数据结构实验指导书

张德海 李劲 朱艳萍 金鑫 刘宇 秦江龙

本书是以《数据结构》2015 版课程教学大纲和实验教学大纲为指导，分别从实验（包含 ABC 三个级别的实验）的九个方面（实验目的、问题描述、基本要求、CDIO 项目要求、实验内容、实现提示、参考程序、教学环节组织以及思考题）来组织内容。其中实验目的强调了每个实验要掌握的内容；问题描述是对实验题目作进一步的解释；CDIO 项目要求给出了按 CDIO 标准进行实现的基本要求；实验内容是对基本实验和扩展实验的问题进行描述和标明要达到的层次，并且对实现要求进行了规划；实现提示对整个实验的结构进行了组织和指导；而参考程序则列出了实验要求中，特别是一些重要算法的参考描述；教学环节组织则明确了教学中师生的互动及职责；思考题则是布置给学生对同类实验的一种思考和提示。

一. 关于实验步骤的要求和建议

从以往的教学事先实验的经验来看，在初学阶段执行严格的实验步骤规范（包括上机操作规范），机时利用率会大大提高，有助于养成良好的程序编制风格，培养严谨、科学、高效的工作方式。

在以往的教学实践中，经常发现很多学生抱怨说，花了两个小时才找出一个错误，甚至一无所获。他们不明白造成这种情况的原因，正是他们自己。有的学生不屑于按实验步骤规范去做，甚至对于实验步骤的要求和建议看都不看一遍，认为那是浪费时间，这是及其害的。实验步骤规范不但可以培养科学化的工作作风，而且还能有效地避免错误。

具体的步骤规范如下：

1.问题分析与系统的结构设计：

充分地分析和理解问题本身，弄清要求作什么，限制条件是什么。按照以数据结构为中心的原则划分模块，即定义数据结构及其在这些结构之上的操作，使得对数据结构的存取通过这些操作加以实现。在这个过程中，要综合考虑系统功能。要考虑系统结构清晰、合理、简单并且易于调试。最后写出每个子程序（过

程或函数）的规格说明，列出它们之间的调用关系，可以使用调用关系图表示则更加清晰，这样便完成了系统结构设计。

2.详细设计和编码

详细设计的目的是对子程序（过程或函数）的进一步求精。用 IF 、WHILE 和赋值语句等，以及自然语言写出算法的框架。利用自然语言的目的是避免陷入细节。在编码时，可以对详细设计的结果进一步求精，用高级语言表示出来。

程序的每一行最好不超过 60 个字符。每个子程序（或过程、函数）通常不要太长，以 40 行为宜。子程序（或过程、函数）包含的程序行数太多，容易造成理解的困难。控制 IF 、WHILE 等语句的连续嵌套的深度。程序的目的性必须明确。对每一段程序完成的作用，除非常明显的除外（如：x = x + 1；注释为 x 加 1，没有什么意义），都应加以注释。这会对程序的调试提供很多方便。根据情况可以设立若干调试点，即输出若干信息，用于验证和你的设想是否一致。另外，对于输入输出语句，必须对它们的作用加以说明。否则，在调试程序时，无法了解系统需要输入什么样的数据，系统输出的又是什么。程序的书写，必须按照一定的规范，如保留字小写时涂黑，或者大写等等，风格要统一。具体的要求可参看软件工程中的有关规定。

3.上机准备和静态检查

上机准备：

- 高级语言文本
- 熟悉机器的用户手册，熟悉常用的命令。
- 准备调试的工具，考虑调试方案。如果机器上没有现成的调试工具可供利用，可以自己先设计一些以供使用。
- 静态检查

自己用一组数据手动执行程序；或同同学一起阅读自己的程序，以全面地了解该程序的逻辑。

4.上机调试程序

自底向上，先调试底层模块，再调试上层模块。最后，整个程序进行联合调试。调试正确后将源程序和运行结果加以打印输出。

5. 实验报告的整理

- 需求及规格说明

问题描述，求解的问题是什么。

- 设计：

设计思想：存储结构、主要的算法思想。

设计表示：子程序（过程或函数）的规格说明，通过调用关系图表　示它们之间的调用关系。

实现注释：

详细设计表示：主要算法的框架。

- 用户手册：使用说明。
- 调试报告：问题是如何解决的，讨论与分析、改进设想、经验与体会、时空复杂度等。
- 附录

源程序清单和结果：源程序必须有注释，以及必要的测试数据和运行结果数据。提倡用英文描述。

- 实验报告要求：

按实验报告模板要求编写。

实验一 抽象数据类型设计与实现

本次实验的主要目的在于帮助读者熟悉抽象数据类型的表示和实现方法。抽象数据类型需借助固有数据类型来表示和实现，即利用高级程序设计语言中已存在的数据类型来说明新的结构，用已经实现的操作来组合新的操作，具体实现细节则依赖于所用语言的功能。通过本次实验还可以帮助学生复习高级语言的使用方法。

【问题描述】

用C或C++语言设计并实现一个可进行复数运算的演示程序。

【基本要求】

1. 由输入的实部和虚部生成一个复数
2. 两个复数求和
3. 两个复数求差
4. 从已知复数中分离出实部和虚部
5. 复数及相应运算结果以相应的表现形式显示。

【CDIO 项目要求】

1. 有完整的 CDIO 四个阶段描述
2. 有友好美观的操作界面
3. 有软件使用说明或帮助文档
4. 项目成员分工明确，团结协作

【实验内容】

难度 A：设计一个可进行复数运算的演示程序。实现抽象数据类型—复数，及构造复数；个人完成，评分最高 70 分。

难度 B：在 A 的基础上实现复数求和、求差、求积、复数显示等操作；个人完成，评分最高至 90 分。

难度 C：在 B 的基础上设计一个桌面计算器，能够进行实数或复数的四则混合运算。由 2-3 个人的团队完成，评分最高可至 100 分。

【实现提示】

定义复数为由两个相互之间存在次序关系的实数构成的抽象数据类型，则可以利用实数的操作来实现复数的操作。

（1）抽象数据类型的含义

一个抽象数据类型ADT就是：数据 + 操作，例如CpxNum其数据部分包括实部和虚部；而操作包括CPlus（加法）、CMinus（减法）、cmultiply（乘法）、cdivide（除法）等运算。

