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 №6**

**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: Перегрузка операторов и функций

Цель:

Получение практических навыков перегрузки операторов и разработки перегруженных и виртуальных функций.

Задание на практическое занятие

Вариант A

1. Использовать структуру классов, спроектированную на практическом занятии №5. Наследование одного из производных классов осуществить по типу private. Перегрузить оператор для работы с объектами классов и использовать его для определения вычисляемого показателя.

2. Перегрузить функцию, определяющую вычисляемый показатель. В проект включить виртуальную функцию, выводящую на экран характеристики объекта каждого класса (кроме наследника с типом private). Добавить в проект статический член в каждый класс, обозначающий общее свойство для всех объектов класса.

3. Создать массивы объектов спроектированных классов и разработать меню для демонстрации содержимого проекта.

Код программы

main.cpp

#include <iostream>

#include "operations.h"

using namespace std;

int main() {

    const int arraySize = 5; // Размер массивов

    LocalNetwork\* networks[arraySize]; // Массив указателей на базовый класс

    // Создание объектов производных классов

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

        if (i % 2 == 0) {

            networks[i] = new PeerToPeerNetwork("Сеть P2P " + to\_string(i + 1), 2000 + i \* 100, rand() % 10 + 1);

        } else {

            networks[i] = new ClientServerNetwork("Сеть C/S " + to\_string(i + 1), 3000 + i \* 150, rand() % 5 + 1);

        }

    }

    // Меню для демонстрации содержимого проекта

    int choice;

    do {

        cout << "\nМеню:\n";

        cout << "1. Показать характеристики сетей\n";

        cout << "2. Расчитать стоимость установки сетей\n";

        cout << "0. Выход\n";

        cout << "Выберите опцию: ";

        cin >> choice;

        switch (choice) {

            case 1:

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

                    networks[i]->showDetails();

                }

                break;

            case 2:

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

                    cout << "Итоговая стоимость монтажа для " << networks[i]->getNetworkName() << ": "

                         << networks[i]->calculateInstallationCost() << " рублей" << endl;

                }

                break;

            case 0:

                cout << "Выход из программы." << endl;

                break;

            default:

                cout << "Неверный выбор. Пожалуйста, попробуйте снова." << endl;

        }

    } while (choice != 0);

    // Освобождение памяти

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

        delete networks[i];

    }

    return 0; // Завершение программы

}

operations.h

#ifndef OPERATIONS\_H

#define OPERATIONS\_H

#include <iostream>

#include <string>

using namespace std;

namespace Network {

class LocalNetwork {

protected:

    string networkName; // Название сети

    double installationCost; // Стоимость установки

    static int totalNetworks; // Общее количество объектов этого класса

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

    string getNetworkName() const; // Метод для получения названия сети

    static int getTotalNetworks(); // Статический метод для получения общего количества объектов

};

// 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

};

} // namespace Network

#endif // OPERATIONS\_H

operations.cpp

#include "operations.h"

using namespace std;

namespace Network {

// Инициализация статического члена

int LocalNetwork::totalNetworks = 0;

// Конструктор для базового класса LocalNetwork

LocalNetwork::LocalNetwork(string name, double cost)

    : networkName(name), installationCost(cost) {

    totalNetworks++; // Увеличиваем общее количество объектов

}

// Метод для отображения деталей сети

void LocalNetwork::showDetails() {

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

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

}

// Метод для получения названия сети

string LocalNetwork::getNetworkName() const {

    return networkName;

}

// Статический метод для получения общего количества объектов

int LocalNetwork::getTotalNetworks() {

    return totalNetworks;

}

// 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" << 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" << 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

Пример работы программы

Меню:

1. Показать характеристики сетей

2. Расчитать стоимость установки сетей

0. Выход

Выберите опцию: 1

Детали сетей:

Сеть: Сеть P2P 1

Стоимость монтажа: 2000 рублей

Тип сети: Одноранговая сеть

Количество узлов: 7

Сеть: Сеть C/S 1

Стоимость монтажа: 3150 рублей

Тип сети: Сеть типа клиент-сервер

Количество клиентов: 2

Выводы по разработанным элементам программы

Программа успешно выполняет все поставленные задачи. В первом задании она корректно создает массив объектов 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. Реализованы статические члены и перегрузка функций, а также виртуальные функции для демонстрации характеристик объектов, что улучшает модульность и гибкость кода.