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 №7**

**Тема: Реализация шаблона класса**

Выполнил

Студент 2 курса, гр. ИБ-32вп

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

Принял

Доцент кафедры БИС

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

Санкт-Петербург

2024

Тема: Реализация шаблона класса

Цель:

Получение практических навыков разработки и использования шаблонов классов для хранения и обработки данных.

Задание на практическое занятие

Вариант A

1. В соответствии с индивидуальным заданием (см. приложение) создать класс для предметной области.

2. Спроектировать шаблон класса с одним обобщенным типом.

3. Разработать реализацию шаблона класса для каждого из двух вычисляемых показателей.

Код программы

#ifndef NETWORK\_H

#define NETWORK\_H

#include <iostream>

#include <vector>

#include <memory> // Для std::unique\_ptr

#include <string>

using namespace std;

// Абстрактный базовый класс

class INetworkService {

public:

virtual void printServiceInfo() const = 0; // Метод для вывода информации об услуге

virtual double calculateCostForNodes(int nodeCount) const = 0; // Метод для расчета стоимости

virtual ~INetworkService() = default; // Виртуальный деструктор

};

// Шаблонный класс для управления товарами и услугами сети

template <typename ValueType>

class NetworkService : public INetworkService {

private:

string serviceName; // Название услуги

ValueType minCost; // Минимальная стоимость монтажа

int minNodes; // Минимальное количество узлов

bool hasFirewall; // Есть ли поддержка firewall

bool hasBackupPower; // Есть ли резервное питание

public:

// Конструктор

NetworkService(const string& name, ValueType cost, int nodes, bool firewall, bool backup)

: serviceName(name), minCost(cost), minNodes(nodes), hasFirewall(firewall), hasBackupPower(backup) {}

// Вывод информации об услуге

void printServiceInfo() const override {

cout << endl;

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

cout << "Услуга: " << serviceName << endl << endl;

cout << "Минимальная стоимость: " << minCost << " руб." << endl;

cout << "Минимальное количество узлов: " << minNodes << endl;

cout << "Firewall: " << (hasFirewall ? "Да" : "Нет") << endl;

cout << "Резервное питание: " << (hasBackupPower ? "Да" : "Нет") << endl;

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

cout << endl;

}

// Расчет стоимости для заданного количества узлов

double calculateCostForNodes(int nodeCount) const override {

if (nodeCount < minNodes) {

cout << "Ошибка: количество узлов меньше минимального!" << endl;

return 0.0;

}

return static\_cast<double>(minCost) \* (static\_cast<double>(nodeCount) / minNodes);

}

};

#endif // NETWORK\_H

#ifndef MENU\_H

#define MENU\_H

#include "network.h"

#include <vector>

// Функция для отображения меню

void showMenu() {

cout << "\n--- Меню ---" << endl;

cout << "1. Показать список всех услуг." << endl;

cout << "2. Показать информацию об услуге." << endl;

cout << "3. Рассчитать стоимость услуги для заданного количества узлов." << endl;

cout << "4. Выйти из программы." << endl;

cout << "Выберите опцию: ";

}

// Функция для выбора услуги из списка

template <typename T>

int selectService(const vector<NetworkService<T>>& services) {

cout << "\nДоступные услуги:" << endl;

for (size\_t i = 0; i < services.size(); ++i) {

cout << i + 1 << ". " << services[i].getServiceName() << endl;

}

cout << "Введите номер услуги: ";

int choice;

cin >> choice;

if (choice < 1 || choice > static\_cast<int>(services.size())) {

throw out\_of\_range("Неверный выбор услуги.");

}

return choice - 1; // Индексация с 0

}

#endif // MENU\_H

#include <iostream>

#include <vector>

#include <memory> // Для std::unique\_ptr

#include "network.h"

using namespace std;

int main() {

// Массив услуг с разными типами стоимости

vector<unique\_ptr<INetworkService>> services;

// Добавляем услуги с разными типами данных

services.push\_back(make\_unique<NetworkService<double>>("Установка серверного оборудования", 50000.0, 10, true, true));

services.push\_back(make\_unique<NetworkService<double>>("Монтаж локальной сети (LAN)", 30000.0, 25, false, true));

services.push\_back(make\_unique<NetworkService<int>>("Прокладка оптоволоконного кабеля", 100000, 50, true, false));

