

数据结构

算法

Visual C++ 6.0

程序集

侯识忠 等编著



包含108个完整的可执行程序及源代码



中国水利水电出版社

www.waterpub.com.cn

内 容 提 要

本书收集了当前国内几本比较流行的数据结构算法书中的算法，在 Visual C++ 6.0 平台上将其编写成可以直接运行的程序。对有些算法，还给出了多种程序设计方案实现，以迅速提高程序员的算法与程序设计能力。

本书遵循数据结构算法书中的体系，将全书分为九章对应之，以便阅读。它们是：顺序存储结构的表、堆栈和队列，链式存储结构的表、堆栈和队列，数组、串和广义表，递归，树和二叉树，图，排序，查找，文件。本书的光盘中含有 108 个完整的程序源代码、可执行文件。

本书适合于想要通过编程实例学习数据结构的 C++ 程序员，也可以作为高校师生学习计算机编程与数据结构的参考书，尤其适合报考计算机软件专业研究生的人员与参加信息奥赛班学习的人员参考。

本书配有 PowerPoint 制作的电子教案，教师可以根据情况任意修改，如有需要，可从中国水利水电出版社网站 (<http://www.waterpub.com.cn/softdown/>) 下载。

图书在版编目 (CIP) 数据

数据结构算法。Visual C++ 6.0 程序集 / 侯识忠等编著。—北京：中国水利水电出版社，2005

ISBN 7-5084-2956-7

I. 数… II. 侯… III. ①数据结构②算法分析③C 语言—程序设计
IV. TP311.12

中国版本图书馆 CIP 数据核字 (2005) 第 050421 号

书 名	数据结构算法——Visual C++ 6.0 程序集
作 者	侯识忠 等编著
出版 发行	中国水利水电出版社 (北京市三里河路 6 号 100044) 网址: www.waterpub.com.cn E-mail: mchannel@263.net (万水) sales@waterpub.com.cn 电话: (010) 63202266 (总机)、68331835 (营销中心)、82562819 (万水)
经 售	全国各地新华书店和相关出版物销售网点
排 版	北京万水电子信息有限公司
印 刷	北京蓝空印刷厂
规 格	787mm×1092mm 16 开本 23.75 印张 529 千字
版 次	2005 年 5 月第 1 版 2005 年 5 月第 1 次印刷
印 数	0001—5000 册
定 价	45.00 元 (含 1CD)

凡购买我社图书，如有缺页、倒页、脱页的，本社营销中心负责调换

版权所有·侵权必究

前 言

计算机专业和其他与计算机技术关系密切的许多专业课程都与数据结构这门课程有着紧密联系, 这些课程的学习效果如何, 与数据结构息息相关。

本人多年从事计算机语言课程的教学工作, 如何让学生尽快学会、熟练掌握计算机编程, 需要进行大量的工作, 并提供一定的方法。仅仅让学生了解、掌握计算机语言是不够的, 还必须掌握数据组织、存储和运算的一般方法, 掌握一定的算法。然而, 相当多的数据结构教材给出的算法一是抽象, 二是不能直接用于上机操作, 学生掌握起来有一定困难。

基于此, 作者编写了这本《数据结构算法——Visual C++ 6.0 程序集》。编写本程序集的目的有二, 一是让学习数据结构的读者更容易掌握其中的算法; 二是让学习计算机语言编程的读者尽快学会、熟练掌握数据结构的相关算法, 迅速提高编程能力。

本书对数据结构问题的描述具有相对独立性, 每一个程序都完整地包含了该数据结构问题的各种算法。本书采用 C++ 语言描述具体的数据结构问题, 所有程序都用 C++ Builder 6.0 调试通过, 并输出程序运行结果。

本书第 1 章至第 6 章、第 8 章由侯识忠同志编写, 第 7 章、第 9 章由侯凌翀同志编写, 杜正宽、杜希平、杜琨同志做了大部分程序校对与录入工作, 最后由侯识忠同志统稿。

本书的 108 个完整的可执行程序、源代码都收录在随书光盘中, 光盘中的应用程序除 5 个之外, 其余的都可在光盘中直接执行, 也可将光盘中的程序拷贝到硬盘 D 的子目录中运行。

