

**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： 李 平**

**报告日期： 2019.12.3**

**计算机科学与技术学院**

目 录

[1 流程控制实验 1](#_Toc10476)

[1.1 程序改错 1](#_Toc25858)

[1.2 程序修改替换 2](#_Toc7324)

[1.3 程序设计 5](#_Toc11436)

[1.4 小结 26](#_Toc10390)

[2 数组程序设计实验 27](#_Toc32377)

[2.1 程序改错与跟踪调试 27](#_Toc30239)

[2.2 程序完善与修改替换 29](#_Toc24747)

[2.3 程序设计 31](#_Toc13373)

[2.4小结 51](#_Toc10455)

[3 结构与联合实验 52](#_Toc16968)

[3.1 表达式求值的程序验证 52](#_Toc161)

[3.2 程序完善与修改替换 53](#_Toc5187)

[3.3 程序设计 57](#_Toc24906)

[3.4小结 88](#_Toc18084)

[参考文献 89](#_Toc17595)

# 1 流程控制实验

## 1.1 程序改错

#include<stdio.h>

int main(void)

{

int i,x,k,flag=0;

printf("本程序判断合数，请输入大于1的整数，Ctrl+z 结束\n");

while(scanf("%d",&x)!=EOF){

for(i=2,k=x>>1;i<=k;i++)

if(!x%i){ 错误原因：优先级混乱

改为if(!(x%i))

flag=1;

break;

}

if(flag=1) {printf("%d 是合数\n",x);flag=0;} 错误原因：

未正确理解if中的等于

改为：if(flag==1) 同时flag还原

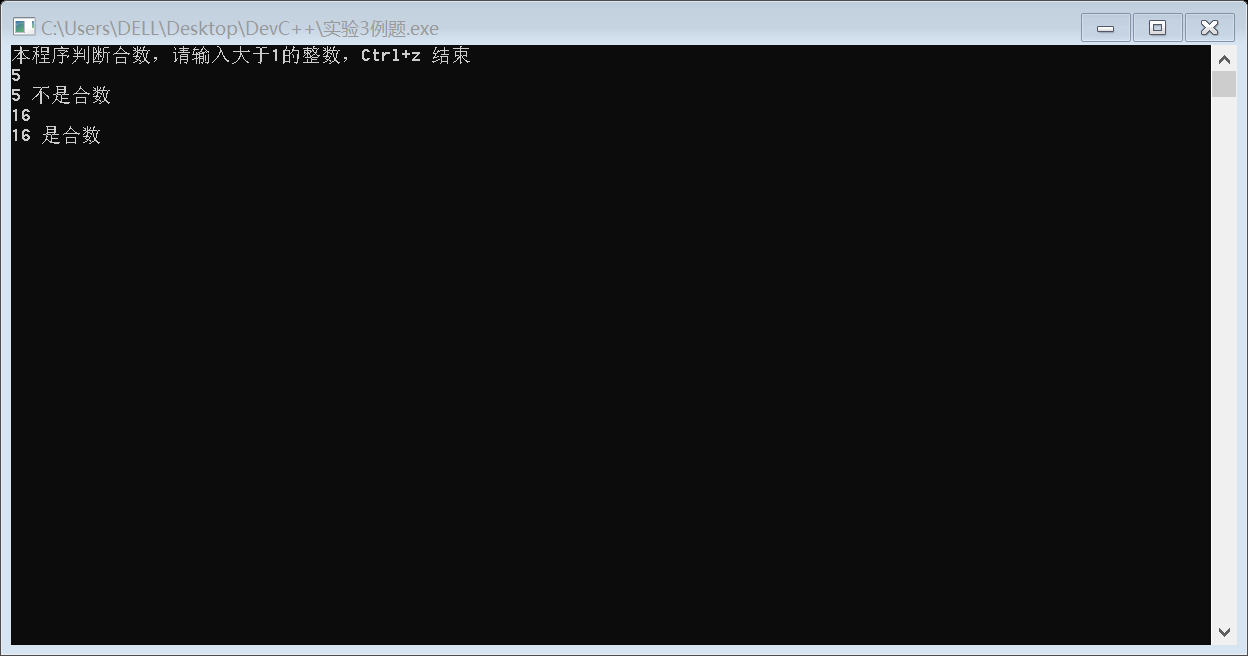
else printf("%d 不是合数\n",x);

}

return 0;

}

运行截图



## 1.2 程序修改替换

源程序通过循环，对每一个小于所输入数的一半的数进行求余操作，一旦有一个数求余为0，则表明整除，即为合数。

1. 将break去掉，在if中增加条件

#include<stdio.h>

int main(void)

{

int i,x,k,flag=0;

printf("本程序判断合数，请输入大于1的整数，Ctrl+z 结束\n");

while(scanf("%d",&x)!=EOF){

for(i=2,k=x>>1;i<=k&&!flag;i++)

if(!(x%i)){

flag=1;

}

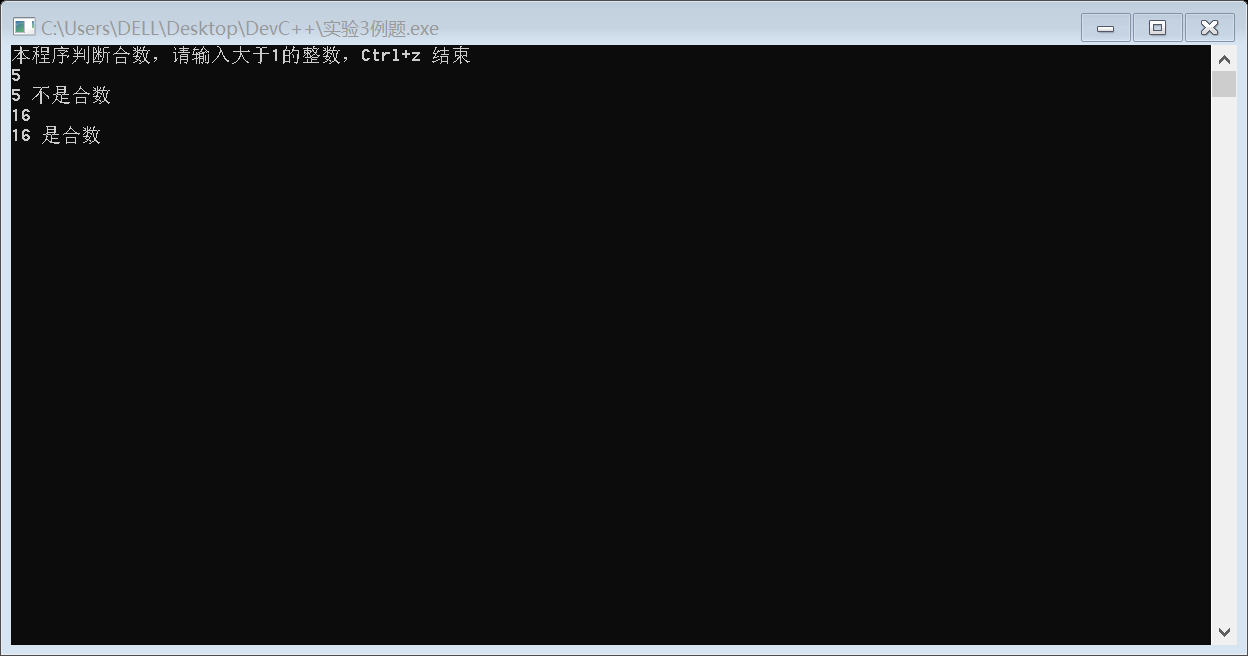
if(flag==1) printf("%d 是合数\n",x);

else printf("%d 不是合数\n",x);

}

return 0;

}



(2)直接根据题意进行更改

#include<stdio.h>

int main(void)

{

int i,x,k,flag=0;

printf("本程序判断合数，请输入大于1的整数，Ctrl+z 结束\n");

while(scanf("%d",&x)!=EOF){

i=2;

k=x>>1;

do

{

if(!(x%i))

{

flag=1;

}

i++;

}while(i<=k&&!flag);

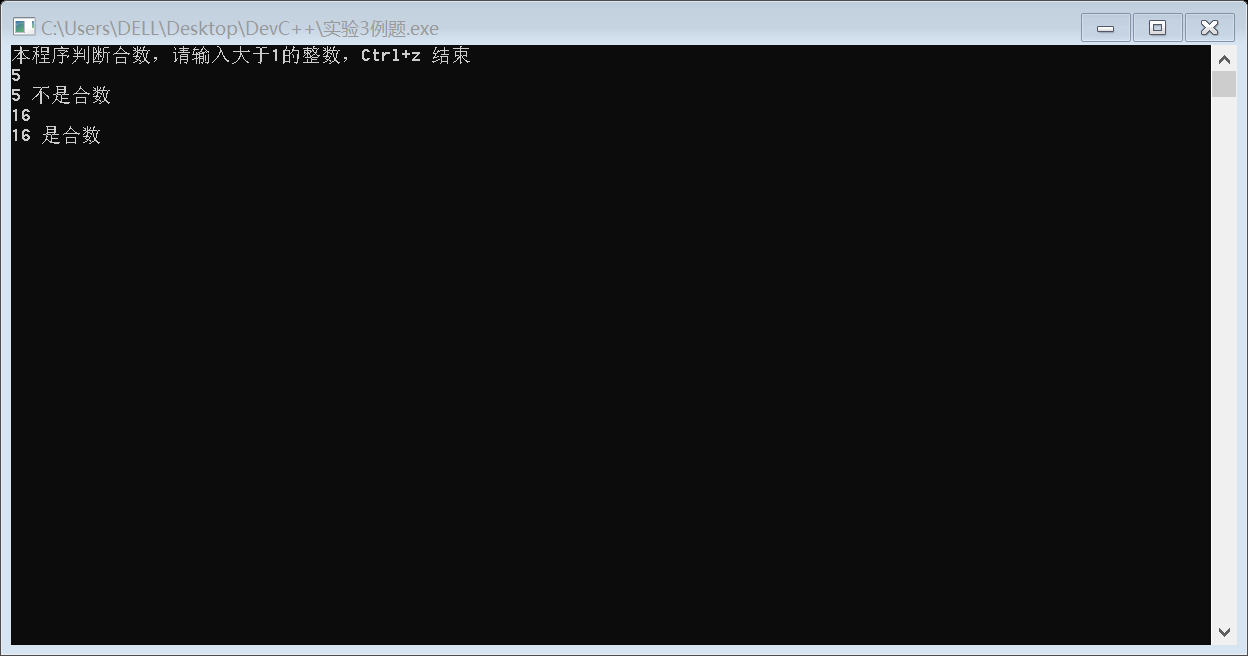
if(flag==1) printf("%d 是合数\n",x);

else printf("%d 不是合数\n",x);

}

return 0;

}



(3)

#include<stdio.h>

#include<math.h>

int IsPrime(int n)

{

int j;

if(n==1&&n==2)

{

return 0;

}

for(j=2;j<=sqrt(n);j++)

{

if(n%j==0)

{

return 1;

}

}

return 0;

}

int main(void)

{

int i;

for(i=400;i<1000;i++)

{

if(IsPrime(i)&&IsPrime(i/10)&&IsPrime(i/100))

