МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт Информационных Технологий

Кафедра МПО ЭВМ

Дисциплина «Объектно-ориентированное программирование»

Лабораторная работа №4

«Шаблоны»

Выполнил:

студент группы 1ПИб-02-3оп-22

Маркелов Сергей Александрович

Проверил:

Табунов Павел Александрович

Череповец, 2023 год

**Цель работы:**

* Изучить способы определения шаблонов, принцип и механизм создания шаблонных классов и шаблонов функций
* Получить практические навыки работы с шаблонными классами.

**Требования к разработке:**

1. Запрещается использовать обработку исключительных ситуаций и генерировать исключения.
2. Придерживаться принципа DRY (Don’t repeat yourself).
3. Обязательно наличие комментариев.
4. Обязательно сделать шаблонным класс-контейнер.

**Задание на лабораторную работу:**

Модифицировать абстрактные типы данных, реализованные по заданию лабораторной работы №3, используя шаблоны определения класса и шаблоны определения функции.

Обязательно сделать шаблонным класс контейнер. Шаблоны определения функции на свое усмотрение.

Проверьте работоспособность АТД на тестовом наборе данных.

**Текст программы**

Заголовочный файл **Queue.h**:

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

using namespace std;

template <class T>

class Queue {

private:

struct Node { //узел очереди

T data;

Node\* next;

};

Node\* head;

Node\* tail;

int size = 0;

public:

Queue() { //конструктор

head = NULL;

tail = NULL;

}

bool empty() { //проверка очереди на пустоту

return head == NULL;

}

void add(T value) { //добавление элемента в очередь

if (empty()) { //если она пустая

head = new Node;

head->data = value;

head->next = NULL;

tail = head;

size++;

}

else { //если она не пустая

tail->next = new Node;

tail = tail->next;

tail->data = value;

tail->next = NULL;

size++;

}

}

T del() { //взятие и удаление элемента из очереди

if (empty()) {

cout << "Queue is empty" << endl;

}

else {

T d = head->data;

Node\* tmp = head;

head = head->next;

delete(tmp);

return d;

}

}

void nullQueue() { //обнуление очереди

Node\* tmp;

while (!empty()) {

tmp = head;

head = head->next;

delete(tmp);

}

}

bool operator >= (const Queue& other) {

return this->size >= other.size;

}

bool operator < (const Queue& other) {

return this->size < other.size;

}

int getSize() { //геттер для логина

return size;

}

friend ostream& operator<<(ostream& del, Queue& q) {

while (!q.empty())

del << q.del() << endl;

return del;

}

void arr(T\* a, int& i) {

while (!empty()) {

a[i] = del();

i++;

}

}

void print\_NewOld() {

string\* networks = new string[size];

int i = 0;

arr(networks, i);

i--;

for (i; i >= 0; i--)

cout << networks[i] << endl;

}

void print\_AZ() {

string\* networks = new string[size];

int i = 0;

arr(networks, i);

string k;

for (i = 0; i < size; i++)

for (int j = i; j < size; j++) {

if (networks[i] > networks[j]) {

k = networks[i];

networks[i] = networks[j];

networks[j] = k;

}

}

for (i = 0; i < size; i++)

cout << networks[i] << endl;

}

void print\_ZA() {

string\* networks = new string[size];

int i = 0;

arr(networks, i);

string k;

for (i = 0; i < size; i++)

for (int j = i; j < size; j++) {

if (networks[i] < networks[j]) {

k = networks[i];

networks[i] = networks[j];

networks[j] = k;

}

}

for (i = 0; i < size; i++)

cout << networks[i] << endl;

}

void search(string name) {

string\* networks = new string[size];

int i = 0;

arr(networks, i);

int k2 = 0;

for (i = 0; i < size; i++) {

int k1 = 0;

string s;

for (int l = 0; l < name.length(); l++) {

s = networks[i];

if (s[l] == name[l]) k1++;

}

if (k1 == name.length()) {

if (k2 == 0) cout << endl << "Найденные сети:" << endl;

cout << networks[i] << endl;

k2++;

