



“十二五”普通高等教育本科国家级规划教材



高等学校数据结构课程系列教材

数据结构教程 (第4版)

上机实验指导

李春葆 主编



清华大学出版社

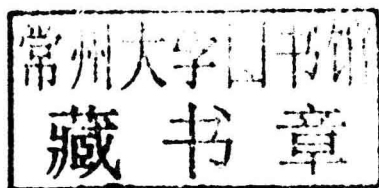
高等学校数据结构课程系列教材

数据结构教程(第4版)

上机实验指导

李春葆 主编

尹为民 蒋晶珏 喻丹丹 安 杨 编著



清华大学出版社
北京

内 容 简 介

本书是《数据结构教程(第4版)》(李春葆等编著,清华大学出版社出版)的配套上机实验指导书。两书章次一一对应,内容包括绪论、线性表、栈和队列、串、数组与稀疏矩阵、递归、树形结构、图、查找、内排序、外排序、文件和综合实验题解析。书后附录中给出了在 VC++ 6.0 环境下编写 C 程序所需要的基本知识及学生提交的实验报告格式。书中所有程序都在 VC++ 6.0 环境下调试通过,读者可以从 <http://www.tup.com.cn> 网站免费下载。书中列出了全部的上机实验题目,因此自成一体,可以脱离主教材单独使用。

本书适合高等院校计算机及相关专业本科生及研究生使用。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

数据结构教程(第4版)上机实验指导/李春葆主编. —北京:清华大学出版社,2013.1

(高等学校数据结构课程系列教材)

ISBN 978-7-302-25018-0

I. ①数… II. ①李… III. ①数据结构—高等学校—教学参考资料 IV. ①TP311.12

中国版本图书馆 CIP 数据核字(2011)第 044865 号

责任编辑:魏江江 薛 阳

封面设计:杨 兮

责任校对:李建庄

责任印制:王静怡

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

印 刷 者:北京富博印刷有限公司

装 订 者:北京市密云县京文制本装订厂

经 销:全国新华书店

开 本:185mm×260mm 印 张:18.5 字 数:465千字

版 次:2013年1月第1版 印 次:2013年1月第1次印刷

印 数:1~3000

定 价:33.00元

前言

本书是《数据结构教程(第4版)》(李春葆等编著,清华大学出版社,以下简称《教程》)的配套上机实验指导书。

全书分为13章,第1章为绪论——上机实验题解析;第2章为线性表——上机实验题解析;第3章为栈和队列——上机实验题解析;第4章为串——上机实验题解析;第5章为递归——上机实验题解析;第6章为数组和广义表——上机实验题解析;第7章为树形结构——上机实验题解析;第8章为图——上机实验题解析;第9章为查找——上机实验题解析;第10章为内排序——上机实验题解析;第11章为外排序——上机实验题解析;第12章为文件——上机实验题解析;第13章为综合实验题——上机实验题解析。各章次与《教程》的章次相对应。

另外,书后两个附录,附录A较系统地给出在VC++ 6.0环境下编写C程序所需要的基本知识。附录B为学生提交的实验报告的格式。

每个实验题设计成一个工程(如Proj1_1表示实验题1对应的第1个工程),在解答时给出了工程文件组成、工程的函数组成及其关系(程序结构图)、各函数的功能说明和实验结果。所有实验题的设计算法与主教程相关算法一一对应,并给出了较详细的注释。在实验题的设计中,采用结构化编程方法,体现了数据结构中数据组织和数据处理的思想。