{

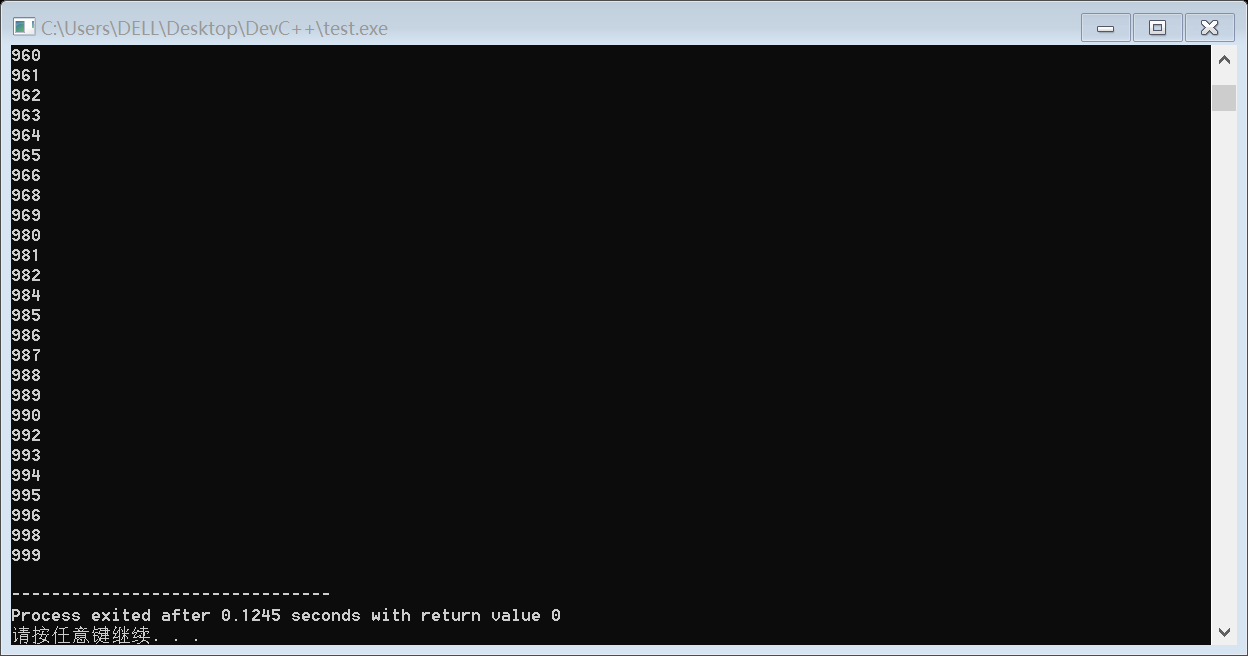
printf("%d\n",i);

}

}

return 0;

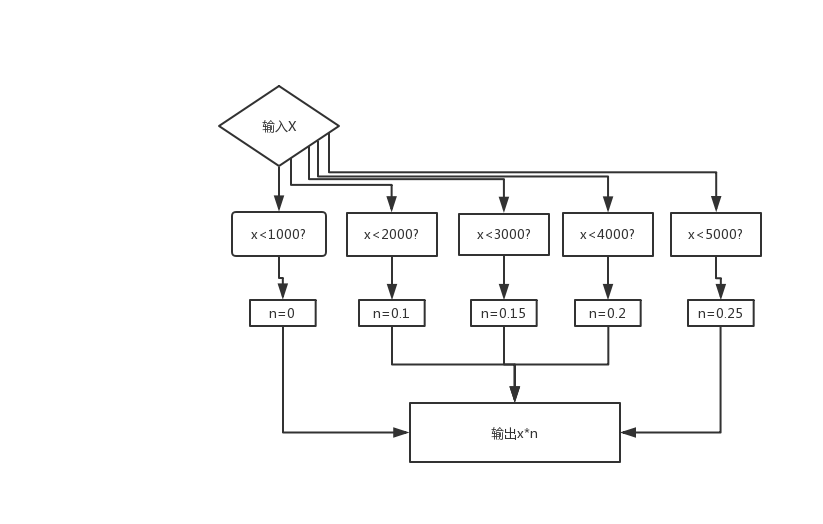
}



## 1.3 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。

解题思路



程序清单

If语句

#include<stdio.h>

int main(void)

{

int x;

printf("Please enter your salary:\n"); /\*输入工资\*/

scanf("%d",&x);

printf("Your tax is:");

if(x/1000==0) printf("0"); /\*根据工资数额大小做出判断\*/

else if(x/1000==1) printf("%.2f",x\*0.05);

else if(x/1000==2) printf("%.2f",x\*0.1);

else if(x/1000==3) printf("%.2f",x\*0.15);

else if(x/1000==4) printf("%.2f",x\*0.2);

else if(x/1000>4) printf("%.2f",x\*0.25);

return 0;

}

Switch语句

#include<stdio.h>

int main(void)

{

int x;

printf("Please enter your salary:\n"); /\*输入工资\*/

scanf("%d",&x);

printf("Your tax is:");

switch((x/1000)) /\*根据工资数额大小做出判断\*/

{

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

case 1: printf("%.2f",x\*0.05);break;

case 2: printf("%.2f",x\*0.1);break;

case 3: printf("%.2f",x\*0.15);break;

case 4: printf("%.2f",x\*0.2);break;

default: printf("%.2f",x\*0.25);break;

}

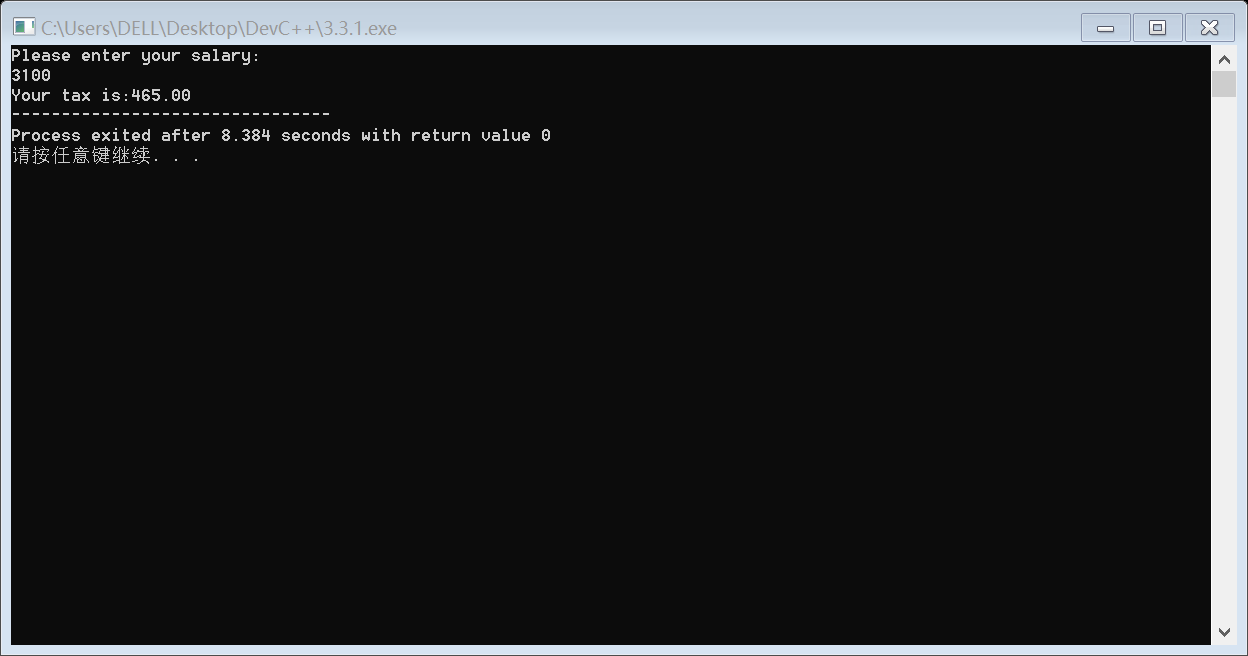
return 0;

