ПРИЛОЖЕНИЕ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»

(СПбГУТ)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Кафедра Систем обработки данных

**Дисциплина «Технологии программирования»**

**ОТЧЕТ**

**за практическое занятие №5**

**Тема: Проектирование классов с наследованием**

Выполнил

Студент 2 курса, гр. ИБ-32вп

\_\_\_\_\_\_\_\_\_\_\_\_\_ Братишкин Д.Е.

Принял

Доцент кафедры БИС

\_\_\_\_\_\_\_\_\_\_\_\_\_ Медведев В.А.

Санкт-Петербург

2024

Тема: Проектирование классов с наследованием

Цель: Получение практических навыков проектирования классов и разработки механизмов открытого и закрытого наследования в C++.

Задание на практическое занятие

**Вариант А.**

**Задание 1.** Спроектировать структуру классов в соответствии с индивидуальным заданием (см. таблица 7). Наследование одного из производных классов спроектировать по типу **private**. *Вычисляемый показатель* реализовать для объектов производных классов в отдельной функции, не являющейся дружественной к классам.

**Задание 2.** Для каждого производного класса динамически выделить память для 5-элементного массива объектов. Свойства объектов задавать случайным образом. Определить *Вычисляемый показатель*.

**Задание 3.** Проект разместить в нескольких файлах.

#include "LocalNetwork.h"

#include <cstdlib>

#include <ctime>

int main() {

const int arraySize = 5;

int price, cnt;

srand(static\_cast<unsigned>(time(nullptr)));

cout << "Введите цену для установки устройств" << endl;

cin >> price;

cout << "Введите количество устройств" << endl;

cin >> cnt;

// Создаем массивы сетей

PeerToPeerNetwork\* peerNetworks = new PeerToPeerNetwork[arraySize]{

PeerToPeerNetwork("PeerNet1", price, cnt),

PeerToPeerNetwork("PeerNet2", price, cnt),

PeerToPeerNetwork("PeerNet3", price, cnt),

PeerToPeerNetwork("PeerNet4", price, cnt),

PeerToPeerNetwork("PeerNet5", price, cnt)

};

ClientServerNetwork\* clientNetworks = new ClientServerNetwork[arraySize]{

ClientServerNetwork("ClientNet1", 5000, 10),

ClientServerNetwork("ClientNet2", 6000, 15),

ClientServerNetwork("ClientNet3", 4500, 12),

ClientServerNetwork("ClientNet4", 5200, 20),

ClientServerNetwork("ClientNet5", 5600, 18)

};

// Выводим информацию о сетях

for (int i = 0; i < arraySize; ++i) {

cout << "Одноранговая сеть #" << (i + 1) << ":\n";

peerNetworks[i].showDetails();

cout << "Итоговая стоимость монтажа: " << peerNetworks[i].calculateInstallationCost() << endl;

cout << "Индикатор: " << computeIndicator(peerNetworks[i]) << endl << endl;

cout << "Сеть клиент-сервер #" << (i + 1) << ":\n";

clientNetworks[i].showDetails();

cout << "Итоговая стоимость монтажа: " << clientNetworks[i].calculateInstallationCost() << endl;

cout << "Индикатор: " << computeIndicator(clientNetworks[i]) << endl << endl;

}

delete[] peerNetworks;

delete[] clientNetworks;

return 0;

}

#ifndef LOCALNETWORK\_H

#define LOCALNETWORK\_H

#include <iostream>

#include <string>

using namespace std;

class LocalNetwork {

protected:

string networkName;

double installationCost;

public:

LocalNetwork(string name = "Unnamed", double cost = 0.0);

virtual ~LocalNetwork();

virtual void showDetails() const = 0;

virtual double calculateInstallationCost() const = 0;

};

class PeerToPeerNetwork : private LocalNetwork {

private:

int numberOfNodes;

public:

PeerToPeerNetwork(string name, double cost, int nodes);

void showDetails() const override;

double calculateInstallationCost() const override;

friend double computeIndicator(const PeerToPeerNetwork& network);

};

class ClientServerNetwork : private LocalNetwork {

private:

int numberOfClients;

public:

ClientServerNetwork(string name, double cost, int clients);

void showDetails() const override;

double calculateInstallationCost() const override;

friend double computeIndicator(const ClientServerNetwork& network);

};

#endif // LOCALNETWORK\_H

#include "LocalNetwork.h"

// LocalNetwork

LocalNetwork::LocalNetwork(string name, double cost) : networkName(name), installationCost(cost) {}

LocalNetwork::~LocalNetwork() {}

// PeerToPeerNetwork

PeerToPeerNetwork::PeerToPeerNetwork(string name, double cost, int nodes)

: LocalNetwork(name, cost), numberOfNodes(nodes) {}

void PeerToPeerNetwork::showDetails() const {

cout << "Сеть: " << networkName << "\nТип: Одноранговая сеть\nКоличество узлов: " << numberOfNodes << "\nСтоимость: " << installationCost << endl;

}

double PeerToPeerNetwork::calculateInstallationCost() const {

return installationCost + numberOfNodes \* 500;

}

double computeIndicator(const PeerToPeerNetwork& network) {

return network.calculateInstallationCost() \* 1.1;

}

// ClientServerNetwork

ClientServerNetwork::ClientServerNetwork(string name, double cost, int clients)

