第3章 实验项目

实验项目1　运行一个C程序

一、实验目的

(1) 熟悉 Visual C++ 6.0 和 C-Free 5.0 两种 C 语言运行环境。

(2) 掌握在上述两种环境下编辑、编译、连接和运行一个 C 程序的方法。

(3) 通过运行简单的 C 程序，认识 C 程序的特点，掌握和理解 C 程序的结构。

二、实验要求

(1) 进入 Visual C++ 6.0 或 C-Free 5.0 的集成环境。

(2) 熟悉 Visual C++ 6.0 或 C-Free 5.0 集成环境，掌握系统主菜单中常用命令的使用。

(3) 运行简单的 C 程序，逐步掌握编辑、编译、连接、运行和调试 C 程序的方法。

三、实验内容

1. 验证性实验

(1) 在 Visual C++ 6.0 或 C-Free 5.0 集成环境下编辑下列 C 语言程序，编译、连接并运行，观察并理解其运行结果。

```
/* 实验 1-1.C */
#include <stdio.h>
int main()
{
      int a,b,c;

      printf("Enter first integer:");
      scanf("%d",&a);
      printf("Enter second integer:");
      scanf("%d",&b);
      c = a + b;
      printf("a + b = %d\n",c);
      return 0;
}
```

程序的运行结果如图 3-1 所示。

思考：

① 去掉语句“#include <stdio.h>”，运行程序，观察运行结果，并分析为什么。

② 去掉语句“int a,b,c;”中的“;”,运行程序,观察运行结果,并分析为什么。

③ 将语句“c=a+b;”改为“C=a+b;”,运行程序,观察运行结果,并分析为什么。

(2) 尝试修改下列程序中的错误,直到程序经编译后没有错误信息,并得到题目要求的运行结果。

题目要求得到的输出结果如图 3-2 所示。

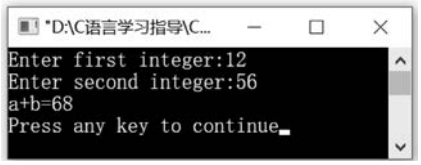


图 3-1 从键盘输入两个整数并求它们的和

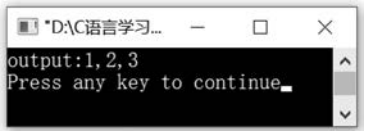


图 3-2 实验 1-2.C 要求得到的输出结果

含有错误的源程序如下:

```
/* 实验 1-2.C */
#include <stdio.h>
int main()
{
    int a=1;b=2,c=3,
    printf("output:%d,%d,%d\n"a,b,c);
    return 0;
}
```

2. 设计性实验

(1) 编写程序,从键盘上输入两个整型变量 a 和 b,求它们的差并输出。

实验提示

① 可参考实验 1-1.C。

② 保存源程序为“练习 1-1.C”。

(2) 编写程序,从键盘上输入 x 的值,根据公式 $y=x^2+1$ 求 y 的值,输出 x 和 y 的值(假设 x 和 y 都是 int 型变量)。

实验提示

① x^2 可表示为 x*x。

② 可参考实验 1-1.C。

③ 保存源程序为“练习 1-2.C”。

(3) 编写程序,从键盘上输入 x 的值,求 x 的平方根并赋给变量 y,输出 x 和 y 的值(假设 x 和 y 都是 int 型变量)。

实验提示

① 可用函数 sqrt(x)求 x 的平方根,要求使用该函数,必须在程序前面包含头文件 math.h。

② 可参考实验 1-1.C。

③ 保存源程序为“练习 1-3.C”。

实验项目 2 数据类型与表达式

一、实验目的

(1) 熟练掌握运行 C 程序的方法。

(2) 正确使用常量和变量,掌握指针变量的简单使用方法。

(3) 掌握基本数据类型的使用。

(4) 掌握常用输入/输出函数的使用。

(5) 理解常用运算符的意义,了解表达式的运算规则。

二、实验要求

(1) 理解常量和变量的概念。

(2) 掌握数据在内存中的存储形式。

(3) 掌握常用输入/输出函数的格式规范。

(4) 熟悉常用运算符的运算规则。

三、实验内容

1. 验证性实验

(1) 输入以下程序,分析程序运行结果。

```
/* 实验 2_1.c */
#include <stdio.h>
int main()
{
    char c1,c2;          /* 第 5 行 */
    c1 = 'a';            /* 第 6 行 */
    c2 = 'b';            /* 第 7 行 */
    printf("%c,%c\n",c1,c2);
    printf("%d,%d\n",c1,c2);
    return 0;
}
```

程序的运行结果如图 3-3 所示。

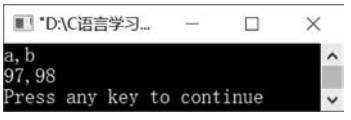


图 3-3 字符变量的输出

思考:

① 将第 6 行和第 7 行改为:

```
c1 = a;
c2 = b;
```

系统报错原因是什么?

② 将第 6 行和第 7 行改为:

```
c1 = "a";
c2 = "b";
```

系统报错原因是什么?

③ 将第 6 行和第 7 行改为:

```
c1 = 97;
c2 = 98;
```

再运行程序,程序输出结果是什么?

④ 将第 6 行和第 7 行改为:

```
c1 = 127;
```

```
c2 = 128;
```

再运行程序,第 2 行为什么输出 127,-128?

⑤ 将第 5 行改为:

```
int c1,c2;
```

再运行程序,程序输出结果是什么? 为什么?

(2) 输入以下程序,分析程序运行结果。

```
/* 实验 2_2.c */
#include <stdio.h>
int main()
{
    int a = 1000,b = 01750,c = 0x3e8;
    printf("%d,%o,%x\n",a,a,a);        /* 第 6 行 */
    printf("%d,%d,%d\n",a,b,c);        /* 第 7 行 */
    return 0;
}
```

程序的运行结果如图 3-4 所示。

思考:

① 将第 6 行和第 7 行改为:

```
printf("%d,%o,%x\n",b,b,b);
printf("%o,%o,%o\n",a,b,c);
```

再运行程序,程序输出结果是什么?

② 将第 6 行和第 7 行改为:

```
printf("%d,%o,%x\n",c,c,c);
printf("%x,%x,%x\n",a,b,c);
```

再运行程序,程序输出结果是什么?

(3) 输入以下程序,分析程序运行结果。

```
/* 实验 2_3.c */
#include <stdio.h>
int main()
{
    float x1,y1;                          /* 第 5 行 */
    x1 = 1111111111111.111111111;
    y1 = 2222222222222.222222222;
    printf("x1 = %f,y1 = %f\n",x1,y1);    /* 第 8 行 */
    printf("x1 + y1 = %f\n",x1 + y1);     /* 第 9 行 */
    return 0;
}
```

程序的运行结果如图 3-5 所示。

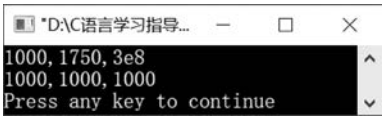


图 3-4 整数的输出

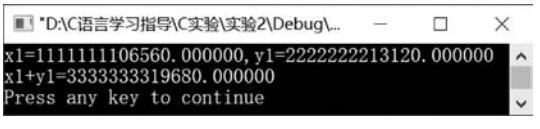


图 3-5 两个单精度数的求和

思考：

① 将第 5 行改为：

```
double  x1,y1;
```

再运行程序，观察程序输出结果。为什么结果不一样了？

② 将第 8 行和第 9 行改为：

```
printf("x1 = %e,y1 = %e\n",x1,y1);
printf("x1 + y1 = %e\n",x1 + y1);
```

再运行程序，观察程序输出结果。

(4) 输入以下程序，分析程序运行结果。

```
/* 实验 2_4.c */
#include <stdio.h>
int main()
{
    int i,j;
    i = 8;
    j = i++;     /* 第 7 行 */
    printf("i = %d,j = %d\n",i,j);
    return 0;
}
```

程序的运行结果如图 3-6 所示。

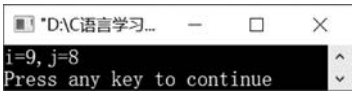


图 3-6　自增运算

思考：

将第 7 行改为：

```
j = ++i;
```

再运行程序，为什么变量 j 的值不一样了？

(5) 输入以下程序，分析程序运行结果。

```
/* 实验 2_5.c */
#include <stdio.h>
int main()
{
    int i;
    unsigned short k = 123;
    float f = 2.67;
    i = f;                          /* 实数赋值给整型变量 */
    printf("i = %d\n",i);
    f = k;
    printf("f = %.2f\n",f);         /* 保留两位小数输出实数 */
    i = -2;
    printf("i = %d,i = %u\n",i,i);
    k = -1;                         /* 第 14 行 */
    printf("k = %d,k = %u\n",k,k);  /* 第 15 行 */
    return 0;
}
```

程序的运行结果如图 3-7 所示。

思考：

① 为什么程序会输出 i=4294967294？

② 为什么程序第15行会输出k=65535?

③ 将第14行改为:

```
k = 65536;
```

再运行程序,程序输出结果是什么?为什么?

(6) 输入以下程序,分析程序运行结果,重点关注指针变量的使用。

```
/* 实验2_6.c */
#include <stdio.h>
int main()
{
    int i,j,*pi,*pj,sum;
    pi = &i;                                  /*指针变量pi指向变量i*/
    pj = &j;                                  /*指针变量pj指向变量j*/
    scanf("%d%d",pi,pj);                      /*输入变量i和j的值*/
    sum = *pi+*pj;                            /*第9行,求变量i和j的和*/
    printf("%d+%d=%d\n",*pi,*pj,sum);         /*第10行,输出整个求和表达式*/
    return 0;
}
```

程序的运行结果如图3-8所示。

图3-7 赋值类型的转换

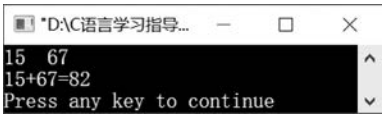


图3-8 求两个整数的和

思考:如果再定义一个指针变量ps,让它指向变量sum,则第9行和第10行中用ps替换sum,程序怎么修改?

2. 设计性实验

(1) 编写程序,从键盘输入一个4位正整数,求其各位上的数字并输出。例如,1234的输出结果为1,2,3,4。

实验提示

① 根据题意,可以先定义5个整型变量,用于保存要处理的4位整数和其各位上的数字。

② 用scanf()函数输入整数,注意scanf()函数的使用规范。

③ 灵活运用除法运算符"/"和求余运算符"%"将其各位上的数字求出来。

④ 用printf()函数输出结果,注意printf ()函数的使用规范。

示例的运行结果如图3-9所示。

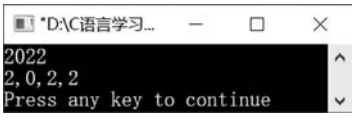


图3-9 输出整数的各位数字

(2) 编写程序,从键盘输入圆的半径,计算并输出圆的周长、面积和相同半径的球的体积(保留两位小数)。要求使用符号常量表示圆周率π的值,π的取值为3.14。

实验提示

① 根据题意,可以定义4个实型变量,分别用于保存圆的半径、周长、面积和球的体积。

② 用宏定义命令#define 定义符号常量 PI 表示圆周率 π 的值。

③ 用 scanf()函数输入半径，注意 scanf()函数中使用正确的格式字符。

④ 计算圆的周长、面积和球的体积(注意算式 4/3 的正确表示)。

⑤ 用 printf()函数输出结果，注意 printf()函数中使用正确的格式字符。

示例的运行结果如图 3-10 所示。

思考：本题若定义只读变量 pi 保存圆周率 π 的值，程序该怎么修改？

(3) 编写程序，键盘输入两个数字字符，分别将它们转换为整数后，计算并输出两个整数的乘积。

实验提示

① 根据题意，定义两个字符变量和两个整型变量。

② 用字符输入函数 getchar()分别输入两个数字字符。

③ 将数字字符转换为整数，方法是让数字字符减去'0'。假设字符变量为 ch，表达式 ch-'0'的值就是转换后的整数。

④ 用 printf()函数输出两个整数的乘积。

示例的运行结果如图 3-11 所示。

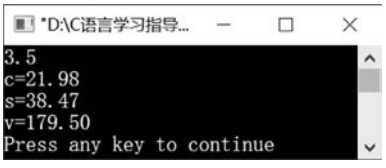


图 3-10　计算并输出圆的周长、面积和球的体积

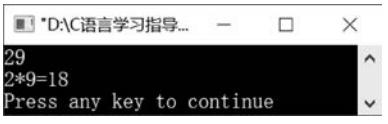


图 3-11　数字字符转换为整数并输出其乘积

(4) 编写程序，键盘输入一个正实数，分别输出该实数的整数部分和小数部分。

实验提示

① 根据题意，定义两个实型变量和一个整型变量。

② 用 scanf()函数输入一个实数。

③ 将实数直接赋值给整数，即可得到实数的整数部分。实数减去整数即可得到该实数的小数部分的值。

④ 用 printf()函数分别输出实数的整数部分和小数部分的值。

示例的运行结果如图 3-12 所示。

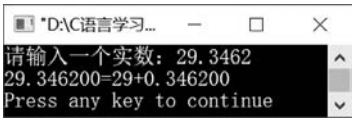


图 3-12　输出实数的整数部分和小数部分的值

3. 提高性实验

(1) 编写程序，输入一个 24 小时制的时间，求出再过 1800 分钟后是 24 小时制的几时几分？

实验提示

① 根据题意，定义两个整型变量，分别保存小时和分钟。

② 总的分钟数/60 为增加的小时数，总的分钟数%60 为剩余的分钟数。

示例的运行结果如图 3-13 所示。

(2) 程序填空，输入三角形三条边的长度，计算并输出三角形的面积(要求用指针变量，结果保留两位小数)。三角形面积的计算公式为$\sqrt{s(s-a)(s-b)(s-c)}$，其中 a，b，c 为三角

形的边长,s=$\frac{a+b+c}{2}$(使用 sqrt(x)可以求出 x 的算术平方根)。

```
/* 实验 2_7.c */
#include <stdio.h>
#include <math.h>
int main()
{
    int a,b,c,;
    float s,area, *ps = &s, *pa = &area;
    printf("请输入三角形的三条边:");
    scanf(_____);
     *ps = _____;
     *pa = _____;   /*求三角形的面积 */
    printf("area = _____\n", *pa);
    return 0;
}
```

示例的运行结果如图 3-14 所示。

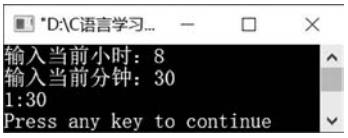


图 3-13　输出 1800 分钟后的时间

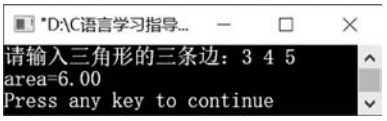


图 3-14　求三角形的面积

实验项目 3　顺序结构程序设计方法

一、实验目的

(1) 掌握 C 语言中使用最多的一种语句——赋值语句的使用方法。

(2) 掌握 C 程序的基本构成,熟悉顺序结构程序设计的一般步骤。

二、实验要求

(1) 理解 C 语言中程序设计的基本思路。

(2) 编写简单的顺序结构程序。

三、实验内容

1. 验证性实验

(1) 很多国家采用华氏温度标准(F)用于温度的计算,而中国则采用的是摄氏温度(C)。根据温度转换的公式设计一个温度转换程序,可以进行温度转换。如果输入摄氏温度,显示转换的华氏温度。运行结果如图 3-15 所示。

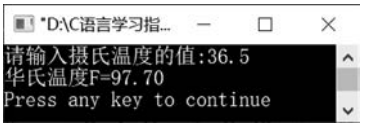


图 3-15　温度转换的运行结果

温度转换的公式为 F=(C×9/5)+32。

实验步骤

① 声明变量 f 和 c。

② 输入 c 的值。

③ 利用温度转换的公式计算 f 的值。

④ 输出 f 的值。

程序代码如下：

```
#include <stdio.h>
int main()
{
    float f,c;
    printf("请输入摄氏温度的值:");
    scanf("%f",&c);
    f = c * 9/5 + 32;
    printf("华氏温度 F = %.2f\n",f);
    return 0;
}
```

采用指针实现的程序代码如下：

```
#include <stdio.h>
int main()
{
    float f,c;
    float *pf = &f, *pc = &c;
    printf("请输入摄氏温度的值:");
    scanf("%f",pc);
    *pf = *pc * 9/5 + 32;
    printf("华氏温度 F = %f\n", *pf);
    return 0;
}
```

思考：如果将语句“f＝c * 9/5＋32;”换成语句“f＝9/5 * c＋32;”输出结果是多少呢?

