第 13 章 综合程序设计

13.1 教学要点

本章主要通过几个综合性程序的设计编程，培养和锻炼学生对具有一定规模和复杂度的问题的分析与求解能力，能综合应用各种数据类型实现较复杂数据的存储，掌握程序设计的综合方法和技能，同时培养良好的程序设计风格与代码规范意识。

13.2 实验指导教材参考答案

1．自动售货机

如图 13.1 所示的简易自动售货机，物品架 1、2 上共有 10 样商品，按顺序进行编号分别为 1－10，标有价格与名称，一个编号对应一个可操作按钮，供选择商品使用。如果物架上的商品被用户买走，储物柜中会自动取出商品送到物架上，保证物品架上一定会有商品。用户可以一次投入较多钱币，并可以选择多样商品，售货机可以一次性将商品输出并找零钱。

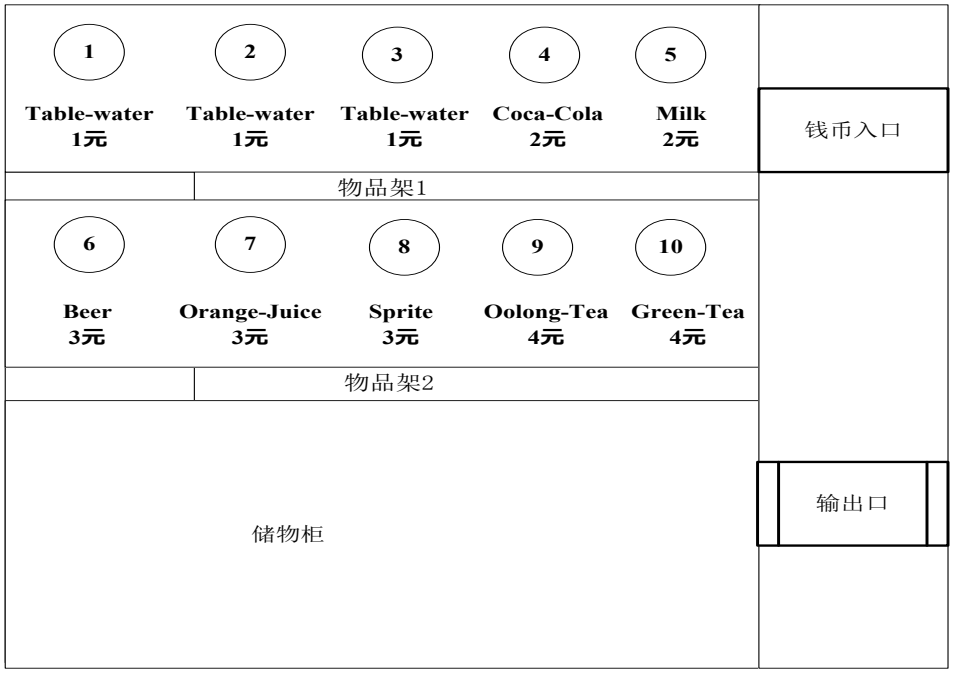

图 13.1 自动售货机示意图

用户购买商品的操作方法是：

（1）从“钱币入口”放入钱币，依次放入多个硬币或纸币。钱币可支持 1 元（纸币、硬币）、2 元（纸币）、5 元（纸币）、10 元（纸币），放入钱币时，控制器会先对钱币进行检验识别出币值，并统计币值总额，显示在控制器显示屏中，提示用户确认钱币放入完毕；

（2）用户确认钱币放入完毕，便可选择商品，只要用手指按对应商品外面的编号按钮即可。每选中一样商品，售货机控制器会判断钱币是否足够购买，如果钱币足够，自动根据编号将物品进行计数和计算所需钱币值，如果钱币不足，则结束购物。

请为自动售货机编程，输入钱币值序列，以-1 作为结束，依次输入多个购买商品编号，若编号超出范围或余额不够则输入结束，输出钱币总额与找回零钱，以及所购买商品名称及数量。