本书适合于想要通过编程实例学习数据结构的 C++ 程序员, 也可以作为高校师生学习计算机编程与数据结构的参考书, 尤其适合报考计算机软件专业研究生的人员与参加信息奥赛班学习的人员参考。

侯识忠 湖南师范大学数学与计算机科学学院

侯凌翀 湖南省电信有限公司长沙市分公司

2005 年 3 月

目 录

前言

第 1 章 顺序存储结构的表、堆栈和队列	1
1.1 线性表的数组表示和实现	1
1.1.1 程序构思	1
1.1.2 线性表的类定义 <code>linelist1.h</code>	1
1.1.3 线性表的实现 <code>linelist1.cpp</code>	2
1.1.4 线性表的测试 <code>linelist1m.cpp</code>	5
1.1.5 <code>linelist1m.cpp</code> 运行结果	7
1.1.6 分析	8
1.2 线性表的动态分配顺序表示和实现	8
1.2.1 程序构思	8
1.2.2 线性表的类定义 <code>linelist2.h</code>	9
1.2.3 线性表的实现 <code>linelist2.cpp</code>	10
1.2.4 线性表的测试 <code>linelist2m.cpp</code>	13
1.2.5 <code>linelist2m.cpp</code> 运行结果	15
1.2.6 分析	16
1.3 顺序堆栈的类定义（动态分配）和实现	16
1.3.1 程序构思	16
1.3.2 顺序栈的类定义 <code>stack1.h</code>	16
1.3.3 顺序栈的实现 <code>stack1.cpp</code>	17
1.3.4 顺序栈的测试 <code>stack1M.cpp</code>	18
1.3.5 <code>stack1M.cpp</code> 运行结果	19
1.4 顺序堆栈的类定义（动态分配）和实现	20
1.4.1 程序构思	20
1.4.2 顺序栈的类定义 <code>stack.h</code>	20
1.4.3 顺序栈的实现 <code>stack.cpp</code>	21
1.4.4 顺序栈的测试 <code>stackmain.cpp</code>	21
1.4.5 <code>stackmain.cpp</code> 运行结果	23
1.5 顺序堆栈的类定义（数组表示）和实现	23
1.5.1 程序构思	23
1.5.2 顺序堆栈的类定义 <code>linearStack1.h</code>	23
1.5.3 顺序堆栈的实现 <code>linearStack1.cpp</code>	24

1.5.4	顺序堆栈的测试 linearStack1m.cpp	25
1.5.5	linearStack1m.cpp 运行结果	26
1.5.6	分析	26
1.6	将中缀表达式转换为后缀表达式	26
1.7	十进制数转换成八进制数	28
1.8	括号匹配的检验	29
1.9	行编辑程序	30
1.10	行编辑程序	32
1.11	表达式求值	35
1.12	顺序循环队列的类定义（数组表示）和实现	37
1.12.1	程序构思	37
1.12.2	顺序循环队列的类定义 queue1.h	37
1.12.3	顺序循环队列的实现 queue1.cpp	38
1.12.4	顺序循环队列的测试 queue1m.cpp	39
1.12.5	queue1m.cpp 运行结果	39
1.13	顺序循环队列的类定义（动态分配）和实现	40
1.13.1	程序构思	40
1.13.2	顺序循环队列的类定义 queue2.h	40
1.13.3	顺序循环队列的实现 queue2.cpp	41
1.13.4	顺序循环队列的测试 queue2m.cpp	41
1.13.5	queue2m.cpp 运行结果	42
1.14	循环双端队列顺序表示和实现	43
1.14.1	程序构思	43
1.14.2	循环双端队列顺序表示 duilie.cpp	43
1.14.3	循环双端队列实现	43
1.14.4	循环双端队列的测试	44
1.14.5	duilie.cpp 运行结果	45
1.14.6	分析	45
1.15	不考虑优先级相同元素的先进先出问题的顺序优先级队列	45
1.15.1	程序构思	45
1.15.2	不考虑优先级相同元素的先进先出问题的顺序优先级队列类 定义 SqPQueue.h	45
1.15.3	顺序优先级队列的实现 SqPQueue.cpp	46
1.15.4	顺序优先级队列的测试 SqPQueueem.cpp	47
1.15.5	SqPQueueem.cpp 运行结果	48
1.16	考虑优先级相同元素的先进先出问题的顺序优先级队列	48
1.16.1	程序构思	48

