МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт Информационных Технологий

Кафедра МПО ЭВМ

Дисциплина «Объектно-ориентированное программирование»

Лабораторная работа №3

«Множественное наследование»

Выполнил:

студент группы 1ПИб-02-3оп-22

Маркелов Сергей Александрович

Проверил:

Кустов Максим Александрович

Череповец, 2024 год

**Цель работы:**

* Изучить принципы и механизмы множественного наследования
* Изучить правила доступа к базовым классам;
* Приобрести практические навыки работы с базовыми и производными классами при множественном наследовании.

**Требования к разработке:**

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

**Задание на лабораторную работу:**

* Создать новый производный класс с применением множественного наследования.
* Проверить 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 **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 {

protected:

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

}

};

Заголовочный файл **Flash.h**:

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

using namespace std;

class Flash {

private:

int capacity;

string name;

protected:

double version;

char connector;

public:

Flash(string valueName, int valueCapacity, double valueVersion, char valueConnector) {

this->name = valueName;

this->capacity = valueCapacity;

this->version = valueVersion;

this->connector = valueConnector;

}

Flash() {

this->name = "Неопознанное USB-устройство";

this->capacity = 0;

this->version = 0.0;

this->connector = 'N';

}

Flash(string valueName) {

this->name = valueName;

this->capacity = 0;

this->version = 0.0;

this->connector = 'N';

}

Flash(string valueName, int valueCapacity) {

this->name = valueName;

this->capacity = valueCapacity;

this->version = 0.0;

this->connector = 'N';

}

Flash(const Flash& obj) {

this->name = obj.name;

this->capacity = obj.capacity;

this->version = obj.version;

this->connector = obj.connector;

}

string getName() {

return name;

}

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

info << "Подключено USB-устройство " << other.name << endl << endl;

info << "Информация об устройстве" << endl;

info << "Объем USB-накопителя: " << other.capacity << " Гб" << endl;

info << "Версия USB: " << other.version << endl;

info << "Разъем: " << "Type-" << other.connector << endl;

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

return info;

}

~Flash() { //деструктор

}

};

Заголовочный файл **Modem.h**:

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <string>

#include "Mobile.h"

#include "Flash.h"

using namespace std;

class Modem : public Mobile, public Flash {

private:

bool antenna, sd;

public:

Modem(string valueName, generation valueType, int valueSpeed, int valueRadius, double valueFrequency, double valueVersion, char valueConnector, bool valueAntenna, bool valueSd) {

this->WiFi::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;

this->version = valueVersion;

this->connector = valueConnector;

this->antenna = valueAntenna;

this->sd = valueSd;

}

Modem() {

this->WiFi::name = "Неопознанное USB-устройство"; //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;

this->version = 0.0;

this->connector = 'N';

this->antenna = 0;

this->sd = 0;

}

Modem(string valueName) {

this->WiFi::name = valueName; //конструктор c 1 параметром

this->type = unknown;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = 0;

this->time = rand() / 100.0;

this->signal = rand() % 5;

this->version = 0.0;

this->connector = 'N';

this->antenna = 0;

this->sd = 0;

}

Modem(string valueName, generation valueType) {

this->WiFi::name = valueName; //конструктор c 2 параметрами

this->type = valueType;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = 0;

this->time = rand() / 100.0;

this->signal = rand() % 5;

this->version = 0.0;

this->connector = 'N';

this->antenna = 0;

this->sd = 0;

}

Modem(string valueName, generation valueType, char valueConnector) {

this->WiFi::name = valueName; //конструктор c 3 параметрами

this->type = valueType;

this->speed = 0;

this->radius = 0;

this->frequency = 0;

this->traffic = 0;

this->time = rand() / 100.0;

this->signal = rand() % 5;

this->version = 0.0;

this->connector = valueConnector;

this->antenna = 0;

this->sd = 0;

}

Modem(const Modem& obj) {

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

this->type = obj.type;

this->speed = obj.speed;

this->radius = obj.radius;

this->frequency = obj.frequency;

this->traffic = obj.traffic;

this->time = rand() / 100.0;

this->signal = rand() % 5;

this->version = obj.version;

this->connector = obj.connector;

this->antenna = obj.antenna;

this->sd = obj.sd;

