《网络程序设计》理论课

实验报告

实验名称：winsock 编程接口实验

实验类型：验证型

指导教师：贾浩

专业班级：

姓　　名：

学　　号：

实验评分

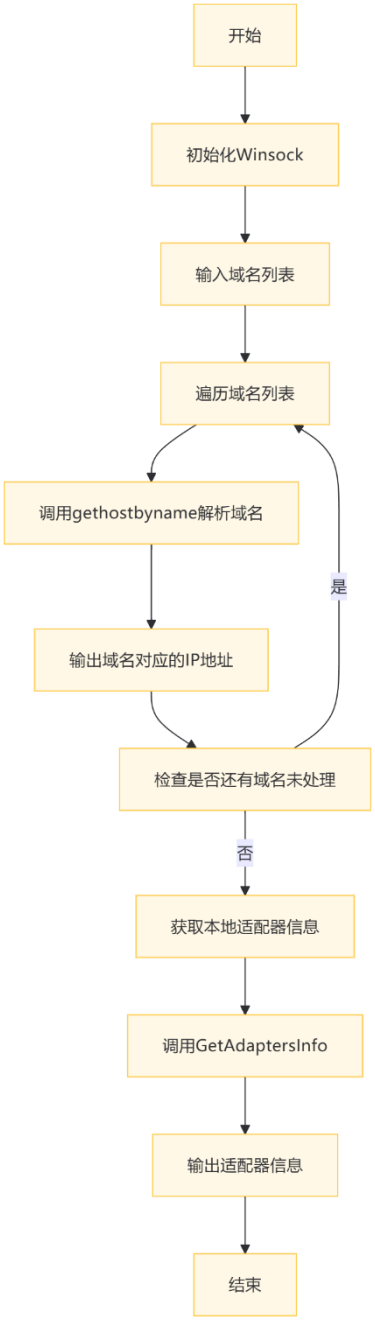
序号	实验分数构成	分数
1	实验结果(20 分)	
2	实验过程(40 分) （包括实验指导教师提问回答情况）	
3	实验报告书写规范（20 分）	
4	实验思考（20 分）	
实验总分		

一、实验目的

掌握 Winsock 的启动和初始化；
掌握 gethostname(),gethostbyname(),GetAdaptersInfo()等信息查询函数的使用。

二、实验设计

开发环境：Visual Studio 2022



该程序完成了两个功能

1. 在控制台输入域名列表能输出对应的 IP 地址列表。
2. 获取并输出本地主机的所有适配器的 IP 地址，子网掩码，默认网关，MAC 地址。

- 初始化 Winsock：通过 WSAStartup 函数初始化 Winsock 库。

- 遍历 hostent 结构体中的 IP 地址列表，使用 inet_ntoa 函数将网络字节序的 IP 地址转换为点分十进制字符串并输出。

- 域名解析：使用 gethostbyname 函数解析域名，获取 hostent 结构体。

- 获取本地适配器信息：调用 GetAdaptersInfo 函数获取本地主机的适配器信息。

- 遍历适配器信息链表，输出每个适配器的 IP 地址、子网掩码、默认网关和 MAC 地址。

三、实验过程（包含实验结果）

实验要求 1

只要 hostent 结构体的 IP 地址列表（h_addr_list），但我还输出了该结构体其他属性，比如 h_name，h_aliases，h_addrtype，h_length，h_addrtype 等，使信息更丰富。

```
请输入要查询的域名列表（输入空行结束）：
域名 1: www.163.com
域名 2: www.swust.edu.cn
域名 3:

*******************************获取IP信息*******************************
www.163.com信息如下：
        官方名称：www.163.com.w.kunluncan.com
        别名 #1: www.163.com
        别名 #2: www.163.com.163jiasu.com
        地址类型：AF_INET (IPv4)
        地址长度：4
        IP 地址 #1: 112.45.29.237
        IP 地址 #2: 112.45.29.240
        IP 地址 #3: 112.45.29.222
        IP 地址 #4: 112.45.29.101
        IP 地址 #5: 36.170.19.16
        IP 地址 #6: 112.45.29.103
www.swust.edu.cn信息如下：
        官方名称：www.swust.edu.cn
        地址类型：AF_INET (IPv4)
        地址长度：4
        IP 地址 #1: 220.166.52.236
```

