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 MAC-адрес компьютера с IP -адресом 192.168.3.12. Для этого он посылает широковещательный запрос, содержащий IP-адрес, с MAC-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 MAC-адрес.

Ответ посылается уже не широковещательно - отправитель знает MAC-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.

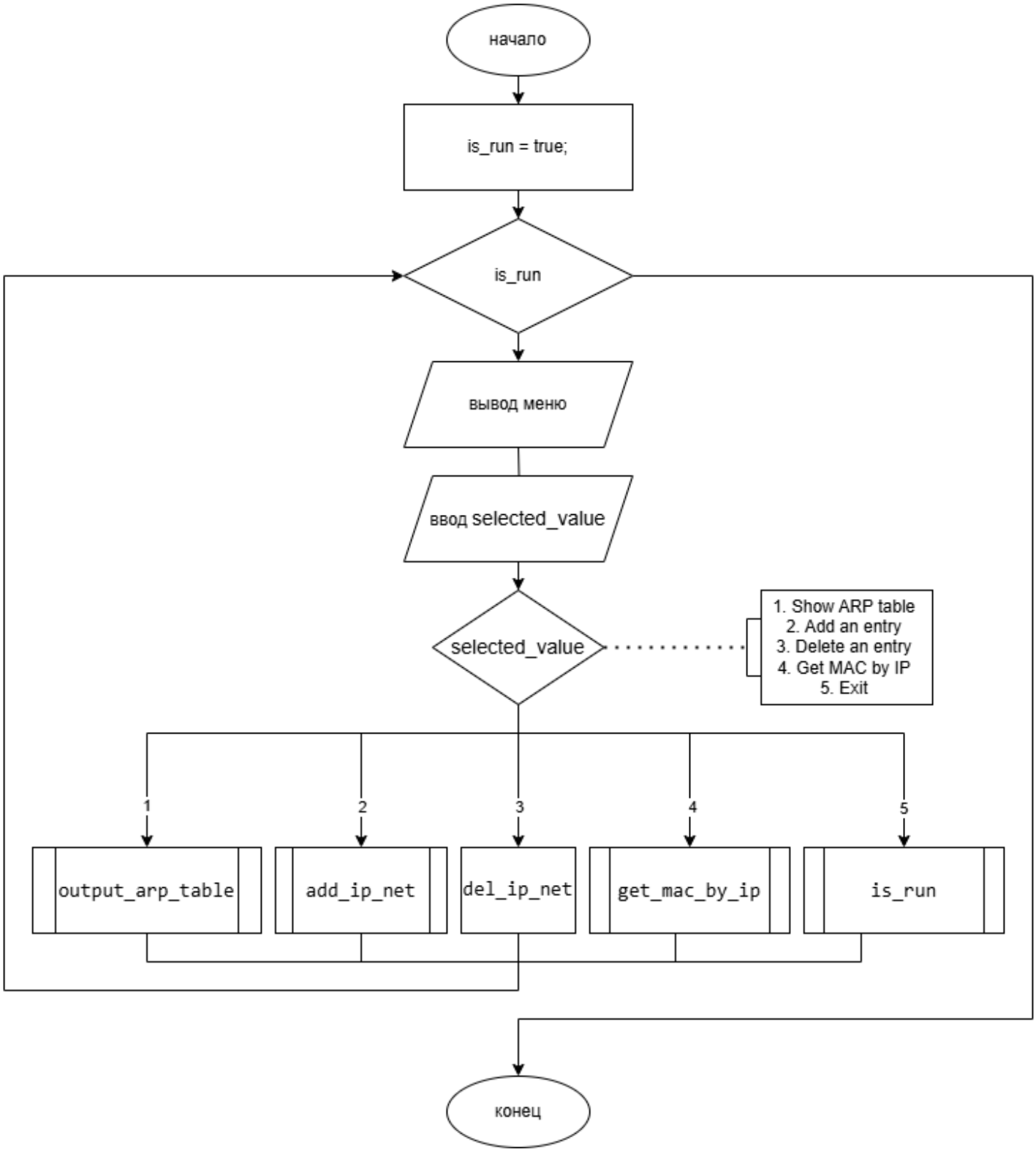
• CreateIpNetEntry2 – добавляет запись в ARP-таблицу, Row – адрес на добавляемый ряд

