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. 所用到得数据结构及其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∪S2={3,5,6,8,9,10,12,27,31,35,42,51,55,63}

   S1∩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={ai|ai∈ElemType,i=1,2...,n,n≥0}
        数据关系：R={<ai-1,ai>|ai-1,ai∈D, ai-1≤ai ,i=2,...,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={ai|ai∈ElemType,且各不相同，i=1,2...,n,n≥0}
        数据关系：R=ϕ
        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
```

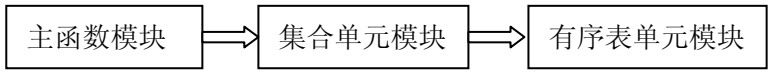
3. 本程序保护模块：

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}

Diffrence:

{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:

{}

Diffrence:

{2, 5, 43, 46}

八、 附录

源程序文件清单：

commmon.h

oderList.h

orderSet.h

set.cpp