1.16.2	考虑优先级相同元素的先进先出问题的顺序优先级队列类 定义 SPQueue.h.....	48
1.16.3	顺序优先级队列的实现 SPQueue.cpp	49
1.16.4	顺序优先级队列的测试 SPQueueem.cpp	50
1.16.5	SPQueueem.cpp 运行结果	51
1.16.6	分析	51
第 2 章	链式存储结构的表、堆栈和队列	52
2.1	单链表的链式表示和实现	52
2.1.1	程序构思	52
2.1.2	单链表的类定义 linklist3.h.....	52
2.1.3	单链表的实现 linklist3.cpp	53
2.1.4	单链表的测试 linklist3m.cpp	57
2.1.5	linklist3m.cpp 运行结果	58
2.1.6	分析	59
2.2	链式堆栈的类定义（动态分配）和实现	59
2.2.1	程序构思	59
2.2.2	链式堆栈的类定义 linearStack2.h	59
2.2.3	链式堆栈的实现 linearStack2.cpp	60
2.2.4	链式堆栈的测试 linearStack2m.cpp	62
2.2.5	linearStack2m.cpp 运行结果	63
2.2.6	分析	63
2.3	后缀表达式求值	63
2.4	链式队列的类定义和实现	65
2.4.1	程序构思	65
2.4.2	链式队列的类定义 linqueue.h	65
2.4.3	链式队列的实现 linqueue.cpp	65
2.4.4	链式队列的测试 linqueuem.cpp	66
2.4.5	linqueuem.cpp 运行结果	67
2.5	单循环链表类定义和实现	68
2.5.1	程序构思	68
2.5.2	单循环链表的类定义 cirlinklist.h.....	68
2.5.3	单循环链表的实现 cirlinklist.cpp.....	69
2.5.4	单循环链表的测试与应用 cirlinklistm.cpp.....	73
2.5.5	cirlinklistm.cpp 运行结果	74
2.5.6	分析	76
2.6	双向循环链表的类定义和实现	76
2.6.1	程序构思	76

2.6.2	双向循环链表的类定义 dcirlinkl.h	76
2.6.3	双向循环链表的实现 dcirlinkl.cpp	77
2.6.4	双向循环链表的测试与应用 dcirlinklm.cpp	82
2.6.5	dcirlinklm.cpp 运行结果	83
2.6.6	分析	85
2.7	迷宫求解	85
第 3 章	数组、串和广义表	88
3.1	变长参数表的应用	88
3.1.1	程序构思	88
3.1.2	VLArgument.cpp 运行结果	89
3.2	建立一维、二维数组的类定义和实现	89
3.2.1	程序构思	89
3.2.2	建立一维、二维数组的类定义 intarray.h	89
3.2.3	建立一维、二维数组的类实现 intarray.cpp	90
3.2.4	建立一维、二维数组的类测试 intarraym.cpp	91
3.2.5	intarraym.cpp 运行结果	92
3.3	稀疏矩阵的类定义与操作	92
3.3.1	程序构思	92
3.3.2	稀疏矩阵的类定义与操作 xishu.h	92
3.3.3	稀疏矩阵相关操作的测试 xishuM.cpp	95
3.3.4	xishuM.cpp 运行结果	97
3.4	十字链表的定义与相关操作	98
3.4.1	程序构思	98
3.4.2	十字链表的定义与相关操作 xishuM1.cpp	98
3.4.3	十字链表的操作测试	100
3.4.4	xishuM1.cpp 运行结果	100
3.5	十字链表的定义与相关操作	100
3.5.1	程序构思	100
3.5.2	十字链表的类定义与相关操作 xishuM2.cpp	100
3.5.3	十字链表相关操作的测试	102
3.5.4	xishuM2.cpp 运行结果	102
3.6	十字链表的定义与相关操作	102
3.6.1	程序构思	102
3.6.2	十字链表的定义与相关操作 xishuM3.cpp	102
3.6.3	十字链表相关操作的测试	104
3.6.4	xishuM3.cpp 运行结果	104
3.7	十字链表的定义与相关操作	104