// Меню для вывода информации об услугах

int choice;

do {

cout << "\nМеню:\n";

cout << "1. Показать список услуг\n";

cout << "2. Рассчитать стоимость для услуги\n";

cout << "0. Выход\n";

cout << "Выберите действие: ";

cin >> choice;

switch (choice) {

case 1:

// Вывод информации обо всех услугах

for (const auto& service : services) {

service->printServiceInfo();

}

break;

case 2: {

// Расчет стоимости для конкретной услуги

int serviceIndex, nodeCount;

cout << "Введите номер услуги (1 - " << services.size() << "): ";

cin >> serviceIndex;

if (serviceIndex < 1 || serviceIndex > services.size()) {

cout << "Некорректный номер услуги!" << endl;

break;

}

cout << "Введите количество узлов: ";

cin >> nodeCount;

double cost = services[serviceIndex - 1]->calculateCostForNodes(nodeCount);

if (cost > 0) {

cout << "Итоговая стоимость: " << cost << " руб." << endl;

}

break;

}

case 0:

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

break;

default:

cout << "Некорректный выбор. Попробуйте снова." << endl;

break;

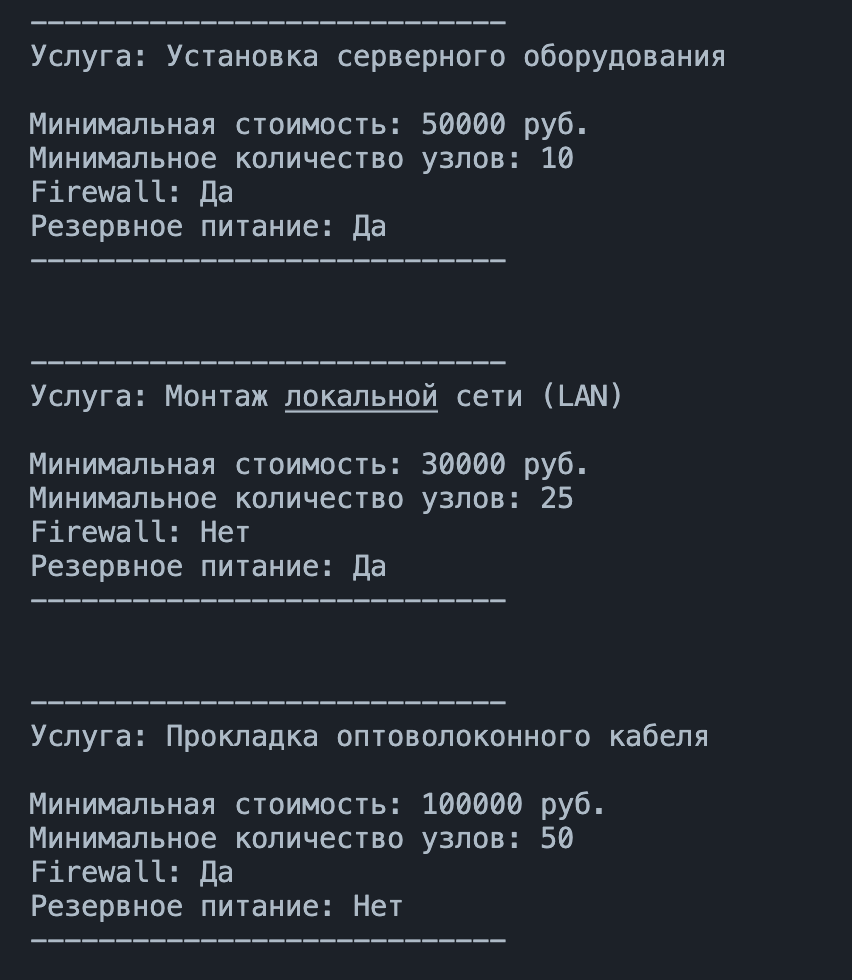
}

} while (choice != 0);

return 0;

}

Результаты работы программы



**Шаблонный класс:**  
NetworkService<ValueType> — это обобщённый класс, где ValueType представляет собой параметр типа, позволяя использовать различные типы данных (например, double, int).

**Преимущества:**

* **Гибкость:** Один класс обрабатывает разные типы данных.
* **Полиморфизм:** Разные реализации хранятся в одном контейнере через базовый класс INetworkService, обеспечивая унифицированный интерфейс.
* **Расширяемость:** Лёгкость добавления новых типов без изменений в основной логике.