解答：

```
#include "stdio.h"
int main()
{
    char  *productname[10]={"Table-water",  "Table-water",  "Table-water",  "Coca-Cola",
 "Milk","Beer", "Orange-Juice", "Sprite", "Oolong-Tea", "Green-Tea"};/*商品名称*/
    int price[10]={1,1,1,2,2,3,3,3,4,4}; /*价格*/
    static int countp[10];
    int tm=0,xm,t,*p=price,i;
    int x;

    /*钱币序列求和*/
    scanf("%d",&x);
    while(x>0)
    {
        tm+=x;
        scanf("%d",&x); /*读入钱币*/
    }

    xm=tm;
    scanf("%d",&t); /*读入商品编号*/
    while((t>=1 && t<=10) && tm>*(p+t-1))
    {
        tm-=*(p+t-1);
        countp[t-1]++; /*统计编号为 t 的物品购买的数量*/
        scanf("%d",&t); /*读入商品编号*/
    }

    /*输出总金额，应找回的金额*/
    printf("Total:%dyuan,change:%dyuan\n",xm,tm);

    /*输出物品名称与数量*/
    for(i=0;i<10;i++)
    {
        if(countp[i]>0)
            printf("%s:%d;",productname[i],countp[i]);
    }
    printf("\n");
    return 0;
}
```

2．自动寄存柜

某超市门口的自动寄存柜有 n 个寄存箱，并且有一个投币控制器，顾客想要寄存小件物品时，只要在投币控制器投入 1 个 1 元的硬币，如果此时有空闲的箱子，寄存柜就会自动打

开一个空的箱子，并且打印输出一张小小的密码纸条；如果没有空闲的箱子，则提示“本柜已满”。当顾客离开超市时，用密码纸条上指定的数字密码依次输入到开箱控制器，则顾客所存包的箱子门就自动打开，顾客取走物品后，关上门。

输入数据时，可先输入寄存箱总数 n，再由用户选择是“投硬币”还是“输密码”。

如果选择“投硬币”，则只有硬币值是 1 才开箱。如果有空闲的箱子，则输出箱子编号及密码（4 位数字）；如果无空闲的箱子，则提示：“本柜已满”。

如果选择“输密码”，若输入的密码与某一箱子密码相符，则显示打开的箱子编号，否则输出提示：“密码错误”。

请编写开箱控制程序实现上述过程。

解答：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

/*寄存柜的最大数量*/
#define MAX_LOCKER_COUNT 100

/*寄存柜结构*/
typedef struct {
    int    used;            /*是否被使用了，非 0 表示被使用了*/
    char password[5]; /*密码*/
}Locker;

Locker Lockers[MAX_LOCKER_COUNT]; /*寄存柜数组*/
int LockerCount; /*寄存柜个数*/


/*查找密码所对应的柜子，找到返回相应的数组下标，未找到返回-1*/
int FindByPassword( char pwd[5] ) {
    int i;

    for( i=0 ; i<LockerCount ; i++ ) {
        /*若该柜子没使用则忽略*/
        if( !Lockers[i].used ) {
            continue;
        }

        /*若找到，返回 i*/
        if( strcmp( pwd, Lockers[i].password ) == 0 ) {
            return i;
        }
    }
```

```
    return -1;
}

/*产生一个与已有密码不同的密码*/
void GeneratePassword( char pwd[ ] ) {
    do {
        /*置随机数种子*/
        srand( (int)time(0) );

        /*产生 4 个 1~9 的随机数*/
        for( int i=0 ; i<4 ; i++ ) {
            pwd[i] = rand()%9 + '1';
        }
        pwd[i] = '\0';
    } while( FindByPassword( pwd ) != -1 );
}

/*执行投入硬币动作*/
void DropCoin() {
    int    i, coin;

    // 输入密码
    printf( "投币值:" );
    scanf( "%d", &coin );
    if( coin != 1 ) {
        puts( "请投入 1 个 1 元的硬币" );
        return;
    }

    /*查找一个空的寄存柜*/
    for( i=0 ; i<LockerCount ; i++ ) {
        if( !Lockers[i].used ) break;
    }
    if( i>=LockerCount ) {
        puts( "本柜已满" );
        return;
    }

    /*产生密码，并将使用标志置为 1*/
    GeneratePassword( Lockers[i].password );
    Lockers[i].used = 1;

    /*输出信息*/
    printf( "寄存箱编号:%d  密码:%s\n", i+1, Lockers[i].password );
```

```
}

/*执行输入密码动作*/
void InputPassword() {
	char pwd[5];

