数据结构程序设计题典

李春葆 张植民 肖忠付 编著

清华 大 学 出 版 社

http://www.tup.tsinghua.edu.cn

计算机基础程序设计题典丛书

（京）新登字158 号

内 容 简 介

数据结构是计算机科学及相关专业的重要核心课程。本书以程序设计的方式全面讨论了数

据结构算法的实现过程。书中收集了大量富于启发性的例题和习题，并全部给出了答案。

全书分为15章，第1章是数据结构的概述。第2章~第15章以每章一个专题的形式，分别讨论

了顺序表、单链表、双链表、栈、队列、串、广义表、数组、稀疏矩阵、树和二叉树、图、查

找、排序和文件等基本类型的数据结构。每章均由本章摘要、例题解析、习题实践和参考答案4

部分组成。所有的程序均在VC++ 6.0环境下调试通过。

本书可作为计算机科学及相关专业的本、专科学生学习数据结构课程的参考书，也适合考

研和计算机等级水平考试者。

版权所有，盗版必究。

本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

图书在版编目（CIP）数据

书 名 ： 数据结构程序设计题典

作 者 ： 李春葆 张植民 肖忠付

出 版 者 ： 清华大学出版社（北京清华大学学研大厦.邮编100084）

http://www.tup.tsinghua.edu.cn

责任编辑 ： 潘秀燕

印 刷 者 ： 北京市耀华印刷有限责任公司

发 行 者 ： 新华书店总店北京发行所

开 本 ： 787×1092 1/16 印张：26.5 字数：644 千字

版 次 ： 2002 年6 第1 版2002 年6 月第1 次印刷

书 号 ： ISBN 7－302－

印 数 ： 0001~5000

定 价 ： 32.00 元

**目录**

第1章概论................................................................................................................... 1

1.1 本章摘要.................................................................................................................................................2

1.1.1 知识点............................................................................................................................................2

1.1.2 内容概要........................................................................................................................................2

1.2 例题解析.................................................................................................................................................4

1.3 习题实践.................................................................................................................................................7

1.4 参考答案...............................................................................................................................................10

第2章顺序表 ............................................................................................................. 14

2.1 本章摘要...............................................................................................................................................15

2.1.1 知识点..........................................................................................................................................15

2.1.2 内容概要......................................................................................................................................15

2.2 例题解析...............................................................................................................................................16

2.3 习题实践...............................................................................................................................................25

2.4 参考答案...............................................................................................................................................26

第3章单链表 ............................................................................................................. 40

3.1 本章摘要...............................................................................................................................................41

3.1.1 知识点..........................................................................................................................................41

3.1.2 内容概要......................................................................................................................................41

3.2 例题解析...............................................................................................................................................42

3.3 习题实践...............................................................................................................................................69

3.4 参考答案...............................................................................................................................................72

第4章双链表 ............................................................................................................. 98

4.1 本章摘要...............................................................................................................................................99

4.1.1 知识点..........................................................................................................................................99

4.1.2 内容概要......................................................................................................................................99

4.2 例题解析.............................................................................................................................................100

4.3 习题实践.............................................................................................................................................117

4.4 参考答案.............................................................................................................................................117

第5章栈 .................................................................................................................. 126

5.1 本章摘要.............................................................................................................................................127

5.1.1 知识点........................................................................................................................................127

II 目 录

5.1.2 内容概要....................................................................................................................................127

5.2 例题解析.............................................................................................................................................128

5.3 习题实践.............................................................................................................................................142

5.4 参考答案.............................................................................................................................................143

第6章队列............................................................................................................... 154

6.1 本章摘要.............................................................................................................................................155

6.1.1 知识点........................................................................................................................................155

6.1.2 内容概要....................................................................................................................................155

6.2 例题解析.............................................................................................................................................156

6.3 习题实践.............................................................................................................................................168

6.4 参考答案.............................................................................................................................................169

第7章串 .................................................................................................................. 182

7.1 本章摘要.............................................................................................................................................183

7.1.1 知识点........................................................................................................................................183

7.1.2 内容概要....................................................................................................................................183

7.2 例题解析.............................................................................................................................................184

7.3 习题实践.............................................................................................................................................190

7.4 参考答案.............................................................................................................................................191

第8章广义表 ........................................................................................................... 198

8.1 本章摘要.............................................................................................................................................199

8.1.1 知识点........................................................................................................................................199

8.1.2 内容概要....................................................................................................................................199

8.2 例题解析.............................................................................................................................................200

8.3 习题实践.............................................................................................................................................211

8.4 参考答案.............................................................................................................................................212

第9章数组............................................................................................................... 223

9.1 本章摘要.............................................................................................................................................224

9.1.1 知识点........................................................................................................................................224

9.1.2 内容概要....................................................................................................................................224

9.2 例题解析.............................................................................................................................................227

9.3 习题实践.............................................................................................................................................230

9.4 参考答案.............................................................................................................................................231

第10章稀疏矩阵...................................................................................................... 241

10.1 本章摘要...........................................................................................................................................242

10.1.1 知识点......................................................................................................................................242

10.1.2 内容概要..................................................................................................................................242

目录 III

10.2 例题解析...........................................................................................................................................243

10.3 习题实践...........................................................................................................................................247

10.4 参考答案...........................................................................................................................................248

第11章树和二叉树 .................................................................................................. 257

11.1 本章摘要...........................................................................................................................................258

11.1.1 知识点......................................................................................................................................258

11.1.2 内容概要..................................................................................................................................259

11.2 例题解析...........................................................................................................................................262

11.3 习题实践...........................................................................................................................................293

11.4 参考答案...........................................................................................................................................295

第12章图 ................................................................................................................ 318

12.1 本章摘要...........................................................................................................................................319

12.1.1 知识点......................................................................................................................................319

12.1.2 内容概要....................................................................................................................................319

12.2 例题解析...........................................................................................................................................322

12.3 习题实践...........................................................................................................................................344

12.4 参考答案...........................................................................................................................................345

第13章查找............................................................................................................. 360

13.1 本章摘要...........................................................................................................................................361

13.1.1 知识点......................................................................................................................................361

13.1.2 内容概要..................................................................................................................................361

13.2 例题解析...........................................................................................................................................362

13.3 习题实践...........................................................................................................................................370

13.4 参考答案...........................................................................................................................................370

第14章排序............................................................................................................. 381

14.1 本章摘要...........................................................................................................................................382

14.1.1 知识点......................................................................................................................................382

14.1.2 内容概要..................................................................................................................................382

14.2 例题解析...........................................................................................................................................383

14.3 习题实践...........................................................................................................................................393

14.4 参考答案...........................................................................................................................................394

第15章文件............................................................................................................. 400

15.1 本章摘要...........................................................................................................................................401

15.1.1 知识点......................................................................................................................................401

15.1.2 内容概要..................................................................................................................................401

15.2 例题解析...........................................................................................................................................402

IV 目 录

15.3 习题实践...........................................................................................................................................404

15.4 参考答案...........................................................................................................................................405

参考文献 .................................................................................................................... 413

**《计算机基础程序设计题典丛书》序**

计算机软件产业是IT行业最前沿和最有前途的产业之一，也最富有竞争性和挑战性。

随着我国加入WTO，软件产业的发展越来越受到重视。目前，全国有33所大学设立了教授

软件开发的专业学院即软件学院。

“软件＝程序＋文档”，程序设计是软件开发的基础。欲成为一个优秀的软件开发人

才，必须首先从程序设计开始培养和训练起步。试想一下，如果没有基本的程序设计思想

和技术，怎么可能成为一个优秀的软件开发工程师呢？

本丛书是面向培养软件开发人才的基础性程序设计教程，它以程序设计为核心，涵盖

了一些重要的基础性程序设计课目。本套丛书包括：

1. 《C语言程序设计题典》

2. 《C＋＋语言程序设计题典》

3. 《数据结构程序设计题典》

4. 《Visual Basic程序设计题典》

本丛书具有以下特点：

○ 强化程序设计的实现过程

要想成为一个优秀的程序设计工程师，必须深入领会程序设计的奥秘，这样才能做到

触类旁通。所以本丛书不像一般的软件系统使用手册那样仅列出命令的使用语法，也不像

一般的语言教科书那样只注重于抽象的理论方法，而是经过仔细的挑选，从程序实现的角

度精心设计出一系列富有代表性的程序设计题目，并给予完整的解析。

○ 内容叙述力求通俗易懂

本丛书不仅面向计算机专业的本、专科学生，更立足于希望通过基础程序设计训练，

继而成为计算机程序设计工程师的有志青年。因此，在文字叙述和内容安排上尽量通俗易

懂，力求讲出题目设计思想的来龙去脉，把程序设计的整个“过程”讲透。

○ 结构布局完整、合理，便于自学

丛书采用统一的结构，以章为单位，每章讨论一个专题，均由内容要点、例题解析、

习题实践和参考答案构成。所有习题都附有参考答案。例题的求解过程详尽，习题参考答

案简略，但可以从相应的例题中找到程序设计思路。因而适合通过自学提高程序设计能力。

本丛书的作者都是长期从事高校计算机基础教学工作的教授和副教授，他们集教学和

程序设计经验于一炉，精心编写了这套丛书，相信会为我国软件人才的培养起到促进作用。

李春葆

2002.1.5

**前言**

数据结构是计算机科学及相关专业的重要核心课程。对于从事计算机应用领域的开发

人员，数据结构基础是必不可少的。本书从程序设计的角度全面讨论了数据结构算法的实

现过程。

本书的特点

本书具有以下几方面的特点：

（1）数据结构课程的目的是设计“好”的程序，但其内容和原理比较抽象，一般是建

立在学生有一门程序设计语言（如C语言）基础后才开设该课程，但学生仍很难深入掌握

其中奥妙。本书以程序（不仅仅是算法描述）加注释的方式实现了数据结构的算法，便于

读者更好地体会算法的思想。

（2）本书并不是简单地实现了数据结构教程中常用算法，而是收集了大量富于启发性

的题目，并全部给予解答。

（3）不同于数据结构教程,本书每一章一个专题，这样内容更加深入。除了少数章节

外，各章的例题1中都给出了该章讨论的数据结构基本运算的实现，并存放在一个文件中，

后面的例题或习题中大量使用这样运算，而不必重复编写。体现了一种结构化程序设计思

想。

（4）书中的程序（绝大多数程序给出了执行结果）在Visual C++ 6.0环境下调试通过。

而且给出main()主函数，读者可直接上机验证。

对于C语言读者

本书程序在Visual C++ 6.0环境下编写的，读者只需要具备简单的C++基础就能阅读书

中的全部程序。为了照顾只熟悉C语言的读者，本书尽量采用C语言的语法。程序中使用到

的C++语法说明如下：

（1）输入输出语句

C语言和C++语言输入输出语句对应如下（仅限本书中使用到的）：

C语言C++语言

输入 scanf("%d",&n);与n的类型有关cin >> n;与n的类型无关

输出 printf("%d",n);与n的类型有关cout << n; 与n的类型无关

域宽 printf("%3d",n); cout << setw(3) << x;

头文件 stdio.h iostream.h(使用域宽还要iomanip.h)

特定符 \n(换一行) endl(换一行)

（2）函数参数调用

C++语言有一种引用类型,其符号为“&”，例如，以下C函数：

void swap1(int x,int y)

{

int temp;

temp=x;x=y;y=temp;

}

在调用swap1(a,b)时，这种传值方式不会使实参a、b之值交换。为此可改为传地址方式，

即：

void swap2(int \*x,int \*y)

{

int \*temp;

temp=x;\*x=\*y;\*y=temp;

}

这样，在调用swap2(a,b)时,实参a、b之值发生了交换。这两种方式在C++中完全兼容，

但使用引用类型时综合了这两种方法的优点，相应的程序如下：

void swap3(int &x,int &y)

{

int temp;

temp=x;x=y;y=temp;

}

这里，在调用swap(a,b)时，实参a、b之值也发生了交换。

对于本书中使用引用类型的地方，读者改为C程序时，采用将swap3()改为等价的swap2()

的方式进行即可。

如何阅读本书

阅读本书，要求读者对数据结构的基本内容有所了解。读者可先阅读各章中给出基本

运算实现过程的例题1，掌握采用C/C++语言实现基本运算的方法，然后阅读后面的例题，

再独立完成后面的相关习题，并可参阅习题答案。相信通过读者的摸索和总结，可以在较

短时间很快地掌握数据结构的内容和方法。

本书可作为计算机科学及相关专业的本、专科学生学习数据结构课程的参考书，也适

合考研和计算机等级水平考试者。

由于水平所限，尽管编者不遗余力，仍可能存在错误和不足之处，敬请读者批评指正。

编 者

2002.5.20

**第1 章概论**

第1章概 论

程序设计题典

2 数据结构程序设计题典

*1.1 本章摘要*

*1.1.1 知识点*

*1. 数据结构的概念*

数据的逻辑结构、数据的存储结构（物理结构）和数据的运算。

*2. 数据的逻辑结构*

? 线性结构

? 树形结构

? 图状结构

*3. 数据的存储结构*

? 顺序存储方式

? 链式存储方式

? 索引存储方式

? 散列存储方式

*4. 数据的运算*

数据的运算指对数据施加的操作。数据运算的实现是与数据的存储结构有关的。

*5. 算法及算法分析*

算法是解决问题的方法和步骤。

算法分析是对算法的时间性能和空间性能进行评价（通常指前者）。

*1.1.2 内容概要*

*1. 基本概念*

（1）数据是信息的载体，它能够被计算机识别、存储和加工处理。

（2）数据元素是数据的基本单位。有时一个数据元素包括若干个数据项。

第1章 概 论

程序设计题典

数据结构程序设计题典 3

（3）数据结构指的是数据之间的相互关系，它一般包括以下三个方面的内容：

? 数据的逻辑结构——数据元素之间的逻辑关系。

? 数据的存储结构——数据元素及其关系在计算机存储器内的表示。

? 数据的运算——对数据施加的操作。

*2. 数据的逻辑结构*

数据的逻辑结构是从逻辑关系上描述数据，它与数据的存储无关，是独立于计算机的。

它有两大类：

（1）线性结构

线性结构的逻辑特征是：若结构是非空集，则有且仅有一个开始节点和一个终端节点，

并且所有节点都最多只有一个直接前驱和一个直接后继。例如，线性表是一个线性结构。

（2）非线性结构

非线性结构的逻辑特征是：一个节点可能有多个直接前驱和直接后继。例如，树和图

都是非线性结构。

*3. 数据的存储结构*

数据的存储结构是逻辑结构用计算机语言的实现，即建立数据的机内表示。存储结构

主要内容指在存储空间中使用一个存储节点来存储一个数据元素；在存储空间中建立各存

储节点之间的关联来表示数据元素之间的逻辑关系。其中存储节点是指一个数据元素在存

储结构中的存储。

数据的存储结构有如下的4种基本方式：

（1）顺序存储方式

每一个存储节点只含一个数据元素。所有的存储节点相继存储在一个连续的存储区里。

用存储节点间的位置关系表示数据元素之间的逻辑关系。

（2）链接存储方式

每一个存储节点不仅含一个数据元素，还包括指针。每一个指针指向一个与本节点有

逻辑关系的节点，即用指针表示逻辑关系。

（3）索引存储方式

每一个存储节点仅含一个数据元素，所有的存储节点都连续存放。此外，增设一个索

引表。

（4）散列存储方式

每一个存储节点仅含一个数据元素，数据元素按散列(Hash)函数确定存储位置。

*4. 数据的运算*

数据的运算是通过算法描述的，所以讨论算法是数据结构课程的重要内容之一。选用

第1章概 论

程序设计题典

4 数据结构程序设计题典

算法首先考虑正确性，还要考虑执行算法所耗费的时间和存储空间，同时算法应易于理解、

编码、调试等等。常用的运算如下：

· 查找运算——从结构中找出满足某种条件的节点的位置。

· 读取运算——读出结构中指定位置上的内容。

· 插入运算——在结构中的某指定位置上增加一个新节点。

· 删除运算——撤消结构中指定位置上的节点。

· 更新运算——修改结构中某指定节点的内容。

*5. 算法分析*

算法分析包括从时间和空间方面对算法进行特性分析，从时间方面的分析称为时间复

杂度；从空间方面的分析称为空间复杂度。

时间复杂度从好到坏的级别依次是：

常量阶O(1)、对数阶O(log2n)、线性阶O(n)、平方阶O(n2)、指数阶O(2n)

*1.2 例题解析*

【例1.1】以下是两个n阶方阵的乘积C=AB，给出该算法的时间复杂度。

#define max 10

void matmult(int n，float A[max][max]，float B[max][max]，

float C[max][max])

{

int i，j，k;

float x;

for (i=1;i<=n;i++) //①

for (j=1;j<=n;j++) //②

{

x=0; //③

for (k=1;k<=n;k++) //④

x+=A[i][k]\*B[k][j]; //⑤

C[i][j]=x; //⑥

}

}

【解】 在算法中，语句①的循环控制变量i要增加到n+1，测试i>n成立才会终止，故

第1章 概 论

程序设计题典

数据结构程序设计题典 5

它的频度是n+1，但是它的循环体却只能执行n次。语句②作为语句①的循环体内的语句应

执行n次，但语句②本身要执行n+1次，所以语句②的频度是n(n+1)。同理得到语句③、④、

⑤、⑥的频度分别为n2、n2(n+1)、n3、n2。则本算法的时间复杂度T(n)为：

T(n)=n+1+n(n+1)+n2+n2(n+1)+n3+n2=2n3+3n2+2n+1=O(n3)

实际上，我们只需考虑基本语句的时间复杂度即可，上述算法中，语句③⑤⑥是基本

语句，它们的时间复杂度为O(n3)。

【例1.2】设n为如下程序段处理的数据个数，求其时间复杂度。

for (i=1;i<=n;i=2\*i)

cout << "i=" << i << endl;

【解】 其中的基本语句是第2行的cout语句，设它的执行次数为f(n)，则有2f(n)≤n，

即f(n) ≤log2n。

本程序段时间复杂度T(n)=f(n)≤log2n=O(log2n)。

故时间复杂度为O(log2n)。

【例1.3】计算n！的递归函数fac(n)如下，试分析它的时间复杂度。

int fac(int n)

{

if (n<=1) return 1; //①

else return (n\*f(n-1)); //②

}

【解】 设T(n)为fac(n)的时间开销函数。显然T(1)=O(1)。①的时间开销为O(1)，递归

调用fac(n-1)的时间开销为T(n-1)，则②的时间开销为O(1)+T(n-1)。则：

T(n)=O(1)+T(n-1)=O(1)+O(1)+T(n-2)=2×O(1)+T(n-2)

=⋯=(n-1)×O(1)+T(1)=n×O(1)=O(n)

【例1.4】设计一个算法用不多于3n/2的平均比较次数，在数组A[1..n]中找出最大和

最小值的元素。

【解】 本题的算法思想是：如果在查找出最大和最小值的元素时各遍历一遍所有元

素，则至少要比较2n次，为此，使用一遍遍历找出最大和最小值的元素。实现本题功能的

函数如下：

第1章概 论

程序设计题典

6 数据结构程序设计题典

void maxmin(datatype A，int n) //datatype为C语言标准数据类型

{

int i;

datatype max，min;

max=A[1];min=A[1];

for (i=2;i<=n;i++)

if (A[i]>max) max=A[i];

else if (A[i]<min) min=A[i];

cout << "max=" << max << "，min=" << min << endl;

}

在这个函数中，最坏的情况是A中的元素按递减次序排列，这时（A[i]>max）条件均

不成立，比较的次数为n-1。另外每次都要比较（A[i]<min），同样比较的次数为n-1，因此，

总的比较次数为：2(n-1)

最好的情况是A中的元素按递增次序排列，这时（A[i]>max）条件均成立，不会再执

行else的比较，所以总的比较次数为：n-1

平均比较次数为：

2

3

2

3

2

3

2

2(*n* 1) *n* 1 *n n*





所以本算法的平均比较次数不多于3n/2。

【例1.5】设A是一个线性表（a1，a2，⋯，an），采用顺序存储结构，则在等概率

的前提下，平均插入一个元素需要移动的元素个数是多少？若元素插入在ai与ai+1之间（0

≤i≤n-1）的概率为，则平均每插入一个元素所要移动的元素个数又是多少？

【解】 A=(a1，a2，⋯，an)，有n+1个位置可插入元素，即a1之前，a1与a2之间，⋯，

an-1与an之间和an之后，分别需要移动的元素个数为n，n-1，⋯，1，0，则平均插入一个元

素需要移动的元素个数为：

若元素插入在ai与ai+1之间（0≤i≤n-1）的概率为，其中移动的元素个数为：

( 1) /2

( )

( 1) / 2

( )

2













*n n*

*n i*

*n n*

*n i*

*n i*

则平均每插入一个元素所要移动的元素个数为：

( 1)/ 2



*n n*

*n i*

( 1)/ 2



*n n*

*n i*

1 2

1

( ( 1) ... 1 0)

*n*

*n*

*n n* 





第1章 概 论

程序设计题典

数据结构程序设计题典 7

3

2 1

( 1) / 2

1 ( )

1

2 











*n*

*n n*

*n n i*

*i*

*1.3 习题实践*

【题1】分析以下各程序段的时间复杂度。

（1） void main()

{

int i=1，k=0，n=10;

while (i<=n-1)

{

k+=10\*i;i++;

}

}

（2） void main()

{

int i=1，k=0，n=100;

do {

k+=10\*i;i++;

} while (i==n);

}

（3） void main()

{

int i=1，j=0，n=10;

while (i+j<=n)

if (i>j) j++;

else i++;

}

（4） void main()

{

int n=10，x=n，y=0;

while (x>=(y+1)\*(y+1))

y++;

}

第1章概 论

程序设计题典

8 数据结构程序设计题典

（5） void main()

{

int n=9，i=1;

while (i<=n)

i=i\*3;

}

【题2】分析以下程序段的时间复杂度。

⋯

i=1;

while (i<=n)

i=i\*2;

⋯

【题3】假设n为2的乘幂，例如n=2，4，8，16，⋯，试求下列程序的时间复杂度及

变量count的值（以n的函数形式表示）。

#include <iostream.h>

void main()

{

int n，x，count;

cout << "n：";

cin >> n;

count=0;

x=2;

while (x<n/2)

{

x=2\*x;

count++;

}

cout << count << endl;

}

【题4】某算法所需时间由下述方程表示，试求出该算法的时间复杂性的级别（以大

O形式表示）。注意，n为求解问题的规模，为简单起见，设n是2的正整数幂。

T(n)=

n>1时

1 n=1时

*T n*) *n*

2

2 (

第1章 概 论

程序设计题典

数据结构程序设计题典 9

【题5】有以下程序，分析其中order函数的时间复杂度。

int a[]={2，5，1，7，9，3，6，8};

order(int j，int m)

{

int i，temp;

if (j<m)

{

for (i=j;i<=n;i++)

if (a[i]<a[j])

{

temp=a[i];a[i]=a[j]; a[j]=temp;

}

j++;

order(j，m); //递归调用

}

}

void main()

{

int i;

order(0，7);

for (i=0;i<=7;i++)

cout << a[i] << " ";

cout << endl;

}

【题6】斐波那契数列Fn定义如下：F0=0，F1=1，Fn= Fn-1+ Fn-2，n=2，3，⋯请就此

斐波那契数列，回答下列问题。

（1）在递归计算Fn的时候，需要对较小的Fn-1，Fn-2，⋯，F1，F0精确计算多少次？

（2）如果用大O表示法，试给出递归计算Fn时递归函数的时间复杂度是多少？

【题7】设计求解下列问题的C语言算法，并分析其最坏情况的时间复杂度。

（1）在数组A[1..n]中查找值为K的元素，若找到则输出其位置i；否则输出0作为标志。

（2）找出数组A[1..n]中元素的最大值和次最大值。

【题8】以下算法是在一个有n个数据元素的数组A中删除第i个位置的数组元素，要

求当删除成功时数组元素个数减1，求平均删除一个数组元素需要移动的元素个数是多少？

其中，数组下标从0至n-1。

第1章概 论

程序设计题典

10 数据结构程序设计题典

int delete(int A[]，int n，int i)

{

int j;

if (i<0 || i>n) return 0;

for (j=i+1;j<n;j++)

A[j-1]=A[j];

n--;

return 1;

}

*1.4 参考答案*

【题1】（1）O(n) （2）O(1) （3）O(n) （4）O( *n* ) （5）O(log3n)

【题2】此处循环体里面是i=i\*2，即每循环一次，i值增加一倍，所以执行次数与n

之间是以2为底的对数关系，故时间复杂度为O(log2n)。

【题3】 n与count的关系如下：

n=20时，count=0；n=21时，count=0；n=22时，count=0；n=23时，count=1；n=24时，

count=2，⋯；n=2m时，count=m-2。则本算法的时间复杂度为O(log2n)。count= log2n-2。

【题4】设n=2m，则

T(n) =T(2m)=2T(2m-1)+2m

=2(2T(2m-2)+2m-2)+2m=22\*T(2m-2)+2\*2m

=⋯

=2m\*T(1)+m\*2m

=(m+1)2m

=(log2n+1)n

=O(nlog2n)

【题5】order()函数是一个递归排序过程，设T(n)是排序n个元素所需要的时间。在排

序n个元素时，算法的计算时间主要花费在递归调用order()上。第一次调用时，处理的元素

序列个数为n-1，也就是对余下的n-1个元素进行排序，所需要的计算时间应为T(n-1)。又

因为在其中的循环中，需要n-1次比较，所以排序n个元素所需要的时间为：

T(n)=T(n-1)+n-1 n>1

这样得到如下方程：

第1章 概 论

程序设计题典

数据结构程序设计题典 11

T(1)=0

T(n)=T(n-1)+n-1 n>1

求解过程为：

T(n) =[T(n-2)+(n-2)]+(n-1)

=[T(n-3)+(n-3)]+(n-2)+(n-1)

=⋯

=(T(1)+1)+2+⋯+n-1

=0+1+2+⋯+n-1

=n(n-1)/2

=O(n2)

故order函数的时间复杂度为O(n2)。

【题6】（1）从题中可知：F0=0，F1=1，F2=1，F3=2，F4=3，F5=5，F6=8，⋯。

Fn = Fn-1+ Fn-2

= 2Fn-2+ Fn-3(=F3Fn-2+F2Fn-3)

= 3Fn-3+ 2Fn-4(=F4Fn-3+F3Fn-4)

= 5Fn-4+ 3Fn-5(=F5Fn-4+F4Fn-5)

=⋯

= Fn-1F1+ Fn-2F0

归纳起来，在递归计算Fn时，需要对较小的Fn-1计算1（即F2次），Fn-2计算2（即F3次），⋯，

F1计算Fn次，F0计算Fn-1次。

以计算F5为例，其递归调用树如图1.1所示。从图中看到，F4计算1次，F3计算2次，F2

计算3次，F1计算F5(=5)次，F0计算F4(=3)次。

（2）设计算Fn的时间复杂度为T(n)，有：

T(n)=T(n-1)+T(n-2)<2T(n-1)

<22T(n-2)< ⋯

<2nT(0)

=O(2n)

又：

T(n) =T(n-1)+T(n-2)>2T(n-2)

>22T(n-4)> ⋯

>2n/2T(1)(n为奇数时，或2n/2T(0)，n为偶数时)

即：

T(n)>O(2n/2)

则：

第1章概 论

程序设计题典

12 数据结构程序设计题典

O(2n/2)<T(n)<O(2n)

取最坏情况上限，即T(n)=O(2n)。

图 1.1 F(5)递归调用树

【题7】（1）的算法如下：

int locate(datatype A[]，datatype k)

//datatype为C语言标准数据类型

{

int i=n;

while (i>=1 && A[i]!=k) i--;

return i;

}

当查找不成功时，总是比较n+1次，所以最坏的时间复杂度为O(n)。

（2）的算法如下：

void max(datatype A[]，datatype m，datatype sm)

//datatype为C语言标准数据类型

{

int i;

m=sm=A[1]; //m存放最大值，sm存放次最大值

for (i=2;i<=n;i++)

if (A[i]>m)