书中所有程序都在VC++ 6.0环境下调试通过,读者可以从<http://www.tup.com.cn>网站免费下载。

书中列出了全部的上机实验题目,因此自成一体,可以脱离《教程》单独使用。

由于水平所限,尽管编者不遗余力,书中仍可能存在错误和不足之处,敬请教师和同学们批评指正。

编 者

2012年12月

目录

第 1 章 绪论——上机实验题 1 解析	1
实验题 1.1 求素数	1
实验题 1.2 求一个正整数的各位数字之和	2
实验题 1.3 求一个字符串是否为回文	3
第 2 章 线性表——上机实验题 2 解析	5
实验题 2.1 实现顺序表各种基本运算的算法	5
实验题 2.2 实现单链表各种基本运算的算法	9
实验题 2.3 实现双链表各种基本运算的算法	13
实验题 2.4 实现循环单链表各种基本运算的算法	18
实验题 2.5 实现循环双链表各种基本运算的算法	23
实验题 2.6 求集合(用单链表表示)的并、交和差运算	28
实验题 2.7 求两个多项式的相加运算	32
第 3 章 栈和队列——上机实验题 3 解析	36
实验题 3.1 实现顺序栈各种基本运算的算法	36
实验题 3.2 实现链栈各种基本运算的算法	39
实验题 3.3 实现环形队列各种基本运算的算法	42
实验题 3.4 实现链队各种基本运算的算法	44
实验题 3.5 求解迷宫问题的所有路径及最短路径程序	47
实验题 3.6 用栈求解皇后问题	50
实验题 3.7 病人看病模拟程序	53
实验题 3.8 停车场管理程序	56
第 4 章 串——上机实验题 4 解析	62
实验题 4.1 实现顺序串各种基本运算的算法	62
实验题 4.2 实现链串各种基本运算的算法	66

实验题 4.3	顺序串的各种模式匹配运算	72
实验题 4.4	文本串加密和解密程序	76
实验题 4.5	求一个串中出现的第一个最长重复子串	78
第 5 章	递归——上机实验题 5 解析	80
实验题 5.1	求解 n 皇后问题	80
实验题 5.2	求解背包问题	83
第 6 章	数组和广义表——上机实验题 6 解析	86
实验题 6.1	求 5×5 阶螺旋方阵	86
实验题 6.2	求一个矩阵的马鞍点	88
实验题 6.3	求两个对称矩阵之和与乘积	89
实验题 6.4	实现稀疏矩阵(采用三元组表示)的基本运算	92
实验题 6.5	实现广义表的基本运算	97
第 7 章	树形结构——上机实验题 7 解析	100
实验题 7.1	实现二叉树的各种基本运算的算法	100
实验题 7.2	实现二叉树的各种遍历算法	105
实验题 7.3	求二叉树中从根节点到叶子节点的路径	109
实验题 7.4	由遍历序列构造二叉树	112
实验题 7.5	实现中序线索化二叉树	116
实验题 7.6	构造哈夫曼树	120
实验题 7.7	用二叉树来表示代数表达式	123
第 8 章	图——上机实验题 8 解析	126
实验题 8.1	实现图的邻接矩阵和邻接表存储	126
实验题 8.2	实现图的遍历算法	131
实验题 8.3	求所有深度优先遍历序列	134
实验题 8.4	用图搜索方法求解迷宫问题	136
实验题 8.5	求有向图的简单路径	140
实验题 8.6	求无向图的深度优先生成树和广度优先生成树	143
实验题 8.7	采用普里姆算法求最小生成树	146
实验题 8.8	采用克鲁斯卡尔算法求最小生成树	148
实验题 8.9	采用狄克斯特拉算法求有向带权图的最短路径	150
实验题 8.10	采用弗洛伊德算法求有向带权图的最短路径	153
第 9 章	查找——上机实验题 9 解析	156
实验题 9.1	实现顺序查找的算法	156
实验题 9.2	实现二分查找的算法	157

实验题 9.3	实现分块查找的算法	159
实验题 9.4	实现二叉排序树的基本运算算法	161
实验题 9.5	统计一个字符串中出现的字符及其次数	166
实验题 9.6	实现二叉平衡树的相关运算算法	168
实验题 9.7	实现 B-树的相关运算算法	175
实验题 9.8	实现哈希表的相关运算算法	183
第 10 章	内排序——上机实验题 10 解析	187
实验题 10.1	实现直接插入排序算法	187
实验题 10.2	实现希尔插入排序算法	189
实验题 10.3	实现冒泡排序算法	190
实验题 10.4	实现快速排序算法	192
实验题 10.5	实现直接选择排序算法	193
实验题 10.6	实现堆排序算法	195
实验题 10.7	实现二路归并排序算法	198
实验题 10.8	实现基数排序算法	200
实验题 10.9	实现可变长度的字符串序列快速排序算法	203
实验题 10.10	实现英文单词按字典序排列的基数排序算法	205
第 11 章	外排序——上机实验题 11 解析	208
实验题 11.1	实现置换-选择算法	208
实验题 11.2	实现多路归并算法	212
第 12 章	文件——上机实验题 12 解析	216
实验题 12.1	实现学生记录文件的基本操作	216
实验题 12.2	实现索引文件建立和查找的算法	221
第 13 章	综合实验题解析	227
综合实验题 1	链表综合算法设计	227
综合实验题 2	求复杂表达式的值	233
综合实验题 3	用二叉树实现家谱的相关运算	244
综合实验题 4	求无向图中满足约束条件的路径	252
综合实验题 5	分析二分查找成功时的平均查找长度	255
综合实验题 6	求各种排序算法的执行时间	258
附录 A	使用 VC++ 6.0 系统	267
附录 B	实验报告格式	286

