# 1 表达式和标准输入与输出实验

## 1.1 实验目的

(1)熟练掌握各种运算符的运算功能，操作数的类型，运算结果的类型及运算过程中的类型转换，重点是C语言特有的运算符，例如位运算符，问号运算符，逗号运算符等；熟记运算符的优先级和结合性。

（2）掌握getchar, putchar, scanf 和printf 函数的用法。

（3）掌握简单C程序（顺序结构程序）的编写方法。

## 1.2 实验内容

### 1.2.1 源程序改错

下面给出了一个简单C语言程序例程，用来完成以下工作：

（1）输入华氏温度f，将它转换成摄氏温度c后输出；

（2）输入圆的半径值ｒ，计算并输出圆的面积ｓ；

（3）输入短整数ｋ、ｐ，将ｋ的高字节作为结果的低字节，ｐ的高字节作为结果的高字节，拼成一个新的整数，然后输出；

在这个例子程序中存在若干语法和逻辑错误。要求参照1.3和1.4的步骤对下面程序进行调试修改，使之能够正确完成指定任务。

1 #include<stdio.h>

2 #define PI 3.14159;

3 voidmain( void )

4 {

5 int f ;

6 short p, k ;

7 double c , r , s ;

8 /\* for task 1 \*/

9 printf(“Input Fahrenheit:” ) ;

10 scanf(“%d”, f ) ;

11 c = 5/9\*(f-32) ;

12 printf( “ \n %d (F) = %.2f (C)\n\n ”, f, c ) ;

13 /\* for task 2 \*/

14 printf("input the radius r:");

15 scanf("%f", &r);

16 s = PI \* r \* r;

17 printf("\nThe acreage is %.2f\n\n",&s);

18 /\* for task 3 \*/

19 printf("input hex int k, p :");

20 scanf("%x %x", &k, &p );

21 newint = (p&0xff00)|(k&0xff00)<<8;

22 printf("new int = %x\n\n",newint);

}

**解答：**

（1）错误修改：

1) 第2行的符号常量定义后不能有分号，正确形式为：

#define PI 3.1415926

2) 第3行的voidmain(void)，正确形式为：int main(void)

3)第10行缺少取地址符scanf(“%d”, f ) ; 应该为scanf(“%d”, &f ) ;

4) 第15行%f应该为%lf

5) 第21行中左移应改为右移，定义变量应使用int newint…

6) 第11行中，表达式应改为5/9.0\*（f-32）

（2）错误修改后运行结果：

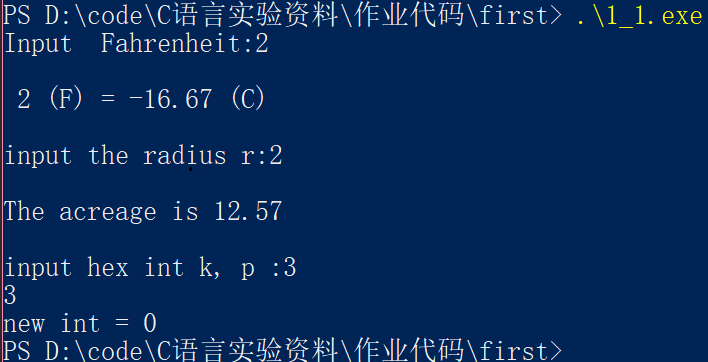


图1-1 源程序改错的运行结果

（3）改正后源代码：

#include<stdio.h>

#define PI 3.14159

int main(void){

int f;

short p, k;

double c , r , s;

/\* for task 1 \*/

printf("Input Fahrenheit:" );

scanf("%d",&f);

c = 5.0/9.0\*(f-32.0);

printf( "\n %d (F) = %.2f (C)\n\n", f, c ) ;

/\* for task 2 \*/

printf("input the radius r:");

scanf("%lf", &r);

s = PI \* r \* r;

printf("\nThe acreage is %.2f\n\n",s);

/\* for task 3 \*/

printf("input hex int k, p :");

scanf("%hd %hd", &k, &p );

short newint = (p&0xff00)+((k&0xff00)>>8);

printf("new int = %hd \n",newint);

return 0;

}

**1.2.2 源程序修改替换**

下面的程序利用常用的中间变量法实现两数交换，请改用不使用第3个变量的方法实现。该程序中t是中间变量，要求将定义语句中的t删除，修改下划线处的语句，使之实现两数对调的操作。

#include<stdio.h>

void main( )

{

int a, b, t;

printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

t=a ；a=b；b=t；

prinf(“\na=%d,b=%d”,a,b);

}

**解答：**

利用按位异或的计算性质，替换后的程序如下所示：

#include<stdio.h>

int main(void){

int a, b;

printf("Input two integers:");

scanf("%d %d",&a,&b);

a = a^b;

b = a^b;

a = a^b;

printf("\na=%d,b=%d", a, b);