{

sm=m;m=A[i];

}

else if (A[i]>sm) sm=A[i];

}

F(5)

F(4) F(3)

F(3) F(2) F(2) F(1)

F(2) F(1) F(1) F(0) F(1) F(0)

F(1) F(0)

第1章 概 论

程序设计题典

数据结构程序设计题典 13

为了得到最大值和次最大值，必须经过n-1次循环，所以最坏时间复杂度为n-1，即O(n)。

【题8】当删除最后1个元素时，移动的元素个数为0；当删除倒数第2个元素时，移

动的元素个数为1；⋯；当删除第1个元素时，移动的元素个数为n-1；也就是说，删除第i

个元素时，移动的元素个数为n-i，设P为删除第i个位置上数据元素的概率，则有Pi=1/n，

平均删除一个数组元素需要移动的元素个数是：

1 ( ) 1( 1) ( 2) ... 2 1 0]

1





*n n*

*n*

*n i*

*n*

*n*

*i*

2

*n* 1

**第2 章顺序表**

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 15

*2.1 本章摘要*

*2.1.1 知识点*

*1. 顺序表的概念*

*2. 顺序表的基本运算及其实现*

? 创建顺序表

? 输出顺序表

? 插入运算

? 删除运算

? 查找运算

*2.1.2 内容概要*

*1. 顺序表的概念*

顺序表是线性表的顺序存储结构，即按顺序存储方式构造的线性表的存储结构。

说明：对于有n个元素的顺序表A，可以表示为A[1..n]（适合Pascal），其下标

从1到n，A[1]称为第1个元素，A[2]称为第2个元素，⋯，A[n]称为第n个元素；

也可以表示为A[0..n-1]（适合C/C++），其下标从0到n-1，A[0]称为第1个元素，

A[1]称为第2个元素，⋯，A[n-1]称为第n个元素；无特殊声明时，本书规定采

用后者。

*2. 顺序表的存储结构*

顺序表节点的类型定义如下：

#define MaxLen 50 //顺序表中最多元素个数

typedef int elemtype; //定义elemtype为int类型

typedef elemtype sqlist[MaxLen];



第2章顺 序 表

程序设计题典

16 数据结构程序设计题典

在顺序表中，每个节点ai的存储地址是该节点在表中的位置i的线性函数，只要知道基

地址和每个节点的大小，就可在相同时间内求出任一节点的存储地址。因此顺序表是一种

随机存储结构。

假设表中每个节点占用c个存储单元，并设表中开始节点a1的存储地址(简称为基地址)

是LOC(a1)，那么节点ai的存储地址LOC(ai)可通过下式计算：

LOC(ai)＝LOC(a1)+(i-1)\*c （1≤i≤n）

*3. 顺序表的运算*

? create(sqlist A)：创建顺序表A，并返回其中的元素个数。

? disp(sqlist A，int n)：输出一个具有n个元素的顺序表A。

? ins(sqlist A，int n，int i，elemtype x)：在顺序表A的第i个元素前插入一个元素x。

若i=0，则新元素作为第1个元素;若i=n，则插入在顺序表的最后。

? del(sqlist A，int n，int i)：在顺序表A中删除第i个元素。

? find(sqlist A，int n，elemtype x)：在一个有n个元素的顺序表A中查找元素值为x的

元素。

*2.2 例题解析*

【例2.1】设计顺序表的基本运算算法。

【解】 所谓顺序表，是采用一段连续的区域存放数据的线性表，如图2.1所示，顺序

表A存放n个元素，其长度为n。元素序号从1到n，顺序表A的下标从0到n-1。

图 2.1 顺序表

实现顺序表的基本运算的sq.cpp文件如下：

序号 1 2 3 ⋯ n

下标 0 1 2 ⋯ n-1

A 长度为m

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 17

#include <iostream.h>

#define MaxLen 50 //顺序表中最多元素个数

typedef int elemtype;

typedef elemtype sqlist[MaxLen];

int create(sqlist A) //创建顺序表

{

int i，n;

cout << "创建一个顺序表" << endl;

cout << " 输入元素个数：";

cin >> n;

for (i=0;i<n;i++)

{

cout << " 输入第" << i+1 << "个元素值：";

cin >> A[i];

}

return n;

}

void disp(sqlist A，int n) //输出一个顺序表

{

int i;

cout << "输出一个顺序表" << endl << " ";

if (n==0) cout << "空表";

for (i=0;i<n;i++)

cout << A[i] << " ";

cout << endl;

}

int ins(sqlist A，int n，int i，elemtype x)

//在顺序表的第i个元素前插入一个元素x

//若i=0，则新元素作为第1个元素;若i=n，则插入在最后

{

int j;

if (i<0 || i>n) cout << "i值下溢或上溢" << endl;

else

{

for (j=n-1;j>=i;j--) A[j+1]=A[j];

//将第i个元素及其后的元素后移

A[i]=x; n++; //顺序表长度增1

第2章顺 序 表

程序设计题典

18 数据结构程序设计题典

}

return n;

}

int del(sqlist A，int n，int i) //在顺序表A中删除第i个元素

{

int j;

if (i<0 || i>n) cout << "i值下溢或上溢" << endl;

else

{

for (j=i-1;j<n;j++) A[j]=A[j+1];

//将第i个元素之后的元素前移覆盖A[i]

n--; //顺序表长度减1

}

return n;

}

int find(sqlist A，int n，elemtype x)

//在一个有n个元素的顺序表A中查找元素值为x的元素

{

int i=0;

while (i<=n && A[i]!=x) i++;

if (i<n) return 1;

else return 0;

}

【例2.2】已知数组A[0..n-1]的元素类型为整型，设计算法调整A，使其左边的所有

元素小于0，右边的所有元素大于等于0（要求算法的时间复杂度和空间复杂度均为O(n)）。

【解】 实现本题功能的程序如下：

#include "sq.cpp"

void adjust(sqlist A，int n)

{

sqlist B;

int i，x=0，y=n-1;

for (i=0;i<n;i++)

{

if (A[i]<0)]

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 19

{

B[x]=A[i];x++;

}

else

{

B[y]=A[i];y--;

}

}

for (i=0;i<n;i++) A[i]=B[i];

}

void main()

{

sqlist A;

int n;

n=create(A);

disp(A，n);

adjust(A，n);

disp(A，n);

}

本程序的一次执行结果如下：

创建一个顺序表

输入元素个数：6l

输入第1个元素值：-1l

输入第2个元素值：2l

输入第3个元素值：0l

输入第4个元素值：-5l

输入第5个元素值：-4l

输入第6个元素值：3l

输出一个顺序表

-1 2 0 -5 -4 3

输出一个顺序表

-1 -5 -4 3 0 2

本算法中使用于两次单层for循环，时间复杂度为O(n)，使用了一个B数组作为临时空

间，所以空间复杂度为O(n)。

第2章顺 序 表

程序设计题典

20 数据结构程序设计题典

【例2.3】编写一个算法实现两个有序（从小到大）顺序表合并成为一个顺序表，

合并后的结果放在第一个顺序表中，不另设新的顺序表存储（假设这两个有序顺序表中没

有相同的元素）。

【解】 两个顺序表A和B的合并过程是从A和B中的最后一个元素逐个向前进行比较，

将较大的元素先定位在A中。实现本题功能的程序如下：

#include "sq.cpp"

int comb(sqlist A，int na，sqlist B，int nb)

{

int n=na，m=nb;

if (na+nb>MaxLen) return 0; //顺序表上溢

while (nb>0)

if ((na==0) || (A[na-1]<B[nb-1]))

{

A[na+nb-1]=B[nb-1]; //说明B[nb-1]是第na+nb大的元素

nb--;

}

else

{

A[na+nb-1]=A[na-1]; //说明A[na-1]是第na+nb大的元素

na--;

}

na=n+m;

return na;

}

void main()

{

sqlist A，B;

int na，nb;

na=create(A);

nb=create(B);

disp(A，na);

disp(B，nb);

na=comb(A，na，B，nb);

disp(A，na);

}

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 21

本程序的一次执行结果如下：

创建一个顺序表

输入元素个数：3l

输入第1个元素值：2l

输入第2个元素值：3l

输入第3个元素值：5l

创建一个顺序表

输入元素个数：3l

输入第1个元素值：6l

输入第2个元素值：7l

输入第3个元素值：9l

输出一个顺序表

2 3 5

输出一个顺序表

6 7 9

输出一个顺序表

2 3 5 6 7 9

【例2.4】设有一个顺序表A，包含n个元素，要求写出一个将该表逆置的算法，并

只允许在原表的存储空间外再增加一个附加的工作单元。

【解】 对于有n个元素的A，设k=n/2（即n除2后取整）。将A[0]与A[n-1]交换，A[1]

与A[n-2]交换，⋯，A[k]与A[n-k-1]交换，则将所有元素逆置了。实现本题功能的程序如下

（其中只使用了一个附加的工作单元temp）：

#include "sq.cpp"

void invert(sqlist A，int n)

{

int m=n/2，i; //m为长度的一半即n/2

elemtype temp;

for (i=0;i<m;i++) //将A[i]与A[n-i-1]进行交换

{

temp=A[i];A[i]=A[n-i-1];A[n-i-1]=temp;

}

}

第2章顺 序 表

程序设计题典

22 数据结构程序设计题典

void main()

{

sqlist A;

int n;

n=create(A);

disp(A，n);

invert(A，n);

disp(A，n);

}

本程序的一次执行结果如下：

创建一个顺序表

输入元素个数：5l

输入第1个元素值：2l

输入第2个元素值：6l

输入第3个元素值：3l

输入第4个元素值：1l

输入第5个元素值：8l

输出一个顺序表

2 6 3 1 8

输出一个顺序表

8 1 3 6 2

【例2.5】假设有n（n>1）个线性表顺序地存放在顺序表S[1..m]中，令F[i]和R[i]指

示第i个（1≤i≤n）表的第1个元素和最后1个元素在S中的位置，并设定R[i]<F[i+1]，

F[n+1]=m+1，如图2.2所示。试写出实现下列要求的算法：

（1）在第i个表中的第j项后面插入1个元素，仅当整个[1..m]空间填满时，不允许进行

插入操作。

（2）删除第i个表中的第j个元素，要求在删除第j个元素后，该表仍为顺序存储的线性

表。

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 23

图 2.2 一个顺序表

【解】 本题实质上是将n个线性表（长度可能不相同）放在一个连续空间（长度为m），

来解决这些线性表的插入和删除问题。

（1）的算法如下：

void ins(i，j，x)

{

int p，k;

if (R[n]==m)

cout << "上溢" << endl;

else

{

for (p=n;p>=i+1;p--) //将i+1到n的线性表后移一个元素

{

for (k=R[p];k>=F[p];k--) //将第p个线性表后移一个元素

S[k+1]=S[k];

R[p]++;F[p]++; //第p个线性表的首尾元素位置均增1

}

for (p=R[i];p>=j+1;p--)

//将第i个线性表中的第j个位置起的元素均后移

S[p+1]=S[p];

S[p]=x; //在第i个线性表的第j个位置处存放x

R[p]++; //第p个线性表的R[p]增1

}

}

（2）的算法如下：

void del(i，j)

{

for (p=F[i]+j;p<=m;p++) //元素前移覆盖要删除的元素

第2章顺 序 表

程序设计题典

24 数据结构程序设计题典

S[p]=S[p+1;

for (p=i;p<=n;p++) //第i个及以后的线性表的R[i]值减1

R[p]--;

for (p=i+1;p<=n;p++) //第i＋1个及以后的线性表的F[i]值减1

F[p]--;

}

【例2.6】设A=(a1，a2，⋯，am)和B=(b1，b2，⋯，bn)均为顺序表。

若n=m且ai=bi(i=1，⋯，n)，则称A=B；若ai=bi(i=1，⋯，j)且aj＋1<bj＋1(j＜n≤m)，则称

A<B；在其他情况下均称A>B。试编写一个比较A和B的算法，当A<B、A=B或A>B时分别

输出-1、0或1。

【解】 算法comp()的思想是先找出A和B中前面对应位置相同值的节点，i和j分别为A

和B中比较不相同元素的下标，然后根据i和j的情况得到A和B的比较结果。实现本题功能的

程序如下：

#include "sq.cpp"

int comp(sqlist A，int na，sqlist B，int nb)

{

int i=0，j=0;

while (i<na && j<nb && A[i++]==B[j++]); //比较相同的部分

i--;j--;

if (i==na && j==nb) return 0;

if (i==na && j!=nb) return -1;

if (i!=na && j==nb) return 1;

if (A[i]>B[j])

return 1;

else

return -1;

}

void main()

{

sqlist A，B;

int na，nb，n;

na=create(A);

nb=create(B);

n=comp(A，na，B，nb);

switch(n)

{

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 25

case 0：cout << "A=B" << endl;break;

case 1：cout << "A>B" << endl;break;

case -1：cout << "A<B" << endl;break;

}

}

本程序的一次执行结果如下：

创建一个顺序表

输入元素个数：3l

输入第1个元素值：2l

输入第2个元素值：3l

输入第3个元素值：5l

创建一个顺序表

输入元素个数：4l

输入第1个元素值：2l

输入第2个元素值：3l

输入第3个元素值：4l

输入第4个元素值：5l

比较结果：A>B

*2.3 习题实践*

【题1】设A为顺序表，试编写删除A中第i个元素起的k个元素的算法。

【题2】编写一个算法，将一个顺序表A(有n个元素)分拆成两个顺序表，使A中大于

0的元素存放在B中，小于等于0的元素存放在C中。

【题3】己知一个顺序表A中的元素按值非递减有序，编写一个函数，插入一个元素

x后保持该顺序表是有序的。

【题4】编写一个算法，将m（m>2）个有序（从小到大）顺序表合并成一个有序顺

序表。合并过程中不另设新的顺序表存储。

【题5】已知顺序表A的元素值按递增存放，而顺序表B的元素值按递减存放。设计

一个算法，将B中的所有元素插入到A中（设A的空间足够大），使其仍为递增有序（假设

A和B中没有相同的元素）。

【题6】设A和B是两个顺序表，其元素按从小到大的顺序排列。编写一个将A和B中

相同元素组成一个新的从大到小的有序顺序表C的算法，并分析算法的时间复杂度。

【题7】设A和B是两个顺序表，其元素按从小到大的顺序排列。编写一个将A和B中

第2章顺 序 表

程序设计题典

26 数据结构程序设计题典

所有元素组成一个新的从小到大的有序顺序表C的算法，要求删除重复的元素。

【题8】己知在一维数组A[1：m+n]中依次存放着两个顺序表(a1，a2，⋯，am)和(b1，

b2，⋯，bn)，编写一个函数，将两个顺序表的位置互换，即把(b1，b2，⋯，bn)放到(a1，a2，⋯，

am)的前面。

【题9】设有大小不等的n个数据组（n个数据组中数据的总数为m），顺序存放在空

间D内，每个数据占一个存储单元，数据组的首地址由数组S给出，如图2.3所示。试编写将

新数据x插入到第i个数据组的末尾且属于第i个数据组的算法，插入后，空间区D和数组S的

相互关系仍保持正确。

图 2.3 数据组存储方式

【题10】设有n个值不同的元素存于顺序表A[1..n]中，试问：

你能否用比（2n-3）少的比较次数选出这n个元素的最大值和最小值？若能，请说明是

如何实现的。在最坏情况下，至少要进行多少次比较。

【题11】设某机器表示的整数不超过5位十进制数字。试设计一种表示任意长的整数

的数据结构，并利用你设计的数据结构，写出计算任意给定的两个整数之和的算法。

*2.4 参考答案*

【题1】实现本题功能的程序如下：

#include "sq.cpp"

int delk(sqlist A，int \*n，int i，int k)

{

int j;

if (i<=0 || i+k-1>\*n)

{

L

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 27

cout << "i，k参数不正确" << endl;

return 0;

}

else

{

for (j=i+k-1;j<\*n;j++)

A[j-k]=A[j];

(\*n)-=k;

return 1;

}

}

void main()

{

sqlist A;

int n，i，k;

n=create(A);

disp(A，n);

cout << "输入i，k：";

cin >> i >> k;

if (delk(A，&n，i，k)==1)

disp(A，n);

}

本程序的一次执行结果如下：

创建一个顺序表

输入元素个数：5l

输入第1个元素值：1l

输入第2个元素值：2l

输入第3个元素值：3l

输入第4个元素值：4l

输入第5个元素值：5l

输出一个顺序表

1 2 3 4 5

输入i，k：2 2l

输出一个顺序表

1 4 5

【题2】本题的算法思想是：依次遍历A中的元素，比较当前的元素值，大于0者赋

给B，小于等于0者赋给C。实现本题功能的程序如下：

第2章顺 序 表

程序设计题典

28 数据结构程序设计题典

#include "sq.cpp"

void split(sqlist A，int na，sqlist B，int \*nb，sqlist C，int \*nc)

{

int i，j=0，k=0;

for (i=0;i<na;i++)

if (A[i]>0)

B[j++]=A[i];

else

C[k++]=A[i];

(\*nb)=j;

(\*nc)=k;

}

void main()

{

sqlist A，B，C;

int na，nb，nc;

na=create(A);

disp(A，na);

split(A，na，B，&nb，C，&nc);

disp(B，nb);

disp(C，nc);

}

本程序的一次执行结果如下：

创建一个顺序表

输入元素个数：6l

输入第1个元素值：-1l

输入第2个元素值：2l

输入第3个元素值：0l

输入第4个元素值：-5l

输入第5个元素值：-4l

输入第6个元素值：3l

输出一个顺序表

-1 2 0 -5 -4 3

输出一个顺序表

2 3

输出一个顺序表

-1 0 -5 -4

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 29

【题3】先找到适当的位置，然后后移元素空出一个位置，再将x插入。实现本题功

能的程序如下：

#include "sq.cpp"

int insert(sqlist A，int n，elemtype x) //顺序表A的长度为n

{

int i，j;

if (x>=A[n-1]) A[n]=x; //若x大于最后的元素，则将其插入到最后

else

{

i=0;

while (x>A[i]) i++; //查找插入位置i

for (j=n;j>=i;j--) A[j+1]=A[j]; //移出插入x的位置

A[i]=x;

}

return (n+1); //顺序表的长度增1

}

void main()

{

sqlist A;

int n;

n=create(A);

disp(A，n);

n=insert(A，n，10); //插入元素10

disp(A，n);

}

本程序的一次执行结果如下：

创建一个顺序表

输入元素个数：5l

输入第1个元素值：6l

输入第2个元素值：9l

输入第3个元素值：14l

输入第4个元素值：23l

输入第5个元素值：28l

输出一个顺序表

6 9 14 23 28

输出一个顺序表

6 9 10 14 23 28

第2章顺 序 表

程序设计题典

30 数据结构程序设计题典

【题4】两个顺序表A和B的合并过程是，从A的最后1个元素和B的第1个元素开始分

别向前向后进行比较，将较大的元素先定位在A中。实现本题功能的程序如下：

#include "sq.cpp"

int comb1(sqlist A，int na，sqlist B，int nb)

{

int n=na，m=0;

while (m<nb)

if (n==0 || A[n-1]<B[m])

{

A[n+nb-m-1]=B[m]; //说明B[m]是第n+nb-m大的元素

m++;

}

else

{

A[n+nb-m-1]=A[n-1]; //说明A[n-1]是第n+nb-m大的元素

n--;

}

return na+nb;

}

void main()

{

sqlist A，B;

int na，nb;

na=create(A);

disp(A，na);

nb=create(B);

disp(B，nb);

na=comb1(A，na，B，nb);

disp(A，na);

}

本程序的一次执行结果如下：

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 31

创建一个顺序表

输入元素个数：3l

输入第1个元素值：2l

输入第2个元素值：3l

输入第3个元素值：5l

输出一个顺序表

2 3 5

创建一个顺序表

输入元素个数：3l

输入第1个元素值：9l

输入第2个元素值：8l

输入第3个元素值：7l

输出一个顺序表

9 8 7

输出一个顺序表

2 3 5 7 8 9

【题5】采用上题中的Comb()函数对m个有序表两两合并。实现本题功能的程序如下：

#include "sq.cpp"

void main()

{

sqlist A,B;

int m,i,na,nb;

cout << "输入m: ";

cin >> m;

cout << 1;

na=create(A);

disp(A,na);

for (i=2;i<=m;i++)

{

cout << i;

nb=create(B);

disp(B,nb);

na=comb(A,na,B,nb); //comb()是题的函数

if (na==0) break; //合并不成功退出

}

cout << "合并结果: " << endl;

disp(A,na);

}

第2章顺 序 表

程序设计题典

32 数据结构程序设计题典

【题6】本算法是从A和B的最后两个元素开始比较，若相等，将其值放入C的第1个

元素中，如此直到A和B的所有元素比较完毕。实现本题功能的程序如下：

#include "sq.cpp"

int intersect(sqlist A，int na，sqlist B，int nb，sqlist C)

{

int i=na，j=nb，k=0;

while (i>=0 && j>=0)

if (A[i-1]>B[j-1])

i--;

else if (A[i-1]<B[j-1])

j--;

else //A[i-1]=B[j-1]

{

C[k++]=A[i-1];

i--;j--;

}

return k-1;

}

void main()

{

sqlist A，B，C;

int na，nb，nc;

na=create(A);

disp(A，na);

nb=create(B);

disp(B，nb);

nc=intersect(A，na，B，nb，C);

disp(C，nc);

}

本算法intersect(A，na，B，nb，C)的时间复杂度为O(na+nb)，其中na和nb分别为A和B

的长度。

本程序的一次执行结果如下：

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 33

创建一个顺序表

输入元素个数：3l

输入第1个元素值：2l

输入第2个元素值：3l

输入第3个元素值：5l

输出一个顺序表

2 3 5

创建一个顺序表

输入元素个数：4l

输入第1个元素值：3l

输入第2个元素值：4l

输入第3个元素值：5l

输入第4个元素值：6l

输出一个顺序表

3 4 5 6

输出一个顺序表

5 3

【题7】实现本题功能的程序如下：

#include "sq.cpp"

int unions(sqlist A，int na，sqlist B，int nb，sqlist C)

{

int i=0，j=0，k=0;

while (i<na && j<nb)

if (A[i]<B[j])

C[k++]=A[i++];

else if (A[i]>B[j])

C[k++]=B[j++];

else //A[i]=B[j]

{

C[k++]=A[i];

i++;j++;

}

if (i<na) //A还有元素

for (j=i;j<na;j++)

C[k++]=A[j];

else if (j<nb) //B还有元素

第2章顺 序 表

程序设计题典

34 数据结构程序设计题典

for (i=j;i<nb;i++)

C[k++]=B[i];

return k;

}

void main()

{

sqlist A，B，C;

int na，nb，nc;

na=create(A);

disp(A，na);

nb=create(B);

disp(B，nb);

nc=unions(A，na，B，nb，C);

disp(C，nc);

}

本程序的一次执行结果如下：

创建一个顺序表

输入元素个数：3l

输入第1个元素值：2l

输入第2个元素值：3l

输入第3个元素值：5l

输出一个顺序表

2 3 5

创建一个顺序表

输入元素个数：4l

输入第1个元素值：3l

输入第2个元素值：4l

输入第3个元素值：5l

输入第4个元素值：6l

输出一个顺序表

3 4 5 6

输出一个顺序表

2 3 4 5 6

【题8】本题的算法思想是：由于顺序表的插入与删除操作需要大量的元素移动，所

用时间多，这里采用先将：

A: (a1，a2，⋯，am，b1，b2，⋯，bn)

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 35

的所有元素逆置，即使之变成：

A: (bn，⋯，b2，b1，am，⋯，a2，a1)

然后将(bn，⋯，b2，b1)逆置为(b1，b2，⋯，bn)，将(am，⋯，a2，a1)逆置为(a1，a2，⋯，

am)，这样便得到最终结果：

A: (b1，b2，⋯，bn，a1，a2，⋯，am)

实现本题功能的程序如下：

#include "sq.cpp"

void invert(sqlist A，int low，int high)

{

int m=(high-low+1)/2，i; //m为长度的一半

elemtype temp;

for (i=0;i<m;i++) //将A[i]与A[n-i-1]进行交换

{

temp=A[low+i];A[low+i]=A[high-i];A[high-i]=temp;

}

}

void main()

{

sqlist A;

int na，n;

na=create(A);

disp(A，na);

cout << "输入n：";

cin >> n;

invert(A，0，na-1);

disp(A，na);

invert(A，0，n-1);

disp(A，na);

invert(A，n，na-1);

disp(A，na);

}

本程序的一次执行结果如下（其中n指定第2组数序的长度）：

第2章顺 序 表

程序设计题典

36 数据结构程序设计题典

创建一个顺序表

输入元素个数：7l

输入第1个元素值：1l

输入第2个元素值：2l

输入第3个元素值：3l

输入第4个元素值：5l

输入第5个元素值：6l

输入第6个元素值：7l

输入第7个元素值：8l

输出一个顺序表

1 2 3 5 6 7 8

输入n：4l

输出一个顺序表

8 7 6 5 3 2 1

输出一个顺序表

5 6 7 8 3 2 1

输出一个顺序表

5 6 7 8 1 2 3

【题9】本题算法如下：

void ins(i，x)

{

int \*p，j;

if (i==n) //若i=n，则将x插入到最后

D[m+1]=x;

else

{

for (p=D+m+1;p>=S[n];p--)

//S[n]的元素均后移，D+m表示D[m]的地址

\*p=\*(p-1);

S[n]++; //将前一个位置的值放入后一个位置上

for (j=n-1;j>i;j--) //S[i+1]到S[n-1]的所有元素后移

{

for (p=S[j+1]-1;p>=S[j];p--) \*p=\*(p-1);

S[j]++;

}

\*(S[i+1]-1)=x;

//将x放在S[i+1]前一个位置上，即S[i]的最后位置

}

}

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 37

【题10】能够实现，算法如下：

void maxmin(datatype A，int n) //datatype为C语言标准数据类型

{

int i;

datatype max，min;

max=A[1];min=A[1];

for (i=2;i<=n;i++)

if (A[i]>max) max=A[i];

else if (A[i]<min) min=A[i];

}

在这个函数中，最坏情况是A中的元素递减次序排列，这时（A[i]>max）比较均要执

行，这时比较的次数为n-1，而且该条件均为假，因此每次都要比较（A[i]<min），同样比

较次数为n-1，因此，总的比较次数为：2(n-1)即2n-2。

【题11】将用户输入的正整数按各位数字存放在一个顺序表中，这样就变成了两个

顺序表中的数字相加。实现本题功能的程序如下：

#include "sq.cpp"

int input(sqlist A)

{

int i;

for (i=0;i<MaxLen;i++)

A[i]=0;

cout << "输入一个正整数的各位(以-1结束)" << endl;

i=0;

while (1)

{

cin >> A[i];

if (A[i]<0) break;

i++;

}

return i;

}

void output(sqlist A，int low，int high)

//输出顺序表A中的A[low]到A[high]

{

int i;

for (i=low;i<high;i++)

第2章顺 序 表

程序设计题典

38 数据结构程序设计题典

cout << A[i];

cout << endl;

}

void move(sqlist A，int na) //将A中存放的数字移到后面

{

int i;

for (i=0;i<na;i++)

A[MaxLen-i-1]=A[na-i-1];

}

int add(sqlist A，int na，sqlist B，int nb)

//实现A，B相加，结果放在A中

{

int nc，i，j，length;

if (na>nb) nc=na;

else nc=nb;

move(A，na);

move(B，nb);

for (i=MaxLen-1;i>=MaxLen-nc;i--)

{

j=A[i]+B[i];

if (j>9)

{

A[i-1]=A[i-1]+1;

A[i]=j-10;

}

else A[i]=j;

if (i==MaxLen-nc)

{

if (j>9)

{

A[i-1]=1;

length=nc+1;

}

else length=nc;

}

}

return length;

第2章 顺 序 表

程序设计题典

数据结构程序设计题典 39

}

void main()

