数据结构

实验与课程设计指导书

数据结构课程组

广东工业大学计算机学院

2015 年 4 月

目　录

第 1 章　概述

1.1 课程、教材和实验

数据结构是计算机科学的算法理论基础和软件设计的技术基础，主要研究信息的逻辑结构及其基本操作在计算机中的表示和实现。数据结构不仅是计算机专业的核心课程，而且已成为其他理工专业的热门选修课。课程的教学要求之一是训练学生进行复杂程序设计的技能和培养良好程序设计的习惯，其重要程度决不亚于知识传授。因此，在数据结构的整个教学过程中，完成习题作业和上机实习是两个至关重要的环节。

习题的作用在于帮助学生深入理解教材内容，巩固基本概念，达到培养良好程序设计能力和习惯的目的。从认知的程度划分，数据结构的习题通常可分为三类：基础知识题、算法设计题和综合实习题。基础知识题主要是检查对概念知识的识记和理解，一般可作为学生自测题。算法设计题的目的是练习对原理方法的简单应用，多数是要求在某种数据存储结构上实现某一操作，是数据结构的基础训练，构成了课外作业的主体。综合实习题则训练对知识的综合应用和软件开发能力，主要是针对具体应用问题，选择、设计、和实现抽象数据类型（ADT）的可重用模块，并以此为基础开发满足问题要求的小型应用软件，应将其看作软件工程的综合性基础训练的重要一环，给予足够的重视。

本实验指导书为采用自编教材的数据结构课程而编写：吴伟民等. 《数据结构》. 广东工业大学计算机学院，2015.1。

数据结构是实践性很强的课程，光是“听”和“读”是绝对不够的。在努力提高课堂教学的同时，必须大力加强对作业实践环节的要求和管理。国内外先进院校一般都要求修读数据结构的学生每周应不少于 4 个作业机时，而且有一套严格的作业和实习规范和成绩评定标准，形成行之有效的教学质量保证体系。教学经验表明，严格实施作业和实习的规范，对于学生基本程序设计素养的培养和软件工作者工作作风的训练，将能起到显著的促进作用。

数据结构及其算法的教学难点在于它们的抽象性和动态性。虽然在书本教材和课堂授课（板书或投影胶片）中采用图示可以在一定程度上化抽象为直观，但很难有效展现数据结构的瞬间动态特性和算法的作用过程。我们自主研发的“C 程序可视化运行调试集成环境 AnyviewC”，以及基于 AnyviewC 开发的数据结构、C 程序设计、离散数学等课程的“编程作业与实验可视化网络平台”，打破了程序运行调试黑箱。学生可通过 AnyviewC 平台可在线编写和可视化调试自己编写的程序，并接受系统的实时自动测评，极大提高了学生程序设计训练的效率和效果。教师也可从繁重的书面作业批改工作中解脱出来，转到有针对性的现场指导和习题讲评上。借助于互联网，AnyviewC 平台将实验室“全天候”和“跨时空”地拓广到每位学生个人的微机或移动终端上。

1.2　作业和实验安排

根据教学计划，本学期数据结构课程进行首轮 MOOC 教学试验：

1. 课堂理论课 40 学时。
2. 实验室研讨课 16 学时。
3. 课程知识测验。主要题型是选择题和填空式简答题。自行登录课程网页，随课程进度完成测验。

4. 算法设计作业和上机实验（课内、外结合，其中课内指导 10 机时）。在“AnyviewC 编程作业与实验可视化网络平台”上机完成约 60 道必做题，学有余力的同学还可以加做选做题。
5. 抽象数据类型的实现（6 学时设计性实验）。实现一组抽象数据类型，并对所采用的存储结构和相关操作的实现进行讨论。
6. 课程设计（一周综合性实验）。

第 2 章　算法设计实验和上机

2.1　数据结构习题概述

数据结构的习题分为“基础知识题”和“算法设计题”两类。

在课程网站上，“基础知识题”主要供学生进行自测和复习之用，目的是帮助学生深化理解教科书的内容，澄清基本概念、理解和掌握数据结构中分析问题的基本方法和算法要点，为完成算法设计题做准备。

“算法设计题”则侧重于基本程序设计技能的训练，相对于实习题而言，这类编程习题属于偏重于编写功能单一的“小”程序的基础训练，然而，它是进行复杂程序设计的基础，是本课程习题作业的主体和重点。

各章的题量根据教学内容的多少和重要程度而定，几乎对教科书的每一小节都安排了对应的习题。

2.2　算法设计的上机作业要求

1．使用 Anyview **C** 语言和算法书写规范写出书面作业的算法（函数），作为上机前的准备。

需要强调的是“算法的可读性”。初学者总是容易忽视这一点。算法不仅是开发程序的基础，还是一种在程序设计者之间交流解决问题方法的手段。因此，可读性具有头等的重要性。不可读的算法是没有用的，由它得到的程序极容易产生很多隐藏很深的错误，且难以调试正确。一般地说，宁要一个可读性好、逻辑清晰简单、但篇幅较长的算法，也不要篇幅较小但晦涩难懂的算法。算法的正确性力求在设计算法的过程中得到保证，然而一开始做不到这一点也没多大关系，可以逐步做到。

算法设计的正确方法是：首先理解问题，明确给定的条件和要求解决的问题，然后按照自顶向下，逐步求精，分而治之的策略逐一地解决子问题，最后严格按照和使用本章后面提供的算法书写规范和类 **C** 语言完成算法的最后版本。

按照规范书写算法是一个值得高度重视的问题。在基础训练中就贯彻这一规范，不但能够有助于写出“好程序”，避免形成一系列难以纠正且遗害无穷的程序设计坏习惯，而且能够培养软件工作者应有的严谨的科学工作作风。

2．对函数进行静态检查修改，形成准备上机的程序文本。

多数初学者在编好程序后处于以下两种状态之一：一种是对自己的“精心作品”的正确性确信不疑；另一种是认为上机前的任务已经完成，查纠错误是上机的工作。这两种态度是极为有害的。事实上，非训练有素的程序设计者编写的程序长度超过 50 行时，极少不含有除语法错误以外的错误。上机动态调试决不能代替静态检查，否则调试效率将是极低的。

静态检查主要有两种方法，一是用一组测试数据手工执行程序（通常应先分模块检查）；二是通过阅读或给别人讲解自己的程序而深入全面地分析理解程序逻辑，在这个过程中再加入一些注解和断言。如果程序中逻辑概念清楚，后者将比前者有效。

3．在“Anyview C 编程作业与实验可视化网络平台”编辑提交程序，并在系统的自动测试和提示下，调试程序，直到能通过系统的测试。

“Anyview C 编程作业与实验可视化网络平台”提供了程序可视化运行和调试的环境，为进行数据结构教学的师生提供了算法设计作业程序的可视化自动测试环境。可在该集成环境编辑 C 源程序，并对其进行可视化运行、分析和调试。通过设置断点、单步或变换速度的连续运行，可

在多个窗口上动态观察程序执行时的数据变量的物理和逻辑 2D 或 3D 视图，使得程序运行期间本来不可见的程序对数据的处理过程和数据之间的动态抽象关系全部可视化。在提交算法设计作业程序时，系统自动进行可视化测试，评判作业程序的正确性。通过对比“标准结果视图”和“作业结果视图”，作业者可对自己的程序进行直观的分析和排错。关于该作业系统的使用，请参阅系统的帮助文档。

在调试过程中可以不断借助系统的可视 **DEBUG** 的各种功能，提高调试效率。调试中遇到的各种异常现象往往是预料不到的，这时不应“苦思冥想”，而应动手确定疑点，通过修改程序来证实它或绕过它。

4．在调试程序的过程中，做好调试笔记，记录心得体会。

调试正确后，认真整理源程序及其注释，记录带有完整注释的且格式良好的源程序清单和结果。

一道算法设计作业文档包括：

（1）上机前编写并经过静态检查的程序文本；

（2）调试笔记；

（3）最后程序文本，及通过时间。

第 3 章　抽象数据类型的实现

3.1 实验概要

实验项目名称：　抽象数据类型的实现
实验项目性质：　设计性实验
所属课程名称：　数据结构
实验计划学时：　6

3.2 实验目的

对某组具体的抽象数据类型，运用课程所学的知识和方法，设计合理的数据结构，并在此基础上实现该抽象数据类型的全部基本操作。通过本设计性实验，检验所学知识和能力，发现学习中存在的问题。 进而达到熟练地运用本课程中的基础知识及技术的目的。

3.3 预习与参考

1．确定要实现的抽象数据类型，并对基本操作做适当的选取和增加；
2．选择存储结构，并写出相应的类型定义；
3．设计各基本操作的实现算法，并表达为函数形式；
4．设计测试方案，编写主函数；
5．将上述 4 步的结果写成预习报告。

3.4 实验要求和设计指标

以教材中讨论的各种抽象数据类型为对象，利用 C 语言的数据类型表示和实现其中某个抽象数据类型。可选的抽象数据类型如下表所列：

编号	抽象数据类型	基本难度	存储结构
1	栈和队列	1.0	顺序　和　链接
2	线性表	1.0	顺序　和　链接
3	哈希表	1.1	任选
4	二叉树	1.2	任选
5	堆	1.2	任选
6	二叉排序树	1.2	任选
7	平衡二叉树	1.3	任选
8	树	1.2	任选
9	并查集	1.2	任选
10	B 树	1.4	任选
11	有向图	1.3	任选
12	无向图	1.3	任选

