# 第 13 章　C 程序案例

在《C 程序设计(第四版)》一书中，为了便于教学，所介绍的例题程序的规模一般都不大。在学完教材的各章内容之后，需要综合应用已学过的知识，编写一些规模稍大、能供实际应用的程序，以提高编程能力。在本章中，介绍 3 个案例供读者参考。这些案例不仅有实用价值，而且对于读者今后编写程序会有很大的帮助。希望读者认真阅读这些程序，并且以此为借鉴，自己编写出类似的或更好的程序。

## 13.1　案例 1：个人所得税计算

### 1. 题目要求

输入一个纳税人的个人月收入，计算应纳个人所得税。

目前，我国的个人所得税税率表——(工资、薪金适用)见表 13.1。

表 13.1　我国现行的个人所得税税率表

| 级数 | 全月应纳税所得额(元) | 税率(%) | 速算扣除数(元) |
|---|---|---|---|
| 1 | 不超过 500 | 5 | 0 |
| 2 | 500～2000 | 10 | 25 |
| 3 | 2000～5000 | 15 | 125 |
| 4 | 5000～20000 | 20 | 375 |
| 5 | 20000～40000 | 25 | 1375 |
| 6 | 40000～60000 | 30 | 3375 |
| 7 | 60000～80000 | 35 | 6375 |
| 8 | 80000～100000 | 40 | 10375 |
| 9 | 超过 100000 | 45 | 15375 |

按照最新的规定，全月应纳税所得额＝月收入－1600。

而应纳个人所得税税额的计算公式是：

应纳个人所得税税额＝应纳税所得额×适用税率－速算扣除数

例如某人月收入 8000，减去 1600 元，全月应纳税所得额为 6400 元，应纳个人所得税税额＝6400×20%－375＝905 元。

### 2. 题目分析

使用条件语句就能完成程序的编写。但是如果考虑到人们的收入和国家的经济状况是

在不断变化的，个人所得税税额计算方法的细节则有可能产生变化。例如，上面公式中的1600是个人所得税的起征点，这个数字就有变化的可能性，有可能变为2000或2500等。程序处理时，可以将该数处理为由用户输入的数值。

又比如，表13.1中表示的全月应纳税所得额、税率和速算扣除数都有可能变化。那么，可以考虑将表中的内容写入一个文件，当数据发生变化时，就修改该文件的内容，而不用修改程序，这对于最终用户来说，是非常方便的。为了方便程序的处理，需要将表13.1略做修改，改为表13.2。

**表13.2　修改后的所得税税率表**

| 区间上限(元) | 区间下限(元) | 税率(%) | 速算扣除数(元) |
|---|---|---|---|
| 0 | 500 | 5 | 0 |
| 500 | 2000 | 10 | 25 |
| 2000 | 5000 | 15 | 125 |
| 5000 | 20000 | 20 | 375 |
| 20000 | 40000 | 25 | 1375 |
| 40000 | 60000 | 30 | 3375 |
| 60000 | 80000 | 35 | 6375 |
| 80000 | 100000 | 40 | 10375 |
| 100000 | | 45 | 15375 |

### 3. 编写程序

针对表13.2，要编写一个程序将这些数据写入一个文件中，程序中需要声明一个结构体类型strut tan_st，并用typedef声明一个类型名TAX_LIST，用它定义变量，存储表13.2的内容(有关typedef声明的用法请参考《C程序设计(第四版)》第9章)。

```
typedef struct tax_st                    //税率表结构体声明
  {int left;                             //区间上限
   int right;                            //区间下限
   double tax;                           //税率
   double deduct;                        //速算扣除数
  } TAX_LIST;
```

首先运行下面的程序将税率表的内容存储到文件d:\TAX.din中。程序运行时，从键盘将表13.2的内容输入，程序执行后，将产生文件d:\TAX.din。

```
#include "stdio.h"
#include "process.h"
#define SIZE 9
typedef struct tax_st                    //税率表数据结构定义
{long left;                              //区间上限
 long right;                             //区间下限
```

```
 int tax;                                          //税率
 long deduct;                                      //速算扣除数
}TAX_LIST;
void acceptdata(TAX_LIST tax_list[])
{int i;
 for((i=0;i<SIZE;i++)                              //接收键盘输入的数据
   {printf("Please enter data:");
    scanf("%ld",&tax_list[i].left);                //输入区间上限
    scanf("%ld",&tax_list[i].right);               //输入区间下限
    scanf("%d",&tax_list[i].tax);                  //输入税率
    scanf("%ld",&tax_list[i].deduct);              //输入速算扣除数
   }
}
int main()
{
 FILE *fp;
 TAX_LIST tax_list[SIZE];
 if(((fp=fopen("d:\\TAX.din","wb"))==NULL)         //打开文件 TAX.din
   {printf("\ncannot open file\n");
    exit(1);                                       //打开失败退出
   }
 acceptdata(tax_list);                             //调用函数从键盘接收数据
 if((fwrite(tax_list,sizeof(TAX_LIST),SIZE,fp)!=SIZE)
                                                   //将数组 tax_list 的结构数据一次写入文件
 printf("file write error\n");
 fclose(fp);                                       //关闭文件
 return 0;
}
```

