# 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);

1. }

**解答：**

（1）错误修改

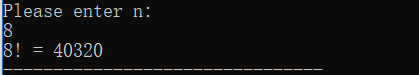
1. 第6行中的赋值语句不正确，正确形式为：

scanf("%d",&n);

1. 第7行中的for语句形式不对，正确形式为：

for(i=1,i<=n,i++)；

(2) 运行结果



### 2.2.2源程序修改替换题

（1）修改第1题，分别用while和do-while语句替换for语句。

（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。

Task1：

#include <stdio.h>

int main()

{

int i=1;

int s=1;

int n;

printf("Please enter n:");

scanf("%d",&n);

while(i<=n)

{

s=s\*i;

i++;

}

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

return 0;

}

Task2：

#include <stdio.h>

int main()

{

int i=1;

int s=1;

int n;

printf("Please enter n:");

scanf("%d",&n);

do

{

s=s\*i;

i++;

}

while(i<=n);

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

return 0;

}

Task3：

#include <stdio.h>

int main()

{

int n=1;

int s;

int m=1;

printf("Please enter s:");

scanf("%d",&s);

do

{

m=m\*n;

n++;

}

while(m<s);

printf("n=%d",n-1);

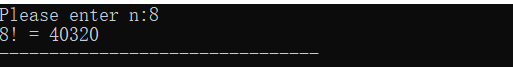
return 0;

}

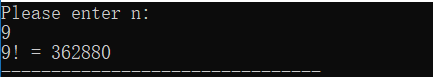
**解答：**

运行结果：

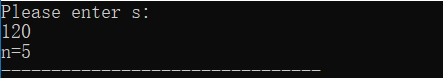
Task1：



Task2：



Task3:



### 2.2.3编程设计题

编写并上机调试运行能实现以下功能的程序。

1. 假设工资税金按以下方法计算：x ＜ 1000元，不收取税金；1000 ≤ x ＜ 2000，收取5%的税金；2000 ≤ x ＜ 3000，收取10%的税金；3000 ≤ x ＜ 4000，收取15%的税金；4000 ≤ x ＜ 5000，收取20%的税金；x＞5000，收取25%的税金。编写一个程序,输入工资金额，输出应收取税金额度，要求分别用if语句和switch语句来实现。

**解答：**

1. 流程图如下所示：

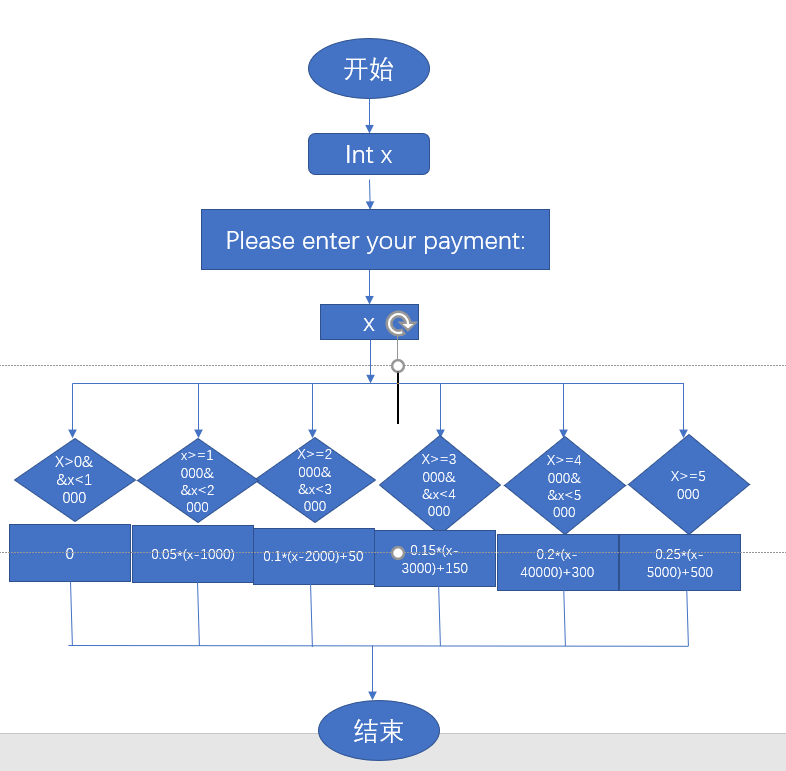


图2-1 编程题1的流程图

1. 源程序代码如下：

#include "stdio.h"

int main()

