高等学校计算机课程规划教材

教育部—微软精品课程建设立项项目

数据结构与算法教程（C++版）实验和课程设计

唐宁九 游洪跃 孙界平 朱宏 杨秋辉 主编
李炳法 伍良富 主审

清华大学出版社

教育部——微软精品课程建设立项项目

高等学校计算机课程规划教材

数据结构与算法教程（C++版）实验和课程设计

唐宁九 游洪跃 孙界平 朱宏 杨秋辉 主编

清华大学出版社

北京

内 容 简 介

本书结合 C++ 面向对象程序设计的特点，讨论了数据结构与算法基础知识，并构建了实验与课程设计，对所有算法都在 Visual C++ 6.0、Visual C++ 2005、Visual C++ 2005 Express、Dev-C++ 和 MinGW Developer Studio 开发环境中进行了严格的测试，并在作者个人网页上提供了大量的教学支持内容。

通过本书的学习，不但能迅速提高数据结构与算法的水平，同时还能提高 C++ 程序设计的能力。本书可作为数据结构、数据结构与算法分析、数据结构与算法设计、数据结构与算法等课程实验与课程设计的教材，也可供其他从事软件开发工作的读者学习参考使用。

图书在版编目（CIP）数据

数据结构与算法教程（C++ 版）实验和课程设计/唐宁九等主编. —北京：清华大学出版社，2012.10
（高等学校计算机课程规划教材）
ISBN 978-7-302-28029-3

Ⅰ. ①数… Ⅱ. ①唐… Ⅲ. ①数据结构－高等学校－教学参考资料 ②算法分析－高等学校－教学参考资料 ③C 语言－程序设计－高等学校－教学参考资料 Ⅳ. ①TP311.12 ②TP312

中国版本图书馆 CIP 数据核字（2012）第 023033 号

责任编辑：汪汉友　徐跃进
封面设计：傅瑞学
责任校对：梁　毅
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795954
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：10.75　　**字　　数**：255 千字
版　　次：2012 年 10 月第 1 版　　**印　　次**：2012 年 10 月第1次印刷
印　　数：1～3000
定　　价：19.50 元

产品编号：042634-01

前　言

数据结构与算法内容丰富，包含了计算机科学与技术的许多重要方面，对学生的计算机软件素质的培养作用明显。要使学生能合理地选用合适的数据结构与算法编写质量高、风格好的应用程序，需要不断地进行编程实践，把实验与课程设计实践环节与理论教学相融合，通过实践教学促进数据结构与算法理论知识的学习。

本书是唐宁九等主编，清华大学出版社出版的《数据结构与算法教程(C++ 版)》(后面简称为主教材)的配套教材。全书共分为 3 部分：第一部分介绍数据结构与算法基础知识，第二部分为实验，第三部分讨论数据结构与算法课程设计。

第一部分包含主教材数据结构与算法的基础知识，主要目的是为没有选用主教材的读者通过第一部分的学习，可以顺利完成数据结构与算法的实验与课程设计。

第二部分包含 8 个实验题目，这些实验题目包括了主教材正文内容的不同实现方式，例如实现不带头结点形式的单链表；包括了对主教材内容的改进，例如对主教材的哈夫曼树类模板的方法 EnCode 加以改进，将查找字符位置通过指向函数的指针来实现，这样在具体应用时可进行优化，进而提高算法效率；包括了对主教材算法的优化，例如用赋值语句代替交换两个数据元素的方法来优化快速排序算法与堆排序，实现优化后的算法；包括了对主教材算法的改造与提高，例如改进主教材实现的求最小生成树的 Kruskal 算法；还包括了数据结结与算法的有趣应用，例如要求在一个 n×n 的棋盘上放置 n 个皇后，要求放置的 n 个皇后不会互相吃掉，皇后棋子可以吃掉任何它所在的那一行、那一列，以及那两条对角线上的任何棋子。通过实验极大地提高读者的数据结构与算法的应用能力。每个实验都有目的与要求、工具、准备工作、实验分析、实验步骤、测试与结论以及思考与感悟。实验给出具体操作步骤，给出具体与实用的指导，让初学者不会面对实验题目束手无策。希望读者通过实验能够学有所思，有所启迪与感悟。

