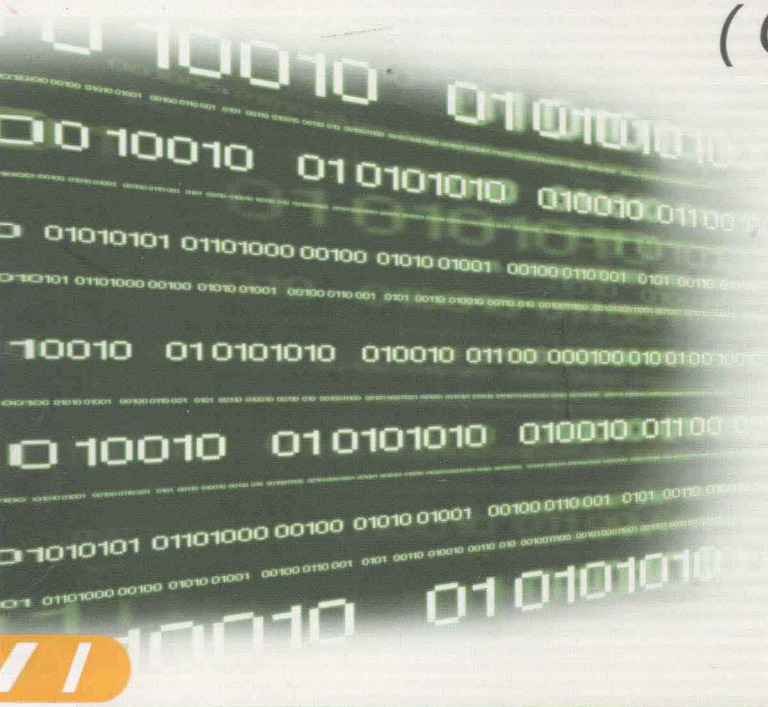
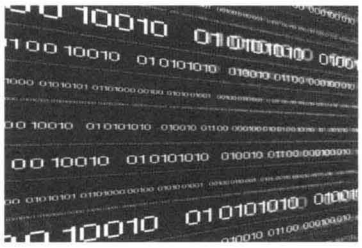
数据结构实验教程

（C/C++语言版）

张 仕 严晓明 编著



厦门大学出版社 国家一级出版社
XIAMEN UNIVERSITY PRESS 全国百佳图书出版单位



数据结构实验教程

（C/C++语言版）

张 仕　严晓明　编著

厦门大学出版社 国家一级出版社
XIAMEN UNIVERSITY PRESS 全国百佳图书出版单位

图书在版编目(CIP)数据

数据结构实验教程:C/C++语言版/张仕,严晓明编著. —厦门:厦门大学出版社,2013.6
ISBN 978-7-5615-4611-6

Ⅰ.①数… Ⅱ.①张…②严… Ⅲ.①数据结构-师范大学-教材②C语言-程序设计-师范大学-教材
Ⅳ.①TP311.12②TP312

中国版本图书馆CIP数据核字(2013)第100871号

厦门大学出版社出版发行
(地址:厦门市软件园二期望海路39号 邮编:361008)
http://www.xmupress.com
xmup@xmupress.com
厦门市金凯龙印刷有限公司印刷
2013年6月第1版 2013年6月第1次印刷
开本:787×1092 1/16 印张:8.5 字数:206千字
定价:18.00元
本书如有印装质量问题请直接寄承印厂调换

前 言

数据结构不但是计算机科学与技术专业的核心课程，而且逐渐成为了所有理工科专业的重要选修课程。其主要研究内容是数据的逻辑结构、存储结构及其操作，目的在于培养学生的数据抽象能力，为学生解决实际问题时进行数据的组织和操作奠定基础。数据结构实验的目的正是配合完成数据结构课程的教学要求。同时，该课程的另一现实教学目的是训练学生进行复杂程序设计的基本技能，培养良好程序设计的习惯，提高学生动手能力，加深对数据结构定义、算法、算法应用的理解。

1. 写作目的

数据结构是软件设计师的必备工具，但是在众多专业课中，数据结构却被很多学生认为是一门很难学习的课程。通过学生的反馈，笔者发现理解数据结构中的线性表、队列、树等概念和操作并不难，数据结构真正的难点在于：(1)难以实现从数据结构概念到程序实现的跨越(即如何实现一个数据结构)；(2)难以实现从实际应用到数据结构抽象的跨越(即利用数据结构解决实际问题)。为此，提高学生在这门课程上的实际动手能力就显得尤为重要。基于此，笔者编写本书，希望能够在具体的实验中为学生提供一些指导，使学生能够掌握数据结构的理论和抽象到实现、应用的方法。

