# 实验2 流程控制实验

## 2.1 实验目的

（1）掌握复合语句、if语句、switch语句的使用，熟练掌握for、while、do-while三种基本的循环控制语句的使用，掌握重复循环技术，了解转移语句与标号语句。

（2）练习循环结构for、while、do-while语句的使用。

（3）练习转移语句和标号语句的使用。

（4）使用Turbo C 2.0集成开发环境中的调试功能：单步执行、设置断点、观察变量值。

## 2.2 实验内容

**2.2.1 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 void main(void)

3 {

4 int i,n,s=1;

5 printf("Please enter n:");

6 scanf("%d",n);

7 for(i=1,i<=n,i++)

8 s=s\*i;

9 printf("%d! = %d",n,s);

10 }

**解答：**

（1）错误修改：

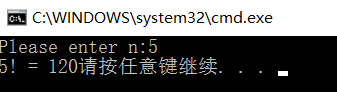
1) 第6行的scanf("%d",n);，正确形式为：

scanf("%d",&n);

2) 第7行的for(i=1,i<=n,i++)，正确形式为：

for(i=1;i<=n;i++)

（2）错误修改后运行结果：



**2.2.2 源程序修改替换**

（1）修改第1题，分别用while和do-while语句替换for语句。

**解答：**

将for (i = 1; i <= n; i++) s = s\*i;

替换为

i = 1;

while (i <= n)

{

s = s\*i;

i++;

}替换后的程序如下所示：

#include <stdio.h>

void main(void)

{

int i, n, s = 1;

printf("Please enter n:");

scanf("%d", &n);

i = 1;

while (i <= n)

{

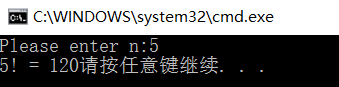
s = s\*i;

i++;

}

printf("%d! = %d", n, s);

}



**解答：**

#include <stdio.h>

void main(void)

{

int i, n, s = 1;

printf("Please enter n:");

scanf("%d", &n);

i = 1;

do

{

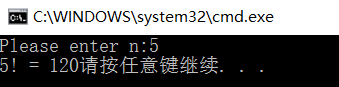
s = s\*i;

i++;

} while (i <= n);

printf("%d! = %d", n, s);

}



（2）修改第1题，输入改为“整数S”，输出改为“满足n！≥S的最小整数n”。例如输入整数40310，输出结果为n=8。

**解答：**

#include <stdio.h>

void main(void)

{

int i, n, s = 1;

int S;

printf("Please enter S:");

scanf("%d", &S);

i = 1;

while (s < S)

{

s = s\*i;

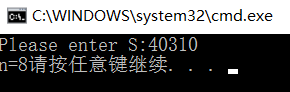
i++;

}

i--;

printf("n=%d", i);

}



**2.2.3 程序设计**

（1）求方程的近似根

**解答：**

1） 算法流程如图2.1所示。



图2-1 编程题1的程序流程图

2）源程序清单

#include<stdio.h>

#include<math.h>

int a, b, c, d;

float Solve(float a, float b, float c, float d);

int main()

{

a = 3; b = -4; c = -5; d = 13;

printf("%lf\n", Solve(a, b, c, d));

return 0;

}

float Solve(float a, float b, float c, float d)

{

float x = 1, x0=2, f, f1;

while (fabs(x - x0) >= 1e-5)

{

x0 = x;

f = ((a\*x0 + b)\*x0 + c)\*x0+d;

f1 = (3 \* a\*x0 + 2 \* b)\*x0 + c;

x = x0 - f / f1;

}

return x;

}

3）测试



（2）打印如下杨辉三角形。

1 /\*第0行 \*/

1 1 /\*第1行 \*/

1 2 1 /\*第2行 \*/

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

1 9 36 84 126 126 84 36 9 1

每个数据值可以由组合计算（表示第i行第j列位置的值），而的计算如下：

 (i=0,1,2,…)

 (j=0,1,2,3,…,i)

本程序中为了打印出金字塔效果，要注意空格的数目。一位数之间是3个空格，两位数之间有2个空格，3位数之间只有一个空格，程序编制过程中要注意区分。

**解答：**

1） 算法流程如图2.1所示。



图2-2 编程题2的程序流程图

2）源程序清单

#include<stdio.h>

void YangHui(int n);

void CalculateYH(int n);

void PrintSpace(int n);

int DigitNum(int n);

int a[13][13];

int main()

{

int n ;

scanf("%d", &n);

while (n != 0)

{

YangHui(n);

printf("\n");

scanf("%d", &n);

}

return 0;

}

void CalculateYH(int n)

{

int i, j;

for (i = 1; i <= n; i++)

{

a[i][1] = 1; /\*第一列全置为一\*/

}

for (i = 2; i <=n; i++)

{

for (j = 1; j <= i; j++)

{

a[i][j] = a[i - 1][j - 1] + a[i - 1][j];