我们可通过在高级语言中实现新的抽象数据类型，来扩展语言的类型体系，如在C++语言原有的int、char、float、double等内建类型的基础上，我们还可根据用户的需求提供诸如：复数、CMyString、一元多项式、发票等等结合实际应用环境特征的新型类型。这样作为高级语言的用户既可用int、char、float等基本类型，又可在其程序中使用复数、CMyString、一元多项式等用户自定义类型，这样大大降低了“类型使用者”编程的工作量和复杂性。

（2）复数的四则运算法则：

设 $z1=a+bi$，$z2=c+di$，（a，b，c，$d\in R$，以下不再说明）

□加减法：$(a+bi)\pm(c+di)=(a\pm c)+(b\pm d)i$

□乘法：$(a+bi)*(c+di)=(ac-bd)+(ad+bc)i$

□除法：$\dfrac{a+bi}{c+di}=\dfrac{(a+bi)(c-di)}{c^2+d^2}=\dfrac{(ac+bd)(bc-ad)i}{c^2+d^2}$

（3）复数的存储结构及接口函数声明：

复数的存储结构

```
//定义复数类型的存储结构，此部分对于cpxNum类型的使用者不公开
typedef struct {
double _real; //复数的实部
double _imag; //复数的虚部
 } cpxNum; //定义结构体类型cpxNum表示“复数”
```

复数的接口函数声明

```
/*复数类型接口函数的定义部分,此部分对cpxNum类型的使用者公开*/
/*用double r, double i 初始化复数c*/
void assign(cpxNum& c, double r, double i);
/*以‘1+2i 或1-3i’的形式将复数c输出到控制台窗口*/
void print(cpxNum& c);
/*实现两个复数c1, c2的加法，和作为函数cplus的返回值*/
cpxNum cplus(const cpxNum& c1, const cpxNum& c2);
/*实现两个复数c1, c2的减法，差作为函数cmilus的返回值*/

cpxNum cmilus(const cpxNum& c1, const cpxNum& c2);
/*实现两个复数c1, c2的乘法，积作为函数cmultiply的返回值*/
cpxNum cmultiply(const cpxNum& c1, const cpxNum& c2);
/*实现两个复数c1, c2的乘法，商作为函数cdivide的返回值*/
cpxNum cdivide(const cpxNum& c1, const cpxNum& c2);
```

/*复数接口函数实现示例*/

```
cpxNum cplus(const cpxNum& c1, const cpxNum& c2)
{
cpxNum result;
```

（4）复数类型“使用者”代码示例：

```
/*复数的加法：c1的实部+c2的实部、c1的虚部+c2的虚部*/
result._real = c1._real + c2._real;
      result._imag = c1._imag + c2._imag;
      return result;
}
#include "cpxNum.h"
int _tmain(int argc, _TCHAR* argv[])
{
cpxNum c1, c2; //声明两个‘复数’类型的变量
double real, imag;
cout <<"请输入第一个复数的实部和虚部: ";
cin >> real >>imag;
assign(c1, real, imag); //初始化‘复数’c1
```

```
cout <<"您生成的第一个复数是: ";
print(c1);
cout << endl <<"请输入第二个复数的实部和虚部: ";
cin >> real >>imag;
assign(c2, real, imag); //初始化‘复数’c2
cout <<"您生成的第二个复数是: ";
print(c2);
cout << endl;
cout << "**********复数运算测试**********"<<endl;
cout << "c1 + c2的结果是: "; print(cplus(c1, c2)); cout << endl;
cout << "c1 - c2的结果是: "; print(cmilus(c1, c2)); cout << endl;
cout << "c1 * c2的结果是: "; print(cmultiply(c1, c2)); cout << endl;
cout << "c1 / c2的结果是: "; print(cdivide(c1, c2)); cout << endl;
cout << "**********复数运算测试结束**********"<<endl;
return 0;
}
```

【测试数据】

参见《数据结构题集》page76.

```
/*复数的加法：c1的实部+c2的实部、c1的虚部+c2的虚部*/
result._real = c1._real + c2._real;
    result._imag = c1._imag + c2._imag;
    return result;
}
#include "cpxNum.h"
int _tmain(int argc, _TCHAR* argv[])
{
cpxNum c1, c2; //声明两个‘复数’类型的变量
double real, imag;
cout <<"请输入第一个复数的实部和虚部: ";
cin >> real >>imag;
assign(c1, real, imag); //初始化‘复数’c1
cout <<"您生成的第一个复数是: ";
print(c1);
cout << endl <<"请输入第二个复数的实部和虚部: ";
cin >> real >>imag;
assign(c2, real, imag); //初始化‘复数’c2
cout <<"您生成的第二个复数是: ";
print(c2);
cout << endl;
cout << "**********复数运算测试**********"<<endl;
cout << "c1 + c2的结果是: "; print(cplus(c1, c2)); cout << endl;
cout << "c1 - c2的结果是: "; print(cmilus(c1, c2)); cout << endl;
```

```
cout << "c1 * c2的结果是: "; print(cmultiply(c1, c2)); cout << endl;
cout << "c1 / c2的结果是: "; print(cdivide(c1, c2)); cout << endl;
cout << "***********复数运算测试结束*************"<<endl;
return 0;
}
```

主要教学环节的组织：

1、指导教师介绍实验环境，说明实验目的，布置实验项目。

2、学生上机编程实践。

3、指导教师验收学生的程序，简单评价并给出建议，必要情况下给出指导。

4、学生对实验进行总结，整理相关材料，完成实验报告。

思考题：

从接到实验任务到编码解决问题，经历了哪些过程？

实验二 线性表及其应用

本次实验的主要目的在于帮助读者熟练掌握线性表的基本操作在链表存储结构上的实现。

【问题描述】

用C或C++语言设计并实现一个一元稀疏多项式的简单计算器。

【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，序列按指数降序排列

3、多项式A(x)和B(x)相加，并建立多项式A(x)+B(x)

4、多项式A(x)和B(x)相减，并建立多项式A(x)-B(x)