	/*输入密码*/
	printf( "输入密码:" );
	scanf( "%s", pwd );

	int i = FindByPassword( pwd );
	if( i != -1 ) {
		/*打开柜子，已使用标志置为 0*/
		Lockers[i].used = 0;

		/*输出信息*/
		printf( "%d 号寄存箱已打开\n", i+1 );
	} else {
		/*密码错误*/
		printf( "密码错误\n" );
	}
}

int main() {
	int i;

	/*输入寄存箱总数*/
	printf( "寄存箱总数:" );
	scanf( "%d", &LockerCount );

	/*初始化*/
	for( i=0 ; i<LockerCount ; i++ ) {
		Lockers[i].used = 0;
	}

	while(1) {
		printf( "1.投硬币  2.输密码  0.退出  请选择: " );
		scanf( "%d", &i );
		if( i==1 ) {
			DropCoin();
		} else if( i==2 ) {
			InputPassword();
		} else if( i==0 ) {
			puts( "结束" );
```

```
                break;
            }
        }

        return 0;
    }
```

3．停车场管理

设有一个可以停放 n 辆汽车的狭长停车场，它只有一个大门可以供车辆进出。车辆按到达停车场时间的先后次序依次从停车场最里面向大门口处停放（即最先到达的第一辆车停放在停车场的最里面）。如果停车场已放满 n 辆车，则以后到达的车辆只能在停车场大门外的便道上等待，一旦停车场内有车开走，则排在便道上的第一辆车可以进入停车场。停车场内如有某辆车要开走，则在它之后进入停车场的车都必须先退出停车场为它让路，待其开出停车场后，这些车辆再依原来的次序进场。每辆车在离开停车场时，都应根据它在停车场内停留的时间长短交费，停留在便道上的车不收停车费。编写程序对该停车场进行管理。

输入数据时，先输入一个整数 n(n<=10)，再输入若干组数据，每组数据包括三个数据项：汽车到达或离开的信息（A 表示到达、D 表示离开、E 表示结束）、汽车号码、汽车到达或离开的时刻。当输入“E 0 0”时程序结束。

若有车辆到达，则输出该汽车的停车位置；若有车辆离开，则输出该汽车在停车场内停留的时间。

解答：

```
#include <stdio.h>
/*停车场容量和便道容量*/
#define PARKING_CAPACITY 100
#define ROAD_CAPACITY       100
int ParkingCapacity;
/*车辆结构*/
typedef struct {
    int id;         /*车辆 ID*/
    int inTime; /*进入时间*/
}ParkingCar;
/*停车场内的车及数量*/
ParkingCar Parking[PARKING_CAPACITY];
int ParkingCount;
/*便道内的车及数量*/
int Road[ROAD_CAPACITY];
int RoadCount;

/*初始化系统*/
void Init() {
    ParkingCount = 0;
    RoadCount = 0;
}
```

```
/*车到达*/
void Arrive( int id, int time ) {
    if( ParkingCount < ParkingCapacity ) {
        /*停车场未停满，停入停车场*/
        Parking[ParkingCount].id     = id;
        Parking[ParkingCount].inTime = time;
        ParkingCount ++;
        printf( "%d 号车停入%d 号位\n", id, ParkingCount );
    } else if( RoadCount < ROAD_CAPACITY ) {   /*否则停入便道*/
        Road[RoadCount++] = id;
        printf( "%d 号车在便道上等待\n", id );
    } else {
        puts( "停车场合便道都已满！" );
    }
}
/*检查停车场是否满,若有空位就将便道中的车停入*/
void CheckAndPark( int time ) {
    while( ParkingCount < ParkingCapacity && RoadCount > 0 ) {
        /*获取过道的第一辆车*/
        int id = Road[0];

        /*后面的车都向前移*/
        for( int i=1 ; i<RoadCount ; i++ ) {
            Road[i-1] = Road[i];
        }
        RoadCount --;