(2) 从键盘上输入任意两个数，输出它们的和值与差值之积。运行结果如图 3-16 所示。

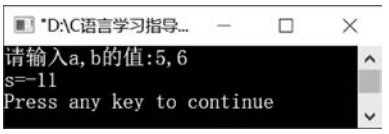


图 3-16　求任意两个数和差之积的运行结果

实验步骤

① 定义 3 个变量 a、b、s。

② 输入 a、b 的值。

③ 计算 a 和 b 值的和值与差值之积，并保存在变量 s 中。

④ 输出 s 的值。

程序代码如下：

```
#include <stdio.h>
int main()
{
    int a,b,s;
    printf("请输入 a,b 的值:");
    scanf("%d,%d",&a,&b);
    s = (a + b) * (a - b);
    printf("s = %d\n",s);
    return 0;
}
```

采用指针实现的代码如下：

```
#include <stdio.h>
int main()
{
    int a,b,s;
    int *pa=&a,*pb=&b,*ps=&s;
    printf("请输入a,b的值:");
    scanf("%d,%d",pa,pb);
    *ps=(*pa+*pb)*(*pa-*pb);
    printf("s=%d\n",*ps);
    return 0;
}
```

思考：能否增加两个变量分别记录两个数之和与差？如果行，改写程序，并说出哪种写法好及其原因。

2. 设计性实验

(1) 编写程序，输入一个小写字母，输出其对应的大写字母。

实验提示

① 声明两个字符型变量，一个用于保存小写字母，一个用于保存大写字母。

② 输入小写字母。

③ 小写字母转换成大写字母。

④ 输出大写字母。

其中，第③步关于大小写字母转换的问题，可以利用字符的 ASCII 码，大写字母 A 的 ASCII 码值为 65，小写字母 a 的 ASCII 码值为 97，即大、小写字母的 ASCII 码的差值为 32。运行结果如图 3-17 所示。

(2) 编写程序，输入两个整数分别赋给变量 a 和 b，然后交换两个变量的值再输出。

实验提示

① 声明两个整型变量 a 和 b。

② 输入 a 和 b 的值。

③ 交换两个变量的值，即变量 a 保存变量 b 的值，变量 b 保存变量 a 的值。

④ 输出变量 a 和 b。

其中，第③步交换两个变量值的问题怎么解决呢？就像有两杯水，现在要把两个杯子中的水交换，那么大家很自然想到利用第三个杯子做中介。同样的道理，要交换两个变量的值，就需要另外一个变量做中介。所以，在第①步需要声明 3 个变量。运行结果如图 3-18 所示。

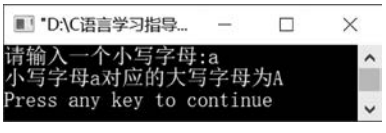


图 3-17　小写字母转换成大写字母的运行结果

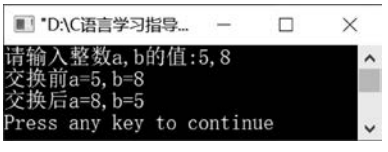


图 3-18　交换两个变量的运行结果

3. 提高性实验

(1) 编写程序，输入存款金额、存款年限和存期的年利率，输出存款到期金额。到期金额＝存款金额×(1＋存期的年利率)^{存款年限}。运行结果如图 3-19 所示。

解题思路

① 声明存款金额、存款年限、存期的年利率、到期金额 4 个变量。

② 输入需要的值。

③ 利用公式计算，其中求 x^n 需要用到库函数 pow(x,n)。库函数 pow(x,n)包含在头文件 math.h 中，因此，在程序开头要添加语句“#include <math.h>”。

④ 输出到期金额。

(2) 编写程序，输入长方形的长和宽，求它的周长和面积。运行结果如图 3-20 所示。

图 3-19　存款到期金额的运行结果

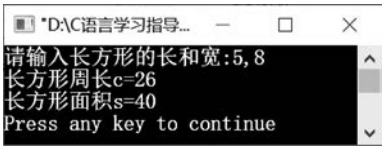


图 3-20　求长方形周长和面积的运行结果

解题思路

① 声明变量。

② 输入需要的值。

③ 利用公式计算长方形的周长和面积。

④ 输出长方形的周长和面积。

(3) 编写程序，求方程 $ax^2+bx+c=0$ 的实数根。其中 a、b、c 由键盘输入，且在输入时保证 $a\neq 0$ 且 $b^2-4ac>0$。运行结果如图 3-21 所示。

解题思路

① 声明 5 个变量 a、b、c、x1、x2，其中 a、b、c 接收从键盘输入的 3 个值，x1 和 x2 用来保存此一元二次方程的两个实数根。

② 输入 a、b、c 的值，同时保证输入的值中 $a\neq 0$ 且 $b^2-4ac>0$。

③ 利用公式计算，其中$\sqrt{x}$需要用到库函数 sqrt(x)。库函数 sqrt(x)包含在头文件 math.h 中，因此，在程序开头要添加语句“#include <math.h>”。

④ 输出两个不相等的实数根。

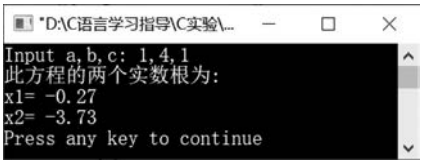


图 3-21　求一元二次方程两个实数根的运行结果

思考：如果任意输入 a、b、c 的值，当 a 为 0 时，当前不是一元二次方程。当 a 不为 0 时，b^2-4ac 可能大于 0，或等于 0，或小于 0，这些情况在程序中如何判断，如何对应输出两个不等的实数根，或相等的实数根或虚数根？

实验项目 4　分支结构程序设计方法

一、实验目的

(1) 掌握 C 程序中 if 语句的格式及使用方法。

(2) 掌握 switch 语句的格式及使用方法。

(3) 使用分支结构编写简单的C语言程序。

(4) 理解分支结构嵌套的格式及使用方法。

二、实验要求

(1) 掌握if和switch语句的不同书写格式。

(2) 熟悉利用if和switch语句编写简单的分支程序。

(3) 理解分支条件的书写方法。

三、实验内容

1. 验证性实验

(1) 输入整数a、b,若a^2+b^2大于100,则输出a^2+b^2百位以上的数字,否则输出两数之和。运行结果如图3-22所示。

实验步骤

分支选择处理的条件是$a^2+b^2>100$。如果大于100,则输出a^2+b^2百位以上的数字;如果小于或等于100,则输出a+b的值。其中,求百位以上的数字,可以根据两个整数相除仍然为整数的原则,使用$(a^2+b^2)/100$求得。

① 首先定义三个整型变量a、b、y,其中y用于保存需要输出的值。

② 使用if…else两分支结构编写代码,其中分支条件为$a^2+b^2>100$。

③ 输出y的值。

程序代码如下:

```
#include <stdio.h>
int main()
{
    int a, b, y;
    printf("enter a, b:");
    scanf("%d, %d",&a, &b);
    if ((a*a+b*b)>100)                    /*第6行*/
      y=(a*a+b*b)/100;                    /*第7行*/
    else
      y=a+b;                              /*第9行*/
    printf("y=%d\n",y);
    return 0;
}
```

思考:

① 程序中第7行语句起什么作用?是否可以使用语句"y=(a^a+b^b)/100"代替?

② 能够使用if的单分支语句实现上面的程序吗?例如,第6行至第9行使用下面的语句代替。

```
y = a+b;
if((a*a+b*b)>100)
  y=(a*a+b*b)/100;
```

③ 分支结构可以用什么运算符实现?改写这个程序。

(2) 高速公路超速处罚规定:在高速公路上行驶的机动车,超出本车道限速的10%,罚款人民币200元,扣3分;如果超出本车道限速的50%,则吊销驾驶证。编写程序实现:输

入两个整型数据，第一个是限速，第二个是当前车速，经过公式计算，屏幕上输出显示相应的信息，不考虑负数。运行结果如图 3-23 所示。

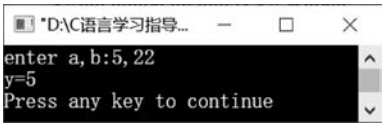


图 3-22　根据 a^2+b^2 是否大于 100 输出不同值的运行结果

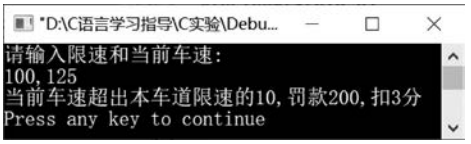


图 3-23　高速公路超速处罚的运行结果

实验步骤

① 定义两个 int 型变量，并从键盘输入，再定义一个 double 型变量 c。

② 利用公式计算当前车速和限速的关系比 c。

③ 若 c 小于 10，则输出“当前未超速”；若大于或等于 10 且小于 50，则输出“当前车速超出本车道限速的 10%，罚款 200，扣 3 分”；若大于或等于 50，则输出“超出 50%，直接吊销驾驶证”。

程序代码如下：

```
#include<stdio.h>
int main()
{
  int a,b;
  double c;
  printf("请输入限速和当前车速:\n");
  scanf("%d,%d",&a,&b);
  c=((double)(b-a)/a)*100;                    /*第 8 行*/
  if(c<10)
    printf("当前未超速\n");
  else if(c<50)                               /*第 11 行*/
       printf("当前车速超出本车道限速的 10%,罚款 200,扣 3 分\n");
  else                                        /*第 13 行*/
       printf("超出 50%,直接吊销驾驶证\n");
  return 0;
}
```

思考：

① 程序第 8 行语句中(double)的作用是什么？

② 第 11 行中条件判断为 c<50，是否可以写成 c>=10&&c<50？

③ 第 13 行中缺省的条件判断是什么？

(3) 使用 switch 语句，根据订单的状态码打印相对应的状态(文字说明)，其对应关系为：1-等待付款；2-等待发货；3-运输中；4-已签收；5-已取消；其他-无法追踪。

实验步骤

① 定义一个整型变量，接受从键盘输入的状态码。

② 根据状态码，输出相对应的状态。

③ 画出流程图。

```
#include<stdio.h>
int main()
```

```
{
    int k;
    printf("请输入状态码:");
    scanf("%d",&k);
    printf("当前状态为:");
    switch(k)
    {
        case 1:
            printf("等待付款\n");
            break;
        case 2:
            printf("等待发货\n");
            break;
        case 3:
            printf("运输中\n");
            break;
        case 4:
            printf("已签收\n");
            break;
        case 5:
            printf("已取消\n");
            break;
        default:
        printf("无法追踪\n");
    }
    return 0;
}
```

程序的运行结果如图 3-24 所示。

思考：

① 加与不加 break 语句对程序的输出结果有何影响？

② 各条 case 语句的先后顺序对输出结果有何影响？

2. 设计性实验

(1) 输入一个字符，如果是小写字母，则输出其对应的大写字母；如果是大写字母，则输出其对应的小写字母；如果是数字，则输出数字本身；如果是空格，则输出“当前字符为空格”；如果不是上述情况，则输出“当前字符为其他字符”。

实验提示：

① 定义一个字符型变量 ch。

② 输入 ch 的值。

③ 根据 ch 的值，进入相应的分支，输出对应的提示信息。

程序的运行结果如图 3-25 所示。

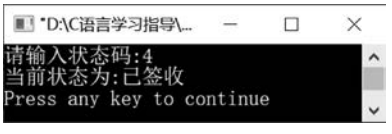


图 3-24　文字状态的运行结果

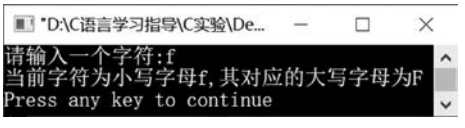
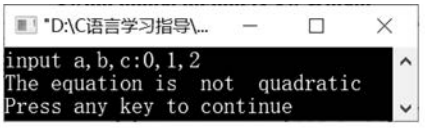
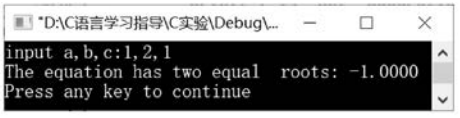


图 3-25　判断字符类型的运行结果

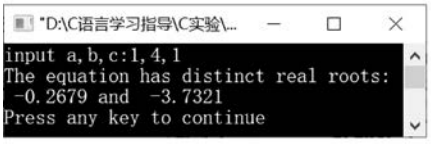
(2) 编写一个程序，求方程 $ax^2+bx+c=0$ 的所有可能的根(包括虚根)。其中 a、b、c 由键盘输入。程序的运行结果如图 3-26 所示。



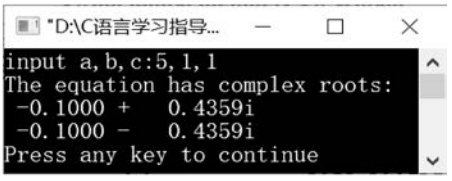
(a) 不是一元二次方程



(b) 有两个相等的实数根



(c) 有两个不等的实数根



(d) 有两个不等的虚数根

图 3-26　求解一元二次方程的运行结果

实验提示

① 根据输入的 a 值进行判断，若 a 为 0，则不是一元二次方程；若 a 不为 0，则继续判断 b^2-4ac 值的大小。

② 若 b^2-4ac 的值大于 0，则有两个不等的实数根；若 b^2-4ac 的值等于 0，则有两个相等的实数根；若 b^2-4ac 的值小于 0，则有两个不等的虚数根。

(3) 已知银行整存整取存款不同期限的月息利率分别为：月息利率＝0.33%，期限为 1 年；月息利率＝0.36%，期限为 2 年；月息利率＝0.39%，期限为 3 年；月息利率＝0.45%，期限为 5 年；月息利率＝0.54%，期限为 8 年。输入存款的本金和年限，使用 switch 语句编程实现求到期时能从银行得到的利息与本金的合计。利息的计算公式：利息＝本金×月息利率×12×存款年限。运行结果如图 3-27 所示。

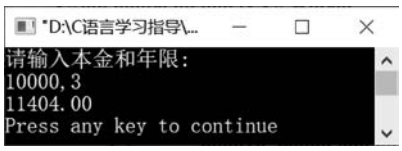


图 3-27　银行存款的运行结果

实验提示

① 输出结果是利息与本金之和，那么只要求出利息值即可。

② 从利息计算公式中可以看出，存款年限和月利息率是息息相关的，其中存款年限为整数，很自然地就想到利用年限值得到相应的 case 分支语句。

③ 假设存款年限用变量 year 表示，那么就可以得到下面的结构。

```
switch(year){
    case 1:
    case 2:
    case 3:
    case 5:
    case 8:
    default:
}
```

(4) 机票的价格受季节旺季、淡季的影响，头等舱和经济舱价格也不同，具体折扣如表 3-1 所示。假设机票原价 5000 元，根据出行的月份和选择的仓位输出实际的机票价格。程序的运行结果如图 3-28 所示。

表 3-1　机票价格折扣表

旺季(4—10月)	头等舱	九折
	经济舱	八折
淡季(11月—次年3月)	头等舱	五折
	经济舱	四折

实验提示

① 输入出行的月份和选择的仓位。

② 根据输入的信息先判断当前选择的仓位,再判断出行的月份是淡季还是旺季,最后计算并输出实际的机票价格。

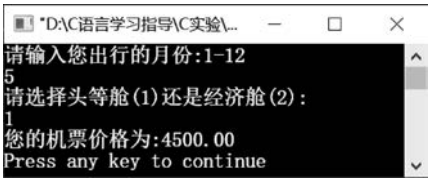


图 3-28　实际机票价格的运行结果

(5) 输入3个正整数,如果其中任一数不是正整数,则程序输出 Invalid number!,并结束运行。当第1个数为奇数时,计算后面两数之和;当第1个数为偶数时,计算第2数减第3数的差。无论哪种情形,当结果超过10时,则输出运算结果,否则什么也不输出。程序的运行结果如图3-29所示。

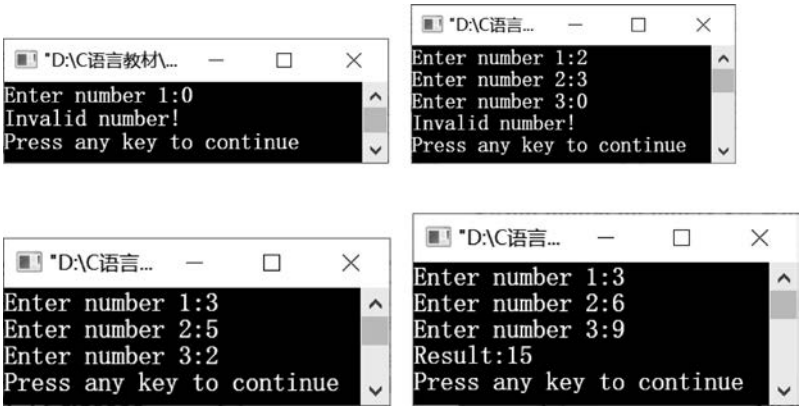


图 3-29　3个正整数的运算结果

实验提示

定义四个整型变量 a、b、c、s,先输入 a 的值,若 a 小于或等于0,则输出 Invalid number!,并结束运行;若 a 为正整数,则继续输入 b 的值,若 b 小于或等于0,则输出 Invalid number!,并结束运行;若 b 为正整数,则继续输入 c 的值,若 c 小于或等于0,则输出 Invalid number!,并结束运行;若 c 为正整数,则判断 a 是奇数还是偶数,若 a 是奇数,则 s=b+c;若 a 是偶数,则 s=b−c。最后根据 s 的值进行判断,若 s 的值大于10,输出 s;否则结束运行。

3. 提高性实验

(1) 编写一个程序实现如下功能:输入一个正整数,判断它能否被3、5、7整除,根据条件输出以下信息。

① 能同时被3、5、7整除。

② 能被其中两个数(要指出哪两个数)整除。

③ 能被其中一个数(要指出哪一个数)整除。

④ 不能被3、5、7任一个整除。

程序的运行结果如图 3-30 所示。

图 3-30　判断能否整除的运行结果

解题思路

① 利用取余运算可以判断一个数是否能够被另一个数整除。

② 本题中还需要输出当前输入的整数能被 3、5、7 中的哪几个数整除。通过对题目要求的理解，实际上可以把题目要求转化为 8 种可能：能同时被 3、5、7 整除；不能被 3、5、7 中任一个整除；能被 3 整除；能被 5 整除；能被 7 整除；能被 3、5 整除；能被 3、7 整除；能被 5、7 整除。

③ 可用 if-else 的嵌套和 switch 语句两种形式分别实现。

(2) 分别输入年、月、日，求出在此年所剩余的天数。

程序的运行结果如图 3-31 所示。

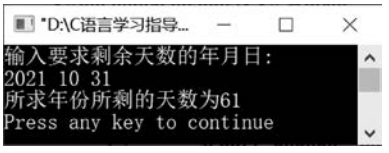


图 3-31　求剩余天数的运行结果

解题思路

① 此题为多分支嵌套结构，先根据月份值进行分支选择。

② 因为月份只有 1～12 这 12 个正整数，因此用 case 语句实现会更简洁。

③ 利用 switch 语句中若无 break 语句，则会顺序求值的特性进行求解，在输入的月份之后，将所有剩余的天数进行相加。

④ 需要注意的是在 2 月份时，还需要判断当前年份是否为闰年。若是闰年，2 月份为 29 天；若是平年，2 月份为 28 天。

实验项目 5　循环结构程序设计方法

一、实验目的

(1) 掌握 3 种循环结构 while、do-while 和 for 的区别与联系，并能正确使用它们。

(2) 掌握与循环语句相关的 break 语句和 continue 语句的使用方法。

(3) 学会使用 C 语言循环结构编写程序。

(4) 理解循环结构嵌套的格式和使用方法。

二、实验要求

(1) 掌握 while、do-while 和 for 语句的不同书写格式。

(2) 熟悉利用 while、do-while 和 for 语句编写简单的循环结构程序。

(3) 掌握 break 语句和 continue 语句的区别。

三、实验内容

1. 验证性实验

(1) 输入一组正整数,以-1 表示结束,分别统计其中包含的奇数和偶数的个数(无须统计最后的-1)。

实验步骤

设整型变量 x 用于表示一个正整数,通过重复(即循环)输入 x 的值即得到一组正整数,并对每次输入的 x 进行三方面的判断,即值为-1、奇数或偶数,以实现相应的数据处理过程,具体可通过以下两种思路(分别见图 3-32 和图 3-33)进行编程实现,其中 nOdd_Cnt 和 nEven_Cnt 为两个计数器,分别用于统计奇数和偶数的个数。