第三部分包含 6 个课程设计项目，简单的项目可以一个人单独完成，复杂的项题目可由几个人共同完成，这些项目包括了对主教材中实例研究的改进(例如从键盘上输入中缀算术表达式，包括括号，计算出表达式的值)，包括了接近实际课题的题目(例如开发采用哈夫曼算法开发一个压缩软件，以及采用图的知识开发公园导游系统)和容易引起读者兴趣的项目(例如词典变位词检索系统)，还包括了开拓学生视野的项目(例如用具有自学习功能的专家系统思想实现“动物游戏”)。课程设计项目一般都提供功能的扩展方法，基础较差的读者可只实现基础功能，对数据结构与算法有兴趣的读者可以实现更强的功能，这样使不同层次的读者都会所有收获，通过做这些项目能快速提高读者解决实际问题的能

力。每个项目都给出了分析与实现方法,还给出了一些改进建议,读者可以在完成基本任务的前提下,对程序加以改进和提高。

对于C++的编译器的问题,在C++之外的任何编程语言中,编译器都没有受到过如此之重视。这是因为C++是一门非常复杂的语言,以至于编译器也难以构造,下面介绍一些常用的优秀C++编译器。

Visual C++ 编译器:由微软开发,现在主要流行Visual C++ 6.0、Visual C++ 2005以及Visual C++ 2005 Express,特点是集成开发环境用户界面友好,信息提示准确,调试方便,对模板支持最完善;Visual C++ 6.0对硬件环境要求低,现在安装机器最多,但对标准C++兼容只有83.43%,Visual C++ 2005与Visual C++ 2005 Express在软件提示信息上做了进一步的优化与改进,并且对标准C++兼容达到了98%以上的程度,但对硬件的要求较高;还有Visual C++ 2005 Express是一种轻量级的Visual C++软件,易于使用。对于编程爱好者、学生和初学者来说是很好的编程工具,微软在2006年4月22日正式宣布Visual Studio 2005 Express版永久免费。

GCC 编译器:著名的开源C++编译器。是类UNIX操作系统(例如Linux)下编写C++程序的首选,有非常好的可移植性,可以在非常广泛的平台上使用,也是编写跨平台、嵌入式程序很好的选择。GCC 3.3与标准C++兼容大概能够达到96.15%。现已有一些移植在Windows环境下使用GCC编译器的IDE(集成开发环境),例如Dev-C++与MinGW Developer Studio,其中Dev-C++是能够让GCC在Windows下运行的集成开发环境,提供了与专业IDE相媲美的语法高亮、代码提示,调试等功能;MinGW Developer Studio是跨平台下的GCC集成开发环境。目前支持WindowS、Linux和FreeBSD;根据作者的实际使用,感觉使用GCC编译器的IED错误信息提示的智能较低,有时一个简单的错误都会显示上百条错误信息,还有就是对模板支持较差,对语法检查较严格,在Visual C++编译器中编译通过的程序可能在GCC编译器的IED还会显示有错误信息。

本书所有算法都同时在Visual C++ 6.0、Visual C++ 2005、Visual C++ 2005 Express、Dev-C++和MinGW Developer Studio中通过测试。读者可根据实际情况选择适当的编译器,建议选择Visual C++ 6.0。