13	有向带权图	1.3	任选

注：如果基本操作数量较多，可选择实现其中一个基本操作子集。

实验要求如下：

1. 首先了解设计的任务，然后根据自己的基础和能力从中选择一题。一般来说，选择题目应以在规定的时间内能完成，并能得到应有的锻炼为原则。**若学生对教材以外的相关题目较感兴趣，希望选作实验的题目时，应征得指导教师的认可，并写出明确的抽象数据类型定义及说明。**

2. 实验前要作好充分准备，包括：理解实验要求，掌握辅助工具的使用，了解该抽象数据类型的定义及意义，以及其基本操作的算法并设计合理的存储结构。

3. 实验时严肃认真，要严格按照要求独立进行设计，不能随意更改。注意观察并记录各种错误现象，纠正错误，使程序满足预定的要求，实验记录应作为实验报告的一部分。

4. 实验后要及时总结，写出实验报告，并附所打印的问题解答、程序清单，所输入的数据及相应的运行结果。

3.5 实验仪器设备和材料

计算机学院实验中心。

编程环境：AnyviewCL 可视化编程环境、TC++、C++Builder、VC++或 Java。

3.6 调试及结果测试

调试内容应包括：调试过程中遇到的问题是如何解决的以及对实验的讨论与分析；基本操作的时间复杂度和空间复杂度的分析和改进设想。列出对每一个基本操作的测试结果，包括输入和输出，测试数据应完整和严格。

3.7 考核形式

考核形式以实验过程和实验报告相结合的方式进行。在实验完成后，应当场运行和答辩，由指导教师验收，只有在验收合格后才能算实验部分的结束。实验报告作为整个设计性实验评分的书面依据。设计性实验的成绩评定以选定题目的难易度、完成情况和实验报告为依据综合评分。从总体来说，所实现的抽象数据类型应该全部符合要求，类型定义，各基本操作的算法以及存储结构清晰；各模快测试运行正确；程序的结构合理；设计报告符合规范。**鼓励实现多种抽象数据类型。**

对于每个题目，选择的同学数量不得超过 5 个/每小班，但对于选多种抽象数据类型实现的同学，多选部分题目不计入。请学委于 17 周实验课时提交最后确定的选题名单【使用 Excel 表汇总哪道题目有哪些同学选，必须注明学号】。

3.8 实验报告要求

实验结束后要写出实验报告，以作为整个设计性实验评分的书面依据和存档材料。实验报告是反映学生实验效果的最主要的依据，也是学生正确地表达问题、综合问题和发现问题的能力的基本培养手段，因而是非常重要的内容。本设计性实验的报告要包括以下几项内容：

（1）设计任务、要求及所用软件环境或工具；

（2）抽象数据类型定义以及各基本操作的简要描述；

（3）所选择的存储结构描述及在此存储结构上各基本操作的实现；

（4）程序清单（计算机打印），输入的数据及各基本操作的测试结果；

（5）实验总结和体会。请勿堆砌各种煽情语句，描述自己在实验和课程设计过程中如何如何辛苦，此类文字并非专业报告必要章节内容。

实验报告以规定格式的电子文档书写、打印并装订，排版及图表要清楚、工整。

3.9 思考题

对设计性实验进行总结和讨论，包括本实验的优、缺点，数据存储结构的特点，与其它存储结构之间的比较等。通过总结，可以对抽象数据类型有更全面、深入的认识，这是设计性实验不可缺少的重要内容。这部分内容应作为实验报告中的一个组成部分。

3.10 示例

1．题目

采用字符类型为元素类型和无头结点单链表为存储结构，实现抽象数据类型 List。

ADT List{

数据对象：D＝{ a_i | a_i∈ElemSet, i=1,2,...,n, n≥0 }

数据关系：R1＝{ <a_{i-1}, a_i>|a_{i-1}, a_i∈D, i=2,...,n }

基本操作：

SetEmpty(**&**L)

操作结果：构造一个空的线性表 L。

Destroy(**&**L)

初始条件：线性表 L 已存在。

操作结果：销毁线性表 L。

Length(L)

初始条件：线性表 L 已存在。

操作结果：返回 L 中元素个数。

Get(L, i, **&**e)

初始条件：线性表 L 已存在，1≤i≤LengthList(L)。

操作结果：用 e 返回 L 中第 i 个元素的值。

Locate(L, e, compare())

初始条件：线性表 L 已存在，compare()是元素判定函数。

操作结果：返回 L 中第 1 个与 e 满足关系 compare()的元素的位序。
若这样的元素不存在，则返回值为 0。

Insert(**&**L, i, e)

初始条件：线性表 L 已存在，1≤i≤LengthList(L)+1。

操作结果：在 L 的第 i 个元素之前插入新的元素 e，L 的长度加 1。

Delete(**&**L, i, **&**e)

初始条件：线性表 L 已存在且非空，1≤i≤LengthList(L)。

操作结果：删除 L 的第 i 个元素，并用 e 返回其值，L 的长度减 1。

Display(L)

初始条件：线性表 L 已存在。

操作结果：依次输出 L 的每个元素。

} ADT List

2．存储结构定义

公用头文件 DS0.h:

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
#include <values.h>

#define TRUE   1
#define FALSE  0
#define OK     1
#define ERROR  0
#define IBFEASIBLE  -1
#define OVERFLOW    -2

#define MAXLEN  20
#define MAXSIZE 20

typedef int Status;
typedef char ElemType; /* 元素类型为字符类型*/
```

（1） 顺序存储结构

```
#define LIST_INIT_SIZE 20 /*线性表存储空间的初始分配量*/
#define LISTINCREMENT  10  /*线性表存储空间的分配增量*/
typedef struct{
    ElemType *elem; /*存储空间基址*/
    int   length;    /*当前长度*/
    int   listsize; /*当前分配的存储容量*/
} SqList;
```

（2） 无头结点单链表存储结构

```
typedef struct LNode {
   ElemType data;
   struct LNode *next;
} LNode, *LList;     /* 不带头结点单链表类型*/
```

（3） 带头结点单链表存储结构

```
typedef struct LNode {  /* 结点类型 */
   ElemType data;
   struct LNode *next;
} LNode, *Link, *Position;

typedef struct LinkList { /* 链表类型 */
   Link head,tail;  /* 分别指向线性链表中的头结点和最后一个结点 */
   int len;         /* 指示线性链表中数据元素的个数 */
} LinkList;
```

3．算法设计

（1） 顺序存储结构

```
Status SetEmpty(SqList &L) { /*构造一个空的顺序线性表 */
```

```
L.elem=(ElemType*)malloc(LIST_INIT_SIZE*sizeof(ElemType));
  if(!L.elem)
    return OVERFLOW; /* 存储分配失败 */
  L.length=0; /* 空表长度为0 */
  L.listsize=LIST_INIT_SIZE; /* 初始存储容量 */
  return OK;
 }
Status Destroy (SqList &L) { /*销毁顺序线性表L */
  free(L.elem);
  L.elem=NULL;
  L.length=0;
  L.listsize=0;
  return OK;
}
int Length(SqList L) { /* 求表长*/
  return L.length;
}
Status Get(SqList &L, int i, ElemType &e) { /* 获取第i元素 */
  if(i<1||i>L.length)
    return ERROR;
  e=*(L.elem+i-1);
  return OK;
}
int Locate(SqList L, ElemType x) { /* 确定x在表中的位序 */
  ElemType *p;
  int i=1; /* i的初值为第1个元素的位序 */
  p=L.elem; /* p的初值为第1个元素的存储位置 */
  while(i<=L.length && *p++!=x)
    ++i;
  if(i<=L.length)
    return i;
  else
    return 0;
}

Status Insert(SqList &L, int i, ElemType e) {
  /* 操作结果：在L中第i个位置之前插入新的数据元素e，L的长度加1 */
  ElemType *newbase,*q,*p;
  if(i<1||i>L.length+1) /* i值不合法 */
    return ERROR;
  if(L.length>= L.listsize) { /* 当前存储空间已满,增加分配 */
    newbase=(ElemType *) realloc(L.elem,
           (L.listsize+LISTINCREMENT)*sizeof(ElemType));
```

```
    if(!newbase) return OVERFLOW; /* 存储分配失败 */
    L.elem=newbase; /* 新基址 */
    L.listsize+=LISTINCREMENT; /* 增加存储容量 */
  }
  q=L.elem+i-1; /* q 为插入位置 */
  for(p=L.elem+L.length-1; p>=q; --p)
    *(p+1)=*p;  /* 插入位置及之后的元素右移 */
  *q=e; /* 插入 e */
  ++L.length; /* 表长增 1 */
  return OK;
}
Status Delete(SqList &L, int i, ElemType &e) {
  /* 初始条件：顺序线性表 L 已存在，1≤i≤ListLength(L) */
  /* 操作结果：删除 L 的第 i 个数据元素，并用 e 返回其值，L 的长度减 1 */
  ElemType *p,*q;
  if(i<1||i> L.length) /* i 值不合法 */
    return ERROR;
  p= L.elem+i-1; /* p 为被删除元素的位置 */
  e=*p; /* 被删除元素的值赋给 e */
  q= L.elem+L.length-1; /* 表尾元素的位置 */
  for(++p; p<=q; ++p) /* 被删除元素之后的元素左移 */
    *(p-1)=*p;
  L.length--; /* 表长减 1 */
  return OK;
}
Status Display(SqList L) { /* 依次显示表中元素 */
  ElemType *p;
  int i;
  p=L.elem;
  printf("( ");
  for(i=1; i<=L.length; i++)
    printf("%c",*p++);
  printf(" )\n");
  return OK;
}
```

（2） 无头结点单链表