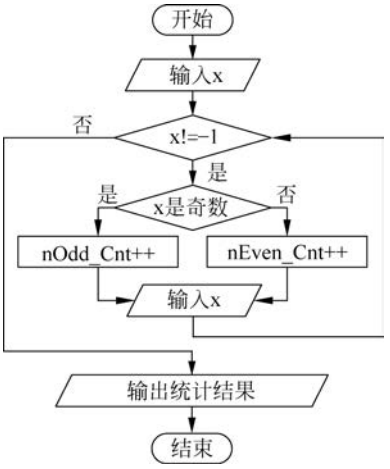


图 3-32 思路一的流程

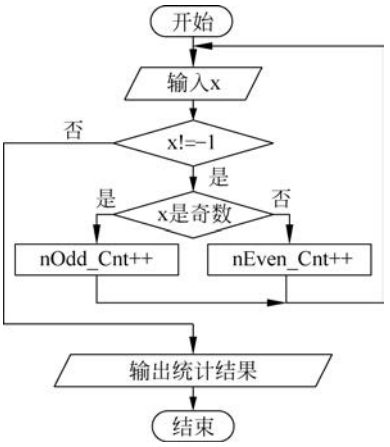


图 3-33 思路二的流程

思路一的程序代码如下:

```
#include <stdio.h>
int main()
{
    int x, nOdd_Cnt = 0, nEven_Cnt = 0;
    printf("请输入一个正整数(-1 表示结束):");
    scanf("%d", &x);
    while(x !=-1)
    {
        x % 2 == 0 ? nEven_Cnt ++: nOdd_Cnt ++;
        printf("请输入一个正整数(-1 表示结束):");
        scanf("%d", &x);
    }
    printf("共有奇数%d个,偶数%d个\n", nOdd_Cnt, nEven_Cnt);
    return 0;
}
```

思路二的程序代码如下:

```
#include <stdio.h>
```

```
int main()
{
    int x, nOdd_Cnt = 0, nEven_Cnt = 0;
    while(1)
    {
        printf("请输入一个正整数(-1表示结束):");
        scanf("%d", &x);
        if(x == -1)
            break;
        x % 2 == 0 ? nEven_Cnt ++ : nOdd_Cnt ++;
    }
    printf("共有奇数%d个,偶数%d个\n", nOdd_Cnt, nEven_Cnt);
    return 0;
}
```

可见,while 循环中的循环条件(见思路一)有时能起到与 break 语句(见思路二)类似的效果,但使用 break 语句会使程序在具体思路上有细微区别,同时,思路二中的 while(1)又类似 do-while 语句形式,与 break 相结合后,实现了先做操作,再进行判断的循环效果。通过两种思路实现的程序运行结果相同,如图 3-34 所示。

(2) 编程以判断一个整数是不是水仙花数,其中水仙花数的定义为各数位上数字的立方和等于整数自身,如 $153=1^3+5^3+3^3$。

实验步骤

对外部输入的一个整数 n,通过对每个数位上数字 x 的 3 次方求累加和,将该累加和与原数据 n 进行比较,当相等时则表示该整数是一个水仙花数。而对于其中各数位上的数字 x,可使用 n%10 求得当前个位数字,再紧跟着使用 n/=10 的操作,将当前个位数去除,使得原来的十位数成为新数字的个位数,继续重复以上两个操作以完成各数位数字的计算,直到 n 的值变为 0 为止,具体流程如图 3-35 所示。

图 3-34 统计一组正整数中奇数和偶数个数的运行结果

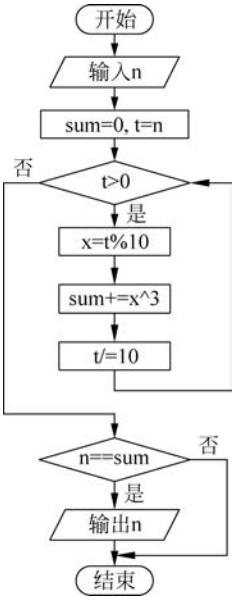


图 3-35 水仙花数的计算流程

程序代码如下：

```
#include <stdio.h>
int main()
{
    int sum, n, t, x;
    sum = 0;
    printf("请输入一个三位正整数:");
    scanf("%d", &n);
    t = n;
    while(t > 0)
    {
        x = t % 10;
        sum += x * x * x;
        t /= 10;
    }
    if(sum == n)
        printf("%d是水仙花数\n", n);
    else
        printf("%d不是水仙花数\n", n);
    return 0;
}
```

程序的运行结果如图3-36所示。

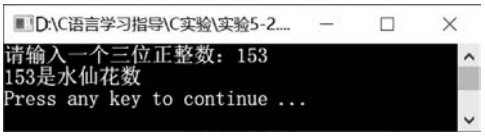


图3-36　水仙花数的计算结果

思考：在上述程序中，需要通过语句“t =n;”对n的值进行备份，如果没有该语句，直接将所有t的位置换为n，那么在后续的判断语句“if(sum==n);”中n的值还是原来的值吗？或者又是多少呢？

(3) 编写程序，计算并输出表达式a+aa+aaa+…+aaa…a(n个a)的值。其中，a和n均为由键盘输入的正整数，且要求a为1～9的单个数字，如a=2，n=3时求得结果为246(即2+22+222的值)。

实验步骤

首先将整个问题分析为一个累加和的问题，而对于每一个累加项通过观察可以发现，新的一项是前一项的10倍再加上a的值，因此得到的累加项可以通过语句“x=x*10+a”来进行迭代求取，且可知初始时的x值为0，具体流程如图3-37所示。

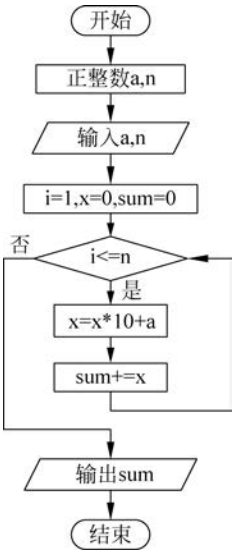


图3-37　表达式的计算流程

程序代码如下：

```
#include <stdio.h>
int main()
{
```

```
    int sum, a,n,i, x;
    printf("请输入 a(1－9 以内)和 n 的值,以逗号隔开:");
    scanf("%d,%d", &a, &n);
    sum = x = 0;
    for(i = 1; i <= n; i ++)
    {
        x = x * 10 + a;
        sum += x;
    }
    printf("sum = %d\n", sum);
    return 0;
}
```

程序的运行结果如图 3-38 所示。比较验证性实验(2)和验证性实验(3)中的流程和程序代码,其中均有求累加和部分,虽在不同的问题中,但可以看出实现累加问题的程序实现中存在先初始化为 0,再通过循环语句进行逐个累加等共性代码。

(4) 使用迭代法求 a 的平方根,其中迭代公式为 $x_{n+1}=(x_n+a/x_n)/2$。假定 x_0 的初值为 a,迭代到 $|x_{n+1}-x_n|<10^{-5}$ 时为止,此时的 x_{n+1} 就代表 a 的平方根,可根据输入的 a 值,对计算结果进行验证,如输入 4 时,计算结果为 2。

实验步骤

迭代法又称为递推法,其主要思路是在给定起始项 x_0 后,根据递推公式依次递推后续各项,直到相邻两项间的精度达到要求精度为止。

程序代码如下:

```
#include <stdio.h>
#include <math.h>
int main()
{
    float xn, a, xn1;
    printf("请输入要计算平方根的值:");
    scanf("%f", &a);
    xn = xn1 = a;
    do
    {
        xn = xn1;
        xn1 = (xn + a / xn) / 2;
    }
    while(fabs(xn1 - xn) > 1e-5);
    printf("xn1 = %f\n", xn1);
    return 0;
}
```

程序的运行结果如图 3-39 所示。

图 3-38　表达式的值计算结果

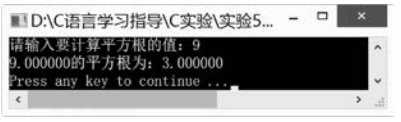


图 3-39　平方根的计算结果

2. 设计性实验

(1) 采用 do-while 循环语句编程以计算 π 的近似值。其中的计算依据为 $\pi^2/6=1/1^2+$

$1/2^2+1/3^2+\cdots$,计算精度要求直到某一累加项的值小于 10^{-6} 为止(说明:在实验时可自行调整该精度值,如调整为 10^{-10},并观察得到结果的变化)。

实验提示

① 使用 do-while 循环语句计算 $1/1^2+1/2^2+1/3^2+\cdots+1/n^2$ 的值,该过程为一个累加和的求取过程,且循环条件根据 $1/n^2$ 大于或等于 10^{-6} 进行设置,并将最终的值记为 s。

② 将 s * 6 记为 t,可知 t 的值相当于 π^2。

③ 通过 t 的开平方,即可得到 π 的值,具体实验的运行结果可参考图 3-40 和图 3-41。

图 3-40 π 的近似值(精度为 10^{-6})

图 3-41 π 的近似值(精度为 10^{-10})

(2) 编写程序,计算并输出 100～600 满足以下条件的数:每个数位上数字的积为 45 且和为 15,同时统计并输出满足以上条件的数的个数。

实验提示

① 使用 for 循环语句控制变量 i 的值从 100 依次以 1 递增到 600,并对其中的每一个 i 计算各数位上数字的乘积 product、数位的和 sum。

② 通过语句"if(sum==15 && product==45)"来判断该数是否满足题目要求的条件。

③ 在满足条件时,则输出该 i 的值,并通过计数器进行计数,最后在 for 语句结束后输出计数器的值,具体实验的运行结果可参考图 3-42。

(3) 编写程序,计算函数 $y=(5*\sin(x)+x-3.6*\cos(x))^2$ 的最小值,其中自变量 x 的取值为闭区间[-50,50]中的所有整数,要求输出取最小值时的 x 和 y 值。程序的运行结果如图 3-43 所示。

图 3-42 设计性实验(2)的运行结果

图 3-43 设计性实验(3)的运行结果

实验提示

① 令 xMin 为-50,根据函数定义计算 yMin 值作为当前最小的函数值。

② 使用 for 循环语句控制变量 x 的值从-49 依次以 1 递增到 50。

③ 对每一个 x,计算对应的 y 值并使用语句"if(yMin>y)"来判断当前的 yMin 值是否大于新计算的 y 值,若大于则令 yMin=y,xMin=x 记录当前的函数最小值及对应的 x。

④ 运行步骤③直到 for 语句结束,输出最终的 xMin 和 yMin。

(4) 编写程序,根据输入的两个不同的正整数(设为 a 和 b),计算并输出它们的最大公约数和最小公倍数。具体实验的运行结果可参考图 3-44。

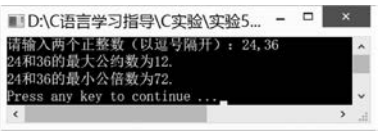


图 3-44 设计性实验(4)的运行结果

实验提示

整个过程可利用辗转相除法进行，具体流程如下。

① 通过 ta＝a，tb＝b 对两个原始的正整数进行备份。

② 根据 c＝ta%tb 计算 c 的值。

③ 若 c＝0，则转到步骤⑤，否则执行步骤④。

④ 执行 ta＝tb，tb＝c，再转到步骤②。

⑤ 返回 tb 为 a 和 b 的最大公约数，a * b/tb 为 a 和 b 的最小公倍数。

3. 提高性实验

(1) 有一筐桃子，第一天被卖掉一半，被吃了一个；第二天又被卖掉剩下的一半，被吃了一个；第三天、第四天、第五天都天天如此，到第六天发现筐里就剩一个桃子了。编写程序，计算最初筐里一共有多少个桃子。程序的运行结果如图 3-45 所示。

(2) 编写程序，计算一个正整数反向后的正整数。例如：输入 12345，则反向后的正整数应为 54321。程序的运行结果如图 3-46 所示。

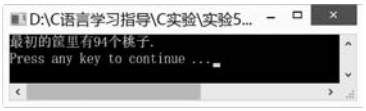


图 3-45　桃子数量计算的运行结果

图 3-46　提高性实验(2)的运行结果

实验项目 6　分支与循环结构综合程序设计

一、实验目的

(1) 掌握 C 语言中的程序控制结构。

(2) 掌握常见的循环结构和分支结构混合使用的程序设计方法。

(3) 学会使用综合程序设计方法实现一些典型的算法及解决一些实际问题。

二、实验要求

(1) 熟悉综合程序设计方法，会编写稍微复杂的程序。

(2) 能够掌握程序的运行步骤，尤其是循环结构程序。

(3) 能够在程序中正确使用 break 语句和 continue 语句。

三、实验内容

1. 验证性实验

(1) 编写程序，根据输入的行数 n，打印如图 3-47 所示的图形。

图 3-47　输出图形的运行结果

实验步骤

分析图案可知，每一行由空格、星号和回车三部分字符组成，而且在行数 n 等于 4 的情况下，这三部分的数量分别如下：

行　数	空　格　数	星　号　数	回　车　数
1	3	1	1
2	2	3	1
3	1	5	1
4	0	7	1

通过总结归纳可知,在一共有n行的图形中,第i行由n－i个空格、2＊i－1个星号、1个回车组成,因此,大致的伪代码描述如下:

```
for(i = 1; i <= n; i ++)                    //表示输出n行
{
    //输出n－i个空格
    //输出2＊i－1个星号
    //输出1个回车
}
```

而输出n－i个空格又可以通过对输出1个空格重复n－i次来实现,进一步形成一个双重循环语句,其中外层循环用于控制图形的行数,而内层循环则分别用来输出第i行中的三部分字符。

程序代码如下:

```
#include <stdio.h>
int main()
{
    int n, i, j;
    printf("请输入图形的行数:");
    scanf("%d", &n);
    for(i = 1; i <= n; i ++)
    {
        for(j = 1; j <= n - i; j ++)            //输出n－i个空格
            printf(" ");
        for(j = 1; j <= 2 * i - 1; j ++)        //输出2＊i－1个星号
            printf("*");
        printf("\n");                           //输出1个回车符
    }
    return 0;
}
```

思考:如何修改以上代码,使其能输出如图3-48、图3-49和图3-50中所示的图形呢?(提示:前面的程序是按第1行到第n行输出,可尝试从第n行到第1行倒序输出并观察结果,另外,原来的字符都是'＊',可以对其进行规律性变化,如初始字符为'1'时,可通过'1'＋i的方式依次得到字符'2'、'3'、'4'……)。

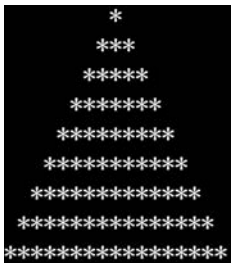


图3-48　图形一

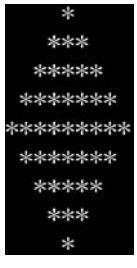


图3-49　图形二

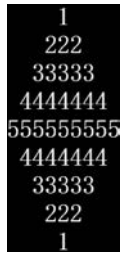


图3-50　图形三

(2) 连续输入以回车结束的多个字符，分别统计出其中的英文字母、空格、数字和其他字符的个数(不含最后的回车)。程序的运行结果如图 3-51 所示。

实验步骤

可以使用语句“while((ch=getchar())!='\n')”经典描述，并在输入过程中，对输入的字符 ch 依次进行检查，并根据检查结果进行统计以得到相应结果。

程序代码如下：

```
#include <stdio.h>
int main()
{
    int nLet, nSpa, nDig, nOth;
    char ch;
    nLet = nSpa = nDig = nOth = 0;
    printf("请输入多个字符,以回车键结束:");
    while((ch = getchar()) != '\n')
    {
        if((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))
            nLet ++;
        else if(ch == ' ')
            nSpa++;
        else if(ch >= '0' && ch <= '9')
            nDig++;
        else
            nOth++;
    }
    printf("字母 %d 个,空格 %d 个,数字 %d 个和其他字符 %d 个\n", nLet, nSpa, nDig, nOth);
    return 0;
}
```

(3) 编程求解鸡兔同笼问题：有若干鸡和兔同在一个笼子里，且共有 35 个头和 94 只脚，问笼中各有多少只鸡和兔？程序的运行结果如图 3-52 所示。

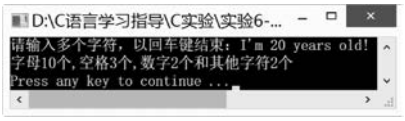


图 3-51　字符统计的运行结果

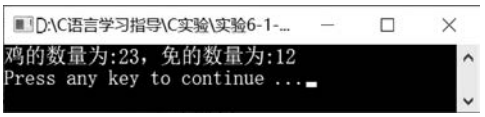


图 3-52　鸡兔同笼问题的求解结果

实验步骤

可以使用 x 和 y 分别用于计算鸡和兔的数量，可知 x 和 y 的取值都在 35 以内，且满足条件 x+y=35 和 2x+4y=94。可通过对 x 和 y 进行穷举的方式进行验证，并输出满足以上条件的 x 和 y 值作为最终的结果。

程序代码如下：

```
#include <stdio.h>
int main()
{
    int x, y;
    for(x = 0; x <= 35; x ++)
    {
        for(y = 0; y <= 35; y ++)
```

```
            if(x + y == 35 && 2 * x + 4 * y == 94)
                printf("鸡的数量为:%d,兔的数量为:%d\n", x, y);
        }
        return 0;
    }
```

(4) 编程计算以下表达式中 s 的值：s=1+(1+2)+(1+2+3)+…+(1+2+3+…+n),其中 n 为外部输入的正整数。程序的运行结果如图 3-53 所示。

实验步骤

整个问题为一个累加过程,而其中每个累加项自身又是一个累加的子过程,所以可以通过两个类似的累加循环进行嵌套,以实现 s 的计算。

程序代码如下：

```
#include <stdio.h>
int main()
{
    int n, i,j, s = 0, sTemp;
    printf("请输入正整数 n 的值:");
    scanf("%d", &n);
    for(i = 1; i <= n; i ++)
    {
        sTemp = 0;
        for(j = 1; j <= i; j ++)
            sTemp += j;
        s += sTemp;
    }
    printf("s = %d\n", s);
    return 0;
}
```

思考：上述实验中(3)和(4)对应的程序都使用了循环嵌套,能否对当前的程序进行修改,使得每个程序中均只使用一层循环(即不使用循环嵌套)即可实现当前题目中所要求的功能?

