《数据结构课程设计》

任务书

适用专业：

计算机科学与技术

软件工程

网络工程

信息管理与信息系统

物联网工程

电子信息与电气技术实验中心

2015 年 2 月

一、课程设计的目的

《算法与数据结构》是计算机专业的核心课程，是一门实践性很强的课程。为了学好这门课程，必须在掌握理论知识的同时，加强上机实践。针对数据结构的课程设计不仅可以加深对课程内容的理解，并且可以通过实践进一步掌握程序设计的技能与方法，学会数据的组织方法和现实世界问题在计算机内部的表示方法，并针对问题的应用背景分析，选择最佳的数据结构和算法。同时通过课程设计，要求学生在完成程序设计的同时能够写出比较规范的设计报告，初步感受软件开发过程的项目管理方法和规范，为进一步学习打下基础。

二、课程设计题目：见附录 A

三、设计要求

1、每人**至少选择一题完成，每道题每个班选择人数不能超过 2 人。**

2、独立思考，独立完成

课程设计中各任务的设计和调试要求独立完成，遇到问题可以讨论，**但不可以拷贝，不允许雷同**，选同一题的同学原则上同一指导老师答辩。

3、求解每个题目时，要求从分析题目的需求入手，按设计抽象数据类型、构思算法、实现抽象数据类型、编制上机程序和上机调试等若干步骤完成题目，最终写出完整的课程设计报告。前期准备工作完备与否直接影响到后序上机调试工作的效率。在程序设计阶段应尽量利用已有的标准函数，加大代码的重用率。

4、对于所用到的基本数据结构如栈、队列、链表，应事先建好相应的函数库，参考附录 C。

4、设计出的系统要有一个易于使用的人机界面。

5、源程序中应对重要程序写出注释语句。

四、课程设计作品

1. 问题求解的源程序。
2. 课程设计报告（电子稿），文档书写格式可参看附录 B。

每个人在课程设计开始时，要先登录课程平台 cms.fjut.edu.cn，选课，在课程平台上有

课程设计任务书和课程设计报告提交的链接，课程设计结束时，提交课程设计报告和源程序

五、课程设计考核

考核方式：演示和答辩相结合

评分标准：

系统分析与设计 60%

报告撰写 30%

答辩 10%

成绩按五分制评定：优、良、中、及格和不及格。

六、时间安排

一般是在春季学期的第 17 周到 18 周，时间为 1 周。

第一天 1、2 节集中辅导，指导老师讲解课程设计的目的和任务和要求。介绍课程设计的题目。然后到指定的机房上机实践。每天上机 6 小时。最后一天，逐个开始答辩。

附录 A：题目

注：题目中的星号表示题目的难度，星号越多难度越大。

1.哥德巴赫猜想问题**

问题描述：

哥德巴赫猜想问题：任何一个大于 2 的偶数均可表示为两个素数之和

任务：计算给定的大偶数可以表示为多少对素数之和，例如 10：3，7 和 5,5

大偶数由 input.txt 文件输入，输出存储到 output.txt 文件中

2、亲兄弟问题(栈)

问题描述：给定 n 个整数 a0….an-1 组成的序列。序列中任意元素 ai 的亲兄弟元素 ak 定义为右边最靠近它且不小于它的元素，即 k=min{j|aj>=ai},如没有，则亲兄弟位置为-1，用栈设

计 O(n)的时间算法，计算相应的亲兄弟位置

例如 n=10,整数序列为 6,1,4,3,6,2,4,7,3,5 时，相应亲兄弟位置序列为 4,2,4,4,7,6,7，-1,9，-1

3、（队列）猴子分桃问题

问题描述：动物园里的 n 只猴子编号为 1,2，…n,依次排成一队等待饲养员按规则分桃。动物园的分桃规则是每只猴子可分的 m 个桃子，但必须排队领取。饲养员循环地每次取出 1 个，2 个，k 个放入筐中，由排在对首的猴子领取。取到筐中桃子树为 k 后，又重新从 1 开始。当框中桃子树加上队首猴子已取的桃子数不超过 m 时，对首的猴子可以全部取出筐中的桃子，取的桃子总数不足 m 个猴子，继续道队尾等候。当筐中桃子树加上对首猴子已取的桃子数超过 m 时，对首的猴子只能取满 m 个，然后离开队列，筐中剩余的桃子由下一只猴子取用。上述分桃过程一直进行到每只猴子都分到 m 个桃子。

完成任务：对于给定的 n,k 和 m，模拟上述猴子分桃过程。

4、最优服务次序问题

问题描述：设有 n 个顾客同时等待一项服务。顾客 i 需要的服务时间为 ti,i<i<n.应如何安排 n 个顾客的服务次序，才能使平均等待时间达到最小？平均等待时间是 n 个顾客等待服务时间的总和除以 n.