}

}

if (k2 == 0) cout << "Ничего не найдено!" << endl;

}

void search(int number) {

int\* networks = new int[size];

int i = 0, k = 0;

arr(networks, i);

for (i = 0; i < size; i++) {

if (networks[i] == number) k++;

}

if (k == 0) cout << "Подключений к сети №" << number << " не было" << endl;

else cout << "К сети №" << number << " подключались " << k << " раз(а)" << endl;

}

~Queue() { //деструктор

}

};

Заголовочный файл **Header.h**:

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

using namespace std;

enum security { WEP, WPA, WPA2, WPA3, NoProtection };

class WiFi {

private:

string login;

string password;

static string work;

security protection;

int speed, radius, traffic;

double frequency, time;

public:

bool operator > (const WiFi& other) {

return this->speed > other.speed;

}

bool operator < (const WiFi& other) {

return this->speed < other.speed;

}

bool operator == (const WiFi& other) {

return this->speed == other.speed;

}

int operator + (const WiFi& other) {

return this->traffic + other.traffic;

}

WiFi(string valueLogin, string valuePassword, security valueProtection, int valueSpeed, int valueRadius, double valueFrequency) {

this->login = valueLogin; //конструктор со всеми параметрами

this->password = valuePassword;

this->protection = valueProtection;

this->speed = valueSpeed;

this->radius = valueRadius;

this->frequency = valueFrequency;

this->work = "Да";

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi() {

this->login = "Неопределенная сеть"; //конструктор по умолчанию

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->work = "Да";

this->traffic = 0;

this->time = rand() / 100.0;

}

WiFi(string valueLogin) {

this->login = valueLogin; //конструктор с 1 параметром

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->work = "Да";

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi(double valueFrequency, string valueLogin) {

this->login = valueLogin; //конструктор с 2 параметрами

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = valueFrequency;

this->work = "Да";

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi(int valueSpeed, int valueRadius, string valueLogin) {

this->login = valueLogin; //конструктор с 3 параметрами

this->password = "0";

this->protection = NoProtection;

this->speed = valueSpeed;

this->radius = valueRadius;

this->frequency = 0;

this->work = "Да";

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi(const WiFi& obj) {

this->login = obj.login; //копирующий конструктор

this->password = obj.password;

this->protection = obj.protection;

this->speed = obj.speed;

this->radius = obj.radius;

this->frequency = obj.frequency;

this->work = "Да";

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

string getLogin() { //геттер для логина

return login;

}

string getPassword() { //геттер для пароля

return password;

}

int getTraffic() {

return traffic;

}

double getTime() {

return time;

}

string getProtection() { //геттер для протокола безопасности

switch (protection) {

case WEP: return "WEP";

case WPA: return "WPA";

case WPA2: return "WPA2";

case WPA3: return "WPA3";

case NoProtection: return "Без защиты";

default: return "N/A";

}

}

template<class T>

T sum(T a, T b) {

return a + b;

}

friend ostream& operator<<(ostream& info, WiFi& other) { //вывод информации о сети

info << "Вы успешно подключились к WiFi-сети " << other.login << endl << endl;

info << "Информация о сети" << endl;

info << "Скорость: " << other.speed << " Мбит/с" << endl;

info << "Частота: " << other.frequency << " ГГц" << endl;

info << "Радиус покрытия: " << other.radius << " м" << endl;

info << "Протокол безопасности: " << other.getProtection() << endl;

info << "Cеть работает: " << other.work << endl;

info << "Передано данных: " << other.traffic << " Мб" << endl;

info << "Время соединения: " << other.time << " сек" << endl;

info << endl << "=======================" << endl << endl;

return info;

}

~WiFi() { //деструктор

}

};

Файл с кодом **лб1.cpp**:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include "Header.h"