{

sqlist A，B;

int na，nb，nc;

na=input(A);

nb=input(B);

cout << "整数A：";

output(A，0，na);

cout << "整数B：";

output(B，0，nb);

nc=add(A，na，B，nb);

cout << "相加：";

output(A，MaxLen-nc，MaxLen);

}

本程序的一次执行结果如下：

输入一个正整数的各位（以-1结束）

9 9 9 9 9 9 9 9 9 9 9 9 9 -1l

输入一个正整数的各位（以-1结束）

6 6 6 6 6 6 6 6 6 6 6 6 6 -1l

整数A：9999999999999

整数B：6666666666666

相 加：16666666666665

**第3 章单链表**

第3章 单 链 表

程序设计题典

数据结构程序设计题典 41

*3.1 本章摘要*

*3.1.1 知识点*

*1. 单链表的结构*

? 带头节点（哨兵）的单链表

? 不带头节点的单链表

*2. 单链表的基本运算*

· 创建一个单链表

· 输出一个单链表

· 求单链表的长度

· 查找第i个节点

· 插入一个节点

· 删除一个节点

· 释放单链表占用的空间

*3.1.2 内容概要*

*1. 链表的概念*

（1）按链式存储方式存储的线性表称为链表。链表分为单链表、双链表和循环链表等。

（2）链表中的元素顺序靠链表节点中的指针指明，链表中的元素顺序与线性表的逻辑

顺序一致。

（3）顺序表的存储空间大小是预先设定的固定值，存储元素的个数等于设定的

MaxLen，存在表满的问题。为了解决该问题，引进了不必考虑表满情况的存储结构，即链

表。

*2. 单链表的存储结构*

单链表采用一个节点存放一个数据元素，每个节点除了包括存放数据元素值的数据域

(data)外，还包括指向下一个元素的存储位置的指针域(next)。最后一个节点的指针域为空。

第3章单 链 表

程序设计题典

42 数据结构程序设计题典

单链表节点的类型定义如下：

typedef int elemtype; //定义数据域的类型

typedef struct linknode { //定义节点类型

elemtype data;

struct linknode \*next;

} nodetype;

循环单链表与单链表基本相同，每个节点也包括一个数据域和一个指针域。但最后一

个节点的指针域不空，指向表头节点。

存储单链表时，可以带一个头节点即哨兵，也可以不带头节点。对于带头节点的单链

表，头节点不计在单链表的节点个数之内，这样多占用一个节点的存储空间，但在实现插

入和删除节点运算时更方便。

*3. 单链表的运算*

· create()：创建单链表，由用户输入各节点data域的值，以0表示输入结束，最后返

回建立的单链表的头指针。

· disp(nodetype \*h)：输出单链表h的所有节点的data域的值。

· len(nodetype \*h)：返回单链表h的长度。

· find(nodetype \*h，int i)：\_\_\_\_\_\_\_返回单链表h中第i个节点的指针。

· ins(nodetype \*h，int i，elemtype x) ：在单链表h中第i个节点(i>=0)之后插入一个data

域为x的节点，并返回插入节点后的单链表的头指针。

? del(nodetype \*h，int i)：删除单链表h中第i个节点，并返回插入节点后的单链表的

头指针。

? dispose(nodetype \*h)：释放单链表h中所有节点占用的存储空间。

*3.2 例题解析*

【例3.1】设计单链表结构并实现单链表的基本运算。

【解】 不带哨兵节点的单链表结构如图3.1所示，其中h为头节点指针，简称为头指

针。

第3章 单 链 表

程序设计题典

数据结构程序设计题典 43

图 3.1 不带哨兵节点的单链表结构

实现本题功能的slink.cpp文件如下（其中的单链表都是不带哨兵节点的单链表）：

#include <iostream.h>

#include <malloc.h>

typedef int elemtype; //定义数据域的类型

typedef struct linknode //定义节点类型

{

elemtype data;

struct linknode \*next;

} nodetype;

nodetype \*create()

//建立单链表，由用户输入各节点data域之值，以0表示输入结束

{

elemtype d;

nodetype \*h=NULL，\*s，\*t;

int i=1;

cout << "建立一个单链表" << endl;

while (1)

{

cout << " 输入第" << i << "节点data域值：";

cin >> d;

if (d==0) break; //以0表示输入结束

if (i==1) //建立第一个节点

{

h=(nodetype \*)malloc(sizeof(nodetype));

h->data=d;h->next=NULL;t=h;

}

else //建立其余节点

{

s=(nodetype \*)malloc(sizeof(nodetype));

s->data=d;s->next=NULL;t->next=s;

第3章单 链 表

程序设计题典

44 数据结构程序设计题典

t=s; //t始终指向生成的单链表的最后一个节点

}

i++;

}

return h;

}

void disp(nodetype \*h) //输出由h指向的单链表的所有data域之值

{

nodetype \*p=h;

cout << "输出一个单链表：" << endl << " ";

if (p==NULL) cout << "空表";

while (p!=NULL)

{

cout << p->data << " "; p=p->next;

}

cout << endl;

}

int len(nodetype \*h) //返回单链表的长度

{

int i=0;

nodetype \*p=h;

while (p!=NULL)

{

p=p->next;i++;

}

return i;

}

nodetype \*find(nodetype \*h，int i) //返回第i个节点的指针

{

nodetype \*p=h;

int j=1;

if (i>len(h) || i<=0) return NULL; //i上溢或下溢

else

{

while (p!=NULL && j<i)

//查找第i个节点，并由p指向该节点

第3章 单 链 表

程序设计题典

数据结构程序设计题典 45

{

j++;p=p->next;

}

return p;

}

}

nodetype \*ins(nodetype \*h，int i，elemtype x)

//在单链表head中第i个节点(i>=0)之后插入一个data域为x的节点

{

nodetype \*p，\*s;

s=(nodetype \*)malloc(sizeof(nodetype)); //创建节点s

s->data=x;s->next=NULL;

if (i==0)

//i=0：s作为该单链表的第一个节点

{

s->next=h;h=s;

}

else

{

p=find(h，i); //查找第i个节点，并由p指向该节点

if (p!=NULL)

{

s->next=p->next;p->next=s;

}

else cout << "输入的i值不正确" << endl;

}

return h;

}

nodetype \*del(nodetype \*h，int i)

//删除第i个节点

{

nodetype \*p=h，\*s;

int j=1;

if (i==1) //删除第1个节点

{

h=h->next;free(p);

}

else

第3章单 链 表

程序设计题典

46 数据结构程序设计题典

{

p=find(h，i-1); //查找第i-1个节点，并由p指向该节点

if (p!=NULL && p->next!=NULL)

{

s=p->next; //s指向要删除的节点

p->next=s->next; free(s);

}

else cout << "输入的i值不正确" << endl;

}

return h;

}

void dispose(nodetype \*h) //释放单链表的所有节点占用的空间

{

nodetype \*pa=h，\*pb;

if (pa!=NULL)

{

pb=pa->next;

if (pb==NULL) //只有一个节点的情况

free(pa);

else

{

while (pb!=NULL) //有两个及以上节点的情况

{

free(pa); pa=pb; pb=pb->next;

}

free(pa);

}

}

}

【例3.2】设A=(a1，a2，⋯，an)和B=(b1，b2，⋯，bm)是两个不带哨兵的单链表。若

n=m且ai=bi(i=1，⋯，n)，则称A=B；若ai=bi(i=1，⋯，j)且aj＋1<bj＋1(j＜n≤m)，则称A<B；

在其他情况下均称A>B。试编写一个比较A和B的算法，当A<B、A=B或A>B时分别输出-1、

0或1。

第3章 单 链 表

程序设计题典

数据结构程序设计题典 47

【解】 算法comp()的思想是先找出A和B单链表中对应位置相同值的节点，pa和pb分

别指向A和B单链表中对应位置不相同的节点，然后根据pa和pb的情况得到A和B单链表的

比较结果。实现本题功能的程序如下：

#include "slink.cpp"

int comp(nodetype \*ha，nodetype \*hb)

{

nodetype \*pa=ha，\*pb=hb;

while (pa!=NULL && pb!=NULL && pa->data==pb->data)

//比较相同的部分

{

pa=pa->next;

pb=pb->next;

}

if (pa==NULL && pb==NULL) return 0;

if (pa==NULL && pb!=NULL) return -1;

if (pa!=NULL && pb==NULL) return 1;

if (pa->data>pb->data)

return 1;

else

return -1;

}

void main()

{

nodetype \*ha，\*hb;

int n;

ha=create();

hb=create();

n=comp(ha，hb);

switch(n)

{

case 0：cout << "A=B" << endl;break;

case 1：cout << "A>B" << endl;break;

case -1：cout << "A<B" << endl;break;

}

}

本程序的一次执行结果如下：

第3章单 链 表

程序设计题典

48 数据结构程序设计题典

建立一个单链表

输入第1节点data域值：5l

输入第2节点data域值：6l

输入第3节点data域值：0l

建立一个单链表

输入第1节点data域值：5l

输入第2节点data域值：6l

输入第3节点data域值：7l

输入第4节点data域值：0l

A<B

【例3.3】有一个单链表的第一个节点指针为head，编写一个函数将该单链表逆置，

即最后一个节点变成第一个节点，原来倒数第二个节点变成第二个节点，如此等等，在逆

置中不能建立新的单链表。

【解】 由于题目规定在逆置中不能建立新的单链表，所以只能在该单链表上边遍历

边逆置。以p为遍历单链表的指针变量，让p指向一个节点，q指向其后的一个节点，r指向q

之后的一个节点(r可以为空)，如图3.2所示。然后让q节点指向r节点的指针反过来指向p节点，

即q->next=p，再p、q所指的节点均后移一个节点，即p=q，q=r。如果q不为NULL，继续这

样下去，则整个单链表逆置了。

图 3.2 一个单链表

实现本题功能的程序如下：

第3章 单 链 表

程序设计题典

数据结构程序设计题典 49

#include "slink.cpp"

nodetype \*invert(nodetype \*h) //实现单链表逆置

{

nodetype \*p，\*q，\*r;

if (len(h)<=1)

{

cout << "逆置的单链表至少有2个节点" << endl;

return NULL;

}

else

{

p=h;q=p->next;

while (q!=NULL)

{

r=q->next;

q->next=p;

p=q;q=r;

}

h->next=NULL;

h=p;

return h;

}

}

void main()

{

nodetype \*head;

head=create();

disp(head);

head=invert(head);

disp(head);

}

本程序的一次执行结果如下：

第3章单 链 表

程序设计题典

50 数据结构程序设计题典

建立一个单链表

输入第1节点data域值：1l

输入第2节点data域值：2l

输入第3节点data域值：3l

输入第4节点data域值：4l

输入第5节点data域值：5l

输入第6节点data域值：0l

输出一个单链表：

1 3 4 5 6

输出一个单链表

5 4 3 2 1

【例3.4】从一个无序的单链表生成一个有序(从小到大)单链表，在生成有序单链表的

过程中不能建立新的单链表，也就是说利用原有的单链表存储空间来生成有序单链表。

【解】 先生成一个带头节点head的单链表，开始时只有该空，以后从原单链表中添

加节点到该有序单链表中，添加时保持节点是有序的。用p指针遍历该单链表，将p所指的

节点有序插入到head单链表中，如图3.3所示。

图 3.3 单链表有序插入

实现本题功能的程序如下：

⋯ 原有的单链表

⋯ 生成的有序单链表

有序插入

head

第3章 单 链 表

程序设计题典

数据结构程序设计题典 51

#include "slink.cpp"

nodetype \*order(nodetype \*h)

{

nodetype \*head，\*p=h，\*q，\*r，\*t;

head=(nodetype \*)malloc(sizeof(nodetype)); //head为哨兵

head->data=0; //存放有序单链表的节点个数

head->next=NULL;

while (p!=NULL)

{

if (head->data==0) //有序单链表为空的情况

{

head->next=p;

q=p->next;

p->next=NULL;

p=q;

}

else

{

q=head->next;r=head;

while (q!=NULL && p->data>q->data)

//找到r节点，将p节点插入其后

{

r=q;

q=q->next;

}

t=p->next;

p->next=r->next;

r->next=p;

p=t;

}

head->data++;

}

t=head;head=head->next;free(t); //删除哨兵

return head;

}

void main()

{

第3章单 链 表

程序设计题典

52 数据结构程序设计题典

nodetype \*head;

head=create();

disp(head);

head=order(head);

disp(head);

}

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：6l

输入第3节点data域值：3l

输入第4节点data域值：4l

输入第5节点data域值：1l

输入第6节点data域值：5l

输入第7节点data域值：0l

输出一个单链表：

2 6 3 4 1 5

输出一个单链表：

1 2 3 4 5 6

【例3.5】设计一个统计选票的算法，输出每个候选的得票结果(假定采用单链表存

放选票，候选人编号依次为1，2，3，⋯，N，且每张选票选且只选一人。

【解】 以单链表存放选票，每个节点的data域存放该选票所选的候选人。用一个数

组a统计得票结果。实现本题功能的程序如下：

#include "slink.cpp"

#include <iomanip.h>

#define N 10

void sat(nodetype \*h，int a[])

{

nodetype \*p=h;

while (p!=NULL)

{

第3章 单 链 表

程序设计题典

数据结构程序设计题典 53

a[p->data]++;

p=p->next;

}

}

void main()

{

int a[N+1]，i;

for (i=0;i<=N;i++)

a[i]=0;

nodetype \*head;

head=create();

sat(head，a);

cout << "候选人：";

for (i=1;i<=N;i++) cout << setw(3) << i;

cout << endl << "得票数：";

for (i=1;i<=N;i++)

cout << setw(3) << a[i];

cout << endl;

}

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：1l

输入第2节点data域值：2l

输入第3节点data域值：1l

输入第4节点data域值：2l

输入第5节点data域值：1l

输入第6节点data域值：3l

输入第7节点data域值：1l

输入第8节点data域值：5l

输入第9节点data域值：6l

输入第10节点data域值：0l

候选人： 1 2 3 4 5 6 7 8 9 10

得票数： 4 2 1 0 1 1 0 0 0 0

第3章单 链 表

程序设计题典

54 数据结构程序设计题典

【例3.6】假设有两个按元素值递增有序排列的单链表A和B，编写一个算法将A和B

归并成一个按元素值递增有序排列的单链表C(相同值的元素只归并一次)，并要求利用原单

链表(A和B)的节点空间存放单链表C。

【解】实现本题功能的程序如下：

#include "slink.cpp"

nodetype \*connect(nodetype \*h1，nodetype \*h2)

{

nodetype \*pa=h1，\*pb=h2，\*h3，\*pc;

h3=(nodetype \*)malloc(sizeof(nodetype)); //创建哨兵

pc=h3; //pc总是指向生成的新单链表的最后一个节点

while (pa!=NULL && pb!=NULL)

if (pa->data<pb->data)

{

pc->next=pa;pc=pa;

pa=pa->next;

}

else if (pa->data>pb->data)

{

pc->next=pb;pc=pb;

pb=pb->next;

}

else //pa->data=pb-data的情况

{

pc->next=pa;pc=pa;

pa=pa->next;pb=pb->next;

}

if (pa!=NULL) pc->next=pa; //h1单链表还有节点时

if (pb!=NULL) pc->next=pb; //h2单链表还有节点时

pc=h3; //删除哨兵

h3=h3->next;

free(pc);

return h3;

}

void main()

{

第3章 单 链 表

程序设计题典

数据结构程序设计题典 55

nodetype \*head1，\*head2，\*head3;

head1=create();

head2=create();

disp(head1);

disp(head2);

head3=connect(head1，head2);

disp(head3);

}

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：5l

输入第3节点data域值：0l

建立一个单链表

输入第1节点data域值：1l

输入第2节点data域值：4l

输入第3节点data域值：8l

输入第4节点data域值：0l

输出一个单链表：

2 5

输出一个单链表：

1 4 8

输出一个单链表：

1 2 4 5 8

【例3.7】假如以单链表表示集合(每个单链表中不存在重复的元素)，设计一个算法

求这两个集合的差。

【解】 假设两个已知集合A和B，根据集合运算的规则可知集合A-B中包含所有属于

集合A而不属于集合B的元素。因此，为了求A-B，需先建立表示集合A和B的单链表，然

后对A单链表中的每个节点p，在B单链表中进行查找，若存在和p相同的元素，则从该链表

中删除，其过程如图3.4所示。其subs()用于计算两个集合之差。实现本题功能的程序如下：

第3章单 链 表

程序设计题典

56 数据结构程序设计题典

图 3.4 从A 单链表生成集合差的过程

#include "slink.cpp"

nodetype \*subs(nodetype \*heada，nodetype \*headb)

{

nodetype \*p，\*q，\*r，\*s;

s=(nodetype \*)malloc(sizeof(nodetype));

//为了算法简单，给A创建一个哨兵

s->next=heada;

heada=s;

//开始求两集合差

p=heada->next;

r=heada;r->next=NULL; //r指向生成的单链表的最后一个节点

while (p!=NULL)

{

q=headb;

while (q!=NULL && q->data!=p->data) q=q->next;

if (q!=NULL) //若p节点在B中，则要删除之

{

s=p->next;

free(p);p=s;

}

else //若p节点不在B中，则链接到新单链表中

{

r->next=p;s=p->next;

r=p;r->next=NULL;

p=s;

}

处理中的A单链表

生成的集合差

第3章 单 链 表

程序设计题典

数据结构程序设计题典 57

}

s=heada;heada=heada->next;free(s); //删除哨兵节点

return heada;

}

void main()

{

nodetype \*ha，\*hb，\*hc;

ha=create(); //创建A集合

hb=create(); //创建B集合

hc=subs(ha，hb);

cout << "A与B集合相减的结果" << endl;

disp(hc);

}

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：5l

输入第3节点data域值：3l

输入第4节点data域值：8l

输入第5节点data域值：0l

建立一个单链表

输入第1节点data域值：3l

输入第2节点data域值：4l

输入第3节点data域值：5l

输入第4节点data域值：0l

A与B集合相减的结果

输出一个单链表：

2 8

【例3.8】编写一个函数，将一个头节点指针(哨兵)为heada的单链表A分解成两个单

链表A和B，其头节点指针分别为heada和headb，使得A链表中含有原链表A中序号为奇数的

元素，而B链表中含有原链表A中序号为偶数的元素，且保持原来的相对顺序。

【解】 split()函数是将单链表A中的所有偶数序号的节点删除，并在删除时把这些节

点链接起来构成单链表B。实现本题功能的程序如下：

第3章单 链 表

程序设计题典

58 数据结构程序设计题典

#include "slink.cpp"

nodetype \*split(nodetype \*ha)

{

nodetype \*p=ha->next，\*hb，\*q，\*ra，\*rb;

ra=ha;ra->next=NULL; //ra指向A的最后一个节点

hb=(nodetype \*)malloc(sizeof(nodetype));//给B链表添加一个哨兵

hb->next=NULL;rb=hb;rb->next=NULL; //rb指向B的最后一个节点

while (p!=NULL)

{

q=p->next; //q指向奇数序号的节点

ra->next=p;ra=p;ra->next=NULL; //将p节点链到A中

p=q;

if (p!=NULL) //若p不为空，将其所指节点链到B中

{

q=p->next;

rb->next=p;rb=p;rb->next=NULL;

p=q;

}

}

return hb;

}

void main()

{

nodetype \*heada，\*headb，\*s;

heada=create();

disp(heada);

s=(nodetype \*)malloc(sizeof(nodetype));//给A链表添加一个哨兵

s->next=heada;

heada=s;

headb=split(heada);

s=heada;heada=heada->next;free(s); //删除A的哨兵

s=headb;headb=headb->next;free(s); //删除B的哨兵

disp(heada);

disp(headb);

}

第3章 单 链 表

程序设计题典

数据结构程序设计题典 59

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：1l

输入第2节点data域值：2l

输入第3节点data域值：3l

输入第4节点data域值：4l

输入第5节点data域值：5l

输入第6节点data域值：0l

输出一个单链表：

1 2 3 4 5

输出一个单链表

1 3 5

输出一个单链表：

2 4

【例3.9】有两个具有相同节点个数的单链表A和B，A={a1，a2，⋯，an}，B={b1，

b2，⋯，bn}，编写一个函数将其合并成一个链表C，C={a1，b1，a2，b2，⋯，an，bn}。

【解】 假设ha、hb分别为指向A、B单链表的指针，本题的算法思想是：循环遍历这

两个链表，每循环一次，将两个链表的对应节点链接起来，如图3.5所示。实现本题功能的

程序如下：

图3.5 算法的实现过程

第3章单 链 表

程序设计题典

60 数据结构程序设计题典

#include "slink.cpp"

nodetype \*combine(nodetype \*ha，nodetype \*hb)

{

nodetype \*hc=ha，\*pa=ha，\*pb=hb，\*q，\*r;

if (len(pa)!=len(pb))

{

cout << "两个单链表的长度不同" << endl;

return NULL;

}

while (pa!=NULL)

{

q=pa->next;r=pb->next;

pa->next=pb;pb->next=q;

pa=q;pb=r;

}

return(hc);

}

void main()

{

nodetype \*heada，\*headb，\*headc;

heada=create();

headb=create();

headc=combine(heada，headb);

disp(headc);

}

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：1l

输入第2节点data域值：3l

输入第3节点data域值：5l

输入第4节点data域值：0l

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：4l

输入第3节点data域值：6l

输入第4节点data域值：0l

输出一个单链表：

1 2 3 4 5 6

第3章 单 链 表

程序设计题典

数据结构程序设计题典 61

【例3.10】设计循环单链表结构并实现循环单链表的基本操作。

【解】 不带哨兵节点的循环单链表结构如图3.6所示。其中h为头节点指针，亦简称

为头指针。

图 3.6 不带哨兵节点的循环单链表结构

实现本题功能的clink.cpp文件如下（后面的有关循环单链表的算法设计都使用其中的

函数，其中的循环单链表都是不带哨兵节点的循环单链表）：

#include <iostream.h>

#include <malloc.h>

typedef int elemtype; //定义数据域的类型

typedef struct linknode //定义节点类型

{

elemtype data;

struct linknode \*next;

} nodetype;

nodetype \*cyccreate()

//创建循环单链表，用户输入各节点data域之值，0表示输入结束

{

elemtype d;

nodetype \*h=NULL，\*s，\*t=NULL;

int i=1;

cout << "建立一个循环单链表" << endl;

while (1)

{

cout << " 输入第" << i << "节点data域值：";

cin >> d;

if (d==0) break; //以0表示输入结束

if (i==1) //建立第一个节点

{

h=(nodetype \*)malloc(sizeof(nodetype));

第3章单 链 表

程序设计题典

62 数据结构程序设计题典

h->data=d;h->next=NULL; t=h;

}

else //建立其余节点

{

s=(nodetype \*)malloc(sizeof(nodetype));

s->data=d;s->next=NULL;t->next=s;

t=s; //t始终指向生成的循环单链表的最后一个节点

}

i++;

}

if (t!=NULL) t->next=h;

//将最后一不为NULL的节点的next置为h

return h;

}

void cycdisp(nodetype \*h)

//输出由h指向的循环单链表的所有data域之值

{

nodetype \*p=h;

cout << "输出一个循环单链表：" << endl << " ";

if (p==NULL)

{

cout << "空表" << endl; return;

}

while (p->next!=h)

{

cout << p->data << " "; p=p->next;

}

cout << p->data << " " << endl;

}

int cyclen(nodetype \*h) //返回循环单链表的长度

{

int i=1;

nodetype \*p=h;

if (p==NULL) return 0;

while (p->next!=h)

{

p=p->next;i++;

第3章 单 链 表

程序设计题典

数据结构程序设计题典 63

}

return i;

}

nodetype \*cycfind(nodetype \*h，int i)

//返回第i个节点的指针

{

nodetype \*p=h;

int j=1;

if (i>cyclen(h) || i<=0) return NULL; //i值上溢或下溢

else

{

if (i==1) return p; //i=1时返回头指针

while (p->next!=h && j<i) //查找第i个节点，并由p指向该节点

{

j++;p=p->next;

}

return p;

}

}

nodetype \*cycins(nodetype \*h，int i，elemtype x)

//在循环单链表head中第i个节点(i>=0)之后插入一个data域为x的节点

{

nodetype \*p，\*s，\*r;

s=(nodetype \*)malloc(sizeof(nodetype)); //创建节点s

s->data=x;s->next=NULL;

if (i==0) //i=0：s作为该循环单链表的第一个节点

{

r=h;

while (r->next!=h) r=r->next; //r为最后节点的指针

s->next=h;h=s;r->next=h; //修改r的next域

}

else

{

p=cycfind(h，i); //查找第i个节点，并由p指向该节点

if (p!=NULL) //将s插入到p之后

{

s->next=p->next;p->next=s;

第3章单 链 表

程序设计题典

64 数据结构程序设计题典

}

else cout << "输入的i值不正确" << endl;

}

return h;

}

nodetype \*cycdel(nodetype \*h，int i) //删除第i个节点

{

nodetype \*p=h，\*s，\*r;

int j=1;

if (i==1) //删除第1个节点

{

r=h;

while (r->next!=h) r=r->next; //r为最后节点的指针

h=h->next;r->next=h; //重置r的next域

free(p);

}

else

{

p=cycfind(h，i-1); //查找第i-1个节点，并由p指向该节点

if (p!=NULL && p->next!=h) //若存在要删除的节点

{

s=p->next; //s指向要删除的节点

p->next=s->next; free(s);

}

else cout << "输入的i值不正确" << endl;

}

return h;

}

void cycdispose(nodetype \*h) //释放循环单链表的所有节点占用的空间

{

nodetype \*pa=h，\*pb;

if (pa!=NULL)

{

pb=pa->next;

if (pb==h) //只有一个节点的情况

free(pa);

else

第3章 单 链 表

程序设计题典

数据结构程序设计题典 65

{

while (pb->next!=h) //有两个及以上节点的情况

{

free(pa); pa=pb; pb=pb->next;

}

free(pa);

}

}

}

【例3.11】假设在长度大于1的循环单链表中，既无头节点也无头指针，p为指向该

链表中某个节点的指针。设计一个算法删除该节点的前驱节点。

【解】 本题利用循环单链表的特点，通过p指针循环找到其前驱节点q及q的前驱节点

r，然后将其删除。实现本题功能的程序如下：

#include "clink.cpp"

nodetype \*delprev(nodetype \*p)

{

nodetype \*r=p，\*q=r->next;

while (q->next!=p) //查找p节点的前驱节点q，r指向q的前驱节点

{

r=r->next;q=r->next;

}

r->next=p; //删除q所指的节点

free(q);

return(p);

}

void main()

{

nodetype \*h;

int i;

h=cyccreate();

cout << "输入任意节点编号：";

cin >> i;

while (i-->1) h=h->next; //产生任一节点的指针h

cycdisp(h);

第3章单 链 表

程序设计题典

66 数据结构程序设计题典

h=delprev(h);

cycdisp(h);

}

本程序的一次执行结果如下：

建立一个循环单链表

输入第1节点data域值：2l

输入第2节点data域值：3l

输入第3节点data域值：4l

输入第4节点data域值：0l

输入任意节点编号：3l

输出一个循环单链表：

4 2 3

输出一个循环单链表：

4 2

【例3.12】已知一带有哨兵节点h的循环单链表中的数据元素含有三类字符（即字

母字符、数字字符和其他字符）。试编写一个算法，构造三个循环单链表，使每个循环单

链表中只含同一类的字符，且利用原表中的节点空间作为这三个表的节点空间（哨兵节点

可另辟空间）。

【解】 实现本题功能的程序如下：

说明：运行本程序时，需将clink.cpp文件中：

（1）第3行的“typedef int elemtype;”改为“typedef char elemtype;”

（2）cyccreate()函数中的“if (d==0) break;”改为“if (d=='0') break;”即在创建

一个循环单链表时，由用户输入各节点data域之值，以输入字符'0'表示输入结

束。

#include "clink.cpp"

void split(nodetype \*ha，nodetype \*hb，nodetype \*hc)

{

char c;

nodetype \*ra，\*rb，\*rc，\*p=ha->next;

ra=ha;ra->next=NULL;



