МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт Информационных Технологий

Кафедра МПО ЭВМ

Дисциплина «Объектно-ориентированное программирование»

Лабораторная работа №1

«Разработка абстрактных типов данных»

Выполнил:

студент группы 1ПИб-02-3оп-22

Маркелов Сергей Александрович

Проверил:

Табунов Павел Александрович

Череповец, 2023 год

**Цель работы:**

* изучить способы определения классов, правила доступа к элементам;
* приобрести практические навыки работы с объектами класса;
* изучить принципы и механизмы создания абстрактных типов данных;
* изучить синтаксис и семантику определения и вызова конструкторов и деструкторов;
* приобрести практические навыки создания и уничтожения объектов;
* изучить особенности применения различных видов конструкторов.

**Требования к разработке:**

1. Каждый класс должен быть оформлен в отдельных файлах: заголовочный (.h) и файл с кодом (.cpp).
2. Запрещается использовать обработку исключительных ситуаций и генерировать исключения.
3. Придерживаться принципа DRY (Don’t repeat yourself).
4. Обязательно наличие комментариев.

**Задание на лабораторную работу:**

Разработать алгоритм и программу, реализующую абстрактный тип данных (АТД) – класс, согласно варианту задания (вариант №14 – устройства передачи данных):

1. Предусмотреть закрытую реализацию и открытый интерфейс.
2. Интерфейс должен содержать функции вывода содержимого и функции доступа к некоторым полям.
3. Элементы данных должны быть представлены из 5 полей, один из которых должен быть статическим, второй иметь тип указатель, а третий тип перечисление, остальные на свое усмотрение.
4. Применить основные виды конструкторов и определиться с деструктором класса.
5. Проверить работоспособность АТД на тестовом наборе данных.

**Текст программы**

#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;

double frequency;

public:

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

login = valueLogin; //конструктор

password = valuePassword;

protection = valueProtection;

speed = valueSpeed;

radius = valueRadius;

frequency = valueFrequency;

work = "Да";

}

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";

}

}

void networkInfo() { //вывод информации о сети

cout << "Вы успешно подключились к WiFi-сети " << login << endl << endl;

cout << "Информация о сети" << endl;

cout << "Скорость: " << speed << " Мбит/с" << endl;

cout << "Частота: " << frequency << " ГГц" << endl;

cout << "Радиус покрытия: " << radius << " м" << endl;

cout << "Протокол безопасности: " << getProtection() << endl;

cout << "Cеть работает: " << work << endl;

}

~WiFi() { //деструктор

}

};

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;

}

}

int main() {

setlocale(LC\_ALL, "Russian");

int number;

string password, name;

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);

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 << endl;

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

cin >> number;

if (number < 1 || number > 4) { //если введен номер несуществующей сети, выводится сообщение об ошибке

NetworkError(1);

return 0;

}

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

cin >> password;

switch (number) {

case 1: {

if (password == Network\_1.getPassword()) Network\_1.networkInfo(); //если пароль верный, то выводится информация о сети

else {

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

return 0;

}

}

break;

case 2: {

if (password == Network\_2.getPassword()) Network\_2.networkInfo();

else {

NetworkError(2);

return 0;

}

}

break;

case 3: {

if (password == Network\_3.getPassword()) Network\_3.networkInfo();

else {

NetworkError(2);

return 0;

}

}

break;

case 4: {

if (password == Network\_4.getPassword()) Network\_4.networkInfo();

else {

NetworkError(2);

return 0;

}

}

break;

}

system("pause");

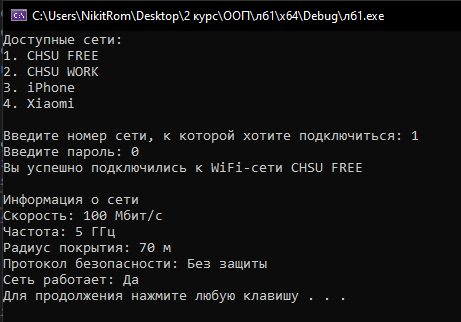
return 0;

