

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级：计算机类（二）2007班**

**学 号：U202017180**

**姓 名：吴彬杰**

**指导教师：万琳**

**报告日期：2020年10月18日**

**软件学院**

**目 录**

[**1 表达式和标准输入输出实验 1**](#_Toc404837920)

[1.1 实验目的 1](#_Toc404837921)

[1.2 实验内容 1](#_Toc404837922)

[1.3 实验小结 1](#_Toc404837923)

[**2 流程控制实验 2**](#_Toc404837924)

[2.1 实验目的 1](#_Toc404837921)

[2.2 实验内容 1](#_Toc404837922)

[2.3 实验小结 1](#_Toc404837923)

[**3 函数与程序结构实验 3**](#_Toc404837929)

[3.1 实验目的 1](#_Toc404837921)

[3.2 实验内容 1](#_Toc404837922)

[3.3 实验小结 1](#_Toc404837923)

[**4 编译预处理实验 4**](#_Toc404837934)

[4.1 实验目的 1](#_Toc404837921)

[4.2 实验内容 1](#_Toc404837922)

[4.3 实验小结 1](#_Toc404837923)

[**5 数组实验 5**](#_Toc404837938)

[5.1 实验目的 1](#_Toc404837921)

[5.2 实验内容 1](#_Toc404837922)

[5.3 实验小结 1](#_Toc404837923)

[**6 指针实验 6**](#_Toc404837943)

[6.1 实验目的 1](#_Toc404837921)

[6.2 实验内容 1](#_Toc404837922)

[6.3 实验小结 1](#_Toc404837923)

[**7 结构与联合实验 7**](#_Toc404837948)

[7.1 实验目的 1](#_Toc404837921)

[7.2 实验内容 1](#_Toc404837922)

[7.3 实验小结 1](#_Toc404837923)

[**8 文件实验 8**](#_Toc404837953)

[8.1 实验目的 1](#_Toc404837921)

[8.2 实验内容 1](#_Toc404837922)

[8.3 实验小结 1](#_Toc404837923)

[**参考文献 9**](#_Toc404837957)

# 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程序的编写方法。

（4）熟悉C语言程序的开发环境，并学会调试程序的方法。

## 1.2 实验内容

**1.2.1 程序改错与跟踪调试**

下面的实验1-1程序用来完成以下任务：

（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）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，拼成一个新的整数后输出。

在这个程序中存在若干语法和逻辑错误，要求先编译程序改正语法错误，再采用单步执行的方式调试程序找出逻辑错误。在单步执行程序的过程中，观察以下变量值：

1. 执行完c = 5/9 \* (f-32)，c的值为多少？
2. 执行完scanf(“%f”, &r)，r的值为多少？
3. 执行完newint = p&0xff00|k>>8， newint的值是多少？表达式k>>8的值是多少？

根据观察结果分析代码并修改程序，使之能够正确完成指定任务。

01 /\*实验1-1程序改错与跟踪调试题源程序\*/

02 #include<stdio.h>

03 #define PI 3.14159;

04 int main( void )

05 {

06 int f ;

07 short p, k ;

08 double c, r, s ;

09

10 /\* 任务1 \*/

11 printf("Input Fahrenheit: " ) ;

12 scanf(“%d”, f ) ;

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

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

15

16 /\* 任务2 \*/

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

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

19 s = PI \* r \* r;

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

21

22 /\* 任务3 \*/

23 k = 0xa1b2, p = 0x8432;

24 newint = p&0xff00|k>>8;

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

26 return 0;

27 }

**解答：**

1. 错误修改：

1）第03行去掉末尾的分号;

2）第12行读入f需要取引用&f;

3）第18行，r为double类型，格式化%lf;

4）第20行，printf不需要&，&s改为s,同时格式化为%lf;

5）第13行，修改表达式为c = (f - 32) \* 5 / 9;

6）第07行，将p，k改为unsigned short; 定义变量newint;

完整三个任务代码如下:

#include <stdio.h>

int main(void)

{

    short p, k;

    double c, r, s,f;

    /\* 任务1 \*/

    printf("Input Fahrenheit: ");

    scanf("%lf", &f);

    c = (f - 32) \* 5 / 9;

    printf( "\n%.0lf(F) = %.2lf(C)\n\n ", f, c);

    return 0;

}

#include <stdio.h>

#define PI 3.14159

int main(void)

{

    int f;

    short p, k;

    double c, r, s;

    /\* 任务2 \*/

    printf("input the radius r:");

    scanf("%lf", &r);

    s = PI \* r \* r;

