МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт Информационных Технологий

Кафедра МПО ЭВМ

Дисциплина «Объектно-ориентированное программирование»

Лабораторная работа №2

«Простое наследование»

Выполнил:

студент группы 1ПИб-02-3оп-22

Маркелов Сергей Александрович

Проверил:

Кустов Максим Александрович

Череповец, 2024 год

**Цель работы:**

* изучить способы создания производного класса и особенности работы с ним;
* изучить правила инициализации и доступа к элементам производного класса;
* приобрести практические навыки наследования.

**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. Разрешается использовать шаблонный класс-контейнер, либо следует заменить хранимый тип объекта на указатель на класс-интерфейс.
5. Допускается заменить хранимый тип объекта на обобщенный указатель (void\*).

**Задание на лабораторную работу:**

Создать производный класс для АТД, реализованного по заданию предыдущих лабораторных работ, используя одиночное наследование.

Проверить работоспособность АТД и производного класса 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 **WiFi.h**:

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

using namespace std;

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

class Site;

class WiFi {

private:

string password;

security protection;

protected:

string name;

static string work;

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 valueName, string valuePassword, security valueProtection, int valueSpeed, int valueRadius, double valueFrequency) {

this->name = valueName; //конструктор со всеми параметрами

this->password = valuePassword;

this->protection = valueProtection;

this->speed = valueSpeed;

this->radius = valueRadius;

this->frequency = valueFrequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi() {

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

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = 0;

this->time = rand() / 100.0;

}

WiFi(string valueName) {

this->name = valueName; //конструктор с 1 параметром

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi(double valueFrequency, string valueName) {

this->name = valueName; //конструктор с 2 параметрами

this->password = "0";

this->protection = NoProtection;

this->speed = 0;

this->radius = 0;

this->frequency = valueFrequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

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

this->name = valueName; //конструктор с 3 параметрами

this->password = "0";

this->protection = NoProtection;

this->speed = valueSpeed;

this->radius = valueRadius;

this->frequency = 0;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

WiFi(const WiFi& obj) {

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

this->password = obj.password;

this->protection = obj.protection;

this->speed = obj.speed;

this->radius = obj.radius;

this->frequency = obj.frequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

}

string getName() { //геттер для логина

return name;

}

string getPassword() { //геттер для пароля

return password;

}

int getTraffic() {

return traffic;

}

double getTime() {

return time;

}

int getSpeed() {

return speed;

}

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.name << 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;

}

void openSite(Site& site); //функция, взаимодействующая с объектом класса Site

~WiFi() { //деструктор

}

};

Заголовочный файл **Site.h**:

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

using namespace std;

enum protocol { http, https, NoProtocol };

class WiFi;

class Site {

private:

string name, type, address;

static string work;

double time;

protocol connect;

public:

Site(string valueName, string valueType, string valueAddress, protocol valueConnect) {

this->name = valueName; //конструктор со всеми параметрами

this->type = valueType;

this->address = valueAddress;

this->connect = valueConnect;

this->time = rand() / 1000.0;

}

Site() {

this->name = "Неизвестный сайт"; //конструктор по умолчанию

this->type = "Неизвестный тип";

this->address = "about:blank";

this->connect = NoProtocol;

this->time = rand() / 1000.0;

}

Site(string valueAddress) {

this->name = "Неизвестный сайт"; //конструктор c 1 параметром

this->type = "Неизвестный тип";

this->address = valueAddress;

this->connect = NoProtocol;

this->time = rand() / 1000.0;

}

Site(string valueAddress, protocol valueConnect) {

this->name = "Неизвестный сайт"; //конструктор c 2 параметрами

this->type = "Неизвестный тип";

this->address = valueAddress;

this->connect = valueConnect;

this->time = rand() / 1000.0;

}

Site(string valueName, string valueAddress, protocol valueConnect) {

this->name = "Неизвестный сайт"; //конструктор c 3 параметрами

this->type = "Неизвестный тип";

this->address = valueAddress;

this->connect = valueConnect;

this->time = rand() / 1000.0;

}

Site(const Site& obj) {

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

this->type = obj.type;

this->address = obj.address;

this->connect = obj.connect;

this->time = rand() / 1000.0;

}

string getName() { //геттер для названия сайта

return name;