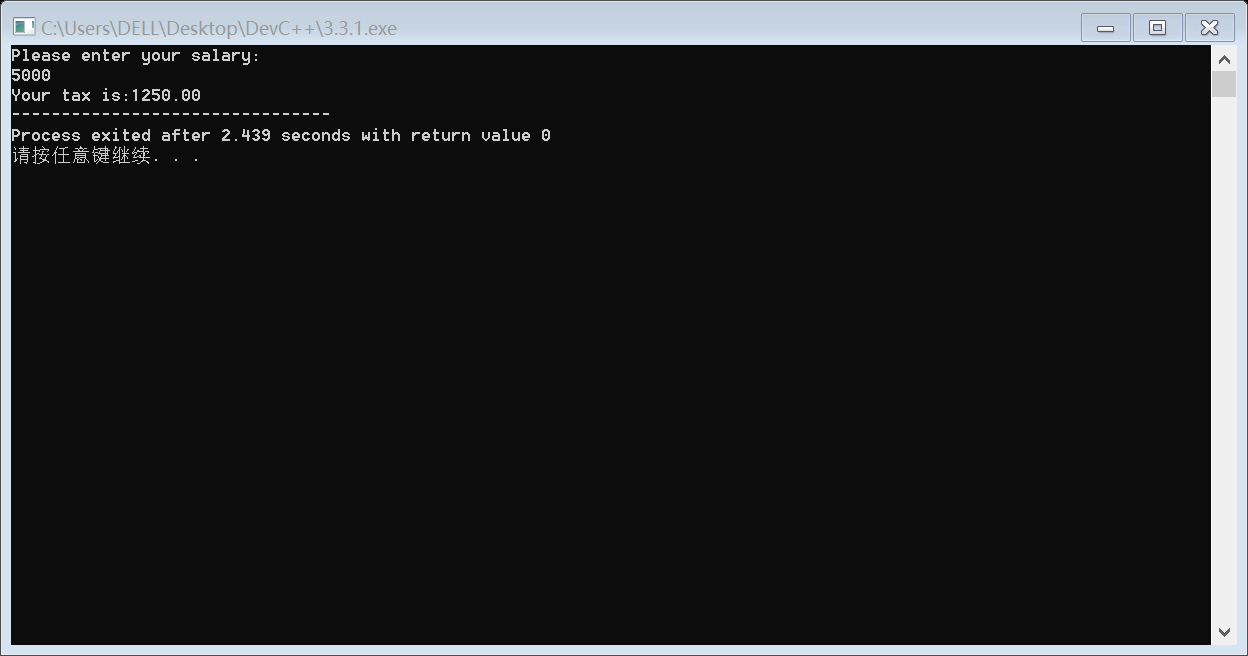
}

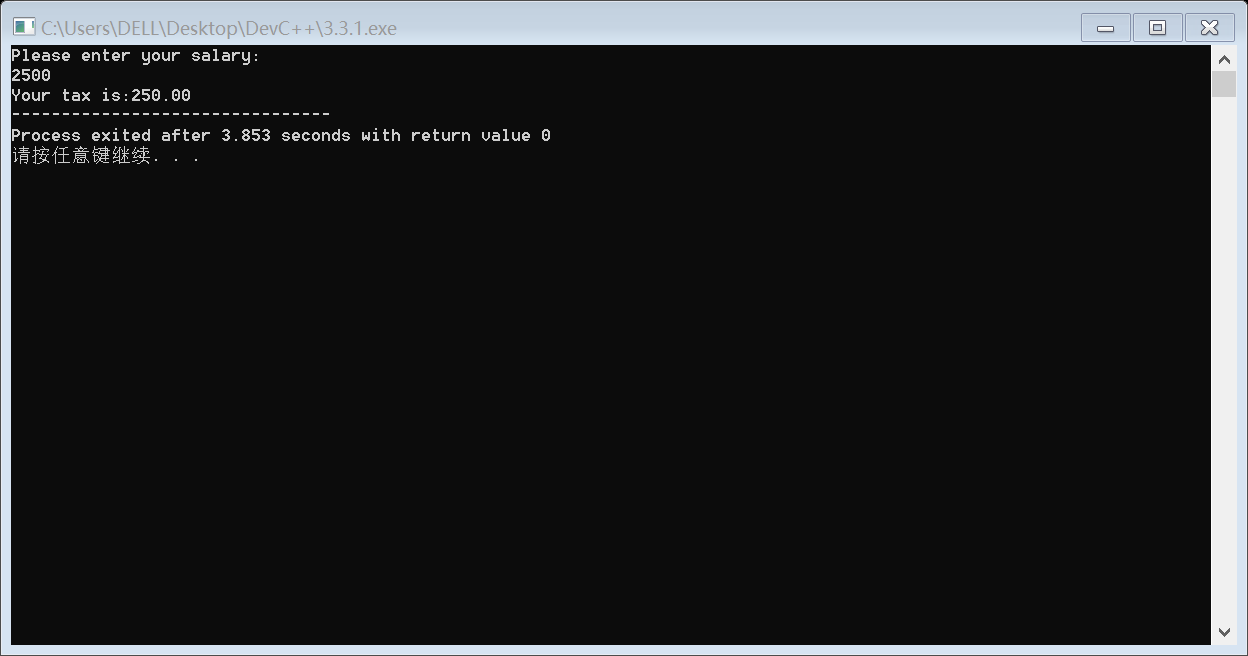
测试

表1-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例一 | 3100 | 465.00 |
| 用例二 | 5000 | 1250.00 |
| 用例三 | 2500 | 250.00 |

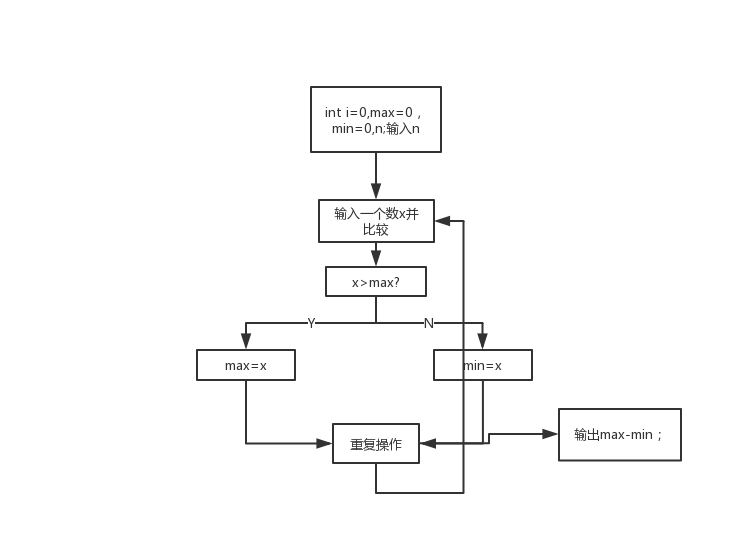






2.输入一只股票连续n天的收盘价格，输出该股票这n天中的最大波动值，波动值是指某天收盘价格与前一天收盘价格之差的绝对值。

解题思路



程序清单

#include<stdio.h>

int main(void)

{

int a[100]; /\*创建数组存储数据\*/

int i,max,min,x,n;

printf("Please enter the days: \n"); /\*输入天数\*/

scanf("%d",&n);

printf("Please enter the value:\n"); /\*输入收盘价格\*/

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

{

scanf("%d ",&a[i]);

}

max=0;

min=a[0];

for(i=0;i<n;i++) /\*开始比较\*/

{

if(a[i]>max)

{

max=a[i];

}

if(a[i]<min)

{

min=a[i];

}

}

x=max-min;

printf("The Difference value is %d",x); /\*输出\*/

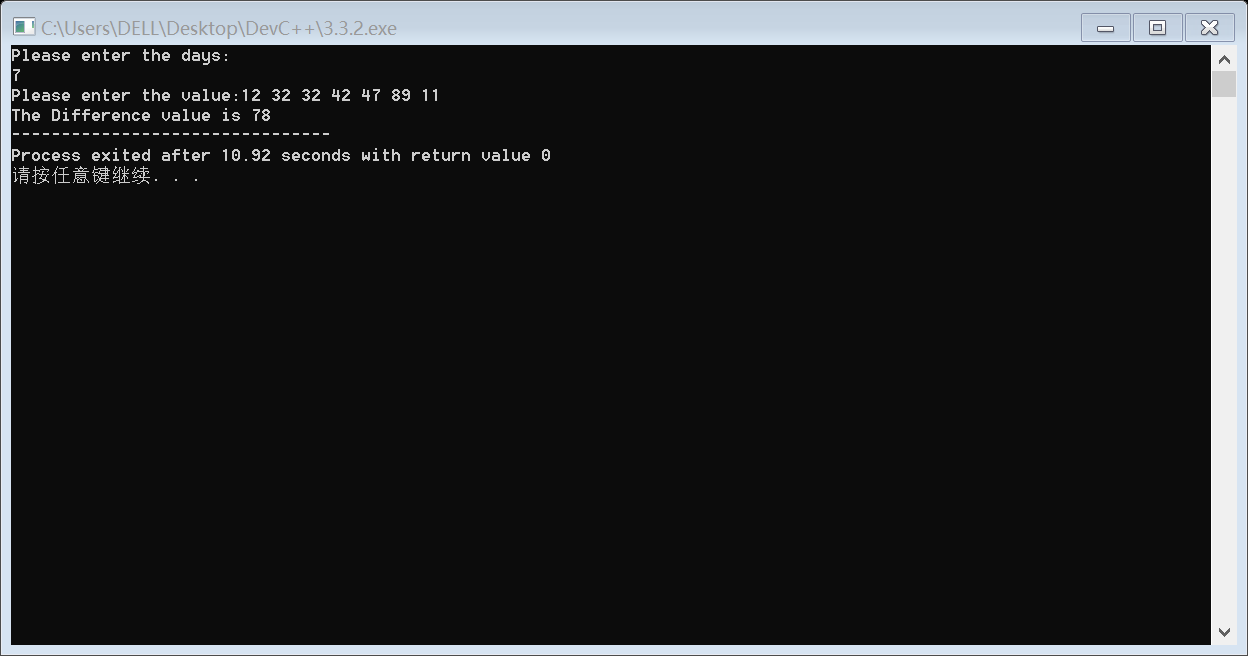
return 0;

}

测试

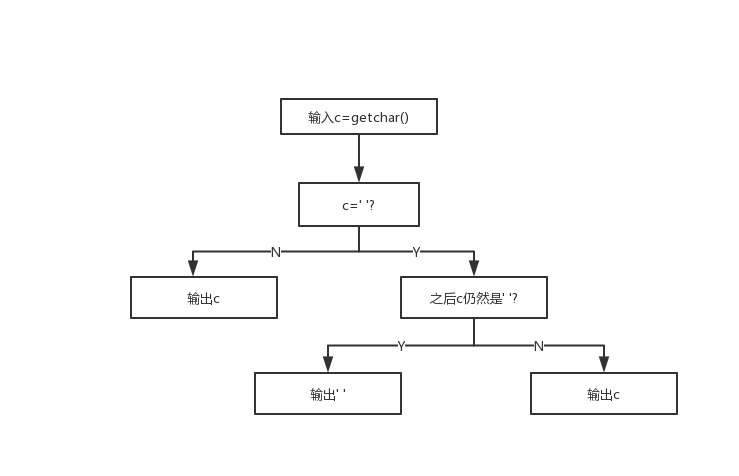
表1-2 编程题2的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 理论结果 |
| 用例一 | 7；12，32，32，42，47，89，11 | 78 |



3.将输入的正文复制到输出，复制过程中将每行一个以上的空格字符用一个空格代替

解题思路



程序清单

#include<stdio.h>

int main(void)

{

char c;

int total = 0; //用于标记连续空格的个数,大于1个的标记为1,否则标记为0

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

do

{

c = getchar();

if (c == ' ')

{

if (!total) //如果当前标记为0

{

printf ("%c", c); //输出一个空格

total = 1; //将标记置为1

}

else

continue; //如果出现连续的空格则跳过

}

else

{

printf ("%c", c); //其他字符原样输出

total = 0; //重新标记空格

}

}while (c != '\n');

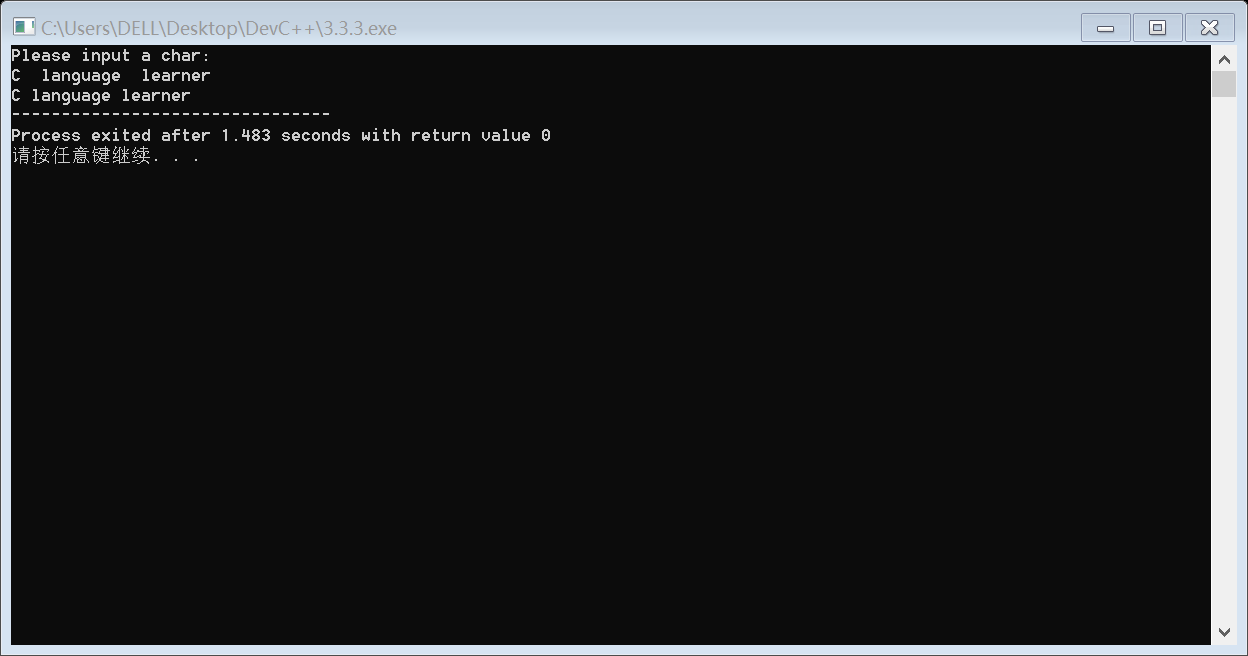
return 0;