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

    return 0;

}

#include <stdio.h>

int main(void)

{

    unsigned short p, k;

    int newint;

    /\* 任务3 \*/

    k = 0xa1b2, p = 0x8432;

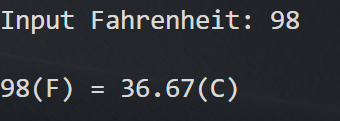
    newint = p & 0xff00 | k >> 8;

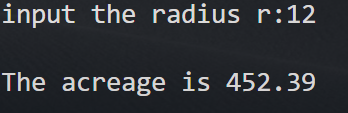
    printf("new int = %#x\n\n", newint);

    return 0;

}

(2)错误修改后的运行结果:







**2、程序分析与修改替换**

下面的实验1-2程序用“更相减损”法求m与n的最大公约数。

/\* 实验1-2程序分析与修改题源程序 \*/

01 #include <stdio.h>

02 int main( )

03 {

04 int m, n, k, p, i, d;

05 printf("input m, n \n");

06 scanf("%d%d", &m, &n);

07 if (m<n) /\* 交换m和n \*/

08 {

09 int t;

10 t = m;

11 m = n;

12 n = t;

13 }

14 k = 0;

15 while (m%2 == 0 && n%2 == 0) /\* m和n均为偶数 \*/

16 {

17 m /= 2; /\* 用2约简m和n \*/

18 n /= 2;

19 k++;

20 }

21 for (p = 1; i = 0; i<k; i++) p \*= 2; /\* 求p=2k \*/

22 while((d=m-n)!=n)

23 {

24 if(d>n) m = d;

25 else

26 {

27 m = n;

28 n = d;

29 }

30 }

31 d \*=p;

32 printf("the greatest common divisor : %d", d);

33 return 0;

34 }

1. 分析程序采用的“更相减损”法的算法步骤。

1）任意给定两个正整数；判断它们是否都是偶数。若是，则用2约简；若不是则执行第二步。

2）以较大的数减较小的数，接着把所得的差与较小的数比较，并以大数减小数。继续这个操作，直到所得的减数和差相等为止。约掉的若干个2与第二步中等数的乘积就是所求的最大公约数。

1. 按以下要求对源程序进行优化，提高程序的执行效率。

将交换m和n、判断一个整数是否是偶数、用2约简m和n、求p=2k等操作改成用位运算实现，并且m和n的交换不能够使用中间变量t，需要删除声明t的语句。

解答：替换后程序代码如下所示：

/\* 实验1-2程序分析与修改题源程序 \*/

#include <stdio.h>

int main()

{

    int m, n, k, p, i, d;

    printf("input m, n \n");

    scanf("%d%d", &m, &n);

    if (m < n) /\* 交换m和n \*/

    {

        m = m ^ n;

        n = m ^ n;

        m = m ^ n;

    }

    k = 0;

    while (m % 2 == 0 && n % 2 == 0) /\* m和n均为偶数 \*/

    {

        m >>= 1; /\* 用2约简m和n \*/

        n >>= 1;

        k++;

    }

    p = 1;

    p <<= k; /\* 求p=2k \*/

    while ((d = m - n) != n)

    {

        if (d > n)

            m = d;

        else

        {

            m = n;

            n = d;

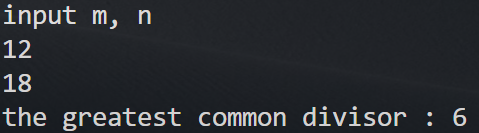
        }

    }

    d \*= p;

    printf("the greatest common divisor : %d", d);

    return 0;

}

1. 将else后的复合语句改用一条表达式语句。

解答：替换后程序代码如下所示：

/\* 实验1-2程序分析与修改题源程序 \*/

#include <stdio.h>

int main()

{

    int m, n, k, p, i, d;

    printf("input m, n \n");

    scanf("%d%d", &m, &n);

    if (m < n) /\* 交换m和n \*/

    {

        int t;

        t = m;

        m = n;

        n = t;

    }

    k = 0;

    while (m % 2 == 0 && n % 2 == 0) /\* m和n均为偶数 \*/

    {

        m >>= 1; /\* 用2约简m和n \*/

        n >>= 1;

        k++;

    }

    p = 1;

    p <<= k; /\* 求p=2k \*/

    while ((d = m - n) != n)

    {

        if (d > n)

            m = d;

        else

        {

            m = n , n = d;

