# 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**

**1．源程序改错题**

下面是计算s=n!的源程序，在这个源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。例如，8！=40320。

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：**本程序共存在3处错误，分析如下:

(1) 第四行ints=1;  
错误原因:  在计算阶乘的时候，由于阶乘值会上升很快，所以不建议使用int类型，可以改为long或者double型。  
改正:改为long型变量。

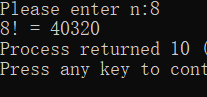
(2) 第六行scanf("%d" ,n);  
错误原因: scanf 函数获取输入值时，引用的是一个变量地址，而不是变量本身。  
改正:  scanf("%d" ,&n);

(3) 第七行for(i=1,i<=n,i++)

错误原因：for后括号里的各项应该用；隔开

更正：for(i=1;i<=n;i++)

(4) } return 0;  
错误原因: return 返回值应该在main函数内部改正: return 0;}

运行截图如下  
**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。

#include <stdio.h>

int main()  
{  
int i=1,n,s=1;  
printf("Please enter n:" );

scanf("%d" ,&n);

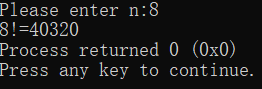
while(i<=n){

s=s\*i;

i++;}  
printf("%d!=%d”,n,s);

return 0;  
}

测试：



运行结果如图  
#include <stdio.h>

int main()

{  
int i=1,n,s=1;  
printf("Please enter n:" );

scanf(" %d" ,&n);

do{  
 s=s\*i;

i++;}

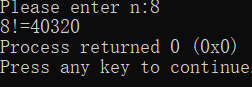
while(i<=n);

printf(“%d!=%d”,n,s);

return 0;

}

测试：

运行结果如图  
（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。

#include <stdio.h>

int main(void)

{

int i,n=0,s,m=1;

printf("Please enter s:");

scanf("%d",&s);

for(i=1;;i++)

{

m=m\*i;

n++;

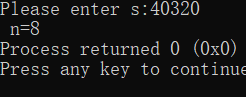
if(m>=s){

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

Break；}}

return 0;

}

测试：运行结果如

**（1）**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。

1. 流程图设计如下

尽量美观！

开始

n=10,i=0,m=0,j=0

结束

输出换行符

**N**

**Y**

m<n-i?

i<n?

**Y**

c=1

输出空格符

输出c（长度为4）

**N**

m=m+1

**Y**

**Y N**

c=c\*(i+1-j)/j

j=j+1

j=0?

j<=i?

**N**

i=i+1

2）源代码如下:

#include<stdio.h>

#define n 10

int main (void)

{

int i,j,c,m;

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

{

for(m=0;m<n-i;m++)

printf(" ");

for(j=0;j<=i;j++)

{

if(j==0)

c=1;

else

c=c\*(i+1-j)/j;

printf("%4d",c);

}

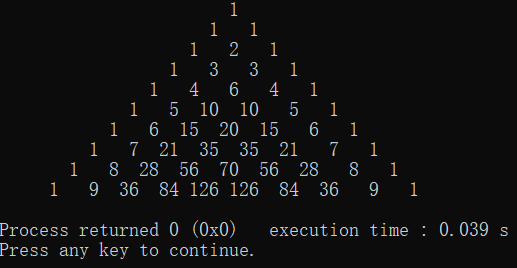
printf("\n");

}

return 0;

}

3）测试：

运行结果如图

**（2）**编写一个程序，将用户输入的任意正整数逆转，例如，输入1234，输出4321。

1）画出如图所示的流程图：

开始

ddddddddddd

输出x%10+’0’

输入正整数x

**Y**

(x/=10)不等于0？

**N**

结束

2）源程序如下：

#include <stdio.h>

int main(void)

{

int x;

scanf("%d",&x);

do

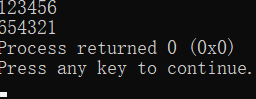
putchar(x%10+'0');

while((x/=10)!=0);

return 0;

}

3）测试：

运行结果如图

## 2.3 选做题

### 2.3.1 牛顿迭代

**【题目】**

编写并上机调试运行能实现以下功能的程序。

编写一个程序，用牛顿迭代法求方程满足精度e=**10-6**的一个近似根，并在屏幕上输出所求近似根。

牛顿迭代法求方程近似根的迭代公式为：



其中,是函数f(x)的导函数。牛顿迭代法首先任意设定的一个实数来作为近似根的迭代初值x0，然后用迭代公式计算下一个近似根x1。如此继续迭代计算x2, x3, …, xn, 直到，此时值xn即为所求的近似根。

**【算法】**

这个程序的设计主要运用的迭代的思想，我设计的程序中，运用的迭代公式为了将计算简单化，令x为0。同时将迭代公式((((3\*x-4)\*x)-5)\*x+13)/((9\*x-8)\*x-5)这种方法表达可以降低计算次数。最后程序如下。

**【程序清单】**

#include<stdio.h>

#include<math.h>

#define EPS 1e-6

int main(void)

{

double x,d;

printf("input initial root:\n");

scanf("%1f",&x);

do{

d=-((((3\*x-4)\*x)-5)\*x+13)/((9\*x-8)\*x-5);

x += d;

}while(fabs(d)>EPS);

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

return 0;

}

**【程序运行结果】**

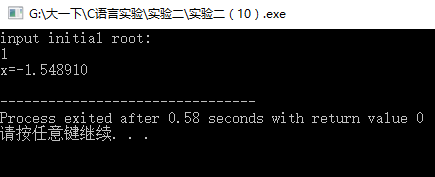


图2-27 实验二选做题运行结果

将x=-1.548910带入方程，发现成立。因此，此输出符合要求，结果正确。

## 2.4 自设题

**2.4.1 找特殊数字**

**【题目】**

编程找出这样的数字，它是三位数，它的三个数字的立方和等于它本身。

**【程序清单】**

#include<stdio.h>

int main()

{

int x,a,b,c;

for(x=100;x<1000;x++)

{

a = x%10;

b = (x/10)%10;

c = x/100;

if(a\*a\*a+b\*b\*b+c\*c\*c==x)

printf("%5d",x);

}

return 0;

}

**【测试结果】**

程序输出结果预测为：153 370 371 407，实际输出结果如下图2-28。

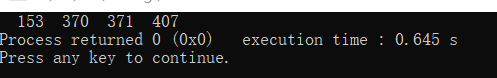


图2-28 实验二自设题输出结果

本次自设题中，运用了循环结构，通过这个实验熟练掌握了运算符%和/的基本用法和根本区别。

## 2.5 小结

源程序改错中遇到的那些错误也是我们平时容易出错的地方，通过这次实验，我总结出一套检查错误的方法：在输入过程中检查出大多数语法错误，或者拼写错误。接下来再检查一下逻辑是否有误，无误则进行多组数据的编译检验，确认最终无误。

源程序修改与替换题锻炼我们举一反三触类旁通的能力，在平时写程序时就应该思考对于一个问题有没有多钟解决方案，使用不同的逻辑结构多种尝试寻求最优解。

程序设计题则是自主的设计实践，在这次实验中我熟练掌握了流程图来表达程序的算法，同时增强对于所学知识尤其循环语句的熟练度和理解程度，学会了在什么地方使用哪种循环语句更加恰当。

在选做题中，运用了多项式的简便计算法的表示，基本掌握了迭代的程序的写法。