实验要求 2

实验要求输出本机所有适配器的 IP 地址，子网掩码，默认网关，MAC 地址。除此之外，我还输出了 IP_ADAPTER_INFO 结构体中 AdapterName、Description、DhcpEnabled 的信息。

```
*******************************获取本机所有适配器信息*******************************
        适配器名称：    {91FA2E68-9F53-4297-9C51-0166094324D9}
        适配器描述：    Bluetooth Device (Personal Area Network)
        适配器地址：    9C-2F-9D-91-11-C4
        IP 地址：       0.0.0.0
        子网掩码：      0.0.0.0
        网关：  0.0.0.0
        DHCP 已启用：是

        适配器名称：    {B327B84B-5219-4F86-9D41-5B69CECF1F61}
        适配器描述：    VMware Virtual Ethernet Adapter for VMnet1
        适配器地址：    00-50-56-C0-00-01
        IP 地址：       192.168.121.1
        子网掩码：      255.255.255.0
        网关：  0.0.0.0
        DHCP 已启用：否

        适配器名称：    {5F0E2BA1-6175-4652-BB51-7677254E4A4D}
        适配器描述：    VMware Virtual Ethernet Adapter for VMnet8
        适配器地址：    00-50-56-C0-00-08
        IP 地址：       61.139.2.1
        子网掩码：      255.255.255.0
        网关：  0.0.0.0
        DHCP 已启用：否
```

```
        适配器名称:     {9E8EF2BD-E790-4F89-BED4-8378B1D07D69}
        适配器描述:     Realtek RTL8852AE WiFi 6 802.11ax PCIe Adapter
        适配器地址:     9C-2F-9D-91-11-C3
        IP 地址:        100.64.96.181
        子网掩码:       255.255.0.0
        网关:   100.64.0.1
        DHCP 已启用: 是

        适配器名称:     {BF20E470-C1DB-4925-A530-B5738C519AE5}
        适配器描述:     Microsoft Wi-Fi Direct Virtual Adapter
        适配器地址:     9E-2F-9D-91-11-C3
        IP 地址:        0.0.0.0
        子网掩码:       0.0.0.0
        网关:   0.0.0.0
        DHCP 已启用: 是

        适配器名称:     {ABA52819-3B3A-4996-B1F9-DF2739E33525}
        适配器描述:     Microsoft Wi-Fi Direct Virtual Adapter #2
        适配器地址:     92-2F-9D-91-11-C3
        IP 地址:        0.0.0.0
        子网掩码:       0.0.0.0
```

除此之外，我还将该程序改为手动输入域名列表，方便用户使用

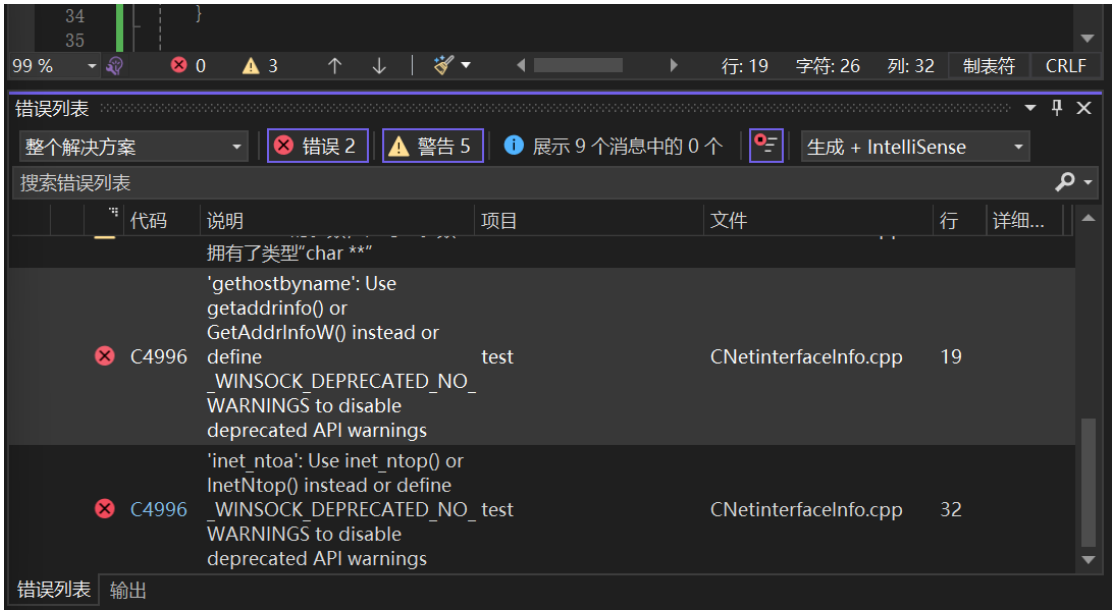
```
请输入要查询的域名列表（输入空行结束）：
域名 1: www.baidu.com
域名 2:

*******************************获取IP信息*******************************
www.baidu.com信息如下：
        官方名称: www.a.shifen.com
        别名 #1: www.baidu.com
        地址类型: AF_INET (IPv4)
        地址长度: 4
        IP 地址 #1: 39.156.70.46
        IP 地址 #2: 39.156.70.239
```