接下来,可以运行下面的程序计算个人所得税额。

```
#include "stdio.h"
#include "process.h"
#define SIZE 9
typedef struct tax_st                              //税率表结构体声明
{long left;                                        //区间上限
 long right;                                       //区间下限
 int tax;                                          //税率
 long deduct;                                      //速算扣除数
}TAX_LIST;
void disp(TAX_LIST tax_list[])
{double salary,s,tax,ff;                           //定义变量
 int i;
 printf("请输入税前扣除数:");                       //提示用户税前扣除数
 scanf("%lf",&ff);
 printf("请输入月收入:");                           //提示用户输入月收入
```

```
 scanf("%lf",&salary);                                    //接收用户输入的月收入
 if(salary>=0)                                            //月收入大于0,开始计算
   {
    s=salary-ff;                                          //计算全月应纳税所得额
    if(s<=0)
      tax=0;                                              //小于0,税额为0
    else
      {for(i=0;i<8;i++)                                   //根据数组内容计算税额
         {if(s<tax_list[i].right&&s>=tax_list[i].left)
            tax=s*tax_list[i].tax/100.-tax_list[i].deduct;
         }
         if(s>=tax_list[8].left)
            tax=s*tax_list[8].tax/100.-tax_list[8].deduct;
      }
   }
 printf("应纳个人所得税额是%.2lf\n",tax);
}
int main()
{FILE *fp;                                                 //定义文件指针
 TAX_LIST tax_list[SIZE];                                 //定义结构体数组
 if((fp=fopen("d:\\TAX.din","rb"))==NULL)                 //打开文件TAX.din
   {printf("\ncannot open file\n");
    exit(1);                                              //打开文件失败,退出
   }
 if(fread(tax_list,sizeof(TAX_LIST),SIZE,fp)!=SIZE)       //将SIZE个结构体一次读入数据区
   {printf("file write error\n");
    exit(1);                                              //读操作失败,退出
   }
 disp(tax_list);                                          //计算税额
 fclose(fp) ;                                             //关闭文件
 return 0;
}
```

**4. 运行结果**

```
请输入税前扣除数:1600↙
请输入月收入:8000↙
应纳个人所得税额是905.00
```

**注意**:本例通过文件操作存储了税率表,一旦税率表有所改动,可以对文件进行更新,便于用户使用,同时程序也变得简单、易读,只是对于不熟悉文件操作的读者会有一定的难度。如果是这样的话,可以考虑将表13.2的数据直接从键盘输入,通过调用acceptdata函数将这些数据读入数组中。

下面是调用acceptdata的主函数。其中相关的数据定义、acceptdata函数定义和disp函数定义与上面的程序一致。

```
int main()
  { TAX_LIST tax_list [SIZE];          //定义结构体数组
    acceptdata(tax_list);              //接收税率表
    disp(tax_list);                    //计算税额
    return 0;
  }
```

当然，这样修改以后，程序运行时的操作会比较麻烦，每次都要输入税率表，所以还是建议读者尽量学会使用文件操作。

另外，本程序还有改进的余地，表 13.2 中的区间上限和区间下限并不需要都存储，只需要存储区间上限就可以编写程序了。留给读者做练习吧！

## 13.2 案例 2：学生试卷分数统计

### 1. 题目要求

作为教师，考试以后对试卷进行分析和研究是必须要做的一个工作，表 13.3 就是某学校要求老师在考试之后填写的一个表格，并要求教师根据考试分数分布情况画出直方图。下面就来解决这个实际问题。

**表 13.3 某高校试卷分析表**

| 分 类 | 项 目 | 不及格 | 60～69 分 | 70～79 分 | 80～89 分 | 90～100 分 | 平均分 | 标准差 |
|---|---|---|---|---|---|---|---|---|
| 平时成绩 | 人数 | 8 | 6 | 16 | 35 | 11 | 77.3 | 15.04 |
| | 比例 | 10.53% | 7.89% | 21% | 46% | 14% | | |
| 期末成绩 | 人数 | 20 | 11 | 14 | 19 | 12 | 71.5 | 19.11 |
| | 比例 | 26% | 14% | 18% | 25% | 16% | | |
| 总评 | 人数 | 12 | 13 | 19 | 19 | 13 | 72.6 | 17.97 |
| | 比例 | 16% | 17% | 25% | 25% | 17% | | |
| 期末考试卷面及格率：73.68%　　期末考试卷面最高分：97　　期末考试卷面最低分：28<br>总评＝平时×20%＋考试×80%　　学生总人数 76 名 | | | | | | | | |

