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

«Перегрузка операторов»

Выполнил:

студент группы 1ПИб-02-3оп-22

Маркелов Сергей Александрович

Проверил:

Табунов Павел Александрович

Череповец, 2023 год

**Цель работы:**

* Изучить понятие перегрузки
* Приобрести практические навыки перегрузки отдельных операторов.

**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:**

Модифицировать абстрактные типы данных, реализованные по заданию лабораторной работы №2, перегрузив минимум 3 ключевых оператора.

Обязательно перегрузить оператор вывода на консоль и один и операторов сравнения у АТД.

Использовать эти операторы в классе-контейнере, в первую очередь в функции сортировки и вывода элементов АТД.

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;

class Queue {

private:

struct Node { //узел очереди

string 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(string 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++;

}

}

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

if (empty()) {

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

}

else {

string 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);

}

}

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

while (!q.empty())

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

return del;

}

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;

}

void print\_NewOld() {

string\* networks = new string[size];

int i = 0;

while (!empty()) {

networks[i] <= del();

i++;

}

i--;

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

cout << networks[i] << endl;

}

void print\_AZ() {

string\* networks = new string[size];

int i = 0;

while (!empty()) {

networks[i] = del();

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;

while (!empty()) {

networks[i] = del();

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;

while (!empty()) {

networks[i] = del();

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\* history) {

int k = 0;

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

if (number == history[i]) 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;

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;

}

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;

}

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;

}

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;

}

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;

}

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;

}

string getLogin() { //геттер для логина

return login;

}

string getPassword() { //геттер для пароля

return password;

}

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";

}

}

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 << 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& count, int& capacity, int number, int\* history, string password, WiFi& Network, Queue& q, Queue& error) {

if (count == capacity - 1) capacity += 100;

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

q.add(Network.getLogin());

history[count] = number;

count++;

cout << Network;

}

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, count = 0, capacity = 100;

string password, name;

Queue q, error;

int\* history = new int[capacity];

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 (count == 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 << 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: ";

int number2;

cin >> number2;

cin.get();

q.search(number2, history);

}

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(count, capacity, number, history, password, Network\_1, q, error);

break;

case 2: authorization(count, capacity, number, history, password, Network\_2, q, error);

break;

case 3: authorization(count, capacity, number, history, password, Network\_3, q, error);

break;

case 4: authorization(count, capacity, number, history, password, Network\_4, q, error);

break;

case 5: authorization(count, capacity, number, history, password, Network\_5, q, error);

break;

case 6: authorization(count, capacity, number, history, password, Network\_6, q, error);

break;

case 7: authorization(count, capacity, number, history, password, Network\_7, q, error);

break;

case 8: authorization(count, capacity, number, history, password, Network\_8, q, error);

break;

case 9: authorization(count, capacity, number, history, password, Network\_9, q, error);

break;

}

}

system("pause");

return 0;

}

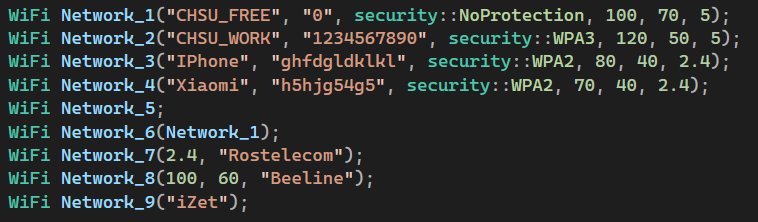
**Описание программы**

В программе были перегружены:

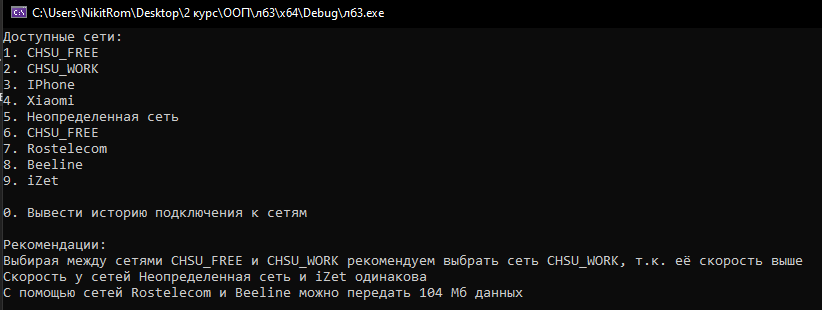
1. Конструктор **WiFi** класса **WiFi**:
   * Конструктор по умолчанию
   * Конструктор с 1 параметром
   * Конструктор с 2 параметрами
   * Конструктор с 3 параметрами
   * Конструктор со всеми параметрами
   * Копирующий конструктор
2. Функция **search** класса **Queue**:
   * **search(string name)** – на вход принимает строку и осуществляет поиск сетей, логин которых начинается с введенной строки (или полностью с ней совпадает) и выводит их на экран (либо выводит надпись «*Ничего не найдено!*», если таких сетей нет)
   * **search(int number, int\* history)** – на вход принимает число (номер сети) и указатель на массив с историей номеров сетей, к которым было осуществлено подключение. Затем на экран выводится количество подключений к этой сети (либо выводится надпись «Подключений к сети №… не было», если к этой сети не подключались)
3. Операторы:
   * Операторы сравнения **>**, **<**, **==** в классе **WiFi**, они сравнивают скорости WiFi-сетей и выводят на экран рекомендации, к какой сети лучше подключаться
   * Арифметический оператор **+** в классе **WiFi**, который складывает объём трафика в двух сетях
   * Функция **NetworkInfo** (вывод всех полей класса с информацией о сети) в классе **WiFi** была заменена на перегрузку оператора вывода <<
   * Операторы сравнения **>=**, **<** в классе **Queue** (Сравнивают очереди **q** с историей удачных подключений и **err** с историей неудачных подключений)
   * Функция **print\_OldNew** (вывод истории подключений от старым к новым) в классе **Queue** была заменена на перегрузку оператора вывода **<<**

**Результаты тестов**

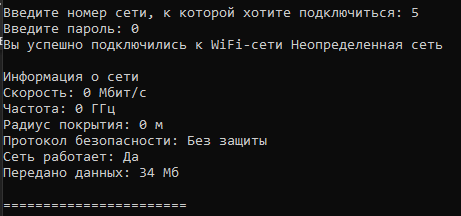
Перегрузка конструктора:



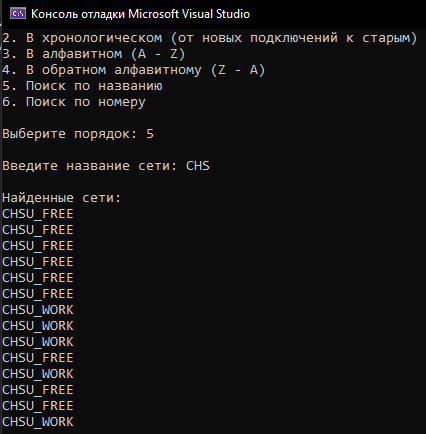
Рекомендации (реализованы с помощью перегрузки операторов сравнения и арифметического оператора):



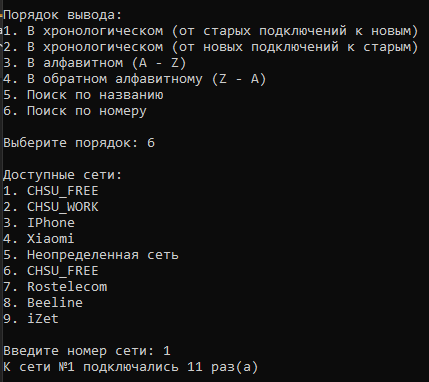
Вывод информации о сети (реализовано с помощью перегрузки оператора вывода, раньше было с помощью функции):



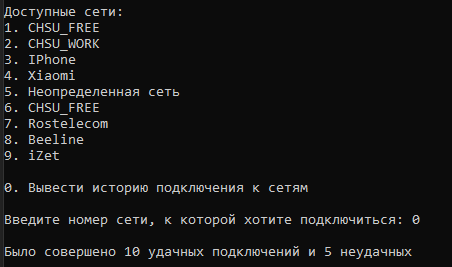
Поиск по строке (реализован с помощью перегрузки функции):