为满足不同层次的教学需求,本教材使用了分层的思想,分层方法如下:没有加星号*及**的部分是基本内容,适合所有读者学习;加有星号*的部分适合计算机专业的读者深入学习的选学部分;加有星号**的部分适合感兴趣的同学研究,尤其适合那些有志于ACM竞赛的读者加以深入研究。作者为本书提供了全面的教学支持,如果在教学或学习过程中发现与本书有关的任何问题都可以与作者联系:youhongyue168@gmail.com,作者将尽力满足各位的要求,并可能将解答公布在作者的教学网站http://teachhelp.changeip.net:9988/或http://teachhelp.3322.org:9988/上。在教学网站上还将提供如下内容:

(1)向教师提供书中所有算法在Visual C++ 6.0、Visual C++ 2005、Visual C++ 2005 Express、Dev-C++和MinGW Developer Studio开发环境中的测试程序,今后还会提供当时流行的C++开发环境的测试程序,对一般读者将在每学期的期末在主页网上公布解压口令。

(2) 提供本书作者开发的软件包(包含所有本书所讲的数据结构与算法的类模板与函数模板)。

(3) 补充实验指导。

(4) 数据结构与算法问答专栏。

(5) 介绍 Visual C++ 6.0、Visual C++ 2005、Visual C++ 2005 Express、Dev-C++ 和 MinGW Developer Studio 开发环境建立工程的步骤,在工程建立新文件与向工程添加已有文件的方法的文档。

(6) 提供数据结构与算法相关的其他资料(例如 Dev-C++ 与 MinGW Developer Studio 软件,流行免费 C++ 编译器的下载网址)。

希望各位读者能够抽出宝贵的时间将你对本教材的建议或意见,当然也可以发表对国内外的数据结构与算法课程教学的任何意见寄给作者,你的意见将是我们再版修订教材的重要参考,作者发自内心地感谢。

张卫华、彭骏、谭斌、李培宇、何凯霖、姜琳、聂清彬、黄维、邹昌文、王文昌、周焯华、胡开文、沈洁、周德华、欧阳、文涛和文波等人做了大量的工作,包括调试算法,提供参考资料,并参加了本书的编写工作。

本书的出版要感谢清华大学出版社各位编辑及评审专家,由于他们为本书的出版倾注了大量热情。也由于他们具有前瞻性的眼光才让读者有机会看到本书。

尽管作者本着负责任的严格态度,并尽了最大努力,但由于作者水平有限,书中难免有不妥之处,因此,敬请各位读者不吝赐教,以便作者有一个提高的机会,并在再版时尽力采用你们的意见,尽快提高本书的水平。

作 者

2012 年 1 月

目　　录

第一部分　基础知识

本书是《数据结构与算法教程(C++ 版)》(以下简称“主教材”)的配套教材,可能有些读者没学过主教材,因此直接做配套的实验与课程设计是非常困难的,就是学过配套教材,也可能有些内容有所遗忘,因此本书特增加第一部分,专门讲解数据结构与算法的基础知识,以便使本书自成体系,可为读者独立做数据结构与算法的实验与课程设计打下基础。

1.1　绪　　论

1.1.1　数据结构的基本概念

数据:数据是客观事物的符号表示,是计算机中可以操作的对象,也就是一切能输入到计算机中并能被处理的符号的总称。

数据元素与数据项:数据元素一般在计算机中能作为整体进行处理,是数据的基本单位,数据元素也称为记录,有的数据元素由若干**数据项**所组成,例如在员工基本信息表中,每个员工记录是一个数据元素,而员工的编号、姓名、性别、籍贯、家庭住址和生日等内容为数据项,数据项是不可分割的最小单位。

数据结构:在现实世界中,不同数据元素之间不是独立的,而是存在着特定的关系,我们将这些关系称为**结构**,**数据结构**指相互之间存在着一定关系的数据元素的集合。

为了方便起见,用示意图表示数据结构,这种图称为**逻辑结构图**,具体表示为:用小圆圈表示数据元素,用小圆圈之间的带有箭头的线段表示数据元素的有序对,具体地讲对于有序对<u,v>可表示为如图 1.1.1 所示的形式。