第3章 单 链 表

程序设计题典

数据结构程序设计题典 67

rb=hb;rb->next=NULL;

rc=hc;rc->next=NULL;

while (p!=ha)

{

c=p->data;

if ((c>='a' && c<='z') || (c>='A' && c<='Z'))

{

ra->next=p;ra=p;

}

else if (c>='0' && c<='9')

{

rb->next=p;rb=p;

}

else

{

rc->next=p;rc=p;

}

p=p->next;

}

ra->next=ha;rb->next=hb;rc->next=hc; //变为循环单链表

}

void main()

{

nodetype \*ha，\*hb=NULL，\*hc=NULL，\*s，\*p;

ha=cyccreate();

p=ha;

while (p->next!=ha) p=p->next; //找ha的最后一个节点p

s=(nodetype \*)malloc(sizeof(nodetype)); //添加哨兵

s->next=ha;ha=s;

p->next=ha;

s=(nodetype \*)malloc(sizeof(nodetype)); //添加哨兵

s->next=hb;hb=s;

s=(nodetype \*)malloc(sizeof(nodetype)); //添加哨兵

s->next=hc;hc=s;

split(ha，hb，hc);

cycdisp(ha);

cycdisp(hb);

cycdisp(hc);

}

第3章单 链 表

程序设计题典

68 数据结构程序设计题典

本程序的一次执行结果如下：

建立一个循环单链表

输入第1节点data域值：al

输入第2节点data域值：5l

输入第3节点data域值：\*l

输入第4节点data域值：8l

输入第5节点data域值：bl

输入第6节点data域值：@l

输入第7节点data域值：0l

输出一个循环单链表：

a b

输出一个循环单链表：

5 8

输出一个循环单链表：

\* @

【例3.13】设一个环上有若干个整数，现采用循环单链表h存储该环，编写一个算

法，判断环上任意两个相邻元素值之差的绝对值是否超过2。

【解】 实现本题的程序如下：

#include "ch32.cpp"

#include <math.h>

int judge(nodetype \*h)

{

nodetype \*p=h;

while (1)

{

if (abs(p->data-p->next->data)<=2)

p=p->next;

else return 0;

if (p==h) break;

}

return 1;

}

第3章 单 链 表

程序设计题典

数据结构程序设计题典 69

void main()

{

nodetype \*h;

h=cyccreate();

cycdisp(h);

cout << "判断结果：" << judge(h) << endl;

}

本程序的一次执行结果如下：

建立一个循环单链表

输入第1节点data域值：2l

输入第2节点data域值：4l

输入第3节点data域值：6l

输入第4节点data域值：8l

输入第5节点data域值：0l

输出一个循环单链表：

2 4 6 8

判断结果：0

*3.3 习题实践*

【题1】有一个单链表(不同节点的数据域值可能相同)，其头指针为head，编写一个

函数，计算数据域为x的节点个数。

【题2】设计一个算法，判断一个单链表h的元素值是否是递增的。

【题3】有一个单链表，其节点只有data和next两个域，由于使用不当等原因，指向

第一个节点的指针丢失了，现要在指针p所指节点前面插入一个data域为x的节点，试编写

一个算法实现上述要求。

【题4】已知非空线性链表第一个节点由list指出，请写一算法，交换p所指的节点与

其下一个节点在链表中的位置(设p指向的不是链表的最后那个节点)。

【题5】编写一个算法，对值递增有序的单链表进行以下操作：若表中存在值为x的

节点，则将它从表中删除；否则，就往表中插入一个值为x的节点，并保持表的节点值递增

有序性质不变（假设表中没有值相同的元素）。

第3章单 链 表

程序设计题典

70 数据结构程序设计题典

【题6】从左到右及从右到左遍历一个单链表是可能的，其方法是在从左向右遍历的

过程中将连接方向逆转，如图3.7所示。在图中的指针p指向当前正在访问的节点，指针pr

指向指针p所指节点的左侧的节点。此时，指针p所指节点左侧的所有节点的连接方向都已

逆转。

（1）使用Pascal或C编写一个算法，从任一给定位置（pr，p）开始，将指针p右移一个

节点。如果p移出链表，则将p置为NULL，并让pr留在链表最右边的节点上。

（2）使用Pascal或C编写一个算法，从任一给定位置（pr，p）开始，将指针p左移一个

节点。如果p移出链表，则将p置为NULL，并让pr留在链表最左边的节点上。

图 3.7 一个单链表

【题7】现有两个带表头节点的单链表la和lb，此两个表本身无相同元素存在，la

表递增有序，lb表递减有序，且lb表中的元素包含在la表中连续的某一部分，如图3.8所示。

设计一个算法要找出这部分元素在la表中的位置，并将它们逆置，使之与lb表中的序相同。

图 3.8 两个单链表

【题8】编写一个程序，根据test\_data[]数组中给定的整数序列建立一个单链表，然

后对该单链表进行排序(元素值从小到大排列)，最后显示排序后的结果。

【题9】已知顺序表中有m个记录，表中记录不依关键字有序排列，编写算法为该顺

序表建立一个有序的索引表，索引表中的每一项含记录的关键字和该记录在顺序表中的序

号，要求算法的时间复杂度在最好的情况下能达到O(m)。

第3章 单 链 表

程序设计题典

数据结构程序设计题典 71

【题10】试编写对单链表实现2－路归并（两两归并）排序算法。

【题11】已知3个单链表A、B和C中的节点均依元素值自小至大非递减排列（可能

存在两个以上值相同的节点），编写算法对A链表进行如下操作：使操作后的链表A中仅留

下3个表中均包含的数据元素的节点，且没有值相同的节点，并释放所有无用节点。限定算

法的时间复杂度为O(m+n+p)，其中m、n和p分别为3个表的长度。

【题12】在下面给出的程序中，函数difference(A，B)用于求集合之差C=A-B，即当

且仅当e是A中一个元素，但不是B中的元素时，e是C中的一个元素。集合用有序链表实现，

用一个空链表表示一个空集合，表示非空集合的链表根据元素之值按递增排列。执行C=A-B

之后，表示集合A和B的链表不变，若结果集合C非空，则表示它的链表根据元素之值按递

增排列。函数append()用于在链表中添加节点。

#include <stdio.h>

struct node { int element;

struct node \*link;

};

typedef struct node NODE;

NODE A，B，\*C;

NODE \*append(NODE \*last，int e)

{

last->link=(NODE \*)malloc(sizeof(NODE));

last->link->element=e;

return (last->link);

}

NODE \*difference(NODE \*A，NODE \*B)

{

NODE C，last;

C=last=(NODE \*)malloc(sizeof(NODE));

while ( (A) )

if (A->element<B->element)

{

last=append(last，A->element);

A=A->link;

}

else if ( (B) )

{

A=A->link;

B=B->link;

第3章单 链 表

程序设计题典

72 数据结构程序设计题典

}

else (C) ;

while ( (D) )

{

last=append(last，A->element);

A=A->link;

}

(E) ;

last=C;

C=C->link;

free(last);

return(C);

}

【题13】假如以单链表表示集合(每个单链表中不存在重复的元素)，设计一个算法

求这两个集合的和。

【题14】设计算法将一个带头节点的单链表A分解为两个具有相同结构的链表B、C。

其中B表的节点是A表中值小于0的节点，而C表中的节点为A表中值大于0的节点（链表A的

元素类型为整型，要求B、C表示利用A表的节点）。

【题15】若有一个带哨兵节点h的单链表，节点的数据域值属于整型。现将其数据域

值除以3，得余数0，1和2。试按此3种不同的情况，把原有的链表分解成3个不同的单链表，

且只增设两个哨兵节点的空间，不允许另辟空间，并写出一个算法实现上述要求，且哨兵

节点的数据域记录该条链表中的数据节点个数。

【题16】给出用单链表存储多项式的结构，并编写函数：

（1）创建多项式链表的函数；

（2）实现两个多项式相加的函数。

【题17】有两个循环单链表，链头指针分别为head1和head2，编写一个函数将链表

head1链接到链表head2之后，链接后的链表仍保持循环链表形式。

【题18】（Josephus环）任给正整数n和k，按下述方法可得排列1，2，⋯，n的一个

置换：将数字1，2，⋯，n环形排列，按顺时针方向从1开始计数，计满k时输出该位置上的

数字（并从环中删除该数字），然后从下一个数字开始继续计数，直到环中所有数字被输

出为止。试编写一个算法，对输入的任意正整数n，k，输出相应的置换。

*3.4 参考答案*

【题1】本题是遍历通过该链表的每个节点，每遇到一个节点，节点个数加1，节点

第3章 单 链 表

程序设计题典

数据结构程序设计题典 73

个数存储在变量n中。实现本题功能的程序如下;

#include "slink.cpp"

int count(nodetype \*head，elemtype x)

{

nodetype \*p;

int n=0;

p=head;

while (p!=NULL)

{

if (p->data==x) n++;

p=p->next;

}

return(n);

}

void main()

{

nodetype \*h;

h=create();

disp(h);

cout << "单链表中2出现的次数：" << count(h，2) << endl;

}

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：1l

输入第3节点data域值：2l

输入第4节点data域值：5l

输入第5节点data域值：2l

输入第6节点data域值：0l

输出一个单链表：

2 1 2 5 2

单链表中2出现的次数：3

【题2】设计思想参见例3.4。实现本题功能的程序如下：

第3章单 链 表

程序设计题典

74 数据结构程序设计题典

#include "slink.cpp"

int order(nodetype \*h) //判断h单链表是否递增有序

{

nodetype \*p=h;

while (p->next!=NULL)

if (p->data<p->next->data)

p=p->next;

else

break;

if (p->next==NULL)

return 1;

else

return 0;

}

void main()

{

nodetype \*h;

h=create();

disp(h);

if (order(h)==1)

cout << "单链表递增有序" << endl;

else

cout << "单链表不是递增有序" << endl;

}

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：8l

输入第3节点data域值：25l

输入第4节点data域值：80l

输入第5节点data域值：0l

输出一个单链表：

2 8 25 80

单链表递增有序

第3章 单 链 表

程序设计题典

数据结构程序设计题典 75

【题3】由于单链表的第一个节点的指针不知道，无法找到p的前一个节点的指针，

所以不能直接插入一个节点，但可以在p之后插入一个节点s，然后交换p和s的data域，再将

p指向节点s，则s节点插入到p之前了。实现本题功能的算法如下：

void preins(nodetype \*p，elemtype x)

{

nodetype \*s;

s=(nodetype \*)malloc(sizeof(nodetype)); //创建一个新节点s

s->next=p->next; //将s插入到p的后面

p->next=s;

s->data=p->data; //交换p和s的data域，则x值的节点插入到p之前了

p->data=x;

p=s;

}

【题4】实现本题功能的程序如下：

#include "slink.cpp"

nodetype \*swap(nodetype \*list，nodetype \*p)

{

nodetype q，r;

if (list==p) //p为第一个节点的情况

{

q=p->next;

if (q!=NULL) //该链表有两个及两个以上节点时

{

list->next=q->next; //list所指节点与q所指节点交换

q->next=list;

list=q; //list仍作为第一个节点

}

else cout << "链表节点个数太少，不能交换" << endl;

}

else //p不为第一个节点时

{

r=list; //查找到p的前一个节点r

while (r->next!=p) r=r->next;

q=p->next;

if (q!=NULL) //p不是最后一个节点时

{

第3章单 链 表

程序设计题典

76 数据结构程序设计题典

r-next=q;p->next=q->next;q->next=p;

}

else cout << "p所指的节点为最后一个节点，不能交换" << endl;

}

return list;

}

【题5】实现本题功能的程序如下：

#include "slink.cpp"

nodetype \*delx(nodetype \*head，elemtype x)

{

nodetype \*p=head->next，\*r=head，\*s;

while (p!=NULL && p->data<x) //找到值域为x的节点p

{

r=p; //r指向p的前一个节点

p=p->next;

}

if (p!=NULL && p->data!=x) //若未找到且p不为空，将节点插入到

r与p之间

{

s=(nodetype \*)malloc(sizeof(nodetype));

s->data=x;

r->next=s;s->next=p;

}

else if (p==NULL)

//若未找到且p为空，将节点插到r之后

{

s=(nodetype \*)malloc(sizeof(nodetype));

s->data=x;s->next=NULL;

r->next=s;

}

else //找到了，删除之

{

r->next=p->next;

free(p);

}

return head;

}

第3章 单 链 表

程序设计题典

数据结构程序设计题典 77

void main()

{

nodetype \*h，\*s;

elemtype x;

h=create();

s=(nodetype \*)malloc(sizeof(nodetype)); //添加一个哨兵

s->next=h;h=s;

cout << "输入x值：";

cin >> x;

h=delx(h，x);

s=h;h=h->next;free(s); //删除哨兵

disp(h);

}

【题6】算法思想参见例3.3。(1)算法是从左向右遍历时将单链表逆转，对应的函数

right()如下：

#include "slink.cpp"

void right(nodetype \*pr，nodetype \*p)

{

nodetype \*q;

if (p!=0)

{

q=p; //q指向p节点

p=p->next; //p指针右移一个节点

q->next=pr; //将q的next域左置

pr=q; //pr指向q节点，即右移一个节点

}

}

（2）算法是从右向左遍历时将单链表逆转，对应的left()函数如下：

#include "slink.cpp"

void left(nodetype \*pr，nodetype \*p)

{

nodetype \*q;

if (pr!=0)

{

q=pr; //q指向pr节点

第3章单 链 表

程序设计题典

78 数据结构程序设计题典

pr=pr->next; //pr指针左移一个节点

q->next=p; //将q的next域右置

p=q; //p指向q节点，即左移一个节点

}

}

【题7】实现本题功能的程序如下：

#include "slink.cpp"

nodetype \*findplace(nodetype \*la，nodetype \*lb)

{

nodetype \*pa，\*pb=lb，\*ra，\*rb=lb;

while (rb->next!=NULL)

rb=rb->next; //找到lb的最后一个节点rb

pa=la->next;ra=la; //ra指向pa的前驱节点

while (pa!=NULL && pa->data!=rb->data)

//在la中找与pb相同的节点pa

{

ra=pa;

pa=pa->next;

}

cout << "ra=" << ra->data << endl;

ra->next=lb->next; //将lb链到ra之后

pb=lb->next;

while (pa!=NULL && pa->data!=pb->data)

pa=pa->next;

rb->next=pa->next;

return la->next;

}

void main()

{

nodetype \*la，\*lb，\*lc，\*t;

la=create();

lb=create();

disp(la);

disp(lb);

t=(nodetype \*)malloc(sizeof(nodetype)); //给la添加哨兵

t->next=la;la=t;

第3章 单 链 表

程序设计题典

数据结构程序设计题典 79

t=(nodetype \*)malloc(sizeof(nodetype)); //给lb添加哨兵

t->next=lb;lb=t;

lc=findplace(la，lb);

disp(lc);

}

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：3l

输入第3节点data域值：5l

输入第4节点data域值：8l

输入第5节点data域值：13l

输入第6节点data域值：21l

输入第7节点data域值：0l

建立一个单链表

输入第1节点data域值：13l

输入第2节点data域值：8l

输入第3节点data域值：5l

输入第4节点data域值：0l

输出一个单链表：

2 3 5 8 13 21

输出一个单链表：

13 8 5

ra=3

输出一个单链表：

2 3 13 8 5 21

【题8】本程序由两个函数组成：create()函数用于根据数组的元素建立一个单链表；

order()函数对该单链表排序，其设计思想参见例3.4。实现本题功能的程序如下：

#include <iostream.h>

#include <malloc.h>

typedef struct linknode

{

int data;

struct linknode \*next;

第3章单 链 表

程序设计题典

80 数据结构程序设计题典

} node;

node \*create(int a[]，int n) //由数组a创建一个单链表

{

node \*h，\*q;

for (h=NULL;n;n--)

{

q=(node \*)malloc(sizeof(node));

q->data=a[n-1];

q->next=h;

h=q;

}

return(h);

}

void disp(node \*h)

{

node \*p;

cout << " ";

for (p=h;p;p=p->next) cout << p->data << " ";

cout << endl;

}

node \*order(node \*h)

{

node \*head，\*p=h，\*q，\*r，\*t;

head=(node \*)malloc(sizeof(node)); //head为哨兵

head->data=0; //存放有序单链表的节点个数

head->next=NULL;

while (p!=NULL)

{

if (head->data==0) //有序单链表为空的情况

{

head->next=p;

q=p->next;

p->next=NULL;

p=q;

}

else

{

第3章 单 链 表

程序设计题典

数据结构程序设计题典 81

q=head->next;r=head;

while (q!=NULL && p->data>q->data)

//找到r节点，将p节点插入其后

{

r=q;

q=q->next;

}

t=p->next;

p->next=r->next;

r->next=p;

p=t;

}

head->data++;

}

t=head;head=head->next;free(t); //删除哨兵

return head;

}

int test\_data[]={5，9，3，1，2，7，8，6，4};

void main()

{

node \*h;

h=create(test\_data，sizeof(test\_data)/sizeof(int));

cout << "排序前：" << endl;

disp(h);

h=order(h);

cout << "排序后：" << endl;

disp(h);

}

本程序的一次执行结果如下：

排序前：

5 9 3 1 2 7 8 6 4

排序后：

1 2 3 4 5 6 7 8 9

【题9】实现本题功能的程序如下（order函数的设计思想参见例3.4）：

第3章单 链 表

程序设计题典

82 数据结构程序设计题典

#include "slink.cpp"

typedef struct node //定义索引表的节点类型

{

int no;

elemtype data;

struct node \*next;

} index;

index \*order(nodetype \*h) //建立索引表

{

nodetype \*p=h;

int n=1;

index \*head，\*q，\*r，\*s;

head=(index \*)malloc(sizeof(index)); //head为哨兵

head->data=0;head->next=NULL;

//data域存放索引表的节点个数

while (p!=NULL) //①

{

s=(index \*)malloc(sizeof(index)); //创建一个节点

s->data=p->data;s->no=n++;s->next=NULL;

p=p->next;

if (head->data==0) //索引表为空的情况

head->next=s;

else

{

q=head->next;r=head;

while (q!=NULL && s->data>q->data)

//找到r节点，将s节点插到其后②

{

r=q;

q=q->next;

}

s->next=r->next;

r->next=s;

}

head->data++;

}

s=head;head=head->next;free(s); //删除哨兵

return head;

}

void list(index \*h)

{

index \*p=h;

第3章 单 链 表

程序设计题典

数据结构程序设计题典 83

cout << "输出一个索引表" << endl << " ";

if (p==NULL)

{

cout << "空表" << endl;

return;

}

while (p!=NULL)

{

cout << "(" << p->data << "，" << p->no << ") ";

p=p->next;

}

cout << endl;

}

void main()

{

nodetype \*head;

index \*p;

head=create();

disp(head);

p=order(head);

list(p);

}

在最好的情况下（即原单链表中的元素递减有序），语句②每次只执行一次，这样order()

函数的时间复杂度为O(m)。

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：8l

输入第3节点data域值：3l

输入第4节点data域值：6l

输入第5节点data域值：1l

输入第6节点data域值：0l

输出一个单链表：

2 8 3 6 1

输出一个索引表

(1，5) (2，1) (3，3) (6，4) (8，2)

第3章单 链 表

程序设计题典

84 数据结构程序设计题典

【题10】与例3.6相同。

【题11】设计思想参见例3.7。先将单链表A中重复的节点删除，然后对单链表A的

每个节点：判断是否在单链表B或C中，若不在则保留；否则删除之。最后的单链表A即为

所求。实现本题功能的程序如下：

#include "slink.cpp"

nodetype \*pub(nodetype \*ha，nodetype \*hb，nodetype \*hc)

{

nodetype \*pa，\*pb，\*pc，\*q，\*r;

//为了算法简单，给A单链表添加一个哨兵

q=(nodetype \*)malloc(sizeof(nodetype));

q->next=ha;ha=q;

//在A单链表中删除元素值相同的节点

pa=ha->next;

while (pa!=NULL && pa->next!=NULL) //①

{

q=pa->next;

if (pa->data==q->data)

{

pa->next=q->next;

free(q);

}

pa=pa->next;

}

pa->next=NULL;

//下面开始对A单链表进行操作

pa=ha->next;

r=ha;r->next=NULL; //r始终指向新单链表的最后一个节点

pb=hb;pc=hc;

while (pa!=NULL) //查找均包含的节点②

{

//判断pa所指节点是否与hb中的节点值相等

while (pb!=NULL && pa->data>pb->data)

pb=pb->next;

//判断pa所指节点是否与hc中的节点值相等

while (pc!=NULL && pa->data>pc->data)

pc=pc->next;

第3章 单 链 表

程序设计题典

数据结构程序设计题典 85

if (pa->data==pb->data && pa->data==pc->data)

//pa所指节点是公共节点

{

r->next=pa;

r=pa;

pa=pa->next;

r->next=NULL;

}

else //pa所指节点不是公共节点，则删除

{

q=pa;

pa=pa->next; //pa移到下一个节点

free(q);

}

}

q=ha;ha=ha->next;free(q); //删除哨兵

return ha;

}

void main()

{

nodetype \*head1，\*head2，\*head3;

head1=create();

head2=create();

head3=create();

head1=pub(head1，head2，head3);

disp(head1);

dispose(head2); //释放无用节点

dispose(head3); //释放无用节点

}

分析算法的复杂度为：①循环语句的时间为m，②循环语句的时间为m+n+p，因为在

该循环中，A单链表遍历一次，B和C单链表也只遍历一次（它们均有序，由pb和pc保存位

置，不是每次都从头开始遍历）。故本算法的时间复杂度为O(m+n+p)。

本程序的一次执行结果如下：

第3章单 链 表

程序设计题典

86 数据结构程序设计题典

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：3l

输入第3节点data域值：3l

输入第4节点data域值：5l

输入第5节点data域值：0l

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：3l

输入第3节点data域值：9l

输入第4节点data域值：0l

建立一个单链表

输入第1节点data域值：1l

输入第2节点data域值：3l

输入第3节点data域值：5l

输入第4节点data域值：0l

输出一个单链表：

3

【题12】设计思想参见例3.7。append(last，e)函数的功能是在last所指节点之后添加

一个值域为e的节点，调用语句last=append(last，e)的作用是在新链表的最后一个节点last之

后添加一个值域为e的节点，而且last已指向该链表最后的一个节点。由于A和B链表均递增

有序，所以得到以下结果：

(A)：A!=NULL && B!=NULL //当A和B均不为空时循环

(B)：A->element==B->element //A、B所指节点的值域相等时

(C)：B=B->link //B指针后移一个节点

(D)：A!=NULL //当A不空时，将其后节点统统添加到C之后

(E)last->link=NULL //给C的最后一个节点的link域置空值

【题13】设计思想参见例3.7。先从B中删除与A重复的节点，然后将A与B链接起来

构成了两个集合的和。实现本题功能的程序如下：

#include "slink.cpp"

nodetype \*add(nodetype \*heada，nodetype \*headb)

{

nodetype \*p，\*q，\*r，\*s;

第3章 单 链 表

程序设计题典

数据结构程序设计题典 87

s=(nodetype \*)malloc(sizeof(nodetype));

//为了算法简单，给B创建一个哨兵

s->next=headb;

headb=s;

//从B中删除在A中的节点

p=headb->next;

r=headb;r->next=NULL; //r指向生成的单链表的最后一个节点

while (p!=NULL)

{

q=heada;

while (q!=NULL && q->data!=p->data) q=q->next;

if (q!=NULL) //若p节点在B中，则要删除之

{

s=p->next;

free(p);p=s;

}

else //若p节点不在B中，则链接到新单链表中

{

r->next=p;s=p->next;

r=p;r->next=NULL;

p=s;

}

}

s=headb;headb=headb->next;free(s); //删除哨兵节点

//将A与B链接起来即为集合的并

p=heada;

while (p->next!=NULL) p=p->next;

p->next=headb;

return heada;

}

void main()

{

nodetype \*ha，\*hb，\*hc;

ha=create(); //创建A集合

hb=create(); //创建B集合

hc=add(ha，hb);

cout << "A与B集合相加的结果" << endl;

disp(hc);

}

第3章单 链 表

程序设计题典

88 数据结构程序设计题典

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：3l

输入第3节点data域值：8l

输入第4节点data域值：4l

输入第5节点data域值：0l

建立一个单链表

输入第1节点data域值：1l

输入第2节点data域值：3l

输入第3节点data域值：4l

输入第4节点data域值：0l

A与B集合相加的结果

输出一个单链表：

2 3 8 4 1

【题14】设计思想参见例3.8。实现本题功能的程序如下：

#include "slink.cpp"

nodetype \*split(nodetype \*ha)

{

nodetype \*hb，\*p，\*ra，\*rb;

hb=(nodetype \*)malloc(sizeof(nodetype));

//给B添加哨兵，A已有哨兵

rb=hb;rb->next=NULL; //rb始终指向B的最后一个节点

p=ha->next;

ra=ha;ra->next=NULL; //ra始终指向新A的最后一个节点

while (p!=NULL)

{

if (p->data>0) //将p节点链接到B中

{

rb->next=p;rb=p;p=p->next;rb->next=NULL;

}

else //将p节点链接到A中

{

ra->next=p;ra=p;p=p->next;ra->next=NULL;

第3章 单 链 表

程序设计题典

数据结构程序设计题典 89

}

}

return hb;

}

void main()

{

nodetype \*heada，\*headb，\*s;

heada=create();

disp(heada);

s=(nodetype \*)malloc(sizeof(nodetype)); //给A添加哨兵

s->next=heada;

heada=s;

headb=split(heada);

s=heada;heada=heada->next;free(s); //删除A的哨兵

s=headb;headb=headb->next;free(s); //删除B的哨兵

disp(heada);

disp(headb);

}

本程序的一次执行结果如下：

建立一个单链表

输入第1节点data域值：2l

输入第2节点data域值：-2l

输入第3节点data域值：4l

输入第4节点data域值：-4l

输入第5节点data域值：8l

输入第6节点data域值：0l

输出一个单链表：

2 -2 4 -4 8

输出一个单链表：

-2 -4

输出一个单链表：

2 4 8

【题15】设计思想参见例3.8。实现本题功能的程序如下：

第3章单 链 表

程序设计题典

90 数据结构程序设计题典

#include "slink.cpp"

void split(nodetype \*ha，nodetype \*hb，nodetype \*hc)

{

nodetype \*ra，\*rb，\*rc，\*p=ha->next;

ra=ha;ra->next=NULL;

rb=hb;rb->next=NULL;

rc=hc;rc->next=NULL;

ha->data=0;hb->data=0;hc->data=0; //置初值

while (p!=NULL)

{

switch (p->data%3)

{

case 0：ra->next=p;ra=p;ha->data++;break;

case 1：rb->next=p;rb=p;hb->data++;break;

case 2：rc->next=p;rc=p;hc->data++;break;

}

p=p->next;

}

ra->next=NULL;rb->next=NULL;rc->next=NULL;

}

void main()

{

nodetype \*ha，\*hb=NULL，\*hc=NULL，\*s;

ha=create();

s=(nodetype \*)malloc(sizeof(nodetype)); //添加哨兵

s->next=ha;ha=s;

s=(nodetype \*)malloc(sizeof(nodetype)); //添加哨兵

s->next=hb;hb=s;

s=(nodetype \*)malloc(sizeof(nodetype)); //添加哨兵

s->next=hc;hc=s;

split(ha，hb，hc);

disp(ha->next);

disp(hb->next);

disp(hc->next);

}

第3章 单 链 表

程序设计题典

数据结构程序设计题典 91

【题16】用单链表存储多项式的结构如下：

struct pnode

{

int coef; /\*系数\*/

int exp; /\*指数\*/

struct pnode \*link;

}

假设输入一组多项式的系数和指数，以输入系数为0标志结束，为了简化本题算法，假

设在建立多项式链表时，总是按照指数从大到小顺序输入的。实现本题功能的程序如下，

其中（1）小题对应的函数为poly()，（2）小题对应的函数为padd()。

#include <iostream.h>

#include <malloc.h>

struct pnode

{

int coef; //系数

int exp; //指数

struct pnode \*link;