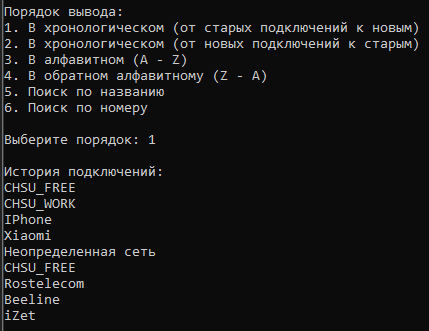
Поиск по номеру (реализован с помощью перегрузки функции):



Информация о количестве удачных и неудачных подключений (реализована с помощью перегрузки операторов сравнения):



Вывод истории подключений в хронологическом порядке (реализован с помощью перегрузки оператора вывода, раньше был реализован с помощью функции):



**Контрольные вопросы**

1. *Назовите алгоритм соответствия для каждого аргумента.*

Перегруженные операторы сравниваются с типами аргументов. Если находится точное соответствие, то выбирается соответствующая перегруженная функция или оператор.

Если точное соответствие не найдено, компилятор ищет перегруженные функции или операторы, для которых аргументы могут быть преобразованы в типы параметров. При этом компилятор рассматривает стандартные преобразования типов, преобразования с сохранением квалификаторов и другие преобразования.

Если есть несколько возможных перегруженных функций или операторов, компилятор выбирает наилучший вариант, исходя из критериев, таких как наименьшее количество преобразований или более специфичные преобразования.

1. *С какой целью применяется перегрузка операторов?*

Перегрузка операторов позволяет определить действия, которые будет выполнять оператор с пользовательским типом данных. Перегрузка необходима для того, чтобы иметь возможность проводить различные операции с пользовательскими типами данных аналогично встроенным типам (таким, как int, double и др.). Это сделает код более удобным и улучшит его читаемость.

1. *Для каких операторов нельзя использовать перегрузку?*

Перегружать нельзя операторы «**::**», «**sizeof**», «**?:**», «**.**», «**.\***»

Оператор «**::**» используется для доступа к членам класса или пространству имен. Перегрузка этого оператора не имеет смысла, т.к. он не предназначен для выполнения каких-либо операций.

Оператор «**sizeof**» определяет размер объекта или типа данных в байтах. Размер типа известен на этапе компиляции, и его перегрузка не имеет смысла.

Тернарный оператор «**?:**» используется для создания условных выражений. Он является частью синтаксиса языка и не поддается перегрузке, т.к. не имеет какого-либо аналогичного использования.

Оператор «**.**» используется для доступа к членам объекта класса. Перегрузка этого оператора может привести к неоднозначным ситуациям и усложнению языка.

Оператор «**.\***» используется для вызова членов класса через указатель на объект класса. Перегрузка этого оператора создала бы сложности, поскольку работа с указателями на члены класса требует особого внимания к типам и безопасности.

1. *В каком случае используется функция-член, имеющая пустой список аргументов?*

Функция-член с пустым списком аргументов является константной функцией-членом. Такая функция-член имеет ключевое слово **const** после списка параметров. Она используется, когда необходимо гарантировать, что вызов функции не изменит состояние объекта.

1. *Что произойдет, если перегруженная операция [ ] будет иметь тип возвращаемого значения int, а не int&?*

Если перегруженная операция **[ ]** будет иметь тип возвращаемого значения **int**, а не **int&**, то операция будет возвращать не ссылку на объект, а лишь его копию. Это может повлиять на эффективность кода и вызвать в дальнейшем ошибки и непредсказуемое поведение программы.

**Вывод**

В ходе выполнения лабораторной работы были изучены способы определения классов, правила доступа к элементам, принципы и механизмы создания абстрактных типов данных, синтаксис и семантика определения и вызова конструкторов и деструкторов, особенности применения различных видов конструкторов. Также были приобретены практические навыки работы с объектами класса, создания и уничтожения объектов.

Были разработаны алгоритм и программа, реализующие абстрактный тип данных (АТД) – класс, обеспечивающий хранение объектов класса из лабораторной работы №1 в очереди.

Интерфейс класса содержит функции добавления, удаления, вывода содержимого и обработки (сортировка по полю класса, поиск по условию).