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 №8**

**Тема: Обработка данных в динамических массивах**

Выполнил

Студент 2 курса, гр. ИБ-32вп

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

Принял

Доцент кафедры БИС

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

Санкт-Петербург

2024

Тема: STL. Обработка данных в динамических массивах

Цель:

Получение практических навыков разработки и анализа объектно-ориентированных программ обработки данных средствами STL.

Задание на практическое занятие

Вариант A

1. Для структуры классов, спроектированной на Практических занятиях №5 и №6, создать вектор из 1000 объектов какого-либо из производных классов. Затем создать второй вектор, в который занести элементы первого вектора в обратном порядке. Рассчитать Вычисляемый показатель.

2. Добавить в середину вектора 500 элементов и рассчитать Вычисляемый показатель. Очистить вектор и убедиться, что он пуст.

3. Разработать меню для демонстрации работы программы.

Код программы с комментариями

main.cpp

#include <iostream>

#include <vector>

#include "LocalNetwork.h"

using namespace std;

// Функция для демонстрации меню программы

void showMenu() {

    cout << "Меню программы:\n";

    cout << "1. Создать вектор из 1000 объектов\n";

    cout << "2. Добавить 500 элементов в середину вектора\n";

    cout << "3. Очистить вектор\n";

    cout << "4. Выйти\n";

}

int main() {

    vector<PeerToPeerNetwork> networks(1000, PeerToPeerNetwork("Network", 10000, 10));

    vector<PeerToPeerNetwork> reversedNetworks(networks.rbegin(), networks.rend());

    cout << "Вычисляемый показатель (суммарная стоимость установки в обратном векторе): "

         << reversedNetworks[0].calculateInstallationCost() \* reversedNetworks.size() << " рублей" << endl;

    networks.insert(networks.begin() + networks.size() / 2, 500, PeerToPeerNetwork("Inserted Network", 5000, 5));

    cout << "Вычисляемый показатель после добавления 500 элементов: "

         << networks[0].calculateInstallationCost() \* networks.size() << " рублей" << endl;

    networks.clear();

    if (networks.empty()) {

        cout << "Вектор очищен и пуст." << endl;

    }

    return 0;

}

LocalNetwork.h

#ifndef LOCALNETWORK\_H

#define LOCALNETWORK\_H

#include <iostream>

#include <string>

#include <vector>

using namespace std;

// Базовый класс для локальных сетей

class LocalNetwork {

protected:

    string networkName;   // Имя сети

    double installationCost;   // Стоимость установки

public:

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

    virtual double calculateInstallationCost() const = 0;

};

// Класс для одноранговой сети

class PeerToPeerNetwork : public LocalNetwork {

private:

    int numberOfNodes; // Количество узлов

public:

    PeerToPeerNetwork(string name, double cost, int nodes)

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

    double calculateInstallationCost() const override {

        return installationCost + (numberOfNodes \* 500);

    }

};

#endif #pragma once

#include <iostream>

#include <iomanip>

#include <string>

namespace Network {

class LocalNetwork {

protected:

std::string networkName;

double installationCost;

int networkSpeed;

static int totalNetworks;

public:

LocalNetwork(std::string name = "Default Network", double cost = 0.0, int speed = 0)

: networkName(name), installationCost(cost), networkSpeed(speed) {

++totalNetworks;

}

virtual ~LocalNetwork() { --totalNetworks; }

virtual void showDetails() const = 0;

virtual double calculateInstallationCost() const = 0;

virtual double calculateInstallationCost(double discount) const {

return installationCost \* (1 - discount);

}

static int getTotalNetworks() { return totalNetworks; }

std::string getNetworkName() const { return networkName; }

};

int LocalNetwork::totalNetworks = 0;

class PeerToPeerNetwork : public LocalNetwork {

private:

int numberOfNodes;

bool isEncrypted;

static std::string networkType;

public:

PeerToPeerNetwork(std::string name = "P2P Network", double cost = 0.0, int speed = 0, int nodes = 0, bool encrypted = false)

: LocalNetwork(name, cost, speed), numberOfNodes(nodes), isEncrypted(encrypted) {}

void showDetails() const override {

std::cout << std::setw(20) << networkName << std::setw(15) << installationCost

<< std::setw(15) << networkSpeed << std::setw(10) << numberOfNodes

<< std::setw(10) << (isEncrypted ? "Yes" : "No") << std::endl;

}

double calculateInstallationCost() const override {

return installationCost + numberOfNodes \* 10;

}

double calculateInstallationCost(double discount) const override {

return LocalNetwork::calculateInstallationCost(discount) + numberOfNodes \* 10;

}

static std::string getNetworkType() { return networkType; }

friend bool operator<(const PeerToPeerNetwork& lhs, const PeerToPeerNetwork& rhs) {

return lhs.calculateInstallationCost() < rhs.calculateInstallationCost();

}

};

std::string PeerToPeerNetwork::networkType = "Peer-to-Peer";

class ClientServerNetwork : public LocalNetwork { // Изменено с private на public

private:

int numberOfClients;

bool hasFirewall;

static std::string networkType;

public:

ClientServerNetwork(std::string name = "CS Network", double cost = 0.0, int speed = 0, int clients = 0, bool firewall = false)

: LocalNetwork(name, cost, speed), numberOfClients(clients), hasFirewall(firewall) {}

void showDetails() const override {

std::cout << std::setw(20) << networkName << std::setw(15) << installationCost

<< std::setw(15) << networkSpeed << std::setw(10) << numberOfClients

<< std::setw(10) << (hasFirewall ? "Yes" : "No") << std::endl;