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

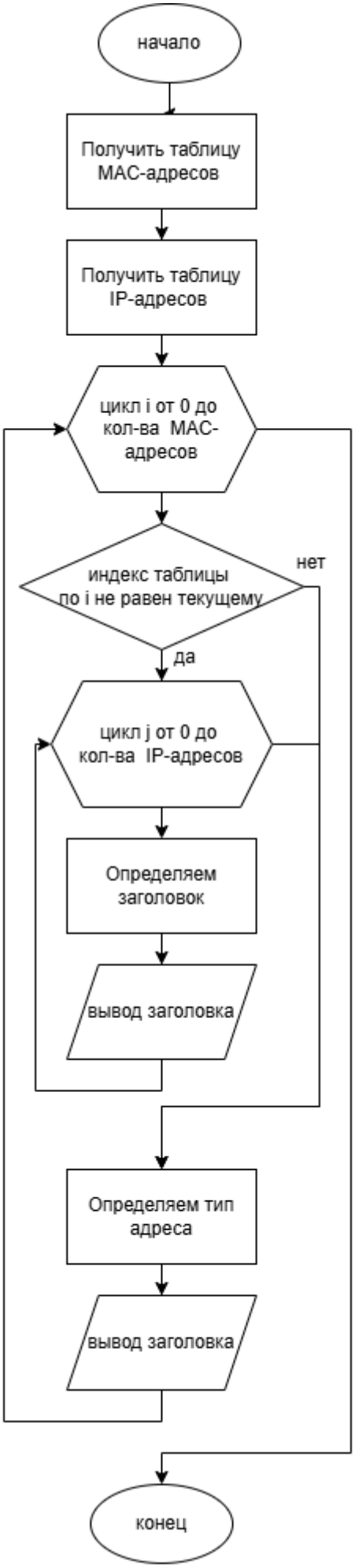
• SendARP – отправляет ARP-запрос, DestIP – IP адрес который нужно найти, SrcIP – принимающий IP адрес, можно указать ADDR_ANY, pMacAddr – указатель на MAC-адрес, результат работы ARP-запроса, PhyAddrLen – указатель на длину MAC адреса

Разработка программы. Блок-схемы программы

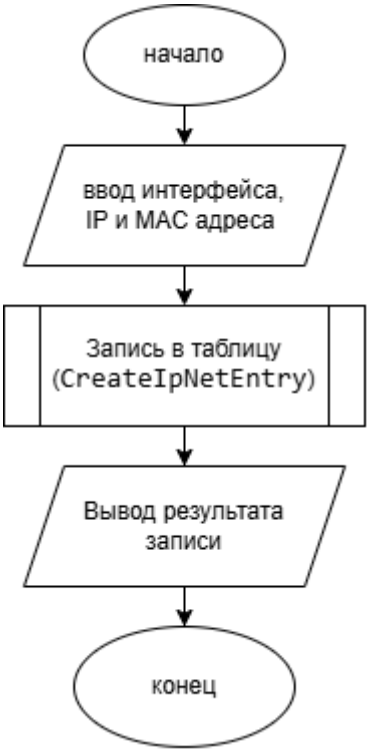
main():



output_arp_table():

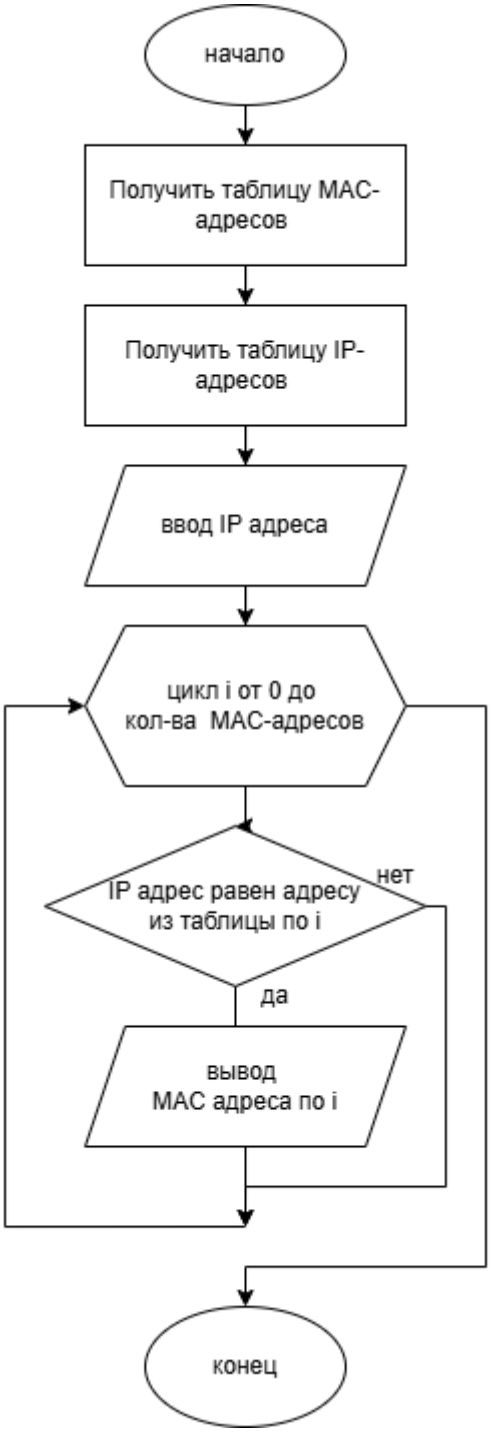


add_ip_net():



del_ip_net():

get_mac_by_ip():