3.7.1	程序构思	104
3.7.2	十字链表的类定义与相关操作 xishuM4.cpp	105
3.7.3	十字链表相关操作的测试	106
3.7.4	xishuM4.cpp 运行结果	106
3.8	广义表的类定义和实现	107
3.8.1	程序构思	107
3.8.2	广义表的类定义 guangyi.h	107
3.8.3	广义表的类实现 guangyi.cpp	108
3.8.4	广义表的相关操作的测试 guangyiM.cpp	114
3.8.5	guangyiM.cpp 运行结果	116
3.8.6	分析	116
3.9	字符串的模式匹配	117
3.9.1	程序构思	117
3.9.2	字符串的模式匹配测试	118
3.9.3	Findstr.cpp 运行结果	119
3.10	串模式匹配的类定义和实现	120
3.10.1	程序构思	120
3.10.2	串模式匹配的类定义 FindSub.cpp	120
3.10.3	串模式匹配的类实现	120
3.10.4	串模式匹配的测试	121
3.10.5	FindSub.cpp 运行结果	122
第 4 章	递归	123
4.1	递归运算(栈的应用)	123
4.1.1	程序构思	123
4.1.2	递归运算(栈的应用) Recurve.cpp	123
4.1.3	链式栈类实现	124
4.1.4	链式栈类操作测试	124
4.1.5	Recurve.cpp 运行结果	124
4.1.6	分析	124
4.2	使用回溯法求解迷宫问题	125
4.2.1	程序构思	125
4.2.2	路口的结构体定义 migong.cpp	125
4.2.3	迷宫类定义与实现	125
4.2.4	迷宫类的测试	126
4.2.5	migong.cpp 运行结果	126
第 5 章	树和二叉树	127
5.1	树的类定义和实现	127

5.1.1	程序构思	127
5.1.2	树的孩子兄弟表示法为存储结构的结构体 Tree.h	127
5.1.3	树类的实现 Tree.cpp	128
5.1.4	树类相关操作的测试 TreeM.cpp	131
5.1.5	TreeM.cpp 运行结果	132
5.2	二叉树的类定义和实现	132
5.2.1	程序构思	132
5.2.2	二叉树类定义 btree2.h	133
5.2.3	二叉树类的实现 btree2.cpp	134
5.2.4	二叉树类相关操作的测试 btree2M.cpp	137
5.2.5	btree2M.cpp 运行结果	138
5.2.6	分析	139
5.3	二叉树的类定义和实现	139
5.3.1	程序构思	139
5.3.2	二叉树类定义 btree.h	139
5.3.3	二叉树类的实现 btree.cpp	140
5.3.4	二叉树类相关操作的测试 btreeM.cpp	144
5.3.5	btreeM.cpp 运行结果	144
5.3.6	分析	145
5.4	二叉搜索树的类定义和实现	145
5.4.1	程序构思	145
5.4.2	二叉搜索树的类定义 BSTree.h	145
5.4.3	二叉搜索树类的实现 BSTree.cpp	146
5.4.4	二叉搜索树相关操作的测试 BSTreeM.cpp	150
5.4.5	BSTreeM.cpp 运行结果	151
5.5	二叉搜索树的类定义和实现	152
5.5.1	程序构思	152
5.5.2	二叉搜索树的类定义与实现 BSTREE1.h	152
5.5.3	二叉搜索树类的实现	153
5.5.4	二叉搜索树的类型测试 BSTree1M.cpp	156
5.5.5	BSTree1M.cpp 运行结果	157
5.6	二叉搜索树的类定义和实现	157
5.6.1	程序构思	157
5.6.2	二叉搜索树的类定义 BSTreeF.h	158
5.6.3	二叉搜索树类的实现 BSTreeF.cpp	159
5.6.4	二叉搜索树类相关操作的测试 BSTreeFM.cpp	164
5.6.5	BSTreeFM.cpp 运行结果	165