}

string getWork() { //геттер для статуса работы сайта

return work;

}

double getTime() { //геттер для времени загрузки сайта

return time;

}

friend ostream& operator<<(ostream& info, Site& other) {

switch (other.connect) {

case NoProtocol: info << other.name << " - " << other.address << " (" << other.type << ")";

break;

case http: info << other.name << " - " << " http://" << other.address << " (" << other.type << ")";

break;

case https: info << other.name << " - " << " https://" << other.address << " (" << other.type << ")";

break;

}

return info;

}

void status(WiFi& wifi); //функция, взаимодействующая с объектом класса WiFi

~Site() { //деструктор

}

};

Заголовочный файл **Mobile.h**:

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include "WiFi.h"

using namespace std;

enum generation { GPRS, Edge, \_2G, \_3G, \_4G, \_5G, unknown };

class Mobile : public WiFi {

private:

generation type;

int signal;

public:

Mobile(string valueName, generation valueType, int valueSpeed, int valueRadius, double valueFrequency) {

this->name = valueName; //конструктор со всеми параметрами

this->type = valueType;

this->speed = valueSpeed;

this->radius = valueRadius;

this->frequency = valueFrequency;

this->traffic = rand() % 100;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

Mobile() {

this->name = "Неопределенный оператор"; //конструктор по умолчанию

this->type = unknown;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = 0;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

Mobile(string valueName) {

this->name = valueName; //конструктор с 1 параметром

this->type = unknown;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = rand() / 100.0;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

Mobile(string valueName, generation valueType) {

this->name = valueName; //конструктор с 2 параметрами

this->type = valueType;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = rand() / 100.0;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

Mobile(string valueName, generation valueType, int valueSpeed) {

this->name = valueName; //конструктор с 3 параметрами

this->type = unknown;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = rand() / 100.0;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

Mobile(const Mobile& obj) {

this->name = obj.name; //конструктор с 3 параметрами

this->type = obj.type;

this->speed = obj.speed;

this->radius = obj.radius;

this->frequency = obj.frequency;

this->traffic = rand() / 100.0;

this->time = rand() / 100.0;

this->signal = rand() % 5;

}

string getType() { //геттер для поколения сети

switch (type) {

case GPRS: return "GPRS";

case Edge: return "Edge";

case \_2G: return "2G";

case \_3G: return "3G";

case \_4G: return "4G";

case \_5G: return "5G";

case unknown: return "unknown";

default: return "N/A";

}

}

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

info << "Вы успешно подключились к мобильной сети " << other.name << endl << endl;

info << "Информация о сети" << endl;

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

info << "Уровень сигнала: " << other.signal << endl;

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

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

info << "Поколение сети: " << other.getType() << 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;

}

~Mobile() { //деструктор

}

};

Файл с кодом **лб1.cpp**:

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include "WiFi.h"

#include "Queue.h"

#include "Site.h"

#include "Mobile.h"

using namespace std;

string WiFi::work = "Да";

string Site::work = "Да";

void WiFi::openSite(Site& site) { //функция класса WiFi, взаимодействующая с объектом класса Site

cout << "Сообщение от провайдера: Вы успешно зашли на сайт " << site.getName() << " через сеть " << getName() << endl;

cout << "Сайт работает: " << site.getWork() << endl;

cout << "Время загрузки сайта: " << site.getTime() << " мс" << endl << endl;

}

void Site::status(WiFi& wifi) { //функция класса Site, взаимодействующая с объектом класса WiFi

cout << "Сообщение от сайта: Подключение к сайту " << getName() << " осуществляется через сеть " << wifi.getName() << endl;

cout << "Скорость сети: " << wifi.getSpeed() << " Мбит/с" << endl << endl;

}

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

switch (err) {

case 1: cout << "Такой сети нет!" << endl << endl;

break;

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

break;

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

break;

case 4: cout << "Такого сайта нет!" << endl << endl;

}

}

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

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

WiFi Network\_Connected(Network);

q.add(Network.getName());

q\_num.add(number);

correct++;

cout << Network;

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

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

return Network\_Connected;

}

else {

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

WiFi Network\_Disconnected(Network);

error.add(Network.getName());

return Network\_Disconnected;