}

string getAntenna() {

if (antenna) return "Есть";

else return "Нет";

}

string getSd() {

if (sd) return "Есть";

else return "Нет";

}

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

info << "Вы успешно подключились к модему " << other.WiFi::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 << "Версия USB: " << other.version << endl;

info << "Разъем: " << "Type-" << other.connector << endl;

info << "Наличие антенны: " << other.getAntenna() << endl;

info << "Наличие разъема для SD-карты: " << other.getSd() << endl;

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

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

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

return info;

}

~Modem() { //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"

#include "Flash.h"

#include "Modem.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;

}

}

template<typename T>

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

T Network\_Connected(Network);

q.add(Network.WiFi::getName());

q\_num.add(number);

correct++;

cout << Network;

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

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

return Network\_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);

Flash USB\_1("Kingston", 8, 3.1, 'A');

Flash USB\_2("WD");

Flash USB\_3("Corsair", 2);

Flash USB\_4;

Flash USB\_5(USB\_1);

Modem Modem\_1("MegaFon", \_4G, 80, 100, 5, 3.0, 'A', 1, 1);

Modem Modem\_2("Yota");

Modem Modem\_3("Rostelecom", \_3G);

Modem Modem\_4("Beeline", \_2G, 'A');

Modem Modem\_5;

Modem Modem\_6(Modem\_1);

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 << "Подключенные USB-устройства: " << endl; //вывод доступных мобильных сетей на экран

cout << "19. " << USB\_1.getName() << endl;

cout << "20. " << USB\_2.getName() << endl;

cout << "21. " << USB\_3.getName() << endl;

cout << "22. " << USB\_4.getName() << endl;

cout << "23. " << USB\_5.getName() << endl << endl;

cout << "Подключенные USB-модемы: " << endl; //вывод доступных мобильных сетей на экран

cout << "24. " << Modem\_1.WiFi::getName() << endl;

cout << "25. " << Modem\_2.WiFi::getName() << endl;

cout << "26. " << Modem\_3.WiFi::getName() << endl;

cout << "27. " << Modem\_4.WiFi::getName() << endl;

cout << "28. " << Modem\_5.WiFi::getName() << endl;

cout << "29. " << Modem\_6.WiFi::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/устройства: "; //выбор сети/устройства

cin >> number;

count++;

if (number < 0 || number > 29) { //если введен номер несуществуещей сети/устройства, 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;

Modem Modem\_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;

case 19: cout << USB\_1;

break;

case 20: cout << USB\_2;

break;

case 21: cout << USB\_3;

break;

case 22: cout << USB\_4;

break;

case 23: cout << USB\_5;

break;

case 24: Modem\_Connected = authorization(number, traffic, time, Modem\_1, q, error, q\_num, correct);

break;

case 25: Modem\_Connected = authorization(number, traffic, time, Modem\_2, q, error, q\_num, correct);

break;

case 26: Modem\_Connected = authorization(number, traffic, time, Modem\_3, q, error, q\_num, correct);

break;

case 27: Modem\_Connected = authorization(number, traffic, time, Modem\_4, q, error, q\_num, correct);

break;

case 28: Modem\_Connected = authorization(number, traffic, time, Modem\_5, q, error, q\_num, correct);

break;

case 29: Modem\_Connected = authorization(number, traffic, time, Modem\_6, 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 if (number > 9 && number <= 18) login(Mobile\_Connected, Site\_1);

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

}

break;

case 2: {

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

else if (number > 9 && number <= 18) login(Mobile\_Connected, Site\_2);

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

}

break;

case 3: {

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

else if (number > 9 && number <= 18) login(Mobile\_Connected, Site\_3);

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

}

break;

case 4: {

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

else if (number > 9 && number <= 18) login(Mobile\_Connected, Site\_4);

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

}

break;

case 5: {

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

else if (number > 9 && number <= 18) login(Mobile\_Connected, Site\_5);