5、给定 x 的值，计算多项式

6、多项式A(x)和B(x)相乘，建立多项式A(x)*B(x) （* 选做，作为难度B的操作）

【CDIO项目要求】

1、有完整的CDIO四个阶段描述

2、有友好美观的操作界面

3、有软件使用说明或帮助文档

4、项目成员分工明确，团结协作

【实验内容】

难度 A：设计一个一元稀疏多项式简单计算器。输入并建立多项式、按指数降序输出多项式；个人完成，评分最高 70 分。

难度 B：在 A 的基础上实现、多项式相加、相减、求导等操作；个人完成，

评分最高至 90 分。

难度 C：在 B 的算法基础上设计一个线性方程组求器，能够进行线性方程组的求解，包括判断是否有解、如果有解，输出基础解系及通解。。由 2-3 个人的团队完成，评分最高可至 100 分。

【实现提示】

1、一元稀疏多项式四则运算示例

$A_3(x)=7+3x+9x^8$

$B_2(x)=-3.2x+x^7$

$A_3(x)+B_2(x)=9x^8+1x^7+(-0.2)x^1+7x^0$

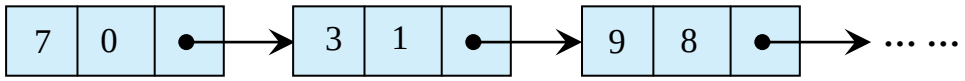
$A_3(x)-B_2(x)=9x^8+(-1)x^7+6.2x^1+7x^0$

$A_3(x)*B_2(x)=9x^{15}+(-28.8)x^8+3x^7+(-9.6)x^2+(-22.4)x^1$

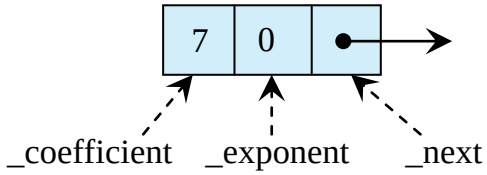
$A_3(1.0)*B_2(1.0)=-41.8$

2、用带头结点的单链表作为多项式的存储结构

h　　Class polynomial　　$A(x)=7+3x+9x^8$



Class node



（1）一元稀疏多项式存储结构定义及接口函数声明

```
//定义多项式链表的'结点'存储结构，每一个结点对应稀疏多项式中的一项
typedef struct LNode{
float _coef; //多项式的系数
int _expn; //多项式的指数
struct LNode *_next; //指向下一个多项式结点的指针
} *Link; //定义了一个新的类型Link，该类型变量是指向多项式结点的指针
//定义'多项式链表'存储结构
typedef struct{
Link _head,_tail; //指向多项式链表的‘头结点’、‘尾结点’的指针
int _len; //多项式链表中除去头结点后的结点数，即多项式的项数
}LinkList;
typedef LinkList Polynomial; //定义一元稀疏多项式类型Polynomial
**********************************************************************************
/*带头节点单链表类型接口函数的定义部分,此部分对生成一元稀疏多项式链表的开发者公开*/
/* 根据用户输入的多项式的系数和指数float coef, int expn，生成多项式结点，
```

```
并用p指针指向生成的新结点，请注意参数p的传递方式，生成新结点可使用
C++中的new操作*/
bool MakeNode(Link& p, float coef, int expn);
/* 初始化一个带头节点单链表头、尾指针均指向_head, _tail头结点*/
void InitList(LinkList& L);
/*销毁带头节点单链表L，多项式链表使用完后，应将链表占用的内存进行释放，
释放结点内存可使用C++中的delete操作*/
void DestroyList(LinkList& L);
/*一元多项式类型接口函数的定义部分,此部分对Polynomial类型的使用者公开*/
/*基于用户输入的一元多项式系数向量和指数向量
vector<double>& coef, vector<int>& expon 生成一元稀疏多项式链表*/
void CreatPolyn(Polynomial& p, vector<double>& coef, vector<int>& expon);
/*销毁一元稀疏多项式链表，程序结束前应调用此函数释放多项式链表内存*/
void DestroyPolyn(Polynomial& p);
/*以多项式指数递减格式输出一元稀疏多项式，例如：
34X^100 + 45X^60 -4X^2 + 3*/
void PrintPolyn(const Polynomial& p);
/*两个一元稀疏多项式Pa、Pb相加、相减、相乘，生成和多项式链表PSum*/
void AddPolyn(Polynomial& Pa,Polynomial& Pb,Polynomial& PSum);
void SubPolyn(Polynomial& Pa,Polynomial& Pb,Polynomial& PSub);
void MultiplyPolyn(Polynomial& Pa,Polynomial& Pb,Polynomial& PMulti);
```

3、多项式的输出形式为类数学表达式，例如，多项式$34x^{100}+45x^{60}-4x^2+3$的输出形式为：34X^100+45X^60-4X^2+3。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$1x^2$输出形式为x^2。

4、简单线性方程组的求解过程将运用到多项式的加、减等运算操作进行消元，如果复杂的线性方程组，需要用矩阵运算求解，具体算法参照工程数学线性代数部分。

【测试数据】

参见《数据结构题集》page81.

【主要教学环节的组织】

1、指导教师介绍实验环境，说明实验目的，布置实验项目。

2、学生上机编程实践。

3、指导教师验收学生的程序，简单评价并给出建议，必要情况下给出指导。

4、学生对实验进行总结，整理相关材料，完成实验报告。

【思考题】

线性表的存储可以采用顺序存储结构和链式存储结构两种方式,在解决问题过程中二者有何不同，如何选择？

实验三 栈和队列及其应用

本次实验的目的在于使学生深入了解栈和队列的特性，以便在实际问题背景下灵活运用它们；同时还将巩固对这两种结构构造方法的理解。

【问题描述】

设计并实现魔王语言的解释器，具体要求如下：大写字母表示魔王语言的词汇；小写字母表示人的词汇语言；魔王语言中可包含括号。

如，我们有魔王语言的解释规则：

B →tAdA；A→ sae；(ehnxgz) → ezegexenehe

则魔王语言B(ehnxgz)B解释成tsaedsaeezegexenehetsaedsae。

【基本要求】

1、 魔王语言的产生式规则可在程序中指定。
2、 能接收用户输入的合法的魔王语言（包含产生式中的大写字母及1个括号）。
3、 解释合法的魔王语言。
4、 在程序代码中预先定义人类词汇所对应的中文字符，将“人类语言”翻译成为中文语言。

