

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 软件工程2003班**

**学 号： U202010851**

**姓 名： 侯皓斐**

**指导教师： 唐赫**

**报告日期： 2020.10.20**

**软件工程**

**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

# 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 3.14159

2) 第6行的p后面赋值正好使short符号位为1，为保证右移补0，正确形式为：

unsigned short p, k ;

3) 第10行的使用scanf读入需要对变量取地址，正确形式为：

scanf("%d", &f ) ;

4) 第11行的5/9默认为int的整除，结果为0错误，正确形式为：

c = 5.0 / 9.0 \* (f-32);

5) 第15行的使用scanf读入double型变量要使用%lf，正确形式为：

scanf("%lf", &r);

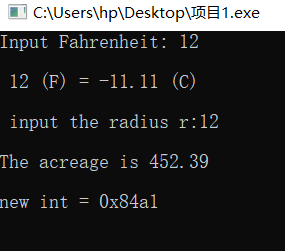
6) 第17行的输出不需要取地址，正确形式为：

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

7) 第20行的newint未被定义，正确形式为：

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

（2）错误修改后运行结果：



**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：**

通过数值的运算可以实现两个数没有第三个变量的交换：a=b-a，求出a与b之间的距离存放在a中；b=b-a，求出原来的a存入b中；a=a+b，从而借助原来的a和a，b间距离求出b存入a中。替换后的程序如下所示：

#include<stdio.h>

void main( )

{

int a, b;

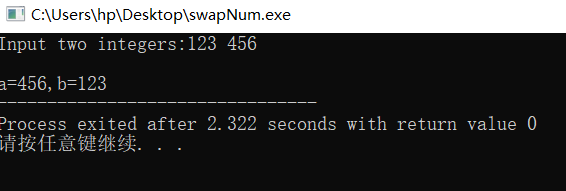
printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

a=b-a ；b=b-a；a=a+b；

prinf(“\na=%d,b=%d”,a,b);

}



**1.2.3 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程序结束。

**解答：**

1） 算法流程如图1.1所示。

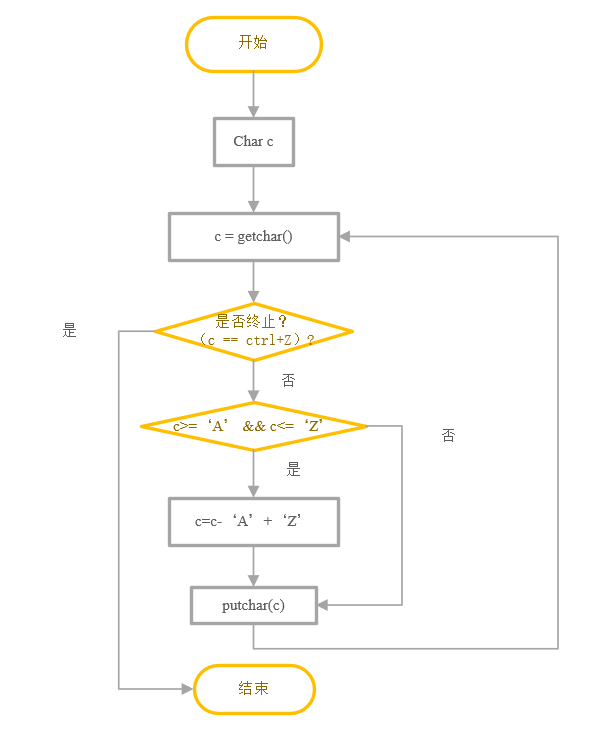


图1-1 编程题1的程序流程图

2）源程序清单

#include<stdio.h>

int main () {

char c;

while((c = getchar()) != EOF) {

if( c >= 'A' && c <= 'Z' )

c = c - 'A' + 'a';

putchar(c);

}

return 0;

}

3）测试

（a） 测试数据：

C

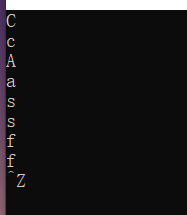
A

s

f

Ctrl+Z

（b） 对应测试数据的运行结果截图



（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：

1.输入x，m，n，为了方便分析测试结果，x的输入采用16进制

2.如果0 ≤ｍ≤ 15, 1 ≤ ｎ≤ 16-ｍ，转2.1，否则转3.

2.1 首先x>>m，将要处理的n位移动到最右；

2.2 再将上一步的结果左移16 - n位，即： 答案为(x >> m) << (16 - n)

2.3 用16进制输出结果并转4.