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

}

break;

case 6: {

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

else if (number > 9 && number <= 18) login(Mobile\_Connected, Site\_6);

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

}

break;

case 7: {

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

else if (number > 9 && number <= 18) login(Mobile\_Connected, Site\_7);

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

}

break;

case 8: {

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

else if (number > 9 && number <= 18) login(Mobile\_Connected, Site\_8);

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

}

break;

case 9: {

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

else if (number > 9 && number <= 18) login(Mobile\_Connected, Site\_9);

else login(Modem\_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 были добавлены 2 класса – **Flash** и **Modem**.

**Flash** – класс USB-устройств. Имеет следующие поля в закрытой части класса:

* **capacity** – число типа **int**, объем памяти флешки
* **name** – строка типа **string**, название производителя флешки

В защищенной части класса следующие поля:

* **version** – число типа **double**, версия USB
* **connector** – символ типа **char**, тип разъема

В открытой части класса добавлены следующие функции:

* Конструкторы и деструктор
* Геттер для поля **name**
* Перегрузка оператора **<<**

**Modem** – класс модемов. Получен путем множественного наследования от классов **Mobile** и **Flash**.

Для реализации наследования все поля класса **Mobile** (**type**, **signal**) переведены из части **private** в часть **protected**.

Помимо полей от исходных классов, класс Modem также имеет дополнительные:

* **antenna** – логическая переменная типа **bool**, показывает наличие антенны
* **sd** – логическая переменная типа **bool**, показывает наличие разъема для SD-карты

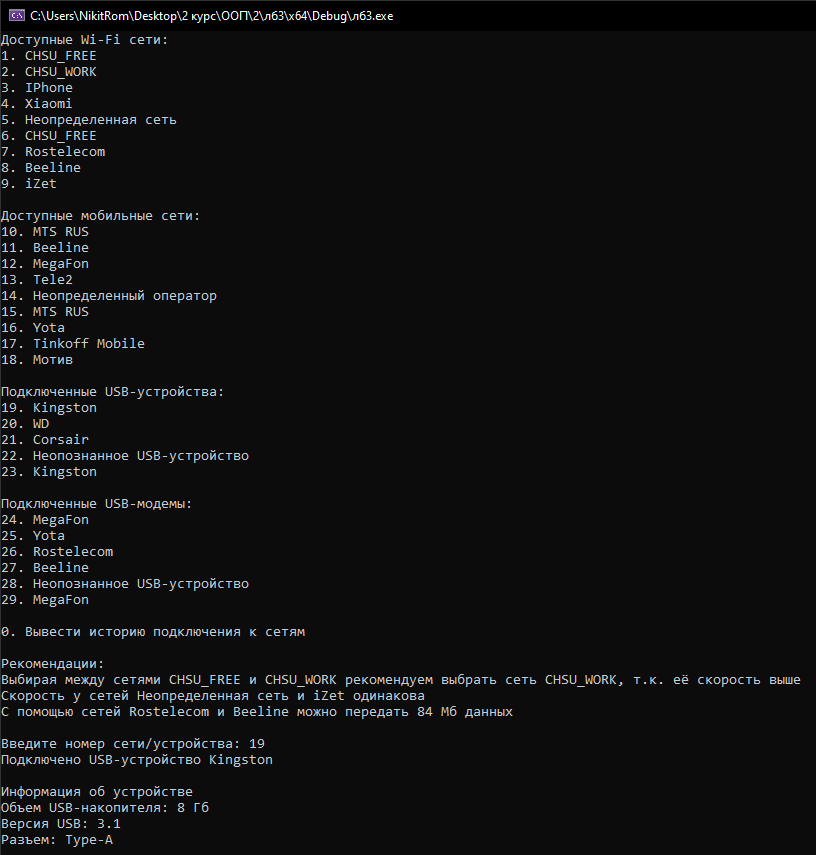
В открытой части реализованы следующие функции:

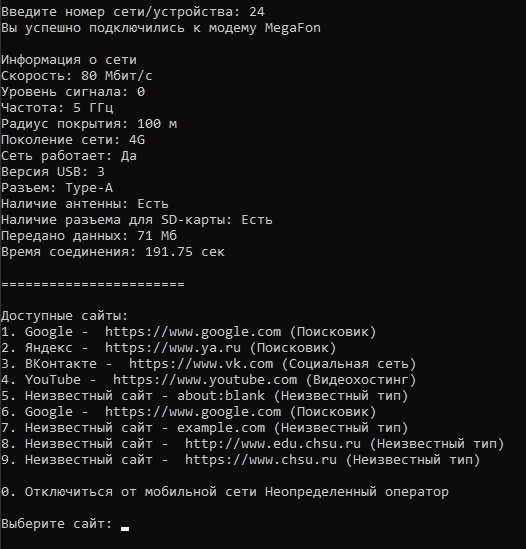
* Конструкторы и деструктор
* Геттеры для полей **antenna** и **sd**
* Перегрузка оператора **<<**

В основной части программы внесены следующие изменения

* Теперь программа предлагает подключаться не только к Wi-Fi и мобильным сетям, но еще и к флешкам и модемам
* В случае подключения к флешке, выводится информация об устройстве, в случае подключения к модему – выполняется функция **authorization**
* Функция **authorization** стала шаблонной, т. к. теперь она принимает объекты не только класса **Mobile**, но еще и **Modem**

**Результаты тестов**





**Контрольные вопросы**

1. *С какой целью и в каких случаях используется множественное наследование?*

Множественное наследование в ООП позволяет классу наследовать свойства и методы от нескольких базовых классов. Оно применяется, если объект должен иметь функциональность или свойства, представленные различными абстракциями или интерфейсами.

1. *Опишите синтаксис заголовка производного класса при множественном наследовании.*

Синтаксис заголовка производного класса при множественном наследовании выглядит следующим образом:

**class *Derived* : *modifier Base1*, *modifier Base2***

где **Derived** – имя производного класса, **Base1** и **Base2** – имена базовых классов, **modifier** – модификатор доступа (public, protected, private).

1. *Дайте определение ориентированного ациклического графа.*

Ориентированный ациклический граф – это граф, состоящий из вершин и направленных ребер между ними, при этом в графе отсутствуют циклы. Другими словами, это граф, в котором нельзя пройти по ребрам от одной вершины и вернуться в неё снова, двигаясь по направлению ребер.

1. *Где инициируются виртуальные базовые классы?*

Виртуальные базовые классы инициируются в списке инициализации конструктора каждого производного класса, который непосредственно наследуется от виртуального базового класса.

Например, у нас есть иерархия, где **Base** является виртуальным базовым классом, а классы **Derived1** и **Derived2** являются его непосредственными производными классами. Когда производный класс **Derived3** наследует от **Derived1** и **Derived2**, он также наследует и от **Base** через оба этих класса.

1. *Что будет, если из объявлений классов Cow и Buffalo убрать ключевое слово virtual (см. последний пример в теоретических положениях)?*

Если убрать ключевое слово **virtual** из объявлений классов **Cow** и **Buffalo**, то множественное наследование будет происходить без использования виртуального наследования. Это приведет к созданию 2 отдельных подобъектов **DomesticAnimal** в объектах классов **Cow** и **Buffalo**, а не одного общего подобъекта, как в случае виртуального наследования.

В результате, при вызове конструктора класса **Beefalo**, объект **Beefalo** будет иметь две копии членов класса **DomesticAnimal** - одну от класса **Cow**, а другую от класса **Buffalo**. Это приведет к проблемам, таким как двойное использование памяти и неоднозначность при обращении к членам класса **DomesticAnimal** из объекта **Beefalo**.

**Вывод**

В ходе выполнения лабораторной работы были изучены принципы и механизмы множественного наследования, а также правила доступа к базовым классам. Были приобретены практические навыки работы с базовыми и производными классами при множественном наследовании.

В программе был добавлен класс **Flash** и реализовано множественное наследование для класса **Modem** от классов **Mobile** и **Flash**.