【CDIO项目要求】

1、有完整的CDIO四个阶段描述
2、有友好美观的操作界面
3、有软件使用说明或帮助文档
4、项目成员分工明确，团结协作

【实验内容】

难度A： 实现一个魔王语言解释器。根据产生式规则

$$\alpha \to \beta_1\beta_2...\beta_m ,$$

$$(\theta\delta_1\delta_2...\delta_n) \to$$
$$\theta\delta_n\theta\delta_{n-1}\theta...\theta\delta_1\theta$$

对魔王语言进行翻译。如：应用规则：

$$B \to tAdA$$

$$A \to sae$$

$$(\theta\delta_1\delta_2\cdots\delta_n) \to \theta\delta_n\theta\delta_{n-1}\cdots\theta\delta_2\theta\delta_1\theta$$

对魔王的话：B(ehnxgz)B 进行翻译。；个人完成，评分最高 70 分。

难度 B： 在 A 的基础上，（根据产生式）自定义规则，将一段魔王的话翻译为有意义的人类语言（中文）；个人完成，评分最高至 90 分。

难度 C：在 B 的基础上，实现一个文件译码器，对加密的文件进行译码。由 2-3 个人的团队完成，评分最高可至 100 分。

【实现提示】

1、 将魔王的语言自右至左进栈，总是处理栈顶字符。若是开括号，则逐一出栈，将字母顺序入队列，直至闭括号出栈，并按规则要求逐一出队列再处理后入栈。（翻译过程中栈和队列的应用可根据算法的设计选取）。

2、 熟悉栈和队列的操作

（1）抽象数据类型‘顺序栈’、‘循环队列’的存储结构及接口声明

```
//---------------抽象数据类型顺序栈的定义-----------------//
#define STACK_INIT_SIZE 100 //栈元素的初始存储空间大小
#define STACK_INCREMENT 10
typedef char SElemType,QElemType;
typedef struct {
SElemType *base; //顺序栈的栈底指针
SElemType *top; //顺序栈的栈顶指针
int StackSize; //栈元素空间的大小
}SqStack; //结构体类型顺序栈
//--------------------顺序栈的接口函数声明----------------------//
//构造一个空栈
bool InitStack(SqStack &S);
//销毁栈,释放栈所占内存空间
void DestroyStack(SqStack &S);
//把S置为空栈
bool ClearStack(SqStack &S);
//若栈S为空栈,返回true,否则返回false
bool StackEmpty(SqStack &S);
//返回栈S的元素个数,即栈的长度
int StackLength(SqStack &S);
//若栈不空,则用e返回S的栈顶元素,并返回ok;否则返回error
bool GetTop(SqStack &S,SElemType &e);
//插入新元素e为新的栈顶元素
bool Push(SqStack &S,SElemType e);
//若栈不空,删除S的栈顶元素,并用e返回其值,并返回true,否则返回false
bool Pop(SqStack &S,SElemType& e);
//---------------抽象数据类型循环队的定义-----------------//
#define MAXQSIZE 100
typedef struct {
QElemType *base; //指向循环队列元素存储空间的指针
int front; //循环队列的队头指针
int rear; //循环队列的队尾指针
}SqQueue; //结构体类型循环队列
//--------------------循环队的接口函数声明----------------------//
//初始化循环队
```

```
bool InitQueue(SqQueue &Q);
//返回Q的元素个数,即队列的长度
int QueueLength(SqQueue &Q);
//插入新元素e为Q的新队尾元素
bool EnQueue(SqQueue &Q, QElemType e);
//若队列不空,则删除Q的队头元素,用e返回其值,并返回true,否则返回false;
bool DeQueue(SqQueue &Q, QElemType &e);
```

（2）魔王语言翻译算法伪码

```
string BeelzebubLang = "";//魔王语言串
cout <<"请输入魔王语言串: ";
cin >> BeelzebubLang;
SqStack ChStack; //定义一个字符栈变量ChStack
SqQueue ChQueue; //定义一个字符队列变量ChQueue
SqStack TransLang;
InitStack(ChStack);
InitQueue(ChQueue);
InitStack(TransLang);
//将魔王语言串按照自左向右顺序进栈
for(int i = 0; i<BeelzebubLang.size(); i++)
Push(ChStack,BeelzebubLang[i]);
SElemType e,ReverseFirstElem;
//栈ChStack不空, 按照预先定义好的产生式规则翻译魔王语言
while(!StackEmpty(ChStack)){
Pop(ChStack,e);
switch( e ) //分以下几种情况对魔王语言串按照产生式进行翻译
{
case 'B':{
//如果栈顶元素是非终结符B,则将产生式B  tAdA中符号 '   ?'
//右边的字符串按照自左至右顺序逐一进栈
}
break;
}//end case
case 'A':{
//如果栈顶元素是非终结符A,则将产生式A  sae中, 符号   ?右边字符串
//按照自左至右顺序进栈
}//end case
case ')':{ //说明需要处理（.....）的内容
//如果栈顶元素是终结符),则连续出栈,将每个出栈的元素进队
//将最后出栈的元素作为ReverseFirstElem;
//先压ReverseFirstElem到栈中;
//将队列中的其余元素进栈,生成新序列;
}//end case
case '(':{ //
```

```
//如果栈顶元素是终结符(,则说明()处理完毕,Reverse置为false
}
default: { //如果不是上述几种情况，则将字符入队列TransLang
Push(TransLang,e);
break;
}
   }//end switch
}//end while
```

3、 文件译码器即可以将一个通过一定的编码规则进行加密的文件（文字等信息）翻译成可用信息。其实现过程和魔王语言的翻译过程类似。

4、 注意合法字符（符合规则，有意义的字符）的判断。

【测试数据】

参见《数据结构题集》page97.