3. 显示输入错误信息；

4. 结束

2）程序清单

#include<stdio.h>

void main() {

unsigned short x,m,n;

printf("输入x（16进制）、m（0~15）和n（1~16-m）：\n ");

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

if ( 0 <= m && m <= 15 && 1 <= n && n <= 16-m ) /\*判断m、n的值是否在合理范围内\*/

printf("ans=%hx\n",(x >> m) << (16 - n));

else printf("输入错误!\n");

}

3）测试

（a） 测试数据：

根据题目要求以及各种可能出现的错误情况构造数据如表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 | 运 行 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 | D000 或 截图 |
| 用例2 | 1101 0101 1000 0011（D583） | 16 | 1 | 输入错误（m值超范围） | 输入错误! |
| 用例3 | 1101 0101 1000 0011（D583） | 13 | 5 | 输入错误（n值超范围） | 输入错误! |

（b） 对应测试测试用例1的运行结果如图1-2所示。

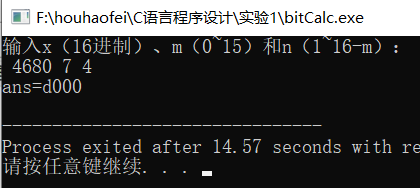


图1-2 编程题3的测试用例一的运行结果

对应测试测试用例2的运行结果如图1-3所示。

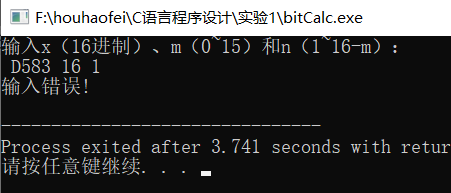


图1-3 编程题3的测试用例二的运行结果

对应测试测试用例3的运行结果如图1-4所示。

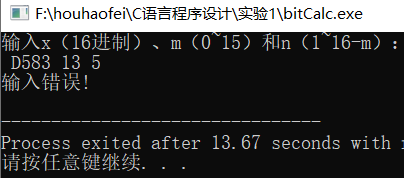


图1-4 编程题3的测试用例三的运行结果

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），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。

**解答：**

1) 解题思路：

1.输入x，如果未输入x，得到 Ctrl+Z，跳转至4结束。

2.定义opt = (1 << 8) – 1,ans[4], i= 1

3.如果i<4执行以下步骤

3.1 首先取出x前八位，ans[i] = x & opt;

3.2 然后x>>8，将要处理的8位移动到最右；

3.3 i = i +1

3. 倒序加结构输出ans数组

4. 结束

2）程序清单

#include <stdio.h>

int main () {

unsigned long long x = 0;

while(scanf("%lld",&x) != EOF) {

unsigned long long opt = (1 << 8) - 1;

unsigned long long ans[4];

for(int i = 0; i < 4; i++) {

ans[i] = x & opt;

x >>= 8;

}

for(int i = 3; i >= 1; i--)

printf("%lld.",ans[i]);

printf("%lld\n",ans[0]);

}

return 0;

}

3）测试

（a） 测试数据：

根据题目要求以及各种可能出现的错误情况构造数据如表1-2所示。

表1-2 编程题4的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 | 运 行 结 果 |
| X |
| 用例1 | 3232235876 | 192.168.1.100 | 192.168.1.100 |
| 用例2 | 66322345376 | 113.30.173.160 | 113.30.173.160 |
| 用例3 | 3232235876  66322345376  187232376 | 192.168.1.100  113.30.173.160  11.40.240.120 | 192.168.1.100  113.30.173.160  11.40.240.120 |

（b） 对应测试测试用例1的运行结果如图1-2所示。

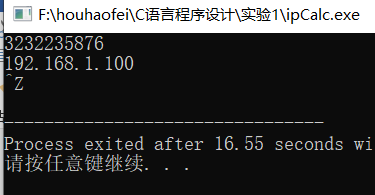


图1-5 编程题4的测试用例一的运行结果

对应测试测试用例2的运行结果如图1-3所示。



图1-6 编程题4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1-4所示。

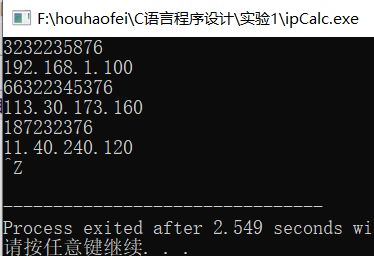


图1-7 编程题4的测试用例三的运行结果

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。

## 1.3 实验小结

主要叙述实验过程中遇到的问题，如何解决的，通过分析、结果问题后的体会。

在实验1源程序改错中，发现在devC++环境下无法执行调试环节，在询问老师后了解了必须在debug环境下且创建项目的情况下才能进行调试。过程中有几个bug是比较难以想到的，不断分析输出结果，调试看 k >> 8 的值，突然回顾到了右移运算的补位性质，最终解决了所有改错。

在编程题1中不知道如何不使用第三个变量交换两个数，和同学讨论后，商量出了一种可行的方案。a=b-a，求出a与b之间的距离存放在a中；b=b-a，求出原来的a存入b中；a=a+b，从而借助原来的a和a，b间距离求出b存入a中。

在编程题2中，如何判断程序结束的“ctrl + Z”是个大问题。通过百度搜索相关资料了解了循环应写的形式。在画流程图的过程中，通过百度一些基本功能学习了Visio软件。