实验 1　一元多项式实验要求（8 课时）

一基本要求：

1.编写程序 polyn.c(或 polyn.cpp)实现 ADT Polynomial，可以使用下列结构实现：

```
typedef    struct{
    float      p;        //系数
    int        e;        //指数
}ElemType;
```

实现基本操作：

CreatePolyn(&p,m)，创建一元多项式，可从终端接受 m 组（p,e）系数/指数组。
AddPolyn(&pa,&pb)，　实现两个一元多项式的加法
SubtractPolyn(&pa,&pb)　实现两个一元多项式的减法
PrintPolyn(p)　　实现在屏幕打印出一个一元多项式，形如 3x^10+9x^5+8
DostroyPolyn(&p);　　销毁多项式

2.编写主程序,实现菜单

1）创建(两个一元多项式 pa、pb)
2）打印
3）求和
4）求差
5）销毁
6）退出

示例数据：

pa=3x^20+12x^15-9x^12+6x^4+8
　　输入 3 20　12 15　-9 12　　6 4　8 0
pb=5x^18-8x^15+9x^12-10x^8-9x^4+7x^2-4
　　输入 5 18　-8 15　　9 12 -10 8　-9 4　7 2　-4 0
输出结果：
和：3x^20+5x^18+4x^15-10x^8-3x^4+7x^2+4
差:　3x^20-5x^18+20x^15-18x^12+10x^8+15x^4-7x^2+12

二.实验拓展

1）对于某个 X 的值，求多项式的值。
2) 实现多项式的微分（N 阶导数）
3）实现多项式的定积分
4）实现一元多项式的乘法
5）实现一元多项式的除法(给出商多项式和余多项式)

实验 2　二叉树（6 课时）

一.基本要求

1.编写程序 bitree.cpp 实现 ADT BiTree，要求使用二叉链表存储。

实现基本操作：

InitBiTree(&T);

DestroyBiTree(&T);

PreOrder(T,visit());

InOrder(T,visit());

PostOrder(T,visit());

2.编码实现以下算法：

1）创建二叉树。（以先序扩展序列给出）

2）输出先序、中序和后序序列。

3）计算机二叉树结点数、叶子结点数、高度。

测试数据：先序扩展序列：ABDF##G##E#H##C

输出：先序 ABDFGEHC 中序 FDGBEHAC 后序 FGDHEBCA

结点数：8 叶子结点数：4 高度：4。

二.实验拓展

1）实现层次遍历。

2）查找：查值为 X 的结点、双亲结点、孩子结点、兄弟结点

3）判断：判断一个二叉树是否为二叉排序树、完全二叉树、平衡和二叉树

4）处理：左右子树互换、复制、删除子树、插入子树

实验 3（选） 图（6 课时）

一.基本要求

1.编写程序 graph.cpp 实现 ADT Graph，可以使用邻接矩阵或邻接表存储。
实现基本操作：
InitGraph(&G);
DFS(G,visit()); 深度优先遍历

2.编码实现以下算法：
1）创建图。（以结点对形势给出狐）
2）输出 DFS 遍历序列。
3）插入或删除狐。
4）求某个结点的度。

测试数据：
输入结点数：5
输入元素值：A B C D E
输入边的顶点对：AB AC BD BE CD
DFS 序列(从 A 开始)：ABDCE(假设字母 ascii 码小的存储靠前)
删除边 C D,DFS 输出:ABDEC
插入边:DE,DFS 输出:ABDEC

二.实验拓展

1）实现广度优先遍历 BFS。
2）求图的最小生成树: 普里姆算法和克鲁斯卡尔算法
3）求网中两点间最短路径。

实验 4（选）　栈和队列的应用（6 课时）

一.基本要求

1.编写程序 stack.cpp 实现 ADT Stack，可以使用顺序栈或链栈。

实现基本操作：

InitStack(&S);
DestroyStack(&S);
StackEmpty(S);
Push(&S,e);
Pop(&S,&e);
GetTop(S,&e);
TraverseStack(S);

2.在以下几个实验中选择一个编码实现：

1）括号匹配判断

2）表达式计算：（逆波兰式输入，单字符操作数，如 347×+表示 3+4×7）

二.实验拓展

1）迷宫求解

2）八皇后问题求解

3）求一个集合的幂集

实验 5（选） 哈夫曼编码（6 课时）

一.基本要求

- ✓ **功能模块** 一个完整的huffman编解码系统应该具有以下功能：

  初始化（**I**nitialization）。从终端读入字符集大小n，以及n个字符和n个权值，建立Huffman 树，并将它存入hfmTree 中。

  编码（**E**ncoding）。利用已经建好的Huffman树（如果不在内存，则应从文件hfmTree中读取），对文件ToBeTran中的正文进行编码，然后将结果存入文件CodeFile中。

  解码（**D**ecoding）。利用已经建立好的Huffman树将文件CodeFile中的代码进行解码，结果存入TextFile中。

  打印代码文件（Print）。将文件CodeFile以紧凑的格式显示在终端上，每行 50 个代码。同时将此字符形式的编码文件写入文件CodePrint中。

  打印Huffman树（**T**ree Printing）。以先序和中序两种序列打印huffman树

- ✓ **测试数据：**

  用下表给出的字符集和频度的实际统计数据建立Huffman树，并对以下报文进行编码和译码：“THIS PROGRAM IS MY FAVORITE”。

字符		A	B	C	D	E	F	G	H	I	J	K	L	M
频度	186	64	13	22	32	103	21	15	47	57	1	5	32	20
字符	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
频度	57	63	15	1	48	51	80	23	8	18	1	16	1	

- ✓ **输入输出：**

- ☞ 字符集大小 n，n个字符和 n个权值均从终端读入，初始化后的huffman树存储在hfmTree文件中，待编码文件为ToBeTran,编码结果以文本的方式存储在文件CodeFile中，解码文件存在TextFile中,打印的编码和赫夫曼树分别存储在CodePrint和TreePrint文件中。
- ☞ 用户界面可以设计为“菜单”方式：显示上述功能符号，再加上一个退出功能“Q”，表示退出（quit）。用户键入一个选择功能符，此功能执行完毕后再显示此菜单，直至某次用户选择了 Q为止。

实验 6（选）　查找（6 课时）

一.基本要求

1.编写程序 dstable.cpp 实现 ADT DynamicSearchTable，使用 hash 存储。
实现基本操作：

InitDsTable(&DT);
DestroyDSTable(&DT);
SearchDSTables(DT,key);
InsertDSTable(&DT,e);
TraverseDSTable(DT);

2.编码实现以下算法：

1）创建哈希表。（从标准输入接受关键字，建立哈希表）
2）遍历输出创建后哈希表的内容。
3）查找某个给定的 key 值的位置。

测试数据：

输入数据 1,14,27,29,55,72,10,11,23
hash 函数取 H(k)=k % 13,表长取 m=13
采用线性散列，遍历输出如下：

0	1	2	3	4	5	6	7	8	9	10	11	12
^	1	14	27	29	55	^	72	^	^	10	11	23

查询 29 的位置：4
查询 16 的位置：-1

二.实验拓展

1）计算哈希表的平均查找长度。
2）重建哈希表（当平均查找长度或插入某关键字的冲突次数大于给定阈值）
3）删除某给定关键字（注意墓碑标志）。
4）使用二叉排序树的存储实现查找表。