};

struct pnode \*poly()

{

struct pnode \*head，\*r，\*s;

int m，n，i=0;

cout << "建立一个多项式的单链表" << endl;

head=(struct pnode \*)malloc(sizeof(struct pnode));

//建立一个哨兵

r=head; //r始终指向最后一个节点

while (1)

{

cout << "输入第" << ++i << "个项式" << endl;

cout << " 系数n 指数m：";

cin >> n >> m;

if (n==0) break; //若n=0则退出while循环

s=(struct pnode \*)malloc(sizeof(struct pnode));

//建立一个新节点s

s->coef=n;s->exp=m;

r->link=s; //把s链接到r的后面

第3章单 链 表

程序设计题典

92 数据结构程序设计题典

r=s;

}

r->link=NULL; //将末尾节点的指针域置空

r=head;head=head->link;free(r); //删除哨兵

return(head);

}

void disp(pnode \*h)

{

pnode \*p=h;

cout << "输出多项式：" << endl;

while (p!=NULL)

{

cout << " " << p->coef << "，" << p->exp << endl;

p=p->link;

}

}

struct pnode \*padd(struct pnode \*heada，struct pnode \*headb)

{

struct pnode \*headc，\*p，\*q，\*r，\*s;

int x;

p=heada;q=headb;

headc=(struct pnode \*)malloc(sizeof(struct pnode));

r=headc; //r始终指向新链表的最后一个节点

while (p!=NULL && q!=NULL)

{

if (p->exp==q->exp)

//两指数相等时，将两系数相加生成一新节点插入C中

{

x=p->coef+q->coef;

if (x!=0)

{

s=(struct pnode \*)malloc(sizeof(struct pnode));

//创建一个节点

s->coef=x;s->exp=p->exp;

r->link=s;r=s;

}

p=p->link;q=q->link;

}

else if (p->exp<q->exp)

第3章 单 链 表

程序设计题典

数据结构程序设计题典 93

//两指数不等时， 复制较小系数的节点并插入C中

{

s=(struct pnode \*)malloc(sizeof(struct pnode));

s->coef=q->coef;s->exp=q->exp;

r->link=s;r=s;

q=q->link;

}

else

{

s=(struct pnode \*)malloc(sizeof(struct pnode));

s->coef=p->coef;s->exp=p->exp;

r->link=s;r=s;

p=p->link;

}

}

if (p!=NULL) q=p; //复制A的余下部分

while (q!=NULL) //复制余下部分

{

s=(struct pnode \*)malloc(sizeof(struct pnode));

//创建一个节点

s->coef=q->coef;s->exp=q->exp;

r->link=s; //把s链接到C中

r=s;q=q->link;

}

r->link=NULL; //最后节点的link域置空

s=headc; //删除C的头节点

headc=headc->link;

free(s);

return(headc);

}

void main()

{

pnode \*head1，\*head2，\*head3;

head1=poly();

head2=poly();

head3=padd(head1，head2);

disp(head3);

}

本程序的一次执行结果如下：

第3章单 链 表

程序设计题典

94 数据结构程序设计题典

建立一个多项式的单链表

输入第1个项式

系数n 指数m：3 2l

输入第2个项式

系数n 指数m：2 1l

输入第3个项式

系数n 指数m：8 0l

输入第4个项式

系数n 指数m：0 0l

建立一个多项式的单链表

输入第1个项式

系数n 指数m：2 3l

输入第2个项式

系数n 指数m：-2 1l

输入第3个项式

系数n 指数m：-5 1l

输入第4个项式

系数n 指数m：0 0l

输出多项式：

2，3

3，2

3，0

【题17】本题的算法思想是，先找到两链表的尾指针，将第一个链表的尾指针与第

二个链表的头节点链接起来，再使之成为循环链表。实现本题功能的程序如下：

#include "clink.cpp"

nodetype \*link(nodetype \*head1，nodetype \*head2)

{

nodetype \*p，\*q;

p=head1; //找到head1的表尾，用p指向它

while (p->next!=head1) p=p->next;

q=head2; //找到head2的表尾，用q指向它

while (q->next!=head2) q=q->next;

p->next=head2; //将head2链表链接到head1链表之后

q->next=head1; //仍保持是循环链表

return(head1);

}

第3章 单 链 表

程序设计题典

数据结构程序设计题典 95

void main()

{

nodetype \*h1，\*h2;

h1=cyccreate();

h2=cyccreate();

h1=link(h1，h2);

cycdisp(h1);

}

本程序的一次执行结果如下：

建立一个循环单链表

输入第1节点data域值：1l

输入第2节点data域值：2l

输入第3节点data域值：3l

输入第4节点data域值：0l

建立一个循环单链表

输入第1节点data域值：4l

输入第2节点data域值：5l

输入第3节点data域值：6l

输入第4节点data域值：0l

输出一个循环单链表：

1 2 3 4 5 6

【题18】先根据正整数n构造一个循环单链表，各节点data域分别为1，2，⋯，n（由

crea函数实现），然后在该循环单链表中每相差k个节点删除一个节点，并输出该节点的data

域值（由jese函数实现）。实现本题功能的程序如下：

#include "clink.cpp"

nodetype \*crea(int n)

{

nodetype \*s，\*r，\*h;

int i;

for (i=1;i<=n;i++)

{

s=(nodetype \*)malloc(sizeof(nodetype));

s->data=i;s->next=NULL;

if (i==1) h=s; //为第一个节点的情况

第3章单 链 表

程序设计题典

96 数据结构程序设计题典

else r->next=s;

r=s; //r始终指向最后一个节点

}

r->next=h; //产生循环单链表

return h;

}

void jese(nodetype \*h，int k)

{

nodetype \*p=h，\*q;

int i;

cout << "输出一个置换：\n ";

while (p->next!=p)

{

for (i=1;i<k-1;i++)

p=p->next;

if (p->next!=p)

{

q=p->next; //删除并输出q所指的节点

cout << q->data << " ";

p->next=q->next;

free(q);

}

p=p->next;

}

cout << p->data << endl; //输出最后一个节点

}

void main()

{

int n，k;

nodetype \*h;

cout << "输入n：";

cin >> n;

cout << "输入k：";

cin >> k;

h=crea(n);

cycdisp(h);

jese(h，k);

}

第3章 单 链 表

程序设计题典

数据结构程序设计题典 97

本程序的一次执行结果如下：

输入n：10l

输入k：3l

输出一个循环单链表：

1 2 3 4 5 6 7 8 9 10

输出一个置换：

3 6 9 2 7 1 8 5 10 4

**第4 章双链表**

第4章 双 链 表

程序设计题典

数据结构程序设计题典 99

*4.1 本章摘要*

*4.1.1 知识点*

*1. 双链表的结构*

· 带头节点（哨兵）的双链表

· 不带头节点的双链表

*2. 双链表的基本运算*

· 创建一个双链表

· 输出一个双链表

· 求双链表的长度

· 查找第i个节点

· 插入一个节点

· 删除一个节点

· 释放双链表占用的空间

*4.1.2 内容概要*

*1. 双链表的存储结构*

双链表的每个节点除包括一个数据域外还包括两个指针域，一个指向后继，另一个指

向前驱。最后一个节点的后继指针指向表头节点。

双链表节点的类型定义如下：

typedef int elemtype; //定义数据域的类型

typedef struct linknode { //定义节点类型

elemtype data;

int freq; //该域在后面有的程序中使用

struct linknode \*left，\*right;

} dnodetype;

第4章双 链 表

程序设计题典

100 数据结构程序设计题典

循环双链表与双链表基本相同，每个节点也包括数据域和两个指针域。第一个节点的

left域指向最后一个节点，最后一个节点的right域指向第一个节点。

存储双链表时，可以带一个头节点即哨兵，也可以不带头节点。对于带头节点的双链

表，头节点不计在双链表的节点个数之内，这样多占用一个节点的存储空间，但在实现插

入和删除节点运算时更方便。

*2. 双链表的运算*

· create()：创建一个双链表，由用户输入各节点data域的值，以0表示输入结束。最

后返回建立的双链表的头指针。

· disp(dnodetype \*h)：输出双链表h的所有节点的data域的值。

· len(dnodetype \*h)：返回双链表h的长度。

· find(dnodetype \*h，int i)：返回双链表h的第i个节点的指针。

· ins(dnodetype \*h，int i，elemtype x)：在双链表h中第i个节点(i>=0)之后插入一个data

域为x的节点。返回插入节点的双链表h的头指针。

· del(dnodetype \*h，int i)：删除双链表h中第i个节点，返回删除节点后的双链表的头

指针。

· dispose(dnodetype \*h)：释放双链表h的所有节点占用的存储空间。

*4.2 例题解析*

【例4.1】设计双链表结构并实现双链表的基本运算。

【解】 不带哨兵节点的双链表结构如图4.1所示，其中h为头节点指针，简称为头指

针。

图 4.1 双链表存储结构

实现本题功能的dlink.cpp文件如下（后面的有关双链表的算法设计都使用其中的函数，

其中的双链表都是不带哨兵节点的双链表）：

第4章 双 链 表

程序设计题典

数据结构程序设计题典 101

#include <iostream.h>

#include <malloc.h>

typedef int elemtype; //定义数据域的类型

typedef struct linknode //定义节点类型

{

elemtype data;

// int freq;

struct linknode \*left，\*right;

} dnodetype;

dnodetype \*create()

//创建一个双链表，由用户输入各节点data域之值，以0表示输入结束

{

elemtype d;

dnodetype \*h=NULL，\*s，\*t;

int i=1;

cout << "建立一个双链表" << endl;

while (1)

{

cout << " 输入第" << i << "节点data域值：";

cin >> d;

if (d==0) break; //以0表示输入结束

if (i==1) //建立第一个节点

{

h=(dnodetype \*)malloc(sizeof(dnodetype));

h->data=d;h->left=NULL;h->right=NULL;

t=h;

}

else //建立其余节点

{

s=(dnodetype \*)malloc(sizeof(dnodetype));

s->data=d;s->right=NULL;

t->right=s;s->left=t;

t=s; //t始终指向生成的双链表的最后一个节点

}

i++;

}

return h;

第4章双 链 表

程序设计题典

102 数据结构程序设计题典

}

void disp(dnodetype \*h) //输出由h指向的双链表的所有data域之值

{

dnodetype \*p=h;

cout << "输出一个双链表：" << endl << " ";

if (p==NULL) cout << "空表";

while (p!=NULL)

{

cout << p->data << " ";

p=p->right;

}

cout << endl;

}

int len(dnodetype \*h) //返回双链表的长度

{

int i=0;

dnodetype \*p=h;

if (p==NULL) return 0;

while (p!=NULL)

{

p=p->right;

i++;

}

return i;

}

dnodetype \*find(dnodetype \*h，int i) //返回第i个节点的指针

{

dnodetype \*p=h;

int j=1;

if (i>len(h) || i<=0) return NULL;

else

{

while (p!=NULL && j<i) //查找第i个节点，并由p指向该节点

{

j++;p=p->right;

}

return p;

第4章 双 链 表

程序设计题典

数据结构程序设计题典 103

}

}

dnodetype \*ins(dnodetype \*h，int i，elemtype x)

//在双链表head中第i个节点(i>=0)之后插入一个data域为x的节点

{

dnodetype \*p，\*s;

s=(dnodetype \*)malloc(sizeof(dnodetype)); //创建节点s

s->data=x;s->left=NULL;s->right=NULL;

if (i==0) //i=0表示将s作为该双链表的第一个节点

{

s->right=h;h->left=s;h=s;

}

else

{

p=find(h，i); //查找第i个节点，并由p指向该节点

if (p!=NULL && p->right!=NULL)

{

s->right=p->right;p->right->left=s;//将s插入到p之后

s->left=p;p->right=s;

}

else if (p!=NULL && p->right==NULL) //s作为最后节点

{

s->left=p;p->right=s;

}

else cout << "输入的i值不正确" << endl;

}

return h;

}

dnodetype \*del(dnodetype \*h，int i) //删除第i个节点

{

dnodetype \*p=h;

int j=1;

if (i==1) //删除第1个节点

{

第4章双 链 表

程序设计题典

104 数据结构程序设计题典

h=h->right;h->left=NULL;

free(p);

}

else

{

p=find(h，i); //查找第i个节点，并由p指向该节点

if (p!=NULL && p->right!=NULL) //p不为最后一个节点

{

p->left->right=p->right;

p->right->left=p->left;

free(p);

}

else if (p!=NULL && p->right==NULL) //p为最后一个节点

{

p->left->right=NULL;

free(p);

}

else cout << "输入的i值不正确" << endl;

}

return h;

}

void dispose(dnodetype \*h)

//释放双链表的所有节点占用的空间

{

dnodetype \*pa=h，\*pb;

if (pa!=NULL)

{

pb=pa->right;

if (pb==NULL) //只有一个节点的情况

free(pa);

else

{

while (pb!=NULL) //有两个及以上节点的情况

{

free(pa);

pa=pb;

pb=pb->right;

第4章 双 链 表

程序设计题典

数据结构程序设计题典 105

}

free(pa);

}

}

}

【例4.2】设有一个双链表，每个节点中除data，left和right三个域外，还有一个访问

频度域freq，在链表被启用之前，其值均初始化为0。每当在双链表上进行一次locate(h，x)

运算时，令元素值为x的节点中freq域的值增1，并使此链表中节点保持按访问频度递减的顺

序排列，以便使频繁访问的节点总是靠近表头。试编写实现符合上述要求的locate运算的算

法。

【解】 在dlink.cpp中在dnodetype类型的定义中添加int freq域，给该域初始化为0。在

每次查找到一个节点时，将其freq域增1，再与它前面的一个节点进行比较，若它的freq域

值较大，则两者交换，如此找一个合适的位置。实现本题功能的程序如下：

#include "dlink.cpp"

int locate(dnodetype \*h，elemtype x)

{

dnodetype \*p=h->right，\*q;

while (p!=NULL && p->data!=x) p=p->right;

//找data域值为x的节点p

if (p==NULL) return 0; //未找到的情况

else //找到的情况

{

p->freq++; //频度增1

q=p->left; //q为p左边节点

while (q!=h && q->freq<p->freq)

{

p->left=q->left;p->left->right=p; //交换p和q的位置

q->right=p->right;

if (q->right!=NULL) //p不为最后一个节点时

q->right->left=q;

p->right=q;q->left=p;

q=p->left; //q重指向p的左节点

}

return 1;

}

第4章双 链 表

程序设计题典

106 数据结构程序设计题典

}

void main()

{

dnodetype \*h，\*p，\*s;

elemtype x;

int ans;

h=create();

s=(dnodetype \*)malloc(sizeof(dnodetype));

s->data=-1;s->left=NULL; //为了算法简便，给双链表添加一个哨兵

s->right=h;h->left=s;

h=s;

p=h;

while (p!=NULL) //将各节点freq域初始化为0

{

p->freq=0;

p=p->right;

}

while (1)

{

cout << "输入x值：";

cin >> x;

if (locate(h，x)==0)

cout << "未找到x的节点" << endl;

cout << "链表重排后=>";

disp(h);

cout << "继续查找吗?(1继续，其他退出)：";

cin >> ans;

if (ans!=1) break;

}

}

本程序的一次执行结果如下（其中输出双链表时的-1是哨兵节点的值，在验证结果时

应忽略它）：

第4章 双 链 表

程序设计题典

数据结构程序设计题典 107

建立一个双链表

输入第1节点data域值：1l

输入第2节点data域值：2l

输入第3节点data域值：3l

输入第4节点data域值：0l

输入x值：3l

链表重反排后=>输出一个双链表：

-1 3 1 2

继续查找吗？（1继续，其他退出）：1l

输入x值：2l

链表重排后=>输出一个双链表：

-1 3 2 1

继续查找吗？（1继续，其他退出）：1l

输入x值：1l

链表重排后=>输出一个双链表：

-1 3 2 1

继续查找吗？（1继续，其他退出）：1l

输入x值：1l

链表重排后=>输出一个双链表：

-1 1 3 2

继续查找吗？（1继续，其他退出）：0l

【例4.3】假设在算法描述语言中引入指针的二元运算“异或”(用“⊙”表示)，若

a和b为指针，则a⊙b的运算结果仍为原指针类型，且：

a⊙(a⊙b)=(a⊙a)⊙b=b

(a⊙b)⊙b=a⊙(b⊙b)=a

则可利用一个指针域来实现双链表。每个节点有两个域：data域和link域，link域存放

该节点前驱与后继节点指针(不存在时为NULL)的异或。若设指针h指向链表中的第一个节

点，e指向链表中最后一个节点，则可实现从前向后或从后向前遍历此双向链表(这种链表

亦称为对称表)。设计一个算法从前向后输出该链表中各元素的值。

【解】 依题意，定义该链表的结构如下：

第4章双 链 表

程序设计题典

108 数据结构程序设计题典

typedef struct dnode

{

int data;

struct dnode \*link;

} dlist;

现在这样的链表己建立好了，且头指针为h，尾指针为e。对其中的一个p指向的节点，

假设其前驱节点指针为pre，其后继节点指针为next，那么有：

p->link=pre⊙next

也就是说，当p和其前驱节点指针pre己知时，可以计算出p的后继节点指针：

pre⊙p->link=pre⊙(pre⊙next)=(pre⊙pre)⊙next=next

即：

next=pre⊙p->link

这说明，第三个节点的指针是通过第一个节点的指针与第二个节点的link域“异或”得

到的。

同样，当p和其后继节点next己知时，可以计算出p的前驱节点指针pre：

p->link⊙next=(pre⊙next)⊙next=pre⊙(next⊙next)=pre

即：

pre=p->link⊙next

同样说明，第一个节点指针是通过第二个节点的link域与第三个节点的指针“异或”得

到的。

根据这个原理得到实现本题功能的程序如下：

#include <iostream.h>

#include <malloc.h>

typedef struct dnode

{

int data;

struct dnode \*link;

} dlist;

dlist \*xor(dlist \*p1，dlist \*p2)

//在C/C++中的异或运算符^不能对地址进行异或运算，所以先将地址值转换为长

//整型，然后进行异或运算，再将结果强制转换成地址。这就是本函数的功能

第4章 双 链 表

程序设计题典

数据结构程序设计题典 109

{

int add1，add2，add3;

add1=(long)p1;

add2=(long)p2;

add3=add1^add2;

return (dlist \*)add3;

}

dlist \*create(dlist \*\*e)

{

int i=1，x;

dlist \*head，\*r，\*s，\*pre;

cout << "创建一个双链表(以0结束)" << endl;

while (1)

{

cout << " 输入第" << i << "节点值：";

cin >> x;

if (x==0) //生成最后一个节点的link值后退出循环

{

pre->link=xor(r，NULL);

//将s->link置为前后节点地址之异或

\*e=pre;

break;

}

s=(dlist \*)malloc(sizeof(dlist)); //创建一个节点

s->data=x;

if (i==1) //是第一个节点的情况

{

pre=head=s;r=NULL; //r为当前节点的前一个节点

}

else

{

pre->link=xor(r，s);

//将s->link置为前后节点地址之异或

r=pre;

pre=s;

}

i++;

}

return head;

}

第4章双 链 表

程序设计题典

110 数据结构程序设计题典

void order(dlist \*h，dlist \*e)

{

dlist \*pre=NULL，\*pre1，\*p=h;

cout << "遍历节点序列：";

if (h==NULL) cout << "空表" << endl;

else

{

while (p!=e) //遍历最后一节点前的所有节点

{

cout << p->data << " ";

pre1=p;

p=xor(pre，p->link); //为下一个节点的地址

pre=pre1;

}

cout << e->data << endl; //遍历最后一节点

}

}

void main()

{

dlist \*h，\*e;

int i;

h=create(&e);

cout << "从左向右";

order(h，e);

cout << "从右向左";

order(e，h);

}

本程序的一次执行结果如下：

创建一个双链表（以0结束）

输入第1节点值：3l

输入第2节点值：5l

输入第3节点值：8l

输入第4节点值：2l

输入第5节点值：6l

输入第6节点值：0l

从左向右遍历节点序列：3 5 8 2 6

从右向左遍历节点序列：6 2 8 5 3

第4章 双 链 表

程序设计题典

数据结构程序设计题典 111

【例4.4】实现循环双链表的基本运算。

【解】 不带哨兵节点的循环双链表结构如图4.2所示，其中h为头节点指针，简称为

头指针。

图 4.2 循环双链表存储结构

实现本题功能的cdlink.cpp文件如下（后面的有关循环双链表的算法设计都使用其中的

函数，其中的循环双链表都是不带哨兵节点的循环双链表）：

#include <iostream.h>

#include <malloc.h>

typedef int elemtype; //定义数据域的类型

typedef struct linknode //定义节点类型

{

elemtype data;

int freq;

struct linknode \*left，\*right;

} dnodetype;

dnodetype \*cyccreate()

//创建一个循环双链表，由用户输入各节点data域之值，以0表示输入结束

{

elemtype d;

dnodetype \*h=NULL，\*s，\*t;

int i=1;

cout << "建立一个循环双链表" << endl;

while (1)

{

cout << " 输入第" << i << "节点data域值：";

cin >> d;

if (d==0) break; //以0表示输入结束

if (i==1) //建立第一个节点

第4章双 链 表

程序设计题典

112 数据结构程序设计题典

{

h=(dnodetype \*)malloc(sizeof(dnodetype));

h->data=d;h->left=NULL;h->right=NULL;

t=h;

}

else //建立其余节点

{

s=(dnodetype \*)malloc(sizeof(dnodetype));

s->data=d;s->right=NULL;

t->right=s;s->left=t;

t=s; //t始终指向生成的循环双链表的最后一个节点

}

i++;

}

if (h!=NULL) //不为空链表时将首尾指针链起来

{

h->left=t;t->right=h;

}

return h;

}

void cycdisp(dnodetype \*h)

//输出由h指向的循环双链表的所有data域的值

{

dnodetype \*p=h;

cout << "输出一个循环双链表：" << endl << " ";

if (p==NULL) cout << "空表";

else

{

while (p->right!=h) //访问最后一个节点之前的节点

{

cout << p->data << " ";

p=p->right;

}

cout << p->data << endl; //访问最后一个节点

}

}

int cyclen(dnodetype \*h) //返回循环双链表的长度

第4章 双 链 表

程序设计题典

数据结构程序设计题典 113

{

int i=1;

dnodetype \*p=h;

if (h==NULL) return 0; //空表长度为0

while (p->right!=h)

{

p=p->right;

i++;

}

return i;

}

dnodetype \*cycfind(dnodetype \*h，int i) //返回第i个节点的指针

{

dnodetype \*p=h;

int j=1;

if (i>cyclen(h) || i<=0) return NULL;

else

{

while (p->right!=h && j<i)//查找第i个节点，并由p指向该节点

{

j++;p=p->right;

}

return p;

}

}

dnodetype \*cycins(dnodetype \*h，int i，elemtype x)

//在循环双链表head中第i个节点(i>=0)之后插入一个data域为x的节点

{

dnodetype \*p，\*s;

s=(dnodetype \*)malloc(sizeof(dnodetype)); //创建节点s

s->data=x;s->left=NULL;s->right=NULL;

if (i==0) //i=0表示将s作为该循环双链表的第一个节点

{

s->right=h;s->left=h->left;

h->left->right=s;h->left=s;h=s;

}

else

第4章双 链 表

程序设计题典

114 数据结构程序设计题典

{

p=cycfind(h，i); //查找第i个节点，并由p指向该节点

if (p!=NULL && p->right!=h)

{

s->right=p->right;p->right->left=s;

//将s插入到p之后

s->left=p;p->right=s;

}

else if (p!=NULL && p->right==h) //s作为最后节点

{

s->left=p;s->right=h;

p->right=s;h->left=s;

}

else cout << "输入的i值不正确" << endl;

}

return h;

}

dnodetype \*cycdel(dnodetype \*h，int i) //删除第i个节点

{

dnodetype \*p=h;

int j=1;

if (i==1) //删除第1个节点

{

h=h->right;h->left=p->left;

h->left->right=h;

free(p);

}

else

{

p=cycfind(h，i); //查找第i个节点，并由p指向该节点

if (p!=NULL && p->right!=h) //p不为最后节点

{

p->left->right=p->right;

p->right->left=p->left;

free(p);

}

else if (p!=NULL && p->right==h) //p为最后一个节点

第4章 双 链 表

程序设计题典

数据结构程序设计题典 115

{

p->left->right=h;h->left=p->left;

free(p);

}

else cout << "输入的i值不正确" << endl;

}

return h;

}

void cycdispose(dnodetype \*h)

//释放循环双链表的所有节点占用的空间

{

dnodetype \*pa=h，\*pb;

if (pa!=NULL)

{

pb=pa->right;

if (pb==h) //只有一个节点的情况

free(pa);

else

{

while (pb!=h) //有两个及以上节点的情况

{

free(pa);

pa=pb;

pb=pb->right;

}

free(pa);

}

}

}

【例4.5】假设一个循环单链表，其节点有left，data和right两个域，其中data为数据

域，存放元素的有效信息，right域为指针域，它指向后继节点，left为指针域，它的值为NULL。

编写一个算法将此表改为循环双链表。

【解】 本题算法是依次从左向右遍历每个节点，并设置每个节点的left值。实现本题

功能的程序如下：

第4章双 链 表

程序设计题典

116 数据结构程序设计题典

#include "cdlink.cpp"

dnodetype \*trans(dnodetype \*h)

{

dnodetype \*p，\*q=h;

if (h==NULL) return NULL;

p=q->right; //p为q的后继节点

while (p!=h) //依次从左向右通过每个节点，设置每个节点的left值

{

p->left=q;

p=p->right;

q=p;

}

h->left=q; //此时q为最后一个节点

return(h);

}

void main()

{

dnodetype \*h，\*p;

h=cyccreate(); //创建一个循环双链表

cycdisp(h);

if (h!=NULL)

{

h->left=NULL; //将循环双链表的所有节点left值置为NULL

p=h->right;

while (p!=h)

{

p->left=NULL;

p=p->right;

}

}

h=trans(h);

cycdisp(h);

}

第4章 双 链 表

程序设计题典

数据结构程序设计题典 117

*4.3 习题实践*

【题1】编写一个算法将双链表中最后两个节点互换。

【题2】采用例4.3中所述的存储结构，设计一个算法在第i个节点之后插入一个data

域为x的节点（i=0表示在第一个节点之前插入新节点）。

【题3】采用例4.3中所述的存储结构，设计一个算法删除这样的双链表中第i个节点

（当i=1时表示删除头节点）。

【题4】已知一个循环双链表，设计一个算法将其中所有的节点逆置。

【题5】设有一个循环双链表，假设所有节点的data域值不相同。编写一个算法将data

域值为x的节点与其右边的一个节点进行交换。

*4.4 参考答案*

【题1】先找到最后的两个节点，p指向最后一个节点，q指向倒数第二个节点。再考

虑该链表只有两个节点和两个以上节点的情况下，p和q节点的互换。

实现本题功能的程序如下：

#include "dlink.cpp"

dnodetype \*swap(dnodetype \*h)

{

dnodetype \*p=h，\*q;

if (len(p)<2)

{

cout << "双链表的节点少于2个" << endl;

return NULL;

}

while (p->right!=NULL) p=p->right; //p指向最后一个节点

q=p->left; //q指向倒数第二个节点

if (q==h) //只有两个节点的情况

{

p->right=q;q->left=p;

p->left=NULL;q->right=NULL;

h=p;

}

第4章双 链 表

程序设计题典

118 数据结构程序设计题典

else

{

q->left->right=p;p->left=q->left;

p->right=q;q->left=p;q->right=NULL;

}

return h;

}

void main()

{

dnodetype \*h;

h=create();

disp(h);

h=swap(h);

disp(h);