{

int x;

printf("Please enter your payment:");

scanf("%d",&x);

if(x>0&&x<1000) printf("0");

if(x>=1000&&x<2000) printf("%f",0.05\*(x-1000));

if(x>=2000&&x<3000) printf("%f",0.1\*(x-2000)+50);

if(x>=3000&&x<4000) printf("%f",0.15\*(x-3000)+150);

if(x>=4000&&x<5000) printf("%f",0.2\*(x-4000)+300);

if(x>=5000) printf("%f",0.25\*(x-5000)+500);

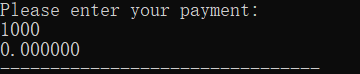
return 0;

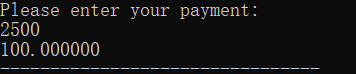
}

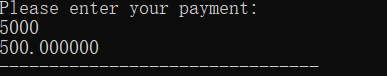
3）调试：

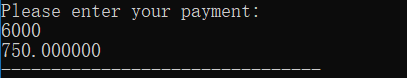
|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| 1000 | 0 | 0．000000 |
| 2500 | 100 | 100.000000 |
| 5000 | 500 | 500.000000 |
| 6000 | 750 | 750.000000 |

4）运行结果：









5）流程图如下：

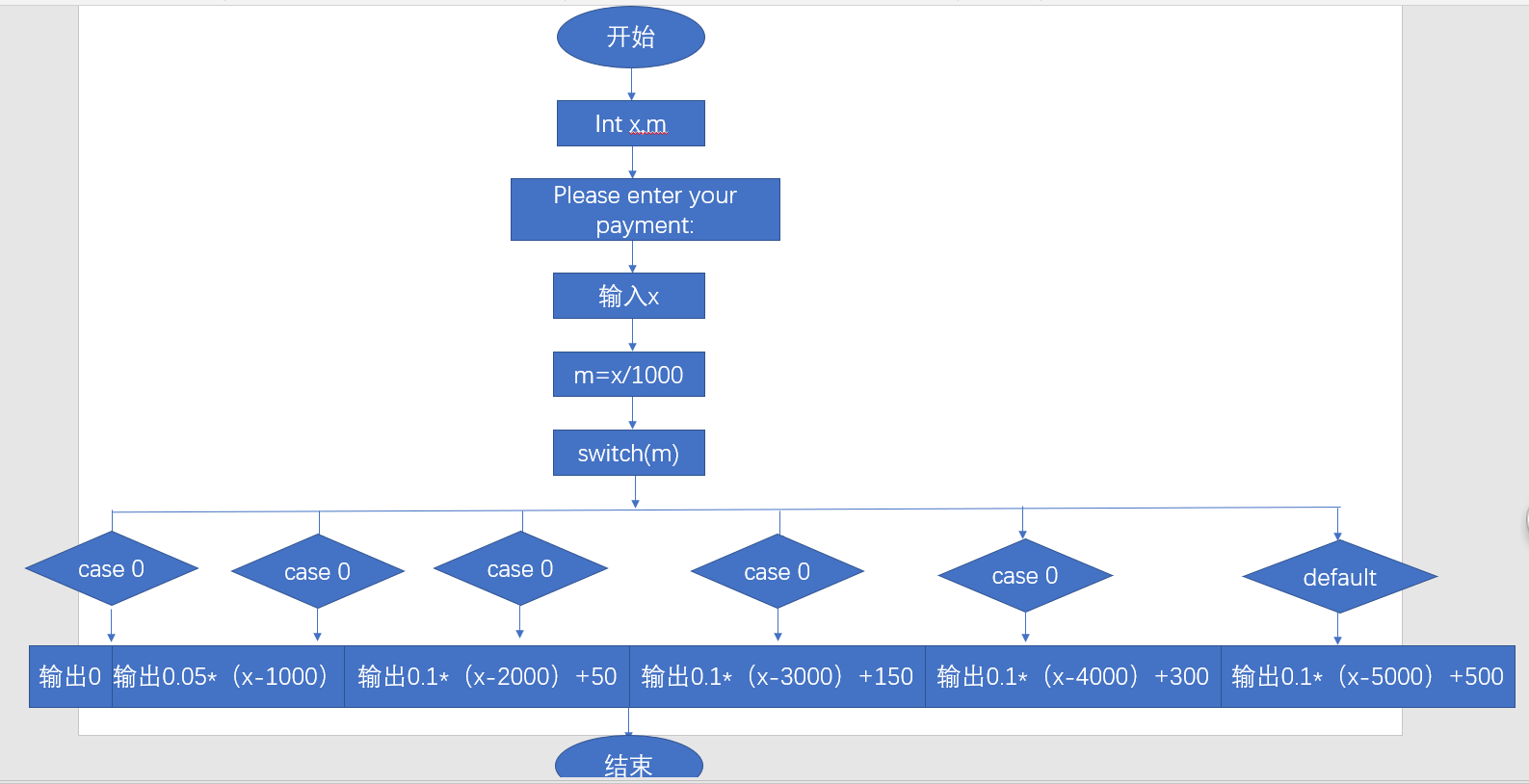


图2-2 编程题1的流程图

6）源代码如下：

#include "stdio.h"

int main()

{

int x,m;

printf("Please enter your payment:");

scanf("%d",&x);

m=x/1000;

switch(m)

{

case 0:printf("0");break;

case 1:printf("%f",0.05\*(x-1000));break;

