;

}

Train& operator = (const Train& train) {

destination = train.destination;

number = train.number;

time = train.time;

mainCount = train.mainCount;

kupeCount = train.kupeCount;

plackartCount = train.plackartCount;

return \*this;

}

void show() {

cout << endl;

cout << "Номер: " << number << endl;

cout << "Общее количество мест: " << mainCount << endl;

}

string getDestination() { return destination; }

int getTime() { return time; }

int getMainCount() { return mainCount; }

int getNumber() { return number; }

int getKupeCount() { return kupeCount; }

int getPupeCount() { return plackartCount; }

string setDestination(string destination) {

this->destination = destination;

}

int setTime(int time) {

this->time = time;

}

int setMainCount(int mainCount) {

this->mainCount = mainCount;

}

int setNumber(int number) {

this->number = number;

}

int setKupeCount(int kupeCount) {

this->kupeCount = kupeCount;

}

int setPupeCount(int plackartCount) {

this->plackartCount = plackartCount;

}

};

void Observer::GetInfo(const Train& trainer) {

cout << "Номер: " << trainer.number << " Количество мест: " << trainer.mainCount << endl;

}

int main()

{

setlocale(LC\_ALL, "rus");

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

string destination = "1";

int number = 1;

int time = 1;

int mainCount = 1;

int kupeCount = 1;

int plackartCount = 1;

vector<Train> v1; // массив поездов

for (size\_t i = 0; i < 2; i++)

{

cout << "Введите данные о поезде " << i + 1 << endl;

cout << "Номер: "; cin >> number;

cout << "Общее количество мест: "; cin >> mainCount;

cout << endl;

Train t{ destination, number, time, mainCount, kupeCount, plackartCount };

v1.push\_back(t);

}

for (size\_t i = 0; i < v1.size() - 1; i++)

{

if (v1[i] == v1[i + 1])

{

cout << "Количество мест у поездов одинаково" << endl;

}

if (v1[i] != v1[i + 1])

{

cout << "Количество мест у поездов разное" << endl;

}

}

v1[0].show();

v1[1].show();

cout << "Приравниваем количества мест 2 поезда 1" << endl;

v1[0] = v1[1];

v1[0].show();

v1[1].show();

v1[0]++;

v1[0]--;

++v1[0];

--v1[0];

cout << "Введите данные о поезде " << endl;

cout << "Номер: "; cin >> number;

cout << "Общее количество мест: "; cin >> mainCount;

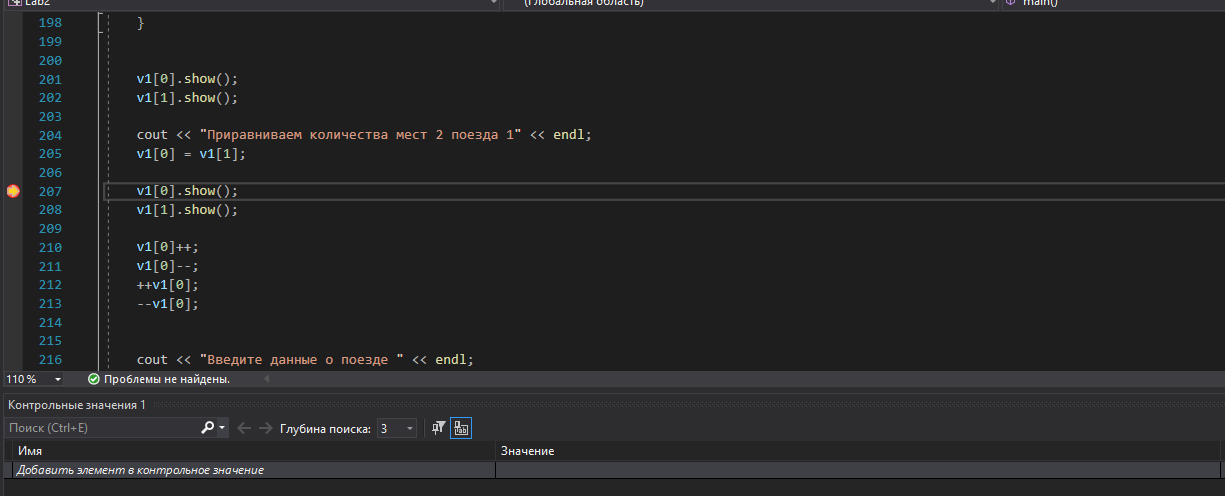
cout << endl;

Observer t{ destination, number, time, mainCount, kupeCount, plackartCount };

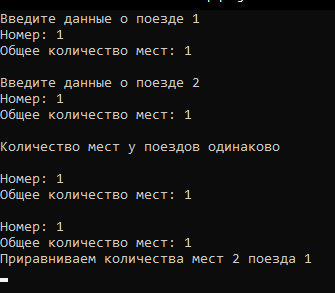
t.GetInfo(v1[0]);

}

**Отладка программы**



**Результат работы программы**

****

**Вывод:** изучили механизмы перегрузки операций и дружественных функций в языке C++.