### 2. 题目分析

(1) 程序运行时，首先必须接收总评成绩的计算比例，因为针对不同的课程，平时成绩和期末考试成绩所占的比例可能不同。

(2) 接收若干同学的平时成绩和期末考试成绩，计算出总评成绩，总评成绩的计算方法是“平时成绩所占比例×平时成绩＋期末成绩所占比例×期末成绩”。

(3) 根据考试成绩计算分数段的分布情况，画出直方图。

(4) 计算平时成绩、期末成绩和总评成绩的平均分和标准差，以及期末考试卷面的及格率、最高分和最低分等。

由于针对一个学生有 3 个有关成绩的数据，因此最简单的方法就是使用结构体数组。

本题使用的存储结构如图 13.1 所示。第 1 列为学生的学号，第 2 列为学生的平时成绩，第 3 列为学生的期末成绩，第 4 列为学生的总评成绩。

| | | | |
|---|---|---|---|
| 1 | 80 | 90 | 87 |
| 2 | 70 | 80 | 77 |
| 3 | 80 | 77 | 66 |
| 4 | 88 | 69 | 75 |
| 5 | 99 | 78 | 84 |

图 13.1

本例使用模块化程序思想，将程序分解为几个函数，函数的功能和调用关系如图 13.2 所示。

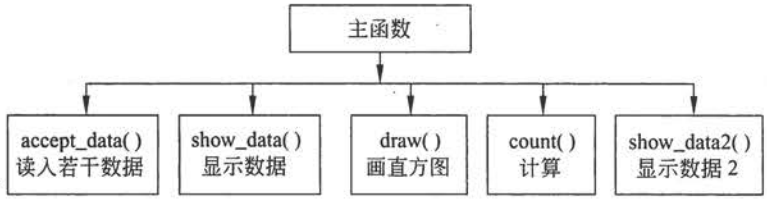


图 13.2

这里需要说明一下标准差的概念：标准差描述一组数据离散程度或个别差异程度。例如，A，B 两班学生各 34 人，其 C 语言考试平均成绩都是 80 分，但甲班最高分数为 98 分，最低 42 分，而乙班最高分数为 86 分，最低 60 分。初步分析，两班考试成绩不一样，A 班学生的成绩个别差异程度大、水平参差不齐，而 B 班学生的成绩个别差异程度小，整齐度大些。标准差就可以刻画一组数据的差异程度，标准差的计算公式是：

$$S=\sqrt{\frac{1}{n}[(x_1-\bar{x})^2+(x_2-\bar{x})^2+\cdots+(x_n-\bar{x})^2]}$$

其中，$x_i$为某个同学的成绩；$\bar{x}$ 为平均成绩。

在本例中，一个学生的情况可以用一个结构体变量来表示，100 个学生的情况就可以用一个结构体数组来存储了。

**3. 编写程序**

学生的结构体类型可以用类型定义表示为：

```
typedef struct student                  //学生数据结构体声明
{ int number;                           //学号
  int score[3];                         //平时、期末和总评成绩
} STUDENT;
```