}

}

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

Mobile Mobile\_Connected(Network);

q.add(Network.getName());

q\_num.add(number);

correct++;

cout << Network;

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

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

return Mobile\_Connected;

}

template<typename T>

void login(T& Network, Site& website) {

Network.openSite(website);

website.status(Network);

}

int main() {

setlocale(LC\_ALL, "Russian");

srand(time(0));

int number, number\_site, count = 0, traffic = 0, correct = 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");

Mobile Mobile\_1("MTS RUS", generation::\_5G, 70, 100, 5); //описание доступных мобильных сетей

Mobile Mobile\_2("Beeline", generation::\_2G, 50, 90, 5);

Mobile Mobile\_3("MegaFon", generation::\_4G, 80, 100, 5);

Mobile Mobile\_4("Tele2", generation::\_3G, 50, 80, 5);

Mobile Mobile\_5;

Mobile Mobile\_6(Mobile\_1);

Mobile Mobile\_7("Yota");

Mobile Mobile\_8("Tinkoff Mobile", generation::\_3G);

Mobile Mobile\_9("Мотив", generation::GPRS, 30);

Site Site\_1("Google", "Поисковик", "www.google.com", protocol::https);

Site Site\_2("Яндекс", "Поисковик", "www.ya.ru", protocol::https);

Site Site\_3("ВКонтакте", "Социальная сеть", "www.vk.com", protocol::https);

Site Site\_4("YouTube", "Видеохостинг", "www.youtube.com", protocol::https);

Site Site\_5;

Site Site\_6(Site\_1);

Site Site\_7("example.com");

Site Site\_8("www.edu.chsu.ru", protocol::http);

Site Site\_9("ЧГУ", "www.chsu.ru", protocol::https);

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

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

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

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

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

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

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

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

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

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

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

cout << "10. " << Mobile\_1.getName() << endl;

cout << "11. " << Mobile\_2.getName() << endl;

cout << "12. " << Mobile\_3.getName() << endl;

cout << "13. " << Mobile\_4.getName() << endl;

cout << "14. " << Mobile\_5.getName() << endl;

cout << "15. " << Mobile\_6.getName() << endl;

cout << "16. " << Mobile\_7.getName() << endl;

cout << "17. " << Mobile\_8.getName() << endl;

cout << "18. " << Mobile\_9.getName() << endl << endl;

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

if (count == 0) {

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

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

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

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

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

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

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

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

}

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

cin >> number;

count++;

if (number < 0 || number > 18) { //если введен номер несуществуещей сети, выводится сообщение об ошибке

NetworkError(1);

continue;

}

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);

continue;

}

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.getName() << endl;

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

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

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

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

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

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

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

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

cout << "10. " << Mobile\_1.getName() << endl;

cout << "11. " << Mobile\_2.getName() << endl;

cout << "12. " << Mobile\_3.getName() << endl;

cout << "13. " << Mobile\_4.getName() << endl;

cout << "14. " << Mobile\_5.getName() << endl;

cout << "15. " << Mobile\_6.getName() << endl;

cout << "16. " << Mobile\_7.getName() << endl;

cout << "17. " << Mobile\_8.getName() << endl;

cout << "18. " << Mobile\_9.getName() << endl;

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

int number2;

cin >> number2;

cin.get();

q\_num.search(number2);

}

break;

}

return 0;

}

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

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

return 0;

}

if (number >= 1 && number <= 9) {

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

cin >> password;

cin.get();

}

WiFi Network\_Connected;

Mobile Mobile\_Connected;

correct = 0;

switch (number) {

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

break;

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

break;

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

break;

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

break;

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

break;

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

break;

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

break;

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

break;

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

break;

case 10: Mobile\_Connected = authorization(number, traffic, time, Mobile\_1, q, error, q\_num, correct);

break;

case 11: Mobile\_Connected = authorization(number, traffic, time, Mobile\_2, q, error, q\_num, correct);

break;

case 12: Mobile\_Connected = authorization(number, traffic, time, Mobile\_3, q, error, q\_num, correct);

break;

case 13: Mobile\_Connected = authorization(number, traffic, time, Mobile\_4, q, error, q\_num, correct);

break;

case 14: Mobile\_Connected = authorization(number, traffic, time, Mobile\_5, q, error, q\_num, correct);

break;

case 15: Mobile\_Connected = authorization(number, traffic, time, Mobile\_6, q, error, q\_num, correct);

break;

case 16: Mobile\_Connected = authorization(number, traffic, time, Mobile\_7, q, error, q\_num, correct);

break;

case 17: Mobile\_Connected = authorization(number, traffic, time, Mobile\_8, q, error, q\_num, correct);

break;

case 18: Mobile\_Connected = authorization(number, traffic, time, Mobile\_9, q, error, q\_num, correct);

break;