2. 本书结构

本书第 1 章为抽象数据类型，该章以复数为例给出抽象数据类型的定义和实现；第 2 章至第 7 章分别是线性表、栈、串、二叉树、图、查找和排序六个重要结构的实现和应用；第 8 章则综合利用了所学结构和算法，给出一个综合实验。本书附录一是实验报告的基本格式和内容要求；附录二是一个在 VS2008 下进行调试程序、发现问题的实例。

3. 本书的使用

在学习数据结构这门课以及进行数据结构实验时，假定学生已经掌握至少一门程序设计语言，并且具备初步的程序设计能力。虽然本书中的所有实例使用 C 或 C++语言实现，但是也可以为使用其他语言实现数据结构提供参考。

为了能够更好地复习所学知识，并在实际中加以应用，实验 1、实验 2、实验 4、实验 7 及实验 8 采用 C 语言作为参考程序的实现语言，而实验 3、实验 5 和实验 6 则采用面向对象的 C++语言作为参考程序的实验语言。在使用本书时，也可以对比面向过程和面向对象的实现方法，以理解它们的实质。本书的参考程序尽量从各种数据结构的通用性出发，考虑其实际的应用，体现了一次构造多次使用的思想。

本书可以作为数据结构课程学习过程中的配套教材使用，也可以为爱好程序设计和自学程序设计的人员提供参考。教师可以根据具体情况调整各部分的课时分布。

4. 一些说明

在一些具体的应用中，需要把数据结构各种操作定义(若是类，则是类定义)写在“.h”文件中，把具体的实现写在“.cpp”文件中。当使用模板时，若把定义和实现分开存放，需要

在应用时引入实现文件".cpp",这在某种程度上丧失了".h"定义的作用。所以,在本实验指导中,凡是使用模板的参考程序,均把".h"和".cpp"文件合并在一起,也就是实现和定义统一存放在".h"文件中,以简化理解和使用。

5. 致谢

本书得到了福建师范大学数学与计算机科学学院郭躬德教授、严宣辉副教授认真细致的审阅,厦门大学出版社的有关人员也为本书的出版做了大量细致的工作,在此,特向他们表示由衷的谢意。由于作者水平和认识所限,本书难免存在疏漏之处,还请读者批评指正。

编写者

2013 年 5 月

目 录

第 1 章　抽象数据类型

抽象数据类型(Abstract Data Type,简称 ADT)是指一个数学模型以及定义在该模型上的一组操作,或者定义为由一个值域和定义在该值域上的一组操作。抽象数据类型实际上是定义数据结构的一种通用的工具,它与具体的语言、具体的实现环境无关。

1.1 实验目的

1. 掌握程序设计的基本方法,复习 C/C++语言,并实现简单的算法设计。
2. 掌握结构体类型/类的定义方法以及自定义数据类型的使用。
3. 掌握函数的设计及调用。
4. 掌握抽象数据类型的基本概念、定义。
5. 学会理解给定的 ADT 描述,从而利用程序语言加以实现。
6. 学习基本的程序组织方法和对抽象数据类型实现的应用。
7. 熟悉开发环境、程序调试的方法。

1.2 实验内容及要求

1.2.1 实验内容

1. 设计一个复数的抽象类型定义。
2. 实现一个结构体类型/类描述“复数”的结构定义。
3. 实现复数的初始化、加法、减法、乘法,以及求复数的实部、虚部等基本操作。
4. 编写函数对所实现的抽象数据类型进行测试,验证其正确性。

1.2.2 实验要求

1. 提前预习 C/C++语言中结构体类型/类的定义与使用方法。

2. 对所定义数据类型的应用中,要求通过该类型所提供的函数/类方法实现对该结构体类型/类内部数据的操作,而不是直接访问结构体类型/类的内部数据,从而达到抽象和封装的效果。

3. 对所定义类型的操作,要求完成复数的加、减、乘、除四则运算,并获取和设置复数实、虚部,初始化等操作。

4. 编写完整的程序,完成实验内容,并上机调试和运行。

5. 整理并上交实验报告。

6. 本次实验要求在 2 学时内完成。

1.3 知识点提示

1.3.1 抽象类型定义

抽象数据类型用一个三元组(D,S,P)表示:其中 D 表示数据对象,S 是 D 上的关系集,P 是对数据对象 D 的基本操作集合。

(1)抽象数据类型的定义格式:

ADT 抽象类型名{

数据对象:<数据对象定义>

数据关系:<数据关系定义>

基本操作:<基本操作定义>

}ADT 抽象数据类型名

(2)抽象数据类型中基本操作的定义格式:

基本操作名(参数表)

初始条件:<初始条件描述>

操作结果:<操作结果描述>

赋值参数:为操作提供输入值;

引用参数:既可为操作提供输入值,还返回操作结果;

初始条件:指操作前数据结构和参数应满足的条件。

若不满足,操作失败,返回相应错误信息。

初始条件为空,省略之。

操作结果:操作正常完成,数据结构的变化和返回结果。

在实际使用抽象数据类型定义某一数据类型时,常常会省略其中的若干部分,从而达到对定义的灵活应用。例如,定义复数加法操作时没有涉及操作定义中的引用参数,那么便可以忽略该部分。

1.3.2 复数

形如 $z=a+bi$(其中 a,b 是任意实数)的数称为复数(complex number),其中规定 i 为虚数单位,且 $i^2=i*i=-1$。将复数 $z=a+bi$ 中的实数 a 称为复数 z 的实部(real part);实数 b 称为复数 z 的虚部(imaginary part)。特殊的,当 $b=0$ 时,$z=a$,这时复数成为实数;而当 $a=0$ 且 $b\neq0$ 时,$z=bi$,则将其称为纯虚数。

1.3.3 抽象数据类型的复数例

```
ADT Complex{
  D:    {real,imag| e1,e2 为实数}
  R:    {<real,imag>}
  P:    float SetReal(Complex x,float real)
            赋值参数:所要操作的复数 x 和将为复数 x 赋的实部 real
```

操作结果：复数 x 的实部设置为 real

float SetImag(Complex x,float imag)

赋值参数：所要操作的复数 x 和将为复数赋的实部 imag

操作结果：复数 x 的虚部设置为 imag

Complex add(Complex x,Complex y)

赋值参数：所要相加的两个复数 x,y

操作结果：返回两个复数 x 与 y 的和

Complex sub(Complex x,Complex y)

赋值参数：相减操作的被减数 x,减数 y

操作结果：返回复数 x－y 的结果

Complex multi(Complex x,Complex y)

赋值参数：相乘操作的被乘数 x,乘数 y

操作结果：返回两个复数 x 与 y 的乘积

Complex except(Complex x,Complex y)

赋值参数：除法操作的被除数 x,除数 y

操作结果：返回复数 x 除以复数 y 的结果

void InitialComplex(Complex &x,float r,float i)

赋值参数：所要初始化复数 x,实数 r,实数 i

操作结果：设置复数 x 的实部为 r,虚部为 i

float GetReal(Complex x)

赋值参数：复数 x

操作结果：返回复数 x 的实部

float GetImag(Complex x)

赋值参数：复数 x

操作结果：返回复数 x 的虚部

}

1.3.4 结构体类型定义

Typedef 是 C/C++语言的关键字，其作用是用于定义一种新的数据类型(也可以理解为是给一种数据类型定义一个新名字)。定义新数据类型的基础可以是程序内部基础数据类型 int、float、char 等，也可以是已经定义的结构体类型。其用法为：

typedef dataType newDataTypeName

例如，函数返回值常用于表示函数执行的状况，则可以定义：typedef int STATUS;该语句定义一个类型 STATUS，专门用于对应函数的返回值，让程序员能够形成条件反射，也让程序变得更具可读性。

在本实验中，将定义一个新的数据类型 Complex，它是由若干部分数据组成的，这时可以利用结构体定义该类型如下：

```
typedef struct {
  float real;
```

```
    float imag;
}Complex;
```

在此定义之后,语句"Complex complex;"就表示定义了一个结构体变量 Complex,该结构体类型中包括两个基本元素,分别是浮点类型的 real 和浮点类型的 imag。

1.3.5 程序的组织

写程序时,很多同学习惯把所有的程序都放到一个文件中保存。当所写程序稍微有些复杂时,就非常容易导致程序文件过于庞大、杂乱,从而降低了程序的可读性。

我们知道,简洁的东西容易理解,过于复杂的东西则不易把握。试想,如果一本书没有章节、没有目录,那么整本书的结构和脉络将会难以把握。一下子展示全部的细节反而会使整体结构模糊不清,写程序也是如此。

基于此,在设计程序时,应该单独定义各个抽象数据类型的实现,让它们成为相对独立的个体,并且尽量减少个体之间的依赖性,以便它们在多数场合都可以单独使用。如何做到呢?其中,文件就是一个最基本的分类组织工具。在组织文件时,应尽量把具有相同特性、属于同一抽象数据类型的定义和该抽象数据类型的操作函数放到同一个文件中,而不是把所有代码都堆积到一个文件中。之后,通过文件之间的相互引用,把这些文件整合成为一个可运行的项目。