        /*过道的第一辆车进入停车场*/
        Arrive( id, time );
    }
}
/*车离开*/
void Leave( int id, int leaveTime ) {
    int i, j, k;
    /*查找 id 的车在停车场中的位置*/
    for( i=0 ; i<ParkingCount ; i++ ) {
        if( Parking[i].id == id ) break;
    }
    if( i>=ParkingCount ) {
        printf( "该车未停在停车场\n" );
        return;
    }
    /*在它之后进入停车场的车先退出,待其开出停车场后再依次进场*/
    ParkingCar temp[PARKING_CAPACITY];
```

```
    int count = 0;
    /* 让道*/
    for( j=ParkingCount-1 ; j>i ; j-- ) {
        temp[count++] = Parking[j];
    }

    /* 离开，输出时间信息*/
    int time = leaveTime - Parking[i].inTime;
    printf( "%d 号车出停车场，停留时间%d\n", id, time );
    ParkingCount --;

    /*让道的车重新进入停车场*/
    j = i;
    for( k=count-1 ; k>=0 ; k-- ) {
        Parking[j++] = temp[k];
    }
    /*检查停车场是否满，若有空位就将便道中的车停入*/
    CheckAndPark( leaveTime );
}

int main() {
    /*输入停车场容量*/
    scanf( "%d", &ParkingCapacity );
    /*初始化系统*/
    Init();

    while( 1 ) {
        char cmd[100];
        int id, time;

        scanf( "%s%d%d", cmd, &id, &time );
        if( cmd[0] == 'A' ) {
            Arrive( id, time );
        } else if( cmd[0] == 'D' ) {
            Leave( id, time );
        } else if( cmd[0] == 'E' ) {
            break;
        }
    }
    return 0;
}
```

4．值班安排

医院有 A、B、C、D、E、F、G 7 位大夫，在一星期内（星期一至星期天）每人要轮流值

班一天，如果已知：

（1）A 大夫比 C 大夫晚 1 天值班；

（2）D 大夫比 E 大夫晚 1 天值班；

（3）E 大夫比 B 大夫早 2 天值班

（4）B 大夫比 G 大夫早 4 天值班；

（5）F 大夫比 B 大夫晚 1 天值班；

（6）F 大夫比 C 大夫早 1 天值班；

（7）F 大夫星期四值班。

就可以确定周一至周日的值班人员分别为：E、D、B、F、C、A、G。

编写程序，根据输入的条件，输出星期一至星期天的值班人员。

输入数据时，先输入一个整数 n，再输入 n 组条件，要求能够根据输入的条件确定唯一的值班表，且输入的 n 组条件中能够直接或间接得到任意两位大夫的关联关系，例如上面的条件(2)直接显示了 D 与 E 间的关系，而通过条件（1)、(6)、(5）可以间接得到 A 与 B 的关系。

条件的输入格式有 2 种：

格式 1：编号 比较运算符 编号 天数

其中比较运算符有 2 种：> 或 < ，分别表示“早”或“晚”

例如：A<C1　表示：A 大夫比 C 大夫晚 1 天值班

格式 2：编号 = 数值

例如：F=4　表示：F 大夫在星期四值班

解答：

解题思路：以某一输入项为出发点，先确定（假设）一个医生的值班日期，再从输入项条件中查找与此医生相关的条件，确定下一个医生的值班日期，直至所有条件都使用一遍，从而确定所有医生的值班顺序。

```
#include <stdio.h>
#include<stdlib.h>
int main(void)
{    int i,j,n,m,start=8;
     char a[20][80],ch1,ch2;
     struct person{
        char ch;
        int day;
     };
     struct person s[7],tmp; /*结构数组 s 用于存放 7 个医生排班日期*/

     for(i=0;i<7;i++)   /*初始化 s*/
     { s[i].ch='A'+i;s[i].day=0; }
     scanf ("%d", &n);

     getchar();
     for(i=0;i<n;i++)   /*输入条件至 a*/
         gets(a[i]);

