**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**



ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И УПРАВЛЯЮЩИХ СИСТЕМ

**Лабораторная работа №5**

по дисциплине: Компьютерные сети

тема: ««Протоколы ARP/RARP»

Выполнил: ст. группы ПВ-223

Игнатьев Артур Олегович

Проверили:

Рубцов Константин Анатольевич

Белгород 2025 г.

**Цель работы:** изучить протоколы ARP/RARP.

**Краткие теоретические сведения**

ARP (Address Resolution Protocol - протокол определения адреса) - протокол канального уровня, предназначенный для определения MAC-адреса (адреса канального уровня) по известному IP-адресу (адресу сетевого уровня). Наибольшее распространение этот протокол получил благодаря распространению сетей IP, построенных поверх Ethernet, поскольку практически в 100 % случаев при таком сочетании используется протокол ARP.

Функциональность протокола ARP сводится к решению двух задач. Одна часть протокола определяет физические адреса при посылке дейтаграммы, другая отвечает на запросы устройств в сети. Протокол ARP предполагает, что каждое устройство «знает» как свой IP -адрес, так и свой физический адрес.

Для того чтобы уменьшить количество посылаемых запросов ARP, каждое устройство в сети, использующее протокол ARP , должно иметь специальную буферную память. В ней хранятся пары адресов (IP-адрес, физический адрес) устройств в сети. Всякий раз, когда устройство получает ARP-ответ, оно сохраняет в буферной памяти соответствующую пару. Если адрес есть в списке пар, то нет необходимости посылать ARP-запрос. Эта буферная память называется ARP-таблицей.

В ARP-таблице могут содержаться как статические, так и динамические записи. Динамические записи добавляются и удаляются автоматически, статические заносятся вручную. Так как большинство устройств в сети поддерживает динамическое разрешение адресов, то администратору, как правило, нет необходимости собственноручно указывать записи протокола ARP в таблице адресов.

Кроме того, ARP-таблица всегда содержит запись с физическим широковещательным адресом (OxFFFFFFFFFFFF) для локальной сети. Эта запись позволяет устройству принимать широковещательные ARP-запросы. Каждая запись в ARP-таблице имеет свое время жизни, например для операционной системы Microsoft Windows 2000 оно составляет 10 минут. При добавлении записи для нее активируется таймер. Если запись не востребована в первые две минуты, она удаляется. Если используется — будет существовать на протяжении 10 минут. В некоторых реализациях протокола ARP новый таймер устанавливается после каждого обращения к записи в ARP -таблице.

Узел, которому нужно выполнить отображение IP-адреса на локальный адрес, формирует ARP запрос, вкладывает его в кадр протокола канального уровня, указывая в нем известный IP-адрес, и рассылает запрос широковещательно. Все узлы локальной сети получают ARP запрос и сравнивают указанный там IP-адрес с собственным. В случае их совпадения узел формирует ARP-ответ, в котором указывает свой IP-адрес и свой локальный адрес и отправляет его уже направленно, так как в ARP запросе отправитель указывает свой локальный адрес. ARP-запросы и ответы используют один и тот же формат пакета. Так как локальные адреса могут в различных типах сетей иметь различную длину, то формат пакета протокола ARP зависит от типа сети.

Пример работы протокола: компьютер с адресом 192.168.3.2 делает попытку узнать МАС-адрес компьютера с IP -адресом 192.168.3.12. Для этого он посылает широковещательный запрос, содержащий IP-адрес, с МАС-адресом, установленным в FF:FF:FF:FF:FF:FF .

Когда компьютер с адресом 192.168.3.12 получает этот широковещательный запрос, он анализирует IP -адрес, для которого выполняется разрешение. Определив, что его адрес совпадает с искомым, он формирует ответ протокола ARP, где указывает свой МАС-адрес.

Ответ посылается уже не широковещательно - отправитель знает МАС-адрес инициатора запроса и поэтому передает пакет целенаправленно.