2. 设计性实验

(1) 编写程序,计算并输出 e 的值,其中依据的计算公式为：

$$e \approx 1+\frac{1}{1!}+\frac{1}{2!}+\cdots+\frac{1}{n!}$$

要求累加到某累加项的值小于或等于 10^{-6} 为止。程序的运行结果如图 3-54 所示。

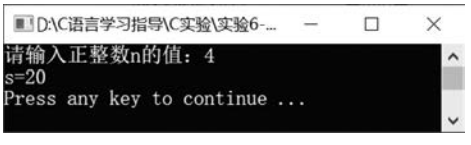


图 3-53　累加问题的计算结果

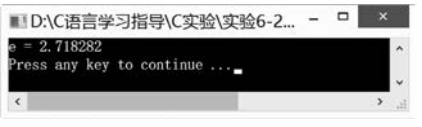


图 3-54　计算 e 的值运行结果

实验提示

① 从整体上看,计算过程为一个求累加和 e 的过程,可通过常规的累加和描述,在 e=1 的前提下,对 e+=1/Xi 进行循环来实现,其中 Xi 表示 i!(其计算见步骤②),同时循环条件可设置为 1(即永远为真),只在累加前判断 1/Xi 是否超过 10^{-6},若超过则直接终止整个累

加循环。

② 观察即可知，i!属于累乘问题，可通过常规的累积运算进行代码实现，在 Xi＝1 的前提下，使用循环语句依次累乘小于或等于 i 的正整数。

③ 通过步骤①和步骤②形成的双层嵌套循环计算出 e 值，在循环结束后输出该值即可。

(2) 编写程序，计算并输出 3～500 的所有素数，要求每行输出 12 个数。

实验提示

① 使用循环语句，控制变量 i 的取值为从 3 到 500 以 1 为递增的所有整数，使用步骤②和步骤③来判断 i 是否为素数，若为素数则输出该数，同时在输出时对输出的数进行计数，当输出的数的个数为 12 的倍数时，输出一个换行符，以达到每行输出 12 个数的效果。

② 集中判断变量 i 是否为素数，根据素数的定义，使用循环语句依次对 2 到 i－1 的整数 x，判断 i％x＝＝0 是否成立，若成立则直接终止该循环，否则继续判断直到循环结束，并根据步骤③得出相应的结论。

③ 步骤②中的循环结束后，判断 x＜＝i－1 是否成立，若成立则说明该循环中途终止过，即步骤②中的 i％x＝＝0 成立所引起的终止，对应得到结论为 i 不是素数，否则表示 i％x＝＝0 一直没成立过，即所有的数都不整除，得到结论为 i 是素数。程序的运行结果如图 3-55 所示。

(3) 为验证哥德巴赫猜想(任何一个大于或等于 6 的偶数都可以表示为两个素数之和。例如：6＝3＋3，8＝3＋5，…，100＝3＋97)，编写程序，将 6～100 的所有偶数都表示成两个素数之和，要求每行输出 5 个等式。程序的运行结果如图 3-56 所示。

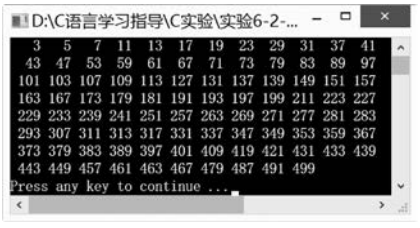


图 3-55　3～500 的所有素数

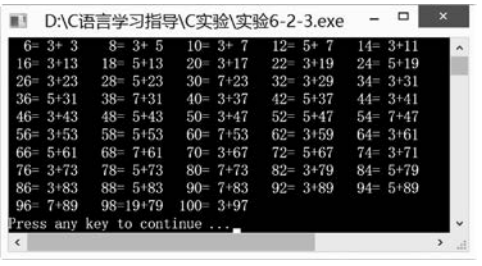


图 3-56　大于或等于 6 的偶数表示为两个素数之和的运行结果

实验提示

① 使用循环语句，控制变量 i 的取值为从 6 到 100 以 2 为递增的所有偶数，接下来判断将 i 分成的两个数是否都为素数，设第一个为 n，则第二个为 i－n，又可以知道 n 和 i－n 都不能为偶数，因为偶数肯定不是素数，所以它们可能的取值为从 3 开始，在小于或等于 i/2 的范围内，以 2 为递增的奇数，因此又需要第二层循环来表示在变量 i 下，n 和 i－n 的取值情况，如 n 的取值可用类似 for(n＝3；n＜＝i/2；n＋＝2)的语句来描述。

② 在步骤①的第二层循环体中，分别以设计性实验(2)中提示步骤②和步骤③所描述的类似过程，先后通过两个循环语句来判断 n 和 i－n 是否为素数，并且若判断发现 n 已不是素数时，则直接跳过 i－n 的判断过程。

③ 综合步骤②的判断结果，只有当 n 和 i－n 都为素数时，才将 n 和 i－n 作为结果输出，并在输出结果时直接进行计数，当计数器为 5 的倍数时，则输出一个换行符。注意，由于对同一个 i 可能得到多个满足的 n 和 i－n 组合，但只要一组即代表题目中的猜想已得到验

证,所以立即进入下一个i的值并继续验证。

(4) 编写程序,找出100～999(含100和999)的所有水仙花数,其中水仙花数的定义为各数位上数字的立方和等于整数自身,如$153=1^3+5^3+3^3$。程序的运行结果如图3-57所示。

实验提示

① 使用循环语句,控制变量i的取值为100～999,接下来使用步骤②判断i是否为水仙花数。

② 根据水仙花数的定义,即每个数位上数字的立方和等于这个数本身,具体判断i是否为水仙花数的过程可参考实验项目5中的验证性实验(2)。

③ 综合步骤②的判断结果,当i是水仙花数时,则输出当前i的值。

3. 提高性实验

(1) 编写程序,求满足条件$1^2+2^2+3^2+\cdots+n^2<=1000$中最大的n,程序的运行结果如图3-58所示。

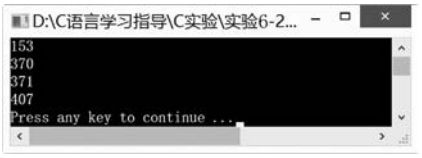


图3-57　100～999的水仙花数

图3-58　提高实验(1)的运行结果

(2) 编写程序,分别输出如图3-59和图3-60所示的九九乘法表。

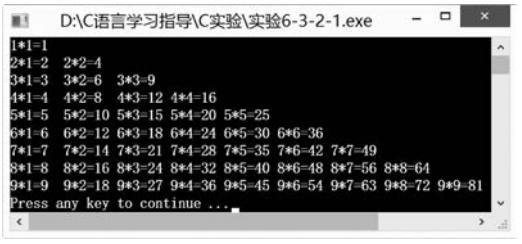


图3-59　九九乘法表一

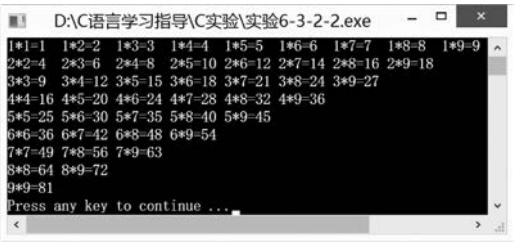


图3-60　九九乘法表二

实验项目7　一维数组程序设计

一、实验目的

(1) 掌握一维数组的定义、赋值和输入输出方法。

(2) 理解一维数组定义时各部分所代表的意义,并能正确引用该数组元素。

(3) 熟悉和掌握与一维数组有关的常用算法,如查找、排序等。

二、实验要求

(1) 理解C语言中数组的作用及应用特点。

(2) 掌握一维数组元素的引用及有序性特点。

(3) 理解一维数组的实际应用。

三、实验内容

1. 验证性实验

(1) 输入并执行如下程序,结合程序的运行结果分析程序的含义和功能。

```
/* 实验 7_1.C */
#include <stdio.h>
int main()
{
    int i, a[5], t;
    printf("Please input 5 numbers:\n");
    for(i = 0; i < 5; i++)
        scanf("%d", &a[i]);
    t = a[0];
    for(i = 0; i < 5; i++)
        if(t < a[i])
            t = a[i];
    for(i = 0; i < 5; i++)
        printf("%d\t", a[i]);
    printf("\nt = %d\n", t);
    return 0;
}
```

实验步骤

① 运行程序,并观察程序的运行结果,如图 3-61 所示。

图 3-61 一维数组的输入和输出

② 将程序中的 t<a[i]改为 t>a[i],再输入相同的数据并观察程序的运行结果。

③ 注意循环变量与下标变量的结合。

思考:

① 将程序中某个循环体的语句 i=0 改为 i=1,则如何修改该循环体的其他部分,使得整个循环的意义不变?

② 若将循环中的 i<5 改为 i<=5,观察发生的情况,为何会发生该情况?

(2) 输入 10 个学生成绩,统计并输出不及格和及格的学生人数。

实验步骤

① 先定义数组,同时定义两个用于计数的变量(设为 m 和 n),并初始化为 0,分别用于统计不及格和及格的学生人数。

② 用循环语句输入 10 个学生成绩到数组中。

③ 使用循环依次比较并统计不及格和及格的学生人数。

④ 输出相应的统计结果。

程序代码如下:

```
/* 实验 7_2.C */
#include <stdio.h>
int main()
{
    int score[10], m = 0, n = 0, i;
```

```
    printf("Input 10 scores:\n");
    for(i = 0; i < 10; i++)
        scanf("%d", &score[i]);
    for(i = 0; i < 10; i++)
    {
        if(score[i] < 60)
            m++;
        else
            n++;
    }
    printf("Failed: %d.\nNot Failed: %d.\n", m, n);
    return 0;
}
```

程序的运行结果如图 3-62 所示。

图 3-62　数组元素信息的统计

思考：

① 将程序改成对分数的多个级别(如 60～70 为及格，71～80 为良好等)统计，则该如何进行？

② 程序中的 if 语句若改为使用“>=”进行比较，则该如何进行？

③ 程序中语句 m=0 只写为 m，则会有什么变化？

2. 设计性实验

(1) 输入 10 个整数到数组 a 中，将它们按逆序保存并输出。

实验提示

① 用循环语句输入 10 个整数到数组中。

② 用循环语句完成以下过程：将第 1 个元素与倒数第 1 个元素互换位置，将第 2 个元素与倒数第 2 个元素互换位置，一直到最中间的元素。

③ 使用循环依次输出互换结束后的数组元素，具体实验的运行结果可参考图 3-63。

(2) 编写程序，根据函数 $y=x^2-8x+\sin x$ 计算 x=1,2,3,…,10 时的函数值 y，并计算和输出这些 y 值中的极小值(保留两位小数)及对应的 x 取值。

实验提示

① 用循环语句初始化包含 10 个元素的数组 X。

② 用循环语句根据函数表达式计算包含 10 个元素的数组 Y。

③ 使用依次比较并更新当前最小值的方法，通过循环得到数组 Y 中的最小值及对应的 X 中的元素。

④ 输出对应的结果，具体实验的运行结果可参考图 3-64。

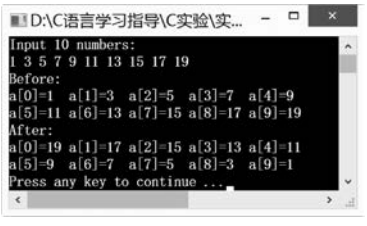


图 3-63　数组元素逆序保存的结果

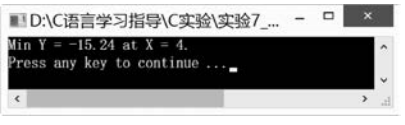


图 3-64　数组元素赋值和极值的计算

3. 提高实验

(1) 输入 10 个整数,删除其中的负数后输出剩下的数据。

解题思路

先定义一个一维数组用以保存输入的 10 个整数且令数据总个数为 10,依次扫描数组的每个元素,并在扫描的同时检测当前元素是否为负数,若是则删除该元素且数据总个数减 1,否则继续扫描并检测下一个元素。删除一个元素的过程为:将当前元素后面的所有元素都向前移动一个位置,此时当前元素会被其后面的第一个元素所覆盖,而其后面其他元素的相对位置均未发生变化。具体实验的运行结果可参考图 3-65。

(2) 用一维数组解决以下问题:输入两个各含有 10 个数据(整型)的数据序列,求取它们共有的数据,并对共有数据进行升序排序后输出。如序列 A 的元素为 2,5,4,1,6,11,10,9,3,7,序列 B 的元素为 9,10,5,15,1,11,13,7,8,3,则输出结果为 1,3,5,7,9,10,11。

解题思路

将序列 A 中的元素逐个取出,与序列 B 中的每个元素进行比较,若遇到相等的元素,即添加到序列 C,等所有相等的元素均已取出后,再对序列 C 进行升序排序。具体实验的运行结果可参考图 3-66。

图 3-65 删除数组中的负数

图 3-66 查找两个数组中的公共元素

实验项目 8 二维数组程序设计

一、实验目的

(1) 掌握二维数组的定义、赋值和输入输出的方法。

(2) 掌握二维数组元素的引用及其在内存中的存储特点。

(3) 掌握与矩阵相关的算法,如矩阵转置、对角线元素求和等。

二、实验要求

(1) 理解 C 语言中二维数组与一维数组的异同。

(2) 掌握二维数组元素的引用及其规律。

(3) 理解二维数组的实际应用。

三、实验内容

1. 验证性实验

(1) 输入并执行下列程序,结合程序的运行结果分析程序的含义与功能。

```
/* 实验 8_1.C */
```

```
#include <stdio.h>
int main()
{
    int i,j,sum = 0;
    int a[6][6];
    for(i = 0; i < 6; i++)
        for(j = 0; j < 6; j++)
            a[i][j] = i * 10 + j;
    for(i = 0; i < 6; i++)
    {
        for(j = 0; j < 6; j++)
            printf("%d\t", a[i][j]);
        printf("\n");
    }
    for(i = 0; i < 6; i++)
        sum += a[i][6 - i - 1];
    printf("sum = %d\n", sum);
        return 0;
}
```

实验步骤

① 程序运行后,观察程序的运行结果,如图 3-67 所示。

② 注意循环变量与下标变量的结合。

③ 将语句“printf("\n");”删除再运行程序,通过对照程序的运行结果分析程序的功能。

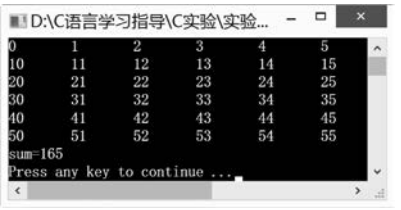


图 3-67 二维数组中的对角线元素求和

思考:

① 如果将程序中的 6-i-1 修改为 i,该程序的功能如何?

② 语句:

```
for(i = 0; i < 6; i++)
{
    for(j = 0; j < 6; j++)
        printf("%d\t", a[i][j]);
    printf("\n");
}
```

和语句:

```
for(j = 0; j < 6; j++)
{
    for(i = 0; i < 6; i++)
        printf("%d\t", a[i][j]);
    printf("\n");
}
```

的功能有什么异同点?

(2) 补充以下程序,使其可以实现:统计 3×5 的矩阵中正数、负数和零的个数。

```
/* 实验 8_2.C */
#include <stdio.h>
int main()
{
```

```
    int a[3][5], i, j, positive, negative, zero;
    for(i = 0; i < 3; i++)
        for(j = 0; j < 5; j++)
            scanf("%d", ____(1)____);
    ____(2)_____;
    for(i = 0; i < 3; i++)
        for(j = 0; j < 5; j++)
            if(____(3)____)
                positive++;
            else if(a[i][j]< 0)
                negtive++;
            ____(4)____
                zero++;
        printf("正数有%d个,负数有%d个,零有%d个\n", positive, negative, zero);
    return 0;
}
```

实验步骤

① 通过观察可知,程序中变量 positive,negative,zero 分别代表正数、负数及零的个数,起计数器的作用。

② 程序的主要步骤分为三步:输入、统计和输出。

③ 首先通过二维数组元素的引用来完善第 1 个填空,再通过使用计数器的统计过程完善第 2 个填空和第 3 个填空,第 4 个填空则为选择逻辑。具体实验的运行结果可参考图 3-68。

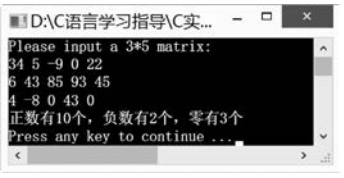


图 3-68　二维数组中特征元素个数的统计

思考:

① 能否修改整个程序,使得只用 positive 和 negative 两个计数器变量就可以完成程序的功能?

② 如果将程序中的以下循环:

```
for(i = 0; i < 3; i++)
    for(j = 0; j < 5; j++)
```

修改为:

```
for(i = 0; i < 5; i++)
    for(j = 0; j < 3; j++)
```

程序的其他部分将会有什么变化?

