

实验一 一元多项式实验要求（12 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)

实验二 栈和队列的应用（4 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 $347\times+$ 表示 $3+4\times 7$ ）

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

实验三 二叉树（4 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

实验四 图（4 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。

实验五 查找（4 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。

实验六 数据库 SQL 语言（4 课时）

一.基本要求

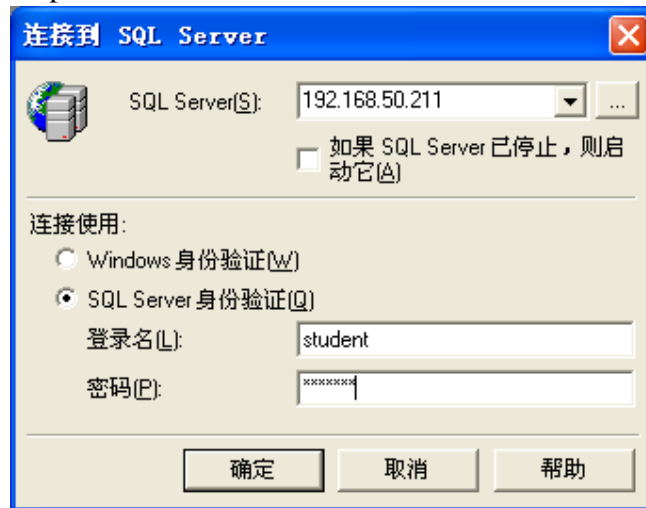
1.熟悉本实验所使用的 DBMS 环境:SQL SERVER 2005, 学会使用 SQL SERVER 客户端工具查询分析器。

2.完成 SQL 语言的基本操作:

- 1) 创建表
- 2) 插入、删除、修改、查询。
- 3) *创建视图

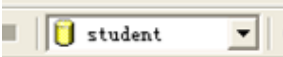
二实验步骤

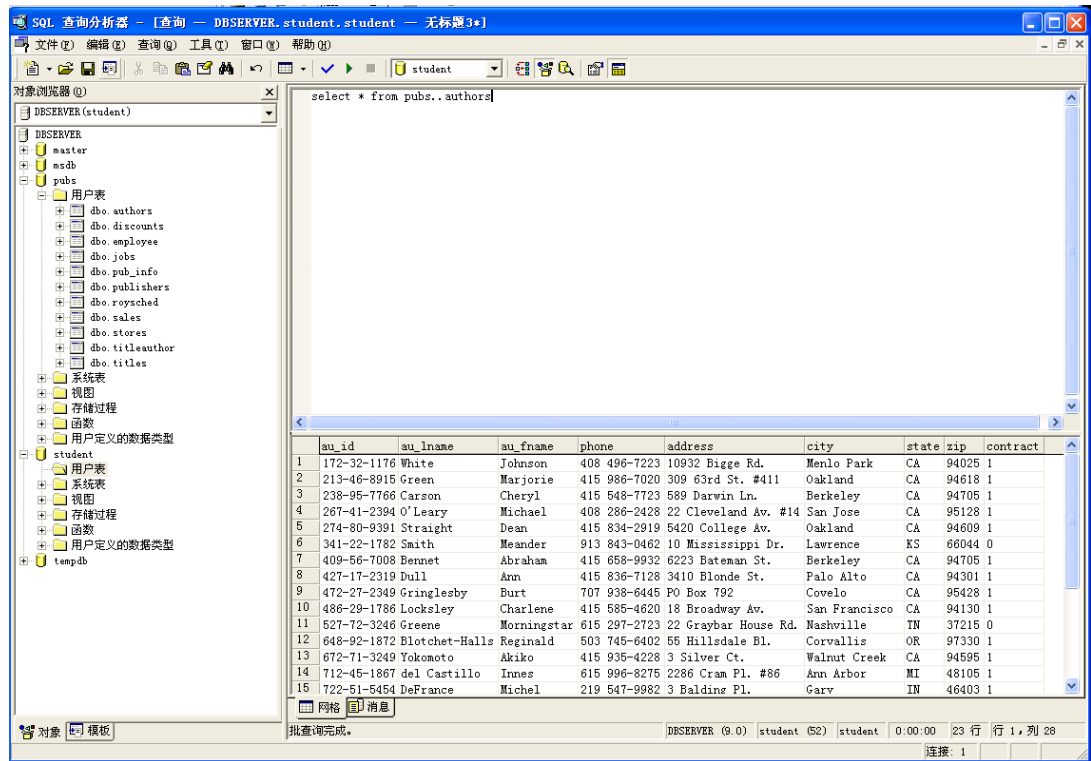
1.下载 SQL Server 客户端工具: [SQLQueryTool](#), 解压后执行 isqlw.exe, 【SQL SERVER(s)】输入 202.38.88.98,1435, 【连接使用】选择“SQL server 身份验证”, 登录名和密码都是 student。此外, 也可以使用实验室机器桌面上的 SQL Server Management Studio Express 登录。