```
void SetEmpty(LList &L) { /* 置无头结点的空单链表*/
  L=NULL;
}
Status Destroy (LList &L) { /* 销毁链表*/
  LList q=L;
  while(L) {
    L=L->next;
    free(q);
```

```
    q=L;
  }
  return OK;
}
int Length(LList L) { /* 求表长*/
  int n=0;
  while(L!=NULL) {
    n++;
    L=L->next;
  }
  return n;
}
Status Get(LList L, int i, ElemType &e) { /* 获取第 i 元素 */
  int j=1;
  while (j<i && L!=NULL) {
    L=L->next;
    j++;
  }
  if(L!=NULL) { e=L->data;  return OK; }
  else return ERROR; /* 位置参数 i 不正确 */
}
int Locate(LList L, ElemType x) { /* 确定 x 在表中的位序 */
  int n=1;
  while (L!=NULL && L->data!=x) {
    L=L->next;
    n++;
  }
  if (L==NULL) return 0;
  else return n;
}
Status Insert(LList &L, int i, ElemType e) { /* 插入第 i 元素*/
  int j=1;
  LList s,q;
  s=(LList)malloc(sizeof(LNode));
  s->data=e;
  q=L;
  if (i==1) { s->next=q;  L=s;  return OK;}
  else {
    while(j<i-1 && q->next!=NULL) {
      q=q->next;
      j++;
    }
    if (j==i-1) {
      s->next=q->next;
```

```
        q->next=s;
        return OK;
      }
      else return ERROR;  /* 位置参数 i 不正确 */
    }
}
Status Delete(LList &L, int i, ElemType &e) { /* 删除第 i 元素*/
    int j=1;
    LList q=L,t;
    if (i==1) {
      e=q->data;
      L=q->next;
      free(q);
      return OK;
    }
    else {
      while (j<i-1 && q->next!=NULL) {
        q=q->next;
        j++;
      }
      if (q->next!=NULL && j==i-1) {
        t=q->next;
        q->next=t->next;
        e=t->data;
        free(t);
        return OK;
      }
      else return ERROR;  /* 位置参数 i 不正确*/
    }
}
void Display(LList L) { /* 依次显示表中元素 */
    printf("单链表显示: ");
    if (L==NULL)
      printf("链表为空!");
    else if (L->next==NULL)
      printf("%c\n", L->data);
    else {
      while(L->next!=NULL) {
        printf("%c->", L->data);
        L=L->next;
      }
      printf("%c", L->data);
    }
    printf("\n");
```

```
}
```

（3） 带头结点单链表

```
Status SetEmpty(LinkList &L) { /* 置带头结点的空单链表*/
  Link p;
  p=(Link)malloc(sizeof(LNode)); /* 生成头结点 */
  if(p) {
    p->next=NULL;
    L.head=L.tail=p;
    L.len=0;
    return OK;
  }
  else
    return ERROR;
}
Status Destroy(LinkList &L) { /* 销毁线性链表 L，L 不再存在 */
  Link p,q;
  if(L.head!=L.tail) { /* 不是空表 */
    p=q= L.head->next;
    L.head->next=NULL;
    while(p!=L.tail) {
      p=q->next;
      free(q);
      q=p;
    }
    free(q);
  }
  free(L.head);
  L.head=L.tail=NULL;
  L.len=0;
  return OK;
}
int Length(LinkList L) { /* 返回线性链表 L 中元素个数 */
  return L.len;
}
Status Get(LinkList L, int i, ElemType &e) { /* 获取第 i 元素 */
  /* i=0 为头结点 */
  Link p;
  int j;
  if(i<1||i>L.len)
    return ERROR;
  else {
    p=L.head;
    for(j=1;j<=i;j++)
      p=p->next;
```

```
      e=p->data;
      return OK;
   }
}
int Locate(LinkList L, ElemType x) { /* 确定 x 在表中的位序 */
   int i=0;
   Link p=L.head;
   do {
      p=p->next;
      i++;
   } while(p && p->data!=x); /* 没到表尾且没找到满足关系的元素 */
   if (!p)
      return 0;
   else
      return i;
}
Status Insert(LinkList &L, int i, ElemType e) { /* 插入第 i 元
素*/
   int j=0;
   Link s,q;
   s=(Link)malloc(sizeof(LNode));
   s->data=e;
   q=L.head;
   while(j<i-1 && q->next!=NULL) {
      q=q->next;
      j++;
   }
   if (j==i-1) {
      s->next=q->next;
      q->next=s;
      if (L.tail==q) L.tail=s;
      L.len++;
      return OK;
   }
   else return ERROR;  /* 位置参数 i 不正确 */
}
Status Delete(LinkList &L, int i, ElemType &e) {
   /* 删除第 i 元素*/
   int j=0;
   Link q=L.head,t;
   while (j<i-1 && q->next!=NULL) {
      q=q->next;
      j++;
   }
```

```
    if (q->next!=NULL && j==i-1) {
      t=q->next;
      q->next=t->next;
      e=t->data;
      if(L.tail==t) L.tail=q;
      free(t);
      L.len--;
      return OK;
    }
    else return ERROR;  /* 位置参数 i 不正确*/
  }
  void Display(LinkList L) { /* 依次显示表中元素 */
    Link p;
    printf("单链表显示: ");
    if (L.head==L.tail)
      printf("链表为空!");
    else {
      p=L.head->next;
      while(p->next!=NULL) {
        printf("%c->", p->data);
        p=p->next;
      }
      printf("%c", p->data);
    }
    printf("\n");
  }
```

4. 测试(需要测试结果截图)

(1) 顺序存储结构

```
SqList head;
void main() { /* 主函数*/
  char e,c;
  int i,n,select,x1,x2,x3,x4,m,g;
  SetEmpty(head);
  n=random(8); /* 随机产生表长 */
  for (i=1; i<=n; i++) { /* 将数据插入到顺序表中 */
    c='A'+random(26);
    Insert(head,i,c);
  }
  do {
    Display(head);
    printf("select 1 求长度 Length()\n");
    printf("select 2 取结点 Get()\n");
    printf("select 3 求值查找 Locate()\n");
```

```
        printf("select 4 删除结点 Delete()\n");
        printf("input your select: ");
        scanf("%d",&select);
        switch(select) {
          case 1: x1=Length(head);
                  printf("顺序表的长度:%d ",x1);
                  break;
          case 2: printf("请输入要取的结点序号: ");
                  scanf("%d",&m);
                  if(Get(head,m,x2)) printf("%c\n",x2);
                  else printf("错误\n");
                  break;
          case 3: printf("请输入要查找的数据元素: ");
                  scanf("\n%c",&e);
                  x3=Locate(head,e);
                  printf("%d\n",x3);
                  break;
          case 4: printf("请输入要删除的元素序号: ");
                  scanf("%d",&g);
                  if(Delete(head,g,x4)) printf("%c\n",x4);
                  else printf("错误\n");
                  break;
        }
      } while (select>0 && select <5);
    }
```

（2） 无头结点单链表

```
    LList head;
    void main() { /* 主函数*/
      char e,c;
      int i,n,select,x1,x2,x3,x4,m,g;
      SetEmpty(head);
      n=random(8); /* 随机产生表长 */
      for (i=1; i<=n; i++) { /* 将数据插入到顺序表中 */
        c='A'+random(26);
        Insert(head,i,c);
      }
      do {
        Display(head);
        printf("select 1 求长度 Length()\n");
        printf("select 2 取结点 Get()\n");
        printf("select 3 求值查找 Locate()\n");
        printf("select 4 删除结点 Delete()\n");
        printf("input your select: ");
        scanf("%d",&select);
```

```
      switch(select) {
        case 1: x1=Length(head);
                printf("顺序表的长度:%d ",x1);
                break;
        case 2: printf("请输入要取的结点序号: ");
                scanf("%d",&m);
                if(Get(head,m,x2)) printf("%c\n",x2);
                else printf("错误\n");
                break;
        case 3: printf("请输入要查找的数据元素: ");
                scanf("\n%c",&e);
                x3=Locate(head,e);
                printf("%d\n",x3);
                break;
        case 4: printf("请输入要删除的元素序号: ");
                scanf("%d",&g);
                if(Delete(head,g,x4)) printf("%c\n",x4);
                else printf("错误\n");
                break;
      }
    } while (select>0 && select <5);
}
```

（3） 带头结点单链表

```
LinkList head;
void main() { /* 主函数*/
  char e,c;
  int i,n,select,x1,x2,x3,x4,m,g;
  SetEmpty(head);
  n=random(8); /* 随机产生表长 */
  for (i=1; i<=n; i++) { /* 将数据插入到顺序表中 */
    c='A'+random(26);
    Insert(head,i,c);
  }
  do {
    Display(head);
    printf("select 1 求长度 Length()\n");
    printf("select 2 取结点 Get()\n");
    printf("select 3 求值查找 Locate()\n");
    printf("select 4 删除结点 Delete()\n");
    printf("input your select: ");
    scanf("%d",&select);
    switch(select) {
      case 1: x1=Length(head);
              printf("顺序表的长度:%d ",x1);
```