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

```
    if(a[i][1]=='=') break;
if(i<n)
     s[a[i][0]-'A'].day=a[i][2]-'0';   /*如果输入中有确定日期的，先对其赋值*/
else
     s[a[0][0]-'A'].day=start;   /*如果无确定日期输入，假设第一个人的日期为 8*/

m=n;
while(m){
     for(i=0;i<n;i++){   /*在输入项中找一个与 s 中已设日期医生有关系的条件*/
        if(a[i][0]=='\0') continue;

        if(a[i][1]=='=') {ch1=a[i][0];ch2='\0';}
        else   { ch1=a[i][0]; ch2=a[i][2];   }
        for(j=0;j<7;j++)
             if(s[j].day!=0&&(s[j].ch==ch1||s[j].ch==ch2))   break;
        if(j<7) break;
    }
     if(i==n){printf("ERROR!\n");   exit(0);} /*找不到此输入项，则输入错误*/

     if(a[i][1]=='='){   /*输入项为格式 2 的情况*/
           s[j].day=a[i][2]-'0';
    }
    else /*输入项为格式 1 的情况*/
        {
           if(s[j].ch==ch1)
           {   if(a[i][1]=='>')
                  s[a[i][2]-'A'].day=s[j].day+(a[i][3]-'0');
               else
                  s[a[i][2]-'A'].day=s[j].day-(a[i][3]-'0');
           }
           else
           {   if(a[i][1]=='>')
                  s[a[i][0]-'A'].day=s[j].day-(a[i][3]-'0');
               else
                  s[a[i][0]-'A'].day=s[j].day+(a[i][3]-'0');
           }
        }
     m--;   /*又确定了一个医生的值班日期*/
     a[i][0]='\0';   /*该输入项已使用过*/
}
/*按日期先后排序*/
for(i=1;i<=6;i++)
     for(j=0;j<7-i;j++)
        if(s[j].day>s[j+1].day)
```

```
                {   tmp=s[j]; s[j]=s[j+1]; s[j+1]=tmp; }
        /*输出值班情况*/
        for(i=0;i<7;i++)
            printf("%c",s[i].ch);
        putchar('\n');
        return 0;
    }
```

5．学生成绩管理

设计一个菜单驱动的学生成绩管理程序，管理 n 个学生的 m 门考试科目成绩，实现以下基本功能：

（1）能够新增学生信息，并计算总分和平均分；

（2）能够根据学号修改和删除某学生信息；

（3）能够显示所有学生的成绩信息；

（4）能够分别按总分和学号进行排序；

（5）能够根据学号查询该学生的基本信息；

（6）学生成绩数据最终保存在文件中，能够对文件读、写学生数据。

程序运行时，菜单形式如下：

```
Management for Students' scores
1. Append record
2. List record
3. Delete record
4. Modify record
5. Search record
6. Sort in descending order by sum
7. Sort in ascending order by sum
8. Sort in descending order by num
9. Sort in ascending order by num
W. Write to a File
R. Read from a File
0. Exit
Please Input your choice:
```

要求用模块化方式组织程序结构，合理设计各自定义函数。同时，程序能够进行异常处理，检查用户输入数据的有效性，在用户输入数据有错误（如类型错误）或无效时，不会中断程序的执行，程序具有一定的健壮性。