ARP-таблица заполняется автоматически модулем ARP, по мере необходимости. Когда с помощью существующей ARP-таблицы не удается преобразовать IP-адрес, то происходит следующее: 1. По сети передается широковещательный ARP-запрос. 2. Исходящий IP-пакет ставится в очередь. Каждый сетевой адаптер принимает широковещательные передачи. Все драйверы Ethernet проверяют поле типа в принятом Ethernet-кадре и передают ARP-пакеты модулю ARP.

Каждый модуль ARP проверяет поле искомого IP-адреса в полученном ARP-пакете и, если адрес совпадает с его собственным IP адресом, то посылает ответ прямо по Ethernet-адресу отправителя запроса.

Этот ответ получает машина, сделавшая ARP-запрос. Драйвер этой машины проверяет поле типа в Ethernet-кадре и передает ARP пакет модулю ARP. Модуль ARP анализирует ARP-пакет и добавляет запись в свою ARP-таблицу.

Полностью порядок преобразования адресов выглядит так:

1. По сети передается широковещательный ARP-запрос.

2. Исходящий IP-пакет ставится в очередь.

3. Возвращается ARP-ответ, содержащий информацию о соответствии IP- и Ethernet-адресов. Эта информация заносится в ARP-таблицу.

4. Для преобразования IP-адреса в Ethernet-адрес у IP-пакета, постав ленного в очередь, используется ARP-таблица.

5. Ethernet-кадр передается по сети Ethernet.

Протокол RARP - это протокол, решающий обратную задачу - нахождение IP-адреса по известному локальному адресу. Он называется реверсивный ARP - RARP (Reverse Address Resolution Protocol) и используется при старте бездисковых станций, не знающих в начальный момент своего IP-адреса, но знающих адрес своего сетевого адаптера. Reverse ARP (или обратное разрешение) работает аналогично протоколу ARP за исключением того, что в его задачи входит определение изического адреса по известному адресу сетевого уровня. Этот протокол требует наличия в сети сервера RARP , подключенного к тому же сегменту сети, что и интерфейс маршрутизатора. Наиболее часто протокол reverse ARP используется для запуска бездисковых рабочих станций.

Используемые функции

• GetAdaptersAddresses – возвращает информацию об интерфейсах текущего компьютера. Family – AF\_INET для данной лабораторной работы, Flags – флаги, Reserverd – неиспользуемое поле, AdapterAdresses – указатель на буфер с адресами, SizePointer – указатель на размер буфера. Если размер буфера недостаточно большой, в SizePointer пишется необходимое количество памяти.

• GetIpNetTable2 – возвращает ARP-таблицу, Family – AF\_INET для данной лабораторной работы, Table – указатель на табллицу.

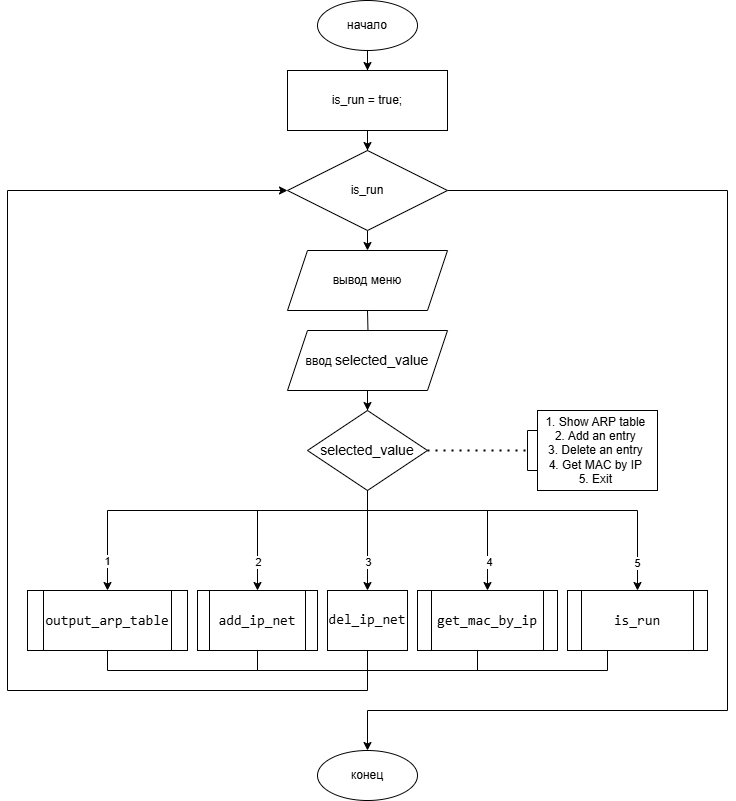
• CreateIpNetEntry2 – добавляет запись в ARP-таблицу, Row – адрес на добавляемый ряд