其中，number 是学号或是排列序号(可以简单一点)。score 数组表示一个学生的 3 个成绩，score[0]为平时成绩，score[1]为期末成绩，score[2]为总评成绩。

```
#include "stdio.h"
```

```
#include "string.h"
#include "conio.h"
#include "math.h"
#define SIZE 300
typedef struct student                                  //学生数据结构体声明
{ int number;                                           //学号
  int score[3];                                         //平时、期末和总评成绩
} STUDENT;
typedef enum boolen                                     //枚举定义
{ False,True
} FLAG;
int accept_data(STUDENT stu[],int grade[]);             //输入数据函数声明
void show_data(STUDENT stu[],int sum,int grade[]);      //输出所有学生的序号、平时成绩、
                                                        //期末成绩和总评成绩函数说明
void draw(int grade[]);                                 //画直方图函数声明
void count(int * max,int * min,double * pass,double ave[],double f[],STUDENT stu[],int sum);
void show_data2(int max,int min,double pass,double ave[],double f[]);
                    //显示期末考试成绩的及格率、最高分、最低分以及平时、期末和总评成绩的
                                                        //平均分和标准差函数说明
int main()
{ int sum,max,min;                                      //数据定义
  double pass=0;
  int grade[11]={0};
  STUDENT stu[SIZE];
  double ave[SIZE],f[SIZE];
  sum=accept_data(stu,grade);                           //输入数据(sum 为总人数)
  show_data(stu,sum,grade);
                                    //输出所有学生的序号、平时成绩、期末成绩和总评成绩
  draw(grade);                                          //画模拟直方图
  count(&max,&min,&pass,ave,f,stu,sum);                 //计算期末考试成绩的及格率最高分、最低分
                                                        //以及平时、期末和总评成绩的平均分和标准差
  show_data2(max,min,pass,ave,f);
            //显示期末考试成绩的最高分、最低分以及平时、期末和总评成绩的平均分和标准差
  return 0;
}

int accept_data(STUDENT stu[],int grade[])
  { int i=0,sum=0,temp,a1,a2;
    FLAG flag;
    printf("\n 请输入计算总评成绩时使用平时成绩与期末成绩的比例,用整数表示");
    scanf("%d%d",&a1,&a2);                              //接收计算总评成绩的比例
    while(i<SIZE)
    {
     printf("\n 请输入学号:");
     scanf("%d",&stu[i].number);                        //输入学号
     if (stu[i].number==-1)                             //序号是-1则跳出循环
     {sum=i;                                            //sum 记录的是输入的人数
```

```
        break;
      }
      printf("\n 请输入学生的平时成绩和期末成绩:");
      flag=True;
      while(flag==True)                                    //重复读入两个成绩,读到正确的为止
      {scanf("%d%d",&stu[i].score[0], &stu[i].score[1]);
       if(stu[i].score[0]<=100&& stu[i].score[0]>=0&&\
          stu[i].score[1]<=100&& stu[i].score[1]>=0)
          flag=False;                                      //输入的两个成绩合理
       else
          printf("\n\007 错误数据! 请再次输入学生的平时成绩和期末成绩:");
                                                           //输入的两个成绩不合理
      }
      temp=(int)(1.0 * a1/100 * stu[i].score[0]+1.0 * a2/100 *  stu[i].score[1]);
                                                           //计算总评成绩
      stu[i].score[2]=temp;                                //总评成绩存入数组
      temp=(stu[i].score[1])/10;                           //计算分数段
      if(temp==10)                                         //分数段存入数组
         grade[10]++;                                      //100 分存入数组元素 grade[10]
      else
         grade[temp+1]++;                                  //90~99 分存入数组元素 grade[9]
                                                           //80~89 分存入数组元素 grade[8]
                                                          //70~79 分存入数组元素 grade[7],依次类推
      i++;
     }
     return sum;                                           //返回人数
   }

void show_data(STUDENT stu[],int sum,int grade[])
{
 int i,j;
 for(i=0;i<sum;i++)                                        //输出所有学号、平时、期末和总评成绩
