**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，了解转义语句与标号语句。

（2） 熟练运用for、while、do-while语句来编写程序。  
（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 **实验内容及结果**

**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源程序进行调试修改，使之能够正确完成指定任务。例如8!=40320。

**程序代码：**

#include<stdio.h>

int 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);

}

return 0;

【**分析及改正**】本程序共存在2处错误，分析如下：

（1）scanf(“%d”,n);

错误原因：应该把要读入的变量的地址传递给scanf才能正确写入

改正：scanf(“%d”,&n);

（2）return 0；

错误原因：主函数的返回语句应写在主函数函数体内部，而源程序写在了外部。

改正：将return 0; 移至大括号的内部。

**2.2.2** 源程序修改替换

（1）修改上述计算s=n!的源程序，要求分别用while和do-while语句替换for语句。

（2）修改上述极端s=n!的源程序，要求输入改为“整数S”，输出改为“满足n！≥S的最小整数n”。例如，输入整数为40310，输出结果为n=8。

**正确的原程序代码：**

#include<stdio.h>

int 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);

return 0;

}

（1） **【原程序分析】**源程序使用了for循环，自变量i从1增长到n。

**【替换方案】**分别使用while和do-while，将赋值语句放在循环前面，自增语句放在循环体中。

**【程序】**

1. 使用while循环：

#include<stdio.h>

int 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);

return 0;

}

1. 使用do-while循环：

#include<stdio.h>

int 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);

return 0;

}

【**测试**】如果输入 9，应该输出：9！=362880，运行结果如图2.1所示，程序正确。

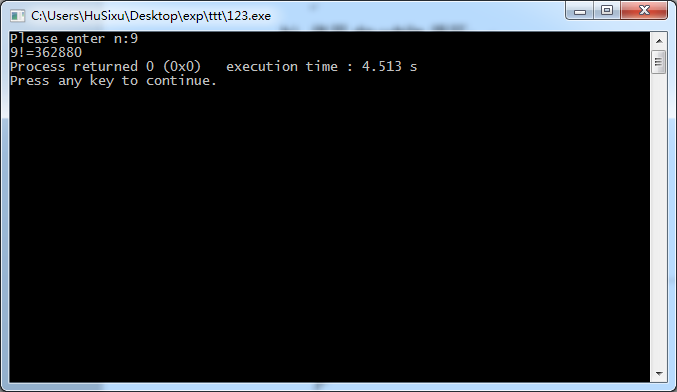


图2.1 程序运行结果截图

（2）【原程序分析】原程序输入n，计算n的阶乘。

【替换方案】输入S,求最大的n使n的阶乘小于S。

【程序】

#include<stdio.h>

int main( void )

{ int i,n,s=1;

printf(“整数S:”);

scanf(“%d”,&s);

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

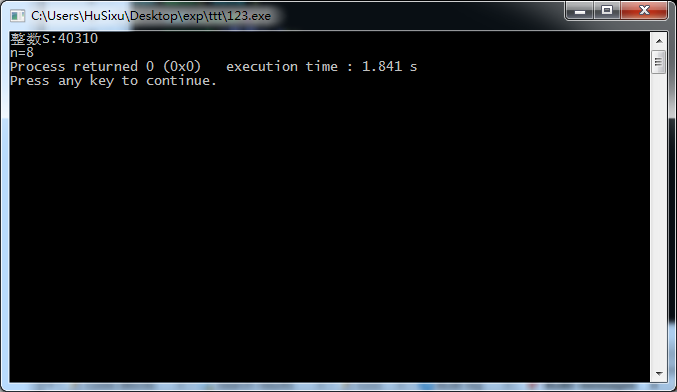
i=i\*n;

printf(“n=%d”,n-1);

return 0;

}

【测试】输入40310，结果应为n=8



**2.2.3** 程序设计

（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。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。

【分析】典型的if-else分支结构和switch-case分支结构。注意if和else的匹配以及每一个case后面的break。

输入格式采取如下方式：第一行输入n表示测试样例的个数，后面紧接着有n行输入，每行输入包含两个数分别表示处理方式（t）和工资金额（x）。当t=0时，输出用if语句处理后的结果；当t=1时，输出用switch语句处理后的结果；当t=2时，同时输出两种语句处理后的结果。

【程序】

#include<stdio.h>

int main(void){

int n;

scanf("%d",&n);

int i,mode;

double no\_money;

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

scanf("%d",&mode);

switch(mode){

case 2:

case 0:

scanf("%lf",&no\_money);

switch((int)no\_money/1000){

case 0:

printf("After if-else processing,the amount of tax to be paid is : 0.00\n");

break;

case 1:

printf("After if-else processing,the amount of tax to be paid is : %.2f\n",no\_money\*0.05);

break;

case 2:

printf("After if-else processing,the amount of tax to be paid is : %.2f\n",no\_money\*0.10);

break;

case 3:

printf("After if-else processing,the amount of tax to be paid is : %.2f\n",no\_money\*0.15);

break;

case 4:

printf("After if-else processing,the amount of tax to be paid is : %.2f\n",no\_money\*0.20);

break;

case 5:

default:

printf("After if-else processing,the amount of tax to be paid is : %.2f\n",no\_money\*0.25);

break;