解答：

说明：

（1）学生信息包括：num，course1，course2，course3，sum；

（2）每个功能自定义一个函数；

（3）菜单功能 1~9 是对内存中的学生信息进行操作，初始时为空数据；

（4）功能 w 是将当前内存中的数据写入一个指定的文本文件；

（5）功能 R 是从指定的一个文件中读入当前内存，并覆盖当前数据；如果当前内存中的数据未保存，需要提示用户。

```
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#define DATA_FILE_NAME "students.txt"            /*保存信息的文件名*/
#define DATA_FILE_HEAD "STUDENT_MANAGEMENT" /*文件的文件头信息，用来判断是否是该系统生成的文件*/
/*学生信息结构*/
typedef struct Student_str {
    char     num[8];   /*学号*/
    double course1; /*科目 1 分数*/
    double course2; /*科目 2 分数*/
    double course3; /*科目 3 分数*/
    double sum;        /*总分*/
}Student;
/*链表结构*/
typedef struct Node_str {
    Student student;
    struct Node_str *next;
}Node;
/*系统的全局变量*/
Node *listHead, *listTail; /*链表的头节点和尾节点*/
int modified; /*当前文件是否被修改标志*/
/*初始化链表*/
void List_Init() {
    listHead = listTail = NULL;
}
/*清空链表*/
void List_Clear() {
    Node *p = listHead;
    while( p != NULL ) {
        Node *next = p->next;
        free( p );
        p = next;
    }
    List_Init();
}
/*添加一个新的节点到链表的最后*/
void List_Append( Student student ) {
    Node *p = (Node*)malloc( sizeof(Node) );

    p->student = student;
    p->next       = NULL;
```

```
    if( listHead == NULL ) {
        listHead = listTail = p;
    } else {
        listTail->next = p;
        listTail = p;
    }
}
/*在链表中删除一个元素(学号为 num 的元素)*/
int List_Delete( char num[ ] ) {
    /*在列表中查找*/
    Node *pre = NULL, *cur = listHead;
    while( cur != NULL ) {
        if( strcmp( cur->student.num, num ) == 0 ) {
            break;
        }
        pre = cur;
        cur = cur->next;
    }
    /*如果找到，则删除*/
    if( cur != NULL ) {
        if( pre == NULL ) { /*删除表头*/
            listHead    = cur->next;
        } else { /*删除的非表头*/
            pre->next = cur->next;
        }
        /*删除的是表尾*/
        if( cur->next == NULL ) {
            listTail = pre;
        }
        free( cur );    /*回收内存*/
        return 1;
    } else {
        return 0;
    }
}
/*在链表中查找学号为 num 的元素，找到返回数据指针，找不到则返回 NULL*/
Student* List_Search( char num[8] ) {
    Node *p = listHead;
    while( p != NULL ) {
        if( strcmp( num, p->student.num ) == 0 ) {
            return &p->student;
        }
        p = p->next;
    }
```

```
    return NULL;
}
/*读入一个单词，取第一个字符，在用户输入是防止空行给输入带来的影响*/
char getFirstChar() {
    char s[100];
    scanf( "%s", s );
    return s[0];
}

void Append() {
    Student s;
    /*输入*/
    printf( "Please Input Stuednt Number: " );
    scanf( "%s", s.num );
    printf( "Plead Input The Score of Course1: " );
    scanf( "%lf", &s.course1 );
    printf( "Plead Input The Score of Course2: " );
    scanf( "%lf", &s.course2 );
    printf( "Plead Input The Score of Course3: " );
    scanf( "%lf", &s.course3 );
    s.sum = s.course1 + s.course2 + s.course3;   /*求各科成绩的和*/
    List_Append( s );   /*添加到列表中*/
    printf( "1 record(s) appended.\n" );
    modified = 1;  /*修改“文件已修改”标志*/
}
void List() {
    printf( "Number\tCourse1\tCourse2\tCourse3\tSum\n" );
    /*遍历链表每个元素*/
    Node *p = listHead;
    while( p != NULL ) {
        Student *s = &p->student;
        printf( "%s\t%.2lf\t%.2lf\t%.2lf\t%.2lf\n", s->num,
            s->course1, s->course2, s->course3, s->sum );
        p = p->next;
    }
}
void Delete() {
    char num[8];
    printf( "Please Input Stuednt Number to Delete: " );
    scanf( "%s", num );
    if( List_Delete( num ) ) {
        puts( "1 record(s) deleted" );
        modified = 1;  /*修改“文件已修改”标志*/
    } else {
```

```
            puts( "Can't Find The Record." );
        }
}

void Modify() {
        char num[8];
        /*输入学号*/
        printf( "Please Input Stuednt Number to Modify: " );
        scanf( "%s", num );
        Student *cur = List_Search( num ); /*在列表中查找*/
        if( cur != NULL ) {   /*如果找到，则修改*/
            printf( "Plead Input The Score of Course1: " );
            scanf( "%lf", &cur->course1 );
            printf( "Plead Input The Score of Course2: " );
            scanf( "%lf", &cur->course2 );
            printf( "Plead Input The Score of Course3: " );
            scanf( "%lf", &cur->course3 );
            cur->sum = cur->course1 + cur->course2 + cur->course3; /*计算分数和*/
            puts( "1 record(s) modified" );
            modified = 1; /*修改“文件已修改”标志*/
        } else {
            puts( "Can't Find The Record." );
        }
}

void Search() {
        char num[8];
        /*输入学号*/
        printf( "Please Input Stuednt Number to Search: " );
        scanf( "%s", num );
        Student *cur = List_Search( num );      /*在列表中查找*/