}

while (correct == 1) {

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

cout << "1. " << Site\_1 << endl;

cout << "2. " << Site\_2 << endl;

cout << "3. " << Site\_3 << endl;

cout << "4. " << Site\_4 << endl;

cout << "5. " << Site\_5 << endl;

cout << "6. " << Site\_6 << endl;

cout << "7. " << Site\_7 << endl;

cout << "8. " << Site\_8 << endl;

cout << "9. " << Site\_9 << endl << endl;

if (number <= 9) cout << "0. Отключиться от Wi-Fi сети " << Network\_Connected.getName() << endl << endl;

else cout << "0. Отключиться от мобильной сети " << Mobile\_Connected.getName() << endl << endl;

cout << "Выберите сайт: ";

cin >> number\_site;

switch (number\_site) {

case 1: {

if (number <= 9) login(Network\_Connected, Site\_1);

else login(Mobile\_Connected, Site\_1);

}

break;

case 2: {

if (number <= 9) login(Network\_Connected, Site\_2);

else login(Mobile\_Connected, Site\_2);

}

break;

case 3: {

if (number <= 9) login(Network\_Connected, Site\_3);

else login(Mobile\_Connected, Site\_3);

}

break;

case 4: {

if (number <= 9) login(Network\_Connected, Site\_4);

else login(Mobile\_Connected, Site\_4);

}

break;

case 5: {

if (number <= 9) login(Network\_Connected, Site\_5);

else login(Mobile\_Connected, Site\_5);

}

break;

case 6: {

if (number <= 9) login(Network\_Connected, Site\_6);

else login(Mobile\_Connected, Site\_6);

}

break;

case 7: {

if (number <= 9) login(Network\_Connected, Site\_7);

else login(Mobile\_Connected, Site\_7);

}

break;

case 8: {

if (number <= 9) login(Network\_Connected, Site\_8);

else login(Mobile\_Connected, Site\_8);

}

break;

case 9: {

if (number <= 9) login(Network\_Connected, Site\_9);

else login(Mobile\_Connected, Site\_9);

}

break;

case 0: {

if (number <= 9) cout << "Вы отключились от Wi-Fi сети " << Network\_Connected.getName() << endl << endl;

else cout << "Вы отключились от мобильной сети " << Mobile\_Connected.getName() << endl << endl;

correct--;

}

break;

default: NetworkError(4);

break;

}

}

}

system("pause");

return 0;

}

**Схема классов**



**Описание программы**

В программу с помощью механизма простого наследования был добавлен класс **Mobile**, объекты которого являются мобильными сетями, к которым устройство может подключиться точно также, как и к Wi-Fi сетям.

Для реализации наследования в родительском классе **WiFi** часть полей были переведены из части **private** в часть **protected** (поля из этой части доступны не только изнутри самого класса, но еще и из классов-наследников):

* **name** – строка типа **string**, название сети
* **work** – строка типа **string**, статическая переменная, показывает, работает ли сеть, всегда имеет значение *«Да»*
* **speed** – число типа **int**, скорость сети
* **radius** – число типа **int**, радиус действия сети
* **traffic** – число типа **int**, количество переданной по сети информации
* **frequency** – число типа **double**, частота сети
* **time** – число типа **double**, время подключения к сети

Помимо полей, наследуемых от класса **WiFi**, в класс **Mobile** также дополнительно добавлены поля:

* **type** – поле типа перечисление. Оно означает поколение мобильной связи и может принимать значения *GPRS*, *Edge*, *\_2G*, *\_3G*, *\_4G*, *\_5G* и *unknown*
* **signal** – число типа **int**. Мощность сигнала

Из открытой части класса **WiFi** наследуются перегрузки операторов, геттеры для полей **name**, **work** и **time**, а также функции **sum** и **openSite**.