【主要教学环节的组织】

1、指导教师介绍实验环境，说明实验目的，布置实验项目。

2、学生上机编程实践。

3、指导教师验收学生的程序，简单评价并给出建议，必要情况下给出指导。

4、学生对实验进行总结，整理相关材料，完成实验报告。

【思考题】

1、栈和队列结构的操作有何特点，在解决哪些问题时会应用到？

2、实现上，栈和队列常用哪种存储结构？为什么？

实验四　数组的表示及其应用

本次实验的主要目的在于帮助读者熟悉矩阵的表示和应用。学会运用矩阵对实际问题进行建模和设计，熟练运用矩阵求解问题。

【问题描述】

以一个 m×n 的长方阵表示迷宫，0 和 1 分别表示迷宫中的通路和障碍。设计一个程序，对任意设定的迷宫，求出一条从入口到出口的通路，或得出没有通路的结论。

【基本要求】

首先实现一个以链表作存储结构的栈类型，然后编写一个求解迷宫的非递归程序。求得的通路以三元组(i，j，d)的形式输出，其中:(i，j)指示迷宫中的一个坐标，d 表示走到下一坐标的方向。如；对于下列数据的迷宫，输出的一条通路为：(l，1，1)，(1，2，2)，(2，2，2)，(3，2，3)，(3，1，2)，…。

【CDIO 项目要求】

4. 有完整的 CDIO 四个阶段描述
5. 有友好美观的操作界面
6. 有软件使用说明或帮助文档

4.项目成员分工明确，团结协作

【实验内容】

难度 A: 运用矩阵来表示迷宫。能根据用户指定的维数自动生成迷宫，并打印迷宫中各个位置的状态。个人完成，评分最高 70 分。

难度 B: 在 A 的基础上实现迷宫的自动路径搜索，判断是否存在从起点到终点的通路；个人完成，评分最高 90 分。

难度 C:在 B 的算法基础上设计一个具有可玩性和趣味性的迷宫游戏。由 2-3 个人的团队完成，评分最高可至 100 分。

【实现提示】

计算机解迷宫通常用的是“穷举求解”方法，即从入口出发，顺着某一个方向进行探索，若能走通，则继续往前进；否则沿着原路退回，换一个方向继续探索，直至出口位置，求得一条通路。假如所有可能的通路都探索到而未能到达出口，则所设定的迷宫没有通路。

可以二维数组存储迷宫数据，通常设定入口点的下标为(1, 1)，出口点的下标为(n, n)。为处理方便起见，可在迷宫的四周加一圈障碍。对于迷宫中任一位置，均可约定有东、南、西、北四个方向可通。

1. 坐标位置类型

```
typedef struct{
int r, c;      //迷宫中r行c列的位置
}  PosType;
```

2. 迷宫类型

```
typedef struct{
int m,n;
char  arr[RANGE] [RANGE];
} MazeType;
void InitMaze (MazeType &maze,int a[][],int row,int col)
//按照用户输入的row行和col列的二维数组（元素值为0或1）
//设置迷宫maze的初值，包括加上边缘一圈的值
bool MazePath(MazeType & maze, PosType start,PosType end)
//求解迷宫maze中，从入口start到出口end的一条路径，
//若存在，则返回TRUE;否则返回FALSE
void PrintMaze(MazeType maze)
//将迷宫以字符型方阵的形式输出到标准输出文件上
```

3. 栈类型

```
typedef struct(
int    step;     //当前位置在路径上的“序号”
PosType    seat;      //当前的坐标位置
directiveType di;     //往下一坐标位置的方向
```

```
    ) ElemType;      //栈的元素类型

typedef struct NodeType{
  ElemType    data;
  NodeType    *next;
}  NodeType, *LinkType; //结点类型, 指针类型

typedef struct{
  LinkType top;
  int    size;
} Stack;      //栈类型
```

4. 求迷宫路径的伪码算法:

```
Status MazePath (MazeType maze. PosType start, PosType end)
{
  //若迷宫maze中存在从入口start到出口end的通道, 则求得一条存放在栈中
  //（从栈底到栈顶为从人口到出口的路径）. 并返回TRUE;否则返回FALSE
    InitStack(S); curpos=start;      //设定“当前位置“为“入口位置”
    Curstep=1; found=FALSE;      //探索第一步
    do{
    jf (Pass  (maze,  curpos))(
    //当前位置可以通过, 即是未曾走到过的通道块留下足迹
    FootPrint (maze, curpos);
    e=( curstep, curpos,1);
    Push(S, e);     //加入路径
    if (Same(curpos,end)) found =TRUE;    //到达终点（出口）
    else{
    curpos=NextPos(curpos, 1);    //下一位置是当前位置的东邻
    curstep- -;     //探索下一步
    ) //else
    }//if
    else     //当前位置不能通过
    if( !StackEmpty(S)){
    Pop (S,e);
    while  (e.di= =4 && !StackEmpTy (S)){
    MarkPrint(maze,e.seat);Pop(S,e);
        curstep- -;     //留下不能通过的标记, 并退回一步
        }//while
        if(e.di<4){
        e.di++;  Push(S, e);     //换下一个方向探索
        curpos=NextPos(e.seat,e.di); //设定当前位置是该新方向上
        的相邻块
        }//if
        }//if
```

```
    }  while( !StackEmpty(S)&& !found){
    return found;
}//MazePath
```

主要教学环节的组织：

1、指导教师介绍实验环境，说明实验目的，布置实验项目。

2、学生上机编程实践。

3、指导教师验收学生的程序，简单评价并给出建议，必要情况下给出指导。

4、学生对实验进行总结，整理相关材料，完成实验报告。

思考题：

从接到实验任务到编码解决问题，经历了哪些过程？

实验五 树及其应用

树是应用极为广泛的数据结构，树的基本理论和实验是本门课程的重点内容之一。树结构的特点在于“非线性”。本实验单元继续突出数据结构 + 操作构成抽象数据类型的程序设计观点，根据树结构非线性的特点，将操作进一步集中到树存储结构生成和遍历操作实现上，因为树的遍历操作是其他众多操作的基础。本实验还希望学生熟悉树结构的逻辑特性，应用树结构解决具体问题。