图 1.1.1　有序对<u,v>示意图

u 称为 v 的直接前驱,简称为前驱,v 称为 u 的直接后继,简称为后继,数据元素之间的**关系**定义为有序对的集合。

根据数据元素之间关系的特性,有如下四类基本结构。

(1) 集合结构:在数据结构中,如果不考虑数据元素之间的关系,这种结构称为**集合结构**,在集合结构中,各个数据元素是“平等”的,它们的共同属性是“同属于一个集合”,如图 1.1.2 所示。

(2) 线性结构:**线性结构**中的数据元素之间存在一个对应一个的关系,也就是除了第一个数据元素没有前驱,最后一个数据元素无后继而外,其他数据元素都有唯一的前驱

和后继，如图 1.1.3 所示。

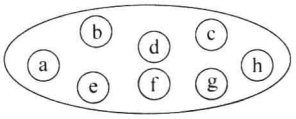

图 1.1.2 集合结构示意图

图 1.1.3 线性结构示意图

(3) 树状结构：**树状结构**中的数据元素之间存在着一个对应多个的关系，数据元素之间存在着层次关系，也就是除了一个特殊的称为树根的数据元素无前驱外，其他数据元素都有唯一的前驱，如图 1.1.4 所示。

(4) 图状结构：图状结构结构中的数据元素之间存在多个对应多个的关系，也就是任一数据元素可能有多个前驱和后继，如图 1.1.5 所示。

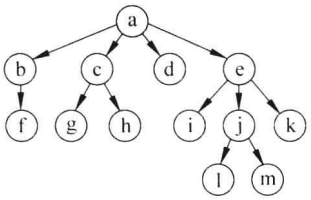

图 1.1.4 树状结构示意图

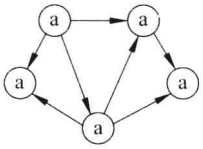

图 1.1.5 图状结构示意图

数据结构在形式上可以定义为如下二元组：

$$\text{DataStructure}=(D,S)$$

其中 D 是一个数据元素的集合，S 是定义在 D 中的数据元素之间的关系的有限集合。

上面数据结构的形式定义实际上是一种数学描述，也就是从解决问题的实际出发，为实现必要的功能建立数学模型，其结构定义中的关系用于描述数据元素之间的逻辑关系，是面向问题的，称为**逻辑结构**；数据结构在计算机中的表示称为**物理结构**或**存储结构**，物理结构是面向计算机的。

数据类型：**数据类型**是指一组性质相同的值的集合以及定义在此集合上的一些操作的总称。

抽象数据类型：**抽象数据类型**指定义用于表示应用问题的数学模型，以及定义在此数学模型上的一组操作(也可称为服务或方法)的总称。

1.1.2 算法和算法分析

1. 算法

算法是解决特定问题求解步骤的描述，在计算机中为指令的有限序列，并且每条指令表示为一个或多个操作。对于给定的问题可以有多种算法来解决，本书的有些问题给出了多种算法。

2. 算法分析

对于一个算法的评价，首先应考虑算法的正确性，其次是运算量（即运行效率的高低），有时还要考虑算法所占的存储空间的大小，为定性分析引入了时间复杂度与空间复杂度的概念，本书重点考虑时间复杂度。

时间复杂度：一个特定算法的“运行工作量”的大小，一般依赖于问题的规模（通常用整数量 n 表示），或者说，它是问题规模的函数。算法中基本操作执行次数通常为问题规模 n 的某个函数 f(n)，则算法时间度量为：T(n)＝O(f(n))。①

O(f(n))称为算法的渐近时间复杂度，或简称为时间复杂度。

空间复杂度：与时间复杂度类似，算法中所需存储空间通常为问题规模 n 的某个函数 f(n)，则算法空间度量为：S(n)＝O(f(n))。

O(f(n))称为算法的空间复杂度。

1.1.3　实用程序软件包

一些语句在逻辑上不相关，但却经常使用，将它们收集起来形成实用程序软件包，今后所有程序都可用它，在本书开发的所有程序中几乎都包含“文件包含”处理：

```
#include "utility.h"
```