2. 设计性实验

(1) 求 4×4 二维数组中主对角线以下(包括主对角线)的元素之和。

实验提示

① 用双重循环语句输入 16 个数据到二维数组中。

② 将表示和的变量初值设置为 0。

③ 用循环语句计算主对角线以下(包括主对角线)的元素之和,通过观察可以发现,设元素描述为 a[i][j]时,要求和的元素为满足 i >=j 的元素。

④ 使用循环累加完成计算元素之和的过程,并将累加和输出。具体实验的运行结果可参考图 3-69。

(2) 输入一个 3×4 的矩阵,计算并输出该矩阵的转置矩阵。

实验提示

① 定义一个 3×4 的二维数组表示原始矩阵 a,一个 4×3 的二维数组表示转置矩阵 b。

② 用双重循环语句输入原始二维数组中的数据。

③ 通过转置矩阵与原始矩阵的关系(b[i][j]=a[j][i])对转置矩阵进行赋值。

④ 输出得到的转置矩阵。具体实验的运行结果可参考图 3-70。

图 3-69 矩阵特征区域求和

图 3-70 计算转置矩阵

3. 提高性实验

(1) 输入 3 个学生 5 门课的成绩,计算并输出每门课的最高分及取得最高分的学生编号。

解题思路

3 个学生 5 门课程的成绩如图 3-71 所示,这些成绩可以使用对应结构的 3×5 的二维数组来进行保存,并对该二维数组进行操作来完成实验要求。

学生编号	课程编号				
	0	1	2	3	4
0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

图 3-71 二维数组元素和 3 个学生 5 门课成绩数据的对应关系

可以看出,第 1 门课的最高分即第 1 列元素(即 a[0][0]、a[1][0]和 a[2][0])的最大值,第 2 门课的最高分即第 2 列元素(即 a[0][1]、a[1][1]和 a[2][1])的最大值,以此类推,可以发现求第 j(设 j 从 0 开始)门课最大值的过程为计算 a[i][j]的最大值,其中 i 的取值范围为从 0 到 2。最后通过整理可知,要求得所有 5 门课的最大值只需将 j 从 0 取到 4,其中 j=0 时,计算的是第 1 门课的最高分;j=1 时,计算的是第 2 门课的最高分,以此类推,一直到 j=4 时,计算的是第 5 门课的最高分。具体实验的运行结果可参考图 3-72。

(2) 任意输入两个 4×4 的矩阵 A 和 B,计算并输出 A 与 B 的乘积矩阵。

解题思路

4×4 矩阵的乘积公式如下:

$$C = A \times B = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix}$$

其中，矩阵 C 中的任意一个元素可由以下公式计算得到：

$$c_{if} = a_{i0} \times b_{0j} + a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + a_{i3} \times b_{3j}$$

具体实验的运行结果可参考图 3-73。

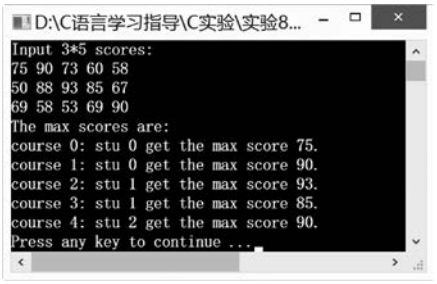


图 3-72　使用二维数组进行成绩统计

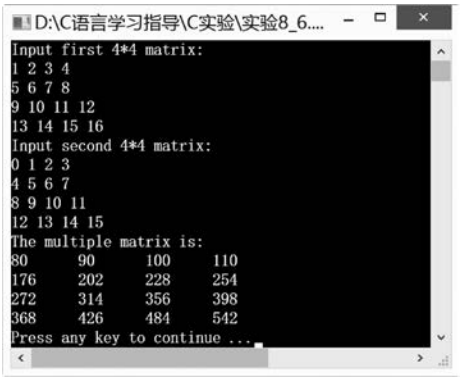


图 3-73　计算矩阵的乘积

实验项目 9　字符数组程序设计

一、实验目的

（1）掌握字符数组的定义和初始化方法。

（2）掌握字符数组的输入、输出方法。

（3）掌握字符串的存取方法和字符串函数的使用。

二、实验要求

（1）理解 C 语言中字符数组与一般数组的异同。

（2）掌握字符串在计算机中的表示方式及引用时的注意事项。

（3）理解字符数组的实际应用。

三、实验内容

1. 验证性实验

（1）输入并执行以下两个程序，结合程序的运行结果理解程序的含义。

程序一：

```
/* 实验 9_1.C */
#include <stdio.h>
int main()
{
        int i = 0;
```

```
        char str[80];
        while((str[i++] = getchar() ) != '?')      /* 第6行 */
            ;
        str[i-1] = '\0';
        puts(str);
    return 0;
}
```

程序二：

```
/* 实验9_2.C */
#include <stdio.h>
int main()
{
    int i = 0, j, count = 0;
    char str1[20], str2[ ] = { 'a', 'e', 'i', 'o', 'u' };
    while((str1[i++] = getchar()) != '#')
        ;
    str1[i-1] = '\0';
    i = 0;
    while(str1[i])                  /* 第10行 */
    {
        for(j = 0; j < 5; j++)
            if(str1[i] == str2[j])
                count++;            /* count用于计数 */
        i++;
    }
    printf("count = %d.\n", count);
    return 0;
}
```

实验步骤

① 程序运行后，观察程序的执行结果，分别如图3-74和图3-75所示。

② 注意循环变量与下标变量的结合。

③ 分别将语句“str[i-1]='\0';”删除再运行程序，通过对照程序的运行结果分析程序的功能。

图3-74　字符串的输入方式

图3-75　统计字符串中的元音字母

思考：

① 对照删除语句“str[i-1]='\0';”后出现的结果，想想该语句的功能是什么。

② 程序中以下语句：

```
while((str[i++] = getchar()) != '?')
    ;
str[i-1] = '\0';
```

的功能是什么?'\0'在此的作用是什么?

(2) 输入一个字符串(少于 20 个字符),统计并输出其中数字字符的个数。

实验步骤

① 输入一个字符串到字符数组中。

② 使用循环语句依次扫描每个字符串中的字符,并使用计数器对其中的数字字符进行计数。

③ 循环结束后,输出相应的统计结果。

程序代码如下:

```
/* 实验 9_3.C */
#include <stdio.h>
int main()
{
    int i, count = 0;
    char str[20];
    printf("Input a string:\n");
    gets(str);
    /* 从第 1 个字符开始扫描,直到遇到'\0'前的字符 */
    for(i = 0; str[i] != '\0'; i++)
    {
        if(str[i] >= '0' && str[i] <= '9')          /* 判定是否为数字字符 */
            count++;
    }
    printf("%s has %d digits.\n", str, count);
    return 0;
}
```

程序的运行结果如图 3-76 所示。

图 3-76　统计字符串中的数字

思考:

① 语句"gets(str);"的作用是什么?能否用 scanf 函数来表达该语句?它们有什么异同?

② 循环语句:

```
for(i = 0; str[i] != '\0'; i++)
```

的作用是什么?能否用其他语句来进行替换?

2. 设计性实验

(1) 输入一个字符串(少于 20 个字符),求取并输出此字符串的长度(要求不使用 strlen 函数)。

实验提示

① 输入一个字符串到字符数组中。

② 设置计数器初值为 0,并使用循环语句依次扫描字符串中的所有字符,每扫描一个计数器加 1,直到遇到字符'\0'时循环结束。

③ 输出相应的计数器的值。具体实验的运行结果可参考图 3-77。

(2) 输入三句话(每句不超过 80 个字符),要求分别统计其中英文大写字母、小写字母、数字和其他字符的个数。

实验提示

① 定义一个3×80的二维字符数组a用于存放要操作的字符串,二维数组中的每一行依次存储输入的一句话。

② 对于其中的每一行数组,可以当作一个字符串(一维字符数组)来进行处理,通过循环和计数器结合统计其中的特征字符个数。

③ 在对每一句话处理时,注意循环的次数不是1～80,而是根据字符串的长度来决定循环次数。

④ 输出相应的计数器统计结果。具体实验的运行结果可参考图3-78。

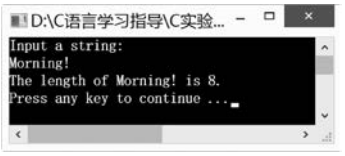


图3-77　计算字符串长度

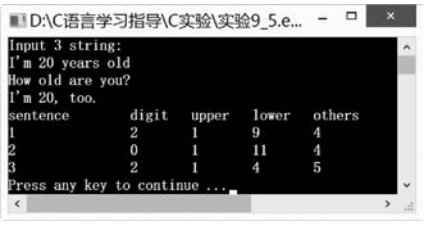


图3-78　统计多个字符串中的特征字符

3. 提高性实验

(1) 输入一个以回车结束的字符串(少于20个字符),删除其中所有的数字字符,同时将所有的小写字母变为大写字母,然后输出该字符串。

解题思路

输入一个字符串到字符数组str中。首先从原字符串的起始位置开始检测,逐个判断当前字符是否是数字字符,若是数字字符,则删除该字符得到新的字符串,并从新字符串的起始位置重新开始检测;若不是数字字符,则继续检测下一个字符。其中,删除字符串中某个字符的过程为:将与该字符相邻的下一个字符开始一直到字符串结束,所有字符依次往前移动一个位置。重复以上过程,直到字符串结束,最后将得到的新字符串用strupr函数处理,即可得到所要求的字符串。具体实验的运行结果可参考图3-79。

(2) 输入一个由数字字符组成的字符串(少于6个字符),将其转换为十进制的整型数据并输出该数据的值。

解题思路

要将数字字符转换为数字,需要利用这些字符所对应的ASCII码及这些ASCII码之间的关系。主要通过它们的ASCII码与字符'0'的ASCII码的差值来计算,如字符'0'要转化为数字0,可通过'0'-'0'表达式得到,同理,'1'-'0'可得到1,以此类推。在得到这些数字序列后,再给每个数字赋以权值就能得到最后的数据,如:123中1的权值为100。赋权值的过程可描述如下(以123为例):s=0,s=s×10+1,s=s×10+2,s=s×10+3。通过整理可结合循环进行程序实现。具体实验的运行结果可参考图3-80。

图3-79　删除字符串中的数字字符

图3-80　字符串转换为十进制数

实验项目 10　数组与指针程序设计

一、实验目的

(1) 了解指针的概念并掌握指针变量的定义。
(2) 掌握指向数组的指针及通过指针来访问数组元素的方法。
(3) 掌握字符串指针及指向字符串的指针变量的使用方法。
(4) 理解指针数组的概念并掌握其简单的使用方法。
(5) 熟悉和掌握指针的概念及其使用方法。

二、实验要求

(1) 了解指针及其使用特点。
(2) 熟悉用指针对数组元素进行引用。
(3) 熟悉用指针对字符串进行简单操作。
(4) 了解指针数组及其使用方法。

三、实验内容

1. 验证性实验

(1) 输入并执行以下程序,结合程序的运行结果理解程序的含义。

```
/* 实验 10_1.C */
#include <stdio.h>
int main()
{
    int a, b, t;
    int *pA, *pB;
    pA = &a;
    pB = &b;
    printf("Please input a and b:\n");
    scanf("%d,%d", pA, pB);
    printf("Before:\n");
    printf("a=%d,b=%d\n", a, b);
    printf("*pA=%d,*pB=%d\n", *pA, *pB);
    t = *pA;
    *pA = *pB;
    *pB = t;
    printf("After:\n");
    printf("a=%d,b=%d\n", a, b);
    printf("*pA=%d,*pB=%d\n", *pA, *pB);
    return 0;
}
```

实验步骤

① 程序运行后,观察程序的执行结果,如图 3-81 所示。

图 3-81　指针简单使用回顾

② 注意语句“scanf("%d,%d",pA,pB);”的描述

方式。

③ 观察指针相关的操作符 * 和 & 的使用方式。

思考：

① 语句“scanf("%d,%d", pA, pB);”在未使用指针时通常是什么形式的？在此的 pA 相当于以前形式中的什么？

② 语句“pA=&a;”和“pB=&b;”的意义是什么？本程序中删除该语句后的结果是什么？为什么会出现如此结果？

(2) 任意输入 5 个整数到数组中，使用指针方式计算这些整数的乘积。

实验步骤

① 定义 5 个整数的数组 a 和一个同类型指针，并将 a 赋值给该指针，再定义一个表示累积的变量，且初始化为 1。

② 使用指针和循环语句相结合输入 5 个整数到数组 a 中。

③ 使用指针和循环语句相结合实现 5 个整数的累积过程。

④ 输出最后的累积结果。

程序代码如下：

```
/* 实验 10_2.C */
#include <stdio.h>
int main()
{
    int a[5], *p = a;
    int i, m = 1;
    printf("Please input 5 integers:\n");
    for(i = 0; i < 5; i ++)
        scanf("%d", p + i);
    for(i = 0; i < 5; i ++)
        m *= *(p + i);
    printf("m = %d\n", m);
    return 0;
}
```

程序的运行结果如图 3-82 所示。

图 3-82 使用指针计算数组元素的积

思考：

① 第一个 for 语句可否修改为如下形式？

```
for(i = 0; i < 5; i ++, p++)
    scanf("%d", p);
```

如果可以的话，原因是什么？是否还有其他改写方式？

② 程序中的指针 p 和数组名称 a 之间有什么异同？

③ 语句“m *= *(p+i);”的作用是什么？其中的 *(p+i)与 a[i]、p[i]和 *(a+i)是什么关系？

2. 设计性实验

(1) 任意输入 3×5 的矩阵数据到二维数组中，使用指针方式计算其中每一行的和。

实验提示

① 定义相应的二维数组 a 和一个指针，以及表示和的数组 s，并初始化 s 的所有元素

为 0。

② 通过二重循环(设循环变量依次为 i，j)结合指针输入数组的所有元素，其中第一层循环体中添加语句“p＝a[i];”，第二层循环体为 scanf("%d",p＋j)。

③ 通过一个二重循环分别计算每一行的和，其中第一层循环体中添加语句“p＝a[i];”，第二层循环体为 s[i] ＋＝ *(p＋j)。

④ 使用一重循环输出所有的和。具体实验的运行结果可参考图 3-83。

(2) 使用指针方式实现两个字符串的连接，并将连接后的字符串输出。

实验提示

① 定义两个字符数组，其中第一个字符数组的大小必须能够容纳连接后的字符串。定义两个字符类型的指针，分别初始化为两个数组的首地址。

② 通过指针与循环结合找到第一个字符串中'\0'的位置。

③ 通过指针与循环结合将第二个字符串中的内容依次复制到第一个字符数组中，直到第二个字符串为'\0'时结束。

④ 将整个字符串的末尾添加'\0'标志后输出整个新字符串。具体实验的运行结果可参考图 3-84。

图 3-83　使用指针计算二维数组一行的和

图 3-84　使用指针实现两个字符串的连接

3. 提高性实验

(1) 输入 10 个数据，使用指针引用的方式将它们按从大到小的顺序排列。

解题思路

在进行定义时，可直接定义指针并初始化为数组名称，在输入数据后，可通过简单的指针替换数组名称的方式实现相应的排序算法。同时需要注意数组名称表示地址时为常量，不能出现自增和自减操作。具体实验的运行结果可参考图 3-85。

图 3-85　使用指针对数组元素进行排序

(2) 使用指针数组完成：输入 3 个单词到数组中，再输入另外一个单词，查找前面的数组中是否出现过该单词，并输出相应的提示信息。

解题思路

在进行定义时，可直接定义一个二维字符数组和一维的指针数组，并将该指针数组初始化为二维字符数组中每一行的首地址，如此，指针数组的每一个元素即代表题目中的一个单词，可通过字符串比较函数 strcmp 进行较简单的单词比较并得到结果，同时输出相应的提示信息。具体实验的运行结果可参考图 3-86 和图 3-87。

图 3-86　使用指针查找字符串情况一

图 3-87　使用指针查找字符串情况二

实验项目 11　函数的定义和调用

一、实验目的

(1) 理解函数的作用及程序中使用函数的意义。

(2) 掌握 C 语言程序中函数定义的一般格式。

(3) 区别有返回值函数与无返回值函数调用的不同方式。

(4) 掌握函数实参与形参的对应关系及“值传递”的方式。

(5) 区分函数原型声明与函数定义的区别。

二、实验要求

(1) 了解标准库函数与用户自定义函数及其使用。

(2) 熟悉函数的定义和调用方法。

(3) 明确函数的实参与形参的对应关系。

三、实验内容

1. 验证性实验

(1) 定义一个函数 int prime(int n),其功能是判断一个整数是不是素数。当 n 为素数时,函数返回值为 1,否则返回值为 0。在 main 函数中输入一个整数,调用 prime 函数,输出该整数是否为素数的信息。

实验步骤

① 定义函数 int prime(int n),判断 n 是否为素数,若是,函数返回值为 1,否则返回 0。

② 编写 main 函数,输入一个整数,调用①中的函数 prime,判断此整数是否为素数,并输出结果。

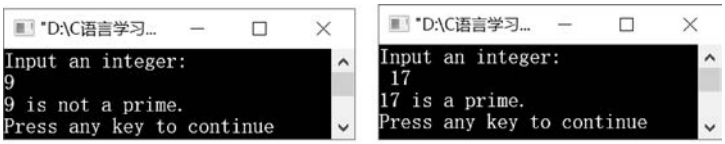
③ 对于多函数程序,可以每个函数独立进行编辑、编译,如果编译有错,可分别修改,最后再合并,这样便于调试。

程序代码如下:

```
/* 实验 11_1.C */
#include <stdio.h>
int main()
{
    int prime(int n);                          /* 第 5 行 */
    int n;
```

```
    printf("Input an integer:\n ");
    scanf(" %d",&n);
    if (prime(n))                                    /* 第 9 行 */
        printf("\n %d is a prime.\n",n);
    else
        printf(" %d is not a prime.\n",n);
    return 0;
}
int prime(int n)
{
    int flag = 1,i;
    for(i = 2;i <= n/2 && flag;i++)
      if(n % i == 0) flag = 0;
    return(flag);
}
```

程序的运行结果如图 3-88 所示。



(a) 不是素数　　(b) 是素数

图 3-88　判断输入的整数是否为素数

思考：

① 程序中第 5 行语句“int prime(int n);”起什么作用？是否可以省略？

② 程序第 9 行 if 语句的表达式为什么是 prime(n)？还有其他的表示方法吗？

③ 函数 int prime(int n)的定义中，变量 flag 起什么作用？若删除变量 flag，如何修改程序，使其得到相同的结果？

(2) 统计一个整数中某数字出现的次数。输入一个正整数 repeat (0＜repeat＜10)，做 repeat 次如下运算：输入一个整数，统计并输出该数中数字 2 的个数。

要求定义并调用函数 int countdigit(long number,int digit)，它的功能是统计整数 number 中数字 digit 的个数。例如：countdigit(10090,0)的函数值是 3，countdigit(34567,2)的函数值是 0。

实验步骤

① 定义函数 int countdigit(long number,int digit)。利用单循环结构，将整数 number 中的数字逐个求出，并与数字 digit 进行比较，若相等，则统计计数。

② 编写 main 函数，输入 repeat 的值(明确统计操作的次数)，做 repeat 次如下操作：输入一个整数，调用函数 int countdigit(long number,int digit)，统计该整数中数字 2 出现的次数，并输出结果。

③ 根据题意，main 函数调用 countdigit 函数时，形式参数 digit 对应的实参固定为 2。

程序代码如下：

```
/* 实验 11_2.C */
#include <stdio.h>
int countdigit(long number, int digit)
{
```

```
    int num,count = 0;
    number = number < 0? - number:number;          /* 第 6 行 */
    while(number){                                 /* 第 7 行 */
        num = number % 10;
        if(num == digit) count++;
        number/ = 10;
    }                                              /* 第 11 行 */
    return count;
}
int main()
{
    int i,repeat;
    int count;
    long in;
    printf("Input repeat:\n");
    scanf(" % d",&repeat);
    for(i = 1;i < = repeat;i++){
        printf("Input a number:\n");
        scanf(" % ld",&in);
        count = countdigit(in,2);
        printf("count = % d\n",count);
    }
    return 0;
}
```

程序的运行结果如图 3-89 所示。

图 3-89　统计一个整数中数字 2 的个数

思考：

① 程序的第 6 行语句起什么作用？

② 程序的第 7 行语句中 while(number)表示什么意义？还可以写成怎样的形式？

③ 第 7 行至第 11 行的程序段实现了什么功能？举例说明其实现步骤。

2. 设计性实验

(1) 编写程序，输入三角形的三边长 a、b、c，求三角形面积 area。具体要求如下。

① 定义函数 area(a,b,c)，给定三角形的三条边长，求三角形的面积。

② 定义 main 函数，输入三角形的边 a、b、c，判定能否构成三角形，若能构成三角形，则调用函数 area 求取三角形面积并输出；否则输出“不能构成三角形!”的提示语句。

例如：

输入：3.1,4.2,5.3

输出：area＝6.51

输入：1.1,2.2,3.3

输出：不能构成三角形!