}

本程序的一次执行结果如下：

建立一个双链表

输入第1节点Data域值：1l

输入第2节点Data域值：2l

输入第3节点Data域值：3l

输入第4节点Data域值：4l

输入第5节点Data域值：0l

输出一个双链表：

1 2 3 4

输出一个双链表：

1 2 4 3

【题2】根据例4.3的分析，插入一个节点的函数应修改其前驱节点和后继节点的link

域以及计算该节点的link域，实现本题功能的程序如下，其中的create()，order()和xor()函数

是例4.3程序中的函数：

void ins(dlist \*\*h，dlist \*\*e，int i，int x)

{

dlist \*s，\*p，\*pre，\*pre1，\*next，\*next1;

int j;

s=(dlist \*)malloc(sizeof(dlist)); //创建一个待插入的节点

s->data=x;

第4章 双 链 表

程序设计题典

数据结构程序设计题典 119

if (i==0) //在第1个节点前插入s

{

s->link=xor(NULL，\*h);

next=xor(NULL，(\*h)->link); //next指向原链表h的后继节点

(\*h)->link=xor(s，next);

\*h=s; //修改头节点h

}

else

{

pre=NULL;p=\*h;j=1;

while (j<i && p!=(\*e)) //查找第i个节点p，其前驱节点为pre

{

j++;

pre1=p;

p=xor(pre，p->link);

pre=pre1;

}

if (p==(\*e) && j<i)

cout << "i参数溢出" << endl;

else if (p==(\*e) && j==i) //p为最后节点，在其后插入s

{

s->link=xor(p，NULL); //在p之后插入s

p->link=xor(pre，s); //修改p的link域

\*e=s;

}

else //在中间插入节点s

{

next=xor(pre，p->link); //p之后的节点

next1=xor(p，next->link); //next之后的节点

s->link=xor(p，next); //在p之后插入s

p->link=xor(pre，s); //修改p的link域

next->link=xor(s，next1); //修改next的link域

}

}

}

void main()

{

第4章双 链 表

程序设计题典

120 数据结构程序设计题典

dlist \*h，\*e;

int i;

h=create(&e);

order(h，e);

cout << "输入插入10的位置i：";

cin >> i;

cout << "插入之后";

ins(&h，&e，i，10);

order(h，e);

}

本程序的一次执行结果如下：

创建一个双链表（以0结束）

输入第1节点值：1l

输入第2节点值：3l

输入第3节点值：5l

输入第4节点值：2l

输入第5节点值：6l

输入第6节点值：0l

遍历节点序列：1 3 5 2 6

输入插入10的位置i：3l

插入之后遍历节点序列：1 3 5 10 2 6

【题3】根据例4.3的分析，插入一个节点的函数应修改其前驱节点和后继节点的link

域以及计算该节点的link域。实现本题功能的函数如下，其中的create()，order()和xor()函数

是例4.3程序中的函数：

void del(dlist \*\*h，dlist \*\*e，int i)

{

dlist \*p，\*pre，\*pre1，\*next，\*next1;

int j;

if (i==1) //删除第1个节点的情况

{

next=xor(NULL，(\*h)->link); //next指向原链表h的后继节点

if (next==NULL) //只有一个节点的情况

{

free(\*h);

\*h=NULL;\*e=NULL; //修改头节点h和尾节点e

第4章 双 链 表

程序设计题典

数据结构程序设计题典 121

}

else

{

p=xor(\*h，next->link); //p指向第3个节点，可能为NULL

next->link=(NULL，p);

//修改第2个节点的link域，使其成为头节点

free(\*h);

\*h=next; //修改头指针h

}

}

else //删除非第1个节点的情况

{

pre=NULL;p=\*h;j=1;

while (j<i && p!=(\*e)) //查找第i个节点p，其前驱节点为pre

{

j++;

pre1=p;

p=xor(pre，p->link);

pre=pre1;

}

if (p==(\*e) && j<i) cout << "i参数溢出!" << endl;

else

{

next=xor(pre，p->link); //next指向p的后继节点

if (next==NULL) //p是最后一个节点

{

pre1=xor(pre->link，p); //pre1指向pre的前驱节点

pre->link=xor(pre1，NULL); //修改pre的link域

\*e=pre;

free(p);

}

else //p不是最后一个节点的情况

{

next1=xor(p，next->link); //next1指向next的后继节点

next->link=xor(pre，next1);

pre1=xor(pre->link，p); //pre1指向pre的前驱节点

pre->link=xor(pre1，next); //修改pre的link域

free(p);

}

}

第4章双 链 表

程序设计题典

122 数据结构程序设计题典

}

}

void main()

{

dlist \*h，\*e;

int i;

h=create(&e);

order(h，e);

cout << "输入要删除的位置i：";

cin >> i;

cout << "删除之后";

del(&h，&e，i);

order(h，e);

}

本程序的执行结果如下：

创建一个双链表（以0结束）

输入第1节点值：1l

输入第2节点值：5l

输入第3节点值：2l

输入第4节点值：4l

输入第5节点值：9l

输入第6节点值：0l

遍历节点序列：1 5 2 4 9

输入要删除的位置i：2l

删除之后遍历节点序列：1 2 4 9

【题4】实现本题功能的程序如下：

#include "cdlink.cpp"

dnodetype \*invert(dnodetype \*h)

{

dnodetype \*p=h，\*q;

if (p!=NULL && p!=p->right) //有两个及以上节点

{

q=p->right;

h->right=h->left; //逆置头节点

h->left=p->right;

while (q!=h)

{

第4章 双 链 表

程序设计题典

数据结构程序设计题典 123

q->left=q->right;q->right=p;

p=q;q=q->left;

}

h=h->right; //h指向最后一个节点

}

return h;

}

void main()

{

dnodetype \*h;

h=cyccreate();

cycdisp(h);

h=invert(h);

cycdisp(h);

}

本程序的一次执行结果如下：

建立一个循环双链表

输入第1节点data域值：1l

输入第2节点data域值：2l

输入第3节点data域值：3l

输入第4节点data域值：4l

输入第5节点data域值：5l

输入第6节点data域值：0l

输出一个循环双链表：

1 2 3 4 5

输出一个循环双链表：

5 4 3 2 1

【题5】本题利用循环双链表的特点先找到p节点的右边节点q，然后将p与q进行交换。

实现本题功能的程序如下：

#include "cdlink.cpp"

dnodetype \*swap(dnodetype \*h，elemtype x)

{

dnodetype \*p=h->right，\*q;

if (h==NULL || h->right==h) return h;

//为空或一个节点时直接返回

if (h->data==x) //第1个节点的情况

第4章双 链 表

程序设计题典

124 数据结构程序设计题典

{

q=h->right;

h->left->right=q;q->left=h->left; //删除h节点

h->right=q->right;q->right->left=h;//将h节点插入到q之后

q->right=h;h->left=q;

h=q;

}

else

{

while (p!=h && p->data!=x) //查找data域值为x的节点p

p=p->right;

if (p!=h) //找到了data值为x的节点

{

q=p->right;

if (q==h) //p为最后一个节点

{

p->left->right=h;h->left=p->left; //删除p节点

h->left->right=p;p->left=h->left;

//将p作为第1个节点

p->right=h;h->left=p;

h=p;

}

else

{

p->left->right=q;q->left=p->left; //删除p节点

p->right=q->right;q->right->left=p;

//将p插入到q之后

q->right=p;p->left=q;

}

}

else cout << "未找到data值为x的节点" << endl;

}

return h;

}

void main()

{

第4章 双 链 表

程序设计题典

数据结构程序设计题典 125

dnodetype \*h;

elemtype x;

h=cyccreate();

cycdisp(h);

cout << "输入x值：";

cin >> x;

h=swap(h，x);

cout << "交换后";

cycdisp(h);

}

本程序的一次执行结果如下：

建立一个循环双链表

输入第1节点data域值：1l

输入第2节点data域值：2l

输入第3节点data域值：3l

输入第4节点data域值：4l

输入第5节点data域值：5l

输入第6节点data域值：0l

输出一个循环双链表：

1 2 3 4 5

输入x值：3l

交换后输出一个循环双链表：

1 2 4 3 5

**第5 章栈**

数据结构程序设计题典127

第5章 栈

程序设计题典

*5.1 本章摘要*

*5.1.1 知识点*

*1. 栈*\_\_\_\_\_\_\_\_\_\_\_\_的概念及特点

*2. 栈的存储结构*

· 顺序栈

· 链式栈

*3. 栈的基本运算*

· 初始化栈

· 进栈

· 退栈

· 判断栈是否为空栈

· 获取栈顶元素

· 输出栈的所有元素

*5.1.2 内容概要*

*1. 栈的概念*

栈是限定只在一端进行插入、删除的特殊线性表。

从这个定义中应当理解到以下几点：

· 栈属于加了限制条件的线性结构。

· 栈是后进先出的线性表。

· 进栈和出栈只能从栈的一个端点进行，该端点称为栈顶。不能从另一个端点进入或

删除栈元素，该端点称为栈底；亦不能从栈的中间进入或删除栈元素。

· 栈中的元素个数可以为0，此时是空栈。

· 栈中的元素个数是可以变化的，可以是多个，但不能是无穷多个。

· 同一个栈中的元素的类型相同，亦称元素长度相同。

128 数据结构程序设计题典

第5章栈

程序设计题典

*2. 栈的存储结构*

栈也是一种线性结构，其存储方式有顺序存储方式（顺序栈）和链式存储方式（链式

栈）。在顺序栈中，使用一个数组存放栈的元素，并用一个栈指针指向栈顶元素；在链式

栈中，用一个单链表存储栈的元素，链表的第一个节点定义为栈顶节点，存放栈顶元素。

本章只介绍顺序栈上的算法实现。

顺序栈的节点的类型定义如下：

#define MaxLen 20 //顺序栈存放的最多元素个数为MaxLen-1

typedef char elemtype;

typedef struct sqstack {

elemtype data[MaxLen];

int top; //栈顶指针

} stack;

*3. 栈的运算*

栈的基本运算如下：

? init(stack \*st)：初始化栈st；

? push(stack \*st，elemtype x)：在栈st中进栈一个元素x；

? pop(stack \*st，elemtype \*x)：在栈st中退栈一个元素，并赋给x；

? empty(stack \*st)：判断栈st是否为空栈。若为空栈，返回1；否则返回0；

? gettop(stack \*st，elemtype \*x)：将栈st的栈顶元素赋给x，但栈指针不变；

? disp(stack \*st)：输出栈st的所有元素。

*5.2 例题解析*

【例5.1】以顺序栈的存储方式实现栈的基本运算。

【解】 顺序栈的存储结构如图5.1所示，其中top为栈顶指针。

数据结构程序设计题典129

第5章 栈

程序设计题典

图 5.1 顺序栈的存储结构

实现本题功能的算法如下（本文件名为stack.cpp）：

#include <iostream.h>

#define MaxLen 20 //顺序栈存放的最多元素个数为MaxLen-1

typedef char elemtype;

typedef struct sqstack

{

elemtype data[MaxLen];

int top;

} stack;

void init(stack \*st) //初始化栈st

{

st->top=0;

}

int push(stack \*st，elemtype x) //入栈

{

if (st->top==MaxLen-1)

{

cout << "栈上溢出" << endl;

return 0;

}

else

{

st->top++;

st->data[st->top]=x;

return 1;

}

}

130 数据结构程序设计题典

第5章栈

程序设计题典

int pop(stack \*st，elemtype \*x) //退栈

{

if (st->top==0)

{

cout << "栈下溢出" << endl;

return 0;

}

else

{

\*x=st->data[st->top];

st->top--;

return 1;

}

}

int empty(stack \*st) //判断栈是否为空栈

{

if (st->top==0)

return 1;

else

return 0;

}

int gettop(stack \*st，elemtype \*x) //获取栈顶元素

{

if (st->top==0)

{

cout << "栈下溢出" << endl;

return 0;

}

else

{

\*x=st->data[st->top];

return 1;

}

}

void disp(stack \*st) //输出栈的所有元素

{

int i;

数据结构程序设计题典131

第5章 栈

程序设计题典

for (i=st->top;i>0;i--)

cout << st->data[i] << " ";

cout << endl;

}

【例5.2】设单链表中存放n个字符，试设计一个算法，使用栈判断该字符串是否中

心对称，如xyzzyx即为中心对称字符串。

【解】 先使用create()函数从用户输入的字符串创建相应的单链表，然后调用judge()

函数判断是否为中心对称字符串。在judge()中先将字符串进栈，然后将栈中的字符逐个与

单链表中的字符进行比较。实现本题功能的程序如下：

#include <iostream.h>

#include <malloc.h>

#define MaxLen 100

typedef struct node

{

char data;

struct node \*next;

} cnode;

cnode \*create(char s[])

{

int i=0;

cnode \*h，\*p，\*r;

while (s[i]!='\0')

{

p=(cnode \*)malloc(sizeof(cnode));

p->data=s[i];p->next=NULL;

if (i==0)

{

h=p;r=h; //r始终指向最后一个节点

}

else

{

r->next=p;r=p;

}

i++;

}

return h;

}

132 数据结构程序设计题典

第5章栈

程序设计题典

int judge(cnode \*h)

{

char st[MaxLen];

int top=0;

cnode \*p=h;

while (p!=NULL)

{

st[top]=p->data;

top++;

p=p->next;

}

p=h;

while (p!=NULL)

{

top--;

if (p->data==st[top])

p=p->next;

else

break;

}

if (p==NULL)

return 1;

else

return 0;

}

void main()

{

char str[MaxLen];

cnode \*h;

cout << "输入一个字符串：";

cin >> str;

h=create(str);

if (judge(h)==1)

cout << "'" << str << "'是中心对称字符串" << endl;

else

cout << "'" << str << "'不是中心对称字符串" << endl;

}

本程序的一次执行结果如下：

数据结构程序设计题典133

第5章 栈

程序设计题典

输入一个字符串：xyzzyxl

'xyzzyx'是中心对称字符串

【例5.3】己知有一个多项式Fn(X)，可递归定义如下：

Fn(X)=

试编写出计算Fn(X)值的递归算法和非递归算法。

【解】 f1(n，x)函数为递归算法，f2(n，x)函数为非递归算法。在f2(n，x)函数中使用

了一个栈用于保存n和对应的fn(x)值，然后边退栈边计算fn(x)，栈空后该值就计算出来了。

实现本题功能的程序如下：

#include <iostream.h>

#define MaxLen 20

double f1(int n，double x)

{

if (n==0)

return 1;

else if (n==1)

return 2\*x;

else

return 2\*x\*f1(n-1，x)-2\*(n-1)\*f1(n-2，x);

}

double f2(int n，double x)

{

struct stack

{

int no; //保存n

double val; //保存fn(x)值

} st[MaxLen];

int top=0，i;

double fv1=1; //n=0时的初值

double fv2=2\*x; //n=1时的初值

for (i=n;i>=2;i--)

1 当n=0时

2X 当n=1时

2XFn-1(X)-2(n-1)Fn-2(X) 当n>1时

134 数据结构程序设计题典

第5章栈

程序设计题典

{

top++; //入栈

st[top].no=i;

}

while (top>0)

{

st[top].val=2\*x\*fv2-2\*(st[top].no-1)\*fv1;

fv1=fv2;fv2=st[top].val;

top--; //退栈

}

return fv2;

}

void main()

{

double x;

int n;

cout << "输入n x：";

cin >> n >> x;

cout << "递归求解： F" << n << "(" << x << ")="

<< f1(n，x) << endl;

cout << "非递归求解： F" << n << "(" << x << ")="

<< f2(n，x) << endl;

}

本程序的一次执行结果如下：

输入n x：5 2.4l

递归求解： F5(2.4)=624.2

非递归求解：F5(2.4)=624.2

【例5.4】有如下递归计算公式：

C(n，0)=1 n≥0

C(n，n)=1 n≥0

C(n，m)=C(n-1，m)+C(n-1，m-1) n>m，n≥0，m≥0

数据结构程序设计题典135

第5章 栈

程序设计题典

（1）编写一个计算C(n，m)的递归过程；

（2）将上述过程转换成非递归过程，并给出计算C(5，3)时栈的变化过程。

【解】 （1）本题功能的递归程序如下：

#include <iostream.h>

int f1(int n，int m)

{

if ((n>=0 && m==0) || (n>=0 && m==n))

return(1);

else

{

if (n>m && n>=0 && m>=0)

return(f1(n-1，m)+f1(n-1，m-1));

else

{

cout << "n，m值不正确" << endl;

return(-1);

}

}

}

void main()

{

int n，m;

cout << "输入n m：";

cin >> n >> m;

cout << "C(" << n << "，" << m << ")=" << f1(n，m) << endl;

}

（2）将上述递归函数转换成非递归函数时，使用了一个栈，用于存储调用参数和函数

值，这里采用一个二维数组作为栈，其内容如下：

st(top，1)存储C(n，m)之值

st(top，2)存储n之值

st(top，3)存储m之值

由此，实现本题功能的程序如下：

136 数据结构程序设计题典

第5章栈

程序设计题典

#include <iostream.h>

#define MaxLen 50

int f2(int n，int m)

{

int st[MaxLen][3]，top=1，s1，s2;

if (n<m || n<0 || m<0)

{

cout << "n，m值不正确" << endl;

return(-1);

}

st[top][1]=0; //初值0进栈

st[top][2]=n; //初值n进栈

st[top][3]=m; //初值m进栈

do //循环

{

cout << "top=" << top << " " << st[top][1] << " "

<< st[top][2] << " " << st[top][3] << endl;

//如果栈顶为(0，n，m)，表示要对其分解，将(0，n-1，m)入栈

if (st[top][1]==0)

{

top++;

st[top][1]=0;

st[top][2]=st[top-1][2]-1;

st[top][3]=st[top-1][3];

//若栈顶位置2的值为0或位置2的值等于位置3的值，则给位置1赋初值1

if (st[top][3]==0 || st[top][2]==st[top][3])

st[top][1]=1;

}

//如果栈顶位置1的值大于0，表示该值已计算好，若栈top-1处位置1

//的值为(0，n，m)，表示需要求第二部分，因此，(0，n-1，m-1)入栈

if (top>=2 && st[top][1]>0 && st[top-1][1]==0)

{

top++;

st[top][1]=0;

st[top][2]=st[top-2][2]-1;

st[top][3]=st[top-2][3]-1;

//若栈顶位置2的值为0或位置2的值等于位置3的值，则给位置1赋初值1

if (st[top][3]==0 || st[top][2]==st[top][3])

st[top][1]=1;

}

//若栈顶和次栈顶位置1的值都大于0，则退栈两次，计算新栈顶位置1之值

数据结构程序设计题典137

第5章 栈

程序设计题典

if (top>2 && st[top][1]>0 && st[top-1][1]>0)

{

s1=st[top][1];

s2=st[top-1][1];

top=top-2;

st[top][1]=s1+s2;

}

//当栈中再次只有一个元素时，则计算完成，退出循环

} while (top>1);

return(st[1][1]);

}

void main()

{

int n，m;

cout << "输入n m：";

cin >> n >> m;

cout << "C(" << n << "，" << m << ")=" << f2(n，m) << endl;

}

本程序的一次执行结果如下所示，其结果说明了计算C(5，3)时栈的变化过程。

输入n m：5 3l

top =1 0 5 3

top =2 0 4 3

top =4 0 3 2

top =6 0 2 1

top =6 2 2 1

top =4 3 3 2

top =2 4 4 3

top =3 0 4 2

top =4 0 3 2

top =6 0 2 1

top =6 2 2 1

top =4 3 3 2

top =5 0 3 1

top =6 0 2 1

top =6 2 2 1

top =5 3 3 1

top =3 6 4 2

C(5，3)=10

138 数据结构程序设计题典

第5章栈

程序设计题典

【例5.5】设计一个算法，将一般算术表达式转化为逆波兰表达式，并求逆波兰表

达式的值。

【解】 算术表达式由用户输入并存放在字符型数组str中，逆波兰表达式用字符型数

组exp存储。在转化过程中使用一个字符型数组st作为栈。将算术表达式转换成逆波兰表达

式的过程如下：

依次遍历str的每个字符ch，对于每一个ch：

（1）若ch为数字，则将ch及以后的数字依次存入数组exp中，数字以一个特殊字符'#'

标志结束；

（2）若ch为左括弧'('，则将此括弧压入栈st；

（3）若ch为右括弧') '，则将栈st中左括弧'('以前的字符依次弹出存入数组exp中，然后

将左括弧'('弹出；

（4）若ch为'+'或'-'，则将当前栈st中'('以前的所有字符(运算符)依次弹出存入数组exp

中，然后将ch压入栈st中；

（5）若ch为'\*'或'/'，则将当前栈st中的栈顶端连续的'\*'或'/'弹出并依次存入数组exp中，

然后将ch压入栈st中；

（6）若ch为'\0'，则将栈st中的所有运算符依次弹出并存入数组exp中，最后可得到表

达式的波兰表示在数组exp中。

对逆波兰表达式求值需要使用一个数栈st，其实现函数如下：逆波兰表达式存放在字

符型数组exp中。从exp的开头依次遍历，当遇到数字字符时，就把它转换成对应的数字并

压入数栈st中；当遇到运算符时，就执行两次弹出数栈st中的数字的操作，对弹出的数字进

行该运算符所指定的运算，再把结果压入数栈st，重复上述过程，直至该表达式终止。这

时数栈顶部之值即为表达式的值。

根据上述原理得到如下程序：

#include <iostream.h>

#define MaxLen 100

int trans(char str[]，char exp[])

{

int st[MaxLen]; //作为栈使用

char ch;

int i=0，t=0，top=-1;

//t作为exp的下标，top作为st的下标，i作为str的下标

while ((ch=str[i++])!='\0')

{

if (ch>='0' && ch<='9') //判断为数字

数据结构程序设计题典139

第5章 栈

程序设计题典

{

exp[t]=ch;t++;

while ((ch=str[i++])!='\0' && ch>='0' && ch<='9')

{

exp[t]=ch;t++;

}

i--;

exp[t]='#';t++;

}

else if (ch=='(') //判断为左括号

{

top++;st[top]=ch;

}

else if (ch==')') //判断为右括号

{

while (st[top]!='(')

{

exp[t]=st[top];top--;t++;

}

top--;

}

else if (ch=='+' || ch=='-') //判断为加减号

{

while (top>=0 && st[top]!='(')

{

exp[t]=st[top];top--;t++;

}

top++;st[top]=ch;

}

else if (ch=='\*' || ch=='/') //判断为'\*'或'/'号

{

while (st[top]=='\*' || st[top]=='/')

{

exp[t]=st[top];top--;t++;

}

top++;st[top]=ch;

}

140 数据结构程序设计题典

第5章栈

程序设计题典

}

while (top>=0)

{

exp[t]=st[top];t++;top--;

}

exp[t]='\0';

return 1;

}

int compvalue(char exp[]，int \*n)

{

int st[MaxLen]，d; //作为栈使用

char ch;

int t=0，top=-1; //t作为exp的下标，top作为st的下标

while ((ch=exp[t++])!='\0')

{

if (ch>='0' && ch<='9') //为数字字符时转换为数字

{

d=0;

do

{

d=10\*d+ch-'0';

} while ((ch=exp[t++])!='#');

top++;st[top]=d; //数字进栈

}

else //为运算符时，计算并退栈

{

switch (ch)

{

case '+'：st[top-1]=st[top-1]+st[top];

break;

case '-'：st[top-1]=st[top-1]-st[top];

break;

case '\*'：st[top-1]=st[top-1]\*st[top];

break;

case '/'：if (st[top]!=0)

st[top-1]=st[top-1]/st[top];

else

数据结构程序设计题典141

第5章 栈

程序设计题典

return 0; //除0错误

break;

}

top--;

}

}

(\*n)=st[top];

return 1;

}

void main()

{

char str[MaxLen]; //存储原算术表达式

char exp[MaxLen]; //存储转换成的波兰表达式

int n;

cout << "算术表达式：";

cin >> str;

if (trans(str，exp)==0)

cout << "原算术表达式不正确" << endl;

else

{

cout << "波兰表达式：" << exp << endl;

if (compvalue(exp，&n)==1)

cout << "计算结果：" << n << endl;

else

cout << "计算错误" << endl;

}

}

本程序的一次执行结果如下所示（由于本算法未考虑加减乘除法的优先级，必要时要

使用括弧，如为了计算1+2\*3，应输入为1+(2\*3) ，否则计算不正确）。输入算术表达式

(5\*(12-3)+4)/2，产生的逆波兰表达式为5#12#3#-\*4#+2#/（其中每个数字后带一“#”标志

数字结束，可忽略它），最后的计算结果为（其中的除法是整除）24。

算术表达式：(5\*(12-3)+4)/2

波兰表达式：5#12#3#-\*4#+2#/

计算结果：24

142 数据结构程序设计题典

第5章栈

程序设计题典

*5.3 习题实践*

【题1】假设一个算术表达式中包含圆括号、方括号和花括号三种类型的括号，编写

一个算法判断其中的括号是否匹配。

【题2】已知递归函数(其中DIV为整除)：

F(n)=

（1）写出求F(n)递归算法；

（2）写出求F(n)的非递归算法。

【题3】已知Ackermann函数的定义如下：

F(n)=

（1）写出Ack(2，1)的计算过程。

（2）写出计算Ack(m，n)的非递归算法。

【题4】设有3个分别命名为X，Y和Z的塔座，在塔座X上插有n个直径各不相同，从

小到大依次编号为1，2，⋯，n的圆盘，现要求将X塔座上的n个圆盘移到塔座Z上并仍按同

样顺序叠放，圆盘移动时必须遵守以下规则：每次只能移动一个圆盘;圆盘可以插在X，Y

和Z中任一塔座；任何时候都不能将一个较大的圆盘放在较小的圆盘上。要求用递归和非递

归两种方法求解，并在n=4时比较两者的结果。

【题5】设有两个栈S1、S2都采用顺序栈方式，并且共享一个存储区[0..MaxLen-1]，

为了尽量利用空间，减少溢出的可能，可采用栈顶相向、迎面增长的存储方式，如图5.2所

示。设计一个有关栈的入栈和出栈算法。

图 5.2 两个栈共享存储空间

1 当n=0时

nF(n DIV 2) 当n>0时

n+1 m=0

Ack(m-1，1) m≠0，n=0

Ack(m-1，Ack(m，n-1) m≠0，n≠0

栈1底栈1顶栈2顶栈2底

栈1 栈2

0 MaxLen-1

数据结构程序设计题典143

第5章 栈

程序设计题典

*5.4 参考答案*

【题1】本题使用一个运算符栈st，当遇到的'('、'['或'{'时进栈，当遇到的'}'、']'或')'

时判断栈顶是否为相应的括号，若是退栈继续执行；否则算法结束。实现本题功能的程序

如下：

#include "stack.cpp"

#include <malloc.h>

int correct(char \*str)

{

stack st;

char x;

int i，ok=1;

init(&st);

for (i=0;str[i]!='\0';i++)

{

switch(str[i])

{

case '('：push(&st，'(');break;

case '['：push(&st，'['];break;

case '{'：push(&st，'{'};break;

case ')'：if (!(pop(&st，&x) && x=='('))

ok=0;

break;

case ']'：if (!(pop(&st，&x) && x=='['])

ok=0;

break;

case '}'：if (!(pop(&st，&x) && x=='{'})

ok=0;

break;

}

if (!ok) break;

}

if (empty(&st) && ok)

return 1;

else

144 数据结构程序设计题典

第5章栈

程序设计题典

return 0;

}

void main()

{

char \*str;

str=(char \*)malloc(100\*sizeof(char));

cout << "str：";

cin >> str;

if (correct(str))

cout << "表达式括号匹配" << endl;

else

cout << "表达式括号不匹配" << endl;

}

本程序的一次执行结果如下：

str：{1+[2+(3+4)+5]+6}l

表达式括号匹配

【题2】f1(n)函数为递归算法，f2(n)函数为非递归算法。在f2(n)函数中使用了一个栈，

用于存储调用参数和函数值，这里采用一个二维数组作为栈，其内容如下：

st[top][0]存储F(n)的值

st[top][1]存储n的值

st[top][2]存储F(n/2)的值

因此栈每个记录有如下关系：

st[top][0]=st[top][1]\*st[top][2]