case 2:printf("%f",0.1\*(x-2000)+50);break;

case 3:printf("%f",0.15\*(x-3000)+150);break;

case 4:printf("%f",0.2\*(x-4000)+300);break;

default:printf("%f",0.25\*(x-5000)+500);

}

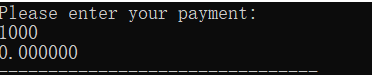
return 0;

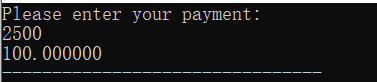
}

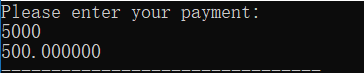
7）测试数据如下：

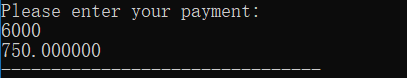
|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| 1000 | 0 | 0．000000 |
| 2500 | 100 | 100.000000 |
| 5000 | 500 | 500.000000 |
| 6000 | 750 | 750.000000 |

8）运行结果：









1. 编写一个程序,将输入的一行字符复制到输出，复制过程中将一个以上的空格字符用一个空格代替。

**解答：**

1. 流程图如下：

#include "stdio.h"

int main()

{

const int a=1;

const int b=0;

int state=1;

char c;

printf("Please input words:\n");

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

{

if(state==a)

{

putchar(c);

if(c==' ')

state=b;

}

else if(state==b)

{

if(c!=' ')

{

putchar(c);

state=a;

}

}

}

return 0;

}

1. 流程图如下：

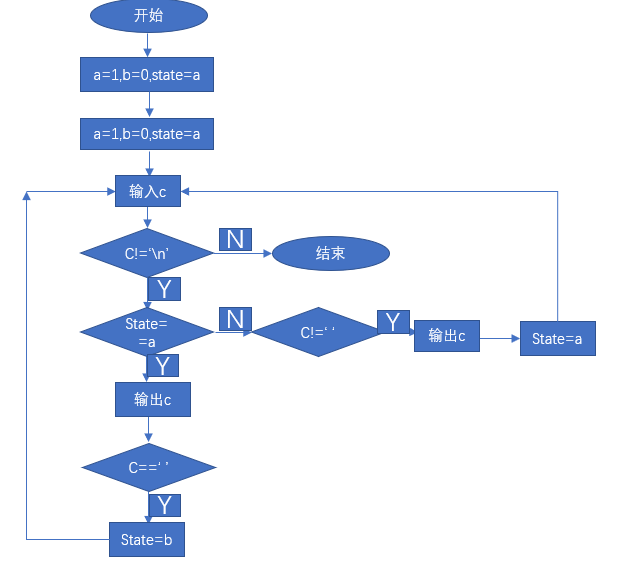
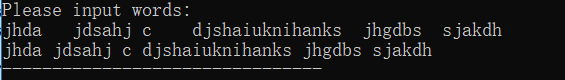