这样允许程序访问此实用程序软件包的内容。

1. 标准库系统

在实用程序软件包中包含常用的标准库系统，ANSI C++ 中的标准库包含在命名空间 std 中，为了能不带域解析运算符使用标准库，在 ANSI C++ 实用程序软件包中加入命令 using namespace std，具体内容如下：

```
#include<string>                    //标准串和操作
#include<iostream>                  //标准流操作
#include<limits>                    //极限
#include<cmath>                     //数据函数
#include<fstream>                   //文件输入输出
#include<cctype>                    //字符处理
#include<ctime>                     //日期和时间函数
#include<cstdlib>                   //标准库
#include<cstdio>                    //标准输入输出
#include<iomanip>                   //输入输出流格式设置
#include<cstdarg>                   //支持变长函数参数
#include<cassert>                   //支持断言
using namespace std;                //标准库包含在命名空间 std 中
```

① O一般读为大O，对于非负函数 T(n)，若存在两个正常数 c 和 n_0，对任意 $n>n_0$，有 $T(n)\leqslant cf(n)$，则称 T(n) 在集合 O(f(n))中，一般简写为 T(n)＝O(f(n))。

2. 定义宏 DEFAULT_SIZE

有些存储结构的构造函数在初始化时需要提供元素个数的默认值的宏，有时还需要定义表示无穷大的宏，为方便起见，专门定义宏如下：

```
#define DEFAULT_SIZE 100                    //默认元素个数
#define DEFAULT_INFINITY 1000000            //默认无穷大
```

3. UserSaysYes()函数

在很多程序中都可加入实用 UserSaysYes()函数，而在标准库中并没有此函数，为此在 utility.h 中加入此函数的定义，具体定义如下：

```
static bool UserSaysYes()
//操作结果：当用户肯定回答(yes)时，返回 true，用户否定回答(no)时，返回 false
{
    char ch;                                //用户回答字符
    bool initialResponse=true;              //初始回答

    do
    {                                       //循环直到用户输入恰当的回答为止
        if (initialResponse) cout<<"(y, n)?"; //初始回答
        else cout<<"用 y 或 n 回答:";         //非初始回答
        while ((ch=cin.get())==' '||ch=='\t'||ch=='\n');
                                            //跳过空格，制表符及换行符获取一字符
        while (cin.get() !='\n');           //跳过当前行后面的字符
        initialResponse=false;
    } while (ch !='y' && ch !='Y' && ch !='n' && ch !='N');

    if (ch=='y'||ch=='Y') return true;      //肯定回答
    else return false;                      //否定回答
}
```

4. 定时器类 Timer

在比较不同算法与数据结构时，了解一个程序与另一个程序运行的计算机时间是非常有用的，为此目的，开发了一个实用的定时器类 Timer，它的构造函数用于启动定时器的工作，如果要重新设置定时器，可使用 Timer 类的方法 Reset()，方法 Elapsed_Time()用于返回从 Timer 对象启动或最后一次调用方法 Reset()后所使用的 CPU 时间。

在 C++ 系统中提供了头文件 ctime 或 time.h，它包含了标准函数 clock()以及类型 clock_t，函数 clock()返回程序从开始运行到当前经过的滴答(ticks)数，函数 clock()返回值类型为 clock_t，clock_t 在 Visual C++ 6.0 中的声明如下：

```
typedef long clock_t;
```