```
                break;
            case 2: printf("请输入要取的结点序号: ");
                scanf("%d",&m);
                if(Get(head,m,x2)) printf("%c\n",x2);
                else printf("错误\n");
                break;
            case 3: printf("请输入要查找的数据元素: ");
                scanf("\n%c",&e);
                x3=Locate(head,e);
                printf("%d\n",x3);
                break;
            case 4: printf("请输入要删除的元素序号: ");
                scanf("%d",&g);
                if(Delete(head,g,x4)) printf("%c\n",x4);
                else printf("错误\n");
                break;
        }
    } while (select>0 && select<5);
}
```

5. 三种存储结构的比较（没有则可不写）

	存储结构	顺序映象	无头结点单链表	带头结点单链表
基本操作时间复杂度	SetEmpty(&L)	O(1)	O(1)	O(1)
	Destroy(&L)	O(1)	O(n)	O(n)
	Length(L)	O(1)	O(n)	O(1)
	Get(L, i, &e)	O(1)	O(n)	O(n)
	Locate(L,e,compare())	O(n)	O(n)	O(n)
	Insert(&L, i, e)	O(n)	O(n)	O(n)
	Delete(&L, i, &e)	O(n)	O(n)	O(n)
	Display(L)	O(n)	O(n)	O(n)
优缺点分析	优 点	可以随机存取	插入删除时不需要移动元素	1．对空表不需要额外进行判断处理； 2．求表长度方便
	缺 点	插入删除时需要移动元素	1．对空表需要额外进行判断处理； 2．求表长不方便	不能随机存取

6. 思考与小结

（1）无头结点单链表的插入和删除操作的实现，要区分该表是否为空，并编写相应的代码。

（2）在算法设计时，要注意判断有关参数值的合法性。

（3）三种存储结构的主函数相同。设计主函数及测试运行时，要考虑测试的完备性。

7. 预习报告和实验报告

（1）预习报告：包括 1-4 步的初稿。

（2）实验报告：在预习报告的基础上，增加在实验中，对算法修改核调试的收获体会，以及思考和小结的内容。

第 4 章　课程设计

4.1　课程设计概述

课程设计是对学生的一种全面综合训练，是与课堂听讲、自学和练习相辅相成的必不可少的一个教学环节。通常，课程设计题目中的问题比平时的习题复杂得多，也更接近实际。实习着眼于原理与应用的结合点，使学生学会如何把书上学到的知识用于解决实际问题，培养软件工作所需要的动手能力；另一方面，能使书上的知识变“活”，起到深化理解和灵活掌握教学内容的目的。平时的练习较偏重于如何编写功能单一的“小”算法，而课程设计是软件设计的综合训练，包括问题分析，总体结构设计，用户界面设计，程序设计基本技能和技巧，多人合作，以至一整套软件工作规范的训练和科学作风的培养。此外，还有很重要的一点是：机器是比任何教师都严厉的检查者。

为了达到上述目的，本学期的数据结构课程设计列出了十余个题目供选做。训练重点在于基本的数据结构，而不强调面面俱到。一个课程设计题目可能涉及多个知识点。根据题目的基本要求，每个题目的题号之后标有难度系数。如果选做了题目的扩充内容，则视完成情况酌情增加难度系数。

每个课程设计题目采取了统一的格式，由问题描述、基本要求、测试数据、实现提示和选做内容等五个部分组成。

问题描述旨在为学生建立问题提出的背景环境，指明问题“是什么”。

基本要求则对问题进一步求精，划出问题的边界，指出具体的参量或前提条件，并规定该题的最低限度要求。

测试数据部分旨在为检查学生上机作业提供方便。在完成课程设计题目时，应自己设计完整和严格的测试方案，当数据输入量较大时，提倡以文件形式向程序提供输入数据。

在实现提示部分，对实现中的难点及其解法思路等问题作了简要提示。

选做部分向那些尚有余力的学生提出了更严峻的挑战，同时也能开拓其他学生的思路，在完成基本要求时就力求避免就事论事的不良思想方法，尽可能寻求具有普遍意义的解法，使得程序结构合理，容易修改扩充。

不难发现，这里与传统的做法不同，题目设计得非常详细。会不会限制学生的想象力，影响创造力的培养呢？回答是:软件发展的一条历史经验就是要限制程序设计者在某些方面的创造性，从而使其创造能力集中地用到特别需要创造性的环节之上。课程设计题目本身就给出了问题说明和问题分解求精的范例，使学生在无形中学会模仿，它起到把学生的思路引上正轨的作用，避免坏结构程序和坏习惯，同时也传授了系统划分方法和程序设计的一些具体技术，保证实现预定的训练意图，使某些难点和重点不会被绕过去，而且也便于教学检查。题目的设计策略是：一方面使其难度和工作量都较大，另一方面给学生提供的辅助和可以模仿的成分也较多。当然还应指出的是，提示的实现方法未必是最好的，学生不应拘泥于此，而应努力开发更好的方法和结构。

经验表明，如果某题的难度略高于自己过去所对付过的最难题目的难度，则选择此题能够带来最大的收益。切忌过分追求难题。较大的题目，或是其他题目加上某些选做款项适合于多人合作。

4.2　课程设计时间和内容

数据结构课程设计通常在数据结构课程后段启动，在课程结束后，有一周时间作最后实现、

调试和撰写设计报告。

题目 1　长整数四则运算（难度系数：1.0）

[问题描述]

一个实现任意长的整数进行加法运算的演示程序。

[基本要求]

利用双向循环链表实现长整数的存储，每个结点含一个整型变量。任何整型变量的范围是$-(2^{15}-1)$～$(2^{15}-1)$。输入和输出形式：按中国对于长整数的表示习惯，每四位一组，组间用逗号隔开。

[测试数据]

(1) 0；0；应输出“0”。

(2) -2345,6789；-7654,3211；应输出“-1,0000,0000”。

(3) -9999,9999；1,0000,0000,0000；应输出“9999,0000,0001”。

(4) 1,0001,0001；-1,0001,0001；应输出“0”。

(5) 1,0001,0001；-1,0001,0000；应输出“1”。

(6) -9999,9999,9999；-9999,9999,9999；应输出“-1,9999,9999,9998”。

(7) 1,0000,9999,9999；1；应输出“1,0001,0000,0000”。

[实现提示]

(1) 每个结点中可以存放的最大整数为 $2^{15}-1=32767$，才能保证两数相加不会溢出。但若这样存，即相当于按 32768 进制数存，在十进制数与 32768 进制数之间的转换十分不方便。故可以在每个结点中仅存十进制数的 4 位，即不超过 9999 的非负整数，整个链表视为万进制数。

(2) 可以利用头结点数据域的符号代表长整数的符号。用其绝对值表示元素结点数目。相加过程中不要破坏两个操作数链表。两操作数的头指针存于指针数组中是简化程序结构的一种方法。不能给长整数位数规定上限。

[选做内容]

（1）实现长整数的四则运算；

（2）实现长整数的乘方和阶乘运算；

（3）整型量范围是$-(2^n-1)$～(2^n-1)，其中，n 是由程序读入的参量。输入数据的分组方法可以另行规定。

题目 2　一元稀疏多项式计算器（难度系数：1.0）

[问题描述]

设计一个一元稀疏多项式简单计算器。

[基本要求]

一元稀疏多项式简单计算器的基本功能是：

（1）输入并建立多项式；

（2）输出多项式，输出形式为整数序列：$n,c_1,e_1,c_2,e_2,...,c_n,e_n$，其中 n 是多项式的项数，$c_i$ 和 e_i 分别是第 i 项的系数和指数，序列按指数降序排列；

（3）多项式 a 和 b 相加，建立多项式 a+b;

（4）多项式 a 和 b 相减，建立多项式 a-b;

[测试数据]

(1) $(2x+5x^8-3.1x^{11}) + (7-5x^8+11x^9) = (-3.1x^{11}+11x^9+2x+7)$

(2) $(6x^{-3}-x+4.4x^2-1.2x^9) - (-6x^{-3}+5.4x^2-x^2+7.8x^{15}) = (-7.8x^{15}-1.2x^9+12x^{-3}-x)$

(3) $(1+x+x^2+x^3+x^4+x^5) + (-x^3-x^4) = (1+ x+ x^2+ x^5)$

(4) $(x+x^3) + (-x-x^3) = 0$

(5) $(x+x^{100}) + (x^{100}+x^{200}) = (x+2x^{100}+x^{200})$

(6) $(x+x^2+x^3) + 0 = x+x^2+x^3$

(7) 互换上述测试数据中的前后两个多项式

[实现提示]

用带表头结点的单链表存储多项式，多项式的项数存放在头结点。

[选做内容]

（1）计算多项式在 x 处的值。

（2）求多项式 a 的导函数 a'。

（3）多项式 a 和 b 相乘，建立多项式 a×b。

（4）多项式的输出形式为类数学表达式。例如，多项式 $-3x^8+6x^3-18$ 的输出形式为 -3x^8+6x^3-18，$x^{15}+(-8)x^7-14$ 的输出形式为 x^15-8x^7-14。注意，系数值为 1 的非零次项的输出形式中略去系数 1，如项 $1x^8$ 的输出形式为 x^8，项 $-1x^3$ 的输出形式为 $-x^3$。

（5）计算器的仿真界面。

题目 3　池塘夜降彩色雨（难度系数：1.2）

[问题描述]

设计一个程序，演示美丽的“池塘夜雨”景色：色彩缤纷的雨点飘飘洒洒地从天而降，滴滴入水有声，溅起圈圈微澜。

[基本要求]

（1）雨点的空中出现位置、降落过程的可见程度、入水位置、颜色、最大水圈等等，都是随机确定的；

（2）多个雨点按照各自的随机参数和存在状态，同时演示在屏幕上。

[测试数据]

适当调整控制雨点密度、最大水圈和状态变化的时间间隔等参数。

[实现提示]

（1）每个雨点的存在周期可分为三个阶段：从天而降、入水有声和圈圈微澜，需要一个记录存储其相关参数、当前状态和下一状态的更新时刻；

（2）在图形状态编程。雨点下降的可见程度应是断断续续、依稀可见；圈圈水波应是由里至外逐渐扩大和消失。

（3）每个雨点发生时，生成其记录，并预置下一个雨点的发生时间；

（4）用一个适当的结构管理当前存在的雨点，使系统能利用它按时更新每个雨点的状态，一旦有雨点的水圈全部消失，就从结构中删去。

[选做内容]

（1）增加“电闪雷鸣”景象。

（2）增加风的效果，展现“风雨飘摇”的情景。

（3）增加雨点密度的变化：时而“和风细雨”， 时而“暴风骤雨”。

（4）将“池塘”改为“荷塘”，雨点滴在荷叶上的效果是溅起四散的水珠，响声也不同。

题目 4　银行业务模拟（难度系数：1.3）

[问题描述]

客户业务分为两种。第一种是申请从银行得到一笔资金，即取款或借款。第二种是向银行投入一笔资金，即存款或还款。银行有两个服务窗口，相应地有两个队列。客户到达银行后先排第一个队。处理每个客户业务时，如果属于第一种，且申请额超出银行现存资金总额而得不到满足，

则立刻排入第二个队等候，直至满足时才离开银行；否则业务处理完后立刻离开银行。每接待完一个第二种业务的客户，则顺序检查和处理（如果可能）第二个队列中的客户，对能满足的申请者予以满足，不能满足者重新排到第二个队列的队尾。注意，在此检查过程中，一旦银行资金总额少于或等于刚才第一个队列中最后一个客户(第二种业务)被接待之前的数额，或者本次已将第二个队列检查或处理了一遍，就停止检查(因为此时已不可能还有能满足者)转而继续接待第一个队列的客户。任何时刻都只开一个窗口。假设检查不需要时间。营业时间结束时所有客户立即离开银行。

写一个上述银行业务的事件驱动模拟系统，通过模拟方法求出客户在银行内逗留的平均时间。

[基本要求]

利用动态存储结构实现模拟，即利用 **C** 语言的动态分配函数 malloc 和 free。

[测试数据]

一天营业开始时银行拥有的款额为 10000(元)，营业时间为 600(分钟)。其他模拟参量自定，注意测定两种极端的情况：一是两个到达事件之间的间隔时间很短，而客户的交易时间很长，另一个恰好相反，设置两个到达事件的间隔时间很长，而客户的交易时间很短。

[实现提示]

事件有两类：到达银行和离开银行。初始时银行现存资金总额为 total。开始营业后的第一个事件是客户到达，营业时间从 0 到 closetime。到达事件发生时随机地设置此客户的交易时间和距下一到达事件之间的时间间隔。每个客户要办理的款额也是随机确定的，用负值和正值分别表示第一类和第二类业务。变量 total、closetime 以及上述两个随机量的上下界均交互地从终端读入，作为模拟参数。

两个队列和一个事件表均要用动态存储结构实现。注意弄清应该在什么条件下设置离开事件，以及第二个队列用怎样的存储结构实现时可以获得较高的效率。注意：事件表是按时间顺序有序的。

[选做内容]

自己实现动态数据类型。例如对于客户结点，定义 pool 为