实验题 1.1 求素数

设计一个程序 `expl-1.cpp`, 输出所有小于等于 n (n 为一个大于 2 的正整数) 的素数。要求: (1) 每行输出 10 个素数; (2) 尽可能采用较优的算法。

解: 本工程 `Proj1_1` 的组成结构如图 1.1 所示。本程序的模块结构图如图 1.2 所示。图中方框表示函数, 方框中指出函数名, 箭头方向表示函数间的调用关系, 虚线方框表示文件的组成, 即指出该虚线方框中的函数存放在哪个文件中。

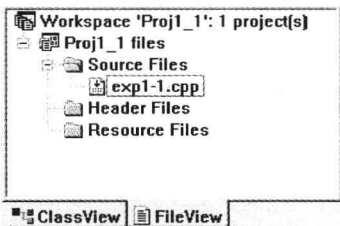


图 1.1 Proj1_1 工程组成

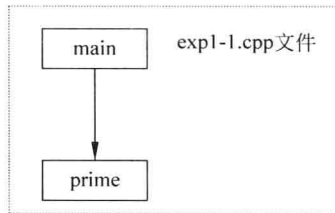


图 1.2 Proj1_1 工程的程序结构图

`expl-1.cpp` 文件包含的函数 `prime(n)` 的功能是判断正整数 n 是否为素数。采用的方法是: 若 n 是素数, 则 n 不能被 $2 \sim \sqrt{n}$ 的任何整数整除。

对应的程序如下 (设计思路详见代码中的注释):

```
//文件名: expl-1.cpp
#include <stdio.h>
#include <math.h>
bool prime(int n)                                //判断正整数 n 是否为素数
{
    int i;
    for (i = 2; i <= (int)sqrt(n); i++)
        if (n % i == 0)
            return false;                        //若 n 不是素数, 则退出并返回 false
}
```



```

        return true;
    }
}
void main()
{
    int n, i, j = 0; //j 用于累计素数个数
    printf("n:");
    scanf("%d", &n);
    printf("小于等于 %d 的素数:\n", n);
    if (n > 2)
    {
        printf("%4d", 2);
        j++;
    }
    for (i = 3; i <= n; i += 2)
        if (prime(i))
        {
            printf("%4d", i);
            if (j != 0 && ++j % 10 == 0) //每行最多显示 10 个素数
                printf("\n");
        }
    printf("\n");
}

```

连编本工程生成可执行文件 Proj1_1.exe。程序的一次执行结果如下：

```

n:100
小于等于 100 的素数:
 2     3     5     7    11    13    17    19    23    29
31    37    41    43    47    53    59    61    67    71
73    79    83    89    97

```

对于 prime(n), 其时间复杂度为 $O(\sqrt{n})$, 由于偶数不可能是素数, 所以程序中只对奇数进行素数的判断。因此, 上述程序的时间复杂度较低。

实验题 1.2 求一个正整数的各位数字之和

编写一个程序 exp1-2.cpp, 计算任一输入的正整数的各位数字之和, 并分析算法的时间复杂度。

解: 本工程 Proj1_2 的组成结构如图 1.3 所示。本程序的模块结构图如图 1.4 所示, 图中方框表示函数, 方框中指出函数名, 箭头方向表示函数间的调用关系, 虚线方框表示文件的组成, 即指出该虚线方框中的函数存放在哪个文件中。