 {
  printf("%4d",stu[i].number);                             //输出所有学号
  for(j=0;j<3;j++)                                         //输出 3 个成绩
  printf("%4d ", stu[i].score[j]);
  printf("\n");
 }
 for(i=1;i<=10;i++)                                        //输出分数的分布情况
    printf(" %d ",grade[i]);
}
void count(int * max,int * min,double * pass,double ave[],double f[],STUDENT stu[],int sum)
{ int i,j, p_sum=0;
  int total[3];
  double temp;
  * max= * min=stu[0].score[1];                            //设卷面成绩的最高分、最低分初值
  if(stu[0].score[1]>=60)
```

```
    p_sum++;
  for(i=1;i<sum;i++)
  { if ((stu[i].score[1])>*max)                    //若高于最高分,将其覆盖
        *max=stu[i].score[1];
    if ((stu[i].score[1])<*min)                    //若低于最低分,将其覆盖
        *min=stu[i].score[1];
    if(stu[i].score[1]>=60)
        p_sum++;                                   //计算及格的人数
  }
  *pass=(1.0*p_sum/sum)*100;                       //计算及格率
  for(i=0;i<=2;i++)                                //平时、期末、总评的初值设置为 0
     total[i]=0;
  for(j=0;j<3;j++)                                 //求平时、期末、总评 3 个总分
     for(i=0;i<sum;i++)
       {
        total[j]=total[j]+stu[i].score[j];
       }
  for(j=0;j<3;j++)                                 //求平时、期末、总评 3 个平均分
    { ave[j]=total[j]/sum;
    }
  for(j=0;j<3;j++)                                 //求平时、期末、总评标准差
    {f[j]=0;                                       //标准差初值设置为 0
     for(i=0;i<sum;i++)                            //计算标准差
       {
        temp=stu[i].score[j]-ave[j];
        f[j]=f[j]+temp*temp;
       }
     f[j]=sqrt(fabs(f[j])/sum);
    }
}

void show_data2(int max,int min, double pass,double ave[],double f[])
                        //输出期末及格率、最高分、最低分以及平时、期末、总评的平均分和标准差
{ int j;
  char str1[3][20]={"平时成绩平均分","期末成绩平均分","总评成绩平均分"};
  char str2[3][20]={" 平时成绩标准差","期末成绩标准差","总评成绩标准差"};
  printf("\n 及格率=%6.2f %% 最高分=%d 最低分=%d\n",pass, max,min);
                                       //输出期末及格率、最高分、最低分
  for(j=0;j<3;j++)
                                       //循环 3 次分别输出平时、期末、总评的平均分和标准差
     printf("\n %s=%6.2f %s=%6.2f\n",str1[j],ave[j],str2[j],f[j]);
}

void draw(int grade[])                 //输出模拟直方图
{ int i, j,max,k,temp,x;
  char screen[22][44];                 //定义一个字符型数组,用来表示屏幕的输出
  printf("\n 模拟直方图\n");
```

```
  max=0;
  for(i=1;i<=10;i++)                          //寻找分数段中人数最多的
    if(grade[i]>max)
      max=grade[i];
  for(i=1;i<=10;i++)
  {grade[i]=(int)(20.0*grade[i]/max+0.5);     //计算显示时应该输出的*号的个数
  }
  for(i=0;i<=21;i++)                          //先将输出的所有点清零
  for(j=0;j<=41;j++)
     screen[i][j]=0;
  //画 x 轴
  for(i=0;i<=41;i++)                          //x 轴的所有点设置为符号-
     screen[21][i] = '-';
  screen[21][41] = 'X';                       //显示 X 字样
  //画 y 轴
  screen[0][0] = 'Y';                         //显示 Y 字样
  for(i=1; i<=21;i++)                         //y 轴的所有点设置为符号|
     screen[i][0] = '|';

                                              //将符合条件的点(x,y)赋值为星号
  k=1;
  for(x=1;x<=10;x++,k=k+4)                    //x 控制输出的行,k 控制输出的列
  {temp=grade[x];                             //temp 取分数的值
   if(temp!=0)
     for(i=1;i<=temp;i++)                     //分数不为 0,赋值为星号
     {
        for(j=1;j<=4;j++)                     //该分数段的每行对应 4 个星号
        {screen[20-i+1][j+k]='*';
        }
     }
  }
                                              //输出数组,在屏幕上画图
for( i = 0; i <=21; i++)
{for( j = 0; j <=41; j++)
    if(screen[i][j]!=0)                       //数组内容不为 0,输出数组中的内容
      printf("%c",screen[i][j]);
    else
      printf(" ");                            //否则输出空格
    printf("\n");
}
printf(" 0 10 20 30 40 50 60 70 80 90 100\n");   //输出分数段
getch();
}
```

