

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期： 2019.10.25**

**网络空间安全学院**

**目 录**

**[2 流程控制实验 1](#_Toc404837920)**

[2.1 实验目的 1](#_Toc404837921)

[2.2 实验内容 1](#_Toc404837922)

[2.3 实验小结 1](#_Toc404837923)

# 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。

#include <stdio.h>

void main(void)

{

int i,n,s=1;

printf("Please enter n:");

scanf("%d",n);

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

s=s\*i;

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

}

**解答：** （1）错误修改：

1) 第2行的void main( void )，正确形式为：

main( void )

2）第6行scanf 语句赋值需要加&，正确形式为：scanf(“%d”,&n);

3）第7行的 for 语句中不能用逗号，正确形式为：for (i = 1;i <= n; i++)

（2）错误修改后运行结果：



**图2-1 改错题的运行结果**

**2.2.2 源程序修改替换**

**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。

**解答：**

替换后的程序如下所示：

（1）while

#include <stdio.h>

int main(void)

{

int i,n;

i=n=1;

unsigned long s;

s=1;

printf("Please enter n:");

scanf("%d",&n);

while (i<=n)

{s=s\*i;

i++;}

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

}

Do-while 型

#include <stdio.h>

int main(void)

{

int i,n;

i=n=1;

unsigned long s;

s=1;

printf("Please enter n:");

scanf("%d",&n);

do

{s=s\*i;

i++;}

while (i<=n);

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

}

（2）#include <stdio.h>

int main(void)

{

int i;

i=1;

unsigned long n,s;

n=s=1;

printf("Please enter S:");

scanf("%lu",&n);

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

s=s\*i;

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

}



图2-2 程序修改替换题运行结果1



图2-3 程序修改替换题运行结果2

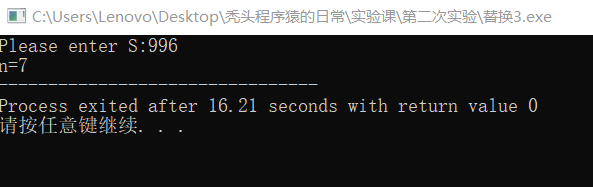


图2-4 程序修改替换题运行结果3

**2.2.3 程序设计**

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. 算法流程如图1.1所示。

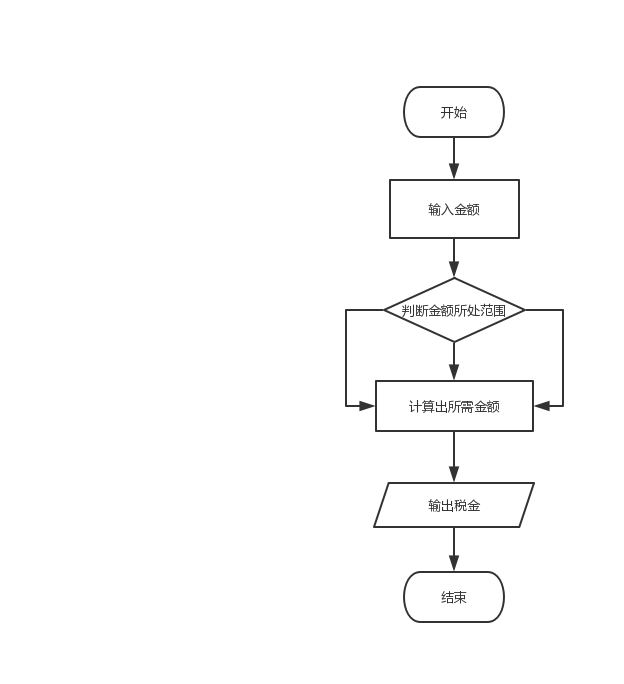


图2-5 编程题1的程序流程图

2）源程序清单

If语句

#include <stdio.h>

int main(void)

{

float a,b;

printf("请输入工资金额\n");

scanf("%f",&a);

if (a<1000)

b=0;

else

if (a>=1000&&a<2000)

b=(a-1000)\*0.05;

else

if (a>=2000&&a<3000)

b=(a-2000)\*0.1+0.05\*1000;

else

if (a>=3000&&a<4000)

b=(a-3000)\*0.15+0.15\*1000;

else

if (a>=4000&&a<5000)

b=(a-4000)\*0.2+0.3\*1000;

else

if (a>=5000)

b=(a-5000)\*0.25+0.5\*1000;

printf("%f",b);

return 0;