```
CustNode    pool[MAX];
// 结构类型 CustNode 含四个域: arrtime,durtime,amount,next
```

或者定义四个同样长的，以上述域名为名字的数组。初始时，将所有分量的 next 域链接起来，形成一个静态链栈，设置一个栈顶元素下标指示量 top，top=0 表示栈空。动态存储分配函数可以取名为 myMalloc，其作用是出栈，将栈顶元素的下标返回。若返回的值为 0，则表示无空间可分配。归还函数可取名为 myFree，其作用是把该分量入栈。用 **FORTRAN** 和 **BASIC** 等语言实现时只能如此地自行组织。

题目 5　航空客运订票系统（难度系数：1.2）

[问题描述]

航空客运订票的业务活动包括：查询航线、客票预订和办理退票等。试设计一个航空客运订票系统，以使上述业务可以借助计算机来完成。

[基本要求]

(1) 每条航线所涉及的信息有：终点站名、航班号、飞机号、飞行周日(星期几)、乘员定额、余票量、已订票的客户名单(包括姓名、订票量、舱位等级 1，2 或 3)以及等候替补的客户名单(包括姓名、所需票量)；

(2) 作为示意系统，全部数据可以只放在内存中；

(3) 系统能实现的操作和功能如下：

① 查询航线：根据旅客提出的终点站名输出下列信息：航班号、飞机号、星期几飞行，最近一天航班的日期和余票额；

② 承办订票业务：根据客户提出的要求(航班号、订票数额)查询该航班票额情况，若尚有余票，则为客户办理订票手续，输出座位号；若已满员或余票额少于订票额，则需重新询问客户要求。若需要，可登记排队候补；

③ 承办退票业务：根据客户提供的情况(日期、航班)，为客户办理退票手续，然后查询该航班是否有人排队候补，首先询问排在第一的客户，若所退票额能满足他的要求，则为他办理订票手续，否则依次询问其它排队候补的客户。

[测试数据]

由读者指定。

[实现提示]

两个客户名单可分别由线性表和队列实现。为查找方便，已订票客户的线性表应按客户姓名有序，并且，为插入和删除方便，应以链表作存储结构。由于预约人数无法预计，队列也应以链表作存储结构。整个系统需汇总各条航线的情况登录在一张线性表上，由于航线基本不变，可采用顺序存储结构，并按航班有序或按终点站名有序。每条航线是这张表上的一个记录，包含上述八个域、其中乘员名单域为指向乘员名单链表的头指针，等候替补的客户名单域为分别指向队头和队尾的指针。

[选做内容]

当客户订票要求不能满足时，系统可向客户提供到达同一目的地的其它航线情况。

读者还可充分发挥自己的想象力，增加你的系统的功能和其它服务项目。

题目 6　电梯模拟（难度系数：1.4）

[问题描述]

设计一个电梯模拟系统。这是一个离散的模拟程序，因为电梯系统是乘客和电梯等“活动体”构成的集合，虽然它们彼此交互作用，但它们的行为是基本独立的。在离散的模拟中，以模拟时钟决定每个活动体的动作发生的时刻和顺序，系统在某个模拟瞬间处理有待完成的各种事情，然后把模拟时钟推进到某个动作预定要发生的下一个时刻。

[基本要求]

（1）模拟某校五层教学楼的电梯系统。该楼有一个自动电梯，能在每层停留。五个楼层由下至上依次称为地下层、第一层、第二层、第三层和第四层，其中第一层是大楼的进出层，即是电梯的“本垒层”，电梯“空闲”时，将来到该层候命。

（2）乘客可随机地进出于任何层。对每个人来说，他有一个能容忍的最长等待时间，一旦等候电梯时间过长，他将放弃。

（3）模拟时钟从 0 开始，时间单位为 0.1 秒。人和电梯的各种动作均要耗费一定的时间单位（简记为 t)，比如：

有人进出时，电梯每隔 40t 测试一次，若无人进出，则关门；

关门和开门各需要 20t；

每个人进出电梯均需要 25t；

如果电梯在某层静止时间超过 300t，则驶回 1 层候命。

（4）按时序显示系统状态的变化过程：发生的全部人和电梯的动作序列。

[测试数据]

模拟时钟 Time 的初值为 0，终值可在 500~10000 范围内逐步增加。

[实现提示]

（1）楼层由下至上依次编号为 0,1,2,3,4。每层有要求 Up（上）和 Down（下）的两个按钮，

对应十个变量 CallUp[0..4]和 CallDown[0..4]。电梯内 5 个目标层按钮对应变量 CallCar[0..4]。有人按下某个按钮时，相应的变量就置为 1，一旦要求满足后，电梯就把该变量清为 0。

（2）电梯处于三种状态之一：GoingUp（上行），GoingDown（下行）和 Idle（停候）。如果电梯处于 Idle 状态且不在 1 层，则关门并驶回 1 层。在 1 层停候时，电梯是闭门候命。一旦收到往另一层的命令，就转入 GoingUp 或 GoingDown 状态，执行相应的操作。

（3）用变量 Time 表示模拟时钟，初值为 0，时间单位(t)为 0.1 秒。其它重要的变量有：

Floor -- 电梯的当前位置（楼层）；

D1 -- 值为 0，除非人们正在进入和离开电梯；

D2 -- 值为 0，如果电梯已经在某层停候 300t 以上；

D3 -- 值为 0，除非电梯门正开着又无人进出电梯；

State -- 电梯的当前状态（GoingUp, GoingDown, Idle）。

系统初始时，Floor=1, D1=D2=D3=0, State=Idle。

（4）每个人从进入系统到离开称为该人在系统中的存在周期。在此周期内，他有六种可能发生的动作：

M1.[进入系统，为下一人的出现作准备] 产生以下数值：

InFloor -- 该人进入哪层楼；

OutFloor -- 他要去哪层楼；

GiveupTime -- 他能容忍的等候时间；

InterTime -- 下一人出现的时间间隔，据此系统预置下一人进入系统的时刻。

M2.[按电钮并等候] 此时应对以下不同情况作不同的处理：

1）Floor=InFloor 且电梯的下一个活动是 E6（电梯在本层，但正在关门）；

2）Floor=InFloor 且 D3≠0（电梯在本层，正有人进出）；

3）其它情况，可能 D2=0 或电梯处于活动 E1（在 1 层停候）。

M3.[进入排队] 在等候队列 Queue[InFloor]末尾插入该人，并预置在 GiveupTime 个 t 之后他若仍在队列中将实施动作 M4。

M4.[放弃] 如果 Floor≠InFloor 或 D1=0，则从 Queue[InFloor]和系统删除该人。如果 Floor=InFloor 且 D1≠0，他就继续等候（他知道马上就可进入电梯）。

M5.[进入电梯] 从 Queue[InFloor]删除该人，并把他插入到 Elevator（电梯）栈中。置 CallCar[OutFloor]为 1。

M6.[离去] 从 Elevator 和系统删除该人。

（5）电梯的活动有九种：

E1.[在 1 层停候] 若有人按下一个按钮，则调用 Controler 将电梯转入活动 E3 或 E6。

E2.[要改变状态？] 如果电梯处于 GoingUp（或 GoingDown）状态，但该方向的楼层却无人等待，则要看反方向楼层是否有人等候，而决定置 State 为 GoingDown（或 GoingUp）还是 Idle。

E3.[开门] 置 D1 和 D2 为非 0 值，预置 300 个 t 后启动活动 E9 和 76 个 t 后启动 E5，然后预置 20 个 t 后转到 E4。

E4.[让人出入] 如果 Elevator 不空且有人的 OutFloor=Floor，则按进入的倒序每隔 25 个 t 让这类人立即转到他们的动作 M6。Elevator 中不再有要离开的人时，如果 Queue[Floor]不空，则以 25 个 t 的速度让他们依次转到 M5。Queue[Floor]空时，置 D1 为 0，D3≠0，而且等候某个其它活动的到来。

E5.[关门] 每隔 40 个 t 检查 D1，直到是 D1=0（若 D1≠0，则仍有人出入）。置 D3 为 0 并预置电梯再 20 个 t 后启动活动 E6（再关门期间，若有人到来，则如 M2 所述，门再次打开）。