#include "Queue.h"

using namespace std;

string WiFi::work;

void NetworkError(int err) { //функция вывода ошибок

switch (err) {

case 1: cout << "Такой WiFi-сети нет!";

break;

case 2: cout << "Пароль неверный!" << endl;

break;

case 3: cout << "Такого варианта нет!" << endl;

break;

}

}

void authorization(int number, int& traffic, double& time, string password, WiFi& Network, Queue<string>& q, Queue<string>& error, Queue<int>& q\_num) {

if (password == Network.getPassword()) { //если пароль верный, то выводится информация о сети

q.add(Network.getLogin());

q\_num.add(number);

cout << Network;

traffic = Network.sum(traffic, Network.getTraffic());

time = Network.sum(time, Network.getTime());

}

else {

NetworkError(2); //если пароль неверный, выводится сообщение об ошибке

error.add(Network.getLogin());

}

}

int main() {

setlocale(LC\_ALL, "Russian");

srand(time(0));

int number, traffic = 0;

double time = 0;

string password, name;

Queue<string> q, error;

Queue<int> q\_num;

while (true) {

WiFi Network\_1("CHSU\_FREE", "0", security::NoProtection, 100, 70, 5); //описание доступных WiFi-сетей

WiFi Network\_2("CHSU\_WORK", "1234567890", security::WPA3, 120, 50, 5);

WiFi Network\_3("IPhone", "ghfdgldklkl", security::WPA2, 80, 40, 2.4);

WiFi Network\_4("Xiaomi", "h5hjg54g5", security::WPA2, 70, 40, 2.4);

WiFi Network\_5;

WiFi Network\_6(Network\_1);

WiFi Network\_7(2.4, "Rostelecom");

WiFi Network\_8(100, 60, "Beeline");

WiFi Network\_9("iZet");

cout << "Доступные сети: " << endl; //вывод доступных сетей на экран

cout << "1. " << Network\_1.getLogin() << endl;

cout << "2. " << Network\_2.getLogin() << endl;

cout << "3. " << Network\_3.getLogin() << endl;

cout << "4. " << Network\_4.getLogin() << endl;

cout << "5. " << Network\_5.getLogin() << endl;

cout << "6. " << Network\_6.getLogin() << endl;

cout << "7. " << Network\_7.getLogin() << endl;

cout << "8. " << Network\_8.getLogin() << endl;

cout << "9. " << Network\_9.getLogin() << endl << endl;

cout << "0. " << "Вывести историю подключения к сетям" << endl;

if (q.getSize() == 0 && error.getSize() == 0) {

cout << endl << "Рекомендации: " << endl;

cout << "Выбирая между сетями " << Network\_1.getLogin() << " и " << Network\_2.getLogin() << " рекомендуем выбрать сеть ";

if (Network\_1 > Network\_2) cout << Network\_1.getLogin();

else if (Network\_1 < Network\_2) cout << Network\_2.getLogin();

cout << ", т.к. её скорость выше" << endl;

if (Network\_5 == Network\_9) cout << "Скорость у сетей " << Network\_5.getLogin() << " и " << Network\_9.getLogin() << " одинакова" << endl;

if (Network\_3 == Network\_4) cout << "Скорость у сетей " << Network\_3.getLogin() << " и " << Network\_4.getLogin() << " одинакова" << endl;

cout << "С помощью сетей " << Network\_7.getLogin() << " и " << Network\_8.getLogin() << " можно передать " << Network\_7 + Network\_8 << " Мб данных" << endl;

}

cout << endl << "Введите номер сети, к которой хотите подключиться: "; //выбор сети

cin >> number;

if (number < 0 || number > 9) { //если введен номер несуществуещей сети, выводится сообщение об ошибке

NetworkError(1);

return -1;