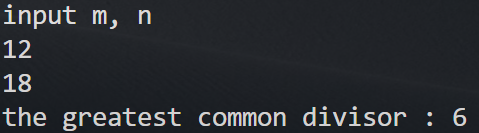
        }

    }

    d \*= p;

    printf("the greatest common divisor : %d", d == 0 ? 1 : d);

    return 0;

}

**3、程序设计**

（1）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，输入Ctrl+Z程序结束。要求①用条件表达式；②字符的输入输出用getchar和putchar函数。程序应能循环接受用户的输入，直至输入Ctrl+Z程结束。例如：

A （键盘输入）

a

^Z （键盘输入）

解答：

1. 算法流程图如图1-1所示。



图1-1 编程题1的程序流程图

1. 源程序清单

#include <stdio.h>

int main(void)

{

    char c;

    c = getchar();

    while (c != EOF)

    {

        if ((c >= 'A') && (c <= 'Z'))

            putchar(c + 32);

        else

            putchar(c);

        c = getchar();

    }

    return 0;

}

1. 测试数据

根据大小写字母、其他符号分别选取测试数据，如表1-1所示：

表1-1 编程题1的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | A | a | a |
| 用例2 | c | c | c |
| 用例3 | ; | ; | ; |

（2）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。要求：①检查m和n的范围；②x的值以十六进制输入，m和n以十进制输入；③结果以十六进制输出。

1. 1）算法流程图如图1-2所示。



图1-2 编程题2的程序流程图

2）源程序清单

#include<stdio.h>

int main()

{

    unsigned short x, m, n;

    scanf("%hu%hu%hu",&x,&m,&n);

    if (m > 15)

    {

        puts("m out of range!");

        return 1;

    }

    if (n < 1 || n > 16 - m)

    {

        puts("n out of range!");

        return 1;

    }

    x &= (0xFFFF << m);

    x <<= 16 - n - m;

    printf("%X", x);

    return 0;

}

3）测试数据如表1-2所示：

表1-2 编程题2的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入x,m,n | 理论结果 | 运行结果 |
| 用例1 | 12 3 5 | 800 | 800 |
| 用例2 | 11 3 3 | 2000 | 2000 |
| 用例3 | 2 4 1 | 0 | 0 |

（3）IP地址通常是4个用句点分隔的小整数（即点分十进制），但这些地址在机器中是用一个无符号长整型数表示的。例如3232235876，其机内二进制表示就是11000000 10101000 00000001 01100100，按照8位一组用点分开，该IP地址就写成192.168.1.100。

读入无符号长整型数表示的互联网IP地址，对其译码，以常见的点分十进制形式输出。要求循环输入和输出，直至输入Ctrl+Z结束。

1. 1）算法流程图如图1-3所示。



图1-3 编程题3的程序流程图

2）源程序清单

#include <stdio.h>

int main(void)

{

    union ipaddr

    {

        unsigned long i;

        unsigned char ch[4];

    };

    union ipaddr n;

    while (scanf("%lu", &n.i) != EOF)

    {

        printf("%hhu.%hhu.%hhu.%hhu\n",n.ch[3],n.ch[2],n.ch[1],n.ch[0]);

    }

    return 0;

}

1. 测试数据如表1-3所示：
2. 表1-3 编程题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例1 | 3232235876 | 192.168.1.100 | 192.168.1.100 |
| 用例2 | 0 | 0.0.0.0 | 0.0.0.0 |
| 用例3 | 123456789 | 7.91.205.21 | 7.91.205.21 |

上述的运行结果与理论分析吻合，验证了程序的正确性。

4. 选做题

某加密算法对数据按字节进行加密，具体为：对字节的8个二进制位从右向左用0～7编号，先将0、2、4位分别与1、3、5位两两对应交换，接着对0～5位进行循环左移（左边移出的位接在右边），循环左移的位数有6、7两位的值决定。例如，若6、7位组成的二进制数为01，则将0～5位左移1位，最后得到加密结果，如图1-1所示。