• DeleteIpNetEntry2 – удаляет запись из ARP-таблицы, Row – адрес на удаляемый ряд

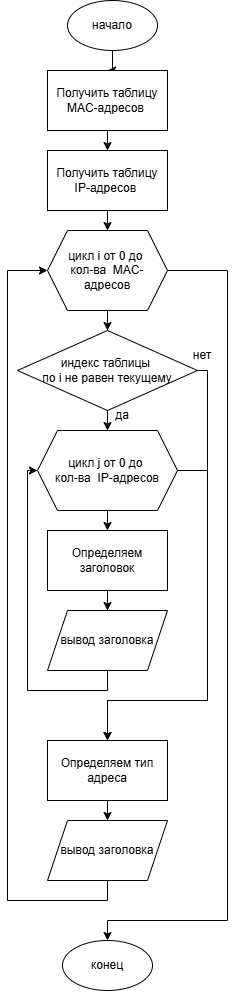
• SendARP – отправляет ARP-запрос, DestIP – IP адрес который нужно найти, SrcIP – принимающий IP адрес, можно указать ADDR\_ANY, pMacAddr – указатель на MAC-адрес, результат работы ARP-запроса, PhyAddrLen – указатель на длину MAC адреса

**Разработка программы. Блок-схемы программы**

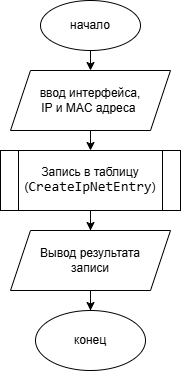
main():



output\_arp\_table():



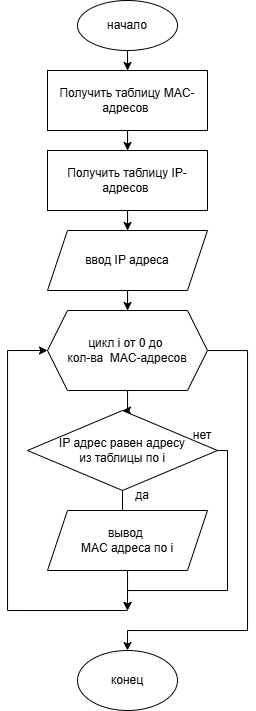
add\_ip\_net():



del\_ip\_net():