图2-3 编程题2的流程图

1. 测试如下：



（3）编写一个程序，打印如下的杨辉三角形。

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. 源程序代码如下所示：

#include "stdio.h"

int main()

{

int i,k,m,n,p,s[10][10];

for(i=0;i<=9;i++)

{

for(k=1;k<=18-2\*i;k++)

{

putchar(' ');

}

s[0][0]=1;

for(m=1;m<=9;m++)

{

s[m][0]=1;

s[m][m]=1;

}

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

{

s[i][n-1]=s[i-1][n-2]+s[i-1][n-1];

}

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

{

printf("%4d",s[i][p]);

}

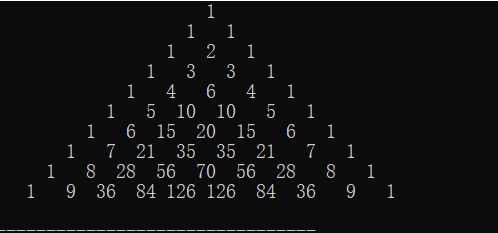
printf("\n");

}

return 0;

}

1. 输入测试：



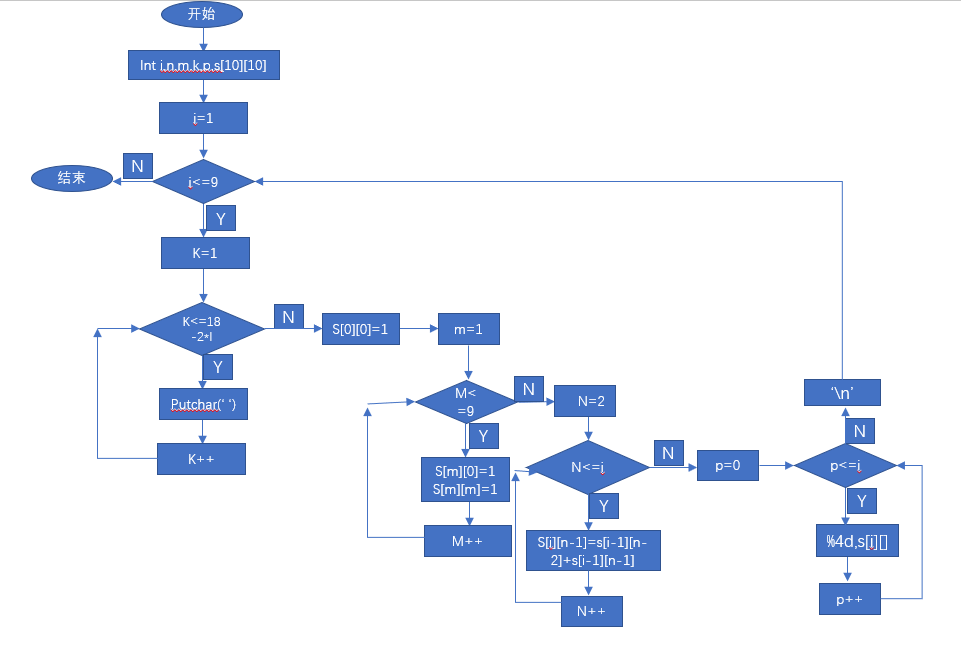
1. 流程图如下所示：  
   

图2-4 编程题3的流程图

（4）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。

**解答：**

1. 源程序代码如下所示：

#include "stdio.h"

int main()

{

int o, i,m,n,s[10];

printf("Please input a number:\n");

scanf("%d",&m);