实验提示

① 三角形面积的计算公式为 $area=\sqrt{s(s-a)(s-b)(s-c)}$，其中，a、b、c 分别为三角形的三边长，$s=\frac{a+b+c}{2}$。程序中需要使用求平方根的函数 sqrt()。

② 根据题意，函数 area()的返回值类型和其三个形式参数的数据类型定义为实型。

③ main 函数中，根据 area 函数的定义，明确其调用的形式，及其所提供的实际参数。

(2) 对实数 x 和整数 n，编写函数 expon 求 x^n，函数的返回值类型为 double。在 main 函数中输入实数 x 和整数 n 的值，调用函数 expon 求实数 x 的 n 次方的值，并输出计算结果(保留 3 位小数)。例如：

输入：2.1,3

输出：9.261

输入：2,－3

输出：0.125

实验提示

① 根据题意，函数 expon 的原型为 double expon(double x,int n)。

② 实数 x 的 n(n＞0)次方，即 x＊x＊…＊x(n 个 x 相乘)，利用循环语句即可求得该值。

③ 当 n＜0 时，x^n 为 1/(x＊x＊…＊x)(－n 个 x 相乘)。

④ 定义函数 main()，输入实数 x 和整数 n 的值，调用函数 expon 求得 x^n，输出结果。

(3) 程序填空。计算代数多项式 $1.1+2.2x+3.3x^2+4.4x^3+5.5x^4+6.6x^6+7.7x^7+8.8x^8+9.9x^9$ 的值，要求采用循环语句求和。例如：

输入：1.5

输出：596.43

```
/* 实验 11_5.C */
#include <stdio.h>
double poly(double);
int main()
{
    double x,y;
    scanf("%lf",&x);
    /* --- 请填上适当的语句 ------ */
    printf("%.2f\n",y);
    return 0;
}
/* --- 请填上适当的语句 ------ */
```

实验提示

① 定义 poly 函数，其功能是求多项式的值。

② 分析多项式的特点，采用循环语句求和。

③ main 函数调用 poly 函数得到求和结果。

3. 提高性实验

编写程序，N 名裁判给某歌手打分(假定分数都为整数)。评分原则是去掉一个最高分，去掉一个最低分，剩下的分数取平均值为歌手的最终得分。裁判给分的范围是：60≤分数≤100，裁判人数 N＝10。要求：每个裁判的分数由键盘输入。例如：

输入：89 90 92 95 90 93 88 92 93 91

输出：91.25

解题思路

① 根据题意,需要求最高分和最低分,所以可以定义以下两个函数。

- max(): 返回两个数中较大的值。
- min(): 返回两个数中较小的值。

② 在 main 函数中。

- 定义两个整型变量并赋初值 maxscore=0,minscore=100,分别用于存放 N 个数中的最大值和最小值。
- 定义整型变量并赋初值 sum=0,用于求累加和。
- 采用循环结构,逐个输入每位裁判的分数,并随时记录输入过程中的最大值 maxscore 和最小值 minscore 及累加值 sum。
- 输出(sum-maxscore-minscore)/(N-2)的值。

实验项目 12　函数的嵌套调用与递归函数

一、实验目的

(1) 进一步掌握函数的定义和调用。

(2) 深入理解函数的形参和实参的概念。

(3) 掌握函数嵌套调用的方法。

(4) 掌握函数递归调用的方法。

二、实验要求

(1) 熟悉函数调用的方法。

(2) 熟悉函数调用时形参和实参的一致性对应关系。

(3) 理解递归的概念。

(4) 掌握用递归方法解决问题。

三、实验内容

1. 验证性实验

(1) 按下面要求编写程序。

① 定义函数 int total(m)计算 1+2+3+…+m 的值。

② 定义函数 main(),输入正整数 n,计算并输出下列算式的值。要求调用函数 total 计算 1+2+3+…+n 的值。

$$s=1+\frac{1}{1+2}+\frac{1}{1+2+3}+\cdots+\frac{1}{1+2+3+\cdots+n}$$

实验步骤

① 根据题意,total 函数的原型为 int total(int m)。

② 在 total 函数体内,使用循环语句计算表达式 1+2+3+…+m 的值,并将求和得到的值返回给函数 total。

③ main 函数中输入正整数 n 的值，使用循环语句计算算式的值。算式累加和的通项式表示为$\frac{1}{1+2+3+\cdots+k}$，其中 k 的取值范围为 1～n。

程序代码如下：

```
/* 实验 12_1.C */
#include <stdio.h>
int total(int m);              /* 第 3 行 */
int main()
{
    int k,n;
    double s = 0;              /* 第 7 行 */
    printf("Input n:\n");
    scanf("%d",&n);
    for(k = 1;k <= n;k++)
        s += 1.0/total(k);     /* 第 11 行 */
    printf("s = %.2lf\n",s);
    return 0;
}
int total(int m)
{
    int i,sum = 0;
    for(i = 1;i <= m;i++)
        sum += i;
    return sum;
}
```

程序的运行结果如下。

```
输入:6
输出:s = 1.71
输入:16
输出:s = 1.88
```

思考：

① 程序中的第 3 行语句起什么作用？还可以写在什么位置？什么情况下可以省略？

② 程序中的第 7 行语句是否可以改写为“int s=0;”？为什么？

③ 程序中的第 11 行语句是否可以改写为“s=s+1/total(k);”？为什么？

(2) 求 Fibonacci(斐波那契)数列前 20 项的值。

要求：用递归法，定义函数 int fib(int k)，求 Fibonacci 数列第 k 项的值。Fibonacci 数列又称黄金分割数列，指的是这样一个数列：1,1,2,3,5,8,13,21,…，即第一项和第二项的值为 1，从第三项开始，以后每一项的值为其前两项之和。

实验步骤

① 在数学上，斐波那契数列以如下递归的方法定义：F0=0，F1=1，F(n)=F(n−1)+F(n−2)(n≥2)。利用递归式即可定义递归函数 int fib(int k)；

② 定义 main 函数，使用循环语句调用函数 int fib(int k)逐项求得 Fibonacci 数列的前 20 项的值，并输出结果。

程序代码如下：

```
/* 实验 12_2.C */
```

```
#include <stdio.h>
int fib(int k);
int main()
{
    int k,count = 0;
    for(k = 0;k < 20;k++)          /* 第7行 */
    {
        printf("%5d",fib(k));
        count++;
        if(count % 5 == 0)printf("\n");
    }
    return 0;
}
int fib(int k)
{
    if(0 == k||1 == k)
        return 1;
    else
        return fib(k - 2) + fib(k - 1);
}
```

程序的运行结果如图3-90所示。

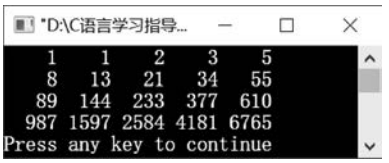


图3-90 Fibonacci数列前20项的值

思考：

① 程序中的第7行语句是否可以改写为for(k=1;k<=20;k++)？为什么？

② main函数中的变量count起什么作用？

③ 若不用递归，函数int fib(int k)如何实现求Fibonacci数列第k项的值？

2. 设计性实验

(1) 计算并输出下列算式的值。

$$s=1+\frac{1+2}{2!}+\frac{1+2+3}{3!}+\cdots+\frac{1+2+\cdots+n}{n!}$$

具体要求如下。

① 定义函数fact(n)，计算n的阶乘n!=1*2*3*…*n，函数返回值类型是double。

② 定义函数cal(m,n)，计算任意区间(m～n)内整数的累加和s=m+(m+1)+…+n，函数返回值类型是double。

③ 定义函数main()，输入正整数n，计算并输出算式的值(保留两位小数)。算式中每一项的分子是累加和，要求调用函数cal(m,n)，计算分子1+2+…+n；每一项的分母是阶乘，要求调用函数fact(n)计算n!。例如：

输入：3

输出：3.50

输入：10

输出：4.08

实验提示

① 函数 cal(m,n)的功能是计算累加和 m+(m+1)+…+n,其两个参数分别为求和数的下限和上限。

② 算式的求和通式为$\frac{1+2+3+\cdots+k}{k!}$,其中 k 的取值范围是 1～n。求和时,每加一项,k 的值需要递增 1。

③ main 函数中调用函数 cal(m,n)计算分子的值,注意分子的算式求和总是从 1 开始的。

(2) 编写程序,输入 n 的值,求 s=1!+2!+3!+…+n!。要求定义递归函数 fact(n)求 n!。例如:

输入:6

输出:873

输入:15

输出:1401602636313

实验提示

① 求 n! 的递归公式为

$$\text{fact}(n)=\begin{cases}1 & (n=0 \text{ 或 } 1)\\ \text{fact}(n)=n\times\text{fact}(n-1) & (n>1)\end{cases}$$

② 根据递归公式,定义递归函数 fact(n),用于计算 n!。

③ 定义 main 函数,输入 n 的值,用循环语句实现 i 为 1～n 的循环,核心计算公式为 s=s+fact(i)。

④ 注意程序中求和变量 s 的数据类型为 double 型。

(3) 编写程序,定义函数 DtoB(n),其功能是将一个十进制整数转换成二进制数。例如:

输入:25

输出:11001

实验提示

① 将一个十进制整数 n 转换为二进制数可以采用“除 2 取余”法,即 n 除以 2 取余数,再用商除 2 取余数,反复计算,直到被除数为 0 为止。

② 函数 DtoB(n)的原型为 void DtoB(int n),函数中可以定义一个 int 型数组,将 n 除 2 的余数逐个保存到数组中,然后逆序将数组元素的值逐个输出。也可以将函数 DtoB 定义成一个递归函数。

③ 在 main 函数中,输入 n 的值,然后调用函数 DtoB(n)输出 n 的二进制数形式。

3. 提高性实验

(1) 王小二自夸刀功不错。有人放一张煎饼在砧板上,问他:“饼不许离开砧板,切 100 刀最多可以分为多少块?”。例如:

输入:100

输出:切 100 刀最多可以分为 5051 块。

解题思路

① 令 q(n)为切 n 刀能将饼分为最多的块数,可以得到以下结果。

q(1)=1+1=2

q(2)=1+1+2=4

q(3)=1+1+2+3=7

q(4)=1+1+2+3+4=11

…

② 在切法上是让每两线都有交点。不难得出以下公式。

$$q(n)=q(n-1)+n \quad (n\geqslant 1)$$

$$q(0)=1 \quad (一刀都不切当然只有一块)$$

③ 采用循环结构,用递推法求解问题(或者定义递归函数)。

(2) 这是一个古典的数学问题:相传在古代印度的 Bramah 庙中,有位僧人整天把 3 根柱子上的金盘倒来倒去,原来他是想把 64 个一个比一个小的金盘从一根柱子上移到另一根柱子上去。移动过程中恪守下述规则:每次只允许移动一只金盘,且大盘不得落在小盘上面。假设 3 根柱子的编号分别为 A、B、C。利用递归算法,编写程序,描述将 A 上 4 个盘子移到 C 的操作步骤。

程序的运行结果如图 3-91 所示。

图 3-91　将 A 上 4 个盘子移到 C 的步骤

解题思路

有人会觉得这很简单,事实却恰恰相反。n 个盘子由 A 移到 C,需移动的次数是 2^n-1,64 个盘子移动的次数为 $2^{64}-1=18\ 446\ 744\ 073\ 709\ 552\ 000$ 次,一年的秒数是:$365\times 24\times 60\times 60=31\ 536\ 000$ 秒,若 1 秒移 1 个盘子,则 $18\ 446\ 744\ 073\ 709\ 552\ 000\div 31\ 536\ 000=584\ 942\ 417\ 355$ 年,即 5849 亿年,从能源角度推算,太阳系寿命只有 150 亿年。

问题似乎一下变得很复杂,但换个角度考虑:如果有办法先将 63 个盘子按要求移到别的柱子上,那问题就容易解决了。只需做:

① 将 63 个盘子从 A 移动到 B。

② 将最底下的盘子从 A 移到 C。

③ 将 63 个盘子从 B 移到 C。

这样全部任务完成了。但是,有一个问题实际上未解决:如何将 63 个盘子从 A 移到 B 呢?用类似的方法:

① 将 62 个盘子从 A 移动到 C。

② 将最底下的盘子从 A 移到 B。

③ 将 62 个盘子从 C 移到 B。

这就是递归方法。如此层层递归,直到最后完成将 1 个盘子从一根柱子移到另一根柱子,问题就解决。

综合以上分析,得到递归算法如下。

① 将 A 上 n－1 个盘子移动到 B(借助 C)。

② 将 A 上 1 个盘子移动到 C。

③ 将 B 上 n－1 个盘子移动到 C(借助 A)。

实验项目 13　变量的作用域与存储属性

一、实验目的

(1) 理解变量的作用域概念并掌握全局变量和局部变量的作用域特点。

(2) 掌握局部变量之间或全局变量和局部变量同名时的系统处理原则。

(3) 理解变量的生存期的概念。

(4) 掌握 4 种不同性质变量的存储特点及其使用特点。

二、实验要求

(1) 熟悉全局变量和局部变量的概念。

(2) 熟悉不同函数内部变量同名时的屏蔽原则。

(3) 了解变量的静态存储属性和动态存储属性。

(4) 掌握变量的生存期概念。

三、实验内容

1. 验证性实验

(1) 完成主教材实例 6-7 和实例 6-9,体会变量的作用域特点。

(2) 编写程序,用一维数组存放 n(n＜30)个学生某课程的成绩,求最高分、最低分和平均分。具体要求如下。

① 编写函数 float average(int n),求最高分、最低分和平均分,函数返回值为平均分,最高分和最低分用相应的全局变量分别保存。

② 编写主函数,输入 n 的值,然后输入 n 个学生的成绩,调用函数 average 求最高分、最低分和平均分并输出(保留两位小数)。

实验步骤

① 定义一个全局数组 score,用于保存 10 个学生的成绩。

② 因为一个函数只能返回一个函数值。题中要求输出 3 个值(最高分、最低分和平均分),所以定义两个全局变量 max 和 min,分别用于保存最高分和最低分,平均分由函数 average 返回。

③ 在 main 函数中,输入整数 n 的值,用循环语句实现将 n 个学生的成绩保存到一维数

组 score 中,然后调用函数 average,输出最高分、最低分和平均分(保留两位小数)。

程序代码如下:

```
/* 实验 13_1.C */
#include <stdio.h>
#define N 30
int score[N],max, min;                    /*第 4 行 */
float average(int n)
{
    int i;
    float aver,sum = score[0];
    max = min = score[0];
    for(i = 1; i < n; i++)
    {
        if(score[i] > max) max = score[i];
        else if(score[i] < min) min = score[i];
        sum = sum + score[i];
    }
    aver = sum/n;
    return(aver);
}
int main()
{
    float ave;
    int i,n;
    printf("Input n:\n");
    scanf("%d",&n);
    printf("Input %d scores:\n",n);
    for(i = 0; i < n; i++)
      scanf("%d",&score[i]);
    ave = average(n);
    printf("max = %d\nmin = %d\n", max, min);
    printf("ave = %.2f\n",ave);
    return 0;
}
```

程序的运行结果如图 3-92 所示。

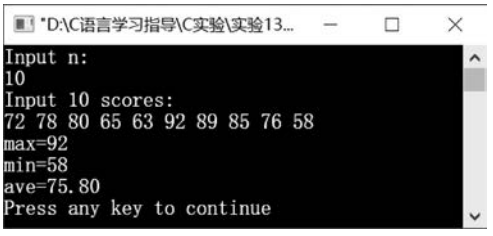


图 3-92 求最高分、最低分和平均分

思考:

① 将程序第 4 行中的 score[N]改为 score[n],可以吗?为什么?

② 将程序第 4 行中定义数组 score[N]移到 main 函数体内,可以吗?为什么?

③ 将第 4 行定义变量 max 和 min 的语句移到 average 函数体内,可以吗?为什么?

(3) 阅读以下程序,并回答问题。

```
/* 实验 13_2.C */
#include <stdio.h>
int k = 1;
void fun();
int main()
{
    int j;
```

```
    for(j = 0; j < 2; j++)
        fun();
    printf("k = %d", k);
    return 0;
}
void fun()
{
    int k = 1;        /* 第 14 行 */
    printf("k = %d,", k);
    k++;
}
```

问题：

① 程序的输出结果是什么？说明理由。

② 将程序的第 14 行改为“static int k＝1;”后，程序的输出结果是什么？说明理由。

③ 将程序的第 14 行改为“k＝1;”后，程序的输出结果是什么？说明理由。

④ 将程序的第 14 行改为“;”后，程序的输出结果是什么？说明理由。

2. 设计性实验

(1) 程序填空。已知函数 void add()，在 main 函数中输入变量 n 的值，调用 add 函数，求 1＋2＋3＋…＋n 的值，并输出求和结果。例如：

```
输入:50
输出:Sum is 1275.
/* 实验 13_3.C */
#include <stdio.h>
void add();
int result;
int main()
{
    /* -- 请填上适当的语句 -- */
    return 0;
}
void add()
{
    static int num = 0;
    num++;
    result += num;
}
```

实验提示

① add 函数被调用一次，全局变量 result 就累加一次变量 num 的值。因为 num 是静态局部变量，所以调用一次 add 函数，其值就增加 1。

② 利用这个特点，main 函数中只要循环调用 add 函数 n 次就可以计算得到结果。

(2) 程序填空。打印 2!，4!，6!，…，20! 的值。要求在 fact 函数中不使用循环语句，利用静态局部变量的特点求得阶乘的值。

程序的运行结果如图 3-93 所示。

图 3-93　打印 2!，4!，6!，…，20!的值

```
/* 实验 13_4.C */
#include <stdio.h>
double fact(int n);
int main()
{
    int i;
    for(i = 2; i <= 20; i += 2)
        printf("%d != %.0f\n", i, fact(i));
    return 0;
}
double fact(int n)
{
    /* ----- 请填上适当的语句 ----- */
}
```

实验提示

① 根据题意,相邻的两个阶乘值并非连续,如2!与4!。

② 若已知2!的值,要求出4!,需要在2!的基础上再乘以3和4,即计算阶乘值的表达式中需要乘以两个数。

③ 参考主教材实例6-12的源程序。

(3) 程序填空。求Fibonacci(斐波那契)数列前20项的值(每行输出4个数,每个数占5位,行末无空格)。

```
/* 实验 13_5.C */
#include <stdio.h>
int f1 = 1,f2 = 1;
void fib();
int main()
{
    int i;
    for(i = 1; i <= 10; i++){
    /* ----- 请填上适当的语句 ----- */
    }
    return 0;
}
/* ----- 请填上适当的语句 ----- */
```

程序的运行结果如图3-94所示。

实验提示

① 在数学上,斐波那契数列有以下特点:F0=0,F1=1,F(n)=F(n-1)+F(n-2)(n≥2),即第一项和第二项的值为1,从第三项开始,它的值为其前两项的值之和。