【问题描述】

用C或C++语言建立二叉树，并实现二叉树的先根、中根、后根遍历。在此基础上，实现树与二叉树的相互转换。

实现一个简单的哈夫曼编码/译码器，能根据一段电文设计赫夫曼编码，并能用该编码对一段给定的电文进行译码。

【基本要求】

1、熟悉树和二叉树结构。

2、掌握树和二叉树的存储以及各种操作。

3、学会运用树和二叉树结构求解问题。

【CDIO 项目要求】

7. 有完整的 CDIO 四个阶段描述

8. 有友好美观的操作界面

9. 有软件使用说明或帮助文档

10. 项目成员分工明确，团结协作

【实验内容】

难度 A：建立二叉树，并实现二叉树的先根、中根、后根遍历；个人完成，评分最高 70 分。

难度 B：在 A 的基础上，实现树与二叉树的相互转换；个人完成，评分最高至 90 分。

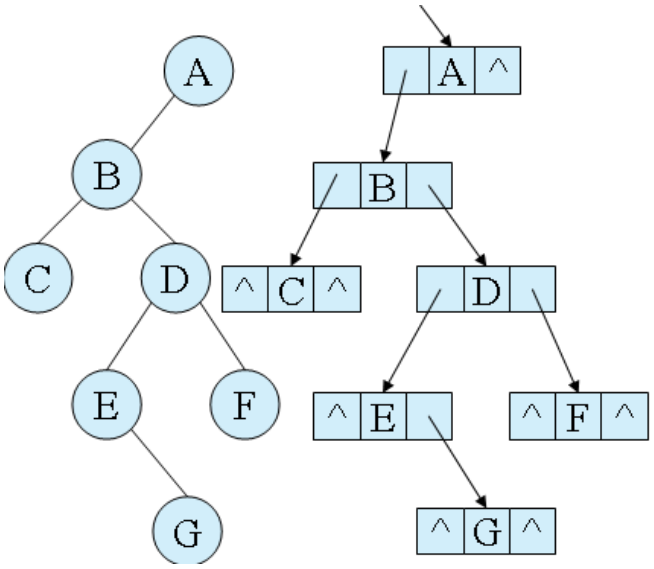
难度 C：实现一个简单的哈夫曼编码/译码器，能根据一段电文设计赫夫曼编码，并能用该编码对一段给定的电文进行译码。由 2-3 个人的团队完成，评分最高可至 100 分。

【实现提示】

1、用链式存储结构作为二叉树的存储结构。

```
//－－－二叉树的二叉链表存储结构表示－－－－
typedef struct BiTNode
 { TElemType data; // 结点的值
   struct BiTNode *lchild,*rchild; // 左右孩子指针
 }BiTNode,*BiTree;
```

2、根据二叉树的定义建立二叉树的二叉链表，不同的定义方法有不同的二叉树建立算法，实验时，可根据情况实现一种建立二叉树的算法。这里以给定先序序列建立二叉链表为例，给出建立二叉树的参考程序。如现有带空格先序序列的字符串：ABC##DE#G##F###（其中#表示空格字符），可以建立如下图所示的相应二叉链表。



根据给定先序序列建立二叉链表的程序示例如下（教科书算法6.4）：

```
Status CreateBiTree(BiTree &T)
{ //按先序次序输入二叉树中结点的值(可为字符型或整型，在主程中定义)，
  // 构造二叉链表表示的二叉树T。空格字符表示空树。
  TElemType ch;
  scanf("%c",&ch); // 输入结点的值
  if(ch==' ') // 结点的值为空
    T=NULL;
  else // 结点的值不为空
  { T=(BiTree)malloc(sizeof(BiTNode)); // 生成根结点
    if(!T)
      exit(OVERFLOW);
    T->data=ch; // 将值赋给T所指结点
    CreateBiTree(T->lchild); // 递归构造左子树
    CreateBiTree(T->rchild); // 递归构造右子树
  }
  Return OK;
}//CreateBiTree
```

3、二叉树遍历主要有先根、中根和后根遍历，可以用递归算法或非递归算法来实现三种遍历。下面以中序遍历为例，给出递归和非递归算法的程序示例：

```
void InOrderTraverse(BiTree T,void(*Visit)(TElemType))
{ // 初始条件：二叉树T存在，Visit是对结点操作的应用函数
  // 操作结果：中序递归遍历T，对每个结点调用函数Visit一次且仅一次
  if(T)
  { InOrderTraverse(T->lchild,Visit); // 先中序遍历左子树
    Visit(T->data); // 再访问根结点
    InOrderTraverse(T->rchild,Visit); // 最后中序遍历右子树
```

```
   }
 }

void InOrderTraverse(BiTree T,void(*Visit)(TElemType))
 { // 采用二叉链表存储结构，Visit是对数据元素操作的应用函数。
   // 中序遍历二叉树T的非递归算法(利用栈),对每个数据元素调用函数Visit
   SqStack S;
   BiTree p;
   InitStack(S); // 初始化栈S
   Push(S,T); // 根指针进栈
   while(!StackEmpty(S)) // 栈不空
   { while(GetTop(S,p)&&p) // 栈顶元素不为空指针
       Push(S,p->lchild); // 向左走到尽头，入栈左孩子指针
     Pop(S,p); // 空指针退栈，退掉最后入栈的空指针
     if(!StackEmpty(S)) // 访问结点，向右一步
     { Pop(S,p); // 弹出栈顶元素(非空指针)
       Visit(p->data); // 访问刚弹出的结点(目前栈顶元素的左孩子)
       Push(S,p->rchild); // 入栈其右孩子指针
     }
   }
  }
```

4、树与二叉树的相互转换：在树与二叉树之间有一个自然的一一对应的关系，每一棵树都能唯一转换到它所对应的二叉树，而二叉树也可以转换成一棵树。通常情况下，把树转换成二叉树更有意义，这样可以把对于树的相关操作转换成二叉树的操作来实现。

请根据教科书上树与二叉树转换的规则，通过查找资料，实现树与二叉树的相互转换的算法程序，注意选择树的存储结构。

5、利用哈夫曼编码进行通信可以大大提高信道利用率，缩短信息传输时间，降低传输成本。但是，这要求在发送端通过一个编码系统对待传数据预先编码，在接收端将传来的数据进行译码（复原）。对于双工信道（即可以双向传输信息的信道），每端都需要一个完整的编/译码系统。为这样的信息收发站写一个哈夫曼码的编/译码系统。

提示：
(1) 分析待编码的电文中出现的字符集，并统计各字符出现的频度(作为权值)
(2) 从终端读入字符集大小 n，以及 n 个字符和 n 个权值，建立哈夫曼树
(3) 利用已建立好的哈夫曼树对正文进行编码，并显示编码结果