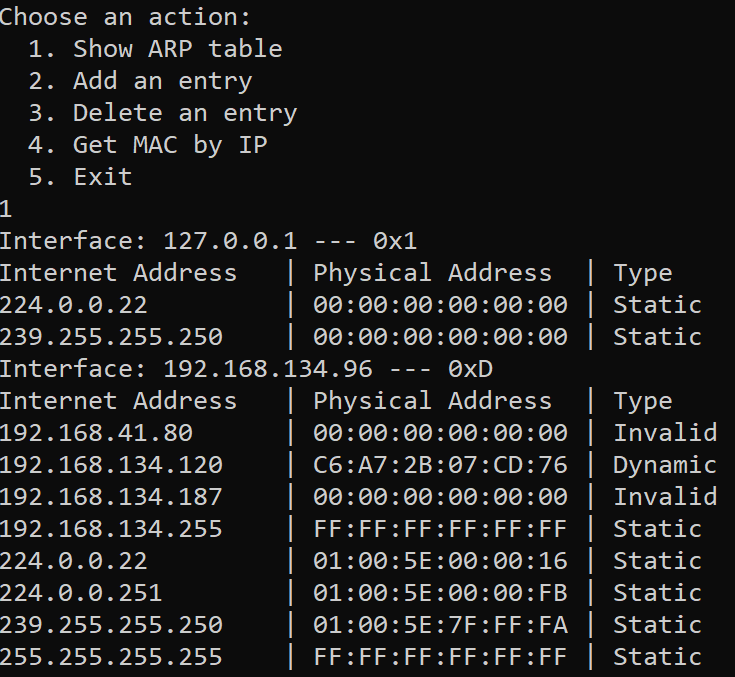


get\_mac\_by\_ip():

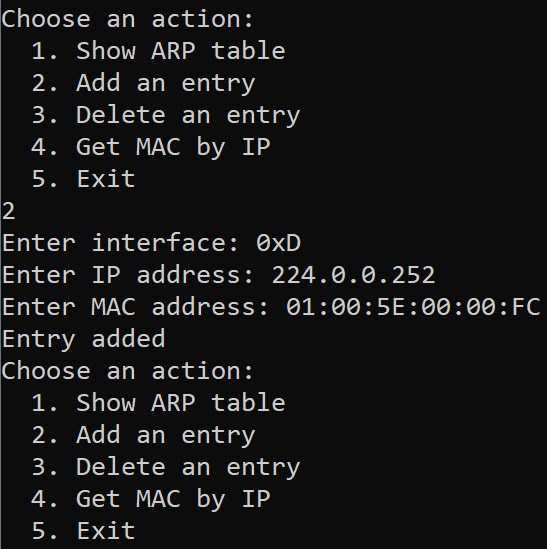
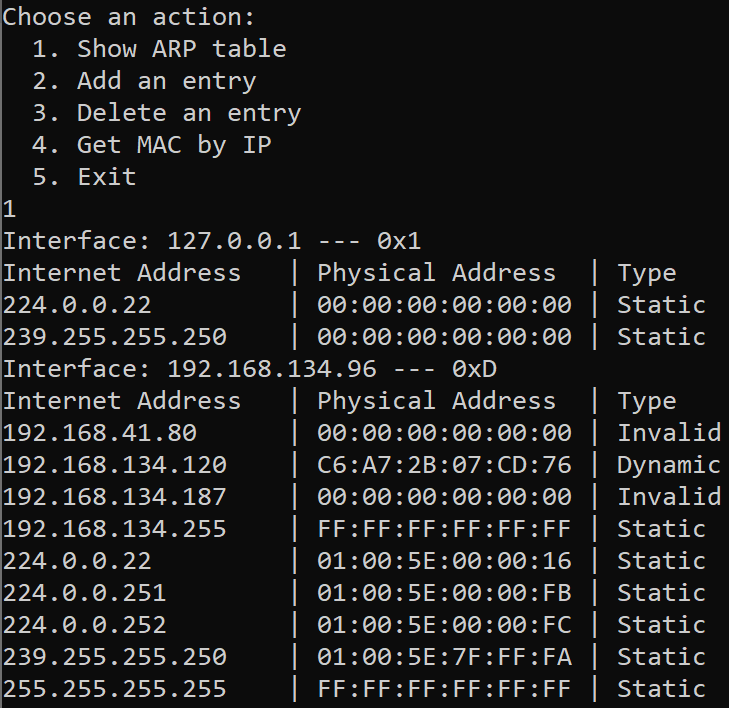