}

Switch语句

#include <stdio.h>

int main(void)

{

float a,b;

printf("请输入工资金额\n");

scanf("%f",&a);

switch ((int)a/1000)

{

case 0: b=0;break;

case 1: b=(a-1000)\*0.05;break;

case 2: b=(a-2000)\*0.1+0.05\*1000;break;

case 3: b=(a-3000)\*0.15+0.15\*1000;break;

case 4: b=(a-4000)\*0.2+0.3\*1000;break;

default: b=(a-5000)\*0.25+0.5\*1000;break;

}

printf("%f",b);

return 0;

}

3）测试

（a） 测试数据：

2500 9966

（b） 对应测试数据的运行结果截图

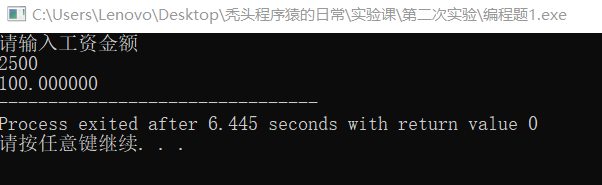


图2-6 编程题1的运行结果实例1

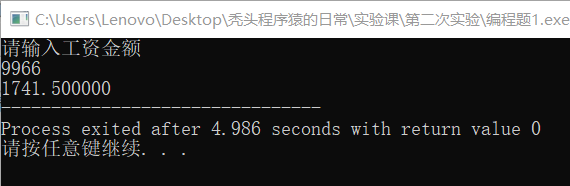


图2-7 编程题1的运行结果实例2

测试数据的方法如表1-1所示

表2-1 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 |
| 用例1 | 2500 | 100 | 100 |
| 用例2 | 9966 | 1741.5 | 1741.5 |

（2）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. 解题思路：

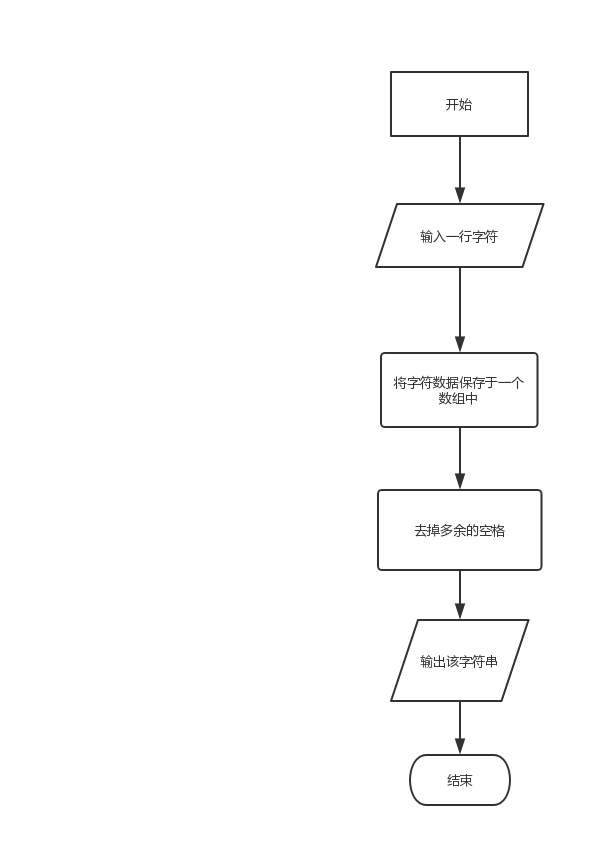


图2-8编程题3的程序流程图

2）程序清单：

#include <stdio.h>

int main(void)

{

char c[100];

int i=0;

gets(c);

while (1)

{

switch (c[i])

{

case 32: if (c[i+1]==32) {i++; break;} else {putchar(32);i++;break;}

case 0: return 0;

default: putchar(c[i]);i++;

}

}

return 0;