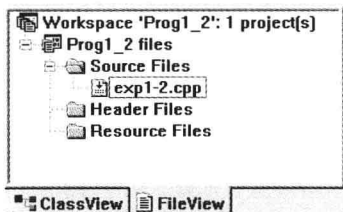


图 1.3 Proj1_2 工程组成

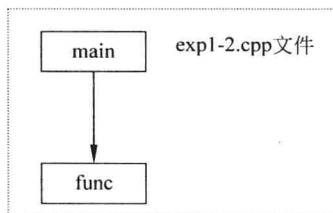


图 1.4 Proj1_2 工程的程序结构图

exp1-2.cpp 文件包含的函数 func(int num)的功能是分解 num 的各位数字,返回这些数字之和。采用的方法是:对 num 边分解边进行累加,直到分解完毕。

对应的程序如下(设计思路详见代码中的注释):

```
//文件名: exp1-2.cpp
#include <stdio.h>
int func(int num)
{
    int s = 0;
    do
    {
        s += num % 10;
        num /= 10;
    } while(num);
    return (s);
}
void main()
{
    int n;
    printf("输入一个整数:");
    scanf("%d", &n);
    printf("各位数字之和: %d\n", func(n));
    printf("\n");
}
```

连编本工程生成可执行文件 Proj1_2.exe。程序的一次执行结果如下:

```
输入一个整数:12345678 ↵
各位数字之和:36
```

func(n)的时间复杂度为 $O(\text{len}(n))$, len(n)为正整数 n 的位数。程序的时间复杂度也为 $O(\text{len}(n))$ 。

实验题 1.3 求一个字符串是否为回文

编写一个程序 exp1-3.cpp,判断一个字符串是否为“回文”(顺读和倒读都一样的字符串称为“回文”),并分析算法的时间复杂度。

解:本工程 Proj1_3 的组成结构如图 1.5 所示。本程序的模块结构图如图 1.6 所示。图中方框表示函数,方框中指出函数名,箭头方向表示函数间的调用关系,虚线方框表示文件的组成,即指出该虚线方框中的函数存放在哪个文件中。

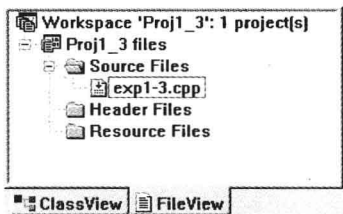


图 1.5 Proj1_3 工程组成

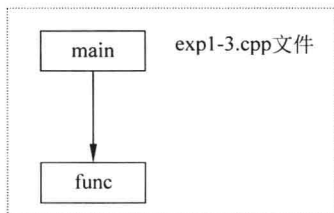


图 1.6 Proj1_3 工程的程序结构图

expl-3.cpp 文件包含的函数 `func(char s[])` 的功能是判断串 `s` 是否为回文。采用的方法是: 用 `flag` 表示是否为回文, 其初值为 `true`。用 `i` 从左向右扫描字符串 `s`, 用 `j` 从右向左扫描字符串 `s`, 若 `s[i]` 与 `s[j]` 不相等, 则 `flag=false` (表示不是回文) 并退出循环, 否则, 继续比较直到 `i<j` 不成立。

对应的程序如下(设计思路详见代码中的注释):

```
//文件名: expl-3.cpp
#include <stdio.h>
#include <string.h>
#define MAX 100 //字符串的最大长度
bool func(char s[])
{
    bool flag = true;
    int i, j, slen = strlen(s); //slen 为字符串 s 的长度
    for (i = 0, j = slen - 1; i < j; i++, j--)
        if (s[i] != s[j])
        {
            flag = false;
            break;
        }
    return (flag);
}
void main()
{
    char s[MAX];
    printf("输入一字符串:");
    scanf("%s", s);
    if (func(s))
        printf("%s 字符串是回文\n", s);
    else
        printf("%s 字符串不是回文\n", s);
}
```

连编本工程生成可执行文件 `Proj1_3.exe`。程序的一次执行结果如下:

```
输入一字符串: abcba ↵
abcba 字符串是回文
```

在 `func(s)` 算法中, `for` 循环语句的执行次数为 $n/2$ (n 为字符串 `s` 的长度), 则它的时间复杂度为 $O(n)$ 。程序的时间复杂度也为 $O(n)$ 。