5、最近公共祖先问题

问题描述：设计一个算法，对于给定的树中两个结点返回它们的最近公共祖先。

6. 基本线性表的就地逆置*

在基本线性表原有空间的基础上，将线性表中的数据元素逆置，使新的顺序序列与原来的顺序序列刚好相反。如原来顺序序列“abcdef”，逆置之后的新顺序序列为”fedcba”。

基本要求：利用顺序结构和链式结构分别实现；操作过程必须在线性表的原有空间，不能借助临时变量所申请的临时空间，也不能借助其他形式的临时空间。

7.链表存储结构实现简单排序算法**

实现直接插入、冒泡排序、简单选择的排序算法。

基本要求： 待排序表的表长为 20000；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(包含正序、逆序、基本有序、随机)比较；比较的指标为有关键字参

加的比较次数和关键字移动次数（关键字交换计为 3 次移动）

8.排序算法的实现***

实现折半插入、快速排序、堆排序的排序算法。

基本要求： 待排序表的表长为 20000；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(包含正序、逆序、基本有序、随机)比较；比较的指标为有关键字参加的比较次数和关键字移动次数（关键字交换计为 3 次移动）

9.二叉排序树的创建**

输入任意的数列创建二叉排序树，并进行先序、中序、后序和层次遍历。

基本要求：存储结构利用二叉链表

10. 0/1 背包问题：题目：有 N 件物品和一个容量为 V 的背包。第 i 件物品的重量是 c[i]，价值是 w[i]。求解将哪些物品装入背包可使这些物品的重量总和不超过背包容量，且价值总和最大。这是最基础的背包问题，特点是：每种物品仅有一件，可以选择放或不放。

11.哈希表***

针对同班同学信息设计一个通讯录，学生信息有姓名，学号，电话号码等。以学生姓名为关键字设计哈希表，并完成相应的建表和查表程序。

基本要求：姓名以汉语拼音形式，待填入哈希表的人名约 30 个，自行设计哈希函数和冲突处理方法；在查找的过程中给出比较的次数。完成按姓名查询的操作。

要求：实现信息的增、删、改。将初始班级的通讯录信息存入文件

12 .校园导游程序***

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）)设计学校的旗山校区北区校园平面图，所含场所不少于 10 个。以图中顶点表示校内各场所，存放场所名称、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场所相关信息的查询。

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

要求：实现场所和路径的增加、删除。数据的保存、调入。

13.航空客运订票系统***

通过此系统可以实现如下功能：

录入：可以录入航班情况（数据可以存储在一个数据文件中，数据结构、具体数据自定）；

查询： 可以查询某个航线的情况（如，输入航班号，查询起降时间，起飞抵达城市，航班票价，票价折扣，确定航班是否满仓）； 可以输入起飞抵达城市，查询飞机航班情况；

订票：（订票情况可以存在一个数据文件中，结构自己设定） 可以订票，如果该航班已经无票，可以提供相关可选择航班；

退票： 可退票，退票后修改相关数据文件； 客户资料有姓名，证件号，订票数量及航班情况，订单要有编号。

修改航班信息： 当航班信息改变可以修改航班数据文件

要求： 根据以上功能说明，设计航班信息，订票信息的存储结构，数据的存盘和调入，设计程序完成功能；

14.哈夫曼编码和译码****

利用哈夫曼编码进行信息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。试为这样的信息收发站写一个哈夫曼编/译码系统。

基本要求:一个完整的系统应具有以下功能：

（1）初始化（Initialization）。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，(选做：并将它存于文件 hfmTree 中)。并显示出每个字符的编码。

（2）编码（Encoding）。利用已建好的哈夫曼树（选做：如不在内存，则从文件 htmTree 中读入），对输入的字符串文本（选做：对文件 ToBeTran 中的正文）进行编码，（选做：然后将结果存入文件 CodeFile 中。）并显示在屏幕上。

（3）译码（Decoding）。利用已建好的哈夫曼树将输入的代码进行译码（选做：将文件 CodeFile 中的代码进行译码，结果存入文件 TextFile 中。），并显示在屏幕上。

（4）打印哈夫曼树（Tree Printing）。将已在内存中的哈夫曼树以直观的方式显示在屏幕上。

15.链表的基本操作**

1. 掌握线性链表的建立。

2．掌握线性链表的基本操作。

二、设计内容和要求

利作链表的插入运算建立线性链表，然后利用链表的查找、删除、计数、输出等运算反复实现链表的这些操作（插入、删除、查找、计数、输出单独写成函数的形式），并能在屏幕上输出操作前后的结果。

16.一元稀疏多项式计算器***

基本功能定为

(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,c1,e1,c2,e2,.....,Cn,en,其中 n 是多项式的相数,Ci 和 Ei 分别是第 i 项的系数和指数,序列按指数降序排列