5.7	线索二叉树类定义和实现	165
5.7.1	程序构思	165
5.7.2	线索二叉树结点类型存储结构体 TBSTree.h	166
5.7.3	线索二叉树类的实现	167
5.7.4	线索二叉树类相关操作的测试 TBSTreeM.cpp	170
5.7.5	TBSTreeM.cpp 运行结果	171
5.8	线索二叉树类定义和实现	171
5.8.1	程序构思	171
5.8.2	线索二叉树结点类型存储结构体 TBSTree1.h	172
5.8.3	线索二叉树类的实现	173
5.8.4	线索二叉树类相关操作的测试 TBSTree1M.cpp	175
5.8.5	TBSTree1M.cpp 运行结果	176
5.9	线索二叉树类定义和实现	176
5.9.1	程序构思	176
5.9.2	线索二叉树结点类型存储结构体 TBSTree2.h	176
5.9.3	线索二叉树类与派生类的实现	177
5.9.4	线索二叉树类相关操作的测试 TBSTree2M.cpp	180
5.9.5	TBSTree2M.cpp 运行结果	181
5.10	赫夫曼树与赫夫曼编码	181
5.10.1	程序构思	181
5.10.2	赫夫曼树与赫夫曼编码 Huffman.cpp	181
5.10.3	类实现	182
5.10.4	赫夫曼编码问题的测试	183
5.10.5	Huffman.cpp 运行结果	184
5.11	赫夫曼树与赫夫曼编码	184
5.12	赫夫曼树与赫夫曼编码	186
5.13	线性表的动态分配顺序表示和实现	188
5.14	最小堆的类定义和实现	189
5.14.1	程序构思	189
5.14.2	最小堆的类定义 minheap.h	190
5.14.3	最小堆的实现 minheap.cpp	190
5.14.4	最小堆类的测试 minheapm.cpp	192
5.14.5	minheapm.cpp 运行结果	193
5.15	利用最小堆相关操作进行堆排序	193
5.16	最大堆的类定义和实现	194
5.16.1	程序构思	194
5.16.2	最大堆的类定义 maxheap.h	195

5.16.3	最大堆的实现 maxheap.cpp.....	195
5.16.4	最大堆类的测试 maxheapm.cpp.....	197
5.16.5	maxheapm.cpp 运行结果.....	198
5.17	利用最大堆相关操作进行堆排序.....	198
第 6 章	图	200
6.1	图的类定义和实现.....	200
6.1.1	程序构思.....	200
6.1.2	图的相关数据类型的定义 graph.h.....	200
6.1.3	图的相关运算的实现 graph.cpp.....	201
6.1.4	图的相关运算的测试 graphM.cpp.....	206
6.1.5	graphM.cpp 运行结果.....	207
6.2	图的类定义和实现.....	208
6.2.1	程序构思.....	208
6.2.2	图的相关数据类型的定义 graph0.h.....	209
6.2.3	图的相关运算的实现 graph0.cpp.....	210
6.2.4	图的相关运算的测试 graph0M.cpp.....	215
6.2.5	graph0M.cpp 运行结果.....	216
6.2.6	分析.....	217
6.3	图的类定义和实现.....	217
6.3.1	程序构思.....	217
6.3.2	图的相关数据类型的定义 graph1.h.....	217
6.3.3	图的相关运算的实现 graph1.cpp.....	218
6.3.4	图的相关运算的测试 graph1M.cpp.....	224
6.3.5	graph1M.cpp 运行结果.....	225
6.4	利用普里姆算法求出用邻接矩阵表示的图的最小生成树.....	226
6.4.1	程序构思.....	226
6.4.2	图的相关数据类型的定义 graph2.h.....	226
6.4.3	图的运算的实现文件 graph2.cpp.....	227
6.4.4	图的相关运算的测试 graph2M.cpp.....	229
6.4.5	graph2M.cpp 运行结果.....	230
6.4.6	分析.....	230
6.5	利用克鲁斯卡尔方法求边集数组所示图的最小生成树.....	230
6.5.1	程序构思.....	230
6.5.2	图的相关数据类型的定义 graph3.h.....	231
6.5.3	图的运算的实现文件 graph3.cpp.....	232
6.5.4	图的相关运算的测试 graph3M.cpp.....	235
6.5.5	graph3M.cpp 运行结果.....	235