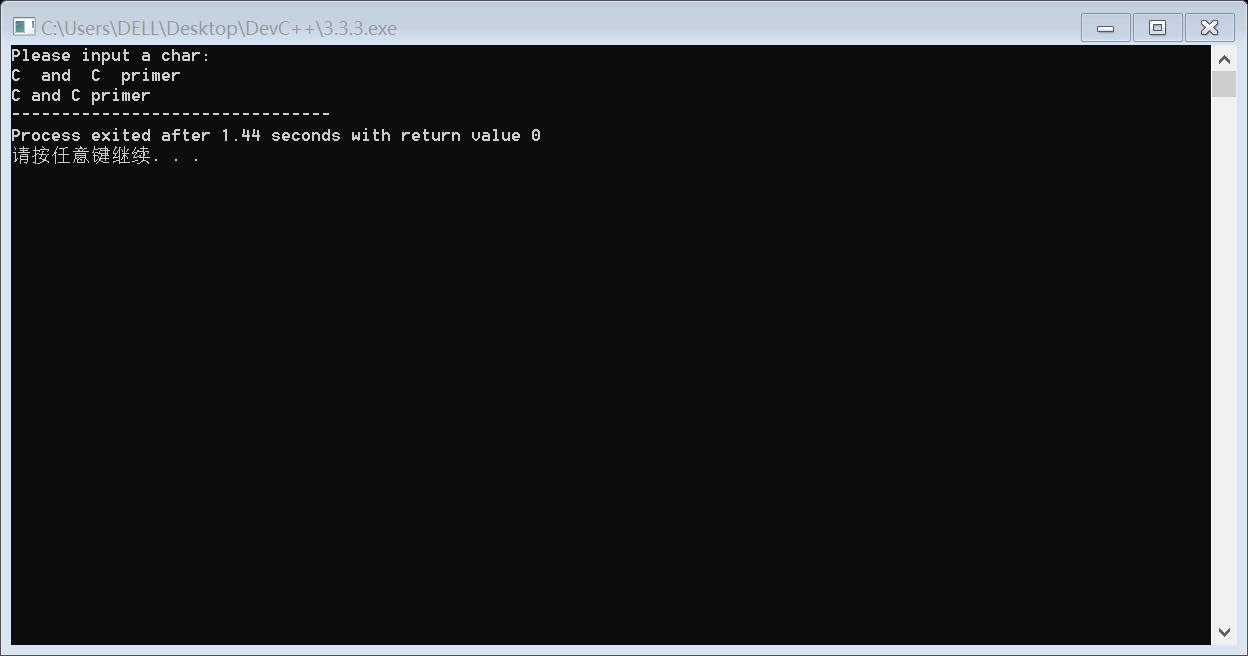
}

测试

表1-3 编程题3的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 理论结果 |
| 用例一 | C language learner | C language learner |
| 用例二 | C and C primer | C and C primer |

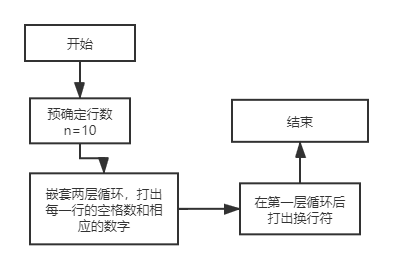




4.打印如下的杨辉三角。

根据公式采用顺推法编程输出金字塔效果的杨辉三角形。特别要注意空格的数目，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。

解题思路



程序清单

#include<stdio.h>

#define n 10 /\*行数\*/

int main(void)

{

int i,j,k,c,m;

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

{

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

{

printf(" "); /\*打印空格\*/

}

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

{

if(j==0)

c=1;

else

c=c\*(i+1-j)/j; /\*根据公式打印相应的数字\*/

printf("%4d",c);

}

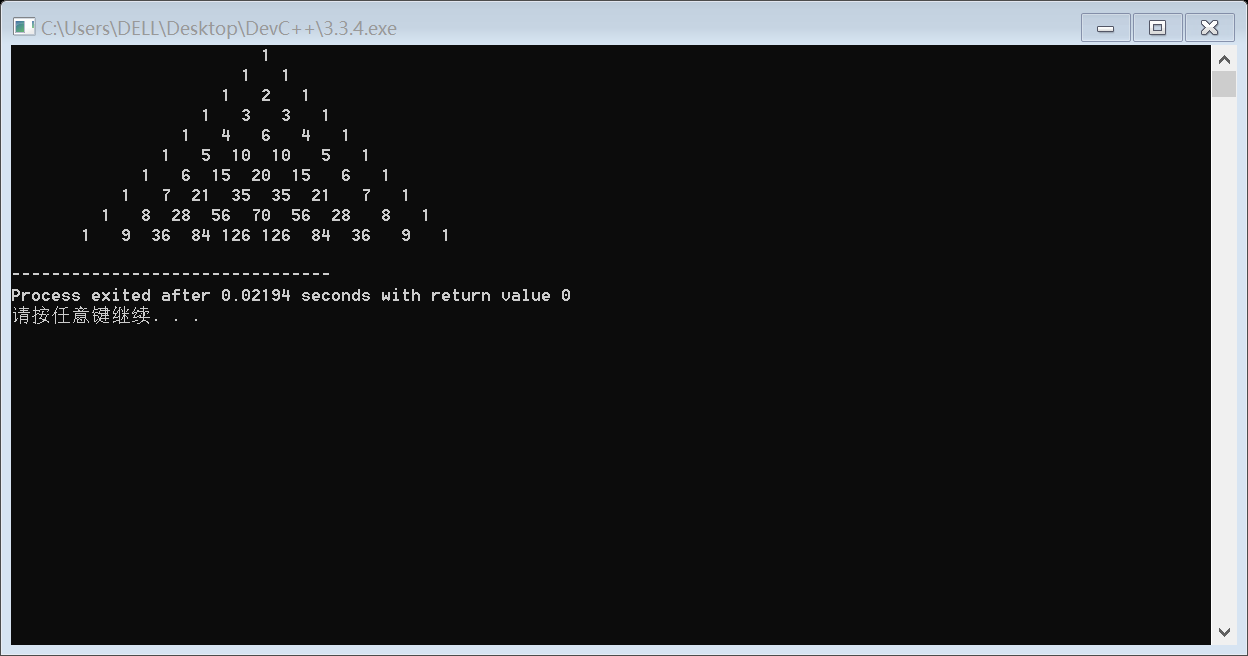
printf("\n");

}

return 0;

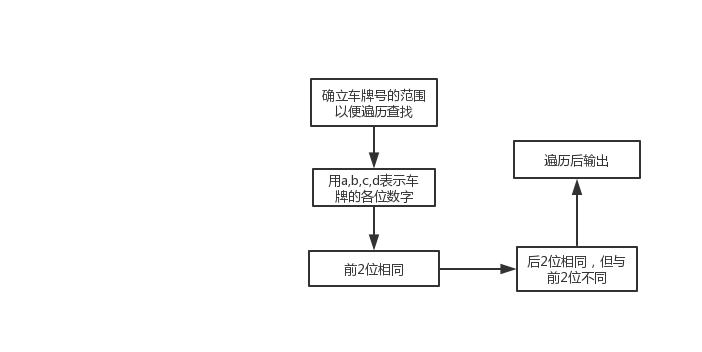
}

测试



5.一辆卡车违反交通规则，撞人后逃跑。现场有三人目击事件，但都没记住车号，只记下车号的一些特征。甲说：牌照的前两位数字是相同的；乙说：牌照的后两位数字是相同的，但与前两位不同；丙是位数学家，他说：四位的车号刚好是一个整数的平方。现在请根据以上线索帮助警方找出车号以便尽快破案。

解题思路



程序清单

#include<stdio.h>

#include<math.h>

int main(void)

{

int n,a,b,c,d;

for(n=1000;n<10000;n++) /\*范围确定\*/

{

d=n%10; /\*个位数\*/

c=(n/10)%10; /\*十位\*/

b=(n/100)%10; /\*百位\*/

a=(n/1000)%10; /\*千位\*/

if((sqrt(n)==int(sqrt(n)))&&(a==b)&&(c==d)&&(a!=c))

{ /\*通过条件筛选\*/

printf("%d",n);

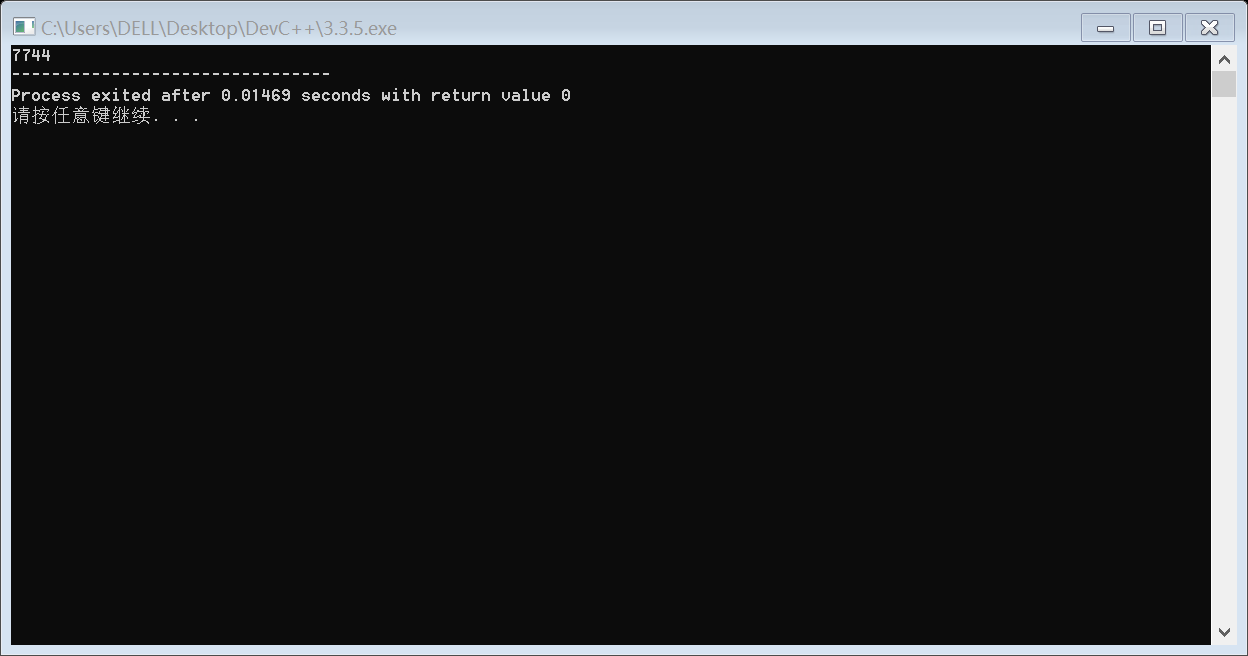
}

}

return 0;

}

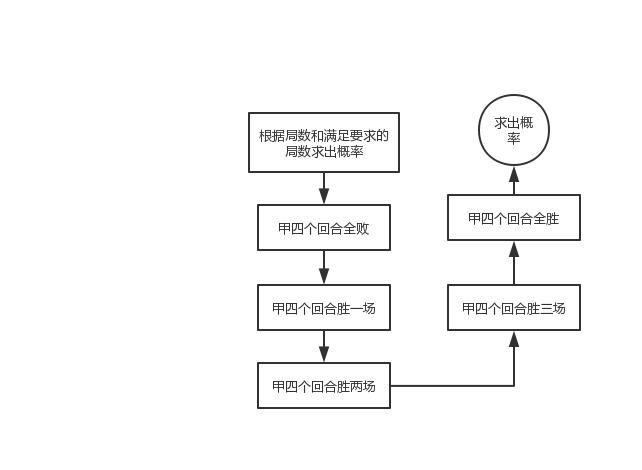
测试



6.【选做】某游戏规则中，甲乙双方每个回合的战斗总是有一方胜利，一方一方失败。失败的一方要把自己的体力值的1/4加给胜利的一方。例如，如果双方的体力值都为4，经过一轮战斗后，双方的体力值会变为5，3.

