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: Взаимодействие функций в программах**

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 <iostream>

#include "LocalNetwork.h" // Подключаем заголовочный файл с определением классов

using namespace std;

int main() {

    // Создание динамических массивов объектов производных классов

    const int arraySize = 5;

    PeerToPeerNetwork\* peerToPeerNetworks = new PeerToPeerNetwork[arraySize];

    ClientServerNetwork\* clientServerNetworks = new ClientServerNetwork[arraySize];

    // Заполнение массивов случайными значениями

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

        peerToPeerNetworks[i] = PeerToPeerNetwork("Peer Network " + to\_string(i+1), rand() % 5000 + 1000, rand() % 20 + 1);

        clientServerNetworks[i] = ClientServerNetwork("Client-Server Network " + to\_string(i+1), rand() % 5000 + 1000, rand() % 10 + 1);

    }

    // Вывод информации и расчет итоговой стоимости монтажа

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

        cout << "Детали одноранговой сети #" << (i+1) << ":" << endl;

        peerToPeerNetworks[i].showDetails();

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

        cout << "Детали сети типа клиент-сервер #" << (i+1) << ":" << endl;

        clientServerNetworks[i].showDetails();

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

    }

    // Освобождение памяти

    delete[] peerToPeerNetworks;

    delete[] clientServerNetworks;

    return 0; // Завершение программы

}

#ifndef LOCALNETWORK\_H // Защита от повторного включения

#define LOCALNETWORK\_H

#include <iostream>

#include <string>

using namespace std;

// Базовый класс для локальных сетей

class LocalNetwork {

protected:

    string networkName; // Название сети

    double installationCost; // Стоимость установки

public:

    LocalNetwork(string name, double cost); // Конструктор

    virtual void showDetails(); // Метод для отображения деталей сети

    virtual double calculateInstallationCost() = 0; // Чисто виртуальный метод для расчета стоимости установки

};

// Класс для одноранговых сетей

class PeerToPeerNetwork : private LocalNetwork {

private:

    int numberOfNodes; // Количество узлов

public:

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

    void showDetails(); // Метод для отображения деталей сети

    double calculateInstallationCost(); // Метод для расчета стоимости установки

};

// Класс для сетей типа клиент-сервер

class ClientServerNetwork : private LocalNetwork {

private:

    int numberOfClients; // Количество клиентов

public:

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

    void showDetails(); // Метод для отображения деталей сети

    double calculateInstallationCost(); // Метод для расчета стоимости установки

};

#endif // LOCALNETWORK\_H

#include "LocalNetwork.h"

// Конструктор для базового класса LocalNetwork

LocalNetwork::LocalNetwork(string name, double cost)

    : networkName(name), installationCost(cost) {}

// Метод для отображения деталей сети

void LocalNetwork::showDetails() {

    cout << "Сеть: " << networkName << endl;

    cout << "Стоимость монтажа: " << installationCost << " рублей" << endl;

}

// Конструктор для класса PeerToPeerNetwork

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

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

// Переопределённый метод для отображения деталей одноранговой сети

void PeerToPeerNetwork::showDetails() {

    LocalNetwork::showDetails(); // Вызов метода базового класса

    cout << "Тип сети: Одноранговая сеть: все узлы равны и могут выступать как клиент и сервер." << endl;

    cout << "Количество узлов: " << numberOfNodes << endl;

}

// Метод для расчета стоимости установки для одноранговой сети

double PeerToPeerNetwork::calculateInstallationCost() {

    return installationCost + numberOfNodes \* 500; // Стоимость монтажа + дополнительная стоимость за узлы

}

// Конструктор для класса ClientServerNetwork

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

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

// Переопределённый метод для отображения деталей сети типа клиент-сервер

void ClientServerNetwork::showDetails() {

    LocalNetwork::showDetails(); // Вызов метода базового класса

    cout << "Тип сети: Сеть типа клиент-сервер: один сервер обслуживает несколько клиентов." << endl;

    cout << "Количество клиентов: " << numberOfClients << endl;

}

// Метод для расчета стоимости установки для сети типа клиент-сервер

double ClientServerNetwork::calculateInstallationCost() {

    return installationCost + numberOfClients \* 1000; // Стоимость монтажа + дополнительная стоимость за клиентов

}

Результаты работы программы

Задание 1

- Одноранговая сеть:

- Ввод: Сеть 1, стоимость монтажа 2000 рублей, количество узлов 10.

- Вывод:

```

Детали одноранговой сети:

Сеть: Peer Network 1

Стоимость монтажа: 2000 рублей

Тип сети: Одноранговая сеть: все узлы равны и могут выступать как клиент и сервер.

Количество узлов: 10

Итоговая стоимость монтажа: 7000 рублей

```

- Сеть типа клиент-сервер:

- Ввод: Сеть 2, стоимость монтажа 3000 рублей, количество клиентов 5.

- Вывод:

```

Детали сети типа клиент-сервер:

Сеть: Client-Server Network 1

Стоимость монтажа: 3000 рублей

Тип сети: Сеть типа клиент-сервер: один сервер обслуживает несколько клиентов.

Количество клиентов: 5

Итоговая стоимость монтажа: 8000 рублей

```

Выводы по разработанным элементам программы

Программа успешно выполняет все поставленные задачи. В первом задании она корректно создает объекты классов `PeerToPeerNetwork` и `ClientServerNetwork`, инициализирует их параметры, вычисляет и выводит детали сети и итоговую стоимость монтажа.

Код организован в виде базового класса `LocalNetwork` и производных классов `PeerToPeerNetwork` и `ClientServerNetwork`, что делает его более структурированным и легким для понимания и отладки. Динамическое выделение памяти для массивов объектов и расчет вычисляемого показателя продемонстрированы правильно. Обработка ошибок и ввод данных выполнены корректно, что повышает надежность и удобство использования программы.