登录后显示如下界面, 则可以执行 sql 语句了。

要注意, pubs 数据库是系统样例数据库, 里面提供了 titles 等实验中用到的示例表, student 用户对其只有查询权, 不可以修改。学生可以在 student 数据库中创建自己的表, 创建表时请按照题目中要求的命名规则来命名。

在操作数据库是要注意当前数据库是 pubs 还是 student, 可以在界面面的 combobox 选择窗口中  选择, 也可以使用命令 use pubs 更改当前数据库。



2. 熟悉 pubs 数据库中各个 table 的定义和关系。<http://192.168.50.203/pubs.pdf> (东区机房外使用 <http://202.38.88.99/pubs.pdf>)

3. 查询操作(在 pubs 数据库中执行):

1) 无条件查询

- 查找 pub 库中 authors 表的全部信息。

2) 简单条件查询

- 查找 titles 表中全部书号及书名。
- 查找 titles 表中价格在 \$15~18 元之间的书的书名。
- 查找 titles 表中书名以 T 开头的书号, 书名。
- 对其他样例表构造各种简单查询条件, 进行查询。

3) 多条件查询

- 查找书名起始字符为 T, 价格小于\$16 元的书名及价格。
- 查找书名起始字符不为 T 的, 价格大于\$16 元的书号, 书名及价格。

4) 用连接操作(或嵌套查询)进行查询

- 对表 titles, publishers 进行查询: 查找出版社的名称以及所出的书名。
- 对表 authors, titleauthor, titles 进行查询: 查找作者的姓、名和所写的书名。

5) 对查询结果排序

- 查找作者的姓、名、电话号码, 并按作者姓、名排列。
- 查找书名和书的价格, 按书价由大到小的次序排列。

6) 使用函数进行查找

- 列出有多少类书。
- 列出书的定价有多少种。
- 列出书价最低的书名和书价。
- 查出书价最高的书名及书价。
- 列出当年销量的总和。

4.表和视图创建及插入

1) **建表：**创建两个表 S**和 T**，并插入下面内容，其中**是本人的学号(下同)。

S**

Title	author	t_no	price	QTY
计算机原理	张一平	S3092	20.80	200
C 语言程序设计	李华	H1298	15.30	300
数据库原理	王家树	D1007	22.70	150
计算机网络	高明	S5690	18.90	230
软件工程	鲁廷璋	S2005	35.00	200

T**

t_no	Page	pub-date
S3092	304	1986
D1007	280	1993
S5006	315	1987
S5690	300	1993
H1298	210	1989

2) 用子查询方式建新表 SS** (包含 title 和 price 两个属性)。

3) *用子查询方式建视图 VS** (包含 title 和 QTY 两个属性)。

5.记录的删除与更新

- 1) 删除书名为“计算机网络”的元组。
- 2) 把书价调整到原来价格的 95%。
- 3) 把书号以 D 开头的那些书的书价减掉 2.00 元。

三.实验拓展

- 1) 创建存储过程。
- 2) 编写简单的数据库应用：可以基于 ASP、PHP 网页，也可以是 CS 结构的应用。

第五章 实验报告

在做完实验以后写实验报告，是整个实验过程的一个重要环节。报告的撰写过程实际是对整个实验的总结与回顾，有助加深对实验的理解，提高对实验的认识。本章介绍如何撰写实验报告。