}

else if (number == 0 && !q.empty()) {

int num;

if (q >= error) cout << endl << "Было совершено " << q.getSize() << " удачных подключений и " << error.getSize() << " неудачных" << endl;

if (q < error) cout << endl << "Было совершено " << error.getSize() << " неудачных подключений и " << q.getSize() << " удачных" << endl;

cout << "Всего было передано " << traffic << " Мб данных" << endl;

cout << "Общее время соединения: " << time << " сек" << endl;

cout << endl << "Порядок вывода:" << endl; //вывод доступных сетей на экран

cout << "1. " << "В хронологическом (от старых подключений к новым)" << endl;

cout << "2. " << "В хронологическом (от новых подключений к старым)" << endl;

cout << "3. " << "В алфавитном (A - Z)" << endl;

cout << "4. " << "В обратном алфавитному (Z - A)" << endl;

cout << "5. " << "Поиск по названию" << endl;

cout << "6. " << "Поиск по номеру" << endl;

cout << endl << "Выберите порядок: ";

cin >> num;

if (num < 1 || num > 6) {

NetworkError(3);

return -1;

}

if (num >= 1 && num <= 4) cout << endl << "История подключений: " << endl;

switch (num) {

case 1: cout << q;

break;

case 2: q.print\_NewOld();

break;

case 3: q.print\_AZ();

break;

case 4: q.print\_ZA();

break;

case 5: {

cout << endl << "Введите название сети: ";

cin >> name;

cin.get();

q.search(name);

}

break;

case 6: {

cout << endl << "Доступные сети: " << endl; //вывод доступных сетей на экран

cout << "1. " << Network\_1.getLogin() << endl;

cout << "2. " << Network\_2.getLogin() << endl;

cout << "3. " << Network\_3.getLogin() << endl;

cout << "4. " << Network\_4.getLogin() << endl;

cout << "5. " << Network\_5.getLogin() << endl;

cout << "6. " << Network\_6.getLogin() << endl;

cout << "7. " << Network\_7.getLogin() << endl;

cout << "8. " << Network\_8.getLogin() << endl;

cout << "9. " << Network\_9.getLogin() << endl;

cout << endl << "Введите номер сети: ";

cin >> number;

cin.get();

q\_num.search(number);

}

break;

}

return 0;

}

else if (number == 0 && q.empty()) {

cout << "Подключений не было" << endl;

return 0;

}

cout << "Введите пароль: "; //ввод пароля

cin >> password;

cin.get();

switch (number) {

case 1: authorization(number, traffic, time, password, Network\_1, q, error, q\_num);

break;

case 2: authorization(number, traffic, time, password, Network\_2, q, error, q\_num);

break;

case 3: authorization(number, traffic, time, password, Network\_3, q, error, q\_num);

break;

case 4: authorization(number, traffic, time, password, Network\_4, q, error, q\_num);

break;

case 5: authorization(number, traffic, time, password, Network\_5, q, error, q\_num);

break;

case 6: authorization(number, traffic, time, password, Network\_6, q, error, q\_num);

break;

case 7: authorization(number, traffic, time, password, Network\_7, q, error, q\_num);

break;

case 8: authorization(number, traffic, time, password, Network\_8, q, error, q\_num);

break;

case 9: authorization(number, traffic, time, password, Network\_9, q, error, q\_num);

break;

}

}

system("pause");

return 0;

}

**Описание программы**

В программе был реализован шаблон определения класса **Queue**

Благодаря этому элементами очереди теперь могут становиться любые типы данных, а не только строки типа **string**, как это было ранее.

Это позволило записывать историю номеров сетей в очередь **q\_num**, а не в массив **history**, как это было ранее. Также благодаря исключению из программы массива удалось избавиться от переменных **count** и **capacity**. Код был упрощен.

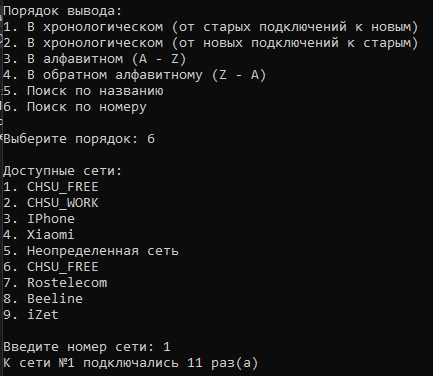
Также в классе **WiFi** был реализован шаблон определения функции **sum**.