}

if(mode!=2)

break;

else;

case 1:

if(mode==1)

scanf("%lf",&no\_money);

if(no\_money<1000)

printf("After switch processing,the amount of tax to be paid is : 0.00\n");

else if(1000<=no\_money && no\_money<2000)

printf("After switch processing,the amount of tax to be paid is : %.2f\n",no\_money\*0.05);

else if(2000<=no\_money && no\_money<3000)

printf("After switch processing,the amount of tax to be paid is : %.2f\n",no\_money\*0.10);

else if(3000<=no\_money && no\_money<4000)

printf("After switch processing,the amount of tax to be paid is : %.2f\n",no\_money\*0.15);

else if(4000<=no\_money && no\_money<5000)

printf("After switch processing,the amount of tax to be paid is : %.2f\n",no\_money\*0.20);

else if(5000<=no\_money)

printf("After switch processing,the amount of tax to be paid is : %.2f\n",no\_money\*0.25);

break;

}

if(i!=n-1)

putchar('\n');

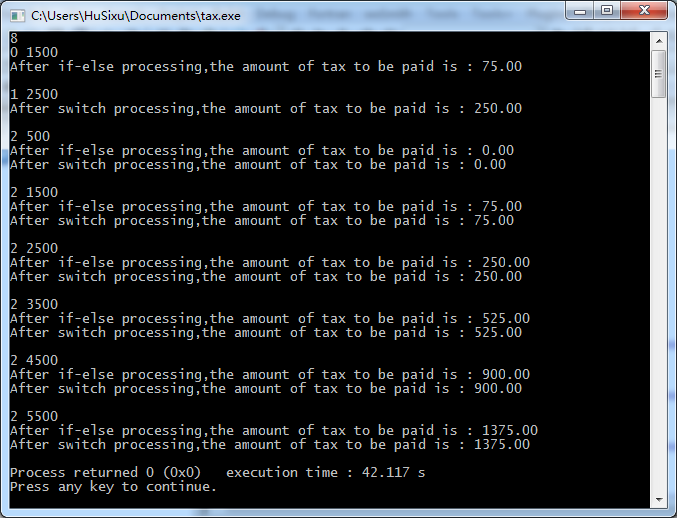
else;

}

return 0;

}

【测试】 输入测试个数，并输入各个区间段和各种模式下的样例，输出应与数学计算的结果相同。



经测试，结果与预想相同

（2）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。

【分析】使用一个足够大的数组作为缓冲区。使用switch-case和if-else分支控制，遇到空格则将其之后的空格删除，遇到换行符输出，遇到EOF结束程序。注意case和if的嵌套提高了代码的利用率，减少代码量。

【程序】

#include<stdio.h>

#define MAX\_STR\_SIZE ((1<<(4\*sizeof(int)))-1)

int main(void){

char ch;

char str[MAX\_STR\_SIZE];

int i=0;

for(;(ch=getchar())!=EOF;i++){

switch(ch){

case ' ':

str[i]=ch;

for(;(ch=getchar())==' ';);

i++;

default:

if(ch == '\n'){

str[i]=ch;

str[i+1]='\0';

printf("%s",str);

i=0;

}

else if(ch == EOF)

return 0;

else

str[i]=ch;

break;

}

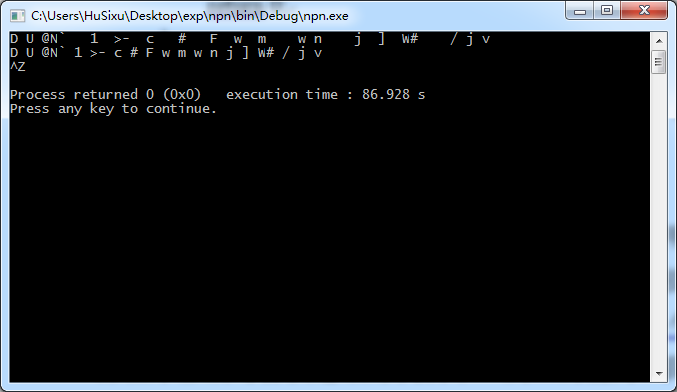
}

return 0;

}

【测试】输入D U @N` 1 >- c # D F w m w n j ] W# / j v

应该输出D U @N` 1 >- c # D F w m w n j ] W# / j v



经测试，与预想吻合。

（3）（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)



说明：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。

【分析】基础的循环的嵌套加上函数的调用，注意格式化输出以及边界问题。

【程序】

#include<stdio.h>

int C(int u,int d);

int main(void){

int line,i,j;

scanf("%d",&line);

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

for(int temp=0;temp<line-i;temp++)

printf(" ");

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

printf("%-4d",C(j,i));

}

putchar('\n');

}

return 0;

}

int C(int up,int down){

if(up==0)