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

{

s[i]=m%10;

n=m/10;

m=n;

}

o=i;

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

{

printf("%d",s[i]);

}

return 0;

}

1. 流程图如下所示：

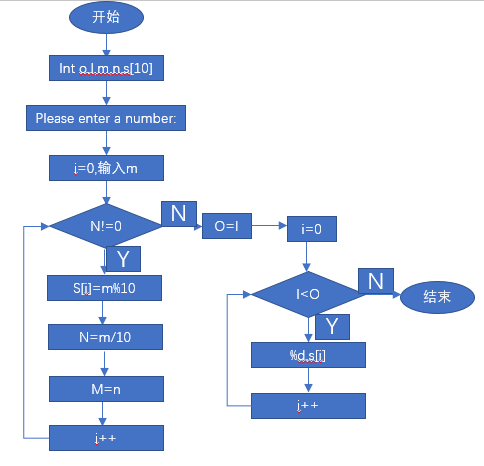
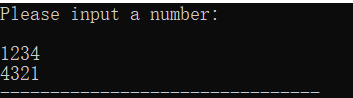
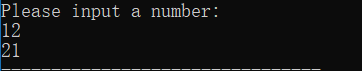


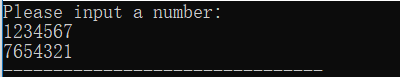
图2-5 编程题4的流程图

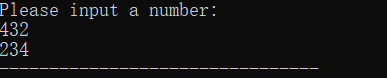
1. 数据测试如下：

|  |  |  |
| --- | --- | --- |
| 输入数据 | 应该输出 | 实际输出 |
| 1234 | 4321 | 4321 |
| 12 | 21 | 21 |
| 1234567 | 7654321 | 7654321 |
| 432 | 234 | 234 |









### 2.2.4选做题

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*f(x)=*3*x3-*4*x3-5x+*13*=0*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, *f '(x)*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*x*0，然后用迭代公式计算下一个近似根*x*1。如此继续迭代计算*x*2, *x*3, …, *x*n, 直到 |*x*n- *x*n-1|≤精度e，此时值*x*n即为所求的近似根。

解答

1. 源程序代码如下所示：

#include "stdio.h"

#include "math.h"

int main()

{

const double STEP=1e-6;

double x,d;

printf("Please enter initial root:\n");

scanf("%lf",&x);

do

{

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

x=x+d;

}

while(fabs(d)>STEP);

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

return 0;

}

1. 流程图如下所示：

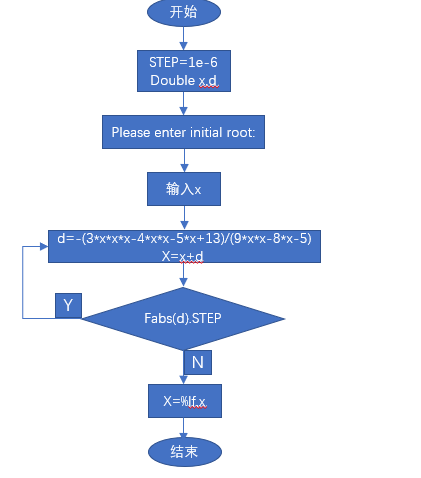
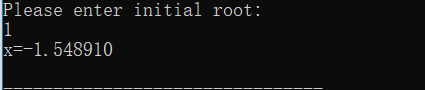


图2-6 选做题的流程设计图

1. 数据测试：



## 2.3实验小结

1）本次实验中感觉较难的是编程题的2，3，花费了我较多的时间，由于在机房较吵，所以没有静下心来做题，导致我没有在机房完成任务，花费了课余的时间，这是教训。

2）C语言必须要有良好的数学功底，在做税收实验中，假如数学功底不够，常见的数学问题不知道，就会直接导致不会做那个题。因此，在以后，一定要给数学留足够的时间来学习，数学是一切的基础。

3）由于有了第一次的经验，第二次的实验与实验报告就顺利多了。很重要的是我学会了用powerpoint画流程图，而且找到了技巧，事半功倍。