6.5.6	分析	236
6.6	狄克斯特拉算法（从一个顶点到其余各顶点的最短路径）	236
6.6.1	程序构思	236
6.6.2	最短路径（狄克斯特拉算法）PshortPh	236
6.6.3	狄克斯特拉算法测试 PshortPM.cpp	237
6.6.4	PShortPM.cpp 运行结果	238
6.6.5	分析	238
6.7	最短路径（从一个顶点到其余各顶点的最短路径）	238
6.7.1	程序构思	238
6.7.2	最短路径（狄克斯特拉算法）PshortP1.cpp	239
6.7.3	算法测试	239
6.7.4	PShortP1.cpp 运行结果	240
6.7.5	分析	241
6.8	最短路径（所有顶点之间的最短路径）	241
6.8.1	程序构思	241
6.8.2	最短路径 SShortP.cpp	241
6.8.3	算法测试	242
6.8.4	SShortP.cpp 运行结果	242
6.8.5	分析	244
6.9	最短路径（所有顶点之间的最短路径）	244
6.9.1	程序构思	244
6.9.2	最短路径 SShortP1.cpp	244
6.9.3	算法测试	244
6.9.4	SShortP1.cpp 运行结果	245
6.9.5	分析	247
6.10	最短路径（弗洛伊德算法、所有顶点之间的最短路径）	247
6.10.1	程序构思	247
6.10.2	最短路径（弗洛伊德算法）FloydPh	247
6.10.3	弗洛伊德算法测试 FloydP.cpp	248
6.10.4	FloydP.cpp 运行结果	248
6.10.5	分析	250
6.11	最短路径（弗洛伊德算法、所有顶点之间的最短路径）	250
6.11.1	程序构思	250
6.11.2	弗洛伊德算法测试 FloydP1.cpp	250
6.11.3	FloydP1.cpp 运行结果	251
6.11.4	分析	253
6.12	拓扑排序	253

第7章 排序	256
7.1 桶排序	256
7.2 插入排序法（类方法）	257
7.3 插入排序法（类方法）	259
7.4 插入排序法	260
7.5 希尔排序法（类方法）	262
7.6 希尔排序	263
7.7 快速排序（类方法）	265
7.8 快速排序（类方法）	266
7.9 快速排序	268
7.10 快速排序	269
7.11 通用选择排序法	271
7.12 选择排序	272
7.13 选择排序法（下沉）	273
7.14 选择排序法（类方法）	274
7.15 选择排序法（函数模板、排序不改变原数组各元素的值）	276
7.16 选择排序法（函数模板、上浮）	277
7.17 选择排序法（排序后不改变原数组各元素的值）	279
7.18 通用冒泡排序法（下沉）	280
7.19 冒泡排序法（上浮）	281
7.20 冒泡排序法（下沉）	282
7.21 通用冒泡排序法（上浮）	283
7.22 归并排序（类方法）	285
7.23 归并排序法	287
7.24 堆排序法（类方法）	288
7.25 堆排序法	290
7.26 堆排序法（迭代器）	291
7.27 基数排序法（类方法）	293
7.28 基数排序法（函数模板）	294
7.29 锦标赛排序法	295
7.30 多种排序方法	297
7.31 K 路平衡归并	301
7.32 对外存文件（磁盘文件）进行选择排序的算法	303
7.33 外存文件的排序操作（二路平衡归并）	305
第8章 查找	309
8.1 二分查找法（递归调用）	309
8.2 二分查找法（非递归调用）	309

8.3	二叉排序树的类定义与实现	310
8.3.1	程序构思	310
8.3.2	二叉排序树的类定义	311
8.3.3	二叉排序树的类实现	311
8.3.4	二叉排序树类的测试	313
8.3.5	BinSortT.cpp 运行结果	314
8.4	Fibonacci 查找法	314
8.5	平衡二叉搜索树类定义与实现	315
8.5.1	程序构思	315
8.5.2	平衡二叉搜索树类定义与实现 AVLTREE.h	316
8.5.3	平衡二叉搜索树的类模板实现	317
8.5.4	平衡二叉搜索树类测试 AVLTREEM.cpp	321
8.5.5	AVLTREEM.cpp 运行结果	322
8.5.6	分析	322
8.6	顺序表的查找	322
8.7	B-树的操作	325
8.7.1	程序构思	325
8.7.2	B-树的操作 B_Tree.cpp	325
8.7.3	B-树的相关操作的测试	329
8.7.4	B_Tree.cpp 运行结果	329
8.8	B-树的(类方法)操作	330
8.8.1	程序构思	330
8.8.2	B-树的结点类型定义	330
8.8.3	B-树的类实现	331
8.8.4	B-树的类相关操作的测试	334
8.8.5	B_Tree1.cpp 运行结果	334
8.9	哈希表	334
8.9.1	程序构思	334
8.9.2	哈希表的结点类型定义	335
8.9.3	哈希表的操作实现	335
8.9.4	哈希表的相关操作的测试	336
8.9.5	hashtable.cpp 运行结果	337
8.10	散列表类定义与实现	337
8.10.1	程序构思	337
8.10.2	散列表类定义 LHashL.h	338
8.10.3	散列表类的实现 LHashL.cpp	338
8.10.4	散列表类的相关操作的测试	340