另外，st[top][\*]与st[top-1][\*]有如下关系：

st[top][2]=st[top-1][0]

因此，实现本题功能的程序如下：

#include <iostream.h>

#define MaxLen 20

int f1(int n)

{

if (n==0)

return 1;

数据结构程序设计题典145

第5章 栈

程序设计题典

else

return n\*f1(n/2);

}

int f2(int n)

{

int st[MaxLen][2];

int top=1，fval;

st[top][1]=n; //初值进栈

while (n!=0) //n不为0进栈

{

top++;

n=n/2;

st[top][1]=n;

}

st[top][0]=1; //给栈顶的st[top][0]赋初值

while (top>1) //开始退栈求值

{

fval=st[top][0]; //保存栈顶的st[top][0]之值

top--; //退栈

st[top][2]=fval;

st[top][0]=st[top][1]\*st[top][2];

}

return(st[top][0]);

}

void main()

{

int n;

cout << "输入n：";

cin >> n;

cout << "递归求解： F(" << n << ")=" << f1(n) << endl;

cout << "非递归求解： F(" << n << ")=" << f2(n) << endl;

}

本程序的一次执行结果如下：

输入n：24l

递归求解： F(24)=5184

非递归求解： F(24)=5184

146 数据结构程序设计题典

第5章栈

程序设计题典

【题3】本题设计思想与例5.4相似。

（1）Ack(2，1)=Ack(1，Ack(2，0))

=Ack(1，Ack(1，1))=Ack(1，Ack(0，Ack(1，0)))

=Ack(1，Ack(0，Ack(0，1)))=Ack(1，Ack(0，2))

=Ack(1，3)=Ack(0，Ack(1，2))

=Ack(0，Ack(0，Ack(1，1)))=Ack(0，Ack(0，Ack(0，Ack(1，0))))

=Ack(0，Ack(0，Ack(0，Ack(0，1))))=Ack(0，Ack(0，Ack(0，2)))

=Ack(0，Ack(0，3))=Ack(0，4)=5

（2）计算Ack(m，n)的递归算法为f1()函数，非递归算法为f2()函数。在f2()中采用一个

二维数组作为栈，其内容如下：

st(top，0)存储1(m=0)、2(m≠0，n=0)或3(m≠0，n≠0)

st(top，1)存储Ack(m，n)之值，初值为0

st(top，2)存储m的值

st(top，3)存储n的值

由此，实现本题功能的程序如下：

#include <stdio.h>

#include <iostream.h>

#define MaxLen 5000

//此数应足够大，因为m和n值的较小增长会引起函数值的极快增长，用的栈空间也

极大

int f1(int m，int n)

{

if (m==0)

return n+1;

else if (n==0)

return f1(m-1，1);

else

return f1(m-1，f1(m，n-1));

}

int no(int m，int n)

{

if (m==0)

return 1;

else if (n==0)

数据结构程序设计题典147

第5章 栈

程序设计题典

return 2;

else

return 3;

}

int f2(int m，int n)

{

int st[MaxLen][4]，top=1，m1，n1;

st[top][0]=no(m，n);

st[top][1]=0; //初值0进栈

st[top][2]=m; //初值m进栈

st[top][3]=n; //初值n进栈

do //开始循环

{

if (st[top][1]==0)

{

if (st[top][0]==3)

{

top++;

st[top][1]=0;

st[top][2]=st[top-1][2];

st[top][3]=st[top-1][3]-1;

st[top][0]=no(st[top][2]，st[top][3]);

}

else if (st[top][0]==2)

{

top++;

st[top][1]=0;

st[top][2]=st[top-1][2]-1;

st[top][3]=1;

st[top][0]=no(st[top][2]，st[top][3]);

}

if (st[top][0]==1)

st[top][1]=st[top][3]+1;

}

if (top>=1 && st[top][1]!=0 && st[top-1][0]==2)

{

st[top-1][1]=st[top][1];

148 数据结构程序设计题典

第5章栈

程序设计题典

top--;

}

else if (top>=1 && st[top][1]!=0 && st[top-1][0]==3)

{

n1=st[top][1];

m1=st[top-1][2]-1;

top--;

st[top][1]=0;

st[top][2]=m1;

st[top][3]=n1;

st[top][0]=no(st[top][2]，st[top][3]);

}

if (top==1 && st[top][1]!=0)

//栈中只有1个元素，且已计算出值，则退出循环

break;

} while (top>=1);

return(st[1][1]);

}

void main()

{

int n，m;

cout << "输入m n：";

cin >> m >> n;

cout << "递归 C(" << m << "，" << n << ")=" << f1(m，n) <<

endl;

cout << "非递归 C(" << m << "，" << n << ")=" << f2(m，n) <<

endl;

}

本程序的一次执行结果如下：

输入m n：3 8l

递 归 C(3，8)=2045

非递归 C(3，8)=2045

数据结构程序设计题典149

第5章 栈

程序设计题典

【题4】递归求解的原理是，设hanoi(n，x，y，z)表示将n个圆盘从x通过y移动到z上，

则有：

hanoi(n，x，y，z)=>hanoi(n-1，x，z，y);

将第n个圆盘从x移到z;

hanoi(n-1，y，x，z)

由此得到hanoi1()函数。

非递归求解的原理是使用一个栈，其结构如下：

struct stack

{

int no，ns;

char x，y，z;

} st[MaxLen];

其中no存放一个标识，为0时表示直接移动一个圆盘，为1时表示需进一步分解;ns存放

当前圆盘数;x，y和z表示三个塔座。由此得到hanoi2()函数。

实现本题功能的程序如下：

#include <iostream.h>

#define MaxLen 50

void hanoi1(int n，char a，char b，char c)

{

if (n==1)

cout << "\t将第" << n << "个盘片从" << a << "移动到" << c <<

endl;

else

{

hanoi1(n-1，a，c，b);

cout << "\t将第" << n << "个盘片从" << a << "移动到" << c

<< endl;

hanoi1(n-1，b，a，c);

}

}

void hanoi2(int n，char a，char b，char c)

{

struct stack

{

int no，ns;

char x，y，z;

} st[MaxLen];

150 数据结构程序设计题典

第5章栈

程序设计题典

int top=1，n1，a1，b1，c1;

st[top].no=1; //初值入栈

st[top].ns=n;

st[top].x=a;

st[top].y=b;

st[top].z=c;

while (top>0)

{

if (st[top].no==1)

{

n1=st[top].ns; //退栈hanoi(n，x，y，z)

a1=st[top].x;

b1=st[top].y;

c1=st[top].z;

top--;

top++; //将hanoi(n-1，x，z，y)入栈

st[top].no=1;

st[top].ns=n1-1;

st[top].x=b1;

st[top].y=a1;

st[top].z=c1;

top++; //将第n个圆盘从x移到z

st[top].no=0;

st[top].ns=n1;

st[top].x=a1;

st[top].y=c1;

top++; //hanoi(n-1，y，x，z)入栈

st[top].no=1;

st[top].ns=n1-1;

st[top].x=a1;

st[top].y=c1;

st[top].z=b1;

}

while (top>0 && (st[top].no==0 || st[top].ns==1))

{

if (top>0 && st[top].no==0) //将第n个圆盘从x移到z并退栈

{

cout << "\t将第" << st[top].ns << "个盘片从"

<< st[top].x << "移动到" << st[top].y << endl;

数据结构程序设计题典151

第5章 栈

程序设计题典

top--;

}

if (top>0 && st[top].ns==1) //hanoi(1，x，y，z)退栈

{

cout << "\t将第" << st[top].ns << "个盘片从"

<< st[top].x << "移动到" << st[top].z << endl;

top--;

}

}

}

}

void main()

{

int n=3;

cout << "hanoi(3)递归求解结果：" << endl;

hanoi1(n，'X'，'Y'，'Z');

cout << "hanoi(3)非递归求解结果：" << endl;

hanoi2(n，'X'，'Y'，'Z');

}

本程序的一次执行结果如下：

hanoi(3)递归求解结果：

将第1个盘片从X移动到Z

将第2个盘片从X移动到Y

将第1个盘片从Z移动到Y

将第3个盘片从X移动到Z

将第1个盘片从Y移动到X

将第2个盘片从Y移动到Z

将第1个盘片从X移动到Z

hanoi(3)非递归求解结果：

将第1个盘片从X移动到Z

将第2个盘片从X移动到Y

将第1个盘片从Z移动到Y

将第3个盘片从X移动到Z

将第1个盘片从Y移动到X

将第2个盘片从Y移动到Z

将第1个盘片从X移动到Z

152 数据结构程序设计题典

第5章栈

程序设计题典

【题5】设两个栈的栈顶分别为top1，top2，有关栈的入栈和出栈算法如下：

#include <iostream.h>

#define MaxLen 5

char st[MaxLen];

int top1=0，top2=MaxLen-1; //两个栈的栈顶分别为top1和top2

int push(int i，char x)

{

if (top1+1==top2) //栈上溢出

return 0;

else

{

switch (i)

{

case 1：top1++;st[top1]=x;break;

case 2：top2--;st[top2]=x;break;

}

}

return 1;

}

int pop(int i，char \*x)

{

switch(i)

{

case 1：if (top1==0) //栈1下溢出

return 0;

else

{

\*x=st[top1];top1--;

}

break;

case 2：if (top2==MaxLen-1) //栈2下溢出

return 0;

else

{

\*x=st[top2];top2++;

}

数据结构程序设计题典153

第5章 栈

程序设计题典

break;

}

return 1;

}

void main()

{

char x;

push(1，'a');

push(2，'b');

push(1，'c');

pop(1，&x);

cout << "st1：" << x << endl;

pop(2，&x);

cout << "st2：" << x << endl;

}

**第6 章队列**

第6章 队 列

程序设计题典

数据结构程序设计题典 155

*6.1 本章摘要*

*6.1.1 知识点*

*1. 队列的概念及特点*

*2. 队列的存储结构*

· 顺序队列：非循环队列和循环队列

· 链式队列

*3. 队列的基本运算*

· 初始化队列

· 入队列

· 出队列

· 判断队列是否为空队

· 获取队列头元素

*6.1.2 内容概要*

*1. 队列的概念*

队列是限定只能在一端进队、在另一端出队的特殊线性表。

从这个定义中应当理解到以下几点：

· 队列属于加了限制条件的线性结构。

· 队列是先进先出的线性表。

· 进队只能从称为队尾的一端进行，出队只能从队的另一端，称为队头的端点进行，

不能从队列的中间进入或删除元素。

· 队列中的元素个数可以为0，此时是空队列。

· 队列中的元素的个数是可以变化的，可以是多个，但不能是无穷多个。

· 同一个队列中的所有元素的类型相同。

第6章队 列

程序设计题典

156 数据结构程序设计题典

*2. 队列的存储结构*

队列也是一种线性结构，其存储方式有顺序存储方式（顺序队列）和链式存储方式（链

式队列）。在顺序队列中，使用一个数组存放队列的元素，并用一个队列指针（front）指

向队头元素，一个队列指针（rear）指向队尾元素；在链式队列中，用一个单链表存储队列

的元素，链表的第一个节点定义为队头节点，链表的最后一个节点定义为队尾节点。

顺序队列又分为循环队列和非循环队列，循环队列解决了假溢出的问题。本章只介绍

顺序循环队列上的算法实现。

顺序队列的节点的类型定义如下：

#define MaxLen 20

typedef char elemtype;

typedef struct {

elemtype data[MaxLen];

int front，rear;

} queue;

在循环队列sq中：

· 初始时：sq->front=sq->rear=0

· 队空条件：sq->rear==sq->front

· 队满条件：(sq->rear+1)%MaxLen==sq->front

*3. 队列的运算*

· init(queue \*sq)：初始化队列sq。

· enqueue(queue \*sq，elemtype x)：在队列sq中入队一个元素x。

· outqueue(queue \*sq，elemtype \*x)：在队列sq中出队一个元素，并将其值赋给x。

· empty(queue \*sq)：判断队列sq是否为空队。若为空，返回1；否则返回0。

· gethead(queue \*sq，elemtype \*x)：将队列sq的队头元素赋给x，但不移动队头指针。

*6.2 例题解析*

【例6.1】以环形顺序队列的存储方式实现队列的基本运算。

【解】 环形顺序队列的存储结构如图6.1所示，其中front为队头指针，rear为队尾指

针。

第6章 队 列

程序设计题典

数据结构程序设计题典 157

图 6.1 环形顺序队列的存储结构

实现本题功能的算法如下（本文件名为queue.cpp，在后面的题解中使用其中的函数）：

#include <iostream.h>

#define MaxLen 20

typedef char elemtype;

typedef struct

{

elemtype data[MaxLen];

int front，rear;

} queue;

void init(queue \*sq) //初始化队列

{

sq->front=0;

sq->rear=0;

}

int enqueue(queue \*sq，elemtype x) //入队列

{

if ((sq->rear+1)%MaxLen==sq->front) //队列上溢出

return 0;

else

{

sq->rear=(sq->rear+1)%MaxLen;

sq->data[sq->rear]=x;

return 1;

}

}

int outqueue(queue \*sq，elemtype \*x) //出队列

第6章队 列

程序设计题典

158 数据结构程序设计题典

{

if (sq->front==sq->rear) //队列下溢出

return 0;

else

{

sq->front=(sq->front+1)%MaxLen;

\*x=sq->data[sq->front];

return 1;

}

}

int empty(queue \*sq) //判断队列是否为空队

{

if (sq->rear==sq->front)

return 1;

else

return 0;

}

int gethead(queue \*sq，elemtype \*x) //取队头

{

if (sq->rear==sq->front)

return 0; //队列下溢出

else

{

\*x=sq->data[(sq->front+1)%MaxLen];

return 1;

}

}

【例6.2】对于一个具有MaxLen个单元的环形队列，设计一个算法求其中共有多少

个元素。

【解】 环形队列中的元素个数计算公式如下：

if (rear=front) 元素个数=0

if (rear>front) 元素个数=rear-front

if (rear<front) 元素个数=MaxLen+rear-front

归纳起来，公式为：

第6章 队 列

程序设计题典

数据结构程序设计题典 159

元素个数=(rear-front+MaxLen)%MaxLen

因此，实现本题功能的程序如下：

#include "queue.cpp"

int count(queue \*sq)

{

return (sq->rear-sq->front+MaxLen)%MaxLen;

}

void main()

{

queue sq;

elemtype x;

init(&sq);

cout << "a入队" << endl;

enqueue(&sq，'a');

cout << "出队一次" << endl;

outqueue(&sq，&x);

cout << "b入队" << endl;

enqueue(&sq，'b');

cout << "c入队" << endl;

enqueue(&sq，'c');

cout << "出队一次" << endl;

outqueue(&sq，&x);

cout << "d入队" << endl;

enqueue(&sq，'d');

cout << "出队一次" << endl;

outqueue(&sq，&x);

cout << "e入队" << endl;

enqueue(&sq，'e');

cout << "f入队" << endl;

enqueue(&sq，'f');

cout << "队列中的元素个数：" << count(&sq) << endl;

}

本程序的执行结果如下：

第6章队 列

程序设计题典

160 数据结构程序设计题典

a入队

出队一次

b入队

c入队

出队一次

d入队

出队一次

e入队

f入队

队列中的元素个数：3

【例6.3】假设以数组sequ[0..MaxSize-1]存放环形队列的元素，同时设变量rear和len

分别指示环形队列中队尾元素的位置和内含元素的个数。试写出此环形队列队满的条件，

并设计相应入队和出队的算法。

解：该环形队列队满的条件为：len==MaxSize，队空的条件为：lem==0。由rear，len

求队列头指针front的过程为：

front=rear-len+1;

if (front<0) front=front+MaxSize;

归纳起来：

front=(rear-len+1+MaxSize)%MaxSize

实现本题功能的程序如下：

#include <iostream.h>

#define MaxSize 6

typedef char queue[MaxSize];

int rear=0，len=0; //队列初态

int enqueue(queue qu，char x)

{

if (len==MaxSize) //队满，上溢出

return 0;

else

{

rear=(rear+1)%MaxSize;

第6章 队 列

程序设计题典

数据结构程序设计题典 161

qu[rear]=x;

len++; //队列长度增1

return 1;

}

}

int dequeue(queue qu，char \*x)

{

int front;

if (len==0) //队列为空时下溢出

return 0;

else

{

front=(rear-len+1+MaxSize)%MaxSize;

\*x=qu[front];

len--; //队\_\_\_\_\_\_\_列长度减1

return 1;

}

}

void main()

{

char x;

queue qu;

cout << "a入队" << endl;

enqueue(qu，'a');

cout << "b入队" << endl;

enqueue(qu，'b');

cout << "c入队" << endl;

enqueue(qu，'c');

cout << "出队一次：";

dequeue(qu，&x);

cout << x << endl;

cout << "d入队" << endl;

enqueue(qu，'d');

cout << "e入队" << endl;

enqueue(qu，'e');

cout << "出队一次：";

dequeue(qu，&x);

第6章队 列

程序设计题典

162 数据结构程序设计题典

cout << x << endl;

cout << "f入队" << endl;

enqueue(qu，'f');

cout << "g入队" << endl;

enqueue(qu，'g');

cout << "出队一次：";

dequeue(qu，&x);

cout << x << endl;

cout << "余下元素出列：";

while (len>0)

{

dequeue(qu，&x);

cout << x << " ";

}

cout << endl;

}

本程序的执行结果如下：

a入队

b入队

c入队

出队一次：a

d入队

e入队

出队一次：b

f入队

g入队

出队一次：c

余下元素出列：d e f g

【例6.4】到医院看病的过程是，患者先排队等候，排队过程中主要重复两件事：

（1）病人到达诊室时，将病历交给护士，排到等候队列中候诊。

（2）护士从等候队列中取出下一个患者的病历，该患者进入诊室就诊。

第6章 队 列

程序设计题典

数据结构程序设计题典 163

在排队时按照“先到先服务”的原则。设计一个算法模拟病人等候就诊的过程。其中

“病人到达”用命令'A' ('a')表示，“护士让下一位患者就诊”用命令'N' ('n')表示，命令'Q' ('q')

表示不再接收病人排队。

【解】 这里采用一个队列，有“病人到达”命令时即入队，有“护士让下一位患者

就诊”命令时即出队，命令'Q'即队列中所有元素出队。实现本题功能的程序如下：

#include "queue.cpp"

void seedoctor()

{

queue sq; //将queue.cpp文件中elemtype改为int类型

char ch;

int n;

init(&sq);

while (1)

{

cout << "输入命令：";

cin >> ch;

switch (ch)

{

case 'a'：

case 'A'：cout << " 病历号：";

cin >> n;

enqueue(&sq，n);

break;

case 'n'：

case 'N'：if (empty(&sq)==0)

{

outqueue(&sq，&n);

cout << " 病历号为" << n << "的病人就诊" << endl;

}

else cout << " 无病人等候就诊" << endl;

break;

case 'q'：

case 'Q'：cout << " 排队的病人依次就诊：";

while (empty(&sq)==0)

{

outqueue(&sq，&n);

cout << n << " ";

第6章队 列

程序设计题典

164 数据结构程序设计题典

}

cout << endl;

}

if (ch=='q' || ch=='Q')

break;

}

}

void main()

{

seedoctor();

}

本程序的一次执行结果如下：

输入命令：al

病历号：12l

输入命令：al

病历号：45l

输入命令：nl

病历号为12的病人就诊

输入命令：al

病历号：8l

输入命令：al

病历号：56l

输入命令：nl

病历号为45的病人就诊

输入命令：ql

排队的病人依次就诊：8 56

【例6.5】双端队列（Deque）是一个可以在任何一端进行插入和删除的线性表，现

采用一个一维数组作为双端队列的数据存储结构，使用C语言描述如下：

第6章 队 列

程序设计题典

数据结构程序设计题典 165

#define MaxSize 6

typedef char elemtype;

typedef struct

{

elemtype data[MaxSize]; //环形队列的存放数组

int end1，end2; //环形队列的两端

} deque;

试编写两个算法add(deque Qu，char x，int tag)和delete(deque Qu，char \*x，int tag)，用

以在此双端队列的任一端进行插入和删除。当tag＝1时在左端end1端操作，当tag＝2时在右

端end2端操作。

【解】 双端队列的初始状态（空队列）为Qu.end1+1=Qu.end2，其队列满的条件为

Qu.end1=Qu.end2。本程序的双端队列初态如图6.2所示（应为环形队列）。实现本题功能的

程序如下：

图 6.2 双端队列初态

#include <iostream.h>

#define MaxSize 6

typedef char elemtype;

typedef struct

{

elemtype data[MaxSize]; //环形队列的存放数组

int end1，end2; //环形队列的两端

} deque;

int dqadd(deque \*Qu，elemtype x，int tag)

{

switch(tag)

{

case 1：if (Qu->end1!=Qu->end2)//队列未满时

{

第6章队 列

程序设计题典

166 数据结构程序设计题典

Qu->data[Qu->end1]=x;

Qu->end1=(Qu->end1-1)%MaxSize;

return 1;

}

else return 0;

case 2：if (Qu->end2!=Qu->end1) //队列未满时

{

Qu->data[Qu->end2]=x;

Qu->end2=(Qu->end2+1)%MaxSize;

return 1;

}

else return 0;

}

return 0;

}

int dqdelete(deque \*Qu，elemtype \*x，int tag)

{

switch(tag)

{

case 1：if (((Qu->end1+1)%MaxSize)!=Qu->end2)//队列不为空时

{

Qu->end1=(Qu->end1+1)%MaxSize;

\*x=Qu->data[Qu->end1];

return 1;

}

else return 0;

case 2：if (((Qu->end2-1)%MaxSize)!=Qu->end1)//队列不为空时

{

Qu->end2=(Qu->end2-1)%MaxSize;

\*x=Qu->data[Qu->end2];

return 1;

}

else return 0;

}

return 0;

}

第6章 队 列

程序设计题典

数据结构程序设计题典 167

void main()

{

deque Qu;

Qu.end1=2;//设置双端队列的初始状态

Qu.end2=3;

elemtype x;

cout << "a右端进队列" << endl;

dqadd(&Qu，'a'，1);

cout << "b右端进队列" << endl;

dqadd(&Qu，'b'，1);

cout << "c右端进队列" << endl;

dqadd(&Qu，'c'，1);

cout << "d左端进队列" << endl;

dqadd(&Qu，'d'，2);

cout << "e左端进队列" << endl;

dqadd(&Qu，'e'，2);

cout << "右端出队一次：";

dqdelete(&Qu，&x，1);

cout << x << endl;

cout << "右端出队一次：";

dqdelete(&Qu，&x，1);

cout << x << endl;

cout << "左端出队一次：";

dqdelete(&Qu，&x，2);

cout << x << endl;

cout << "左端出队一次：";

dqdelete(&Qu，&x，2);

cout << x << endl;

cout << "右端出队一次：";

dqdelete(&Qu，&x，1);

cout << x << endl;

}

本程序的执行结果如下：

第6章队 列

程序设计题典

168 数据结构程序设计题典

a右端进队列

b右端进队列

c右端进队列

d左端进队列

e左端进队列

右端出队一次：c

右端出队一次：b

左端出队一次：e

左端出队一次：d

右端出队一次：a

*6.3 习题实践*

【题1】对于一个具有MaxLen个单元的环形队列，设计一个算法输出其中的所有元

素。

【题2】假设以数组Q[0..MaxSize-1]存放环形队列中的元素，同时设置一个标志tag，

以tag＝0和tag＝1来区别在队头指针（front）和队尾指针（rear）相等时，队列状态为“空”

还是“满”。试编写与此结构相应的插入（enqueue）和删除（dequeue）元素的算法。

【题3】编写一个程序求解迷宫问题。迷宫是一个m行n列的0-1矩阵，其中0表示无

障碍，1表示有障碍。设入口为(1，1)，出口为(m，n)，每次移动只能从一个无障碍的单元

移到其周围8个方向上任一无障碍的单元，编制程序给出一条通过迷宫的路径或报告一个

“无法通过”的信息。

【题4】某汽车轮渡口，过江渡船每次能载10辆车过江。过江车辆分为客车类和货车

类，上船有如下规定：同类车先到先上船，客车先于货车上渡船，且每上4辆客车，才允许

上一辆货车；若等待客车不足4辆，则以货车代替，若无货车等待则允许客车都上船。设计

一个算法模拟渡口管理。

【题5】一个双端队列Deque是限定在两端end1、end2都可进行插入和删除的线性表。

队空的条件是end1＝end2。若用顺序方式来组织双端队列，试根据下列要求，定义双端队

列的结构，并给出指定端（i＝1，2）进行插入enq和删除操作的实现。

要求：

（1）当队满时，最多只能有一个元素空间可以是空的。

（2）在做两端的插入和删除时，队列中的其他元素一律不动。

第6章 队 列

程序设计题典

数据结构程序设计题典 169

*6.4 参考答案*

【题1】设计思想参见例6.2。实现本题功能的程序如下：

#include "queue.cpp"

void disp(queue \*sq)

{

int i;

cout << "队列中的元素：";

if (sq->rear==sq->front)

cout << "空队列";

if (sq->rear>sq->front)

for (i=sq->front+1;i<=sq->rear;i++)

cout << sq->data[i] << " ";

else

for (i=sq->front+1;i<=sq->rear+MaxLen;i++)

cout << sq->data[i%MaxLen] << " ";

cout << endl;

}

void main()

{

queue sq;

elemtype x;

init(&sq);

cout << "a入队" << endl;

enqueue(&sq，'a');

cout << "b入队" << endl;

enqueue(&sq，'b');

cout << "出队一次" << endl;

outqueue(&sq，&x);

cout << "c入队" << endl;

enqueue(&sq，'c');

disp(&sq);

cout << "d入队" << endl;

enqueue(&sq，'d');

cout << "e入队" << endl;

enqueue(&sq，'e');

cout << "出队一次" << endl;

第6章队 列

程序设计题典

170 数据结构程序设计题典

outqueue(&sq，&x);

cout << "出队一次" << endl;

outqueue(&sq，&x);

cout << "f入队" << endl;

enqueue(&sq，'f');

cout << "g入队" << endl;

enqueue(&sq，'g');

cout << "出队一次" << endl;

outqueue(&sq，&x);

cout << "h入队" << endl;

enqueue(&sq，'h');

disp(&sq);

}

本程序的执行结果如下：

a入队

b入队

出队一次

c入队

队列中的元素：b c

d入队

e入队

出队一次

出队一次

f入队

g入队

出队一次

h入队

队列中的元素：e f g h

【题2】该环形队列队满条件为：rear==front and tag==1。队空的条件为：tag==0。

实现本题功能的程序如下：

#include <iostream.h>

#define MaxSize 6

typedef char queue[MaxSize];

int front=0，rear=0; //队列初态

int enqueue(queue qu，int \*tag，char x)

第6章 队 列

程序设计题典

数据结构程序设计题典 171

{

if (\*tag==1 && front==rear) //队满，上溢出

return 0;

else

{

rear=(rear+1)%MaxSize;

qu[rear]=x;

if (\*tag==0) //原队为空时，插入元素后改为不空

\*tag=1;

return 1;

}

}

int dequeue(queue qu，int \*tag，char \*x)

{

if (\*tag==0)//队列为空时下溢出

return 0;

else

{

front=(front+1)%MaxSize;

\*x=qu[front];

if (front==rear)//队列空时，设置tag值

\*tag=0;

return 1;

}

}

void main()

{

int tag=0;

char x;

queue qu;

cout << "a入队" << endl;

enqueue(qu，&tag，'a');

cout << "b入队" << endl;

enqueue(qu，&tag，'b');

cout << "c入队" << endl;

enqueue(qu，&tag，'c');

cout << "出队一次：";

第6章队 列

程序设计题典

172 数据结构程序设计题典

dequeue(qu，&tag，&x);

cout << x << endl;

cout << "d入队" << endl;

enqueue(qu，&tag，'d');

cout << "e入队" << endl;

enqueue(qu，&tag，'e');

cout << "出队一次：";

dequeue(qu，&tag，&x);

cout << x << endl;

cout << "出队一次：";

dequeue(qu，&tag，&x);

cout << x << endl;