начало
Получить таблицу MAC-адресов
Получить таблицу IP-адресов
ввод IP адреса
цикл i от 0 до кол-ва MAC-адресов
IP адрес равен адресу из таблицы по i
нет
да
вывод MAC адреса по i
конец

Анализ функционирования программы

Просмотр ARP таблицы:

```
Choose an action:
  1. Show ARP table
  2. Add an entry
  3. Delete an entry
  4. Get MAC by IP
  5. Exit
1
Interface: 127.0.0.1 --- 0x1
Internet Address    | Physical Address   | Type
224.0.0.22          | 00:00:00:00:00:00 | Static
239.255.255.250     | 00:00:00:00:00:00 | Static
Interface: 192.168.134.96 --- 0xD
Internet Address    | Physical Address   | Type
192.168.41.80       | 00:00:00:00:00:00 | Invalid
192.168.134.120     | C6:A7:2B:07:CD:76 | Dynamic
192.168.134.187     | 00:00:00:00:00:00 | Invalid
192.168.134.255     | FF:FF:FF:FF:FF:FF | Static
224.0.0.22          | 01:00:5E:00:00:16 | Static
224.0.0.251         | 01:00:5E:00:00:FB | Static
239.255.255.250     | 01:00:5E:7F:FF:FA | Static
255.255.255.255     | FF:FF:FF:FF:FF:FF | Static
```

Добавление адреса:

```
Choose an action:
  1. Show ARP table
  2. Add an entry
  3. Delete an entry
  4. Get MAC by IP
  5. Exit
2
Enter interface: 0xD
Enter IP address: 224.0.0.252
Enter MAC address: 01:00:5E:00:00:FC
Entry added
Choose an action:
  1. Show ARP table
  2. Add an entry
  3. Delete an entry
  4. Get MAC by IP
  5. Exit
```

```
Choose an action:
  1. Show ARP table
  2. Add an entry
  3. Delete an entry
  4. Get MAC by IP
  5. Exit
1
Interface: 127.0.0.1 --- 0x1
Internet Address    | Physical Address   | Type
224.0.0.22          | 00:00:00:00:00:00 | Static
239.255.255.250     | 00:00:00:00:00:00 | Static
Interface: 192.168.134.96 --- 0xD
Internet Address    | Physical Address   | Type
192.168.41.80       | 00:00:00:00:00:00 | Invalid
192.168.134.120     | C6:A7:2B:07:CD:76 | Dynamic
192.168.134.187     | 00:00:00:00:00:00 | Invalid
192.168.134.255     | FF:FF:FF:FF:FF:FF | Static
224.0.0.22          | 01:00:5E:00:00:16 | Static
224.0.0.251         | 01:00:5E:00:00:FB | Static
224.0.0.252         | 01:00:5E:00:00:FC | Static
239.255.255.250     | 01:00:5E:7F:FF:FA | Static
255.255.255.255     | FF:FF:FF:FF:FF:FF | Static
```

Удаление адреса:

```
Choose an action:
  1. Show ARP table
  2. Add an entry
  3. Delete an entry
  4. Get MAC by IP
  5. Exit
3
Enter interface: 0xD
Enter IP address: 224.0.0.252
Entry deleted
Choose an action:
  1. Show ARP table
  2. Add an entry
  3. Delete an entry
  4. Get MAC by IP
  5. Exit
```

```
Choose an action:
  1. Show ARP table
  2. Add an entry
  3. Delete an entry
  4. Get MAC by IP
  5. Exit
1
Interface: 127.0.0.1 --- 0x1
Internet Address   | Physical Address  | Type
224.0.0.22         | 00:00:00:00:00:00 | Static
239.255.255.250    | 00:00:00:00:00:00 | Static
Interface: 192.168.134.96 --- 0xD
Internet Address   | Physical Address  | Type
192.168.41.80      | 00:00:00:00:00:00 | Invalid
192.168.134.120    | C6:A7:2B:07:CD:76 | Dynamic
192.168.134.187    | 00:00:00:00:00:00 | Invalid
192.168.134.255    | FF:FF:FF:FF:FF:FF | Static
224.0.0.22         | 01:00:5E:00:00:16 | Static
224.0.0.251        | 01:00:5E:00:00:FB | Static
239.255.255.250    | 01:00:5E:7F:FF:FA | Static
255.255.255.255    | FF:FF:FF:FF:FF:FF | Static
```

Вывод MAC-адреса по IP:

```
Choose an action:
  1. Show ARP table
  2. Add an entry
  3. Delete an entry
  4. Get MAC by IP
  5. Exit
4
Enter IP address: 224.0.0.252
MAC address: 01:00:5E:00:00:FC on interface with index d
```

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;
}
```