**Анализ функционирования программы**

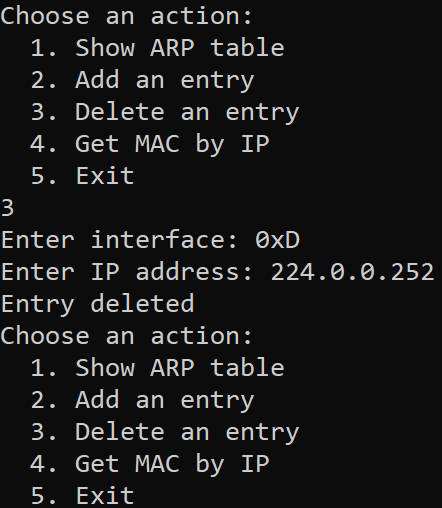
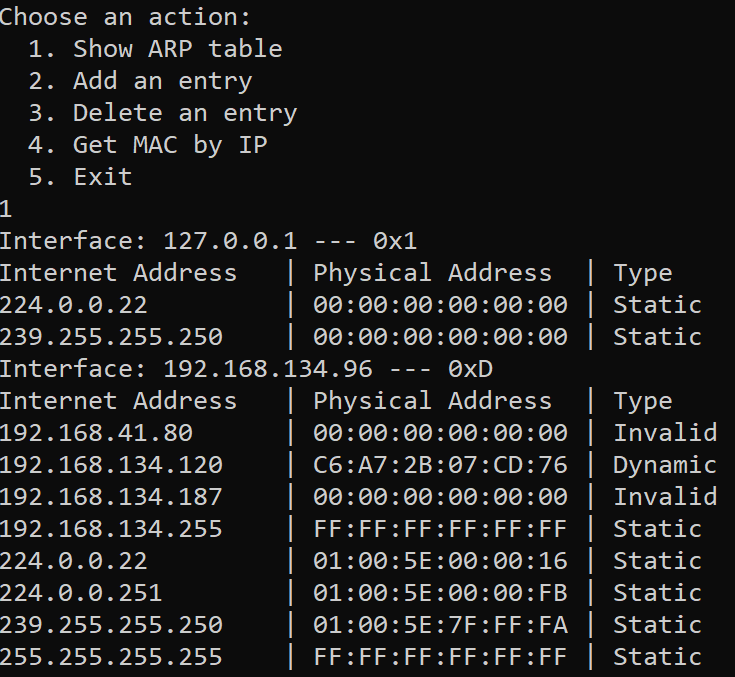
Просмотр ARP таблицы:



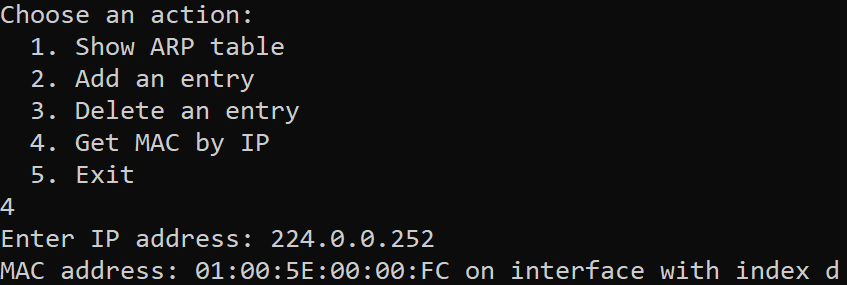
Добавление адреса:

Удаление адреса:

Вывод MAC-адреса по IP:



**Вывод:** в ходе работы изучены протоколы ARP/RARP. Реализованы функции вывода ARP-таблицы, добавление записи в ARP-таблицу, удаление записи из ARP-таблицы, получение MAC-адреса по IP-адресу.

Код программы:

#include <cstdlib>

#include <iostream>

#include <winsock.h>

#include <iphlpapi.h>

using namespace std;

// Функция вывода ARP-таблицы

int output\_arp\_table()

{

    PMIB\_IPNETTABLE ip\_arp\_table = NULL;

    DWORD actual\_size = 0;

    // Получаем размер таблицы

    GetIpNetTable(ip\_arp\_table, &actual\_size, true);

    ip\_arp\_table = (PMIB\_IPNETTABLE)malloc(actual\_size);

    // Получаем ARP-таблицу

    if (GetIpNetTable(ip\_arp\_table, &actual\_size, true) != NO\_ERROR)