② 在fib函数中,将两个全局变量f1和f2作为迭代变量,用迭代法计算出后面的数据,每次计算两项。

③ 在mian函数中,使用循环结构,每次循环输出两个Fibonacci数列的值。

3. 提高性实验

从键盘输入n(0<n<11)个整数的值,先将这n个数原样输出,然后对其按从大到小的顺序进行排序后输出。具体要求如下。

① 定义函数void input(),输入n个整数到一维数组。

② 定义函数 void sort(),对 n 个数从大到小排序并输出。

③ 定义函数 void output(),将 n 个整数输出。

④ 定义 main 函数,输入 n 的值,根据题意分别调用 input 函数、sort 函数和 output 函数进行必要的数据处理。

程序的运行结果如图 3-95 所示。

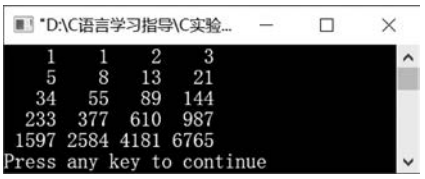


图 3-94　Fibonacci 数列前 20 项的值

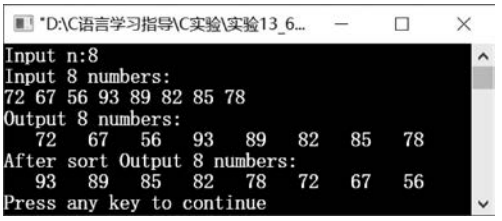


图 3-95　对 n 个数进行从大到小的排序

解题思路

① 因为要求定义的函数都是无参的,其共同的操作目标是 n 个整数,所以要定义全局变量 n 和全局数组用于保存 n 个整数。

② 用冒泡或选择排序法对数组中的 n 个数进行排序。

③ main 函数中调用 input 函数输入 n 个整数,调用 output 函数输出其原始值,然后调用 sort 函数对 n 个整数进行排序并输出(sort 函数调用 output 函数将排序后的数输出)。

实验项目 14　指针与函数

一、实验目的

(1) 掌握以指针变量作为参数的函数的定义和调用。

(2) 掌握数组作为参数时,函数的多种定义形式及其调用。

(3) 掌握返回指针的函数的定义和调用。

(4) 理解函数指针的概念。

(5) 掌握使用函数指针调用函数的方法。

(6) 了解 main 函数参数的使用。

二、实验要求

(1) 复习指针及其使用特点。

(2) 巩固指针和数组的相关知识。

(3) 复习指针数组的使用。

(4) 熟悉用指针对函数的引用。

(5) 理解 main 函数参数的意义。

三、实验内容

1. 验证性实验

(1) 完成教材实例 6-15 和实例 6-16,比较两个程序的差异。体会局部变量的特点,掌

握用指针变量访问其所指对象的方法。

(2) 编写程序,定义一个函数,其功能是求一个字符串的长度。

实验步骤

① 定义函数 int strlength(char *),其功能是求字符串的长度。

② 在 main 函数中输入一个字符串,调用函数 int strlength(char *) 求该字符串的长度。

③ 输出计算结果。

程序代码如下:

```
/* 实验 14_1.C */
#include <stdio.h>
int main()
{
    int strlength(char *);
    int len;
    char str[30];
    printf("Input a string: \n");
    gets(str);                              /* 第 9 行 */
    len = strlength(str);                   /* 第 10 行 */
    printf("The length of string is %d \n",len);
    return 0;
}
int strlength(char *p)
{
    int n;
    n = 0;
    while(*p!= '\0'){
        n++;
        p++;
    }
    return n;
}
```

程序的运行结果如图 3-96 所示。

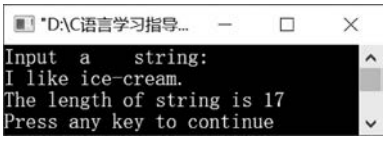


图 3-96 求字符串的长度

思考:

① 将程序中的第 9 行语句改为“scanf("%s",str);”,运行程序,分析结果。

② 将程序中的第 10 行语句改为“len = strlength(str[30]);”,可以吗? 为什么?

③ 程序执行过程中,形参变量 p 与实参数组 str 之间有什么关系?

(3) 编写程序,定义函数 double poly(double *,double),计算多项式 $a_0+a_1\sin x+a_2\sin x^2+a_3\sin x^3+\cdots+a_9\sin x^9$ 的值。

实验步骤

① 定义函数 double poly(double *,double),其功能是求多项式的值。

② 在 main 函数中:

- 将多项式各项系数保存到一维数组 a 中。
- 输入 x 的值。

• 调用函数 double poly(double *,double) 求多项式的值。

• 输出计算结果。

程序代码如下：

```
/* 实验 14_2.C */
#include<stdio.h>
#include<math.h>
double poly(double *p,double x)
{
    double sum,t=1;
    int i;
    sum=*p;                    /*第 8 行*/
    p++;
    for(i=1;i<10;i++,p++){
        t*=x;
        sum+=(*p)*sin(t);
    }
    return sum;
}
int main()
{
    double x,y;
    double a[10]={1.2,-1.4,-4.0,1.1,2.1,-1.1,3.0,-5.3,6.5,-0.9};
    printf("Input x:\n");
    scanf("%lf",&x);
    y=poly(a,x);               /*第 23 行*/
    printf("%.2f\n",y);
    return 0;
}
```

程序的运行结果如图 3-97 所示。

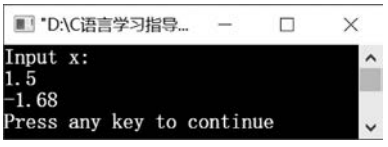


图 3-97　求多项式的值

思考：

① 程序第 8 行中的 *p 的值是什么？

② 将程序中的第 23 行改为“y=poly(&a,x);”，可以吗？为什么？

③ 将程序中的第 23 行改为“y=poly(a[10],x);”，可以吗？为什么？

2. 设计性实验

(1) 编写程序，输入 n(0<n<11)个学生的英语成绩，输出大于平均分的成绩。具体要求如下。

① 定义函数 void aboveAve(int score[],int n)，求 n 个整数的平均值，并输出大于平均值的数。

② 定义 main 函数，输入 n 的值，将 n 个整数输入一个一维数组中，调用 aboveAve 函数，输出大于平均分的值。例如：

输入：6(n 的值)

　　67　78　56　89　92　75

输出：78　89　92

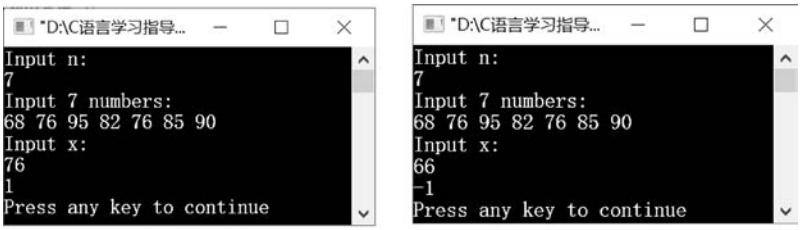
实验提示

① 在 main 函数中将 n 个整数输入一个一维数组中,要将该数组中的所有元素传递给 aboveAve 函数,所以要以数组名作为实参。

② aboveAve 函数中首先要求出 n 个数的平均值,然后再利用循环结构,将数组中的元素逐个与平均值比较大小,若大于平均值,则输出,否则不进行处理。

(2) 程序填空。定义函数 int search(int list[],int n,int x),在有 n 个元素的整型数组 list 中查找元素 x,若找到则返回数组元素的下标,否则返回-1。在 main 函数中输入 n 的值,将 n 个整数输入数组 a 中,然后输入 x 的值,调用 search 函数并输出函数值。

程序的运行结果如图 3-98 所示。



(a) 查找成功　　(b) 查找失败

图 3-98　在 n 个数中查找 x

```
/* 实验 14_4.C */
#include <stdio.h>
int search(int list[],int n,int x);
int main()
{
    int i, n,x, a[10], res;
    printf("Input n:\n");
    scanf("%d", &n);
    printf("Input %d numbers:\n",n);
    for(i = 0; i < n; i++)
      scanf("%d", &a[i]);
    printf("Input x:\n");
    scanf("%d", &x);
    res = search(a, n, x);
    printf("%d\n", res);
    return 0;
}
/* ----- 请填上适当的语句 ----- */
```

实验提示

① 在数组 list 中查找元素 x 可以用单循环,采用顺序查找的方法实现。

② 注意控制循环语句的执行。

3. 提高性实验

(1) 编写程序,定义函数 char *substrcpy(char *str1,int m,char *str2),其功能是将字符串 str1 中第 m 个字符开始的所有字符复制成一个新的字符串 str2,函数 substrcpy 返回新字符串的首地址。若 m 大于字符串的长度,则函数返回 NULL。

程序的运行结果如图 3-99 所示。

(a) 输入有效m值　　(b) 输入无效m值

图 3-99　求字符串的子串

解题思路

① 在函数 char *substrcpy(char *str1,int m,char *str2)中先求字符串 str1 的长度 len,然后与变量 m 的值比较,若 len<m,则函数返回值为 NULL,否则求新字符串,并将新字符串的首地址返回给函数。

② 在主函数中:

- 将字符串输入一维字符数组中,再输入 m 的值。
- 调用 substrcpy 函数得到新的字符串。
- 输出新字符串。

(2) 编写程序,求两个整数的较大值,要求用命令方式运行该程序,两个整数在命令行中输入。程序的运行结果如图 3-100 所示。

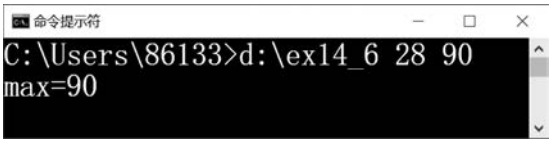


图 3-100　用命令行方式求两个整数的较大值

解题思路

① 根据题意,程序需要带参数的 main 函数。

② 命令行由 3 个字符串(可执行文件名"d:\ex14_6"和两个整数的值 28、90)组成,执行 main 时,参数 argc 的初值为 3,指针数组 argv 的 3 个元素分别保存 3 个字符串的首地址。

③ 因为命令行中输入的两个整数是以字符串的形式保存的,要比较整数的大小还需要将字符串转换成整数。可以考虑使用将字符串转换为整数的系统函数 atoi,该函数原型为 int atoi(char *str),程序中需要包含头文件 stdlib.h。

实验项目 15　结构体应用

一、实验目的

(1) 掌握结构体类型的定义,结构体类型变量的说明,以及成员的引用方法。

(2) 掌握结构体类型数组的概念和应用。

(3) 掌握结构体指针变量的概念和应用。

(4) 掌握单链表的概念和基本操作。

二、实验要求

(1) 熟悉结构体的概念和应用。

(2) 了解结构体变量的定义与引用。

(3) 了解结构体数组的定义与使用。

(4) 理解结构体指针变量的定义与使用。

(5) 熟悉单链表的基本操作与综合应用。

三、实验内容

1. 验证性实验

(1) 输入一位同学某门课程考试的相关信息,包括学号、姓名、性别和某门课程的成绩。

实验步骤

① 声明一个结构体类型,包含学号、姓名、性别和成绩四个成员项。

② 编写 main 函数,用步骤①中声明的结构体类型定义一个结构体变量和一个指向这种结构体类型的指针变量,给 4 个成员项赋值并输出。

程序代码如下:

```
#include <stdio.h>
#include <string.h>
struct exam                    /*第3行*/
{    long num;
     char name[10];
     char sex;
     float score;
};
int main()
{
     struct exam stud1, *p;
     char ch;
     p = &stud1;               /*第12行*/
     stud1.num = 200701;
     strcpy(stud1.name,"wang");
     ch = getchar();
     stud1.sex = ch;
     (*p).score = 543;         /*第17行*/
     printf("%ld,%c,%.2f,%s\n",p->num,p->sex,p->score,p->name);
     return 0;
}
```

程序的运行结果如图 3-101 所示。

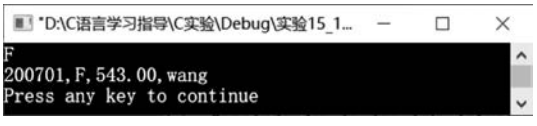


图 3-101　结构体变量的定义和表示

思考:

① 程序第 3 行定义的结构体类型的名字是什么?

② 程序中的第 12 行语句“p=&stud1;”起什么作用？是否可以省略？

③ 程序中的第 17 行语句“(*p).score=543;”中*p 两侧的括号能不能省略？访问结构中的成员有哪几种方法？具体在本程序中哪些语句中体现？

(2) 程序填空题。完善下列程序段，以 sum 成员项为准实现结构体数组的排序。

```
#include<stdio.h>
int main()
{
    struct student
    {
        char name[20];
        int num;
        float math;
        float eng;
        float cuit;
        float sum;
    }stu[4];
    struct student temp;
    int i,j,k;
    for(i=0;i<4;i++)
    {
        printf("input %d name=",i);
        gets(stu[i].name);
        printf("input %d num,math,eng,cuit=",i);
        scanf("%d%f%f%f",&stu[i].num,&stu[i].math,
            &stu[i].eng,&stu[i].cuit);
        getchar();
        stu[i].sum=stu[i].math+stu[i].eng+stu[i].cuit;
    }
    for(i=0;i<3;i++)
    {
        k=i;
      for(j=i+1;j<4;j++)
          if(____i____) k=j;
      if(k!=i)

          {
              temp=____ii_____;
              _____iii_____;
              ____iv_____=temp;
          }
    }
    printf("姓名        学号      数学      英语      语文      总分      \n");
    for(i=0;i<4;i++)
    {
        printf("%-10s%d%8.2f%8.2f%8.2f%8.2f\n",
            stu[i].name,stu[i].num,stu[i].math,
            stu[i].eng,stu[i].cuit,stu[i].sum);
    }
    return 0;
}
```

实验提示

① 目前已学到的排序算法有冒泡法和选择法。冒泡法的思路是：将相邻的两个数比较，将小的数调到前头。选择法的思路是：每一趟选出一个最小的数，并和当前位置第一的

数交换。此题中用的是选择法,以 sum 成员项为准实现排序,则:

```
(i)   stu[j].sum < stu[k].sum
```

② ANSI C 标准允许相同类型的结构体变量相互赋值,题中要实现两个结构体变量的整体交换,则:

```
(ii)   stu[i]
(iii)   stu[i] = stu[k]
(iv)   stu[k]
```

实验运行的结果如图 3-102 所示。

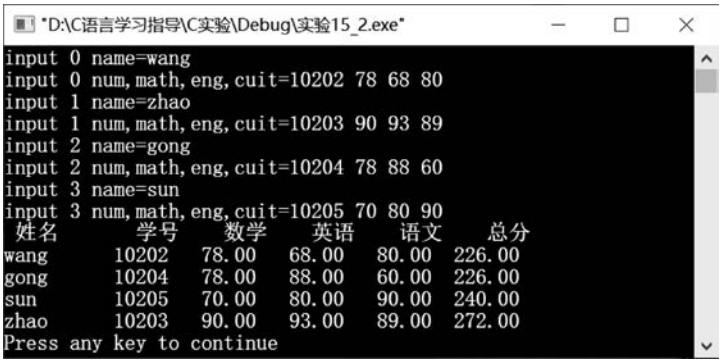


图 3-102　结构体数组的排序

2. 设计性实验

(1) 定义一个包括 5 个学生信息的结构体数组,每个学生包括学号、姓名和总分。输入一个学生的学号,在该结构体数组中查找该学号,如果找到,则输出该学生的姓名和总分;如果找不到,则输出显示"Not Found!"。

实验提示

查找的过程是将用户输入的学号和结构体数组中的学号成员项中的内容逐个比对,如果找到,则循环提前结束;若找不到,则循环正常结束(也就是说,一直找到数组中的最后一个元素都没有符合的信息)。为了区分循环究竟是提前结束还是正常结束,引入标志量 flag,flag=1 表示找到,则循环提前结束;flag=0 表示没有找到。

实验运行的结果分别如图 3-103 和图 3-104 所示。

图 3-103　找到时的实验运行结果

图 3-104　未找到时的实验运行结果

(2) 有 10 个学生，每个学生的数据包括学号、姓名、数学、物理和化学 3 门课程的成绩。从键盘输入 10 个学生的数据，具体要求如下。

① 输出数学最低分的学生的数据(包括学号、姓名、3 门课程成绩)。

② 求出每个学生的平均成绩，并按平均成绩由高到低排序。

实验提示

① 要求输出数学最低分的学生的数据，归结为求极值问题。解决此类问题的方法是：假定第一个数组元素的值是最小的，用 min 记录其下标，即 min=0。然后利用循环在数组中进行查找，若当前元素的成绩项的值大于假定的成绩项的值(即 stu[i].math<stu[min].math)，则当前最大的值的下标应修改为 min=i。当循环结束后，min 记录的即为数组中数学最低分的下标。

② 要求按平均成绩由高到低排序，可采用冒泡法或选择法。

实验的运行结果如图 3-105 所示。

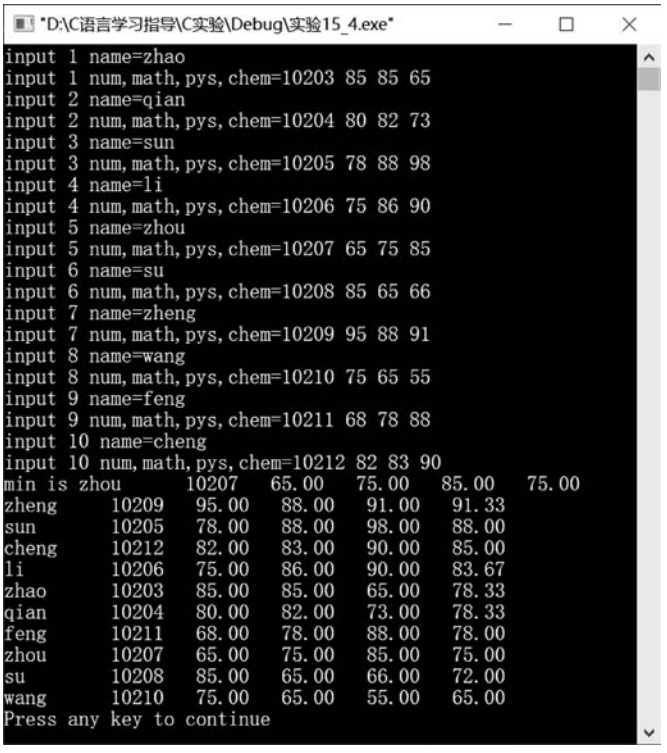


图 3-105　排序和求数学最低分

(3) 创建一个学生信息单链表，学生信息包括学生学号和成绩。创建结束后将所有的学生信息输出。要求按指定的长度创建单链表。

实验提示

先输入单链表的长度 n，再用头插法或尾插法创建一个单链表。最后将创建的单链表输出。

头插法的具体生成过程如下。

① 建立一个“空”链表。

② 输入数据元素 a_n,建立结点并插入在表头。

③ 输入数据元素 a_{n-1},建立结点并插入在表头。

④ 以此类推,直至输入 a_1 为止。

注意:所谓头插法指的是建立的新结点插入在表头位置,即插入点在头结点的后面。因此如果要建立一个顺序为 $a_1,a_2,\cdots,a_{n-1},a_n$ 的序列,需要逆序输入 n 个数据元素的值。