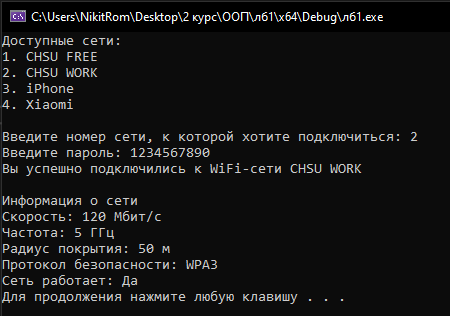
}

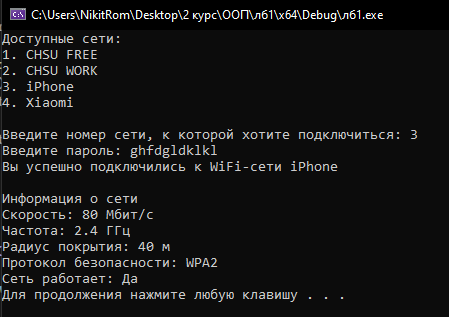
**Описание программы**

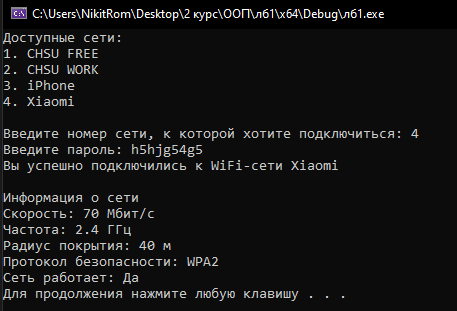
1. Определяется статическая переменная **WiFi::work**
2. Задается функция **NetworkError**:
   * Если **err = 1**, на экран выводится фраза «*Такой WiFi-сети нет!*»
   * Если **err = 2**, на экран выводится фраза «*Пароль неверный!*»
3. Инициализируется целочисленная переменная number и строки password и name
4. Вызывается конструктор класса **WiFi** для объектов **Network\_1**, **Network\_2**, **Network\_3**, **Network\_4**
   * Класс **WiFi** содержит поля **login**, **password**, **work**, **protection**, **speed**, **radius**, **frequency**
   * Конструктор инициализирует поля **login**, **password**, **protection**, **speed**, **radius**, **frequency** как параметры, полю **work** присваивается значение «*Да*»
5. На экран выводятся доступные сети (значение поля **login**) с помощью геттера **getLogin**
6. Программа просит пользователя ввести значение переменной **number** – номер сети
7. Если пользователь ввел значение меньше 1 или больше 4 (т.е. указал номер несуществующей сети), вызывается функция **NetworkError** с параметром **err = 1**, затем выполнение программы завершается
8. Программа просит пользователя ввести пароль (в строку **password**)
9. Если значение переменной password равно значению поля password класса **WiFi** (для сравнения вызывается геттер **getPassword**) вызывается функция **NetworkInfo**, которая выводит значения всех полей
10. Если значение переменной **password** не равно значению поля **password** класса **WiFi**, вызывается функция **NetworkError** с параметром **err = 2**, затем выполнение программы завершается
11. Вызывается деструктор класса **~WiFi**, который уничтожает класс **WiFi**

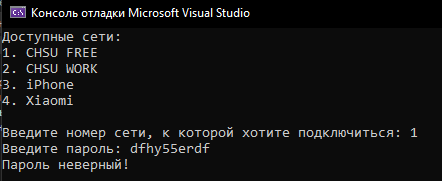
**Результаты тестов**

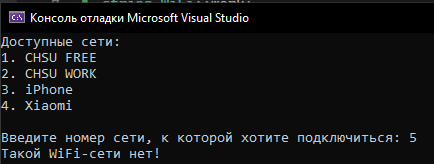












**Контрольные вопросы**

1. *Дайте определение понятия «класс». Сформулируйте правила доступа к его элементам.*

Класс – это абстрактный тип данных, созданный пользователем. Зачастую классы описывают какие-либо объекты реального мира. Как и реальный объект, класс содержит свой набор параметров и характеристик. Каждый такой параметр называется поле класса (очень похоже на обычные переменные). Также класс способен манипулировать своими характеристиками (полями) с помощью методов класса (похожи на функции в процедурных языках).