线性表——上机实验题 2 解析 第 2 章

实验题 2.1 实现顺序表各种基本运算的算法

编写一个程序 algo2-1.cpp, 实现顺序表的各种基本运算(假设顺序表的元素类型为 char), 并在此基础上设计一个主程序完成如下功能:

- (1) 初始化顺序表 L;
- (2) 依次采用尾插法插入 a, b, c, d, e 元素;
- (3) 输出顺序表 L;
- (4) 输出顺序表 L 的长度;
- (5) 判断顺序表 L 是否为空;
- (6) 输出顺序表 L 的第 3 个元素;
- (7) 输出元素 a 的位置;
- (8) 在第 4 个元素位置上插入 f 元素;
- (9) 输出顺序表 L;
- (10) 删除 L 的第 3 个元素;
- (11) 输出顺序表 L;
- (12) 释放顺序表 L。

解: 本工程 Proj2_1 的组成结构如图 2.1 所示。本程序的模块结构图如图 2.2 所示。图中方框表示函数, 方框中指出函数名, 箭头方向表示函数间的调用关系, 虚线方框表示文件的组成, 即指出该虚线方框中的函数存放在哪个文件中。

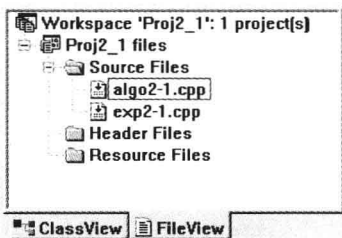


图 2.1 Proj2_1 工程组成

根据《教程》中 2.2 节的算法得到 algo2-1.cpp 程序, 其中包含如下函数:

- InitList(Sqlist * &L): 初始化顺序表 L。
- DestroyList(Sqlist * L): 释放顺序表 L。
- ListEmpty(Sqlist * L): 判断顺序表 L 是否为空表。

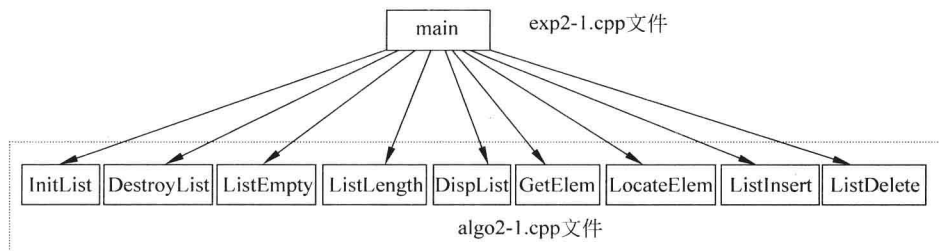


图 2.2 Proj2_1 工程的程序结构图

- ListLength(SqList * L): 返回顺序表 L 的元素个数。
 - DispList(SqList * L): 输出顺序表 L。
 - GetElem(SqList * L, int i, ElemType &e): 获取顺序表 L 中第 i 个元素。
 - LocateElem(SqList * L, ElemType e): 在顺序表 L 中查找元素 e。
 - ListInsert(SqList * &L, int i, ElemType e): 在顺序表 L 中第 i 个位置上插入元素 e。
 - ListDelete(SqList * &L, int i, ElemType &e): 在顺序表 L 中删除第 i 个元素。
- 对应的程序如下(设计思路详见代码中的注释):

```

//文件名:algo2-1.cpp */
#include <stdio.h>
#include <malloc.h>
#define MaxSize 50
typedef char ElemType;
typedef struct
{
    ElemType data[MaxSize];
    int length;
} SqList;
void InitList(SqList * &L) //初始化线性表
{
    L = (SqList *)malloc(sizeof(SqList)); //分配存放线性表的空间
    L->length = 0; //置空线性表长度为 0
}
void DestroyList(SqList * L) //销毁线性表
{
    free(L);
}
bool ListEmpty(SqList * L) //判断线性表是否为空表
{
    return (L->length == 0);
}
int ListLength(SqList * L) //求线性表的长度
{
    return (L->length);
}
void DispList(SqList * L) //输出线性表
{
    int i;
    if (ListEmpty(L)) return;
  