(3) 两个多项式相加，建立并输出和多项式

(4) 两个多项式相减，建立并输出差多项式

(5) 两个多项式相乘, 建立乘积多项式

(6) 计算多项式在 x 处的值

17.宿舍管理查询软件***

任务：为宿舍管理人员编写一个宿舍管理查询软件, 程序设计要求：采用交互工作方式；建立数据文件 ，数据文件按关键字（姓名、学号、房号）进行排序(冒泡、选择、插入排序等任选一种)

查询菜单:（用不同的查找方法实现）

按姓名查询

按学号查询

按房号查询

18.设计一个计算机管理系统完成图书管理基本业务***

基本要求：

每种书的登记内容包括书号、书名、著作者、现存量和库存量；

对书号建立索引表（线性表）以提高查找效率；

a)系统主要功能如下：采编入库：新购一种书，确定书号后，登记到图书帐目表中，如果表中已有，则只将库存量增加；

b)借阅：如果一种书的现存量大于 0，则借出一本，登记借阅者的书证号和归还期限，改变现存量；

c)归还：注销对借阅者的登记，改变该书的现存量。

19.学生成绩管理系统**

现有学生成绩信息文件 1（1.txt），内容如下

姓名	学号	语文	数学	英语
张明明	01	67	78	82
李成友	02	78	91	88
张辉灿	03	68	82	56
王露	04	56	45	77
陈东明	05	67	38	47
….	..	..	..	…

学生成绩信息文件 2（2.txt）,内容如下:

姓名	学号	语文	数学	英语
陈果	31	57	68	82
李华明	32	88	90	68
张明东	33	48	42	56
李明国	34	50	45	87
陈道亮	35	47	58	77
….	..	..	..	…

试编写一管理系统,要求如下:

1、 实现对两个文件数据进行合并,生成新文件 3.txt

2、 抽取出三科成绩中有补考的学生并保存在一个新文件 4.txt

3、 对合并后的文件 3.txt 中的数据按总分降序排序(至少采用两种排序方法实现)

4、 输入一个学生姓名后,能查找到此学生的信息并输出结果(至少采用两种查找方法实现)

5、 要求使用结构体,链或数组等实现上述要求.

20.教学计划安排检验程序（拓扑排序）****

本次课程设计的任务是：针对学院的计算机系本科课程，根据课程之间的依赖关系，制定课程安排计划，并满足各学期课程数大致相同。按照用户输入的课程数，学期数，课程间的先后关系数目以及课程间两两间的先后关系，程序执行后会给出每学期应学的课程。

(1) 输入的形式和输入值的范围：输入间用空格隔开。要求用户输入的课程数小于 20，学期数小于或是等于 8，课程名的长度小于等于 10 个字符。

(2) 程序所能达到的功能：按照用户的输入，给出每学期应学的课程。

(4) 测试数据：输入：学期数：5，课程数：12，课程间的先后关系数：16，课程的代表值：v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12。课程间两两间的先后关系：v1 v2,v1 v3, v1 v4,v1 v12,v2 v3,v3 v5,v3 v7,v3 v8,v4 v5, v5 v7,v6 v8,v9 v10, v9 v11 , v9 v12,v10 v12,v11 v6

输出：第 1 学期应学的课程：v1 v9

第 2 学期应学的课程：v2 v4 v10 v11

第 3 学期应学的课程：v3 v6 v12

第 4 学期应学的课程：v5 v8

第 5 学期应学的课程：v7

21.停车场问题***

停车场是一条可以停放 n 辆车的狭窄通道，且只有一个大门汽车停放安到达时间的先后依次由北向南排列（大门在最南端，最先到达的第一辆车停在最北端）若停车场已经停满 n 辆车，后来的汽车在便道上等候，一旦有车开走，排在便道上的第一辆车可以开入；当停车场的某辆车要离开时，停在他后面的车要先后退为他让路，等它开出后其他车在按照原次序开入车场，每两停在车场的车要安时间长短缴费。 要求：以栈模拟停车场，以队列车场外的便道，按照从终端输入的数据序列进行模拟管理。每一组数据包括三个数据项：汽车“到达”或“离去”信息、汽车牌照号码、以及到达或离去的时刻。对每一组数据进行操作后的信息为：若是车辆到达，则输出汽车在停车场的内或便道上的位置：若是车辆离去则输出汽车在停车场内的停留时间和应缴纳的费用（在便道上的停留时间不收费）。栈以顺序结构实现，队列以链表结构实现。

22.括号匹配情况***

假设在一个算术表达式中，可以包含三种括号：圆括号"（"和"）"，方括号"["和"]"和花括号"{"和"}"，并且这三种括号可以按任意的次序嵌套使用。比

如，...[...{...}...[...]...]...[...]...(...)..。现在需要设计一个算法，用来检验在输入的算术表达式中所使用括号的合法性。

算术表达式中各种括号的使用规则为：出现左括号，必有相应的右括号与之匹配，并且每对括号之间可以嵌套，但不能出现交叉情况。我们可以利用一个栈结构保存每个出现的左括号，当遇到右括号时，从栈中弹出左括号，检验匹配情况。在检验过程中，若遇到以下几种情况之一，就可以得出括号不匹配的结论。（1）当遇到某一个右括号时，栈已空，说明到目前为止，右括号多于左括号；（2）从栈中弹出的左括号与当前检验的右括号类型不同，说明出现了括号交叉情况；（3）算术表达式输入完毕，但栈中还有没有匹配的左括号，说明左括号多于右括号。要求用栈完成。

