1. 代码实践

```
//通讯录管理系统
#include <iostream>
#include <string>
#define MAX 1000
using namespace std;

//联系人结构体
struct Person
{
    string m_name; //姓名
    int m_Sex; //性别 1：男 2：女
    int m_Age; //年龄
    string m_phone; //电话
    string m_Addr; //住址
};

//通讯录结构体
struct Addressbooks
{
    struct Person personArray[MAX];
    int m_size; //通讯录中当前的人数
};

void addPerson(Addressbooks *abs)
{
    //先判断通讯录是否已满，如果满了就不再添加
    if(abs->m_size==MAX)
        cout << "通讯录已满，请勿继续添加" << endl;
    else
    {
        //添加具体联系人
        string name;
        cout << "请输入姓名：  " << endl;
        cin >> name;
        abs->personArray[abs->m_size].m_name=name;

        int sex; //性别 1：男 2：女
        while(true)
        {
            cout << "请输入性别：  " << endl;
            cin >> sex;
            if(sex==1||sex==2)
```

```
            {
                abs->personArray[abs->m_size].m_Sex=sex;
                break;
            }
            else
               cout << "输入有误，请重新输入" << endl;
        };

        int age; //年龄
        cout << "请输入年龄:  " << endl;
        cin >> age;
        abs->personArray[abs->m_size].m_Age=age;

        string phone; //电话
        cout << "请输入电话:  " << endl;
        cin >> phone;
        abs->personArray[abs->m_size].m_phone=phone;

        string addr; //住址
        cout << "请输入住址:  " << endl;
        cin >> addr;
        abs->personArray[abs->m_size].m_Addr=addr;

        abs->m_size++;
        cout << "添加成功" << endl;

        system("pause"); //请按任意键继续
        system("cls"); //清屏
    }
}

void showPerson(Addressbooks *abs)
{
    if(abs->m_size==0)
    {
        cout << "当前记录为空" << endl;
    }
    else
        for(int i=0;i<abs->m_size;i++)
        {
            cout << "姓名:  " << abs->personArray[i].m_name << "\t" ;
            cout << "性别:  " << (abs->personArray[i].m_Sex==1?"男":"女
") << "\t" ;
            cout << "年龄:  " << abs->personArray[i].m_Age << "\t" ;
```

```
            cout << "电话：  " << abs->personArray[i].m_phone << "\t" ;
            cout << "地址：  " << abs->personArray[i].m_Addr << "\t" ;
        }
    system("pause"); //请按任意键继续
    system("cls"); //清屏
}
void showMenu()
{
    cout << "*****     1. 添加联系人     *****" << endl;
    cout << "*****     2. 显示联系人     *****" << endl;
    cout << "*****     3. 删除联系人     *****" << endl;
    cout << "*****     4. 查找联系人     *****" << endl;
    cout << "*****     5. 修改联系人     *****" << endl;
    cout << "*****     6. 清空联系人     *****" << endl;
    cout << "*****     0. 退出通讯录     *****" << endl;
}
int main()
{
    Addressbooks abs; //创建通讯录结构体变量
    abs.m_size=0; //初始化通讯录中当前人员个数

    int select;
    while(true)
    {
        showMenu();
        cin >> select;
        switch(select)
        {
            case 1://添加联系人
                addPerson(&abs);
                break;
            case 2://显示联系人
                showPerson(&abs);
                break;
            case 3://删除联系人
                break;
            case 4://查找联系人
                break;
            case 5://修改联系人
                break;
            case 6://清空联系人
                break;
            case 0://退出通讯录
                break;
```

```
            default:
                break;

        }
    }
    system("pause");
    return 0;
}
```

2. 计算机基础知识整理

2.1 TCP 对应的协议和 UDP 对应的协议

TCP 对应的协议：

（1） FTP ：定义了文件传输协议，使用 21 端口。常说某某计算机开了 FTP 服务便是启动了文件传输服务。下载文件，上传主页，都要用到 FTP 服务。

（2） Telnet ：它是一种用于远程登陆的端口，用户可以以自己的身份远程连接到计算机上，通过这种端口可以提供一种基于 DOS 模式下的通信服务。如以前的 BBS 是纯字符界面的，支持 BBS 的服务器将 23 端口打开，对外提供服务。

（3） SMTP ：定义了简单邮件传送协议，现在很多邮件服务器都用的是这个协议，用于发送邮件。如常见的免费邮件服务中用的就是这个邮件服务端口，所以在电子邮件设置中常看到有这么 SMTP 端口设置这个栏，服务器开放的是 25 号端口。

（4） POP3 ：它是和 SMTP 对应，POP3 用于接收邮件。通常情况下，POP3 协议所用的是 110 端口。也是说，只要你有相应的使用 POP3 协议的程序，就可以不以 Web 方式登陆进邮箱界面，直接用邮件程序就可以收到邮件（如是 163 邮箱就没有必要先进入网易网站，再进入自己的邮箱来收信）。

（5）HTTP 协议： 是从 Web 服务器传输超文本到本地浏览器的传送协议。

UDP 对应的协议：

（1） DNS ：用于域名解析服务，将域名地址转换为 IP 地址。DNS 用的是 53 号端口。

（2） SNMP ：简单网络管理协议，使用 161 号端口，是用来管理网络设备的。由于网络设备很多，无连接的服务就体现出其优势。

（3） TFTP (Trival File Transfer Protocal)，简单文件传输协议，该协议在熟知端口 69 上使用 UDP 服务。

2.2　何为死锁？何为系统调用？

答：

- B-树的特性：

①关键字集合分布在整颗树中；

②任何一个关键字出现且只出现在一个结点中；

③搜索有可能在非叶子结点结束；

④其搜索性能等价于在关键字全集内做一次二分查找；

⑤自动层次控制；

由于限制了除根结点以外的非叶子结点，至少含有 M/2 个儿子，确保了结点的至少利用率，其最底搜索性能为：O(log2n)只与节点数有关，与 M 无关。

- B+的特性：

①所有关键字都出现在叶子结点的链表中（稠密索引），且链表中的关键字恰好是有序的；

②不可能在非叶子结点命中；

③非叶子结点相当于是叶子结点的索引（稀疏索引），叶子结点相当于是存储（关键字）数据的数据层；

④更适合用于数据库和操作系统的文件系统

B+树的查找：

对 B+树可以进行两种查找运算：

1）从最小关键字起顺序查找；

2）从根结点开始，进行随机查找。

B+树是应文件系统所需而出的一种 B-树的变型树。

一棵 m 阶的 B+树和 m 阶的 B-树的差异在于：

①有 n 棵子树的结点中含有 n 个关键字，每个关键字不保存数据，只用来索引，所有数据都保存在叶子节点。

②所有的叶子结点中包含了全部关键字的信息，及指向含这些关键字记录的指针 ，且叶子结点本身依关键字的大小自小而大顺序链接。

③所有的非终端结点可以看成是索引部分，结点中仅含其子树（根结点）中的最大（或最小）关键字。通常在 B+树上有两个头指针，一个指向根结点，一个指向关键字最小的叶子结点。

3. 当日工作总结

① 熟悉了一些 Git 相关操作

② 上传学习笔记至小组仓库。