    {

        cout << "Error getting ARP table\n" << endl;

        if (ip\_arp\_table) free(ip\_arp\_table);

        return 1;

    }

    // Переменные для хранения индекса интерфейса, типа записи и IP-адреса

    DWORD current\_index;

    char type[256], address[256];

    PMIB\_IPADDRTABLE ip\_address\_table = NULL;

    actual\_size = 0;

    // Получаем таблицу IP-адресов интерфейсов

    GetIpAddrTable(ip\_address\_table, &actual\_size, true);

    ip\_address\_table = (PMIB\_IPADDRTABLE)malloc(actual\_size);

    GetIpAddrTable(ip\_address\_table, &actual\_size, true);

    current\_index = -1;

    // Перебираем все записи ARP-таблицы

    for (int i = 0; i < ip\_arp\_table->dwNumEntries; i++)

    {

        // Если интерфейс сменился — выводим его IP и индекс

        if (ip\_arp\_table->table[i].dwIndex != current\_index)

        {

            current\_index = ip\_arp\_table->table[i].dwIndex;

            IN\_ADDR in\_address;

            for (int j = 0; j < ip\_address\_table->dwNumEntries; j++)

            {

                if (current\_index != ip\_address\_table->table[j].dwIndex)

                    continue;

                in\_address.S\_un.S\_addr = ip\_address\_table->table[j].dwAddr;

                strcpy(address, inet\_ntoa(in\_address));

            }

            printf("Interface: %s --- 0x%X\n", address, current\_index);

            cout << "Internet Address   | Physical Address  | Type" << endl;

        }

        // Определяем тип ARP-записи

        switch (ip\_arp\_table->table[i].dwType)

        {

            case 1: strcpy(type, "Other"); break;

            case 2: strcpy(type, "Invalid"); break;

            case 3: strcpy(type, "Dynamic"); break;

            case 4: strcpy(type, "Static"); break;

            default: strcpy(type, "");

        }

        // Выводим IP, MAC и тип

        IN\_ADDR in\_address;

        in\_address.S\_un.S\_addr = ip\_arp\_table->table[i].dwAddr;

        printf("%-18s |", inet\_ntoa(in\_address));

        printf(" %02X:%02X:%02X:%02X:%02X:%02X | %-11s\n",

               ip\_arp\_table->table[i].bPhysAddr[0],

               ip\_arp\_table->table[i].bPhysAddr[1],

               ip\_arp\_table->table[i].bPhysAddr[2],

               ip\_arp\_table->table[i].bPhysAddr[3],

               ip\_arp\_table->table[i].bPhysAddr[4],

               ip\_arp\_table->table[i].bPhysAddr[5], type);

    }

    free(ip\_arp\_table);

    cout << endl;

    return 0;

}

// Функция добавления ARP-записи

int add\_ip\_net()

{

    char array\_inet\_address[255], mac\_address[255], net\_interface[255];

    cout << "Enter interface: ";

    cin >> net\_interface;

    cout << "Enter IP address: ";

    cin >> array\_inet\_address;

    cout << "Enter MAC address: ";

    cin >> mac\_address;

    DWORD inet\_address = inet\_addr(array\_inet\_address);

    if (inet\_address == INADDR\_NONE)

    {

        cout << "Invalid IP address.\n" << endl;

        return 1;

    }

    MIB\_IPNETROW arp\_row;

    sscanf(net\_interface, "%x", &(arp\_row.dwIndex)); // Преобразуем строку интерфейса в hex-индекс

    arp\_row.dwPhysAddrLen = 6;

    // Считываем MAC-адрес

    sscanf(mac\_address, "%hx:%hx:%hx:%hx:%hx:%hx",

           &arp\_row.bPhysAddr[0],

           &arp\_row.bPhysAddr[1],

           &arp\_row.bPhysAddr[2],

           &arp\_row.bPhysAddr[3],

           &arp\_row.bPhysAddr[4],

           &arp\_row.bPhysAddr[5]);

    arp\_row.dwAddr = inet\_address;

    arp\_row.dwType = MIB\_IPNET\_TYPE\_STATIC; // Тип записи — статическая

    // Пытаемся добавить запись

    switch (CreateIpNetEntry(&arp\_row))