        if( cur != NULL ) { /*如果找到，则输出*/
            printf( "Number: %s\n", cur->num );
            printf( "Course1: %.2lf\n", cur->course1 );
            printf( "Course2: %.2lf\n", cur->course2 );
            printf( "Course3: %.2lf\n", cur->course3 );
            printf( "Sum: %.2lf\n", cur->sum );
        } else {
            puts( "Can't Find The Record." );
        }
}

void Sort( int (*cmp)(Student*,Student*) ) {
        Node *i, *j;
```

```
        for( i=listHead ; i!=NULL ; i=i->next ) {
            /*遍历求最值*/
            Node *m = i;
            for( j=i->next ; j!=NULL ; j=j->next ) {
                if( (*cmp)(&j->student, &m->student) < 0 ) {
                    m = j;
                }
            }
            /*交换*/
            if( m!=i ) {
                Student t = m->student;
                m->student = i->student;
                i->student = t;
            }
        }
        modified = 1; /*修改“文件已修改”标志*/
        puts( "Sorted." );
}

int SortNumDesc_Cmp( Student *a, Student *b ) {
        return strcmp( b->num, a->num );
}

void SortNumDesc() {
        Sort( SortNumDesc_Cmp );
}

int SortNumAsce_Cmp( Student *a, Student *b ) {
        return SortNumDesc_Cmp( b, a );
}

void SortNumAsce() {
        Sort( SortNumAsce_Cmp );
}

int SortSumDesc_Cmp( Student *a, Student *b ) {
        return a->sum > b->sum ? -1 : 1;
}

void SortSumDesc() {
        Sort( SortSumDesc_Cmp );
}

int SortSumAsce_Cmp( Student *a, Student *b ) {
```

```
    return SortSumDesc_Cmp( b, a );
}

void SortSumAsce() {
    Sort( SortSumAsce_Cmp );
}

void Write() {
    FILE* f = fopen( DATA_FILE_NAME, "w" );
    fprintf( f, "%s\n", DATA_FILE_HEAD );
    /*写入数据*/
    Node* s = listHead;
    while( s != NULL ) {
        fprintf( f, "%s %lf %lf %lf\n", s->student.num,
            s->student.course1, s->student.course2,
            s->student.course3 );
        s = s->next;
    }
    fclose( f );
    puts( "File Saved.\n" );
    modified = 0;  /*修改“文件已修改”标志*/
}

void Read() {
    char head[1000];
    FILE* f = fopen( DATA_FILE_NAME, "r" );
    if( f == NULL ) {
        printf( "Can Not Open File.\n" );
        return;
    }
    /*读入文件头*/
    fscanf( f, "%s", head );
    if( strcmp( head, DATA_FILE_HEAD ) != 0 ) {
        printf( "Bad File Format.\n" );
        fclose( f );
        return;
    }
    List_Clear();   /*清空表格*/
    /*写入数据*/
    Student s;
    while( fscanf( f, "%s%lf%lf%lf", s.num, &s.course1, &s.course2, &s.course3 ) == 4 ) {
        s.sum = s.course1 + s.course2 + s.course3;
        List_Append( s );
    }
```

```
    fclose( f );
    puts( "File Loaded.\n" );
    modified = 0; /*修改“文件已修改”标志*/
}

void Exit() {
    /*如果当前文件还没保存，则提示保存*/
    if( modified ) {
        printf( "File is modified, want to save?(y/n)" );
        if( toupper( getFirstChar() ) == 'Y' ) {
            Write();
        }
    }
    exit(0);
}

/*菜单项结构*/
typedef struct {
    const char *txt; /*显示文字*/
    char    key;            /*按键*/
    void (*fun)();     /*执行函数*/
}MenuItem;

/*菜单*/
MenuItem menuItems[] = {
    { "Management for Students' scores", 0, NULL },
    { "1. Append record", '1', Append },
    { "2. List record", '2', List },
    { "3. Delete record", '3', Delete },
    { "4. Modify record", '4', Modify },
    { "5. Search record", '5', Search },
    { "6. Sort in descending order by sum", '6', SortSumDesc },
    { "7. Sort in ascending order by sum", '7', SortSumAsce },
    { "8. Sort in descending order by num", '8', SortNumDesc },
    { "9. Sort in ascending order by num", '9', SortNumAsce },
    { "W. Write to a File", 'W', Write },
    { "R. Read from a File", 'R', Read },
    { "0. Exit", '0', Exit },
    { NULL, 0, NULL }
};

int main( ) {
    int i;
    List_Init();     /*初始化系统*/
```

```
    modified = 0;

    while(1) {
        /*显示菜单*/
        for( i=0 ; menuItems[i].txt != NULL ; i++ ) {
            puts( menuItems[i].txt );
        }
        /*读入选项*/
        printf( "Please Input your choice:" );
        char choice = getFirstChar();
        /*执行用户所选择的功能*/
        for( i=0 ; menuItems[i].txt != NULL ; i++ ) {
            if( toupper( choice ) == toupper( menuItems[i].key ) ) {
                if( menuItems[i].fun != NULL ) {
                    (*menuItems[i].fun)();
                    printf( "press any key to continue" );
                    getchar(); getchar();
                }
                break;
            }
        }
        puts( "" );
    }
    return 0;
}
```