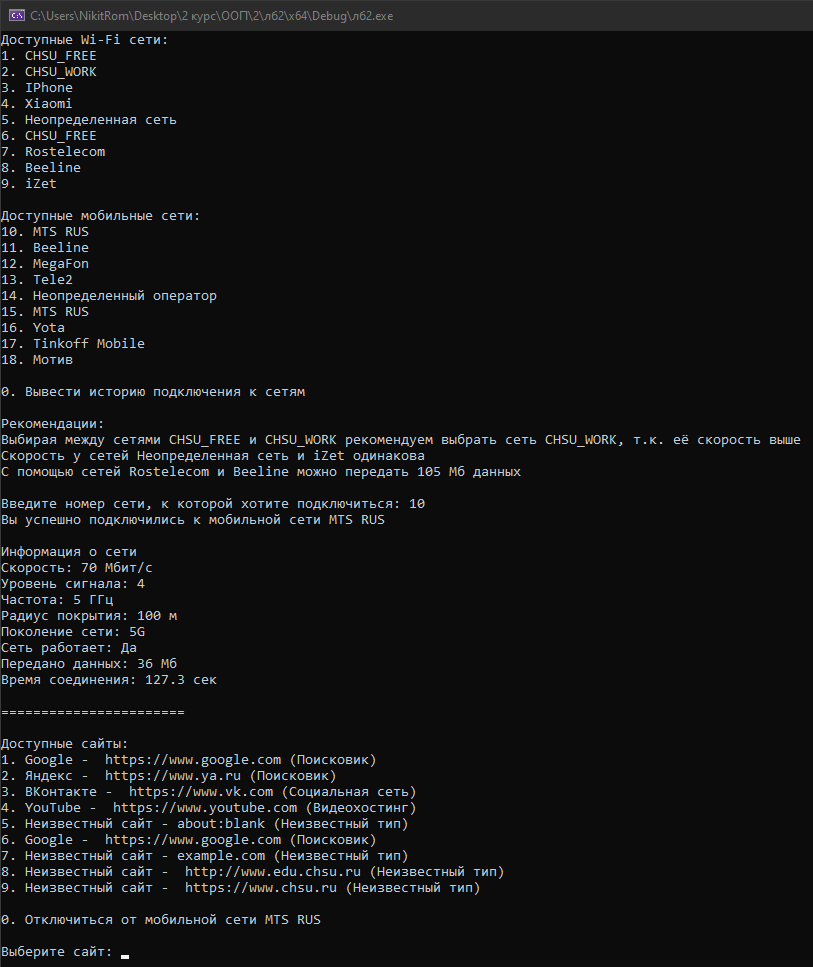
Помимо вышеперечисленных функций, также были добавлены следующие:

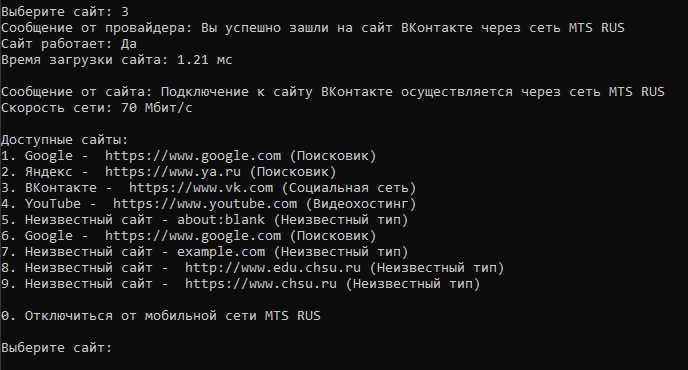
* Конструкторы и деструктор
* **getType** – геттер для поля type – поколения сети
* Была добавлена своя реализация перегрузки оператора **<<**

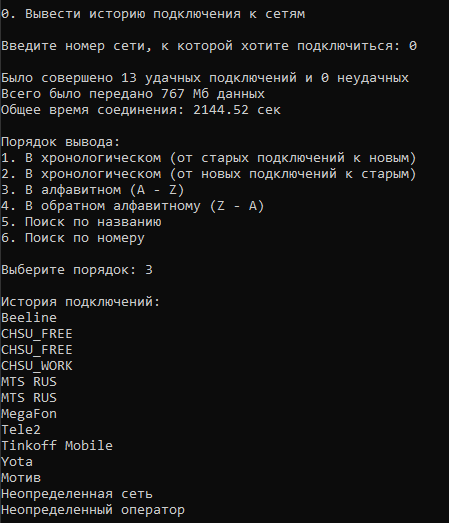
Теперь программа предлагает подключаться не только к Wi-Fi сетям, но еще и к мобильным. В случае подключения к мобильным сетям пароль не запрашивается. Других изменений для пользователей больше нет.