以本次实验为例,要实现一个复数及其基本操作,同时对该复数要加以应用,所以可以利用两个文件来组织本次实验的程序代码。其中一个文件"Complex.h"专门用于定义复数的结构以及复数所要实现的函数的定义,而"Complex.cpp"则用于复数基本操作的实现,而另外一个对复数加以应用的文件(包含 main 函数以实现复数的应用),只需要引入复数定义文件,同时对其加以利用便可。对复数的使用者而言,只需要关心"Complex.h"中所定义的函数头,知道如何调用每一个函数便可,而无须细读"Complex.cpp"文件中复数操作的具体实现。

当然对于一些更加大规模的程序,还可以利用头文件、文件夹/包等方式组织程序,从而让程序具有更好的可读性,这些将在以后的实验中逐步加以应用。

1.4 实验步骤

1. 熟悉复数的基本概念,掌握复数的基本组成、基本运算原理。

2. 建立本次实验的项目文件,添加一个"Complex.h"文件、一个"Complex.cpp"文件和一个"main.cpp"文件。其中"Complex.h"文件用于存储复数的结构定义和操作对应的函数头,而"Complex.cpp"文件则用于实现"Complex.h"中所定义的操作,"main.cpp"文件用于存储主函数,实现复数的应用。

3. 根据复数的基本组成,设计用于表示复数的结构体类型/类,也就是要求用结构体类型/类"封装"复数对应的实部、虚部等具体内容。

4. 根据复数抽象类型定义中所指的基本操作,定义并实现复数初始化,实部、虚部的获取和设置函数。

5. 根据复数抽象类型定义中所指的基本运算,定义并实现复数的加、减、乘、除四则运算。

6. 编写一个主函数，接收用户的输入，并根据输入构造两个复数变量。

7. 在主函数中对步骤 6 中构造的复数进行各种运算，验证各种运算结果的正确性。

8. 对以下实例进行计算，并察看结果的正确性：

(3+5i)+(-5-2i)=-2+3i

(3+5i)-(-5-2i)=8+7i

(3+5i)*(-5-2i)=-5-31i

3+5i 实部为 3，虚部为 5

9. 完成实验，撰写实验报告并提交。

1.5 实验程序参考

实验参考程序由三个文件组成，其中 Complex.h 定义了复数数据类型的结构和函数头，Complex.cpp 则实现了 Complex.h 中所定义的函数，最后 Main.cpp 中包含了对复数的简单应用。

1.5.1 Complex.h

```
/*********************************************************************
```

Complex.h 中定义了抽象数据类型 Complex，包括了结构体的定义，以及建立在结构体上的运算。对于一个抽象数据类型而言，建立相应的数据类型，和对这个抽象数据类型进行运算的定义，同样是必要的。