现在已知：双方开始时的体力值，甲：1000，乙：2000.假设战斗中，甲乙获胜的概率都是50%。求解：双方经过4个回合的战斗，体力值之差小于1000的理论概率。

解题思路



程序清单

#include<stdio.h>

#include<math.h>

double pass=0,count=0; //pass成立次数，count 4回合总可能

void f(double Ta,double Tb,int n){ //定义函数

if(n==4){

if(fabs(Ta-Tb)<1000)

{

pass++;}

count++;

return;

}

double a,b;

a=Ta-Ta\*0.25;b=Tb+Ta\*0.25;//a败

f(a,b,n+1); //递归

a=Ta+Tb\*0.25;b=Tb-Tb\*0.25;//a胜

f(a,b,n+1);

}

int main(void)

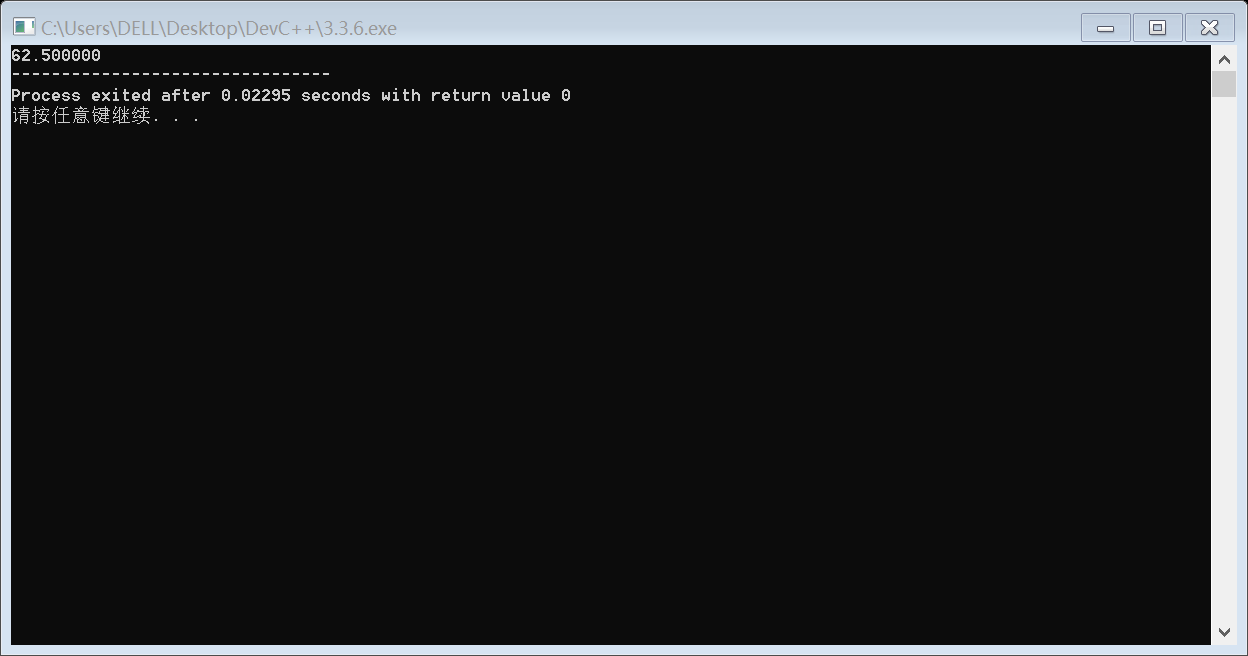
{

f(1000,2000,0);

printf("%f",(pass/count)\*100);

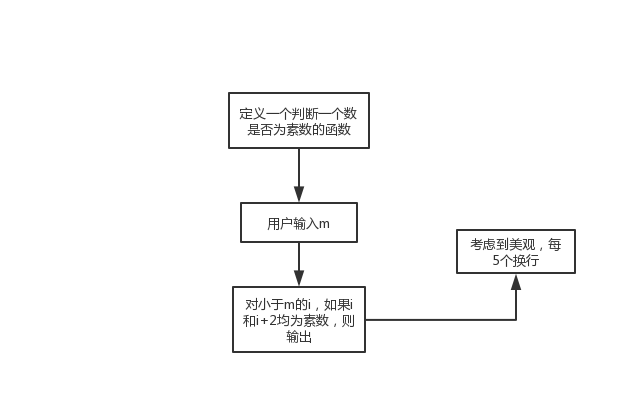
}

测试



7.【选做】用筛法构造素数表，输入m(4<=m<=1000000),查表找出m之前的所有孪生素数。孪生素数就是指距离为2的相邻素数。例如（3，5），（5，7）。

解题思路



程序清单

#include<stdio.h>

#include<math.h>

int IsPrime(int n) //函数定义

{

int j;

for(j=2;j<=sqrt(n);j++)

{

if(n%j==0)

{

return 0;

}

}

return 1;

}

int main(void)

{

int i,count=0,m;

printf("请输入m\n");

scanf("%d",&m);

for(i=3;i<m;i++)

{

if(IsPrime(i)&&IsPrime(i+2)) //判断是否为孪生素数

{

printf("(%d,%d) ",i,i+2);

count++;

if(count%5==0) //每5个换行一次

{

printf("\n");

}

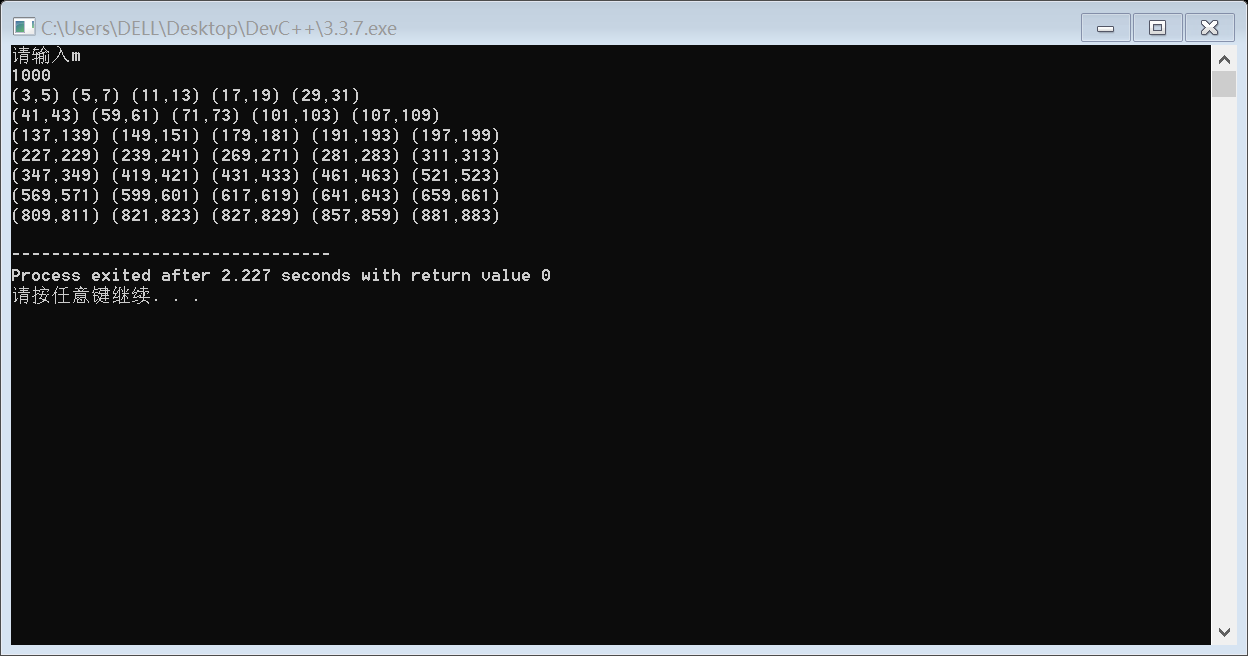
}

}

return 0;

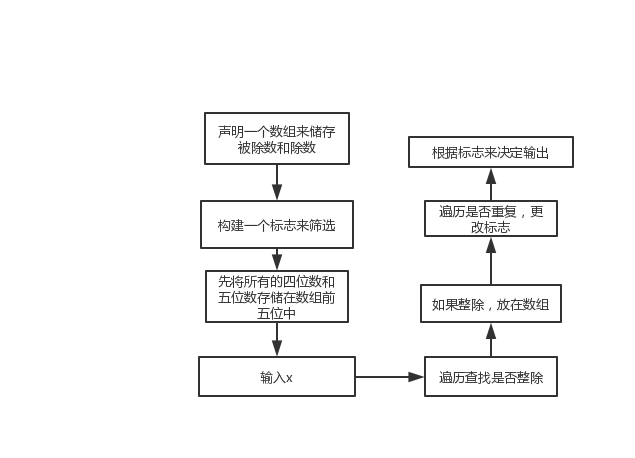
}

测试



8．【选做】输入正整数x(2<=x<=79),输出所有形如abcde/fghij=x的表达式，其中a-j由不同的0-9九个数组成。例如，x=32时，输出为：75168/02349=32

解题思路



程序清单

#include<stdio.h>

int main(void)

{

int x,n,i,j,m,s,a[10],flag;

printf("请输入正整数x：\n");

scanf("%d",&x);

for(n=1234;n<98765;n++)

{

flag=0;

m=n; //防止被更改

for(i=4;i>=0;i--)

{

a[i]=m%10; //存入数组中

m=m/10;

}

if(n%x!=0)

{

continue;

}

else //整除则继续存入

{

s=n/x;

for(i=9;i>=5;i--)

{

a[i]=s%10;

s=s/10;

}

}

for(i=0;i<=8;i++) //查看是否有重复的数字

{

for(j=i+1;j<=9;j++)

{

if(a[i]==a[j])

{

flag=1;

}

}

}

if(flag==0) //无重复则可以输出

{

for(i=0;i<=4;i++)

{

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

}

printf("/");

for(i=5;i<=9;i++)