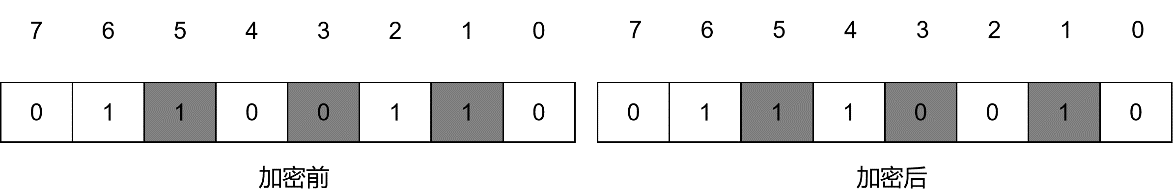


图1-1 加密示意图

输入一行明文字符串，按该算法进行加密后输出密文。例如，输入“abcd”，则输出“dbfp”。

**解答：**

1. 算法思路

主程序：

循环读入字符，调用encrypt函数进行逐字加密；

Encrypt函数：

拆分字符ASCII码为二进制，存入数组x；

按照题意进行换位、移位操作；

重新合并为十进制；

2）程序清单

#include <stdio.h>

#include <string.h>

char encrypt(char c);

int x[8];

int main()

{

    char tmp;

    scanf("%c", &tmp);

    while (tmp != EOF)

    {

        printf("%c", encrypt(tmp));

        scanf("%c", &tmp);

    }

}

char encrypt(char c)

{

    int inte = (int)c;

    memset(x, 0, sizeof(x));

    x[7] = inte / 128;

    x[6] = inte % 128 / 64;

    x[5] = inte % 64 / 32;

    x[4] = inte % 32 / 16;

    x[3] = inte % 16 / 8;

    x[2] = inte % 8 / 4;

    x[1] = inte % 4 / 2;

    x[0] = inte % 2;

    int temp;

    temp = x[0];

    x[0] = x[1];

    x[1] = temp;

    temp = x[2];

    x[2] = x[3];

    x[3] = temp;

    temp = x[4];

    x[4] = x[5];

    x[5] = temp;

    int num = x[6] + 2 \* x[7];

    for (int cou = 0; cou < num; cou++)

    {

        int temp = x[5];

        x[5] = x[4];

        x[4] = x[3];

        x[3] = x[2];

        x[2] = x[1];

        x[1] = x[0];

        x[0] = temp;

    }

    char ret = (char)(x[0] + x[1] \* 2 + x[2] \* 4 + x[3] \* 8 + x[4] \* 16 + x[5] \* 32 + x[6] \* 64 + x[7] \* 128);

    return ret;

}

解法二：

#include <stdio.h>

#include <string.h>

// Rotate union ch x according to rule.

#define ROL(x) for(int i=0, temp; i < x->hi2; i++)\

{\

    temp = x->r0;\

    x->r0 = x->r5;\

    x->r5 = x->r4;\

    x->r4 = x->r3;\

    x->r3 = x->r2;\

    x->r2 = x->r1;\

    x->r1 = temp;\

}

union ch {

    char orig;

    struct {

        unsigned char lo6:6;

        unsigned char hi2:2;

    };

    struct {

        unsigned char r0:1;

        unsigned char r1:1;

        unsigned char r2:1;

        unsigned char r3:1;

        unsigned char r4:1;

        unsigned char r5:1;

        unsigned char :2;

    };

};

void encrypt(union ch \*);

int main(void)

{

    union ch ch[BUFSIZ];

    int len, i;

    gets((char \*)ch);

    len = strlen(&ch->orig);

    for (i = 0; i < len; i++)

        encrypt(&ch[i]);

    puts(&ch->orig);

    return 0;

}

void encrypt(union ch \*ch)

{

    int temp;

    // Swap bit0 and bit1

    temp = ch->r0;

    ch->r0 = ch->r1;

    ch->r1 = temp;

    // Swap bit2 and bit3

    temp = ch->r2;

    ch->r2 = ch->r3;

    ch->r3 = temp;

    // Swap bit4 and bit5

    temp = ch->r4;

    ch->r4 = ch->r5;

    ch->r5 = temp;

    ROL(ch);

}

解法三：

#include <stdio.h>

int main()

{

    char c;

    while ((c = getchar()) != EOF && c != '\n')

    {

        unsigned short step = c >> 6;

        unsigned short c1, c2;

        c1 = (c & 0x2a) >> 1 | (c & 0x15) << 1 | step << 6;

        if (!step)

            c2 = c1;

        else

            c2 = (c1 & 0x1f) << 1 | (c1 & 0x20) >> 5 | 0x40;

        putchar(c2);

    }

    return 0;

}

3）测试数据

输入：abcd

预期输出：dbfp

实际输出：dbfp

上述的运行结果与理论分析吻合，验证了程序的正确性。

## 1.3 实验小结

（1）编写程序经验来自于实践。将算法转变为真正的代码过程中会出现诸多的细节问题，源于实践，需要实践来解决。

（2）位运算一定程度上可以利用底层运算提高运算效率，但也会造成代码可理解性降低，应合理使用。