说明：在 C++ 中，一秒钟的滴答(ticks)数等于 CLK_TCK，所以时间间隔的滴答(ticks)数除以 CLK_TCK 就是时间间隔的秒数。

Timer 类的定义如下：

```
//定时器类 Timer
class Timer
{
private:
//数据成员
    clock_t startTime;                                    //起始时间

public:
//方法声明
    Timer() {startTime=clock();}                          //构造函数
    ~Timer() {};                                          //析构函数
    double ElapsedTime()                                  //返回已过的时间
    {
        clock_t endTime=clock();                          //结束时间
        return (double)(endTime-startTime)/(double)CLK_TCK;
            //返回从 Timer 对象启动或最后一次调用 reset()后所使用的 CPU 时间
    }
    void Reset() {startTime=clock();}                     //重置开始时间
};
```

5. 有关随机数的函数

在编写测试程序时，使用随机数生成测试数据更具有代表性，为了更好地应用，特编写有关随机数的几个函数组成随机数类，具体定义如下：

```
//随机数类 Rand
class Rand
{
public:
//静态成员函数
    static void SetRandSeed() {srand((unsigned)time(NULL));}
                                                          //设置当前时间为随机数种子
    static int GetRand(int n) {return rand()%(n);}        //生成 0~n-1 之间的随机数
    static int GetRand() {return rand();}                 //生成随机数
};
```

6. 其他实用函数

下面是本书中经常用到的实用函数，使用这些函数将使编程更简单。

```
template<class ElemType>
void Swap(ElemType &e1, ElemType &e2)
//操作结果：交换 e1, e2 之值
{
    ElemType temp=e1; e1=e2; e2=temp;                     //循环赋值实现交换 e1, e2
```

```
}

template<class ElemType>
void Show(ElemType elem[], int n)
//操作结果：显示数组 elem 的各数据元素值
{
    for (int i=0; i<n; i++)
        cout<<elem[i]<<"  ";                          //显示数组 elem
    cout<<endl;                                       //换行
}

template<class ElemType>
void Show(const ElemType &e)
//操作结果：显示数据元素
{
    cout<<e<<"  ";                                    //输出 e
}
```

1.2 线 性 表

1.2.1 线性表的逻辑结构

1. 线性表的概念

线性表是由类型相同的数据元素组成的有限序列，不同线性表的数据元素类型可以不同，可以是最简单的数值和字符，也可以是比较复杂的信息。

2. 线性表的形式定义

线性表的形式定义如下：

$$\text{Linear\_list}=(D,R)$$

其中 $D=\{a_i \mid a_i \in \text{ElemSet}, i=1,2,3,\cdots,n, n\geqslant 0\}$，ElemSet 为某个数据集合；$R=\{N\}$，$N=\{<a_i,a_{i+1}> \mid i=1,2,\cdots,n-1\}$。

说明：

(1) N 为一个有序偶的集合，用于表示线性表中数据元素之间的相邻关系。

(2) 线性表中数据元素的个数 n 称为线性表的长度，当 n=0 时称为空表。

其中，a_1 为第一个数据元素，a_n 为最后一个数据元素；$i=1,2,\cdots,n-1$ 时，a_i 有且仅有一个直接后继；当 $i=2,3,\cdots,n$ 时，a_i 有且仅有一个直接前驱。

3. 线性表的基本操作

在实际应用中，线性表还包括了如下基本操作。

1) int Length() const

初始条件：线性表已存在。

操作结果：返回线性表元素个数。

2) bool Empty() const

初始条件：线性表已存在。

操作结果：如线性表为空，则返回 true，否则返回 false。

3) void Clear()

初始条件：线性表已存在。

操作结果：清空线性表。

4) void Traverse(void (* visit)(const ElemType &)) const

初始条件：线性表已存在。

操作结果：依次对线性表的每个元素调用函数(* visit)。

5) bool GetElem(int position,ElemType &e) const

初始条件：线性表已存在，1≤position≤Length()。

操作结果：用 e 返回第 position 个元素的值。

6) bool SetElem(int position,const ElemType &e)

初始条件：线性表已存在，1≤position≤Length()。

操作结果：将线性表的第 position 个位置的元素赋值为 e。

7) bool Delete(int position,ElemType &e)

