实验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 源程序改错题

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。要求在计算机上对这个例子程序进行调试修改，使之能够正确完成指定任务。

**解答**

#include<stdio.h>

#define PI 3.14159; // #define PI 3.14159

voidmain( void ) // void main(void)

{

int f ;

short p, k ;

double c , r , s ;

/\* for task 1 \*/

printf(“Input Fahrenheit:” ) ;

scanf(“%d”, f ) ; // scanf(“%d”, &f);

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

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

/\* for task 2 \*/

printf("input the radius r:");

scanf("%f", &r); // scanf(“%lf”, &r);

s = PI \* r \* r;

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

// printf(“\nThe acreage is %.2f\n\n”, s);

/\* for task 3 \*/

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

scanf("%x %x", &k, &p ); // scanf(“%hx %hx”, &k, &p);

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

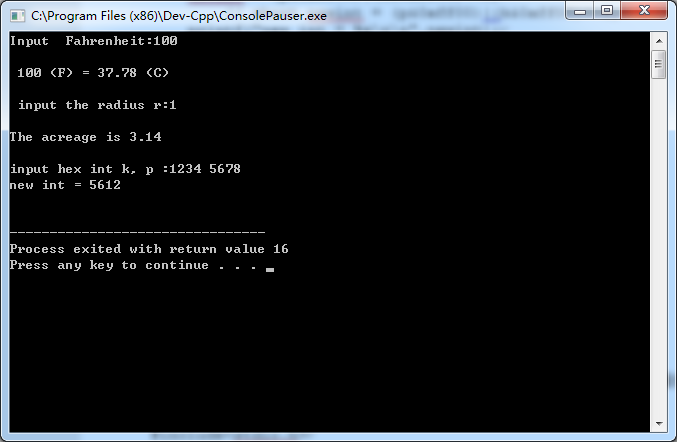
// short newint = (p&0xff00)|(k&0xff00)>>8;

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

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

}

**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，请改用不用第三个变量的交换法实现。

**解答**

#include<stdio.h>

void main( )

{

int a, b, t;

printf(“Input two integers:”);

scanf(“%d %d”,&a,&b);

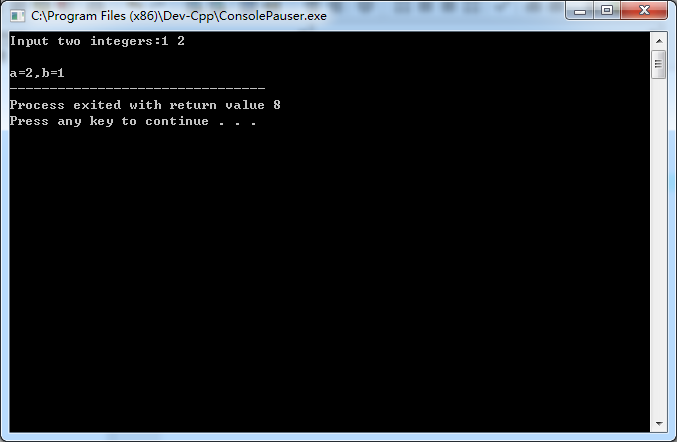
t=a, a=b, b=t;

// b += a, a = b - a, b -= a;

printf(“\na=%d,b=%d”,a,b);

}

**替换后运行结果：**

****

1.2.3 编程设计

（1）编写一个程序，输入无符号短整型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。

1）算法流程：

1. 输入数据组数；

2. 对每一组数据：

1. 输入x, m, m；

2. 根据m, n制作mask；

3. 输出x, mask与运算后结果；

4. 下一组数据；

**源程序清单：**

#include "stdio.h"

int main(void) {

int total\_count; // 数据总组数

unsigned short x, m, n; // 输入数据单元

unsigned short mask = 0; // mask单元

unsigned short i; // 循环控制变量

// 输入数据组数 - 并限制其不能小于0

scanf("%d", &total\_count);

if (total\_count < 0) { puts("negative input!"); return 0; }

for (; total\_count > 0; --total\_count) {

// 一行输入一行输出

scanf("%hu %hu %hu", &x, &m, &n);