尾插法的具体生成过程如下。

① 建立一个"空"链表。

② 输入数据元素 a_1,建立结点并插入在表尾。

③ 输入数据元素 a_2,建立结点并插入在表尾。

④ 以此类推,直至输入 a_n 为止。

⑤ 循环结束,将 a_n 结点的指针域设置为"空"。

注意:所谓尾插法,指的是始终在链表的尾部插入,因此引用尾指针的概念。程序中 q 即为尾指针,它始终指向当前链表中最后一个结点的位置。在循环结束后要将最后一个结点的指针域赋值设置为"空"(q→next=NULL)。如果要建立一个顺序为 $a_1,a_2,\cdots,a_{n-1},a_n$ 的序列,顺序输入 n 个数据元素的值即可。

实验运行结果如图 3-106 所示。

图 3-106 用尾插法创建一个单链表

3. 提高性实验

(1) 编写程序,实现对带头结点的单链表进行就地逆置的操作。

解题思路

① 首先利用头插法或尾插法创建一个单链表,并输出。

② 单链表的逆置操作是单链表中较典型的应用。要将单链表中的结点逆置存放,可以借助用头插法建立单链表的思想。因为用头插法建立的单链表的结点顺序与读入的数据元素值的顺序正好相反。具体的逆置过程是:先将头结点的指针域设置为"空",形成一个空链表,再将原链表中的结点依次插入头结点的后面,直到所有的结点都插入为止,则实现了链表逆置。

说明:在算法的实现过程中,特别要注意在将原链表中的某个结点插入新形成的链表中之前,一定要记录它的后继结点。为此在算法中需引进两个指针,假设为 p 和 q,p 指向原单链表中待插入的结点,而 q 则指向 p 的后继。p 的初始值是指向单链表的第一个结点。

实验运行的结果如图 3-107 所示。

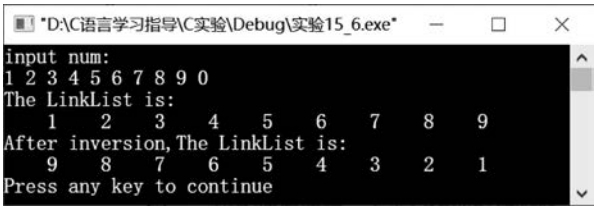


图 3-107 单链表的逆置

(2) 已知两个一元多项式 A(x)和 B(x),编程实现两个一元多项式的加法运算,即 A(x)=A(x)+B(x)。具体要求如下。

① 分别输入两个多项式的项数及各项的系数和指数,创建两个多项式。

② 对两个多项式求和,并输出求和后的多项式。

③ 程序的输入与输出严格按要求设计。效果如图 3-108 所示。

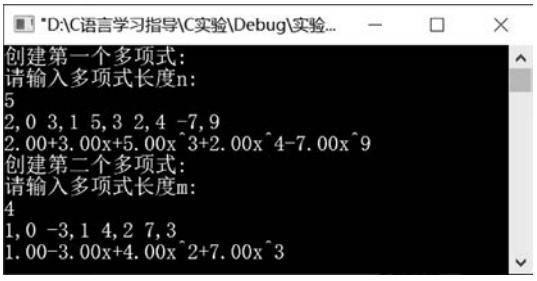


图 3-108　创建多项式输入示例

解题思路

① 根据示例的要求,多项式按升幂的形式输入和输出。

② 多项式如何存放? 如果只是将各项的系数采用数组的形式存放,当多项式阶数很高,且相邻之间的阶数相差很大时,会造成大量存储空间的浪费。例如:$A(x)=8+4x^{1002}-3x^{20003}$,这个多项式按上述方式存放时,需要一个长度为 20004 的数组,且数组中只有 3 项是非 0 元素。为了避免这种情况,可以采用只存储非 0 项,且在存储非 0 项系数的同时存储非 0 项的指数。同时,因为两个多项式的长度和其中的阶数都有可能不同,因此用带头结点的单链表方式实现。

定义一个结构体类型:

```
struct PolyNode{                /*项的表示*/
    float coef;                 /*系数*/
    int expn;                   /*指数*/
    struct PolyNode *next;
}
```

为了描述得简洁,利用 typedef 语句有以下定义:

```
typedef struct PolyNode{            /*项的表示*/
    float coef;                     /*系数*/
    int expn;                       /*指数*/
    struct PolyNode *next;
  } PolyNode , *PolyNomial;
```

说明:

① PolyNode 为结点类型,PolyNomial 为指向 PolyNode 结点类型的指针类型。即 PolyNomial p 等价于 PolyNode *p。

图 3-109　结点类型的存储结构

② 结点类型的存储结构如图 3-109 所示。

如果 $A(x)=2+3x+5x^3+2x^4-7x^9$,$B(x)=1-3x+4x^2+7x^3$,则它们的链式存储结构如图 3-110 所示,其中的"∧"代表空指针域。

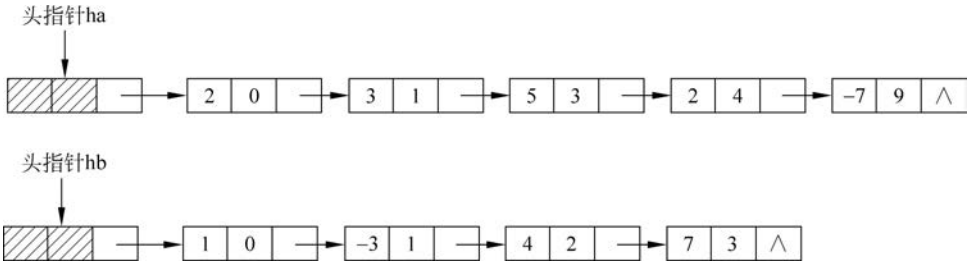


图 3-110　两个多项式的存储结构

③ 图 3-110 中的多项式链表的创建可以用头插法或尾插法实现。

④ 多项式相加的结果如图 3-111 所示,其中的“×”标志的结点是相加后被删除的结点。

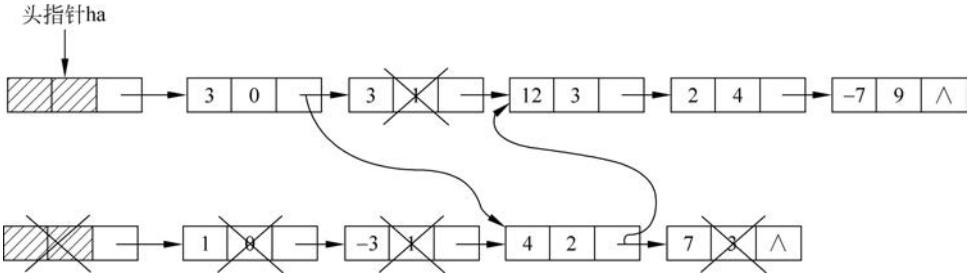


图 3-111　两个多项式相加后的结果

程序运行的结果分别如图 3-112(相加结果非空)和图 3-113(相加结果为空)所示。

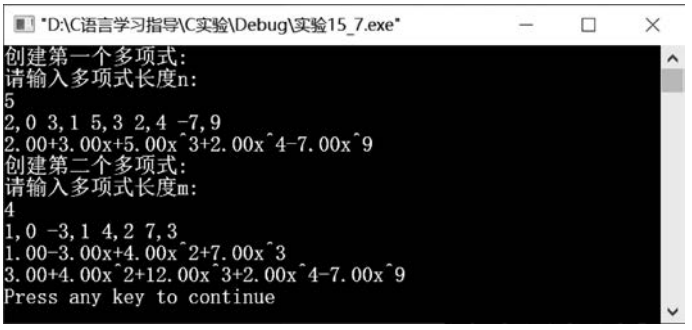


图 3-112　多项式求和(相加结果非空)

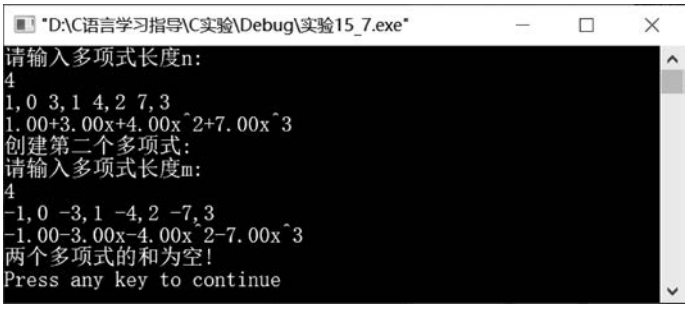


图 3-113　多项式求和(相加结果为空)

实验项目 16　文件及应用

一、实验目的

（1）理解文件和文件指针的概念。

（2）掌握文件的打开、关闭以及文件的读写操作。

（3）了解数据文件在程序中的应用。

二、实验要求

（1）预习数据流文件与程序文件的区别。

（2）熟悉各种文件读写函数的使用。

（3）预习数据流文件操作的特点。

三、实验内容

1. 验证性实验

新建一个磁盘文件 ex16_1.dat，从键盘输入一个字符串，将其中的小写字母全部转换成大写字母写入到该文件中（要求写入文件的同时将结果显示在屏幕上）。

实验步骤

① 以写的方式打开文件 ex16_1.dat。

② 从键盘输入一个字符串。

③ 用单循环结构，将字符串中所有的小写字母转换成大写字母，并写入到该文件中。

④ 关闭文件。

程序代码如下：

```
/* 实验 16_1.C */
#include <stdio.h>
#include <stdlib.h>
int main()
{
    FILE *fp;                                          /*第 6 行*/
    char str[80];
    int i = 0;
    if((fp = fopen("ex16_1.dat", "w")) == NULL){       /*打开文件*/
        printf("Can not open the file\n");
        exit(1);
    }
    printf("Input a string: \n");
    gets(str);
    printf("The string in file is: \n");
    while(str[i]!= '\0'){
        if(str[i]>= 'a'&&str[i]<= 'z')
            str[i] = str[i] - 32;
        putchar(str[i]);
        fputc(str[i],fp);                              /*第 20 行*/
        i++;
```

```
    }
    putchar('\n');
    fclose(fp);
    return 0;
}
```

程序的运行结果如图 3-114 所示。

图 3-114 字符串写入文件

思考：

① 将程序中的第 6 行改为"file * fp;"可以吗？为什么？

② 第 20 行语句的功能是用 fputc 函数将一个字符写入文件中。若改用 fprintf 函数，如何修改语句？

2. 设计性实验

(1) 完成以下功能：f(x,y)=(3.14 * x−y)/(x+y)，若 x、y 取值为区间[1,6]的整数，找出使 f(x,y)取最小值的 x1、y1，并将 x1、y1 以格式"%d,%d"写入到文件 design.dat 中(要求写入文件的同时将结果显示在屏幕上)。

程序的运行结果：

```
文件 design.dat 中的内容为:1,6
显示屏幕上输出:1,6
```

实验提示

① 定义一个函数求表达式(3.14 * x−y)/(x+y)的值，函数原型为：double f(int x,int y)。

② 利用双重循环结构求 f(x,y) 取最小值的 x1、y1。

③ 对文件 design.dat 的操作过程参考实验 16_1.c 的内容。

(2) 新建一个含有字符'$'的 data.txt 文本文件，统计文本文件 data.txt 中字符'$'出现的次数，并将统计结果写入文件 result.txt 中。

例如：

文件 data.txt 中的内容为：books 15$,pens 20$,pencils 10$,bags 35$.

文件 result.txt 中的内容为：字符'$'出现 4 次。

实验提示

① 利用记事本创建文本文件 data.txt。

② 利用单循环结构，使用 fgetc 函数将文件 data.txt 中的字符逐个读出，并统计字符'$'的个数。使用 fprintf 函数将结果写入文件 result.txt 中。

③ 注意：无论文件 data.txt 还是文件 result.txt，对文件进行读写操作前都需要先打开文件，操作结束后关闭文件。

(3) 在文件 string.txt 中输入一个字符串，将组成字符串的所有非英文字母的字符删除，将该字符串保存到文件 letter.txt 中(要求写入文件的同时将结果显示在屏幕上)。

例如：

文件 string.txt 中的内容为：a+5x−siny+b

文件 letter.txt 中的内容为：axsinyb

实验提示

① 利用记事本创建文本文件 string.txt。

② 使用 fgets 函数将文件 string.txt 中的字符串读出，保存到一个一维字符数组中。用单循环结构，将字符串中的非英文字母的字符删除。

③ 用 fputs 函数将字符串写入文件 letter.txt 中。

④ 注意：文件 string.txt 需要以读的方式打开，文件 letter.txt 需要以写的方式打开，操作结束后要关闭文件。

3. 提高性实验

从文件 stud.dat 中输入 5 位学生的信息，包括学号、姓名和三门课程成绩。计算每个学生的平均成绩。将每个学生的信息（学号、姓名、三门课程成绩和平均分）保存到磁盘文件 stud_info.dat 中（要求写入文件的同时将结果显示在屏幕上）。

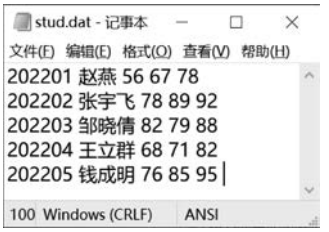


图 3-115　学生基本信息

文件 stud.dat 中的内容如图 3-115 所示，程序的运行结果如图 3-116 所示。

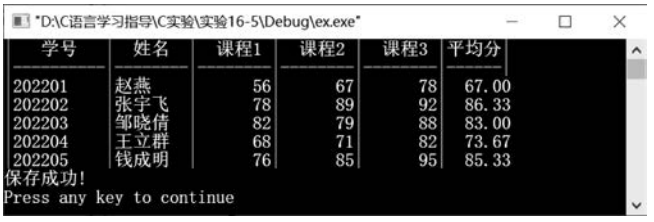


图 3-116　学生成绩的管理

解题思路

① 根据题意，考虑构造结构体类型 struct student，有 4 个成员项，分别为学号、姓名、三门课程成绩（一维数组）和平均成绩，并定义结构体数组。

② 利用单循环结构输入学生的信息。

③ 利用单循环结构计算学生的平均分（也可以在②中输入成绩后就计算平均分）。

④ 建议使用 fwrite 函数将数据保存到文件 stud_info.dat 中，注意打开文件时文件操作方式的选择。

⑤ 建议分别定义多个函数完成程序。

实验项目 17　C 语言程序综合应用

一、实验目的

（1）加深对 C 语言程序设计基本理论和基本知识的理解。

（2）深入理解结构化和模块化程序设计思想。

（3）把所学的理论知识与实践相结合，提高用计算机解决实际问题的能力。

（4）培养逻辑思维能力、团队合作精神、实际动手能力和创新能力。

二、实验要求

（1）具备一定的 C 语言程序设计基础。

(2) 确定所选课题的功能模块,详细描述各模块的具体内容。

(3) 认真分析设计过程中涉及的算法,用流程图描述算法。

(4) 熟练掌握程序调试的方法和步骤。

三、实验内容

1. 学生档案管理

(1) 题目描述。

编写一个程序,管理学生的档案,系统能实现以下功能。

① 输入信息:首先输入记录数,然后逐条输入学生的基本信息。

② 修改信息:根据学号对学生的基本信息进行修改。

③ 增加信息:添加新学生的基本信息。

④ 插入信息:在指定位置插入一个学生的信息。

⑤ 删除信息:根据学号或姓名删除指定学生的信息。

⑥ 查询:根据学号或姓名查询某个学生的信息;根据学号区间段查询某些学生的信息。

⑦ 排序:选择根据学号进行升序或按性别进行降序排序,并显示排序后的结果。

⑧ 统计:根据性别统计男、女生比例;根据家庭地址统计各省份生源数。

⑨ 输出:输出所有学生信息或查询学生信息的结果。

⑩ 保存:将所有学生信息保存到文件 student.dat 中。

(2) 设计提示。

① 先确定学生档案管理的数据结构。如每个学生的信息包括学号、姓名、性别、出生日期、家庭地址等,每个数据项各用什么数据类型。

② 划分实现学生档案管理的功能模块,如主菜单、输入数据、修改、增加、插入、删除、查询、排序和输出等功能,并确定各功能模块的实现算法。

2. 选票统计

(1) 题目描述。

从 100 名优秀运动员中评选出 10 名最佳运动员,具体规则如下。

① 运动员号按 1,2,3,…顺序编号。

② 由键盘接受所收到的选票,每张选票至多可写 10 个不同的编号。

③ 对应名次的运动员编号可以有空缺,但必须用 0 表示。

④ 若选票中编号超出规定的范围,或编号出现重复,则作废选票。

⑤ 按选票中所列最佳运动员顺序给他们计分,计分标准如下:从第 1 名至第 10 名所得分数依次为:15,12,9,7,6,5,4,3,2,1。

⑥ 按各运动员所得分数的高低进行排序,列出前 10 名最佳运动员排名,格式为:

名次　　运动员编号　　合计得分　　合计得票数

如果得分相同,则得票多者在前;如果得分与票数都相同,则编号小的在前。

(2) 设计提示。

① 根据题意,明确选票及其运动员的完整数据信息。

② 选票统计过程分为三个步骤:投票、选票检查(即确认选票是否为有效票)和有效票

统计。

③ 根据要求,可将问题分解为以下多个功能模块。

- 输入选票。
- 选票检查。
- 统计每个运动员的总得票数和总分。
- 对运动员的得分高低进行排序。
- 输出运动员的排名。

④ 可以考虑通过无限循环接收所有选票,用输入运动员编号为－1 表示本张选票结束,输入－2 表示所有选票结束。

3. 多项式求和

(1) 题目描述。

编写一个程序,实现多项式的求和。例如:多项式 $2x^5+3x^2-5.1x+6$ 与多项式 x^4-3x^2+x-2 求和的结果为 $2x^5+x^4-4.1x+4$。系统能实现以下功能。

① 输入:输入一个多项式。

② 插入:在一个多项式中插入一项。

③ 删除:在一个多项式中删除一项。

④ 查找:在一个多项式中查找某一项。

⑤ 合并:多项式求和。

⑥ 输出:输出多项式。

(2) 设计提示。

① 先确定多项式的数据结构。例如,确定每一项由哪些数据组成,每个数据项用什么数据类型比较合适。

② 明确功能模块:如主菜单、输入数据、插入数据、删除数据、查询、合并和输出等功能,并确定各功能模块的实现算法。