(4) 利用已建立好的哈夫曼树对编码后的代码进行译码

测试数据：

(1) 利用教科书例 6-2 中的数据调试程序

(2) 用下表给出的字符集和频度的实际统计数据建立哈夫曼树，并实现以下报文的编码和译码：“THIS PROGRAM IS MY FAVORITE”。

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

主要教学环节的组织：

1、指导教师说明实验目的，布置实验项目。

2、学生上机编程实践。

3、指导教师验收学生的程序，简单评价并给出建议，必要情况下给出指导。

4、学生对实验进行总结，整理相关材料，完成实验报告。

思考题：

不同存储结构对于算法效率有哪些影响？

实验六　图及其应用

图是本门课程所涉及的另一种重要的数据结构。本实验继续训练学生在理解数据结构特性基础上，灵活运用数据结构解决实际问题的能力。

【基本要求】

1、熟悉图结构。

2、掌握图的存储以及图形结构上的各种操作。

3、学会运用图结构求解问题

【问题描述】

根据云南大学呈贡校区主要道路建筑示意图（见下图，可放大），以已标出的距离为参照，测量出主要建筑（教学楼、办公楼、食堂、学生宿舍、运动场、乘车点等）之间的路程（距离）。以图为工具，建立模型，求解不同难度的问题。要求能够回答有关建筑介绍、游览路径等问题。用户（师生）通过终端可询问：

（1）从某一建筑到另一建筑的最短路径。(最短路径问题)
（2）用户从校园某处（由用户选定）进入，选取一条最佳路线。
（3）使用户可以不重复地浏览各建筑，最后回到出口（出口就在入口旁边）。

【CDIO 项目要求】

1. 有完整的 CDIO 四个阶段描述
2. 有友好美观的操作界面
3. 有软件使用说明或帮助文档
4. 项目成员分工明确，团结协作

【实验内容】

难度 A：用图来表示一个校园内各种地名（不少于 10 个），及其相对位置，实现校园内主要地点的遍历。个人完成，评分最高 70 分。

难度 B：在 A 的基础上，能求解任意两个地点之间的最短距离。个人完成，评分最高 90 分。

难度 C：实现一个能为新生指路的校园自动导游程序。由 2-3 个人的团队完成，评分最高可至 100 分。

【实现提示】

（1）问题解析

用无向网表示校区内的各建筑的平面图，图中顶点表示主要建筑，存放建筑的编号、名称、简介等信息，图中的边表示建筑间的道路，存放路径长度等信息。

（2）将导游图看作一张带权无向图，顶点表示校园的各个建筑，边表示各建筑之间的道路，边上的权值表示距离，为此图选择适当的数据结构。

（3）把各种路径都显示给用户，由用户自己选择浏览路线。

2. 算法提示

（1）构造一个无向图 G 并用邻接矩阵来存储。

（2）利用迪杰斯特拉算法来计算出起点到各个顶点之间的最短路径用二维数组 p[i][]来记 录，最短路径长度就用一维数组 d[i]存放；i 的范围：0～20。

（3）一维数组 have[]可用来记录最短路径出现顶点的顺序。

（4）根据起点和终点输出最短路径和路径长度。

参考程序结构及界面：

```
#include "string.h"
#include "stdio.h"
#include "stdio.h"
#include "malloc.h"
#include "stdlib.h"
#define Max 20000
#define NUM 9
typedef struct ArcCell{
  int adj;   /* 相邻接的建筑之间的路程 */
}ArcCell;     /* 定义边的类型 */
typedef struct VertexType{
  int number;   /* 建筑编号 */
  char *sight;   /* 建筑名称 */
  char *description; /* 建筑描述 */
}VertexType;     /* 定义顶点的类型 */
typedef struct{
  VertexType vex[NUM]; /* 图中的顶点，即为建筑 */
  ArcCell arcs[NUM][NUM]; /* 图中的边，即为建筑间的距离 */
  int vexnum,arcnum; /* 顶点数，边数 */
}MGraph;     /* 定义图的类型 */
MGraph G;     /* 把图定义为全局变量 */
int P[NUM][NUM];   /*   */
long int D[NUM];   /* 辅助变量存储最短路径长度 */
int     x[9]={0};
void CreateUDN(int v,int a); /* 造图函数 */
void narrate();     /*说明函数*/
void ShortestPath(int num); /*最短路径函数*/
void output(int sight1,int sight2); /*输出函数*/
char Menu();     /* 主菜单 */
void search();     /* 查询建筑信息 */
char SearchMenu();   /* 查询子菜单 */
void     HaMiTonian(int);     /* 哈密尔顿图的遍历 */
void     NextValue(int);
void     display(); /* 显示遍历结果 */
```

```
void main() /*  主函数  */
{

  int v0,v1;
  char ck;
  system("color fc");
  CreateUDN(NUM,11);
  do
  {
    ck=Menu();
    switch(ck)
    {
    case '1':
      system("cls");
      narrate();
      printf("\n\n\t\t\t 请选择起点建筑（0～8）：");
      scanf("%d",&v0);
      printf("\t\t\t 请选择终点建筑（0～8）：");
      scanf("%d",&v1);
      ShortestPath(v0);    /*  计算两个建筑之间的最短路径  */
      output(v0,v1);          /*  输出结果  */
      printf("\n\n\t\t\t\t 请按任意键继续...\n");
      getchar();
      getchar();
      break;
    case '2':search();
      break;
    case '3':
      system("cls");
      narrate();
      x[0]=1;
      HaMiTonian(1);
      printf("\n\n\t\t\t\t 请按任意键继续...\n");
      getchar();
      getchar();
      break;
    };
  }while(ck!='e');


}
char Menu()    /*  主菜单  */
{
  char c;
```

```
 int flag;
 do{
  flag=1;
  system("cls");
  narrate();
  printf("\n\t\t\t ┌──────────────────────────────┐ \n");
  printf("\t\t\t │                              │ \n");
  printf("\t\t\t │        1、查询建筑路径       │ \n");
  printf("\t\t\t │        2、查询建筑信息       │ \n");
  printf("\t\t\t │        3、选择出发点和目的地 │ \n");
  printf("\t\t\t │        e、退出               │ \n");
  printf("\t\t\t │                              │ \n");
  printf("\t\t\t └──────────────────────────────┘ \n");
  printf("\t\t\t\t 请输入您的选择： ");
  scanf("%c",&c);
  if(c=='1'||c=='2'||c=='3'||c=='e')
    flag=0;
 }while(flag);
 return c;
}
```