{

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

}

}

}

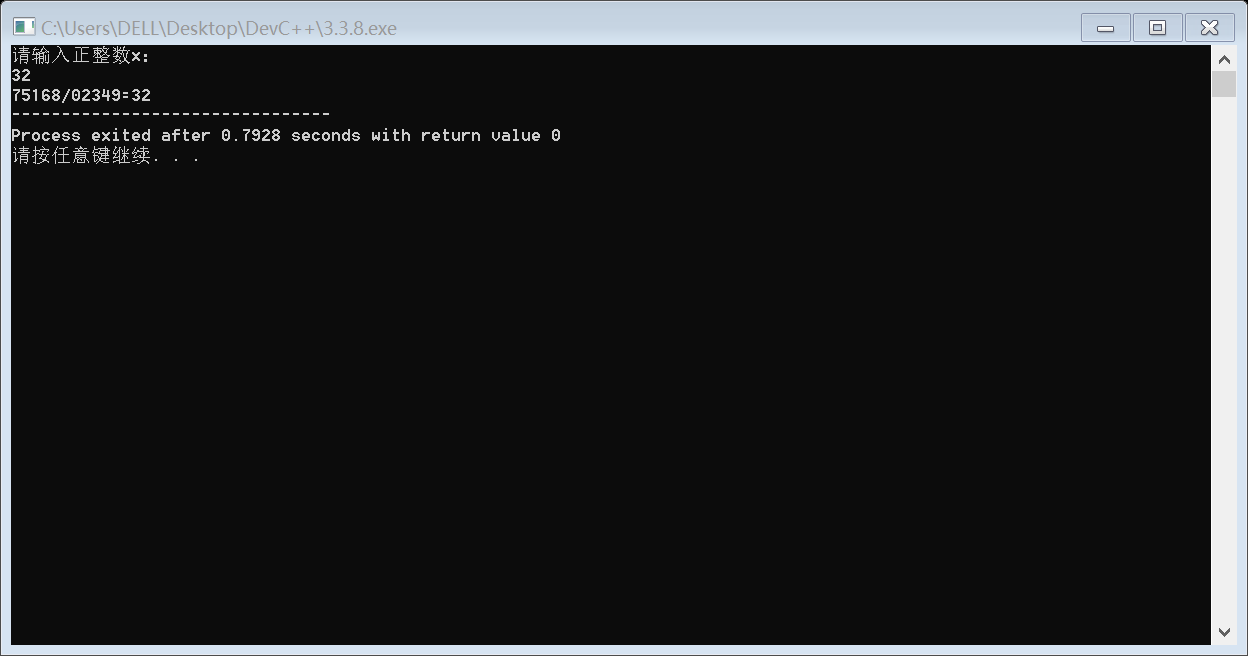
return 0;

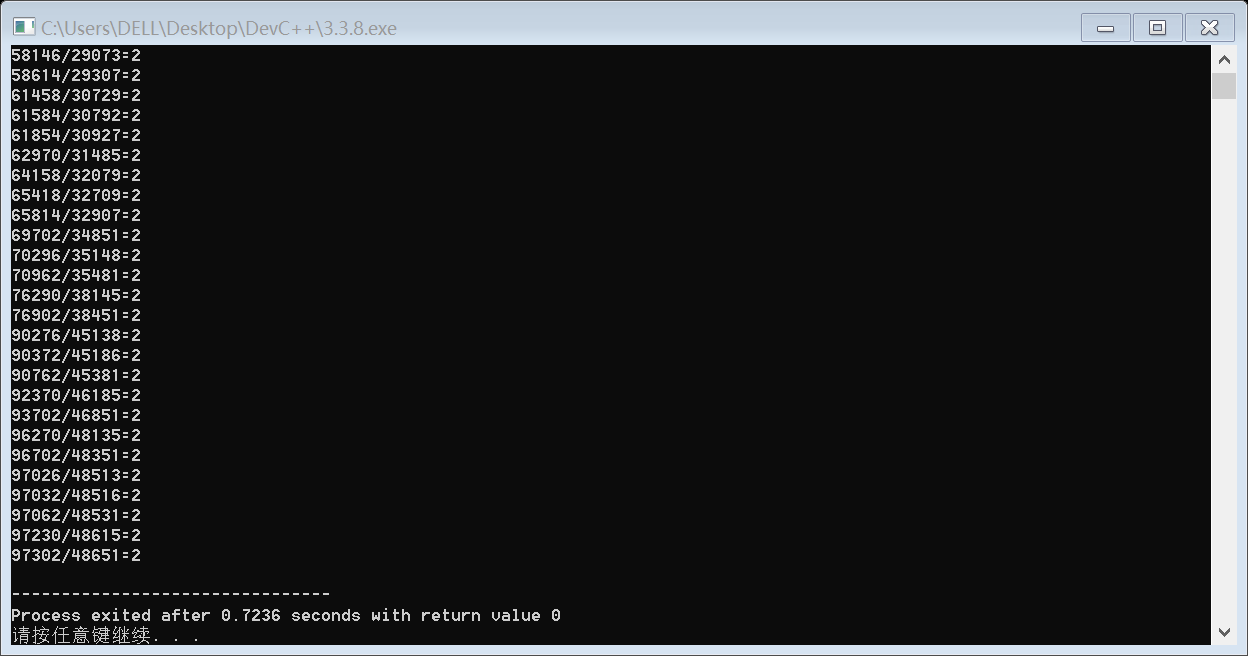
}

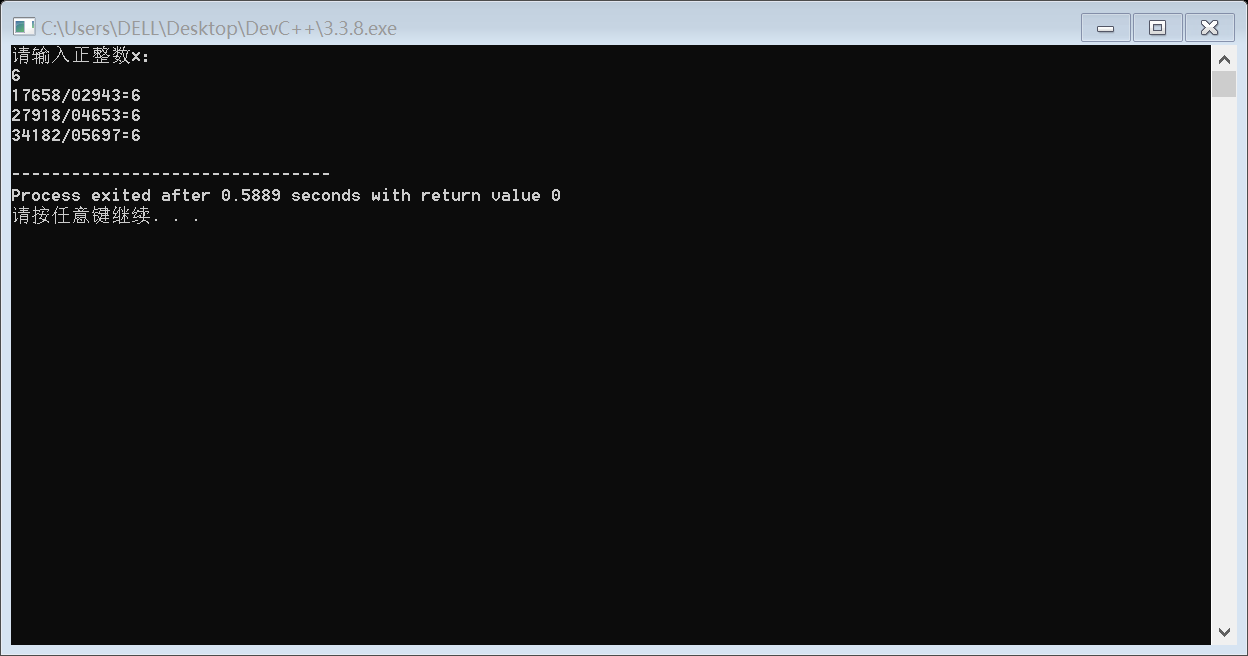
测试

表1-4 编程题8的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例一 | 32 | 75168/02349=32 |
| 用例二 | 2 | 13458/06729=2... |
| 用例三 | 6 | 17658/02943=6... |







## 1.4 小结

本次实验的标题是流程控制，也就是需要灵活地控制程序的输入和输出，懂得按照题目的要求来实现相应的功能。但是，在完成程序中我也碰到了一些困难，也从中学到了代码的规范性。例如，在第3题中将多个空格换成一个空格时，我就忘记了加上break；，导致最后的结果虽然实现了多个空格变为一个，但无法实现完美的复制。

流程控制中有一个递归的难点，对于我们来说，递归确实不是很好理解，我也在网上看了汉诺塔的典型例子，而其他的包括斐波那契数列等等都运用到了递归的知识，也许不是很会用，但是能够掌握和理解也是非常重要的。

另外，在做选做题第6题时，刚看到题目时我一脸懵逼，我也是在网上查找了相关题目才知道这是蓝灯杯竞赛题，查找了解答后我也学到了一个非常巧妙的解法——递归，网上查找答案固然是种学习方式，但最终的还是要能够理解，真正地吸收。

# 

# 2 数组程序设计实验

## 2.1 程序改错与跟踪调试

#include<stdio.h>

void strcate(char [],char []); 错误原因如下

改为：char \* strcate(char [],char []);

void strdelc(char [],char);

int main(void)

{

char a[]="language",b[]="Programming"; 错误原因：b数组太小

改为：b[50]="Programming";

printf("%s %s\n",b,a);

strcate(b,a);

printf("%s %s\n",b,a);

strdelc(b,'a');

printf("%s\n",b);

return 0;

}

void strcate(char t[],char s[]) 错误原因：无返回值

改为：char \* strcate(char t[],char s[])

{

int i=0,j=0;

while(t[i++]);

while((t[i++]=s[j++])!='\0');

}

void strdelc(char s[],char c)

{

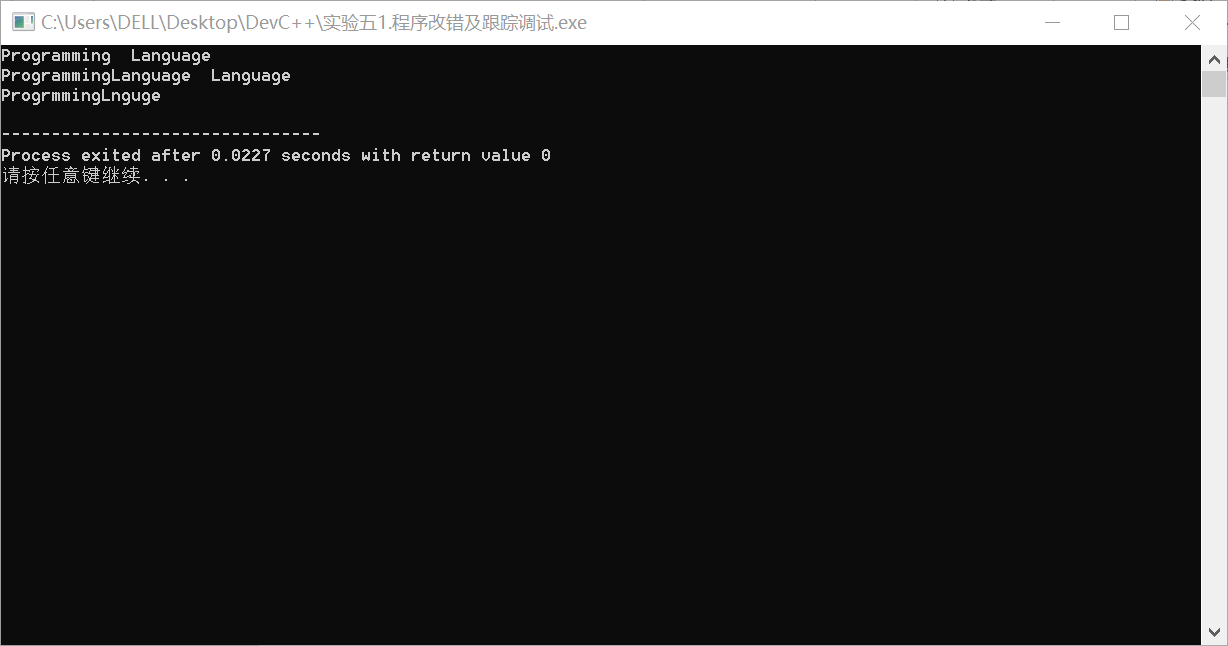
int j,k;

for(j=l=0;s[j]!='\0';j++)

if(s[j]!=c) s[k++]=s[j];