}

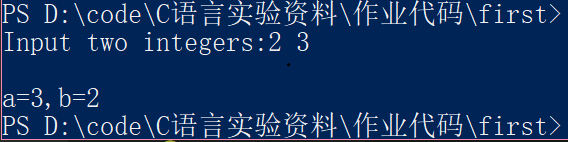


图1-2 源程序修改替换题运行结果

**1.2.3 程序设计**

**（1）**编写一个程序，输入字符ｃ，如果ｃ是大写字母，则将ｃ转换成对应的小写，否则ｃ的值不变，最后输出ｃ。

**解答：**

1） 算法流程如图1.1所示。

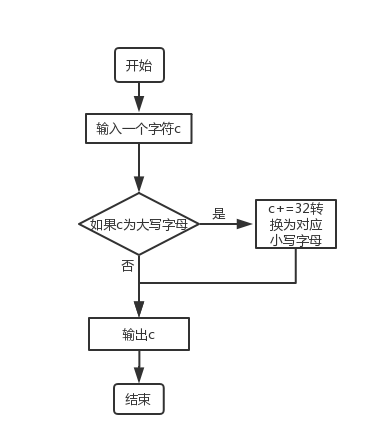


图1-3 编程题1的程序流程图

2）源程序清单

#include <stdio.h>

int main(void){

char c;

printf("Input a letter please:");

scanf("%c", &c);

if(c <= 90 && c >= 65)

printf("%c", (c+32));

else

printf("%c", c);

return 0;

}

3）测试

（a） 测试数据：

a A b B

（b） 对应测试数据的运行结果截图

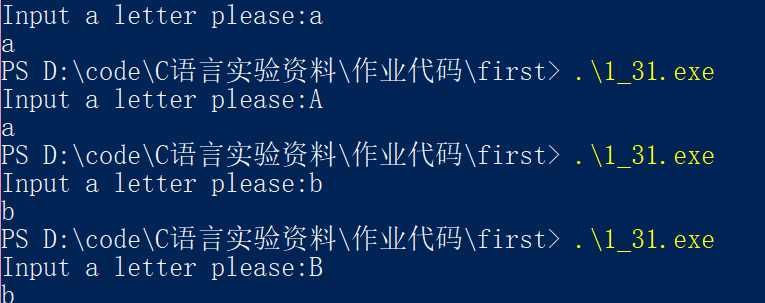


图1-4 编程题1的运行结果截图

（2）编写一个程序，输入无符号短整数x，ｍ，ｎ（0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ）,取出x从第ｍ位开始向左的ｎ位（ｍ从右至左编号为0～15），并使其向左端（第15位）靠齐。

**解答：**

1) 解题思路：

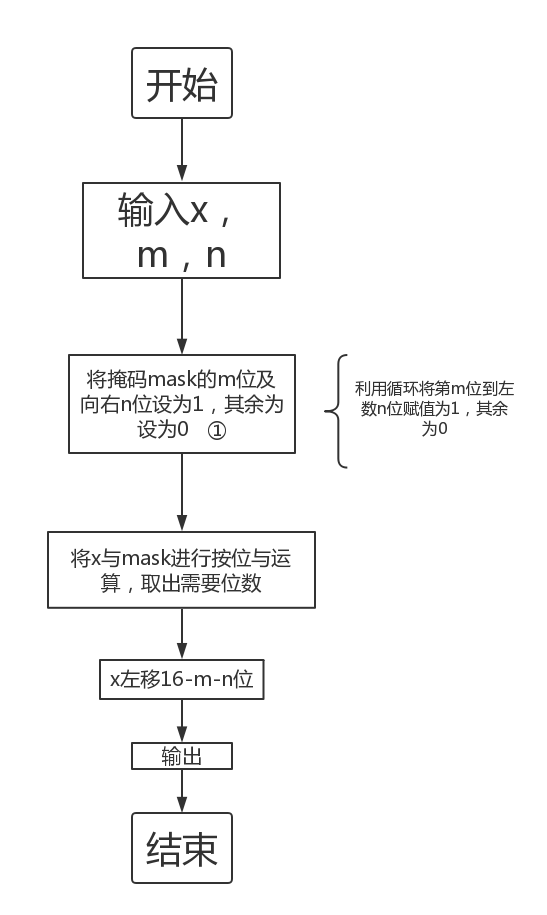


图1-5 编程题2的算法流程图

2）程序清单

#include <stdio.h>

#include <math.h>

int main(void){

unsigned short x;

int m, n;

unsigned short mask = 0;

int maskl[16];

printf("please input x, m, n:");

scanf("%hx %d %d", &x, &m, &n);

if(m>15 || m<0 || n>16-m || n<1)

printf("error\n");

else{

for(int i = 0; i <= 15; i++){ //利用循环将第m位到左数n位

//赋值为1，其余为0

if(i>=m && i<(m+n))

maskl[i] = 1;

else

maskl[i] = 0;

mask = maskl[i] \* pow(2.0, i) + mask;