// 限制m, n范围

if (m>15) { puts("m out of range!"); ++total\_count; continue; }

if (n>16-m) { puts("n out of range!"); ++total\_count; continue; }

// 不需要判断它们是否小于0，因为它们都是无符号的

// 制作mask

mask = 1U << m; // mask起点

for (i=0; i<n; ++i) {

mask |= mask << 1; // 一直拖到终点

} // endfor

// 一行输入一行输出

printf("%hu\n", (x&mask)<<(16-m-n));

} // endif

return 0;

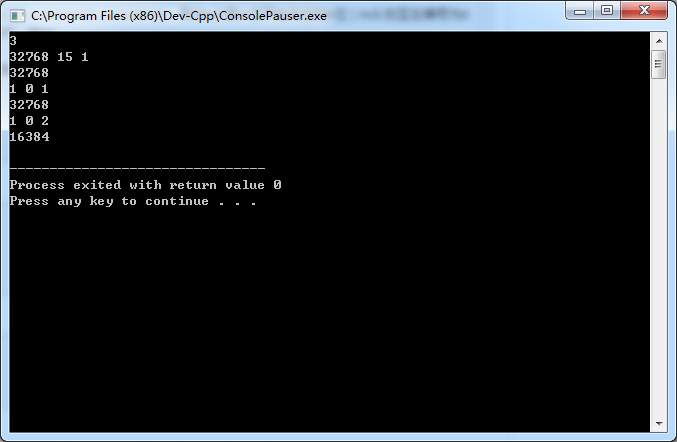
}

**测试：**

（a）测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例 | 程序输入 | | | 理论结果 |
| x | m | n |
| 用例1 | 32768 | 15 | 1 | 32768 |
| 用例2 | 1 | 0 | 1 | 32768 |
| 用例3 | 1 | 0 | 2 | 16384 |

（b）运行结果：



（2）IP地址通常是4个用句点分隔的小整数，如32.55.1.102。这些地址在机器中用无符号长整型表示。编写一个程序，以机器存储的形式读入一个32位的互联网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。

1）算法流程：

1. 制作静态的mask；

2. 输入数据组数；

3. 对每一组数据：

1. 输入一个IP源数据；

2. 将IP数据要输出的部分移到mask的有效位；

3. 输出结果；

4. 下一组数据；

**源程序清单：**

#include "stdio.h"

#define LEN 32 // 无符号长整型位数（made static）

#define MASK 255UL // 即 00000000 00000000 00000000 11111111

int main(void) {

int total\_count; // 数据总组数

unsigned long current; // 存储当前输入数据

char i; // 循环控制变量

scanf("%d", &total\_count); // 限制 - 输入的数据组数必须大于0

if (total\_count < 0) { puts("negative count!"); return 0; }

for (; total\_count > 0; --total\_count) {

// 每次循环输入一次 - 一行输入一行输出

scanf("%ld", &current);

// 笨办法 - 从左至右依次输出

printf("%ld.", (current&MASK));

// 输出一次，加工一次数据 - mask不动

current >>= 8;

printf("%ld.", (current&MASK));

current >>= 8;

printf("%ld.", (current&MASK));

current >>= 8;

printf("%ld\n", (current&MASK));

}

return 0;