Complex.h 文件中定义了复数类型所有函数头，具体的实现在 Complex.cpp 中。

这些定义和操作主要包括：

1. 定义数据类型 Complex，封装了两个单精度的实数。

2. 函数 GetReal(Complex x)返回复数 x 的实部。

3. 函数 GetImag(Complex x)返回复数 x 的虚部。

4. 函数 void SetReal(Complex &x,float real)设置复数 x 的实部为 real。

5. 函数 void SetImag(Complex &x,float imag)设置复数 x 的虚部为 imag。

6. 函数 add(Complex x,Complex y)返回两个复数 x 与 y 的和。

7. 函数 sub(Complex x,Complex y)返回两个复数 x 与 y 的差。

8. 函数 multi(Complex x,Complex y)返回两个复数 x 与 y 的乘积。

9. 函数 except(Complex x,Complex y)返回复数 x 除以复数 y 的结果。

10. 函数 InitialComplex(Complex &x,float r,float i)利用输入的实虚部数据初始化复数 x。

```
*********************************************************************/
#include<stdio.h>
/* 定义数据类型 complex，封装了两个单精度的实数，其中 real 是实数，imag 是虚部。
   用typedef 把结构体类型用指定的关键字 Complex 来表示，这样可以和用 C 语言的
     类型关键字一样的形式去定义变量。 */
```

```
typedef struct {
  float real;
  float imag;
}Complex;
/* 对数据类型 Complex 的操作,在.h 文件中只定义函数头。 */
//初始化复数值
void InitialComplex(Complex &x,float r,float i);
//函数 GetReal(complex x)返回复数 x 的实部。形参为复数类型。
//返回值为单精度实数。
float GetReal(Complex x);
//函数 GetImag(complex x)返回复数 x 的虚部。形参为复数类型。
//返回值为单精度实数。
float GetImag(Complex x);
//函数 SetReal(complex x)设置复数 x 的实部。形参为复数类型以及实部值。
void SetReal(Complex &x,float real);
// 函数 SetImag(complex x)设置复数 x 的虚部。形参为复数类型以及虚部值。
void SetImag(Complex &x,float imag);
/* 函数 add(complex x,complex y)返回两个复数 x 与 y 的和。形参共有两个,都为复数类型。返回值为复数类型。 */
Complex add(Complex x,Complex y);
/* 函数 sub(complex x,complex y)返回两个复数 x 与 y 的差。形参共有两个,都为复数类型。返回值为复数类型。 */
Complex sub(Complex x,Complex y);
/* 函数 multi(complex x,complex y)返回两个复数 x 与 y 的乘积。形参共有两个,都为复数类型。返回值为复数类型。 */
Complex multi(Complex x,Complex y);
/* 函数 except(complex x,complex y)返回复数 x 除以复数 y 的结果。形参共有两个,都为复数类型。返回值为复数类型。 */
Complex except(Complex x,Complex y);
```

1.5.2 Complex.cpp

```
/*****************************************************************
以下代码均为抽象数据类型 Complex 的操作实现,主要实现了 complex.h 中定义的函数,包括初始化、设置实虚部、获取实虚部、加、减、乘、除。
*****************************************************************/
#include<stdio.h>
#include "complex.h"
//初始化复数值
void InitialComplex(Complex &x,float r,float i)
```

```
{
  x.real=r;
  x.imag=i;
}
//函数 GetReal(complex x)返回复数 x 的实部。形参为复数类型。返回值为单精度实数。
float GetReal(Complex x)
{
  return x.real;
}
//函数 GetImag(complex x)返回复数 x 的虚部。形参为复数类型。返回值为单精度实数。
float GetImag(Complex x)
{
  return x.imag;
}
//函数 SetReal(complex x)设置复数 x 的实部。形参为复数类型以及实部值。
void SetReal(Complex &x,float real)
{
  x.real=real;
}
// 函数 SetImag(complex x)设置复数 x 的虚部。形参为复数类型以及虚部值。
void SetImag(Complex &x,float imag)
{
  x.imag=imag;
}
/* 函数 add(complex x,complex y)返回两个复数 x 与 y 的和。形参共有两个,都为复数类型。返回值为复数类型。 */
Complex add(Complex x,Complex y)
{
  Complex z;
  z.real=x.real+y.real;
  z.imag=x.imag+y.imag;
  return z;
}
/* 函数 sub(complex x,complex y)返回两个复数 x 与 y 的差。形参共有两个,都为复数类型。返回值为复数类型。 */
Complex sub(Complex x,Complex y)
{
  Complex z;
  z.real=x.real-y.real;
```

```
    z.imag=x.imag-y.imag;
    return z;
}
/* 函数 multi(complex x,complex y)返回两个复数 x 与 y 的乘积。形参共有两个,都为复数类型。返回值为复数类型。 */
Complex multi(Complex x,Complex y)
{
    Complex z;
    z.real=x.real*y.real-x.imag*y.imag;
    z.imag=x.real*y.imag+x.imag*y.real;
    return z;
}
/* 函数 except(complex x,complex y)返回复数 x 除以复数 y 的结果。形参共有两个,都为复数类型。返回值为复数类型。 */
Complex except(Complex x,Complex y)
{
    Complex z;
    z.real=(x.real*y.real+x.imag*y.imag)/(y.real*y.real+y.imag*y.imag);
    z.imag=(x.real*(-y.imag)+y.real*x.imag)
           /(y.real*y.real+y.imag*y.imag);
    return z;
}
```

1.5.3 main.cpp

```
/*****************************************************************/
//main 函数实现的是对已定义的抽象数据类型 Complex 的应用。
//可以按要求输入测试数据的实部和虚部,或者直接在程序中对这几个复数进
//行赋值,再进行验证。
/*****************************************************************/
#include "complex.h"
void main()
{
    Complex a,b,c;
    float r=0,i=0;
    //输入第一个复数,并初始化复数
    printf("请输入第一个复数的实部,虚部值,用空格隔开:\n");
    scanf("%f %f",&r,&i);
    InitialComplex(a,r,i);
    //输入第二个复数,并通过设置实虚部的方式初始化复数
```