23.商品货架管理***

商店货架以栈的方式摆放商品。商品货架可以看成一个栈，栈顶商品的生产日期最早，栈底商品的生产日期最近。生产日期越接近的越靠栈底，出货时从栈顶取货。一天营业结束，如果货架不满，则需上货。入货直接将商品摆放到货架上，则会使生产日期越近的商品越靠近栈顶。这样就需要倒货架，使生产日期越近的越靠近栈底。请编写程序模拟商品销售，上架倒货架等操作。（设有 5 种商品，每种商品至少有商品名和生产日期两个属性）

25.稀疏矩阵的快速转置**

利用三元组表存储稀疏矩阵，利用快速转置算法进行转置，并输出转置之前和之后的三元组表以及矩阵。

26.寻找最近点对

平面中有若干个点，寻找距离最近的两个点。

27. Strassen 矩阵乘法计算

将方程 C=AB 重写为:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \quad (1)$$

由此可得:

$C_{11}=A_{11}B_{11}+A_{12}B_{21}$ (2)

$C_{12}=A_{11}B_{12}+A_{12}B_{22}$ (3)

$C_{21}=A_{21}B_{11}+A_{22}B_{21}$ (4)

$C_{22}=A_{21}B_{12}+A_{22}B_{22}$ (5)

如果 n=2，则 2 个 2 阶方阵的乘积可以直接用(2)-(3)式计算出来，共需 8 次乘法和 4 次加法。当子矩阵的阶大于 2 时，为求 2 个子矩阵的积，可以继续将子矩阵分块，直到子矩阵的阶降为 2。这样，就产生了一个分治降阶的递归算法。依此算法，计算 2 个 n 阶方阵的乘积转化为计算 8 个 n/2 阶方阵的乘积和 4 个 n/2 阶方阵的加法。2 个 n/2×n/2 矩阵的加法显然可以在 $c*n^2/4$ 时间内完成，这里 c 是一个常数。因此，上述分治法的计算时间耗费 T(n)应该满足：

$$T(n)=8T(n/2)+cn^2, n>2$$

这个递归方程的解仍然是 $T(n)=O(n^3)$。因此，该方法并不比用原始定义直接计算更有效。究其原因，乃是由于式(2)-(5)并没有减少矩阵的乘法次数。而矩阵乘法耗费的时间要比矩阵加减法耗费的时间多得多。要想改进矩阵乘法的计算时间复杂性，必须减少子矩阵乘法运算的次数。按照上述分治法的思想可以看出，要想减少乘法运算次数，关键在于计算 2 个 2 阶方阵的乘积时，能否用少于 8 次的乘法运算。Strassen 提出了一种新的算法来计算 2 个 2 阶方阵的乘积。他的算法只用了 7 次乘法运算，但增加了加、减法的运算次数。这 7 次乘法是:

$M_1=A_{11}(B_{12}-B_{22})$

$M_2=(A_{11}+A_{12})B_{22}$

$M_3=(A_{21}+A_{22})B_{11}$

$M_4=A_{22}(B_{21}-B_{11})$

$M_5=(A_{11}+A_{22})(B_{11}+B_{22})$

$M_6=(A_{12}-A_{22})(B_{21}+B_{22})$

$M_7=(A_{11}-A_{21})(B_{11}+B_{12})$

做了这 7 次乘法后，再做若干次加、减法就可以得到:

$C_{11}=M_5+M_4-M_2+M_6$

$C_{12}=M_1+M_2$

$C_{21}=M_3+M_4$

$C_{22}=M_5+M_1-M_3-M_7$

以上计算的正确性很容易验证。例如:

$C_{22}=M_5+M_1-M_3-M_7$

$=(A_{11}+A_{22})(B_{11}+B_{22})+A_{11}(B_{12}-B_{22})-(A_{21}+A_{22})B_{11}-(A_{11}-A_{21})(B_{11}+B_{12})$

$=A_{11}B_{11}+A_{11}B_{22}+A_{22}B_{11}+A_{22}B_{22}+A_{11}B_{12}$

$-A_{11}B_{22}-A_{21}B_{11}-A_{22}B_{11}-A_{11}B_{11}-A_{11}B_{12}+A_{21}B_{11}+A_{21}B_{12}$

$=A_{21}B_{12}+A_{22}B_{22}$

附录 B:报告模板

福　建　工　程　学　院

数据结构课程设计

课　程：_____

题　目：_____

专　业：_____

班　级：_____

座　号：_____

姓　名：_____

年　月　日

课程设计题目：求迷宫的最短路径

一、要解决的问题

这是实验心理学中的一个经典问题，心理学家把一只老鼠从一个无顶盖的大盒子的入口处赶进迷宫。迷宫中设置很多隔壁，对前进方向形成了多处障碍，心理学家在迷宫的唯一出口处放置了一块奶酪，吸引老鼠在迷宫中寻找通路以达到出口。我们要解决的是如何找到了一条迷宫的最短路径。