Реализация подключения к сайтам была вынесены в шаблонную функцию login. Это необходимо было сделать, так как функция **openSite** в качестве параметра теперь может принимать не только от класса **WiFi**, но еще и от класса **Mobile**. Аналогично, функцию **status** теперь могут вызывать не только объекты класса **WiFi**, но еще и класса **Mobile**.

**Результаты тестов**







**Контрольные вопросы**

1. *Дайте определение наследования*

Наследование – это концепция ООП, которая позволяет классам наследовать свойства и методы других классов. Класс, который наследует свойства и методы, называется дочерним классом, а класс, от которого наследуются эти свойства и методы, называется родительским классом. Наследование позволяет создавать иерархии классов, уменьшает дублирование кода и способствует повторному использованию кода.

1. *Какие модификаторы прав доступа к производному классу вы знаете и в чем их особенности?*

В языке программирования C++ применяются модификаторы доступа **private**, **protected** и **public**.

* **Private**: члены, объявленные с модификатором доступа private, доступны только внутри самого класса. Они недоступны извне класса, в том числе и из производных классов.
* **Protected**: члены, объявленные с модификатором доступа private, доступны внутри самого класса и из производных классов. Извне класса по-прежнему недоступны.
* **Public**: Члены, объявленные с модификатором доступа **public**, доступны из любого места в программе без ограничений. Эти члены будут доступны и в самом классе, и в его производных классах, и за их пределами.

1. *Как выполняется конструктор при наследовании?*

При наследовании в объектно-ориентированном программировании, в том числе и в языке C++, конструкторы выполняются следующим образом:

1. Вызов конструктора базового класса. Первым делом вызывается конструктор базового класса. Это происходит до выполнения кода в конструкторе производного класса. Если конструктор базового класса имеет параметры, они должны быть переданы из конструктора производного класса.
2. Инициализация членов базового класса. После вызова конструктора базового класса происходит инициализация его членов. Это происходит автоматически перед выполнением кода в конструкторе производного класса.
3. Выполнение конструктора производного класса. После инициализации базового класса выполняется конструктор производного класса. Здесь происходит инициализация членов производного класса и выполнение кода, определенного в его конструкторе.
4. *Дайте определение чисто виртуальной функции.*

Чисто виртуальная функция – это виртуальная функция в базовом классе, для которой не предоставлена реализация. Она обозначается в базовом классе при помощи спецификатора **virtual** и суффикса "**= 0**". Это означает, что класс, содержащий чисто виртуальную функцию, является абстрактным и его нельзя создать напрямую.

Чисто виртуальные функции используются для создания интерфейсов, определяющих набор методов, которые должны быть реализованы в производных классах. При этом каждый производный класс обязан реализовать все чисто виртуальные функции базового класса, чтобы стать полноценным классом.

1. *Какие операторы используются для разыменования указателя на член класса?*

Для разыменования указателя на член класса в C++ используется операторы **–>** и **.\***

Оператор **–>** используется для доступа к членам объекта, на который указывает указатель на объект. Предполагает, что объект, на который указывает указатель, является экземпляром класса.

Применяется, когда указатель указывает на объект класса или на объект производного класса.

Оператор **.\*** используется для доступа к членам объекта, на который указывает указатель на объект, в случае, когда член класса является указателем на функцию.

Обычно используется с указателями на функции-члены класса.

**Вывод**

В ходе выполнения лабораторной работы были изучены способы создания производного класса и особенности работы с ним, а также правила инициализации и доступа к элементам производного класса. Бали приобретены практические навыки наследования.

С помощью механизмов наследования от класса **WiFi** был создан производный класс **Mobile**.