}

}

}

void YangHui(int n)

{

CalculateYH(n);

int i, j;

for (i = 1; i<=n; i++) /\*输出杨辉三角\*/

{

PrintSpace(2 \* (n - i));

for (j = 1; j <= i; j++)

{

if ((j == i))

{

printf("%d", a[i][j]);

}

else

{

printf("%d", a[i][j]);

PrintSpace(4 - DigitNum(a[i][j + 1]));

}

}

printf("\n");

}

}

void PrintSpace(int n)

{

for (int i = 1; i <= n; i++)

{

printf(" ");

}

}

int DigitNum(int n)

{

int sum=0;

while (n)

{

sum++;

n /= 10;

}

return sum;

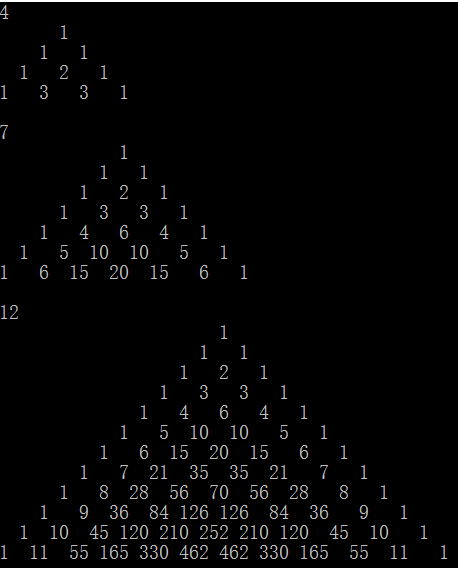
}

3）测试

（a） 测试数据：

4 7 12

（b） 对应测试数据的运行结果截图



（3）正整数逆转，输入任意正整数逆转。

输入：多组输入，每组一个正整数m，输入为0结束

输出：每行对应一个输出

**解答：**

1) 解题思路：



图2-3 编程题3的程序流程图

2）程序清单

/\*整数翻转\*/

#include<stdio.h>

void NumReverse(int n);

int main()

{

int n;

scanf("%d", &n);

while (n != 0)

{

NumReverse(n);

scanf("%d", &n);

}

return 0;

}

void NumReverse(int n)

{

int ModNum = 0;// 余数

int result = 0;

while (n / 10 != 0) //一位数跳出循环

{

ModNum = n % 10;

n = n / 10;

result = result \* 10 + ModNum;

}

result=result \* 10 + n;

printf("%d\n", result);

}

3）测试

（a） 测试数据

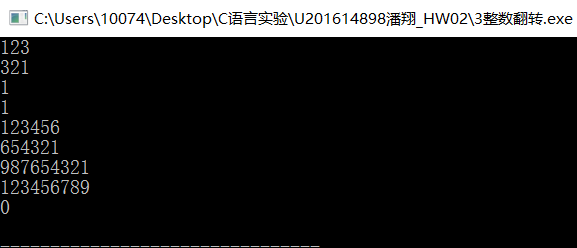
123

1

123456

987654321

（b） 对应测试测试用例1的运行结果如图1-2所示。



**（4）个人所得税。**

**解答：**

1. 解题思路：

使用多层if实现分段函数

1. 程序清单

/\*个人所得税\*/

#include<stdio.h>

#include<stdbool.h>

#include<math.h>

void CountTax(double salary);

bool IsZero(double salary);

int main()

{

double salary;

scanf("%lf", &salary);

while (!IsZero(salary))

{

CountTax(salary);

scanf("%lf", &salary);

}

return 0;

}

void CountTax(double salary)

{

//printf("%.6lf\n", salary);

int n;

double Tax;

if (salary <= 1000)

{

n = 1;

}

if (1000 < salary&&salary <= 2000)//50

{

n = 2;

}

if (2000 < salary&&salary <= 3000)//100

{

n = 3;

}

if (3000 < salary&&salary <= 4000)//150

{

n = 4;

}

if (4000 < salary&&salary <= 5000)//200

{

n = 5;

}

if (5000 < salary)

{

n = 6;

}

switch (n)

{

case 1:Tax = 0; break;

case 2:Tax = (salary-1000)\*0.05; break;

case 3:Tax = 50+(salary - 2000)\*0.10; break;

case 4:Tax = 150 + (salary - 3000)\*0.15; break;

case 5:Tax = 300 + (salary - 4000)\*0.20; break;

case 6:Tax = 500 + (salary - 5000)\*0.25; break;

default:

break;

}

printf("%.6lf\n", Tax);

}

bool IsZero(double salary)

{

if (fabs(salary - 0) < 0.000001)

{

return true;

}

else

{

return false;