四、讨论与分析（实验思考中问题理解）

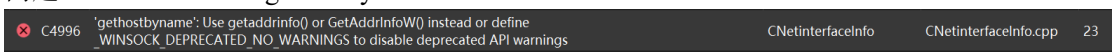
问题 1：在调用 GetAdaptersInfo 函数时，链接器报错，提示未找到 iphlpapi.lib。
解决方案：引入“iphlpapi.h”头文件并添加 iphlpapi.lib 库。

问题 2：在调用 inet_ntoa 函数时，编译器报错，提示未找到 arpa/inet.h 头文件。
解决方案：在 Windows 环境下，inet_ntoa 函数定义在 ws2tcpip.h 中，确保已包含该头文件。



问题 3：输出中文乱码
解决方案：参照这篇文章修改了编码，重启即可输出正常中文。
https://blog.csdn.net/Wumingdegushi/article/details/142220421
错误原因：可能是 VS 的字符集和控制台的不一致导致的。

问题 4：error C4996: 'gethostbyname'



解决方案：参照下面这篇文章进行了修改，在预处理器定义中加入_WINSOCK_DEPRECATED_NO_WARNINGS
https://blog.csdn.net/morningstar929/article/details/101024153
原因分析：该错误表明在 VSgethostbyname 函数被视为已过时的 API，在较新的 Windows 开发环境中不再被推荐使用。

实验思考 1：Winsock 初始化的作用
Winsock 初始化通过 WSAStartup 函数完成，其主要作用是加载 Winsock 库并初始化网络编程环境。只有在成功初始化后，才能调用 Winsock 提供的网络编程相关函数。

实验思考 2：GetAdaptersInfo 函数的正确使用方法
首先，需要包含 iphlpapi.h 头文件，并在项目中链接 iphlpapi.lib 库。调用 GetAdaptersInfo 函数时，还需要先分配一个足够大的缓冲区来存储适配器信息。如果缓冲区大小不足，函数会返回 ERROR_BUFFER_OVERFLOW，此时需要根据返回的缓冲区大小重新分配内存。调用成功后，通过遍历返回的适配器信息链表，可以获取每个适配器的详细信息。

实验思考 3：域名解析时出现域名对应多个 IP 的原因
分散服务器负载：域名解析服务会将一个域名映射到多个 IP 地址，通过轮询等方式将请求分配到不同的服务器，提高网站的响应速度和可用性。
避免单点故障：部署多个服务器实例，每个实例都有一个 IP 地址，域名解析服务会返回这些 IP 地址，以确保在某个服务器故障时，用户仍能访问其他服务器。

提高访问速度：域名解析服务可能会将域名解析为离用户最近的服务器 IP 地址，以减少网络延迟。

五、实验者自评（从实验设计、实验过程、对实验知识点的理解上给出客观公正的自我评价）

通过此次实验，我对网络编程领域中的域名解析以及适配器信息获取知识有了更为深刻且全面的认知。在实验推进的过程中，我熟练掌握了 gethostbyname 以及 GetAdaptersInfo 函数的运用方法，并且成功攻克了在编译以及链接环节所遭遇的一系列难题。让我认识到在网络编程这一专业领域，库的正确链接、缓冲区大小的合理分配等这些细节因素都可能对整个编程过程产生重大影响。