Функция может складывать любые значения, вне зависимости от типа данных.

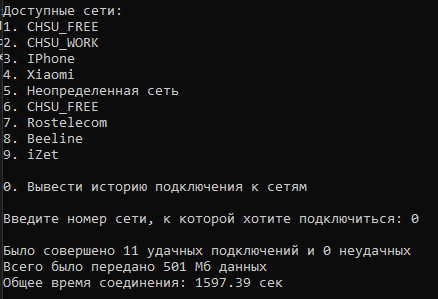
Шаблонная функция считает общий объем переданных данных (тип **int**) и общее время подключения (тип **double**).

**Результаты тестов**

Поиск сетей по номеру (реализован) с использованием шаблонного класса **Queue**:



Вывод информации о количестве переданных данных и времени соединения (реализовано с помощью шаблонной функции **sum**):



**Контрольные вопросы**

1. *Для чего используется ключевое слово template?*

Ключевое слово **template** используется для создания шаблонов. Шаблоны представляют собой механизм, который позволяет писать обобщенный код, который может работать с разными типами данных, без необходимости повторного написания кода для каждого конкретного типа.

1. *В чем заключаются особенности параметров по умолчанию для шаблонов?*

Особенность параметров по умолчанию для шаблонов заключается в том, что они обеспечивают более гибкое использование шаблонов, позволяя указывать значения по умолчанию для типов и значений. Это значит, что при использовании шаблона можно не указывать некоторые параметры, и компилятор будет использовать значения по умолчанию.

Например:

**template <class T = int, int n = 5>**

Этот шаблон можно использовать, не указывая явно все его параметры: тогда тип **T** примет значение **int**, а параметр **n** примет значение **5**. Это делает код более компактным и удобным для использования.

1. *Для чего используются шаблоны функций?*

Шаблоны функций в C++ предоставляют механизм обобщения кода, который позволяет писать функции, способные работать с различными типами данных, не требуя дублирования кода для каждого типа. Поскольку шаблоны функций обеспечивают обобщенность, код становится более переиспользуемым, соблюдается принцип DRY.

Шаблоны функций обеспечивают гибкость в работе с различными типами данных, сохраняя при этом структуру и логику функции.

1. *Назовите разновидности дружественных функций шаблонного класса.*

В шаблонном классе могут быть следующие разновидности дружественных функций:

* Шаблонные дружественные функции (имеют доступ ко всем типам данных класса):

**template <class T>**

**class WiFi {**

**template <class U>**

**friend void Network(WiFi<U>& obj);**

**};**

* Частично специализированные дружественные функции (предназначены для конкретного типа U, имеют доступ ко всем полям класса):

**template <class T>**

**class WiFi {**

**template <class U>**

**friend void Network(WiFi<U>& obj);**

**};**

**template <class U>**

**void Nwtwork(WiFi<U>& obj) {**

**…**

**}**

* Дружественные функции без шаблона:

**template <class T>**

**class WiFi {**

**friend void Network(WiFi<T>& obj) {**

**…**

**}**

**};**

1. *Дайте определение специализации шаблона.*

Специализация шаблона – это возможность предоставить конкретную реализацию шаблонного кода для определенных аргументов типов или значений, отличных от общей реализации. Специализация шаблона позволяет оптимизировать код для конкретных случаев или обеспечивать специфическое поведение для определенных типов данных или значений параметров.

* Полная специализация:

**template <>**

**class WiFi<int> {**

**…**

**};**

* Частичная специализация:

**template <class T, class U>**

**class WiFi<T U> {**

**…**

**};**

**Вывод**

В ходе выполнения лабораторной работы были изучены способы определения шаблонов, принцип и механизм создания шаблонных классов и шаблонов функций. Также были получены практические навыки работы с шаблонными классами.

Были созданы шаблоны определения класса и шаблоны определения функции.

Шаблонным был сделан класс-контейнер. Также был создан шаблон определения функции.