}

double calculateInstallationCost() const override {

return installationCost + numberOfClients \* 20 + (hasFirewall ? 50 : 0);

}

double calculateInstallationCost(double discount) const override {

return LocalNetwork::calculateInstallationCost(discount) + numberOfClients \* 20 + (hasFirewall ? 50 : 0);

}

static std::string getNetworkType() { return networkType; }

friend bool operator<(const ClientServerNetwork& lhs, const ClientServerNetwork& rhs) {

return lhs.calculateInstallationCost() < rhs.calculateInstallationCost();

}

};

std::string ClientServerNetwork::networkType = "Client-Server";

} // namespace Network

#include <iostream>

#include <vector>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include "Network.h" // Подключение файла с описанием классов

// Функция для отображения меню

void displayMenu() {

std::cout << "\nМеню:\n";

std::cout << "1. Создать вектор из 1000 объектов\n";

std::cout << "2. Развернуть вектор\n";

std::cout << "3. Добавить 500 элементов в середину\n";

std::cout << "4. Очистить вектор и проверить, что он пуст\n";

std::cout << "5. Выйти\n";

}

// Функция для вывода первых и последних элементов вектора

void printNetworkDetails(const std::vector<Network::LocalNetwork\*>& networks) {

if (networks.empty()) {

std::cout << "Вектор пуст.\n";

return;

}

// Показать первые 5 элементов

std::cout << "\nПервые 5 элементов:\n";

size\_t limit = std::min(size\_t(5), networks.size());

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

std::cout << "Элемент " << i + 1 << ": ";

networks[i]->showDetails();

}

// Показать последние 5 элементов

std::cout << "\nПоследние 5 элементов:\n";

size\_t startIndex = networks.size() > 5 ? networks.size() - 5 : 0;

for (size\_t i = startIndex; i < networks.size(); ++i) {

std::cout << "Элемент " << i + 1 << ": ";

networks[i]->showDetails();

}

// Общее количество элементов

std::cout << "\nОбщее количество элементов в векторе: " << networks.size() << "\n";

}

// Главная функция

int main() {

std::vector<Network::LocalNetwork\*> networkVector;

bool running = true;

// Инициализация генератора случайных чисел

std::srand(std::time(0));

while (running) {

displayMenu();

int choice;

std::cin >> choice;

switch (choice) {

case 1: {

// Создание вектора из 1000 объектов (разных типов)

networkVector.clear();

for (int i = 0; i < 1000; ++i) {

if (i % 2 == 0) {

networkVector.push\_back(new Network::PeerToPeerNetwork(

"P2P Network " + std::to\_string(i),

rand() % 1000 + 500, rand() % 1000 + 100,

rand() % 100 + 1, rand() % 2 == 0));

} else {

networkVector.push\_back(new Network::ClientServerNetwork(

"CS Network " + std::to\_string(i),

rand() % 1000 + 500, rand() % 1000 + 100,

rand() % 50 + 1, rand() % 2 == 0));

}

}

std::cout << "Создано 1000 объектов.\n";

printNetworkDetails(networkVector);

break;

}

case 2: {

// Создание второго вектора с объектами в обратном порядке

std::vector<Network::LocalNetwork\*> reversedVector(networkVector.rbegin(), networkVector.rend());

std::cout << "Вектор развернут.\n";

printNetworkDetails(reversedVector);

break;

}

case 3: {

// Добавление 500 элементов в середину вектора

std::vector<Network::LocalNetwork\*> additionalNetworks;

for (int i = 0; i < 500; ++i) {

if (i % 2 == 0) {

additionalNetworks.push\_back(new Network::PeerToPeerNetwork(

"P2P Network Mid " + std::to\_string(i),

rand() % 1000 + 500, rand() % 1000 + 100,

rand() % 100 + 1, rand() % 2 == 0));

} else {

additionalNetworks.push\_back(new Network::ClientServerNetwork(

"CS Network Mid " + std::to\_string(i),

rand() % 1000 + 500, rand() % 1000 + 100,

rand() % 50 + 1, rand() % 2 == 0));

}

}

// Добавление элементов в середину

networkVector.insert(networkVector.begin() + networkVector.size() / 2, additionalNetworks.begin(), additionalNetworks.end());

// Информируем пользователя

std::cout << "500 новых элементов добавлено в середину вектора.\n";

std::cout << "Вот несколько примеров новых элементов:\n";

// Показать первые 5 добавленных элементов

for (size\_t i = networkVector.size() / 2; i < std::min(networkVector.size() / 2 + 5, networkVector.size()); ++i) {

std::cout << "Элемент " << i + 1 << ": ";

networkVector[i]->showDetails();

}

printNetworkDetails(networkVector);

break;

}

case 4: {

// Очистка вектора

networkVector.clear();

std::cout << "Вектор очищен.\n";

break;

}

case 5: {

// Выход из программы

running = false;

std::cout << "Выход из программы.\n";

break;

}

default:

std::cout << "Неверный выбор, попробуйте снова.\n";

}

}

// Очистка памяти

for (auto& network : networkVector) {

delete network;

}

return 0;

}

Результаты работы программы

1. **Создание вектора из 1000 объектов:**
   * Вектор заполняется чередующимися объектами типов PeerToPeerNetwork и ClientServerNetwork с случайными параметрами.
2. **Развертывание вектора:**
   * Вектор копируется в обратном порядке с помощью итераторов rbegin() и rend().
3. **Добавление 500 элементов в середину вектора:**
   * 500 новых объектов добавляются в центр вектора с помощью метода insert().
4. **Очистка вектора:**
   * Вектор очищается методом clear(), после чего проверяется его пустота.