#### 4. 运行结果

例 13.2 运行后产生的结果(模拟直方图由于数据太少不符合正态分布)见图 13.3。
例 13.2 运行正常数据产生的模拟直方图见图 13.4。

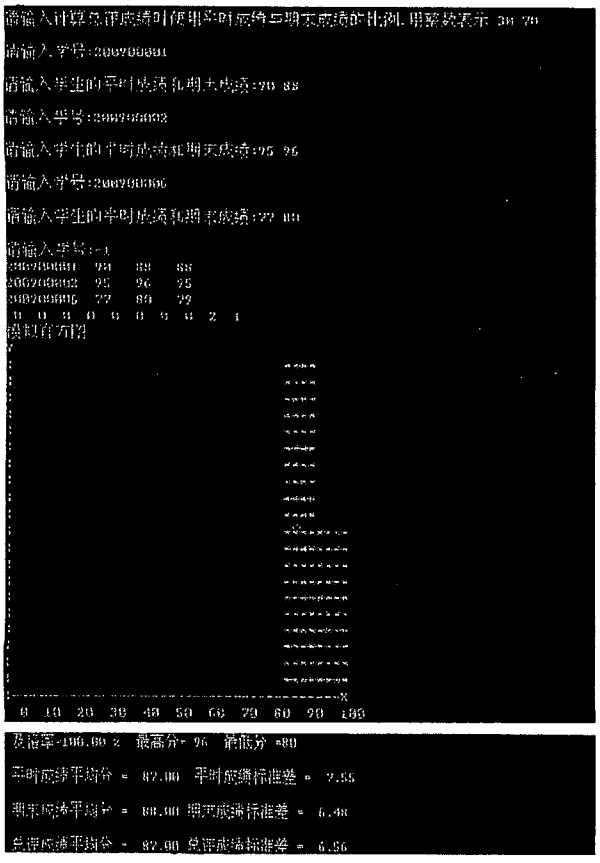

图 13.3

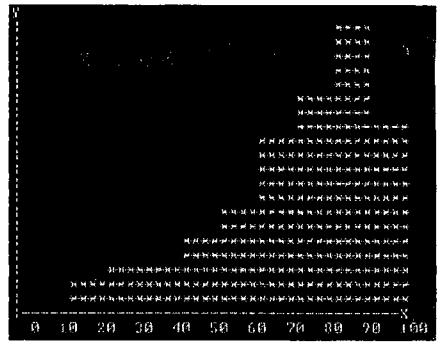

图 13.4

本程序实现的计算与表 13.3 表示的某高校试卷分析表大致相同，个别地方有区别，例如程序中图形的分数分段方法是以 10 分为一档，因为，这样的图形画出来更好看一些，很容易看出是不是正态分布。当然，只要将程序稍加修改，按表 13.3 的分数分段画图也是非常容易的，留给读者作为思考。

另外，有些编译环境有可能不能显示正确的结果，必须使用纯英文的环境。

## 13.3 案例 3：电话订餐信息处理

### 1. 题目要求

一个小饭馆的生意非常红火，要想用餐必须提前一天打电话预订。假设我们是饭馆负责接受电话预订的服务员，我们需要做些什么呢？首先，需要准备一张大小适当的白纸，等待顾客的电话。李红最先打进了电话，她预约第 2 天中午 12 点用餐，用餐人数 5 人，服务员在纸上做了如下记录："李红　12 点　5 人"；接着，刘娜、汪寒、孙杰和赵军分别来了电话，服务员根据他们的要求做了记录，此时纸上记录的内容是：

| | | |
|---|---|---|
| 李红 | 12 点 | 5 人 |
| 刘娜 | 11 点 | 2 人 |
| 汪寒 | 11 点 50 | 3 人 |
| 孙杰 | 10 点 10 | 4 人 |
| 赵军 | 13 点 20 | 6 人 |

孙杰随后又来电话，将用餐时间推后一个小时，那么记录的内容也应该做相应的修改。

刘娜来用餐以后，可以将其信息从纸上划去。

下面编写程序处理电话订餐的情况。

**2. 题目分析**

这是一个小型的管理系统，可以使用结构数组存储订餐的情况。每个结构的数据可以包括姓名、人数、用餐时间等。

为了方便处理，还需要给每个打进电话的客户编个号，就像在饭馆等候用餐时，服务员会发号给客户一样。

**3. 编写程序**

可以声明以下的结构体类型：

```
struct guest_info {
        char name[8];              //姓名
        int sum;                   //人数
        char time[10];             //用餐时间
        int number;                //编号
}GuestList[MaxSize];
```

程序包括 5 个函数 Insert，Search，Update，Delete 和 Show，分别负责插入、查询、修改、删除和显示数据。一般的信息管理系统都应该具备这几个功能，小型信息管理系统也不例外。

程序采用简单菜单驱动方式，屏幕上显示菜单如下：

```
1———插入(Insert)
2———查询(Search)
3———修改(Update)
4———删除(Delete)
5———显示(Show)
6———退出(Exit)
```