}

3）测试

对应测试测试的运行结果如图所示。

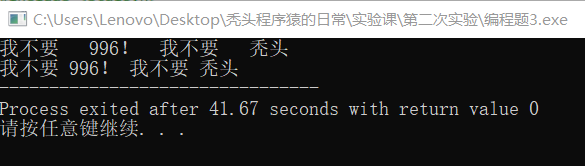


图2-9编程题2的运行结果实例1

验证了程序的正确性。

**（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. 算法流程如图所示。

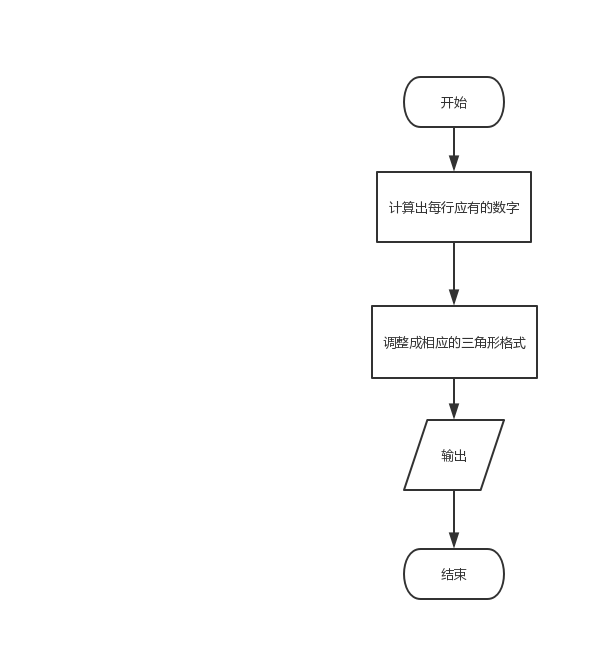


图2-10 编程题3的程序流程图

1. 源程序清单

#include<stdio.h>

#define n 10

#define size n\*(n+1)/2

int main(void)

{

int \*p[n],a[size],sum=0,i,j,k;

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

{

p[i]=&a[sum];

sum+=i+1;}

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

{

\*p[i]=1;

\*(p[i]+i)=1;}

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

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

\*(p[i]+k)=\*(p[i-1]+k-1)+\*(p[i-1]+k);

for(i=0;i<n;i++){ for(j=n-1;j>i;j--)

printf(" ");

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

printf("%4d",\*(p[i]+k));

printf("\n");}

}

1. 测试
2. 对应测试测试用例1的运行结果如图所示。

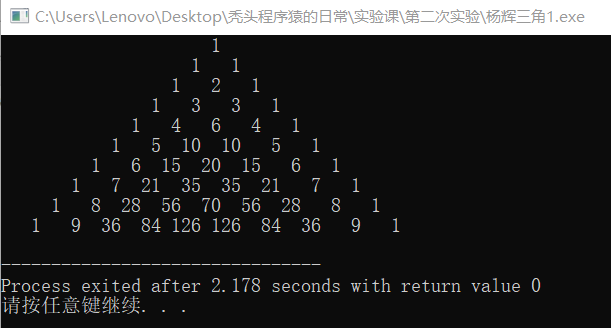


图2-11 编程题3的运行结果实例

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。

（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。

分析：

此题目比上一题容易一些，将输入的每个数字看成单独的字符，再输出，用到了定义函数的结构。

解答：#include <stdio.h>

int main(void)

{

int x;

printf("Input a 整数:\n");

scanf("%d",&x);

do

putchar(x%10+'0');

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

printf("\n");

return 0;