}

运行截图



## 2.2 程序完善与修改替换

在一些文本数据处理中，希望去掉重复（简称去重）的字符，仅保留首次出现的字符。下面的程序能够删除字符串中的重复字符。例如源串为12eerer，去重后位12er。程序采用二重循环，大循环每次从字符串中取出一个字符，小循环遍历串中该字符后面的所有字符，若有重复的字符，则将后面的字符设置为空字符’\0’。

#include<stdio.h>

#include<string.h>

void RemoveDuplicate(char \*s);

int main(void)

{

char str[200];

printf("Input strings,end of Ctrl+z\n");

while(fgets(str,200,stdin)!=NULL)

{

RemoveDuplicate(str);

printf("%s",str);

}

return 0;

}

void RemoveDuplicate(char \*s)

{

int r,w,i,len;

len=strlen(s);

for(r=w=0;r<len;r++)

{

if(s[r]!=EOF)

{

s[r]=s[r];

for(i=r+1;i<len;i++)

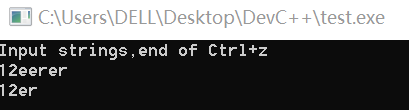
if(s[i]==s[r])

s[i]='\0';

}

}

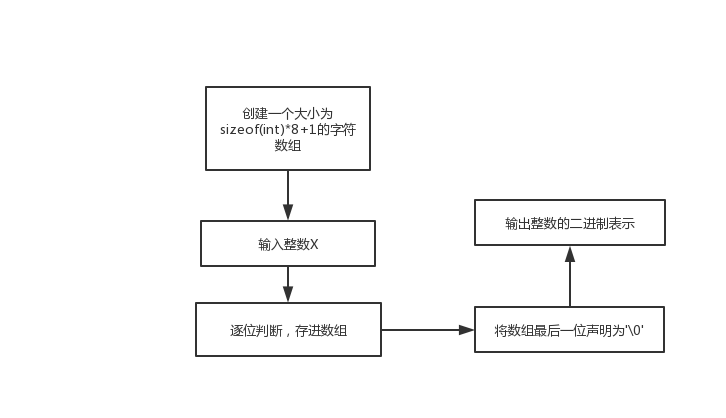
}



## 2.3 程序设计

1.输入一个整数，将它在内存中二进制表示的每一位转换成对应的数字字符并且存放到一个字符数组中，然后输出该整数的二进制表示。

解题思路



程序清单

#include<stdio.h>

#define SIZE sizeof(int)

int main(void)

{

int x;

char a[SIZE\*8+1];

int i;

scanf("%d",&x);

for(i=0;i<SIZE\*8+1;i++,x<<=1)

{

if(x&1<<(SIZE\*8-1))

a[i]='1';

else

a[i]='0';

}

a[SIZE\*8]='\0';

printf("%s",a);

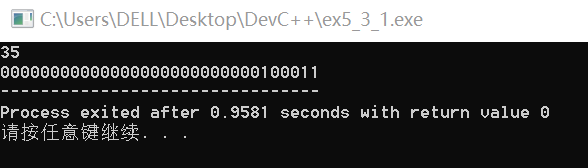
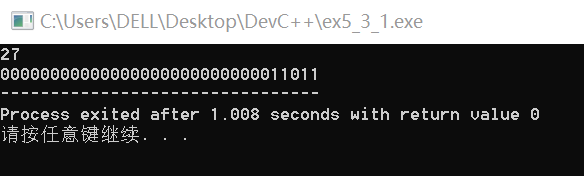
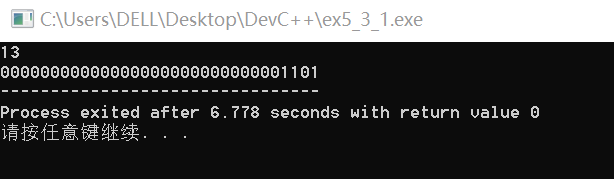
return 0;

}

测试

表2-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例一 | 13 | 00001101 |
| 用例二 | 27 | 00011011 |
| 用例三 | 35 | 00010011 |



2.编写一个C程序，要求采用模块化程序设计思想，将相关功能用函数实现，并提供菜单选项。该程序具有以下功能：

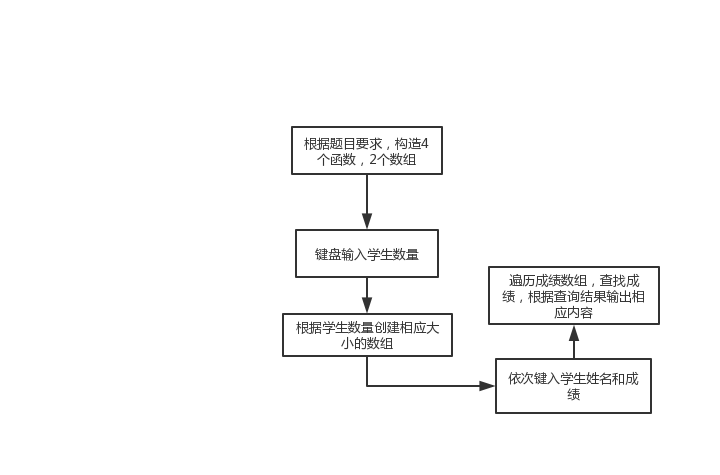
1.输入n个学生的姓名和C语言课程的成绩。

2.将成绩按从高到低的次序排序，姓名同时进行相应调整。

3.输出所有学生的姓名和C语言课程的成绩。

对此题的程序增加查找功能：输入一个C语言课程成绩值，用二分查找进行搜索。如果查找到有该成绩，则输出该成绩学生的姓名和C语言课程的成绩；否则，输出提示“not found！”

解题思路



程序清单

#include<stdio.h>

#include<string.h>

#define STU struct student

STU{

int gra;

char na[20];

};

void input(STU \*p,int n) {

printf("Input names and grades:\n");

int te;

for(te=1; te<=n; te++) {

scanf("%s",&((\*p).na));

scanf("%d",&((\*p).gra));

p++;

}

}

void sort(STU \*p,int n) {

if(n<=1)

return;

int l,r;

l=0;

r=n-1;

STU head=\*p;

while(l!=r) {

while((\*(p+r)).gra<=(\*(p+l)).gra&&l!=r)

r--;

\*(p+l)=\*(p+r);

while((\*(p+r)).gra<=(\*(p+l)).gra&&l!=r)

l++;

\*(p+r)=\*(p+l);

}

\*(p+l)=head;

if(!l)

sort(p,l);

if(r!=n-1)

sort(p+r+1,n-l-1);

}

void output(STU \*p,int n) {

int tem;

for(tem=1; tem<=n; tem++) {

printf("%s\t%d\n",p->na,p->gra);

p++;

}

}

void search(STU \*p,int n,int grade){

int tem;

int judge=1;

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

if((\*(p+tem)).gra==grade){

printf("%s\t%d\n",(\*(p+tem)).na,(\*(p+tem)).gra);

judge=0;

}

}

if(judge){

printf("not found!\n");

}

}

int main()

{

int n;

int jud;

printf("Input the number of students:\n");

scanf("%d",&n);

STU new[n];

STU \*point;

point=new;

input(point,n);

printf("If want to sort,input 1,otherwise input 0:\n");

scanf("%d",&jud);

if(jud)

sort(point,n);

printf("Here are the outputs:\n");

output(point,n);

printf("Input someone's grade for searching:\n");

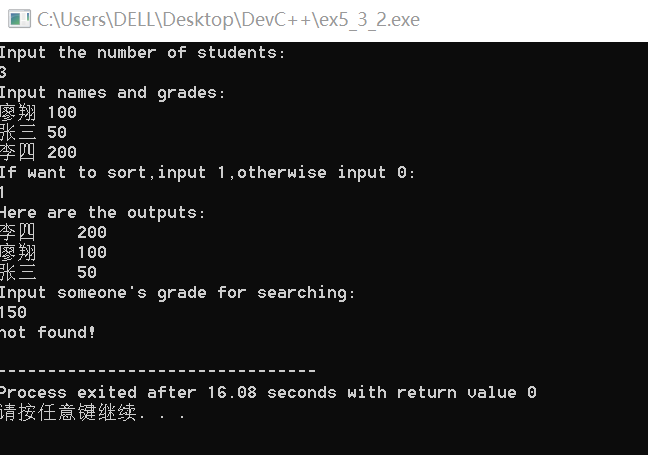
int grade;

scanf("%d",&grade);

search(point,n,grade);

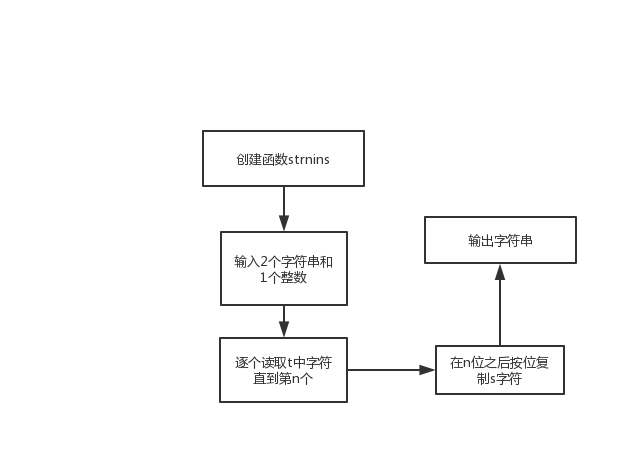
return 0;}

测试



3.编写函数strnins(t,s,n),其功能室：将字符数组s中的字符串插入字符数组t中字符串的第n个字符的后面，编写main函数测试strnins函数的正确性。

解题思路



程序清单

#include<stdio.h>

void strnins(char t[],char s[],int n);

int main(void)

{

char t[15]="C programmar";

char s[30]="language learner";

strnins(t,s,5);

puts(t);

return 0;

}

void strnins(char t[],char s[],int n)

{

int i=0,j=0,a;

while(s[i++]);

while(t[j++]);

for(a=j;a>=n;a--)

{

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

}

j=0;

while(j<i-1)

{

t[j+n]=s[j];

j++;