**完整程序如下：**

```
#include ";stdio.h";
#include ";string.h";
#define MaxSize 20
struct guest_info {
        char name[8];                          //姓名
        int sum;                               //人数
```

```
        char time[10];                          //用餐时间
        int number;                             //编号
} GuestList[MaxSize];
void Insert(int *);
void Search(int );
void Update(int);
void Delete(int *);
void Show(int);
int main()
{
 int i;
 int count=0;                                   //count 为计数器,记录已经登记记录个数
 do                                             //显示一个简易菜单
 { printf("\n");
   printf("1---插入(Insert)\n");
   printf("2---查询(Search)\n");
   printf("3---修改(Update)\n");
   printf("4---删除(Delete)\n");
   printf("5---显示(Show)\n");
   printf("6---退出(Exit)\n");
   scanf("%d",&i);                              //接收用户的选择
   switch(i)
   { case 1:Insert(&count);                     //调用插入运算
            break;
     case 2:Search(count);                      //调用查询运算
            break;
     case 3:Update(count);                      //调用修改运算
            break;
     case 4:Delete(&count);                     //调用删除运算
            break;
     case 5:Show(count);                        //调用显示运算
            break;
     case 6:break;
     default:printf("错误选择! 请重选");break;
   }
 } while(i!=6);
 return 0;
}

void Insert(int *count)
{ int i, in_number;
  if(*count==MaxSize)
  { printf("空间已满!");return;}
  printf("请输入编号:");
  scanf("%d",&in_number);
  for(i=0;i<*count;i++)                         //查找符合条件的记录
  if(GuestList[i].number==in_number)
```

```
    { printf("已经有相同的编号:");return;}
    GuestList[i].number= in_number;                    //接收插入数据
    printf("请输入姓名:");
    scanf("%s",GuestList[i].name);
    printf("请输入人数:");
    scanf("%d",&GuestList[i].sum);
    printf("请输入用餐时间:");
    scanf("%s",GuestList[i].time);
    (*count)++;
}

void Search(int count)
{ int i,number,flag=1;                                 //设置一个标记变量
  printf("请输入要查询的编号:");
  scanf("%d",&number);
  for(i=0;i<count&&flag;i++)
  if(GuestList[i].number==number)                      //检索到则输出
  { printf("姓名:%s",GuestList[i].name);
    printf("人数:%d",GuestList[i].sum);
    printf("用餐时间:%s",GuestList[i].time);
    flag=0;                                            //标记变量值变反
  }
  else
    printf("没有查询到!!");
}

void Update(int count)
{ int i,number, flag=1;                                //设置一个标记变量
  printf("请输入要修改数据的编号:");
  scanf("%d",&number);
  for(i=0;i<count&&flag;i++)
  if(GuestList[i].number==number)                      //检索到则修改
  {
    printf("请输入人数:");
    scanf("%d",&GuestList[i].sum);
    printf("请输入用餐时间:");
    scanf("%s",GuestList[i].time);
    flag=0;                                            //标记变量值变反
  }
  else
    printf("没有查询到可以修改的数据!!");
}
void Delete(int *count)
{ int i,j,number,flag=1;                               //设置一个标记变量
  printf("请输入要删除数据的编号:");
  scanf("%d",&number);
  for(i=0;i<*count&&flag;i++)
```

```
  { if(GuestList[i].number==number)
    { for(j=i;j<*count-1;j++)
          GuestList[j]=GuestList[j+1];
      flag=0;                                    //标记变量值变反
      (*count)--;
    }
    else
       printf("没有查询到可以删除的数据!!");
  }
}

void Show(int count)                             //列表显示数据
{ int i;
  printf("\n");
  printf(" 编号 姓名 人数 用餐时间\n");
  for(i=0;i<count;i++)
  { printf("%10d",GuestList[i].number);         //显示编号
    printf("%12s",GuestList[i].name);           //显示姓名
    printf("%10d",GuestList[i].sum);            //显示人数
    printf("%12s\n",GuestList[i].time);         //显示用餐时间
  }
}
```

在上面的程序中，客户的订餐信息是存储在一个数组中的。数组是一种处理数据的存储方式，下面用单链表存储这组数据。因为指针是 C 语言的精髓，不能掌握指针的用法，不能说学会了 C 语言。

要建立单链表，首先需要正确的定义每个结点的数据是如何构成的，下面是订餐信息存储在链表中的数据定义。图 13.5 则是示意图，表示链表中有 3 个结点时的情况。

```
typedef struct guest_info {
     char name[8];                  //姓名
     int sum;                       //人数
     char time[10];                 //用餐时间
     int number;                    //编号
     struct guest_info * next;
} GuestLink;
```