    {

        case ERROR\_ACCESS\_DENIED:

            cout << "Entry not added. Access denied" << endl;

            break;

        case NO\_ERROR:

            cout << "Entry added" << endl;

            break;

        default:

            cout << "Entry not added" << endl;

    }

    return 0;

}

// Функция удаления ARP-записи

int del\_ip\_net()

{

    char array\_inet\_address[255], net\_interface[255];

    cout << "Enter interface: ";

    cin >> net\_interface;

    cout << "Enter IP address: ";

    cin >> array\_inet\_address;

    DWORD inet\_address = inet\_addr(array\_inet\_address);

    if (inet\_address == INADDR\_NONE)

    {

        cout << "Invalid IP address" << endl;

        return 1;

    }

    MIB\_IPNETROW arp\_row;

    sscanf(net\_interface, "%x", &(arp\_row.dwIndex));

    arp\_row.dwAddr = inet\_address;

    // Пытаемся удалить запись

    switch (DeleteIpNetEntry(&arp\_row))

    {

        case ERROR\_ACCESS\_DENIED:

            cout << "Entry not deleted. Access denied" << endl;

            break;

        case NO\_ERROR:

            cout << "Entry deleted" << endl;

            break;

        default:

            cout << "Entry not deleted" << endl;

    }

    return 0;

}

// Функция поиска MAC-адреса по IP-адресу

void get\_mac\_by\_ip()

{

    DWORD actual\_size = 0;

    PMIB\_IPNETTABLE ip\_address\_table = NULL;

    // Получаем таблицу ARP

    GetIpNetTable(ip\_address\_table, &actual\_size, true);

    ip\_address\_table = (PMIB\_IPNETTABLE)malloc(actual\_size);

    GetIpNetTable(ip\_address\_table, &actual\_size, true);

    char array\_inet\_address[255];

    cout << "Enter IP address: ";

    cin >> array\_inet\_address;

    DWORD inet\_address = inet\_addr(array\_inet\_address);

    if (inet\_address == INADDR\_NONE)

    {

        cout << "Invalid IP address" << endl;

        return;

    }

    bool search\_flag = true;

    // Поиск MAC-адреса по IP

    for (int i = 0; i < ip\_address\_table->dwNumEntries; i++)

    {

        if (inet\_address == ip\_address\_table->table[i].dwAddr)

        {

            printf("MAC address: %02X:%02X:%02X:%02X:%02X:%02X on interface with index %x\n",

                   ip\_address\_table->table[i].bPhysAddr[0],

                   ip\_address\_table->table[i].bPhysAddr[1],

                   ip\_address\_table->table[i].bPhysAddr[2],

                   ip\_address\_table->table[i].bPhysAddr[3],

                   ip\_address\_table->table[i].bPhysAddr[4],

                   ip\_address\_table->table[i].bPhysAddr[5], ip\_address\_table->table[i].dwIndex);

            search\_flag = false;

        }

    }

    if (search\_flag)

        cout << "No matches found" << endl;

}

// Основное меню программы

int main(int argc, char \*argv[])

{

    int selected\_value;

    bool is\_run = true;

    // Меню

    while (is\_run)

    {

        cout << "Choose an action:" << endl

             << "  1. Show ARP table" << endl

             << "  2. Add an entry" << endl

             << "  3. Delete an entry" << endl

             << "  4. Get MAC by IP" << endl

             << "  5. Exit" << endl;

        cin >> selected\_value;

        switch (selected\_value)

        {

            case 1:

                output\_arp\_table();

                break;

            case 2:

                add\_ip\_net();

                break;

            case 3:

                del\_ip\_net();

                break;

            case 4:

                get\_mac\_by\_ip();

                break;

            case 5:

            default:

                is\_run = false;

                break;

        }

    }

    return 0;

}