二、算法基本思想描述：

要用到回溯的思想。从迷宫入口点出发，向四周搜索，记下所有一步能到达的坐标点；然后依次从这些点出发，再记下所有一步能到达的坐标点，.....依此类推，直到到达迷宫的出口点为止，然后从迷宫出口点沿搜索路径回溯。这样就找到了一条迷宫的最短路径，否则迷宫无路径。由于先到达的点先搜索，故用先进先出的数据结构——队列来保存已到达的坐标点。

三、设计

1. 数据结构设计

（1）迷宫的表示

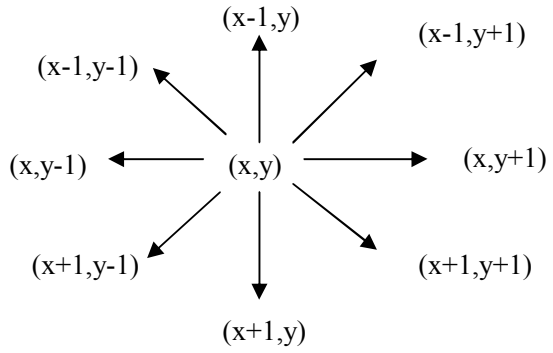
设迷宫为 m 行 n 列，利用 maze[m][n]来表示迷宫，maze[m][n]=0 或 1，其中 0 表示通路，1 表示不通。入口坐标（1，1），出口坐标（m,n）.

迷宫定义如下：

```
#define  m  6
#define  n  8
int maze[m+2][n+2];
```

（2）试探方向的表示

在迷宫中有 8 个方向可以试探，规定：从当前位置向前试探的方向为从正东开始沿顺时针方向进行。为了简化问题，将这 8 个方向的坐标增量放在一个结构数组 move[8]中。在 move 数组中，每个元素有两个域组成，X：横坐标增量；Y：纵坐标增量。



序号	X	Y
0	0	1
1	1	1
2	1	0
3	1	-1
4	0	-1
5	-1	-1
6	-1	0
7	-1	1

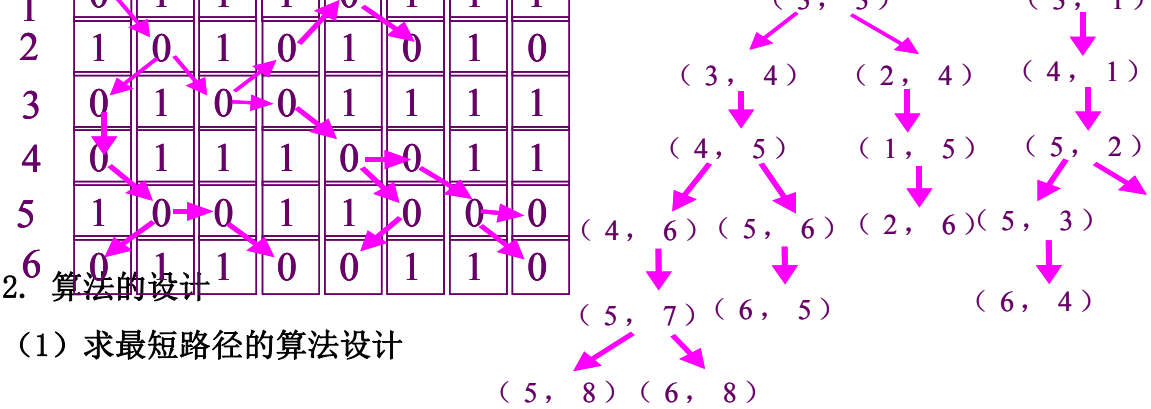
Move 数组定义如下：

```
typedef struct
 {int x,y;}item;
item  move[8]={ {0,1},  {1,1},  {1,0},{1,-1},{0,-1},{-1,-1},{-1,0},{-1,1} };
```

（3）队列的表示

在找到出口点之后，需要沿搜索路径回溯，所以到达某点时，不仅要记下该点的坐标，还要记下该点的前驱。用一个结构数组 sq[num]作为队列的存储空间。Sq 的每一个结构有三个域：x,y,pre,其中 x，y 分别为所到达的点的坐标，pre 为前驱点的坐标。还设队头 front 和队尾 rear 指针。

```
#define num 50
typedef struct
  {int x,y;
   int  pre;
  }SqType;
SqType  sq[num];
int  front，rear;
```