: LocalNetwork(name, cost), numberOfClients(clients) {}

void ClientServerNetwork::showDetails() const {

cout << "Сеть: " << networkName << "\nТип: Сеть клиент-сервер\nКоличество клиентов: " << numberOfClients << "\nСтоимость: " << installationCost << endl;

}

double ClientServerNetwork::calculateInstallationCost() const {

return installationCost + numberOfClients \* 1000;

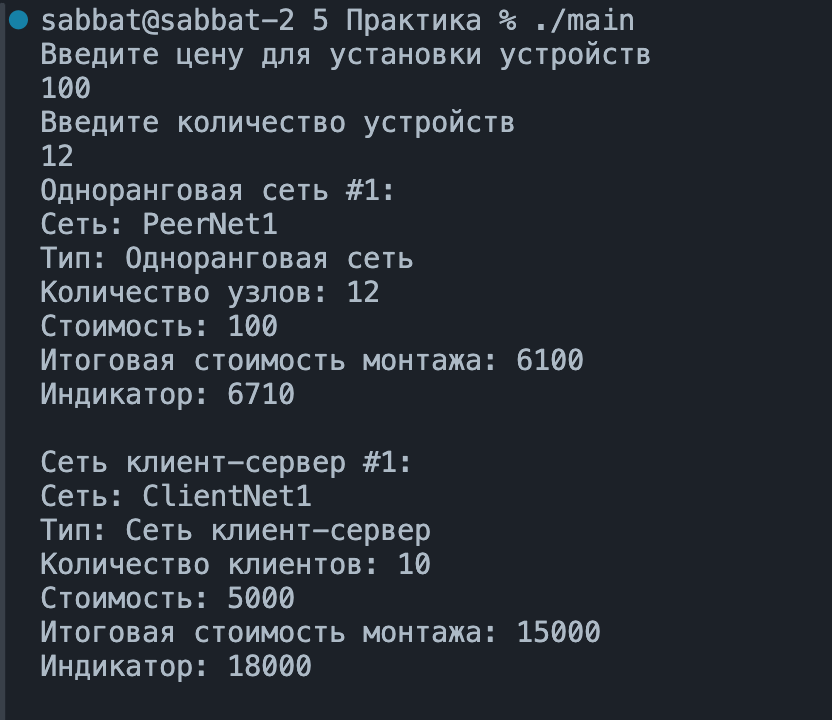
}

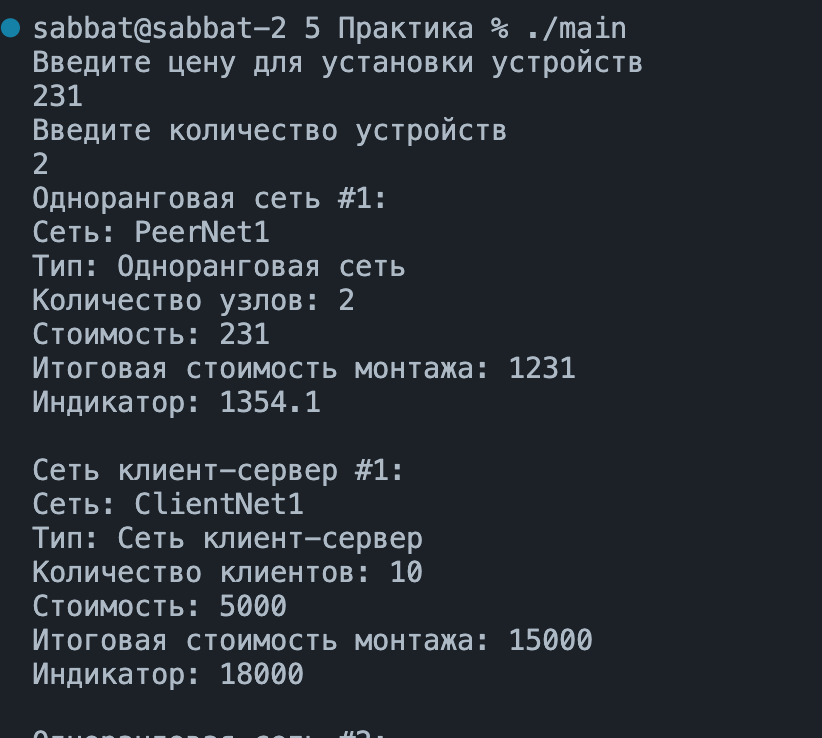
double computeIndicator(const ClientServerNetwork& network) {

return network.calculateInstallationCost() \* 1.2;

}

Результаты работы программы





**Выводы:**

1. **Наследование:** В программе реализованы два производных класса PeerToPeerNetwork и ClientServerNetwork, которые используют **private наследование** от базового класса LocalNetwork
2. **Вычисляемый показатель:** Для расчета вычисляемого показателя использована **отдельная функция** computeIndicator, которая не является дружественной к классам
3. **Динамическое выделение памяти:** Для создания объектов сетей с помощью оператора new[], параметры объектов инициализируются случайным образом через функцию rand().
4. **Проект в нескольких файлах:** Код разделен на несколько файлов:
   * Заголовочный файл LocalNetwork.h, где описаны все классы.
   * Реализационный файл LocalNetwork.cpp, где реализованы методы классов.
   * Основной файл main.cpp, где создаются объекты и выполняются основные операции.