8.10.5	LHashLM.cpp 运行结果.....	341
8.10.6	分析.....	343
第 9 章	文件.....	344
9.1	散列文件的插入、删除和查找操作.....	344
9.1.1	程序构思.....	344
9.1.2	散列文件的类模板定义、插入、删除和查找操作 HashFM.cpp.....	344
9.1.3	散列文件的类模板实现.....	345
9.1.4	散列文件的类模板实现的测试.....	351
9.1.5	HashFM.cpp 运行结果.....	353
9.2	索引文件的相关操作.....	354
9.2.1	程序构思.....	354
9.2.2	索引文件的相关操作 IndexF.cpp.....	355
9.2.3	索引文件的类模板实现.....	356
9.2.4	索引文件的类模板实现的测试 IndexFM.cpp.....	361
9.2.5	IndexFM.cpp 运行结果.....	362
参考文献		364

第 1 章 顺序存储结构的表、堆栈和队列

1.1 线性表的数组表示和实现

1.1.1 程序构思

本程序将 `elem` 表示成数据类型为结构体 `ElemType` 的数组。首先给出了线性表的类定义，然后给出了线性表中函数原型的实现部分，最后对线性表中的主要操作进行了测试，输出运行结果。

其中，对线性表按升序或降序输出 `printlist(int mark)` 函数的形参 `mark` 为标志符，`mark==0` 为对线性表无序输出；`mark==1` 为对线性表升序输出；`mark== -1` 为对线性表降序输出。

1.1.2 线性表的类定义 `linelist1.h`

```
#define MaxListSize 20//顺序表的最大长度
#define EQUAL 1
typedef struct STU{
    char name[10];//姓名
    char stuno[10];//学号
    int age;//年龄
    int score;//成绩
}ElemType;
class List//线性表的类定义
{private:
    //线性表的数组表示
    ElemType elem[MaxListSize];//存线性表元素的数组
    int length; //线性表的当前长度
    int MaxSize; //线性表的最大长度
public:
    //初始化顺序表
    void init(List **L,int ms);
    //删除顺序表
    void DestroyList(List &L){free(&L);}
    //将顺序表置为空表
    void ClearList(){length=0;}
    //判断顺序表是否为空表
    bool ListEmpty()
    {return length==0;}
```

```

//判断顺序表是否为满
bool ListFull()
{return length==MaxSize;}
//决定返回表中元素 pre_e 的前驱
ElemType PriorElem(ElemType cur_e,ElemType &pre_e);
// 决定返回表中元素 next_e 的后继
ElemType NextElem(ElemType cur_e,ElemType &next_e);
//从线性表中删除表头、表尾或等于给定值的元素
bool ListDelete(int,ElemType &e);
//遍历顺序表
void ListTraverse();
//返回顺序表的长度
int ListLength();
// 获取顺序表中第 i 个元素
void GetElem(int,ElemType *);
// 判断顺序表两元素是否相等
bool EqualList(ElemType *,ElemType *);
// 判断顺序表两元素是否不等
bool Less_EqualList(ElemType *,ElemType *);
//顺序表的查找算法
bool LocateElem(ElemType,int);
//更新线性表中的给定元素
bool UpdateList(ElemType& e,ElemType);
//顺序表的合并算法
void MergeList(List *,List *);
//顺序表的插入算法
bool ListInsert(int,ElemType &);
//顺序表的联合算法
void UnionList(List *,List *);
//对线性表按升序或降序输出
void printlist(int);
};

```

1.1.3 线性表的实现 linelist1.cpp

```

#include "linelist1.h"
void List::init(List **L,int ms)
{*L=(List *)malloc(sizeof(List));
 (*L)->length=0;
 (*L)->MaxSize=ms;
}
int List::ListLength()
{return length;}
ElemType List::PriorElem(ElemType cur_e,ElemType &pre_e)
{for(int i=0;i<length;i++)
 if(i!=0&&strcmp(cur_e.name,elem[i].name)==0)
 {pre_e=elem[i-1];return pre_e;}
}

```