6．完美的代价

回文串是一种特殊的字符串，它从左往右读和从右往左读是一样的，有人认为回文串是一种完美的字符串。现在给你一个字符串，它不一定是回文的，请你计算最少的交换次数使得该字符串变成一个回文串。这里的交换指将字符串中两个相邻的字符互换位置。

例如所给的字符串为”mamad”，第一次交换 a 和 d，得到”mamda”，第二次交换 m 和 d，得到”madma”；第三次交换最后面的 m 和 a，得到”madam”。

编写程序，从键盘读入数据。第一行是一个整数 N（N <= 80），表示所给字符串的长度，第二行是所给的字符串，长度为 N 且只包含小写英文字母。如果所给字符串能经过若干次交换变成回文串，则输出所需的最少交换次数；否则，输出 Impossible。

解答：

解题思路：首先判断是否能构成回文，如果累计值为奇数的字符大于 1 个，则不能构成回文。进行比较交换时采用贪心策略：依次遍历字符串的左半部字母，如果为累计值为奇数的字符，则从此位置查找与右侧对称位置相同的字母并交换，如果不是累计值为奇数的字符，则从右侧对称位置开始查找与此字母相同的字母并交换。

```
#include <stdio.h>
int main()
{
    int N;        /*字符串长度*/
```

```
int i;
scanf("%d",&N);
char s[8001];   /*定义字符数组*/
getchar();
gets(s); /*输入字符串*/

/*判断是否可构成回文串*/
int b[26] = {0};     /*记录 'a'~'z' 出现的次数*/
for (i = 0; i < N; i++)
    b[s[i] - 'a']++;   /*相应字母出现次数加 1*/
int odd = 0;       /*有多少个字母出现奇数次*/
char charodd = '\0';   /*出现奇数次的字母*/
for (i = 0; i < 26; i++)
  if (b[i] % 2 == 1)   /*b[i]是奇数*/
  {
     odd++;
     charodd = i + 'a'; /*记录该字母*/
  }
if (odd > 1)
  puts("Impossible"); /*输出*/
else
{
  int change = 0;         /*交换次数*/
  for (int i = 0; i < N/2; i++) /*依次考虑左侧的字母*/
  {
    if (s[i] == charodd)   /*若是 charodd，转而考虑右侧字母*/
    {
        int j = 0;
        for (j = i; j <= N-i-1; j++) /*从左侧该位置开始，找相同字母*/
          if (s[j] == s[N-i-1])   /*找到*/
             break;
        change += j - i;       /*需要 j-i 次移动可到左侧位置 */
        for (int k = j; k > i; k--)   /*实现字母的移动 */
           s[k] = s[k-1];
        s[i] = s[N-i-1];
    }
    else   /*考虑左侧字母*/
    {
        int j = 0;
        for (j = N-i-1; j >= i; j--) /*从右侧对称位置开始，找相同字母*/
           if (s[j] == s[i])     /*找到*/
              break;
        change += N-i-1 - j;     /*需要 N-i-j-j 次移动可到右侧位置*/
        for (int k = j; k < N-i-1; k++) /*实现字母的移动*/
```

```
                    s[k] = s[k+1];
                s[N-i-1] = s[i];
            }
        }
        printf("%d\n",change);    /*输出*/
    }
    return 0;
}
```