В ООП существует 3 основных модификатора доступа, которые определяют, какие части класса могут быть доступны извне класса. Они влияют на поля и методы класса.

* **Public (открытый)**: доступен из любого места кода, как внутри класса, так и извне
* **Private (закрытый)**: доступен только внутри класса, в котором он определен
* **Protected (защищенный)**: доступен внутри класса и его наследных подклассов, но недоступен извне класса

1. *С какой целью в классе объединены компонентные данные и компонентные функции?*

Объединение компонентных данных и компонентных функций является одним из принципов ООП – **инкапсуляцией**

Инкапсуляция представляет собой изоляцию методов и классов, чтобы их нельзя было повредить извне. Идея инкапсуляции заключается в том, чтобы программист пользовался только теми свойствами и методами, которые есть у класса или объекта и не лез внутрь него

1. *Каким образом осуществляется доступ к открытым и закрытым элементам?*

Доступ к открытым элементам класса (public) осуществляется напрямую – как внутри класса, так и извне.

Доступ к закрытым элементам класса (private) осуществляется через геттеры и сеттеры. Геттеры позволяют узнать значение свойства объекта, а сеттеры – установить новое значение свойству объекта

1. *Опишите назначение дружественных функций, назовите их разновидности.*

Дружественной функцией класса называется функция, которая, не являясь его компонентом, имеет доступ к его закрытым (private) и защищенным (protected) компонентам. Функция не может стать дружественной для класса «без его согласия». Для получения прав дружественной функции она должна быть описана в теле класса со спецификатором friend. Дружественные функции используются в тех случаях, когда два класса (или функции) должны иметь доступ к закрытым или защищенным компонентам друг друга без необходимости делать их открытыми для всех.

1. *Что понимается под указателем this?*

this – это указатель на текущий объект данного класса. Через this мы сможем обратиться внутри класса к любым членам этого класса.

1. *Каковы особенности использования статических компонентных данных?*

Статические компонентные данные обладают рядом особенностей:

* Для статических компонентных данных память выделяется всего один раз во время выполнения программы, независимо от того, сколько объектов класса было создано
* Статические компонентные данные могут быть использованы без создания экземпляра класса. Они принадлежат самому классу, а не к конкретному объекту. К ним можно обращаться с использованием имени класса. Изменения в статических данных отразятся на всех объектах класса
* Статические компонентные данные являются общими для всех экземпляров класса, т.е. все объекты класса используют одно и то же статическое поле

1. *В чем заключается синтаксис и семантика компонентных функций static и const?*

Статическая функция static может использоваться как внутри определения класса, так и вне его. При этом она принадлежит классу, а не конкретному объекту. Она не имеет доступа к нестатическим компонентам класса и может вызываться с именем класса, а не через объект. Статическая функция может использоваться для выполнения операций, которые относятся к классу в целом, а не к конкретному экземпляру

У константной функции const может быть использовано после списка параметров функции в объявлении и определении функции. Константная функция не изменяет состояние объекта, для которого она вызывается. Такие функции не могут модифицировать члены класса, кроме mutable-членов. Методы, которые не изменяют состояния объекта, могут быть вызваны для константных объектов и в константных методах

1. *Каким образом могут изменяться компонентные данные объектов, объявленных константами?*

Если объект объявлен с использованием ключевого слова const, это означает, что компонентные данные этого объекта не могут быть изменены напрямую внутри методов этого объекта. Однако существуют способы обойти это ограничение и изменять компонентные данные объектов, объявленных как константы:

* Если внутри класса есть член данных, объявленный с ключевым словом mutable, то он может быть изменен даже внутри константных методов объекта
* Если объект объявлен как константный, но имеется указатель или ссылка на него, которые не являются константными, то можно изменять компонентные данные через этот указатель/ссылку

1. *Каковы особенности создания вложенных классов?*

Вложенные классы в ООП представляют собой классы, объявленные внутри другого класса. Это создает связь между внешним и вложенными классами и обеспечивает логическую группировку классов. Вложенные классы могут иметь доступ к закрытым и защищенным членам внешнего класса. Внешний класс и вложенный класс могут обращаться друг к другу и использовать свои члены, включая приватные. Это приводит к взаимной зависимости между двумя классами.

**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.