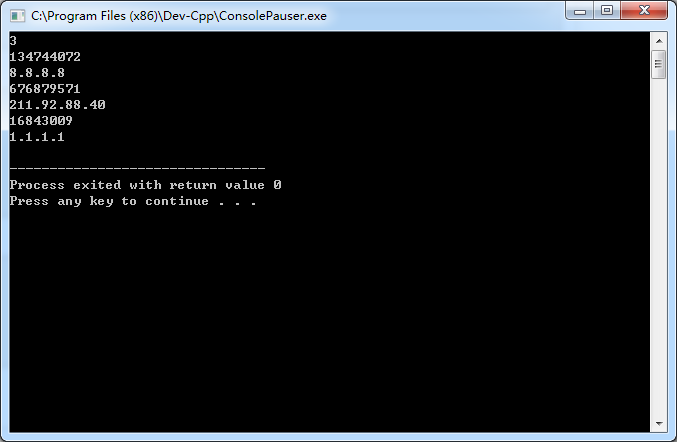
}

**测试：**

（a）测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| IP源数据 |
| 用例1 | 134744072 | 8.8.8.8 |
| 用例2 | 676879571 | 211.92.88.40 |
| 用例3 | 16843009 | 1.1.1.1 |

（b）运行结果



1.3 自设题

（1）自设实验题目，实型数的浮点表示误差的简单验证。

（2）实验目的：通过设计实验程序，理解实数的浮点数表示上的误差，体会浮点数不宜用于复杂计算场合（例如金融计算、科学计算等）的含义。

**源程序清单**

#include "stdio.h"

void main(void) {

puts("Here's a third: ");

float pocket = 1./3;

printf("%f\n", pocket);

puts("\nnow times it with 7 and then subtracts it with 2 like craaaaazy!");

puts("(Note: in analog env, a third will still be a third)");

unsigned long times\_to\_repeat = -1;

printf("\nHow many times should I repeat this operation? ");

scanf("%ld", &times\_to\_repeat);

// start!

for (; times\_to\_repeat != 0; --times\_to\_repeat) {

pocket \*= 7;

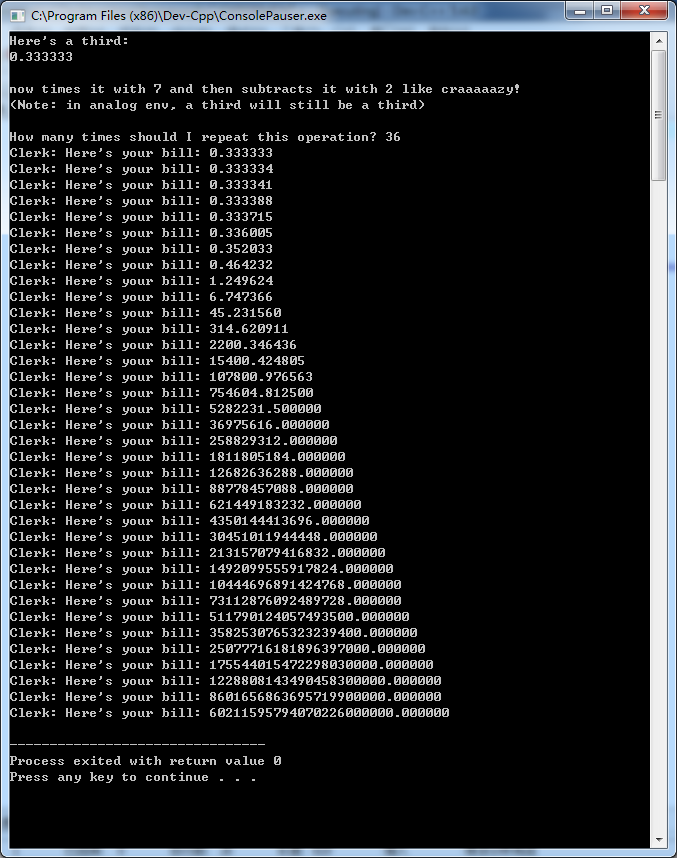
pocket -= 2;

printf("Clerk: Here's your bill: %f\n", pocket);

}

}

**运行结果截图**

****

**实验结论：**

浮点数不适合复杂计算场合。

1.3 实验小结

通过本次实验，我切身体会了从无到有、完整地编写C语言程序的过程。发现理论与实践还是有一定差距的，以后还需要多加练习。