```

```

    for (i = 0; i < L->length; i++)
        printf(" %c ", L->data[i]);
    printf("\n");
}
bool GetElem(SqList * L, int i, ElemType &e)           //求线性表中某个数据元素值
{
    if (i < 1 || i > L->length)                       //参数错误时返回 false
        return false;
    e = L->data[i - 1];                               //取元素值
    return true;                                     //成功找到元素时返回 true
}
int LocateElem(SqList * L, ElemType e)              //按元素值查找
{
    int i = 0;
    while (i < L->length && L->data[i] != e)          //查找元素 e
        i++;
    if (i >= L->length)                               //未找到时返回 0
        return 0;
    else
        return i + 1;                                //找到后返回其逻辑序号
}
bool ListInsert(SqList * &L, int i, ElemType e)      //插入数据元素
{
    int j;
    if (i < 1 || i > L->length + 1)                  //参数错误时返回 false
        return false;
    i--;
    for (j = L->length; j > i; j--)                   //将顺序表逻辑序号转化为物理序号
        L->data[j] = L->data[j - 1];                 //将 data[i] 及后面元素后移一个位置
    L->data[i] = e;                                   //插入元素 e
    L->length++;                                     //顺序表长度增 1
    return true;                                     //成功插入返回 true
}
bool ListDelete(SqList * &L, int i, ElemType &e)    //删除数据元素
{
    int j;
    if (i < 1 || i > L->length)                       //参数错误时返回 false
        return false;
    i--;
    e = L->data[i];
    for (j = i; j < L->length - 1; j++)               //将 data[i] 之后的元素前移一个位置
        L->data[j] = L->data[j + 1];
    L->length--;                                     //顺序表长度减 1
    return true;                                     //成功删除返回 true
}

```

设计如下 exp2-1.cpp 主程序:

```

//文件名:exp2-1.cpp
#include <stdio.h>
#include <malloc.h>
#define MaxSize 50
typedef char ElemType;
typedef struct
{
    ElemType data[MaxSize];

```

```

    int length;
} SqList;
extern void InitList(SqList * &L);
extern void DestroyList(SqList * L);
extern bool ListEmpty(SqList * L);
extern int ListLength(SqList * L);
extern void DispList(SqList * L);
extern bool GetElem(SqList * L, int i, ElemType &e);
extern int LocateElem(SqList * L, ElemType e);
extern bool ListInsert(SqList * &L, int i, ElemType e);
extern bool ListDelete(SqList * &L, int i, ElemType &e);
void main()
{
    SqList * L;
    ElemType e;
    printf("顺序表的基本运算如下:\n");
    printf("  (1)初始化顺序表 L\n");
    InitList(L);
    printf("  (2)依次采用尾插法插入 a,b,c,d,e 元素\n");
    ListInsert(L, 1, 'a');
    ListInsert(L, 2, 'b');
    ListInsert(L, 3, 'c');
    ListInsert(L, 4, 'd');
    ListInsert(L, 5, 'e');
    printf("  (3)输出顺序表 L:");
    DispList(L);
    printf("  (4)顺序表 L 的长度 = %d\n", ListLength(L));
    printf("  (5)顺序表 L 为 %s\n", (ListEmpty(L)?"空":"非空"));
    GetElem(L, 3, e);
    printf("  (6)顺序表 L 的第 3 个元素 = %c\n", e);
    printf("  (7)元素 a 的位置 = %d\n", LocateElem(L, 'a'));
    printf("  (8)在第 4 个元素位置上插入 f 元素\n");
    ListInsert(L, 4, 'f');
    printf("  (9)输出顺序表 L:");
    DispList(L);
    printf("  (10)删除 L 的第 3 个元素\n");
    ListDelete(L, 3, e);
    printf("  (11)输出顺序表 L:");
    DispList(L);
    printf("  (12)释放顺序表 L\n");
    DestroyList(L);
}

```

连编本工程生成可执行文件 Proj2_1.exe。程序执行结果如下:

顺序表的基本运算如下:

- (1)初始化顺序表 L
- (2)依次采用尾插法插入 a,b,c,d,e 元素
- (3)输出顺序表 L:a b c d e
- (4)顺序表 L 的长度 = 5
- (5)顺序表 L 为非空
- (6)顺序表 L 的第 3 个元素 = c