六、 附录：关键代码（给出适当注释，可读性高）

```
// 初始化 Winsock 库并解析域名信息
void InitializeWinsock(const char** domains, int count) {
    WSADATA wasData;
    // 初始化 Winsock 2.2 版本
    int result = WSAStartup(MAKEWORD(2, 2), &wasData);
    if (result != 0) {
        printf("WSAStartup 启动失败: %d\n", result);
        return;
    }

    // 遍历所有域名进行解析
    for (int i = 0; i < count; i++) {
        struct hostent* host = gethostbyname(domains[i]);  // 核心 DNS 解析函数
        if (host == NULL) {
            // 错误处理逻辑
            DWORD dwError = WSAGetLastError();
            if (dwError == WSAHOST_NOT_FOUND) {
                printf("未找到主机：%s\n", domains[i]);
            } else {
                printf("解析失败，错误代码: %ld\n", dwError);
            }
        } else {
            // 输出主机基本信息
            printf("%s 信息如下:\n", domains[i]);
            printf("\t 官方名称: %s\n", host->h_name);

            // 遍历别名列表
```

```
            int aliasIndex = 0;
            for (char** pAlias = host->h_aliases; *pAlias != nullptr; pAlias++) {
                printf("\t 别名  #%d: %s\n", ++aliasIndex, *pAlias);
            }

            // 处理地址类型
            printf("\t 地址类型: ");
            if (host->h_addrtype == AF_INET) {
                printf("IPv4 地址\n");
                // 遍历 IP 地址列表
                int ipIndex = 0;
                while (host->h_addr_list[ipIndex] != nullptr) {
                    struct in_addr addr;
                    addr.s_addr = *(u_long*)host->h_addr_list[ipIndex++];
                    printf("\tIP 地址  #%d: %s\n", ipIndex, inet_ntoa(addr));
                }
            } else {
                printf("未知类型(%d)\n", host->h_addrtype);
            }
        }
    }
    WSACleanup();   // 清理 Winsock 资源
}

// 获取本地网络适配器信息
void AdapterInformation() {
    PIP_ADAPTER_INFO pAdapterInfo;
    DWORD dwRetVal = 0;
    ULONG ulOutBufLen = sizeof(IP_ADAPTER_INFO);

    // 首次调用获取所需缓冲区大小
    pAdapterInfo = (IP_ADAPTER_INFO*)malloc(ulOutBufLen);
    if (GetAdaptersInfo(pAdapterInfo, &ulOutBufLen) == ERROR_BUFFER_OVERFLOW) {
        free(pAdapterInfo);
        pAdapterInfo = (IP_ADAPTER_INFO*)malloc(ulOutBufLen);   // 重新分配足够内存
    }

    if ((dwRetVal = GetAdaptersInfo(pAdapterInfo, &ulOutBufLen)) == NO_ERROR) {
        PIP_ADAPTER_INFO pAdapter = pAdapterInfo;
        while (pAdapter) {
            // 输出适配器基本信息
            printf("\t 适配器名称: %s\n", pAdapter->AdapterName);
            printf("\t 描述: %s\n", pAdapter->Description);
```

```
                // 格式化输出 MAC 地址
                printf("\tMAC 地址: ");
                for (UINT i = 0; i < pAdapter->AddressLength; i++) {
                    printf("%02X%s", pAdapter->Address[i],
                            (i == pAdapter->AddressLength - 1) ? "\n" : "-");
                }

                // 输出 IP 配置信息
                printf("\tIP 地址: %s\n", pAdapter->IpAddressList.IpAddress.String);
                printf("\t 子网掩码: %s\n", pAdapter->IpAddressList.IpMask.String);
                printf("\t 网关: %s\n", pAdapter->GatewayList.IpAddress.String);
                printf("\tDHCP 状态: %s\n\n",
                        pAdapter->DhcpEnabled ? "已启用" : "未启用");

                pAdapter = pAdapter->Next;  // 遍历下一个适配器
            }
        } else {
            printf("获取适配器信息失败，错误码：%d\n", dwRetVal);
        }
        free(pAdapterInfo);  // 释放内存
}

// 主函数
int _tmain(int argc, _TCHAR* argv[]) {
    // 用户输入处理
    std::vector<std::string> domainList;
    std::string input;
    while (true) {
        std::getline(std::cin, input);
        if (input.empty()) break;
        domainList.push_back(input);
    }

    // 转换字符串数组格式
    std::vector<const char*> domains;
    for (const auto& domain : domainList) {
        domains.push_back(domain.c_str());  // 转换为 C 风格字符串
    }

    // 执行核心功能
    InitializeWinsock(domains.data(), domains.size());
    AdapterInformation();
    return 0;
}
```