2. 算法的设计

（1）求最短路径的算法设计

初始状态，队列中只有一个元素 sq[1],记录的是入口点的坐标（1，1），因为该点是出发点，因此没有直接前驱点，pre 域为-1，队头指针 front 的队尾指针 rear 均指向它，此后搜索时都是以 front 所指点为搜索的出发点，即将该点的坐标及 front 所指点的位置入队，这样不但记下了到达点的坐标，还记下了它的前驱点。Front 所指向点的 8 个方向搜索完毕后，则出队，继续对下一点搜索。搜索过程中遇到出口点则成功，搜索结束，打印出迷宫的最短路径，算法结束；或者当前队空，既没有搜索点了，表明没有路径，算法也结束。

（2）防止重复到达某点的考虑

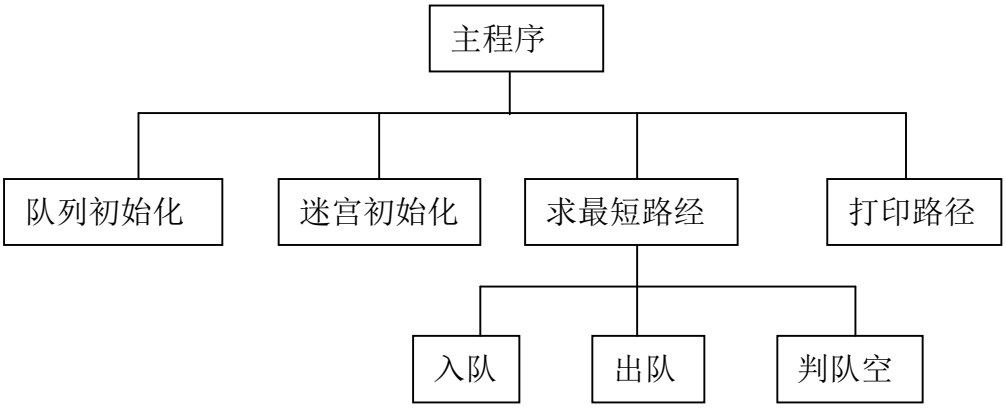
为避免发生死循环，当到达某点（i，j）后，使 maze[i][j]置-1，以便区分未到达过的顶点。算法结束前可恢复原迷宫。

（3）队列头、尾指针的指向

队头指针指向搜索的出发点，当找到一个可到达点，就入队；当 8 个方位都搜索完毕，队头指针往后移一个（出队，但原位置值依然存在，方便最后回溯）。

（4）模块结构及功能：

a) void main() //主程序
b) viod init_maze(int) //迷宫的初始化
c) void init_queue(SqType) //队列的初始化
d) int path(int,int) //求迷宫的最短路径
e) void print_path(SqType,rear) //打印路径
f) void in_queue(SqType ,datatype) //入队操作
g) void out_queue(SqType) //出队操作
h) int empty_queue(SqType) //判队空



（5）主要模块算法描述

求迷宫最短路径的算法描述：

```
path(int maze,int move)
    {
      ①队列头、尾指针初始化（=-1）;
      ②将入口点的前驱设置为-1，入队；
      ③将入口点设置为已走过；
      ④将是否找到出口点信息 found 赋值为 0（未找到）;
      ⑤ while（未找到&&队列非空）
         {
             队头指针后移指向当前搜索点，并将它赋值给 x;
             for  i=1  to  8 搜索 x 点的 8 个方向
                    if（找到一个可走点）
                    {
                           就将该点坐标及前驱值（front）入队;
                           将该点设置为已走过;
                           if（该点是出口点）found=1;
                    }
       }
      ⑥if(找到) 回溯打印最短路径;
             else 打印“该迷宫没有路径”;
```

四、源程序清单：（略）

五、测试数据及测试结果：

例如：

测试数据：

maze[m+2][n+2]={{1,1,1,1,1,1,1,1,1,1},{1,0,1,1,1,0,1,1,1,1},{1,1,0,1,0,1,1,1,1,1},
{1,0,1,0,0,0,0,0,1,1},{1,0,1,1,1,0,1,1,1,1},{1,1,0,0,1,1,0,0,0,1},
{1,0,1,1,0,0,1,1,0,1},{1,1,1,1,1,1,1,1,1,1}};

测试结果：

(6,8)<— (5,7)<—(4,6)<—(4,5)<—(3,4)<—(3,3)<— (2,2)<— (1,1)