- (7)元素 a 的位置 = 1
- (8)在第 4 个元素位置上插入 f 元素
- (9)输出顺序表 L:a b c f d e
- (10)删除 L 的第 3 个元素
- (11)输出顺序表 L:a b f d e
- (12)释放顺序表 L

实验题 2.2 实现单链表各种基本运算的算法

编写一个程序 algo2-2. cpp,实现单链表的各种基本运算(假设单链表的元素类型为 char),并在此基础上设计一个主程序 exp2-2. cpp 完成如下功能:

- (1) 初始化单链表 h;
- (2) 依次采用尾插法插入 a,b,c,d,e 元素;
- (3) 输出单链表 h;
- (4) 输出单链表 h 的长度;
- (5) 判断单链表 h 是否为空;
- (6) 输出单链表 h 的第 3 个元素;
- (7) 输出元素 a 的位置;
- (8) 在第 4 个元素位置上插入 f 元素;
- (9) 输出单链表 h;
- (10) 删除 L 的第 3 个元素;
- (11) 输出单链表 h;
- (12) 释放单链表 h。

解:本工程 Proj2_2 的组成结构如图 2.3 所示。本程序的模块结构图如图 2.4 所示。图中方框表示函数,方框中指出函数名,箭头方向表示函数间的调用关系,虚线方框表示文件的组成,即指出该虚线方框中的函数存放在哪个文件中。

根据《教程》中 2.3.2 节的算法得到 algo2-2. cpp 程序,其中包含如下函数:

- InitList(LinkList * &L): 初始化单链表 L。



图 2.3 Proj2_2 工程组成

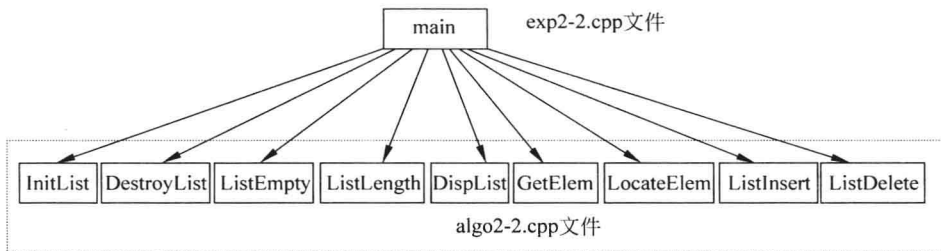


图 2.4 Proj2_2 工程的程序结构图

- DestroyList(LinkList * L): 释放单链表 L。
- ListEmpty(LinkList * L): 判断单链表 L 是否为空表。
- ListLength(LinkList * L): 返回单链表 L 的元素个数。
- DispList(LinkList * L): 输出单链表 L。
- GetElem(LinkList * L, int i, ElemType &e): 获取单链表 L 中第 i 个元素。
- LocateElem(LinkList * L, ElemType e): 在单链表 L 中查找元素 e。
- ListInsert(LinkList * &L, int i, ElemType e): 在单链表 L 中第 i 个位置上插入元素 e。
- ListDelete(LinkList * &L, int i, ElemType &e): 在单链表 L 中删除第 i 个元素。

对应的程序如下(设计思路详见代码中的注释):

```
//文件名:algo2-2.cpp
#include <stdio.h>
#include <malloc.h>
typedef char ElemType;
typedef struct LNode //定义单链表节点类型
{
    ElemType data;
    struct LNode *next;
} LinkList;
void InitList(LinkList * &L) //初始化线性表
{
    L = (LinkList *)malloc(sizeof(LinkList)); //创建头节点
    L->next = NULL;
}
void DestroyList(LinkList * &L) //销毁线性表
{
    LinkList *p = L, *q = p->next;
    while (q != NULL)
    {
        free(p);
        p = q;
        q = p->next;
    }
    free(p);
}
bool ListEmpty(LinkList *L) //判断线性表是否为空表
{
    return (L->next == NULL);
}
int ListLength(LinkList *L) //求线性表的长度
{
    LinkList *p = L; int i = 0;
    while (p->next != NULL)
    {
        i++;
        p = p->next;
    }
    return (i);
}
void DispList(LinkList *L) //输出线性表
{
    LinkList *p = L->next;
    while (p != NULL)
    {
        printf("%c ", p->data);
    }
}
```