}

x = x & mask;

x <<= (15 - (m + n-1));

printf("\n%x\n", x);

}

return 0;

}

3）测试

（a） 测试数据：

叙述选择测试数据的方法。。。如表1-1所示。

表1-1 编程题3的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | | | 理 论 结 果 |
| X | m | N |
| 用例1 | 0100 0110 1000 0000（4680） | 7 | 4 | 计算结果1101 0000 0000 0000 即D000 |
| 用例2 | 1101 0101 1000 0011（D583） | 16 | 1 | 输入错误（m值超范围） |
| 用例3 | 1101 0101 1000 0011（D583） | 13 | 5 | 输入错误（n值超范围） |

（b） 对应测试测试用例1的运行结果如图1-6所示。

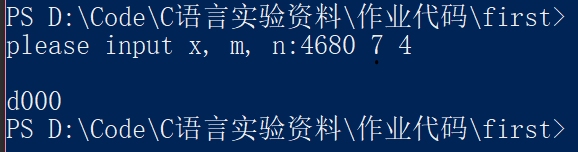


图1-6 编程题3的测试用例一的运行结果

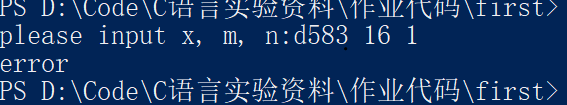
对应测试测试用例2的运行结果如图1-7所示。

图1-7 编程题3的测试用例二的运行结果

对应测试测试用例3的运行结果如图1-8所示。

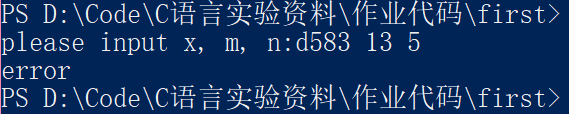


图1-8 编程题3的测试用例三的运行结果

说明上述的运行结果与理论分析吻合，验证了程序的正确性。

（3）IP地址通常是4个用句点分隔的小整数（即点分十进制）,如32.55.1.102。这些地址在机器中用无符号长整形表示。编写一个程序，以机器存储的形式读入一个互联网IP地址，对其译码，然后用常见的句点分隔的4部分的形式输出。例如：

整形676879571的二进制表示是00101000 01011000 01011100 11010011。按照8位一组可表示为40 88 92 211。由于CPU处理数据的差异，它的顺序是颠倒的，所以最终格式为211.92.88.40。

**解答：**

1. 解题思路：输入无符号整形数，并利用四组掩码提取出各需要序列，倒序后输出
2. 程序清单：

#include <stdio.h>

#include <math.h>

int main(void){

unsigned int mask[4] = {0xff000000, 0x00ff0000, 0x0000ff00, 0xff};

unsigned int ipbit[4];

unsigned int ip;

printf("please input a int ip address\n");

scanf("%d", &ip);

ipbit[0] = (ip & mask[0]) >> 24;

ipbit[1] = (ip & mask[1]) >> 16;

ipbit[2] = (ip & mask[2]) >> 8;

ipbit[3] = (ip & mask[3]);

printf("%u.%u.%u.%u", ipbit[3], ipbit[2], ipbit[1], ipbit[0]);

return 0;

}

1. 测试

对应截图：

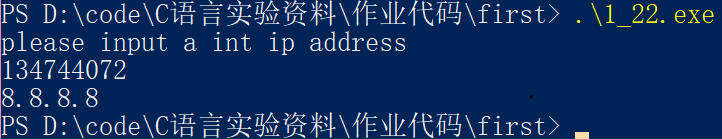


图1-9 编程题4的测试用例的运行结果

## 1.3 实验小结

在本次实验中，第一题主要考察对C语言一些基本语法的理解以及标准输出函数scanf的简单用法以及对位运算的使用，题目设置为改错形式，让我深刻认识到了在编写程序时尽量保证认真，在开始编程前做好计划的重要性。任何一个细节的错误都会对整个程序造成影响。

第二题考察对已知问题的另一种思路，可使用按位异或运算的性质或简单的加减法运算达到目的，打开了思路，提醒我们在面对一个问题时，应该多思考以找到更加高效，或更加简洁，亦或是单纯的更加有趣的方式，以锻炼创新性思维。

第三题考察对于ASCII码表的理解及简单的选择语句使用，利用在ASCII码表中大小写字母编号相差32的特性，用简单的方法完成大小写转换。。

第四题则考察对于位运算的理解与使用，在解决第四题时，使用了不同的方式，先利用数组构造出一个掩码，由按位乘以位权的方式得到整数型掩码再进行按位与及左移运算，避免考虑右移操作时不同机器补位方式不同的问题。

第五题考察了IP地址的构成的常识，以及对掩码的使用。

总之，由第一次C语言上机实验，感受到了在简单程序设计中应该考虑到的，平时未考虑到的诸如对错误输入的处理，可移植性等问题。感受到了编程之美，体会到了当程序正确运行之后的成就感，使我开始习惯编写程序的思维。