六、课程设计总结及心得体会：

可以包括：课程设计过程的收获、遇到的问题、解决遇到问题过程中的思考、程序调试能力的思考、对问题求解方案数据结构重要性的认识和体会。

附录 C 构造自己的库

C/C++的库(library)是一些函数的集合，若干个 C/C++的关于某个方面的函数源程序文件被编译成目标文件之后，再打包在一起，作为一个工具库供本人或他人日后使用。这种库有静态库和动态库之分，静态库通常以 libxxx.a 或 libxxx.lib 命名，动态库在 MS Windows 下以.dll 为扩展名，在 Unix/Linux 下通常以.so 为扩展名(称为共享库)。静态库和动态库的差别在于它们是如何被调用的。应用程序使用静态库在链接阶段把要用的库链接到一起形成可执行文件，而使用动态库或共享库，是在运行时才动态的把库加载到内存中以备调用。从静态库和动态库的调用特点不难看出，静态库是未经链接的目标程序，是不可执行的，只有与应用程序的编译结果链接起来之后才能成为可执行文件。而动态库/共享库是经过编译链接产生的，是可执行的模块，应用程序编译链接时没有真正链接所需的函数模块，而是执行的时候才调用需要的模块。

Code::Blocks 提供了建立静态库和动态库的工程框架，当建立了相应的工程之后，就可以在工程中加入需要在库中包含的函数，经编译链接之后就可以建立库。

1. 静态库的建立和使用

选择 Code::Blocks 的 file 菜单→new→project, 在工程模板对话框中选择 static library 图标，点击之后按照向导的指引提供工程的名字和存放的目录之后就 ok 了。例如我建的工程名叫 testlib，存储在 C:\的根目录下，建好工程之后，在工程中自动包含一个 main.c 文件，其中包含了几个示范函数的定义：

```
#001 // A function adding two integers and returning the result
#002 int SampleAddInt(int i1, int i2)
#003 {
#004     return i1 + i2;
#005 }
#006 // A function doing nothing ;)
#007 void SampleFunction1()
#008 {
#009     // insert code here
#010 }
#011 // A function always returning zero
#012 int SampleFunction2()
#013 {
#014     // insert code here
#015     return 0;
#016 }
```

修改或者替换这些函数，或者移除它，添加自己的函数模块文件。下面是我在 testlib 中建立的两个函数模块文件：

//main.c

```
#001 #include<stdio.h>
#002 int AddInt(int i1, int i2)
#003 {
#004     return i1 + i2;
#005 }
#006 int Substract(int i1, int i2)
#007 {
#008     return i1 - i2;
#009 }
#010 void Hello()
```

```
#011 {
#012    printf("Hello\n");
#013 }
```

//others.c

```
#001 int sumsN(int n)
#002 {
#003    int i,s=0;
#004    for (i=1;i<=n;i++)
#005        s+=i;
#006    return s;
#007 }
```

然后 build 这个工程就会自动产生包含这些函数的库文件 libtestlib.a。注意 testlib 工程只可以编译，不可以运行的。为了能构使用 libtestlib.a 库还要建立库函数的头文件：

//testlib.h

```
#001 #ifndef TESTLIB_H
#002 #define TESTLIB_H
#003    int sumsN(int n);
#004    int AddInt(int i1, int i2);
#005    int Substract(int i1, int i2);
#006    void Hello();
#007 #endif // TESTLIB_H
```

下面测试一下 libtestlib.a 库。建立一个控制台应用程序工程 testtestlib，并建立 test.c：

```
#001 #include <stdio.h>
#002 #include <stdlib.h>
#003 #include "testlib.h"
#004 int main(void){
#006     Hello();
#007     printf("%d\n",sumsN(10));
#008     printf("%d\n",AddInt(3,5));
#009     printf("%d\n",Substract(3,5));
#010     return 0;
#011 }
```

这时如果对 test.c 进行编译，肯定会报错，错误信息是找不到 testlib.h。因此必须对这个测试工程做适当的设置，才能正确的找到所需的头文件。选择 project 菜单中的 build options，点击 Search directories，在 **Compiler** 页面中点击 Add 按钮，增加库头文件 testlib.h 所在的目录 testlib，如图 2.2.4。这时再 build 还会报错，错误信息是

```
undefined reference to `Hello()'
undefined reference to `sumsN(int)'
undefined reference to `AddInt(int, int)'
undefined reference to `Substract(int, int)'
```

意思是那些函数没有定义。还要告诉编译器链接的库是什么。同样是选择 project 菜单中的 build options，但这次是选 **Linker settings**，点击 add 按钮，这时会弹出文件选择对话框，然后去找到 libtestlib.a 文件所在的位置，点击确定，把 libtestlib.a 添加进去即可。

2. 动态库的建立和使用

与建立静态库工程类似，选择 Code::Blocks 的 file 菜单→new→project, 在工程模板对话框中选择 dynamic link library 的图标，点击之后按照向导的指引提供工程的名字和存放的目录之后动态链接库的工程就建立起来了。例如我建的工程名叫 testdll，存储在 C:\的根目录下。需要注意的是动态链接库的工程中**默认包含一个 C++源程序**，即扩展名为.cpp 的源程序文件，和一个包含函数原型的.h 头文件。但在.h 头文件中却把函数的原型放在了 extern “C” 的框架之内了：

```
#ifdef __cplusplus
extern "C"{
   int DLL_EXPORT get_id(void);
   int DLL_EXPORT add(int,int);
}
#endif
```