初始条件：线性表已存在，1≤position≤Length()。

操作结果：删除线性表的第 position 个位置的元素，并用 e 返回其值，长度减 1。

8) bool Insert(int position,const ElemType &e)

初始条件：线性表已存在，1≤position≤Length()+1。

操作结果：在线性表的第 position 个位置前插入元素 e，长度加 1。

在具体实现时，还可能包括其他操作，但本书开发的软件包中不但都包含上述基本操作，还根据 C++ 语言的特点加入了一些其他操作，如赋值操作(由赋值语句重载实现)、由已知线性表构造新线性表(由复制构造函数实现)等；C++ 中实现上面的操作通常也称为方法。

1.2.2　线性表的顺序存储结构

在顺序实现中，数据存储在一个长度为 maxSize，数据类型为 ElemType 的数组中，并用 count 存储数组中存储的线性表的实际元素个数，线性表的顺序存储结构有如下特点。

(1) 线性表的顺序存储结构用一组地址连续的存储单元依次存储线性表的元素；

(2) 线性表的顺序存储结构用元素在存储器中的“物理位置相邻”表示线性表中数据元素之间的逻辑关系，设 $LOC(a_i)$ 表示数据元素 a_i 的存储位置，L 为每个元素占用的存储单元个数，有如下关系：

$$LOC(a_{i+1}) = LOC(a_i) + L$$

$$LOC(a_i) = LOC(a_1) + (i-1) * L$$

(3) 线性表的顺序存储结构可直接随机存取任一个数据元素,线性表的顺序存储结构是一种随机存取的存储结构;

(4) 在进行插入或删除操作时需移动大量的数据元素。

1.2.3 线性表的链式存储结构

顺序存储结构可随机存取线性表中的任一元素,但作插入或删除操作时需要移动大量的元素,链式存储结构不要求逻辑上相邻的数据元素在物理位置上也相邻,插入删除操作时不需要移动元素。

1. 单链表

1) 结构描述

单链表是一种最简单的线性表的链式存储结构,单链表也称为线性链表,用它来存储线性表时,每个数据元素用一个结点(node)来存储,一个结点由两个成员组成,一个是存放数据元素的 data,另一个是存储指向此链表下一个结点(即后继)的指针 next,如图 1.2.1 所示,如 p 指向结点,则结点的数据元素为 p->data,指针为 p->next,p->next 指向结点的后继。

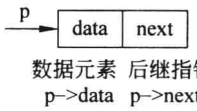

图 1.2.1 单链表结点示意图

一个线性表(a_1,a_2,a_3,…,a_n)的单链表结构通常如图 1.2.2 所示,在图中最前端增加了一个结点,这个结点没有存储任何数据元素,称为头结点,在单链表中增加头结点虽然增加了存储空间,但算法实现更简单,效率更高,单链表的头结点的地址可从指针 head 找到,指针 head 称为头指针,其他结点的地址由前驱的 next 得到。

注:线性链表也可以没有头结点,读者可作为练习加以实现。

当单链表中没有数据元素时,只有一个头结点,这时 head->next==NULL,如图 1.2.3 所示。

图 1.2.2 单链表结构示意图

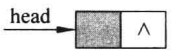

图 1.2.3 空单链表示意图

2) 线性表的链式存储结构的特点

线性表的链式存储结构有如下特点:

(1) 数据元素之间的逻辑关系由结点中的指针表示;

(2) 每个元素的存储位置由其直接前驱的指针所指示;

(3) 线性表的链式存储结构是非随机存取的存储结构;

(4) 线性表的链式存储结构中的尾结点的直接后继为空(即此结点的指针成员为空)。

2. 循环链表

循环链表是另一种形式的线性表链式存储结构,它的结点结构与单链表相同,与单链表不同的是在循环链表中表尾结点的 next 指针不空(NULL),而是指向头结点,如图 1.2.4(a)所示,循环链表为空的条件为 head->next==head,如图 1.2.4(b)所示。