2.1 如何撰写实验报告

对于每一个实验中所要求解决的问题，都应该有规范详细的报告文档，本实验要求报告的规范如下：

一、 问题描述

1. 实验题目：一般教材中会给出实验题目。
2. 基本要求：实验的基本要求，一般也会在教材中给出。
3. 测试数据：实验中要用到的测试数据，部分实验由教材提供。

二、 需求分析

1. 程序所能达到的基本可能。
2. 输入的形式及输入值范围
3. 输出的形式
4. 测试数据要求

三、 概要设计

1. 所用到的数据结构及其ADT
2. 主程序流程及其模块调用关系
3. 核心模块的算法伪码

四、 详细设计

1. 实现概要设计中的数据结构ADT
2. 实现每个操作的伪码，重点语句加注释
3. 主程序和其他模块的伪码

4. 函数调用关系图

五、 调试分析

1. 设计与调试过程中遇到的问题分析、体会
2. 主要算法的时间和空间复杂度分析

六、 使用说明

简要给出程序的运行和操作步骤。

七、 测试结果

给出实验结果，包括输入和输出。

八、 附录

带注释的源程序。

2.2 实验报告样例

一、 问题描述

1. 实验题目：利用有序链表表示正整数集合，实现集合的交、并和差运算。
2. 基本要求：有用户输入两种整数分别作为两个集合元素，由程序计算它们的交、并和差，并输出结果。

3. 测试数据：

$S1 = \{3, 5, 6, 9, 12, 27, 35\}$ $S2 = \{5, 8, 10, 12, 27, 31, 42, 51, 55, 63\}$

运行结果应为：

$S1 \cup S2 = \{3, 5, 6, 8, 9, 10, 12, 27, 31, 35, 42, 51, 55, 63\}$

$S1 \cap S2 = \{5, 12, 27\}$

$S1 - S2 = \{3, 6, 9, 35\}$

二、 需求分析

1. 本程序用来求出任意两个正整数集合的交、并、差。
2. 程序运行后显示提示信息，提示用户输入两组整数，程序需自动顾虑重复的

整数和负数。

3. 用户输入完毕后，程序自动输出运算结果。

三、 概要设计

为了实现上述功能，应以有序链表表示集合，因此需要有序表和集合两个抽象数据类型。

1. 有序表抽象数据类型定义：

```
ADT OrderedList{
    数据对象:  $D=\{a_i | a_i \in \text{ElemType}, i=1, 2, \dots, n, n \geq 0\}$ 
    数据关系:  $R=\{ \langle a_{i-1}, a_i \rangle | a_{i-1}, a_i \in D, a_{i-1} \leq a_i, i=2, \dots, n \}$ 
    基本操作:
        InitList(&L);    //构造空线性表
        DestroyList(&L); //销毁线性表
        ClearList(&L);   //将L置空
        ListEmpty(L);    //检查L是否为空
        ListLength(L);   //返回L中元素个数
        GetElem(L, i, &e); //返回L中第i个元素赋予e
        LocatePos (L, e); //返回L中e的位置
        InsertElem(&L, e); //在L中插入e
        DeleteElem(&L, i); //删除L中第i个元素
        ListTravers (L);  //依次输出L中元素。
}ADT OrderedList
```

2. 集合的抽象数据类型定义：

```
ADT set{
    数据对象:  $D=\{a_i | a_i \in \text{ElemType}, \text{且各不相同}, i=1, 2, \dots, n, n \geq 0\}$ 
    数据关系:  $R=\phi$ 
    基本操作:
        CreateNullSet (&T);
        DestroySet (&T);
        AddElem(&T, e);
        DelElem(&T, e);
        Union(&T, S1, S2);
        Intersection(&T, S1, S2);
        Difference (&T, S1, S2);
        PrintSet (T);
} ADT set
```

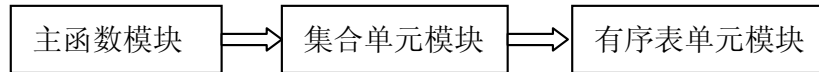
3. 本程序保护模块:

主程序模块

集合单元模块: 实现集合的抽象数据类型

有序表单元模块: 实现有序表抽象数据类型

调用关系:



四、 详细设计

1. 元素类型、结点类型和结点指针类型:

```
typedef int ElemType;
typedef struct NodeType{
    ElemType data;
    NodeType *next;
}NodeType, *LinkType;
```

2. 有序表类型:

```
typedef struct {
    LinkType head,tail;
    int size;
    int curpos;
    LinkType current;
}OrderedList;
//部分基本操作的伪码实现
status InitList(OrderedList &L)
{
    L.head=new Nodetype;
    if(!L.head)retuen false;
    L.head->data=0;
    L.head->next=NULL;
    L.current=L.tail=L.head;
    L.curpos=L.size=0 ;
    return true ;
}
```

```
//... ..(其它略)
```

3. 集合类Set的实现, 利用有序链表来实现:

```
typedef OrderedList OrderedSet;  
status CreateNullSet(OrderedSet &T)  
{  
    if(InitList(T)) return true;  
    else return false;  
}  
// ... ..(其它略)
```

4. 主函数的伪码:

```
void main()  
{  
    cout<<endl<<"请输入 S1:";  
    CreateSet(S1);  
    cout<<endl<<"请输入 S2:";  
    CreateSet(S2);  
    PrintSet(S1);  
    PrintSet(S2);  
    Union(T1,S1,S2);  
    cout<<endl<<"Union:";  
    PrintSet(T1);  
    Intersection(T2,S1,S2);  
    cout<<endl<<"Intersection:";  
    PrintSet(T2);  
    Difference(T3,S1,S2);  
    cout<<endl<<" Difference:";  
    PrintSet(T3);  
    Destroy(T1);  
    Destroy(T2);  
    Destroy(T3);  
    Destroy(S1);  
    Destroy(S2);  
}
```

5. 函数调用关系

(略)

五、 调试分析

1. 程序中将纸张的操作封装在链表的类型中，在集合的类型模块中，只需要引用链表的操作实现相应的集合运算即可，从而使集合模块的调试比较方便。
2. 算法的时空分析：
 - (1) 由于有序表采用带头结点的有序链表，并增设尾指针和表的长度两个标示，各种操作的算法时间复杂度比较合理，LocatePos、GetElem和DestroyList等操作的时间复杂度都是 $O(n)$ ，其中 n 是链表的长度。
 - (2) 构造有序集算法CreateSet读入 n 个元素，需要逐个用LocatePos判断输入元素是不是有重复且确定插入位置后，调用InsertElem插入到有序链表，所以复杂度也是 $O(n)$ 。求并算法Union将两个集合共 $m+n$ 的元素不重复地依次使用InsertElem插入到结果集中，由于插入式按元素值自小到大次序进行，时间复杂度为 $O(m+n)$ 。类似地，求交算法InterSection和求差算法Difference的时间复杂度也是 $O(m+n)$ 。
消耗算法DestroySet和输出PrintSet都是 $O(n)$ 。
所有算法的空间复杂度都是 $O(1)$ 。

六、 使用说明

程序运行后用户根据提示输入集合S1、S2的元素，元素间以空格或回车分隔，输入-1表示输入结束。程序将按照集合内元素从小到大的顺序打印出S1和S2，以及他们的并、交和差。

七、 调试结果

使用两组数据进行了测试：

1.

请输入S1:35 12 9 12 6 6 -12 3 5 9 9 35 27 -1

请输入S2:31 27 5 10 8 -20 55 12 63 42 51 -1

{3, 5, 6, 9, 12, 27, 35}

{5, 8, 10, 12, 27, 31, 42, 51, 55, 63}

Union:

{3, 5, 6, 8, 9, 10, 12, 27, 31, 35, 42, 51, 55, 63}

Intersection:

{5, 12, 27}

Difference:

{3, 6, 9, 35}

2.

请输入S1:43 5 2 46 -1

请输入S2:45 66 -1

{2, 5, 43, 46}

{45, 46}

Union:

{2, 5, 43, 45, 46, 66}

Intersection:

{}

Difference:

{2, 5, 43, 46}

八、 附录

源程序文件清单:

common.h

oderList.h

orderSet.h

set.cpp