E6.[准备移动] 置 CallCar[Floor]为 0，而且若 State≠GoingDown 则置 CallUp[Floor]为 0，若 State≠GoingUp 则置 CallDown[Floor]为 0。调用 Controler 函数。

如果 State=Idle，则即使已经执行了 Controler，亦转到 E1。否则如果 D2≠0，则取消电梯活动 E9。最后，如果 State=GoingUp，则预置 15 个 t 后（电梯加速）转到 E7；如果 State=GoingDown，则预置 15 个 t 后（电梯加速）转到 E8。

E7.[上升一层] 置 Floor 加 1 并等候 51 个 t。如果现在 CallCar[Floor]=1 或 CallUp[Floor]=1，或者如果（（Floor=1 或 CallDown[Floor]=1）且 CallUp[j]=CallDown[j]= CallCar[j]=0 对于所有 j>Floor），则预置 14 个 t 后（减速）转到 E2。否则重复 E7。

E8.[下降一层] 除了方向相反之外，与 E7 类似，但那里的 51 和 14 个 t 此时分别改为 61 和 23 个 t（电梯下降比上升慢）。

E9.[置不活动指示器] 置 D2 为 0 并调用 Controler 函数（E9 是由 E3 预置的，但几乎总是被 E6 取消了）。

（6）当电梯须对下一个方向作出判定时，便在若干临界时刻调用 Controler 函数。该函数有以下要点：

C1.[需要判断？] 若 State≠Idle，则返回。

C2.[应该开门？] 如果电梯处于 E1 且 CallUp[1],CallDown[1]或 CallCar[1]非 0，则预置 20 个 t 后启动 E3，并返回。

C3.[有按钮按下？] 找最小的 j≠Floor，使得 CallUp[j],CallDown[j]或 CallCar[j]非 0，并转到 C4。但如果不存在这样的 j，那么，如果 Controler 正为 E6 所调用，则置 j 为 1，否则返回。

C4.[置 State] 如果 Floor>j，则置 State 为 GoingDown；如果 Floor<j，则置 State 为 GoingUp。

C5.[电梯静止？] 如果电梯处于 E1 而且 j≠1，则预置 20 个 t 后启动 E6。返回。

（7）由上可见，关键是设计合适的数据结构，按时序管理系统中所有乘客和电梯的动作。

[选做内容]

（1）增加电梯数量，模拟多梯系统。

（2）某高校的一座 30 层住宅楼有三部自动电梯，每梯最多载客 15 人。大楼每层八户，每户平均 3.5 人，每天早晨平均每户有 3 人必须在 7 时之前离开大楼去上班或上学。模拟该电梯系统，并分析分别在一梯、二梯和三梯运行情况下，下楼高峰期间各层的住户应提前多少时间候梯下楼？研究多梯运行最佳策略。

题目 7　哈希表设计（难度系数：1.0）

[问题描述]

针对某个集体（比如你所在的班级）中的“人名”设计一个哈希表，使得平均查找长度不超过 R，完成相应的建表和查表程序。

[基本要求]

假设人名为中国人姓名的汉语拼音形式。待填入哈希表的人名共有 30 个，取平均查找长度的上限为 2。哈希函数用除留余数法构造，用伪随机探测再散列法处理冲突。

[测试数据]

取读者周围较熟悉的 30 个人的姓名。

[实现提示]

如果随机函数自行构造，则应首先调整好随机函数，使其分布均匀。人名的长度均不超过 19 个字符(最长的人名如：庄双双（Zhang Shuangshuang）。字符的取码方法可直接利用 C 语言中的 toascii 函数，并可对过长的人名先作折叠处理。

[选做内容]

(1) 从教科书上介绍的几种哈希函数构造方法中选出适用者并设计几个不同的哈希函数，比较它们的地址冲突率(可以用更大的名字集合作试验)。

(2) 研究这 30 个人名的特点，努力找一个哈希函数，使得对于不同的拼音名一定不发生地址冲突。

(3) 在哈希函数确定的前提下尝试各种不同处理冲突的方法，考查平均查找长度的变化和造好的哈希表中关键字的聚簇性。

题目 8　最小生成树问题（难度系数：1.1）

[问题描述]

若要在 n 个城市之间建设通讯网络，只需要架设 n-1 条线路即可。如何以最低的经济代价建设这个通讯网，是一个网的最小生成树问题。

[基本要求]

（1）利用克鲁斯卡尔算法求网的最小生成树。

（2）实现并查集。以此表示构造生成树过程中的连通分量。

（3）以文本形式输出生成树中各条边以及他们的权值。

[测试数据]

参见本题集中的习题。

[实现提示]

通讯线路一旦建立，必然是双向的。因此，构造最小生成树的网一定是无向网。设图的顶点数不超过 30 个，并为简单起见，网中边的权值设成小于 100 的整数，可利用 C 语言提供的随机数函数产生。

图的存储结构的选取应和所作操作向适应。为了便于选择权值最小的边，此题的存储结构既不选用邻接矩阵的数组表示法，也不选用邻接表，而是以存储边(带权)的数组表示图。

[选做内容]

利用堆排序实现选择权值最小的边。

题目 9　表达式类型的实现（难度系数：1.2）

[问题描述]

一个表达式和一棵二叉树之间，存在着自然的对应关系。写一个程序，实现基于二叉树表示的算术表达式 Expression 的操作。

[基本要求]

假设算术表达式 Expression 内可以含有变量（a~z）、常量（0~9）和二元运算符（+,-,*,/,^（乘幂））。实现以下操作：

（1）ReadExpr(E) -- 以字符序列的形式输入语法正确的前缀表示式并构造表达式 E。

（2）WriteExpr(E) -- 用带括弧的中缀表示式输出表达式 E。

（3）Assign(V,c) -- 实现对变量 V 的赋值（V = c），变量的初值为 0。

（4）Value(E) — 对算术表达式 E 求值。

(5) CompoundExpr(P, E1, E2) — 构造一个新的复合表达式 (E1)P(E2)。

[测试数据]

（1） 分别输入 `0;  a;  -91;  +a*bc;  +*15^x2*8x;  +++*3^x3*2^x2x6` 并输出。

（2） 每当输入一个表达式后，对其中的变量赋值，然后对表达式求值。

[实现提示]

（1）在读入表达式的字符序列的同时，完成运算符和运算数（整数）的识别处理，以及相应的运算。

（2）在识别出运算数的同时，要将其字符形式转换成整数形式。

（3） 用后根遍历的次序对表达式求值。

（4） 用中缀表示输出表达式 E 时，适当添加括号，以正确反映运算的优先次序。

[选做内容]

（1）增加求偏导数运算 Diff(E,V) — 求表达式 E 对变量 V 的导数。

（2）在表达式中添加三角函数等初等函数的操作。

（3）增加常数合并操作 MergeConst(E) — 合并表达式 E 中所有常数运算。例如，对表达式 E=(2+3-a)*(b+3*4)进行合并常数的操作后，求得 E=(5-a)*(b+12)。

题目 10 内部排序算法比较（难度系数：1.1）

[问题描述]

在教科书中，各种内部排序算法的时间复杂度分析结果只给出了算法执行时间的阶，或大概执行时间。试通过随机数据比较各算法的关键字比较次数和关键字移动次数，以取得直观感受。

[基本要求]

(1) 对以下 6 种常用的内部排序算法进行比较：起泡排序，直接插入排序，简单选择排序，快速排序，希尔排序，堆排序。

(2) 待排序表的表长不小于 100；其中的数据要用伪随机数产生程序产生；至少要用 5 组不同的输入数据作比较；比较的指标为有关键字参加的比较次数和关键字的移动次数(关键字交换计为 3 次移动)。

(3) 最后要对结果作出简单分析，包括对各组数据得出结果波动大小的解释。

[测试数据]

由随机数产生器生成。

[实现提示]

主要工作是设法在已知算法中的适当位置插入对关键字的比较次数和移动次数的计数操作。程序还可以考虑几组数据的典型性，如，正序、逆序和不同程度的乱序。注意采用分块调试的方法。

[选做内容]