}

本程序的执行结果如下：

a入队

b入队

c入队

出队一次：a

d入队

e入队

出队一次：b

出队一次：c

【题3】要寻找一条通过迷宫的路径，就必须进行试探性搜索，只要有路可走就前进

一步，无路可走时，退回一步，重新选择未走过的可走的路径，如此继续，直至到达出口

或返回入口(无法通过迷宫)。我们可使用如下的数据结构：mg[1..m，1..n]表示迷宫，为了

算法方便，在四周加上一圈“哨兵”即变为数组mg[0..m+1，0..+1]表示迷宫，用数组zx，

zy分别表示X，Y方向的移动增量，其值如表6.1所示。

表6.1 zx，zy数组的方向取值

方向 北 东北 东 东南 南 西南 西 西北

下标 1 2 3 4 5 6 7 8

zx -1 -1 0 1 1 1 0 -1

zy 0 1 1 1 0 -1 -1 -1

第6章 队 列

程序设计题典

数据结构程序设计题典 173

在探索前进路径时，需要将搜索的踪迹记下来，记录的踪迹应包含当前位置以及前驱

位置。在搜索函数中，将所有需要搜索的位置形成一个队列，将队列中的每一个元素可能

到达的位置加入到队列之中，当队列中某元素所有可能到达的位置全部加入到队列之后，

即从队列中将该元素去掉。用变量front及rear分别表示队列的首与尾，当rear指示的元素已

到达出口(m，n)时，根据rear所指示的前驱序号可回溯得到走迷宫的最短路径。

根据上述原理得到的程序如下：

#include <iostream.h>

#include <iomanip.h>

#define m 5 //行数

#define n 6 //列数

#define MaxSize 100

struct stype

{

int x，y，pre;

} queue[MaxSize];

int mg[m+1][n+1]={//一个迷宫，其四周要加上均为1的外框

{1，1，1，1，1，1，1}，

{1，0，0，1，0，1，1}，

{1，0，1，0，1，0，1}，

{1，0，1，1，1，0，1}，

{1，1，1，0，1，0，1}，

{1，1，1，1，1，1，1}

};

int zx[8]，zy[8];

void printlj(int rear)

{

int i，j;

i=rear;

do

{

j=i;

i=queue[i].pre;

queue[j].pre=-1;

} while (i!=0);

cout << "查找到的迷宫路径：\n\t入口->";

i=0;

第6章队 列

程序设计题典

174 数据结构程序设计题典

while (i<MaxSize)

{

if (queue[i].pre==-1)

cout << "(" << queue[i].x << "，" << queue[i].y << ")->";

i++;

}

cout << "出口\n";

}

void mglj()

{

int i，j，x，y，v，front，rear，find=0，no=1;

queue[1].x=1;queue[1].y=1;queue[1].pre=0;

//从(1，1)开始搜索，将该点插入队列

front=1;rear=1; //队列指针置初值1

mg[1][1]=-1; //将其赋值-1，以避免回过来重复搜索

cout << " 步号 x y pre\n";

cout << setw(4) << no++

<< setw(4) << queue[rear].x

<< setw(4) << queue[rear].y

<< setw(4) << queue[rear].pre << endl;

while (front<=rear && !find) //队列不为空且未找到路径时环形

{

x=queue[front].x; y=queue[front].y;

for (v=1;v<=8;v++)

//环形遍历每个方向，把每个可走的方向插入队列中

{

i=x+zx[v];j=y+zy[v]; //选择一个前进方向(i，j)

if (mg[i][j]==0) //如果该方向可走

{

rear++; //将该方向插入到队列中

queue[rear].x=i;

queue[rear].y=j;

queue[rear].pre=front; //指向上一个步号

mg[i][j]=-1; //将其赋值-1，以避免回过来重复搜索

cout << setw(4) << no++

<< setw(4) << queue[rear].x

<< setw(4) << queue[rear].y

第6章 队 列

程序设计题典

数据结构程序设计题典 175

<< setw(4) << queue[rear].pre << endl;

}

if (i==m && j==n) //找到了出口

{

printlj(rear);//打印找到的路径

find=1; //设置为1时便于退出环形

}

}

front++; //从队列中删除一个元素

}

if (!find)

cout << "无法通过!" << endl;

}

void main()

{

zx[1]=-1;zx[2]=-1;zx[3]=0;zx[4]=1; //建立方向数据

zx[5]=1;zx[6]=1;zx[7]=0;zx[8]=-1;

zy[1]=0;zy[2]=1;zy[3]=1;zy[4]=1;

zy[5]=0;zy[6]=-1;zy[7]=-1，zy[8]=-1;

mglj();//产生迷宫路径

}

本程序的执行结果如下所示。上部分给出了找到路径后队列中的数据，从中可以找出

求解路径，其过程是，先从步号为9的记录出发（其位置为（4，5）），其pre为8，找到步

号为8的记录（其位置为（3，5）），其pre为7，找到步号为7的记录，如此这样，直到步

号是1为止。从而给出了下部分的路径。

步号 x y pre

1 1 1 0

2 1 2 1

3 2 1 1

4 2 3 2

5 3 1 3

6 1 4 4

7 2 5 6

8 3 5 7

9 4 5 8

查找到的迷宫路径：

入口->(1，1)->(1，2)->(2，3)->(1，4)->(2，5)->(3，5)->(4，5)->出口

第6章队 列

程序设计题典

176 数据结构程序设计题典

【题4】本算法的描述如下：

初始化，count(上渡船汽车数)=0，countbus(上船客车数)=0，counttruck(上船货车数)=0;

环形检测用户命令：

（1）命令E(e)表示有汽车来等等船渡江，按客车、货车分类进入相应的队列排列。

（2）命令I(i)表示渡船到渡口，在count小于10时环形：

① 若count<4，客车队列又非空，将客车队列的队头汽车出队上渡船。count和

countbus增1；否则，转②。

② 若countbus>=4，或客车队列为空队且货车队列非空，将货车队列的队头汽车

出队上渡船。countbus=0，count和counttruck增1;否则，转③。

③ 若货车队列为空队且客车队非空，将客车队列的队头汽车出队上渡船。count

和countbus增1，counttruck=0;否则，转④。

④ 提示相应错误信息并退出程序运行。

（3）命令Q(q)表示退出程序运行。

实现本题功能的程序如下：

#include "queue.cpp"

void pass()

{

//将queue.cpp文件中elemtype改为int类型

queue bus，truck; //bus表示客车队列，truck表示货车队列

char ch;

int n，tag; //n为车号，tag为标志，tag=0表示客车，tag=1表示货车

int count=0，countbus=0，counttruck=0;

init(&bus);

init(&truck);

while (1)

{

cout << "输入命令：";

cin >> ch;

switch (ch)

{

case 'e'：

case 'E'：cout << " 车号：";

cin >> n;

cout << " 客车/货车(0/1)：";

cin >> tag;

if (tag==0)

enqueue(&bus，n);

else

enqueue(&truck，n);

第6章 队 列

程序设计题典

数据结构程序设计题典 177

break;

case 'i'：

case 'I'：while (count<10)

{

if (count<4 && empty(&bus)==0) //客车出队

{

outqueue(&bus，&n);

cout << " 上船的车号为：" << n << endl;

count++;

countbus++;

}

else if (empty(&truck)==0)//货车出队

{

countbus=0;

outqueue(&truck，&n);

cout << " 上船的车号为：" << n << endl;

count++;

counttruck++;

}

else if (empty(&bus)==0)

{

counttruck=0;

outqueue(&bus，&n);

cout << " 上船的车号为：" << n << endl;

count++;

countbus++;

}

else

{

cout << " 没有10辆车排队轮渡" << endl;

return;

}

}

break;

case 'q'：

case 'Q'：break;

}

if (ch=='q' || ch=='Q')

break;

}

}

void main()

{

pass();

}

第6章队 列

程序设计题典

178 数据结构程序设计题典

本程序的一次执行结果如下所示(图中“↙”表示回车键)：

输入命令：e↙ 车号：1↙ 客车/货车（0/1）：0↙

输入命令：e↙ 车号：2↙ 客车/货车（0/1）：0↙

输入命令：e↙ 车号：3↙ 客车/货车（0/1）：1↙

输入命令：e↙ 车号：4↙ 客车/货车（0/1）：1↙

输入命令：e↙ 车号：5↙ 客车/货车（0/1）：0↙

输入命令：e↙ 车号：6↙ 客车/货车（0/1）：0↙

输入命令：e↙ 车号：7↙ 客车/货车（0/1）：0↙

输入命令：e↙ 车号：8↙ 客车/货车（0/1）：1↙

输入命令：e↙ 车号：9↙ 客车/货车（0/1）：0↙

输入命令：e↙ 车号：10↙ 客车/货车（0/1）：1↙

输入命令：i↙

上船的车号为：1

上船的车号为：2

上船的车号为：5

上船的车号为：6

上船的车号为：7

上船的车号为：3

上船的车号为：4

上船的车号为：8

上船的车号为：10

上船的车号为：9

输入命令：q↙

【题5】设计思想参见例6.5，这里仍采用环形队列。双端队列的初始状态（空队列）

为Qu.end1=Qu.end2 ， 其队列满的条件为(Qu.end1-1)%MaxSize=Qu.end2 或(Qu.end2

+1)%MaxSize=Qu.end1。实现本题功能的程序如下：

#include <iostream.h>

#define MaxSize 6

typedef char elemtype;

typedef struct

{

elemtype data[MaxSize]; //环形队列的存放数组

int end1，end2; //环形队列的两端

} deque;

第6章 队 列

程序设计题典

数据结构程序设计题典 179

int enq(deque \*Qu，elemtype x，int tag)

{

switch(tag)

{

case 1：if ((Qu->end1-1)%MaxSize!=Qu->end2)//队列未满时

{

Qu->end1=(Qu->end1-1)%MaxSize;

Qu->data[Qu->end1]=x;

return 1;

}

else return 0;

case 2：if ((Qu->end2+1)%MaxSize!=Qu->end1) //队列未满时

{

Qu->end2=(Qu->end2+1)%MaxSize;

Qu->data[Qu->end2]=x;

return 1;

}

else return 0;

}

return 0;

}

int deq(deque \*Qu，elemtype \*x，int tag)

{

switch(tag)

{

case 1：if (Qu->end1!=Qu->end2)//队列不为空时

{

\*x=Qu->data[Qu->end1];

Qu->end1=(Qu->end1+1)%MaxSize;

return 1;

}

else return 0;

case 2：if (Qu->end2!=Qu->end1) //队列不为空时

{

\*x=Qu->data[Qu->end2];

Qu->end2=(Qu->end2-1)%MaxSize;

return 1;

第6章队 列

程序设计题典

180 数据结构程序设计题典

}

else return 0;

}

return 0;

}

void main()

{

deque Qu;

Qu.end1=3;//设置双端队列的初始状态

Qu.end2=3;

elemtype x;

cout << "a右端进队列" << endl;

enq(&Qu，'a'，1);

cout << "b右端进队列" << endl;

enq(&Qu，'b'，1);

cout << "c右端进队列" << endl;

enq(&Qu，'c'，1);

cout << "d左端进队列" << endl;

enq(&Qu，'d'，2);

cout << "e左端进队列" << endl;

enq(&Qu，'e'，2);

cout << "右端出队一次：";

deq(&Qu，&x，1);

cout << x << endl;

cout << "右端出队一次：";

deq(&Qu，&x，1);

cout << x << endl;

cout << "左端出队一次：";

deq(&Qu，&x，2);

cout << x << endl;

cout << "左端出队一次：";

deq(&Qu，&x，2);

cout << x << endl;

cout << "右端出队一次：";

deq(&Qu，&x，1);

cout << x << endl;

}

第6章 队 列

程序设计题典

数据结构程序设计题典 181

本程序的执行结果如下：

a右端进队列

b右端进队列

c右端进队列

d左端进队列

e左端进队列

右端出队一次：c

右端出队一次：b

左端出队一次：e

左端出队一次：d

右端出队一次：a

**第7 章串和模式匹配**

第7章 串

程序设计题典

数据结构程序设计题典 183

*7.1 本章摘要*

*7.1.1 知识点*

*1. 串的存储结构*

· 串的顺序存储

· 串的链式存储

*2. 串的基本运算*

· 创建一个串

· 串的复制

· 求串的长度

· 求子串

· 插入一个子串

· 删除一个子串

· 连接两个串

· 输出串

*7.1.2 内容概要*

*1. 串的基本概念*

串是0个或多个字符组成的有限序列。空串是指长度为0的串；空白串是指由一个或若

干个空格组成的串。

*2. 串的存储结构*

串也是一种线性结构，其存储方式有顺序存储方式（顺序串）和链式存储方式（链式

串）。在顺序串中，使用一个数组存放队列的元素，并用一个变量（len）存放串中的字符

个数；在链式串中，用一个单链表存储串的字符。

顺序串的类型定义如下：

第7章串

程序设计题典

184 数据结构程序设计题典

typedef struct {

char ch[MaxLen];

int len;

} strtype;

*3. 串的运算*

· create(strtype \*s，char str[])：将普通字符串str赋给串s。

· length(strtype \*s)：返回串s的长度。

· copy(strtype \*s1，strtype \*s2)：将串s2复制给串s1。

· subs(strtype \*s，int pos，int n)：返回串s中从pos位置的n个字符构成的子串。

· concat(strtype \*s，strtype \*t)：将串t连接到串s之后。

· ins(strtype \*s，strtype \*t，int i)：在串s中的第i个位置插入一个子串t。

· del(strtype \*s，int pos，int n)：从串s中删除从位置pos开始的n个字符构成的子串。

· disp(strtype \*s)：输出串s的所有字符。

*4. 串的模式匹配*

子串定位运算称为串的模式匹配，其运算为index(strtype \*s，strtype \*t)：返回子串t在

主串s中首次出现的位置；如果s中未出现t，则返回-1。

*7.2 例题解析*

【例7.1】设计串结构并实现串的基本运算。

【解】 顺序串“a1a2⋯an”的存储结构如图7.1所示，该串的长度为n，第1个字符为'a1'、

第2个字符为'a2'，⋯，第n个字符为'an'，串以字符'\0'标志结束。

图 7.1 顺序串结构

第7章 串

程序设计题典

数据结构程序设计题典 185

实现本题功能的string.cpp文件如下：

#include <iostream.h>

#include <string.h>

#define MaxLen 20

typedef struct

{

char ch[MaxLen];

int len;

} strtype;

void create(strtype \*s，char str[]) //将普通字符串赋给串

{

strcpy(s->ch，str);

s->len=strlen(str);

}

int length(strtype \*s) //求串的长度

{

return s->len;

}

void copy(strtype \*s1，strtype \*s2) //串的复制

{

int i;

for (i=0;i<s1->len;i++)

s2->ch[i]=s1->ch[i];

s2->len=s1->len;

s2->ch[s2->len]='\0'; //添加字符串结束符

}

strtype subs(strtype \*s，int pos，int n) //求子串

{

int i;

strtype sub;

if (pos+n-1>length(s)) //参数不正确

sub.len=0;

else

{

for (i=pos-1;i<pos+n-1;i++)

sub.ch[i-pos+1]=s->ch[i];

第7章串

程序设计题典

186 数据结构程序设计题典

sub.len=n;

sub.ch[sub.len]='\0';

}

return sub;

}

int concat(strtype \*s，strtype \*t) //连接两个串

{

int i;

if (s->len+t->len>MaxLen)

return 0;

for (i=0;i<t->len;i++)

s->ch[i+s->len]=t->ch[i];

s->len=s->len+t->len;

s->ch[s->len]='\0';

return 1;

}

int ins(strtype \*s，strtype \*t，int i) //插入一个子串

{

int j;

if (s->len+t->len>MaxLen)

return 0;

for (j=s->len-1;j>=i-1;j--) //i之后的所有元素后移t->len个位置

s->ch[j+t->len]=s->ch[j];

for (j=0;j<t->len;j++)

s->ch[j+i-1]=t->ch[j];

s->len=s->len+t->len;

s->ch[s->len]='\0';

return 1;

}

int del(strtype \*s，int pos，int n) //删除一个子串

{

int i;

if (pos+n>s->len)

return 0;

for (i=pos+n-1;i<s->len;i++)

s->ch[i-n]=s->ch[i];

s->len=s->len-n;

第7章 串

程序设计题典

数据结构程序设计题典 187

s->ch[s->len]='\0';

return 1;

}

void disp(strtype \*s) //输出串

{

if (s->len==0)

cout << "空串" << endl;

else

cout << s->ch << endl;

}

【例7.2】采用顺序存储方式存储串，编写一个置换函数，将串s1中的第i个字符开

始的j个字符（包括第i个字符)构成的子串用s2串进行替换，函数名为replace(s1，i，j，s2)。

例如：replace("abcd"，1，3，"xyz")返回"xyzd"。

【解】 先提取s1中位置i之前的所有字符构成的子串str1，再提取位置i+j-1及之后的所

有字符构成的子串str2，最后将str1，s2，str2连接起来便构成了结果串。实现本题功能的程

序如下：

#include <stdio.h>

#include "string.cpp"

strtype replace(strtype \*s1，int i，int j，strtype \*s2)

{

strtype s;

int n，k;

if (i+j-1<=s1->len)

{

for (n=0;n<i-1;n++) //把s1的前i-1个字符赋给s

s.ch[n]=s1->ch[n];

for (n=0;n<s2->len;n++) //连接s2串

s.ch[i+n-1]=s2->ch[n];

s.len=i+s2->len-1;

for (n=s.len，k=i+j-1;k<s1->len;n++，k++)

//连接s1的位置i及之后的字符

s.ch[n]=s1->ch[k];

s.len=n;

s.ch[s.len]='\0';

}

else

第7章串

程序设计题典

188 数据结构程序设计题典

{

s.ch[0]='\0';

s.len=0;

}

return(s);

}

void main()

{

strtype s，s1，s2;

int pos，n;

char str[MaxLen];

printf("字符串：");

gets(str);

create(&s1，str);

cout << "位置：";

cin >> pos;

cout << "长度：";

cin >> n;

printf("子串：");

gets(str);

create(&s2，str);

s=replace(&s1，pos，n，&s2);

disp(&s);

}

本程序的执行结果如下：

字符串：12345678l

位置：3l

长度：4l

子串：abcdefghl

12abcdefgh78

【例7.3】采用顺序结构存储串，编写一个函数index(s1，s2)，用于判定s2是否是s1

的子串。若是子串，返回其在主串中的位置；否则返回-1。

第7章 串

程序设计题典

数据结构程序设计题典 189

【解】 本题的算法思想是：设s1="a1a2⋯am"

s2="b1b2⋯bn"

从s1中找与b1匹配的字符ai，若ai=b1，则判断是否ai+1=b2，⋯，ai+n-1=bn，若都相等，则

S2为S1子串；否则继续比较ai之后的字符。

实现本题功能的程序如下：

#include <stdio.h>

#include "string.cpp"

int index(strtype \*s1，strtype \*s2)

{

int i=0，j，k;

while (i<s1->len)

{

j=0;

if (s1->ch[i]==s2->ch[j])

{

k=i+1;j++;

while (k<s1->len && j<s2->len && s1->ch[k]==s2->ch[j])

{

k++;j++;

}

if (j==s2->len) //s2终止时找到了子串

break;

else i++;

}

else i++;

}

if (i>=s1->len)

return -1;

else

return(i+1);

}

void main()

{

strtype s1，s2;

int n;

char str[MaxLen];

第7章串

程序设计题典

190 数据结构程序设计题典

printf("字符串：");

gets(str);

create(&s1，str);

printf("子串：");

gets(str);

create(&s2，str);

n=index(&s1，&s2);

if (n>0)

cout << "子串位置：" << n << endl;

else

cout << "不是子串" << endl;

}

本程序的一次执行结果如下：

字符串：ababcabcdabcdel

子串：abcdl

子串位置：

*7.3 习题实践*

【题1】设计一个算法将串中的所有字符倒过来重新排列。

【题2】已知s="(xyz)+\*"，t="(x+z)\*y"。试利用连接、求子串和置换等操作，将s转

化为t。

【题3】已知3个字符串分别为s="ab⋯abcaabcbca⋯a"，s1="caab"，s2="bcb"，利用

所学字符串基本运算的函数得到结果串为：

s3="caabcbca⋯aca⋯a"

要求写出得到上述结果串s3所用的函数及执行算法。

【题4】利用串的基本运算，编写一个算法删除串s1中所有s2子串。

【题5】编写一个算法，计算子串s2在主串s1中出现的次数。

【题6】采用顺序结构存储串，编写一个实现串通配符匹配的函数pattern\_index()，

其中的通配符只有'? '，它可以和任一字符匹配成功，例如，pattern\_index("?re"，"there are")

返回的结果是3。

第7章 串

程序设计题典

数据结构程序设计题典 191

*7.4 参考答案*

【题1】设计思想参见例7.1。实现本题功能的程序如下：

#include <stdio.h>

#include "string.cpp"

strtype reverse(strtype \*s)

{

int i;

strtype t;

for (i=0;i<s->len;i++)

t.ch[s->len-i-1]=s->ch[i];

t.len=s->len;

t.ch[t.len]='\0';

return t;

}

void main()

{

strtype s，t;

char str[MaxLen];

printf("输入一个串：");

gets(str);

create(&s，str);

t=reverse(&s);

cout << "原串：";disp(&s);

cout << "逆置串：";disp(&t);

}

本程序的一次执行结果如下：

输入一个串：abcdefghl

原 串：abcdefghl

逆置串：hgfedcbal

【题2】本题有很多解法，一种求解程序如下：

第7章串

程序设计题典

192 数据结构程序设计题典

#include "string.cpp" //replace()为例7.2中的函数

void main()

{

strtype s，s1，s2，s3，s4，s5，s6;

create(&s，"(xyz)+\*");

cout << "s=";

disp(&s);

s1=subs(&s，7，1); //s1=”\*”

s2=subs(&s，3，1); //s2=”y”

s3=subs(&s，6，1); //s3=”+”

s4=replace(&s，3，1，&s3);

s5=replace(&s4，6，1，&s1);

s6=replace(&s5，7，1，&s2);

cout << "t=";

disp(&s6);

}

本程序的执行结果如下：

s=(xyz)+\*

t=(x+z)\*y

【题3】由s1和s2产生"caabcb" ，在s中找到该子串，其位置为n，将位置n及之后的

字符构成一个串s3，再将s的前n-1个字符即"ab⋯ab"替换成"ca⋯a"，最后将s3与之连接即可。

实现本题功能的程序如下：

#include <stdio.h>

#include "string.cpp"

strtype replace(strtype \*s1，int i，int j，strtype \*s2)

//replace()为例7.2的函数

int index(strtype \*s1，strtype \*s2)//index()为例7.3的函数

void main()

{

strtype s，s1，s2，s3，s4，s5，s6，s7;

int n;

create(&s，"ab⋯abcaabcbca⋯a");

cout << "s=";disp(&s);

create(&s1，"caab");

第7章 串

程序设计题典

数据结构程序设计题典 193

cout << "s1=";disp(&s1);

create(&s2，"bcb");

cout << "s2=";disp(&s2);

s4=subs(&s1，1，3); //s4="caa"

concat(&s4，&s2); //s4="caabcb"

s5=subs(&s1，1，2); //s5="ca"

n=index(&s，&s4);

s3=subs(&s，n，length(&s)-n+1); //s3="caabcbca⋯a"

s6=subs(&s，1，n-1);//s6="ab⋯ab"

s6=subs(&s6，1，length(&s6)-1); //s6="ab⋯a"

s7=replace(&s6，1，2，&s5); //s7="ca⋯a"

concat(&s3，&s7); //s3="caabcbca⋯aca⋯a"

cout << "s3=";disp(&s3);

}

本程序的执行结果如下：

s=ab⋯abcaabcbca⋯a

s1=caab

s2=bcb

s3=caabcbca⋯aca⋯a

【题4】本题利用例7.3的index()函数和删除子串的函数循环实现。其程序如下：

#include <stdio.h>

#include "string.cpp"

void delall(strtype \*s1，strtype \*s2)

{

int n;

n=index(s1，s2);//index()为例7.3中的函数

while (n>=0)

{

del(s1，n，length(s2));

n=index(s1，s2);

}

disp(s1);

}

void main()

第7章串

程序设计题典

194 数据结构程序设计题典

{

strtype s1，s2;

char str[MaxLen];

printf("字符串：");

gets(str);

create(&s1，str);

printf("子串：");

gets(str);

create(&s2，str);

delall(&s1，&s2);

}

本程序的一次执行结果如下：

字符串：abcdabceabcfabcgl

子串：abcl

defg

【题5】对例7.3中的函数index()稍加改进成为preindex()，使其返回子串s2在主串s1

中第n个位置之后出现的位置，然后用循环判断即可。实现本题功能的程序如下：

#include <stdio.h>

#include "string.cpp"

int preindex(strtype \*s1，strtype \*s2，int n)

{

int i=n，j，k;

while (i<s1->len)

{

j=0;

if (s1->ch[i]==s2->ch[j])

{

k=i+1;j++;

while (k<s1->len && j<s2->len && s1->ch[k]==s2->ch[j])

{

k++;j++;

}

if (j==s2->len) //s2终止时找到了子串

break;

第7章 串

程序设计题典

数据结构程序设计题典 195

else i++;

}

else i++;

}

if (i>=s1->len)

return -1;

else

return(i+1);

}

int count(strtype \*s1，strtype \*s2)//求出现次数

{

int n=0，i=0;

do

{

i=preindex(s1，s2，i);

if (i!=-1)

{

i=i+length(s2);

n++;

}

} while (i!=-1);

return n;

}

void main()

{

strtype s1，s2;

char str[MaxLen];

printf("字符串：");

gets(str);

create(&s1，str);

printf("子串：");

gets(str);

create(&s2，str);

cout << "出现次数：" << count(&s1，&s2) << endl;

}

本程序的一次执行结果下：

第7章串

程序设计题典

196 数据结构程序设计题典

字符串：abcdabceabcfabcgl

子串：abcl

出现次数：4

【题6】本题和例7.3的算法相似，只是增加了'?'的处理功能。实现本题功能的程序如

下：

#include <stdio.h>

#include "string.cpp"

int pattern\_index(strtype \*subs，strtype \*s)

{

int i，j，k;

for (i=0;s->ch[i];i++)

for (j=i，k=0;

(s->ch[j]==subs->ch[k]) || (subs->ch[k]=='?');j++，k++)

if (subs->ch[k+1]=='\0')

return(i+1);

return(-1);

}

void main()

{

strtype s，t;

int n;

char str[MaxLen];

printf("主串：");

gets(str);

create(&s，str);

printf("子串：");

gets(str);

create(&t，str);

n=pattern\_index(&t，&s);

if (n>0)

cout << "子串在主串中的位置：" << n << endl;

else

cout << "主串中未找到子串" << endl;

}

第7章 串

程序设计题典

数据结构程序设计题典 197

本程序的一次执行结果如下：

主串：there arel

子串：?rel

子串在主串中的位置：3\_\_

【题6】本题和例7.3的算法相似，只是增加了'?'的处理功能。实现本题功能的程序如

下：

#include <stdio.h>

#include "string.cpp"

int pattern\_index(strtype \*subs，strtype \*s)

{

int i，j，k;

for (i=0;s->ch[i];i++)

for (j=i，k=0;

(s->ch[j]==subs->ch[k]) || (subs->ch[k]=='?');j++，k++)

if (subs->ch[k+1]=='\0')

return(i+1);

return(-1);

}

void main()

{

strtype s，t;

int n;

char str[MaxLen];

printf("主串：");

gets(str);

create(&s，str);

printf("子串：");

gets(str);

create(&t，str);

n=pattern\_index(&t，&s);

if (n>0)

cout << "子串在主串中的位置：" << n << endl;

else

cout << "主串中未找到子串" << endl;

}

第7章 串

程序设计题典

数据结构程序设计题典 197

本程序的一次执行结果如下：

主串：there arel

子串：?rel

子串在主串中的位置：3\_\_