主要教学环节的组织：

1、指导教师说明实验目的，布置实验项目。

2、学生上机编程实践。

3、指导教师验收学生的程序，简单评价并给出建议，必要情况下给出指导。

4、学生对实验进行总结，整理相关材料，完成实验报告。

思考题：

比较 Dijkstra 的最短路径算法与 Floyd 算法在问题求解中的效率。

实验七 查找算法的设计与实现

本次实验的主要目的在于帮助读者熟练掌握动态查找表的构造和查找方法。熟悉哈希表的构造方法，深刻理解哈希表与其他结构的表的实质性的差别。掌握描述查找过程的判定树的构造方法。在此基础上，有能力设计复杂的应用程序。

【问题描述】

用 C 或 C++语言设计并实现动态查找表中的二叉排序树的建立与查找，包括新结点的插入和删除等操作。通过哈希表的构造、查找和维护，能够对全年级的学生进行按姓名的哈希查找。

【基本要求】

1. 实现动态查找表的三种基本功能：查找、插入和删除
2. 针对全年级中的“人名”设计一个哈希表，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数量>=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。

【CDIO项目要求】

11.有完整的CDIO四个阶段描述

12.有友好美观的操作界面

13.有软件使用说明或帮助文档

4.项目成员分工明确，团结协作

【实验内容】

难度 A：设计一个可进行动态查找表的二叉排序树的建立与查找的演示程序，包括新结点的插入和删除等操作；个人完成，评分最高70分。

难度B：实现能够对全年级的学生进行按姓名的哈希查找，完成相应的建表和查表程序；个人完成，评分最高至90分。

难度C：设计并实现一个校园内的FTP搜索引擎，能够实现查找表的各种实现方法：顺序表、有序表、树表和哈希表。由2-3个人的团队完成，评分最高可至100分。

【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。字符的取吗法可直接利用C语言中的toascii函数，并可对过长的人名先作折叠处理。

（1）二叉排序树的查找算法及算法原理

对给定的二叉排序树，如果想知道某元素是否在其中出现，可以和根结点比较，如果相等结束；如果不等，若比其大，进入右子树；否则进入左子树；继续按照上面的方法，直到出现相等或者到某分支结束为止，返回查找信息。

程序示例：

```
    BTNode   *BST_search(BTNode *t ,Elemtype k)
{   BTNode   *p ,*f=NULL;
    while(p!=NULL)
    {  if(k==p->key)
Else  if(k<p->key)
       {   f=p ;  p=p->lchild ;   }
       else
```

```
        {    f=p ;   p=p->rchild ;    }
}
    return NULL ;
} /* BST_search */
```

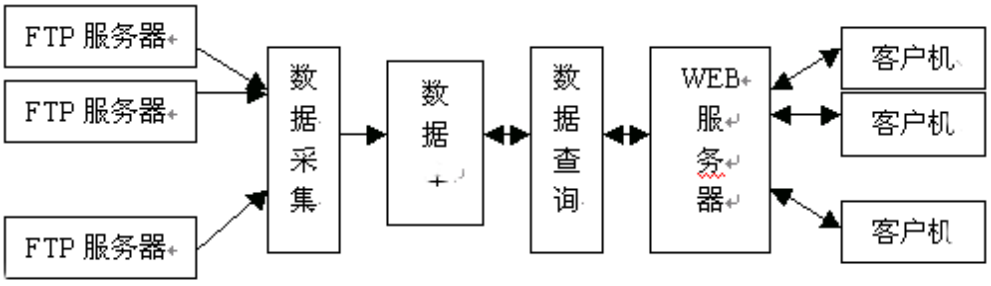
（2）哈希函数的构造方法

哈希函数的构造方法很多，通常根据实际需要，遵循使关键字通过哈希函数转换所得到的地址尽可能地均匀分布在给定空间中的原则。因此，如何构造一个“好”的哈希函数就是带有很强的技术性和实践性的问题。常用的构造哈希函数的方法有：直接定址法、数字分析法、平方取中法、除留余数法，折叠移位法等等，解决冲突的方法主要有开地址法和链地址法。总而言之，构造哈希函数的方法可以多种多样，但以哈希地址分布均匀为优。

（3）FTP搜索引擎

FTP是因特网最主要的服务之一，在FTP服务器上保存有大量的各种各样的共享软件、技术资料和多媒体数据等文件。目前，国内外有很多FTP搜索引擎，国内较著名的有北大天网、百合谷搜索和FTP星空搜索等。

FTP搜索引擎由数据采集、数据查询和站点维护等模块组成。实现一个FTP搜索引擎，首先要收集各个FTP站点上的文件信息，并把这些信息存储到数据库中；然后给用户提供一个查询界面，以收取用户要查询的信息，把这些查询信息转化为数据库语言，并进行数据库查询，把查询结果以友好的界面显示给用户；搜索引擎建立好以后，为了使数据库数据与FTP站点的数据保持一致，需要更新FTP站点的文件信息，添加新的FTP站点等管理和维护。其结构如下图所示。



主要教学环节的组织：

1、指导教师说明实验目的，布置实验项目。

2、学生上机编程实践。

3、指导教师验收学生的程序，简单评价并给出建议，必要情况下给出指导。

4、学生对实验进行总结，整理相关材料，完成实验报告。

思考题：

你所设计的哈希表针对实验给出的问题，选用哪种方法解决冲突比较合适，为什么？

五．教科书（实验指导书）

严蔚敏,吴伟民．**数据结构题集（C 语言版）**[M]．北京：清华大学出版社，

2003.（清华大学计算机系列教材）

参考资料：

[1] 严蔚敏,吴伟民. **数据结构 C 语言版**[M]. 北京：清华大学出版社，2002 年 9 月（清华大学计算机系列教材）.

[2] Mark Allen Weiss. **Data Structures and Algorithm Analysis in C** [M].北京：人民邮电出版社，2005 年.

[3] Robert Sedgewick. **算法 I~IV（C++实现）— 基础、数据结构、排序和搜索（第三版 英文影印版）**[M]. 北京：高等教育出版社，2002 年.

[4] Robert Sedgewick. **算法 V（C++实现）— 图算法（第三版 英文影印版）**[M]. 北京：高等教育出版社，2002 年.