}

算法流程图如图所示

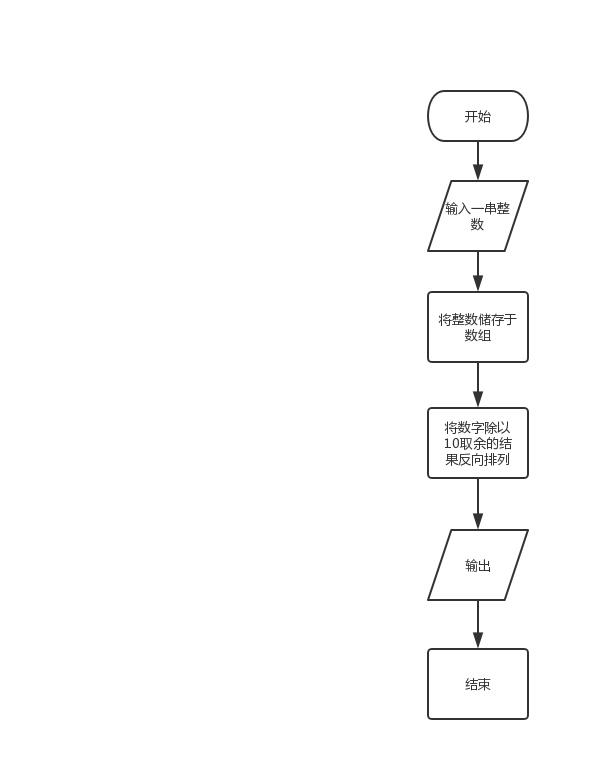


图2-12 编程题4的程序流程图

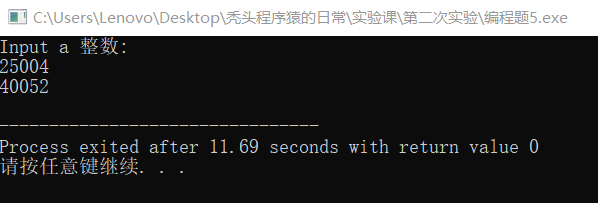


图2-13 编程题4的运行结果实例

## 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。

1. 算法流程图

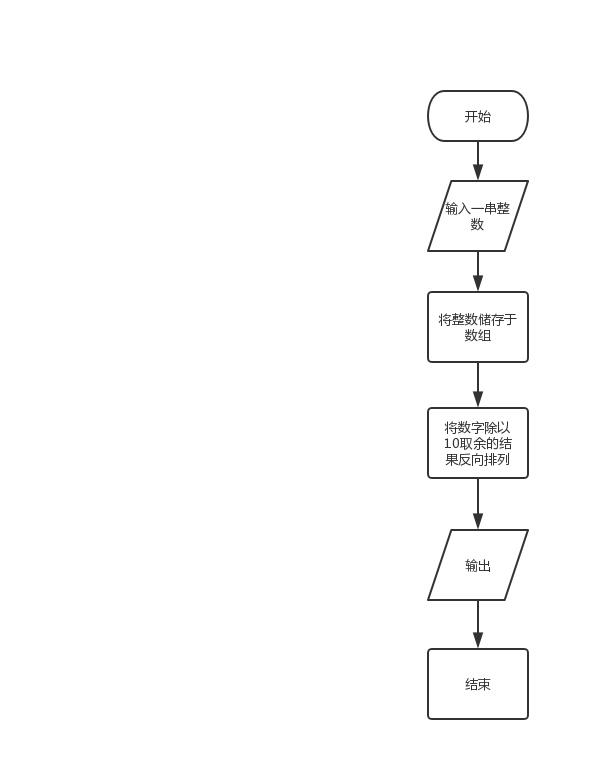


图2-14 选做题的程序流程图

1. 程序源代码：

#include <stdio.h>

int main(void)

{

float a,b;

b=2.3;

scanf("%f",&a);

for(;((3.0\*a\*a\*a-4.0\*a\*a-5\*a+13.0)/(9.0\*a\*a-8.0\*a-5.0)>1e-6)||((3.0\*a\*a\*a-4.0\*a\*a-5\*a+13.0)/(9.0\*a\*a-8.0\*a-5.0)<-1e-6);a=b)

b=a-(3.0\*a\*a\*a-4.0\*a\*a-5\*a+13.0)/(9.0\*a\*a-8.0\*a-5.0);

printf("%f",b);

return 0;

}

1. 测试

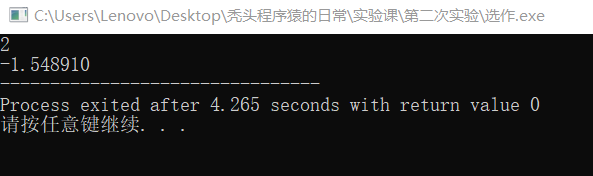


图2-15 选做题的运行结果实例

## 2.3 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。  
 在这次实验的过程中，我在进行打印杨辉三角的实验时曾经遇到过不会正确调整打印结果的格式问题。最终我通过上网和询问同学解决了这一问题。

通过本次实验课，我深刻地体会到了“纸上得来终觉浅，绝知此事要躬行”这句话所蕴含的深刻道理。也许课上听完讲后我觉得自己已经掌握了相关的知识，但直到自己亲自动手编程是才发现自己的能力并没有想象中那么出色。希望我以后可以在这种活动中收获更多！

参考文献

[1] 曹计昌,卢萍,李开. C语言程序设计,北京： 科学出版社,2013

[2] 李开,卢萍,曹计昌. C语言实验与课程设计, 北京：科学出版社,2011