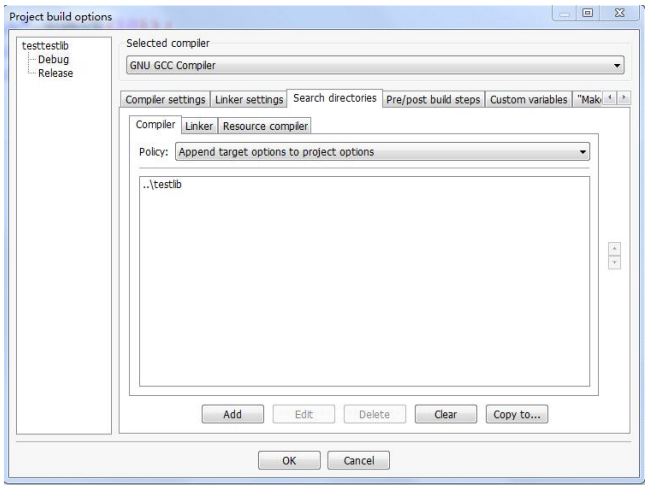


图2.2.4 工程的头文件设置

这说明，如果使用了 C++编译器，要把函数库中的函数作为 C 语言代码来处理。如果不用 C++编译器就可以把默认的头文件和源文件删除替换成.c 相关的文件。工程 testdll 是 C 语言版本的动态库，其中的.h 文件 testdll.h 和.c 源程序文件 testdll.c 分别如下：

//testdll.h:

```
#ifndef TESTDLL_H
#define TESTDLL_H
#ifdef BUILD_DLL
    #define DLL_EXPORT __declspec(dllexport)
#else
    #define DLL_EXPORT __declspec(dllimport)
#endif
int DLL_EXPORT get_id(void);
int DLL_EXPORT add(int,int);
#endif //TESTDLL_H
```

注意在函数原型外部没有使用 extern “C” { … }。文件中的__declspec(dllexport) 是告诉编译器用它修饰的函数是用于输出的，__declspec(dllimport)是告诉编译器用它修饰的函数是用于输入的。**输出是相对于库而言，输入是相对于使用库的应用程序而言**，因此 DLL_EXPORT 的取值是输入还是输出由是否定义了 BUILD_DLL 来决定，当定义了 BUILD_DLL 则 DLL_EXPORT 用于输出，否则用于输入。所以在下面的库源程序文件中，#include “testdll.h”之前定义了 BUILD_DLL，这样 testdll.h 中的函数就是输出的库函数。

//testdll.c

```
#define BUILD_DLL 1
#include "testdll.h"
int DLL_EXPORT get_id(void)
{
    return 10;
}
int DLL_EXPORT add(int x,int y)
{
    return x+y;
}
```

编译链接 testdll 工程将产生三个文件：

```
    libtestdll.a, libtestdll.def, testdll.dll
```

其中 testdll.dll 就是动态链接库，libtestdll.a 中包含了 dll 库的输出符号列表，是使用 dll 库的应用程序链接时必须的一个文件，如果没有这个文件所用的函数就不知道在哪里。libtestdll.def 是一个文本文件

```
EXPORTS
   SomeFunction @1
   add @2
```

在这个文件中自动生成了库输出函数的函数名单，它的作用如同__declspec(dllexport)。如果

库头文件中使用了__declspec(dllexport)，在应用程序中就不需要这个.def 文件了。

下面测试一下动态库 testdll。建立一个控制台应用程序工程 testtestdll，测试程序如下：

```
#001 #include <stdio.h>
#002 #include <stdlib.h>
#003 #include "testdll.h"
#004 int main(void){
#005    printf("Hello world!\n");
#006    printf("%d\n",get_id());
#007    return 0;
#008 }
```

这里没有定义 BUILD_LIB，因此这时嵌入的头文件 testdll.h 中 DLL_EXPORT 的含义是输入，因为现在是在使用 dll 库中的函数。

同静态库的使用类似，同样要对工程进行适当的配置才能使工程正确的编译和链接。在 project 菜单中选 build options，添加库头文件的搜索路径 testdll 和要使用的库符号列表文件 libtestdll.a，具体方法同静态库。如果添加正确，就会编译链接生成可执行程序 testtestdll.exe，但这时运行会弹出一个消息框如图 2.2.5，无法启动程序。出现这个错误的原因是应用程序找不到要用的动态库 testdll，testtestdll.exe 在运行的时候动态加载需要的库 testdll.dll 默认的搜索顺序是当前目录、windows 目录、PATH 中列出的各个目录，因此我们必须把 testdll.dll 放在这样的搜索位置里。可以把 testdll.dll 拷贝到应用程序所在的目录中，也可以把它放在 windows 目录里，两种方法均可。

图2.2.5 应用程序找不到动态库