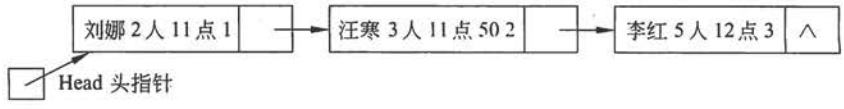


图 13.5

对于单链表，插入、查询、修改、删除和显示也是必须要完成的 5 个操作。曾在前面讨论过有关单链表的操作方式，本例是尝试将有关单链表的操作集中起来，构成一个完整的管理系统，供读者参考和使用。图 13.6 显示程序部分运行情况。

```
# include "stdio.h"
```

```
#include "string.h"
#include "stdlib.h"
#define MaxSize 20
typedef struct guest_info
{
  char name[8];                          //姓名
  int sum;                               //人数
  char time[10];                         //用餐时间
  int number;                            //编号
  struct guest_info * next;
} GuestLink, * Pointer;
void Insert(Pointer * Head);             //函数声明
void Search(Pointer Head);
void Update(Pointer Head);
void Delete(Pointer * Head);
void Show(Pointer Head);
int main()
{
  Pointer Head=NULL;                     //定义表头指针
  int i;
  do                                     //显示一个简易菜单
  { printf("\n");
    printf("1---插入(Insert)\n");
    printf("2---查询(Search)\n");
    printf("3---修改(Update)\n");
    printf("4---删除(Delete)\n");
    printf("5---显示(Show)\n");
    printf("6---退出(Exit)\n");
    scanf("%d",&i);                      //接收用户的选择
    switch(i)                            //调用对应的函数
    { case 1:Insert(&Head);
            break;
      case 2: Search(Head);
            break;
      case 3: Update(Head);
            break;
      case 4: Delete(&Head);
            break;
      case 5: Show(Head);
            break;
      case 6: break;
      default:printf("错误选择! 请重选");break;
    }
  } while(i!=6);
  return 0;
}
```

```
void Insert(Pointer * Head)                    //插入函数的定义
{ int in_number;
  Pointer p,q,r;                               //说明变量
  printf("请输入编号:");
  scanf("%d",&in_number);
  p=q= * Head;                                 //查找符合条件的记录
  while(p!=NULL)
  { if(p->number== in_number)                  //找到相同的编号
    { printf("已经有相同的编号:");return;}
    else
    { q=p;p=p->next;}                          //走链
  }
  r=(Pointer)malloc(sizeof(GuestLink));        //申请空间
  r->next=NULL;                                //设置指针域
  if(r==NULL)
  { printf("分配空间失败!");return;}
  if(q==NULL)                                  //原表为空表
   * Head=r;                                   //新结点作为头元素
  else
  { q->next=r;                                 //在表尾插入元素
  }
  r->number=in_number;                         //接收插入数据
  printf("请输入姓名:");
  scanf("%s", r->name);
  printf("请输入人数:");
  scanf("%d",&r->sum);
  printf("请输入用餐时间:");
  scanf("%s", r->time);
}

void Search(Pointer Head)                      //查找函数的定义
{ int flag=1;                                  //设定标记变量的初值
  int number;
  Pointer p;
  printf("请输入要查询的编号:");
  scanf("%d",&number);
  p=Head;                                      //查找符合条件的记录
  while(p!=NULL&&flag)
  { if(p->number== number)
    { printf("姓名:%s",p->name);
      printf("人数:%d",p->sum);
      printf("用餐时间:%s",p->time);
      flag=0;                                  //找到标记变量设为0
    }
    else
        p=p->next;                             //指针走到下一个结点
  }
```

```
  if(flag)
    printf("没有查询到!!");
}

void Update(Pointer Head)                          //修改函数的定义
{ int flag=1;                                      //设定标记变量的初值
  int number;
  Pointer p;
  printf("请输入要修改的编号:");
  scanf("%d",&number);
  p=Head;                                          //查找符合条件的记录
  while(p!=NULL&&flag)
  { if(p->number== number)
    {
      printf("请输入人数:");
      scanf("%d", p->sum);
      printf("请输入用餐时间:");
      scanf("%s", p->time);
      flag=0;
    }
    else
      p=p->next;                                   //指针走到下一个结点
  }
  if(flag)
      printf("没有找到要修改的记录!!");
}

void Delete(Pointer * Head)                        //删除函数的定义
{ int flag=1;
  int number;
  Pointer p,q;
  printf("请输入要删除数据的编号:");
  scanf("%d",&number);
  p=q= * Head;                                     //查找符合条件的记录
  while(p!=NULL&&flag)
  { if(p->number== number)
    {
      if(p== * Head)                               //删除的是表头元素
      { * Head=p->next; free(p);}
        else
        { q->next=p->next; free(p);}               //删除普通元素
        flag=0;
    }
    else
    { q=p; p=p->next;}                             //指针走到下一个结点,
                                                   //q 所指结点为 p 所指结点的前驱
  }
```

```
    if(flag)
    printf("没有找到可以删除的数据!!");


void Show(Pointer Head)                          //列表显示数据
{ Pointer p;
  p= Head;
  while(p!=NULL)
  { printf("姓名：%-10s",p->name);
    printf("人数：%-10d",p->sum);
    printf("用餐时间：%-10s",p->time);
    printf("编号：%-10d\n",p->number);
    p=p->next;
  }
}
```

### 4. 运行结果

运行结果见图 13.6。


图　13.6

如果能够将本例的数据存储到文件中，那么就真正地实现了一个小型的管理信息系统（能将数据存储在磁盘中），请读者参考本章案例 1 中有关文件操作的使用方法对本例进行修改，相信能有很大的收获。