(1) 增加折半折入排序，二路插入排序，归并排序，基数排序等。

(2) 对不同的输入表长作试验，观察检查两个指标相对于表长的变化关系。还可以对稳定性作验证。

题目 11 多关键字排序（难度系数：1.0）

[问题描述]

多关键字的排序有其一定的实用范围。例如：在进行高考分数处理时，除了需对总分进行排序外，不同的专业对单科分数的要求不同，因此尚需在总分相同的情况下，按单科的分数排出考生录取的次序。

[基本要求]

（1） 假设待排序的记录数不超过 1000，表中记录的关键字数不超过 5，各个关键字的范围均为 0 至 100。按用户给定的进行排序的关键字的优先关系，输出排序结果。

（2） 约定按 LSD 法进行多关键字的排序。在对各个关键字进行排序时采用两种策略：其一是利用稳定的内部排序法，其二是利用“分配”和“收集”的方法。并综合比较这两种策略。

[测试数据]

由随机数产生器生成。

[实现提示]

用 5 至 8 组数据比较不同排序策略所需时间。

由于是按 LSD 方法进行排序，则对每个关键字均可进行整个序列的排序，但在利用通常的内部排序方法进行排序时，必须选用稳定的排序方法。借助“分配”和“收集”策略进行的排序，如同一趟“基数排序”，由于关键字的取值范围为 0 至 100，则分配时将得到 101 个链表。

[选做内容]

增添按 MSD 策略进行排序，并和上述两种排序策略进行综合比较。

题目 12　平衡二叉树操作的演示（难度系数：1.3）

[问题描述]

利用平衡二叉树实现一个动态查找表。

[基本要求]

实现动态查找表的三种基本功能：查找、插入和删除。

[测试数据]

由读者自行设定。

[实现提示]

(1) 初始，平衡二叉树为空树，操作界面给出查找、插入和删除三种操作供选择。每种操作均要提示输入关键字。每次插入或删除一个结点后，应更新平衡二叉树的显示。

(2) 平衡二叉树的显示可采用如 6.3 题要求的凹入表形式，也可以采用图形界面画出树形。

(3) 教科书已给出查找和插入算法，本题重点在于对删除算法的设计和实现。假设要删除关键字为 x 的结点。如果 x 不在叶子结点上，则用它左子树中的最大值或右子树中的最小值取代 x。如此反复取代，直到删除动作传递到某个叶子结点。删除叶子结点时，若需要进行平衡变换，可采用插入的平衡变换的反变换（如，左子树变矮对应于右子树长高）。

[选做内容]

(1) 合并两棵平衡二叉树。

(2) 把一棵平衡二叉树分裂为两棵平衡二叉树，使得在一棵树中的所有关键字都小于或等于 x，另一棵树中的任一关键字都大于 x。

题目 13　教学计划编制问题（难度系数：1.2）

[问题描述]

大学的每个专业都要制定教学计划。假设任何专业都有固定的学习年限，每学年含两学期，每学期的时间长度和学分上限值均相等。每个专业开设的课程都是确定的，而且课程在开设时间的安排必须满足先修关系。每门课程有哪些先修课程是确定的，可以有任意多门，也可以没有。每门课恰好占一个学期。试在这样的前提下设计一个教学计划编制程序。

[基本要求]

(1) 输入参数包括：学期总数，一学期的学分上限，每门课的课程号(固定占 3 位的字母数字串)、学分和直接先修课的课程号。

(2) 允许用户指定下列两种编排策略之一：一是使学生在各学期中的学习负担尽量均匀；二是使课程尽可能地集中在前几个学期中。

(3) 若根据给定的条件问题无解，则报告适当的信息；否则将教学计划输出到用户指定的文件中。计划的表格格式自行设计。

[测试数据]

学期总数：6；学分上限：10；该专业共开设 12 门课，课程号从 C_{01} 到 C_{12}，学分顺序为 2,3,4,3,2,3,4,4,7,5,2,3。先修关系见教科书图 7.26。

[实现提示]

可设学期总数不超过 12，课程总数不超过 100。如果输入的先修课程号不在该专业开设的课程序列中，则作为错误处理。应建立内部课程序号与课程号之间的对应关系。

[选做内容]

产生多种(例如 5 种)不同的方案，并使方案之间的差异尽可能地大。

题目 14　校园导游咨询（难度系数：1.1）

[问题描述]

设计一个校园导游程序，为来访的客人提供各种信息查询服务。

[基本要求]

（1）设计你的学校的校园平面图，所含景点不少于 10 个。以图中顶点表示校内各景点，存放景点名称、代号、简介等信息；以边表示路径，存放路径长度等相关信息。

（2）为来访客人提供图中任意景点相关信息的查询。

（3）为来访客人提供图中任意景点的问路查询，即查询任意两个景点之间的一条最短的简单路径。

[测试数据]

由读者根据实际情况指定。

[实现提示]

一般情况下，校园的道路是双向通行的，可设校园平面图是一个无向网。顶点和边均含有相关信息。

[选做内容]

（1）求校园图的关节点。

（2）提供图中任意景点问路查询，即求任意两个景点之间的所有路径。

（3）提供校园图中多个景点的最佳访问路线查询，即求途经这多个景点的最佳（短）路径。

（4）校园导游图的景点和道路的修改扩充功能。

（5）扩充道路信息，如道路类别（车道、人行道等）、沿途景色等级，以至可按客人所需分别查询人行路径或车行路径或观景路径等。

（6）扩充每个景点的邻接景点的方向等信息，使得路径查询结果能提供详尽的导向信息。

（7）实现校园导游图的仿真界面。

题目 15　全国交通咨询模拟（难度系数：1.2）

[问题描述]

出于不同目的的旅客对交通工具有不同的要求。例如，因公出差的旅客希望在旅途中的时间尽可能短，出门旅游的游客则期望旅费尽可能省，而老年旅客则要求中转次数最少。编制一个全国城市间的交通咨询程序，为旅客提供两种或三种最优决策的交通咨询。

[基本要求]

（1） 提供对城市信息进行编辑（如：添加或删除）的功能。

（2） 城市之间有两种交通工具：火车和飞机。提供对列车时刻表和飞机航班进行编辑（增设或删除）的功能。

（3） 提供两种最优决策：最快到达或最省钱到达。全程只考虑一种交通工具。

（4） 旅途中耗费的总时间应该包括中转站的等候时间。

（5） 咨询以用户和计算机的对话方式进行。由用户输入起始站、终点站、最优决策原则和交通工具，输出信息为：最快需要多长时间才能到达或者最少需要多少旅费才能到达，并详细说明依次于何时乘坐哪一趟列车或哪一次班机到何地。

[测试数据]

参考全国交通图，自行设计列车时刻表和飞机航班。

[实现提示]

（1） 对全国城市交通图和列车时刻表及飞机航班表的编辑，应该提供文件形式输入和键盘输入两种方式。飞机航班表的信息应包括：起始站的出发时间、终点站的到达时间和票价；列车时刻表则需根据交通图给出各个路段的详细信息，例如：对从北京到上海的火车，需给出北京至天津、天津至徐州及徐州至上海各段的出发时间、到达时间及票价等信息。

（2） 以邻接表作交通图的存储结构，表示边的结点内除含有邻接点的信息外，还应包括交通工具、路程中消耗的时间和花费以及出发和到达的时间等多项属性。

[选做内容]

增加旅途中转次数最少的最优决策。

题目 16　图书管理（难度系数：1.3）

[问题描述]

图书管理基本业务活动包括：对一本书的采编入库、清除库存、借阅和归还等等。试设计一个图书管理系统，将上述业务活动借助于计算机系统完成。

[基本要求]

（1）每种书的登记内容至少包括书号、书名、著者、现存量和总库存量等五项。

（2）作为演示系统，不必使用文件，全部数据可以都在内存存放。但是由于上述四项基本业务活动都是通过书号(即关键字)进行的，所以要用 B 树(2-3 树)对书号建立索引，以获得高效率。

（3）系统应实现的操作及其功能定义如下：

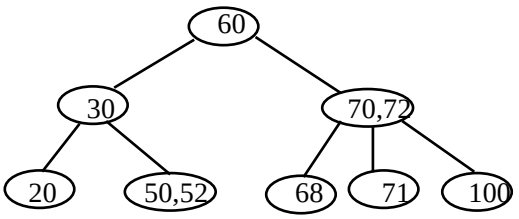
①采编入库：新购入一种书，经分类和确定书号之后登记到图书账目中去。如果这种书在帐中已有，则只将总库存量增加。

②清除库存：某种书已无保留价值，将它从图书账目中注销。

③借阅：如果一种书的现存量大于零，则借出一本，登记借阅者的图书证号和归还期限。

④归还：注销对借阅者的登记，改变该书的现存量。

⑤显示：以凹入表的形式显示 B 树。这个操作是为了调试和维护的目的而设置的。下列 B 树的打印格式如下所示：