}

}

3）测试

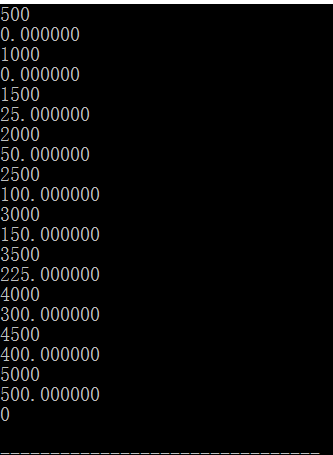
（a） 测试数据：

叙述选择测试数据的方法使数据覆盖尽量广泛 各节相同和不同如表1-2所示。

表1-1 编程题4的测试数据

|  |  |  |
| --- | --- | --- |
| 测试  用例 | 程 序 输 入 | 理 论 结 果 |
| X |
| 用例1 | 500 | 0.000000 |
| 用例2 | 1000 | 0.000000 |
| 用例3 | 1500 | 25.000000 |
| 用例4 | 2000 | 50.000000 |
| 用例5 | 2500 | 100.000000 |
| 用例6 | 3000 | 150.000000 |
| 用例7 | 3500 | 225.000000 |
| 用例8 | 4000 | 300.000000 |
| 用例9 | 4500 | 400.000000 |
| 用例10 | 5000 | 500.000000 |

（b） 对应测试测试用例的运行结果如图1-2所示。



**（5）个人所得税。**

**解答：**

1. 解题思路：



图2-4 编程题5的程序流程图

1. 程序清单

#include<stdio.h>

void fun();

int main()

{

int N;

scanf("%d", &N);

getchar();

//char input[100] = " ";

//fgets(input, 100, stdin);

for (int i = 1; i <= N; i++)

{

fun();

}

}

void fun()

{

char c = 0, flag = 0;

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

{

if (c == ' ')

{

if (flag == 0)

{

flag = 1;

}

else

{

continue;

}

}

else

{

flag = 0;

}

printf("%c", c);

}

printf("\n");

}

3）测试

（a） 测试数据：

选择不同的空格数目和位置

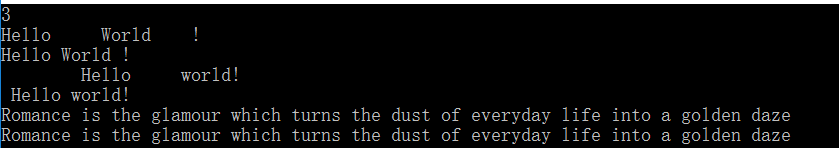
3

Hello World !

Hello world!

Romance is the glamour which turns the dust of everyday life into a golden daze

（b） 对应测试测试用例的运行结果如图1-2所示。



## 2.3 自设题

**（1）** 自设实验题目：

描述：

给定一个由不同的小写字母组成的字符串，输出这个字符串的所有全排列。 我们假设对于小写字母有'a' < 'b' < ... < 'y' < 'z'，而且给定的字符串中的字母已经按照从小到大的顺序排列。

输入：

输入只有一行，是一个由不同的小写字母组成的字符串，已知字符串的长度在1到6之间。

输出：

输出这个字符串的所有排列方式，每行一个排列。要求字母序比较小的排列在前面。字母序如下定义：  
  
已知S = s1s2...sk , T = t1t2...tk，则S < T 等价于，存在p (1 <= p <= k)，使得  
s1 = t1, s2 = t2, ..., sp - 1 = tp - 1, sp < tp成立。

其中, 数字项和日期项单位价格左对齐, 美元数量最大取值为9999.99

**（2）** 实验目的：结合算法设计，熟悉流程实现方法

**（3）**实验程序：

#include <stdio.h>

#include <string>

string array;

int len;

void swap(int a, int b) {

char temp = array[b];

array[b] = array[a];

array[a] = temp;

}

void reverse(int a, int b) {

while (a < b) {

swap(a, b);

a++;

b--;

}

}

void permutation() {

int i,j;

while (true) {

printf("%s\n",array.c\_str());

for (i = len - 2; i >= 0; i--) {

if (array[i] < array[i+1]) {

break;

}

if (i == 0) {

return;

}

}

for (j = len - 1; j > i ; j--) {

if (array[j] > array[i]) {

break;

}

}

swap(i,j);

reverse(i+1,len-1);

}

}

int main(){

char data[10];

while(scanf("%s",&data) != EOF ){

array = data;

len = array.size();

if (len == 1) {

printf("%s\n",array.c\_str());

printf("\n");

continue;

}

permutation();

printf("\n");

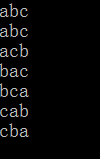
}

return 0;

}

**（4）**实验用例**：**

abc



**（5）**实验结论：

字典序输出的流程控制，需要在每一层逻辑清楚

## 2.4 实验小结

**（1）**对于测试格式的严格要求，当测试不通过的时候检查空格换行等错误

**（2）**对于scanf后getchar()吞回车符的应用

**（3）**学习使用测试用例和有效的测试方法。

**（4）**注重细节，培养习惯才能在工程代码中少出错。

**（5）**结合算法和OJ题目，提高程序编写能力。