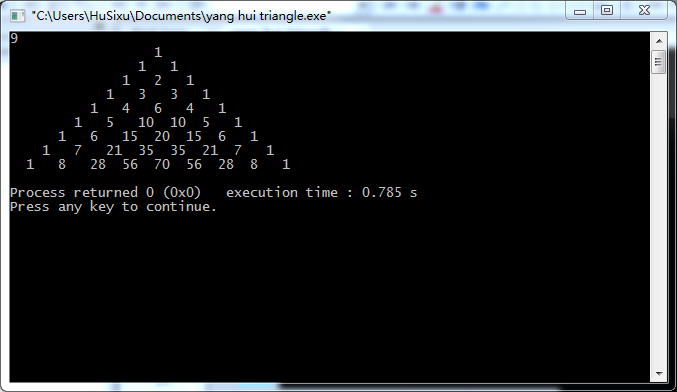
return 1;

else

return (down-up+1)\*C(up-1,down)/up;

}

【测试】输入9，应该输出9阶的杨辉三角。



输出与预想一致。

（4）编写一个程序，将用户输入的任意整数逆转。例如，输入1234，输出4321。

【分析】利用字符串读入，重写string.h中的strrev即可。（或用-ansi参数编译）输入任意整数，将其反转，若输入的整数前面有0，去0后反转，若反转后前面有0，去0后输出。

【程序】

#include<stdio.h>

#include<stdlib.h>

#define MAX\_STR\_SIZE 65535

char \* strrev(char\* s);

inline char \* reduce0(char \* str);

int main(void){

char \*str = (char \*)malloc(sizeof(char)\*MAX\_STR\_SIZE);

scanf("%s",str);

reduce0(str);

printf("%s\n",reduce0(strrev(str)));

return 0;

}

char \* strrev(char\* s)

{

char\* h = s;

char\* t = s;

char ch;

while(\*t++){};

t--;

t--;

while(h < t){

ch = \*h;

\*h++ = \*t;

\*t-- = ch;

}

return s;

}

char \* reduce0(char \* str){

static int i,j;

for(i=0,j=0;str[j]=='0';j++);

for(;str[j]!='\0';i++,j++) /\* reduce 0 \*/

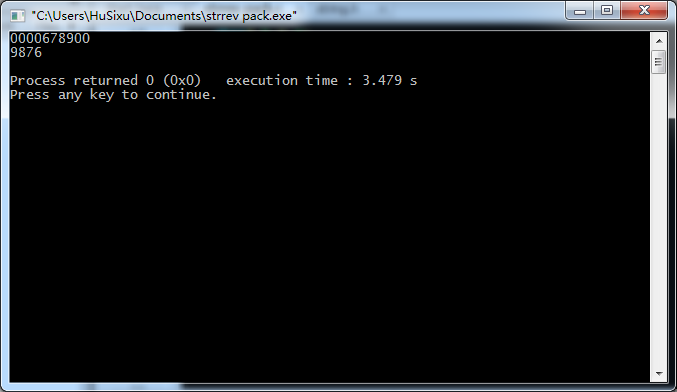
str[i]=str[j];

str[i]='\0';

return str;

}

【测试】输入任意整数，将其反转，若输入的整数前面有0，去0后反转，若反转后前面有0，去0后输出。



输出与预想一致

**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。



【分析】用数组记录Xn，用带标记的递归计算Xn（迭代亦可）。

【程序】

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define E 0.000001

double st[1<<12];

double X(int k);

double F(double x);

double F\_(double x);

int main(void){

memset(st,0,sizeof(int)\*(1<<12));

st[0]=1.00;

long i=1;

for(;i<(1<<12);i++)

st[i]=X(i);

for(i=2;((X(i)>X(i-1))?X(i)-X(i-1):X(i-1)-X(i))>E;i++);

printf("%f\n",X(i));

return 0;

}

double X(int k){

if(st[k]==0&&k!=0)

return X(k-1)-F(X(k-1))/F\_(X(k-1));

else

return st[k];

}

double F(double x){

return (3\*x\*x\*x-4\*x\*x-5\*x+13);

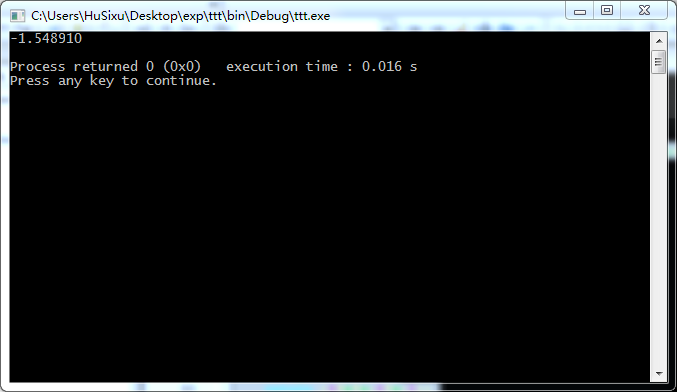
}

double F\_(double x){

return (9\*x\*x-8\*x-5);

}

【测试】答案与预想一致



**1.3 实验体会**

实验然我领悟到编程是要注意新手容易犯的问题，如判断语句写成“=”，case分之的控制，else悬挂等等。还有要尽可能利用数学知识，减少及其的运算量和运算所需的时间、空间。