```
60
    30
        20
        50,52
    70,72
        68
        71
        100
```

[测试数据]

入库书号：35, 16, 18, 70, 5, 50, 22, 60, 13, 17, 12 , 45, 25, 42, 15, 90, 30, 7

然后清除：45, 90, 50, 22, 42

其余数据自行设计。由空树开始，每插入删除一个关键字后就显示 B 树的状态。

[实现提示]

(1) 2-3 树的查找算法是基础，入库和清除操作都要调用。难点在于删除关键字的算法，因而只要算法对 2-3 树适用就可以了，暂时不必追求高阶 B 树也适用的删除算法。

(2) 每种书的记录可以用动(或静)态链式结构。

借阅登记信息可以链接在相应的那种书的记录之后。

[选做内容]

(1) 将一次会话过程(即程序一次运行)中的全部人机对话记入一个日志文件“log”中去。

(2) 增加列出某著者全部著作名的操作。思考如何提高这一操作的效率。

(3) 增加列出某种书状态的操作。状态信息除了包括这种书记录的全部信息外还包括最早到期(包括已逾期)的借阅者证号，日期可用整数实现，以求简化。

(4) 增加预约借书功能。

可以在上述题目中选择，或者从《实验指导书》第二篇中选择题目。

对于每个题目，选择的同学数量不得超过 5 个/每小班。请学委于 17 周实验课时提交最后确定的选题名单【使用 Excel 表汇总哪道题目有哪些同学选，必须注明学号】。

对于希望自定题目的同学，请于 16 周前与老师联系，描述希望做的题目内容，经老师同意后，可以不在上述题目中选择【鼓励】。

4.3 课程设计步骤

一种常见的错误观念是，调试程序全凭运气。花两个小时的上机时间只找出一个错误，甚至一无所获的情况是常见的。其原因在于，很多人只认识到找错误，而没有认识到努力预先避免错误的重要性，也不知道应该如何努力。实际上，结构不好、思路和概念不清的程序可能是根本无法调试正确的。严格按照实习步骤规范进行实习不但能有效地避免上述种种问题，更重要的是有利于培养软件工作者不可缺少的科学工作方法和作风。

（一）问题分析与系统概要设计

充分地分析和理解问题，明确题目要求做什么（而不是怎样做）和限制条件是什么。按照以数据结构为中心的原则划分模块，定义主程序模块和各抽象数据类型。在这个过程中，要综合考虑系统功能，使得系统结构清晰、合理、简单和易于调试。作为概要设计的结果，应写出每个抽象数据类型的定义（包括数据结构的描述和每个基本操作的规格说明），画出模块之间的调用关系图，以及设计测试方案。这是一个不断调整的过程。基本操作的规格说明应尽可能明确具体。

（二）详细设计与编码

本步骤主要是确定数据结构的存储表示和实现抽象数据类型。详细设计就是要对数据结构和基本操作的规格说明作进一步的求精，写出数据存储结构的类型定义，按照算法书写规范用类 **C** 语言写出过程或函数形式的算法框架。在求精的过程中，应尽量避免陷入语言细节，不必过早表述辅助数据结构和局部变量。

编码是把详细设计的结果进一步求精为程序设计语言程序。程序的每行不要超过 60 各字符。每个过程（函数）体，即不计首部和规格说明部分，一般不要超过 40 行，最长不得超过 60 行，否则应该分割成较小的过程（函数）。要控制 **if** 语句连续嵌套的深度。其他要求参见第一篇的算法书写规范。如何编写程序才能较快地完成调试是特别要注意的问题。

（三）上机准备和静态检查

上机准备包括以下几方面：

（1）高级语言文本（体现与编译程序用户手册）的扩充和限制。

（2）如果用 **C++**语言，要特别注意平时惯用的类 **C** 语言与 C++语言之间的细微差别。

（3）熟悉机器的操作系统和语言集成环境的用户手册，尤其是最常用的命令操作，以便顺利进行上机的基本活动。

（4）掌握调试工具，考虑调试方案，设计测试数据并手工得出正确结果。“磨刀不误砍柴工”。计算机各专业的学生应该能够熟练运用高级语言的程序调试器 **DEBUG** 调试程序。**AnyviewC** 是我们自主开发并推荐使用的一个程序可视调试器。

上机动态调试前应先进行静态检查，以提高调试效率。

（四）上机调试程序

上机时要带上语言教材或手册。调试最好分模块进行，自底向上，即先调试低层过程或函数。必要时可以另写一个调用驱动程序。这种表面上麻烦的工作实际上可以大大降低调试所面临的复杂性，提高调试工作效率。

在调试过程中可以借助 **DEBUG** 的各种功能，提高调试效率。调试中遇到的各种异常现象往往是预料不到的，这时不应“苦思冥想”，而应动手设法确定疑点，通过修改程序来证实它或绕过它。调试正确后，认真整理源程序及其注释，记录带有完整注释的且格式良好的源程序清单和结果。

（五）整理课程设计报告

课程设计报告必须采用学校统一规定的报告封面，在首页给出**题目、班级、姓名、学号和完成日期，**并包括以下七个内容：

1. 需求分析

以无歧义的陈述说明程序设计的任务，强调的是程序要做什么？明确规定：

⑴ 输入的形式和输入值的范围；

⑵ 输出的形式；

⑶ 程序所能达到的功能；

⑷ 测试数据，包括正确的输入及其输出结果和含有错误的输入及其输出结果。

2. 概要设计

说明本程序中用到的所有数据类型的定义、主程序的流程以及各程序模块之间的调用关系。

3. 详细设计

实现概要设计中定义的所有数据类型，对每个操作只需要写出伪码算法；对主程序和其他模块也都需要写出伪码算法(伪码算法需要达到的详细程度为：按照伪码算法可以在计算机键盘直接输入高级程序设计语言程序)；画出函数和过程的调用关系图。

4. 调试分析

内容包括：

⑴ 调试过程中遇到的问题是如何解决的以及对设计与实现的回顾讨论和分析；

⑵ 算法的时空分析（包括基本操作和其他算法的时间复杂度和空间复杂度的分析）和改进设想；

⑶ 经验和体会等（可选）。请勿堆砌各种煽情语句，描述自己在实验和课程设计过程中如何如何辛苦，此类文字并非专业报告必要章节内容。

【反面示例】

1. 虽然看上去很简单，但是实现起来总会遇到各种问题，什么抛出异常，什么语法错误，什么空指针等等，弄得我几度想放弃，但是实现出自己想要的功能后还是蛮开心的。当然我这次课程设计弄了太多的窗口，弄得自己都有点昏头转向的，所以归好类是前提，其次就是写方法时最好及时测试其功能，免得等会调试不出错误。还有就是要有耐心，只要功夫深铁杵磨成针。

2. 通过此次课程设计，使我更加扎实的掌握了有关稀疏矩阵方面的知识，在设计过程中遇到了一些问题，也暴露出了前期我在这方面的知识欠缺和经验不足。实践出真知，通过亲自动手制作，使我们掌握的知识不再是纸上谈兵。 在课程设计过程中，不断发现错误，不断改正，不断领悟，不断获取。这次课程设计终于顺利完成了，在设计中遇到了很多问题，最后在同学的指导下，终于游逆而解。在今后社会的发展和学习实践过程中，一定要不懈努力，不能遇到问题就想到要退缩，一定要不厌其烦的发现问题所在，然后一一进行解决，只有这样，才能成功的做成想做的事，才能在今后的道路上劈荆斩棘，而不会知难而退，如果那样将永远不可能收获成功，收获喜悦，也永远不可能让自己变的众所周知，要想成功只有自己不懈努力！

5. 用户使用说明

说明如何使用你编写的程序，详细列出每一步的操作步骤；

6. 测试结果

列出你的测试结果，包括输入和输出。这里的测试数据应该完整和严格，最好多于需求分析中所列。

7. 附录

带注释的源程序。如果提交源程序软盘，可以只列出程序文件名清单。

在各课程设计单元中都提供了实习报告实例。必须注意的是，实习报告的各种文档资料要在程序开发的过程中逐渐充实形成，而不能最后补写（当然可以也应该最后用实验报告纸誊清或打印）。

4.4 课程设计考核形式和评分标准

1．在程序运行界面突出显示设计者的班级、学号和姓名。课程设计结束前，进行程序运行检查和答辩。

2．提交课程设计报告（打印）和光盘。各班统一制作一张光盘，每人一个目录，每题一个子目录，内含：源程序文件、可执行程序文件、测试用例和课程设计报告 WORD 文档。

3．成绩采用五级评分制：优，良，中，及格和不及格。

根据题目的**难度**、选做内容、完成的**程序**和**报告**的质量评定成绩。

只完成基本内容者，成绩至高为“良”。

鼓励完成选做内容，可获得加分到“优”。

鼓励采用 Windows 环境编程。

如果有下列情况，则视情节严重程度，成绩下降若干档次，直至不及格：

- 盘中文件含有病毒或者内容不能正确读出；
- 抄袭、复制别人程序或文档；
- 未能按时提交报告和光盘。