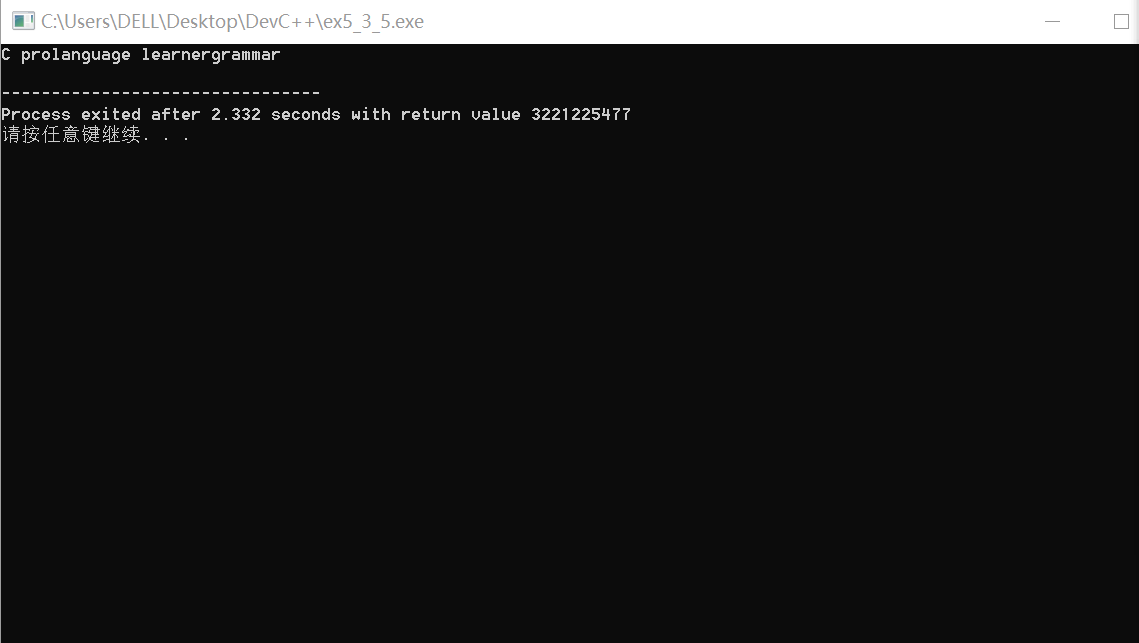
}

}

测试

表2-3 编程题5的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例一 | 见程序清单 | C prolanguage learnergrammar |



4.消除类游戏是一种益智游戏，其核心规则是将一定量的彼此相邻的元素配对消除。现给定一个n行m列的棋盘，棋盘中的每一个方格上放着一个棋子，每个棋子都有颜色，编号用1~9表示。当一行或一列上有连续3个及3个以上同色棋子时，这些棋子都被同时消除，对应的方格用0表示，请输出经过消除后的棋盘。例如，给定棋盘为：

4 4 3 1 4

3 1 1 1 1

4 3 4 1 2

4 4 2 2 2

则输出结果为：

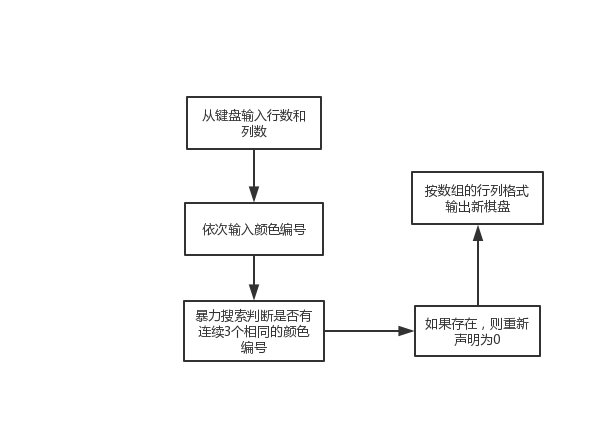
4 4 3 0 4

3 0 0 0 0

4 3 4 0 2

4 4 0 0 0

解题思路



程序清单

#include<stdio.h>

int main()

{

int n, m, i, j;

printf("请输入行数列数\n");

scanf("%d %d",&n, &m);

int a[n][m], b[n][m];

printf("请从左到右，从上到下，依次输入颜色编号\n");

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

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

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

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

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

b[i][j] = 0;

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

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

if((a[i][j]==a[i][j+1]&&a[i][j]==a[i][j+2])||(a[i][j]==a[i][j+1]&&a[i][j]==a[i][j-1])||(a[i][j]==a[i][j-1]&&a[i][j]==a[i][j-2]))

b[i][j] = 1;

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

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

if((a[i][j]==a[i+1][j]&&a[i][j]==a[i+2][j])||(a[i+1][j]==a[i][j]&&a[i][j]==a[i-1][j])||(a[i][j]==a[i-1][j]&&a[i][j]==a[i-2][j]))

b[i][j] = 1;

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

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

if(b[i][j])

a[i][j] = 0;

printf("消除后的棋盘为\n");

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

{

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

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

printf("\n");

}

return 0;

}

测试

表2-4 编程题6的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例一 | 见下图 | 见下图 |



5.整数的划分问题。将一个整数n表示成如下式的一系列正整数的和，称为n的一个分划：n=n1+n2+n3….nk(1<=n1<=n2<=….<=n)。从键盘输入n（范围为1~10），求出n的所有划分。例如，6可以分划为：

6

5+1

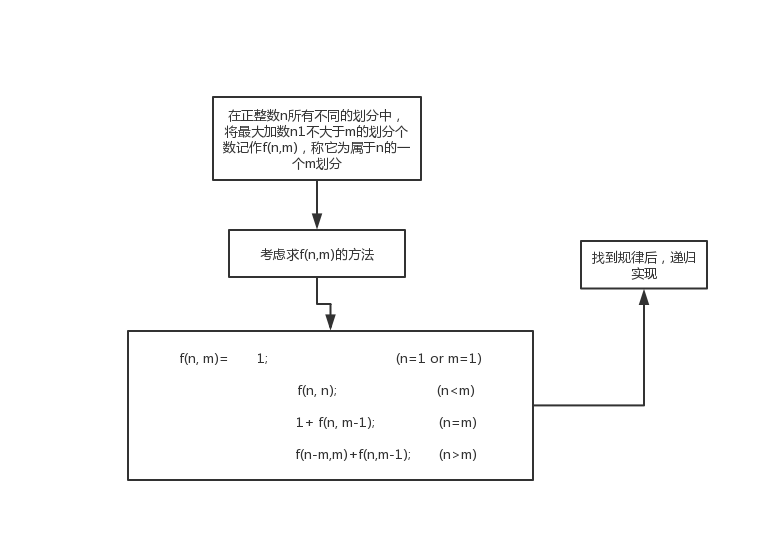
4+2，4+1+1

3+3，3+2+1,3+1+1+1

2+2+2，2+2+1+1，2+1+1+1+1

1+1+1+1+1+1

解题思路



程序清单

#include <stdio.h>

int mark[256];

int n;

void DFS(int sum, int k, int prio) {

if(sum > n)

{

return;

}

else if(sum == n)

{

int i;

printf("=");

for( i = 0; i < k-1; i++)

{

printf("%d+",mark[i]);

}

printf("%d\n",mark[i]);

}

else

{

for(int j = prio; j > 0; j--)

{

mark[k] = j;

sum += j;

DFS(sum,k+1,j);

sum -= j;

}

}

}

int main(void)

{

printf("请输入你要划分的整数（1~10）\n");

scanf("%d", &n);

DFS(0, 0, n);

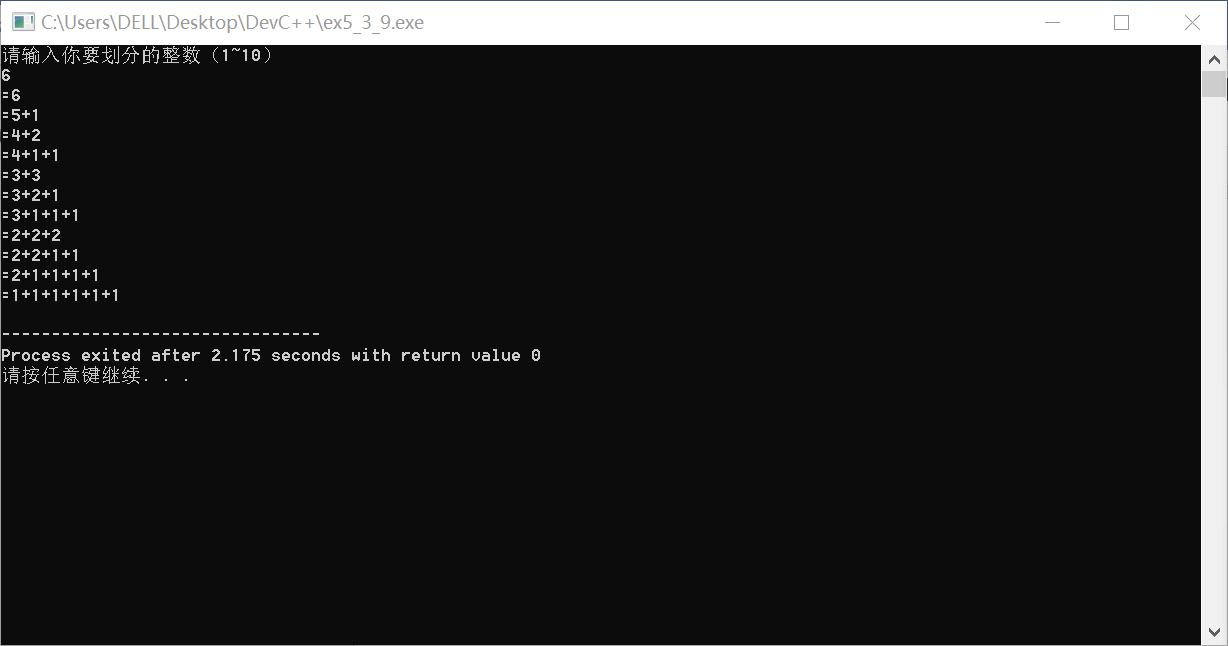
return 0;

}

测试

表2-5 编程题9的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例一 | 6 | 见下图 |



6.迷宫问题。编程找出从入口（左上角方格）经过迷宫到达出口（右下角方格）的所有路径，迷宫问题示意图如图所示，阴影地方是不能通行的，只能从一个空白位置走到与它相邻（四邻域，即上，下，左，右相邻）的空白位置上，且不能走重复路径。

解题思路

递归判断是否到障碍地方，到了回溯到上一个地方，继续试探，一旦找到一条路，则输出\*标记出来

程序清单

#include<stdio.h>

void visited(int (\*pa)[12], int i, int j)

{

int n, m;

pa[i][j] = 1;

if(i == 6 && j == 10){

printf("find path: \n");

for(n=0; n<8; n++){

for(m=0; m<12; m++){

if(pa[n][m] == 1)

printf("\* ");

else if(pa[n][m] == 2)

printf("2 ");

else

printf("0 ");

}

printf("\n");

}

}

if(pa[i-1][j] == 0)

visited(pa, i-1, j);

if(pa[i+1][j] == 0)

visited(pa, i+1, j);

if(pa[i][j-1] == 0)

visited(pa, i, j-1);

if(pa[i][j+1] == 0)

visited(pa, i, j+1);

pa[i][j] = 0;

}

int main()

{

int a[8][12] = {

2,2,2,2,2,2,2,2,2,2,2,2,

2,0,2,2,2,2,2,2,2,2,2,2,

2,0,2,0,0,2,0,0,0,0,2,2,

2,0,0,0,2,2,0,0,2,0,2,2,

2,0,2,2,0,0,0,2,2,0,2,2,

2,0,0,0,0,2,0,0,0,0,2,2,

2,2,2,2,2,2,2,2,2,0,0,2,

2,2,2,2,2,2,2,2,2,2,2,2,

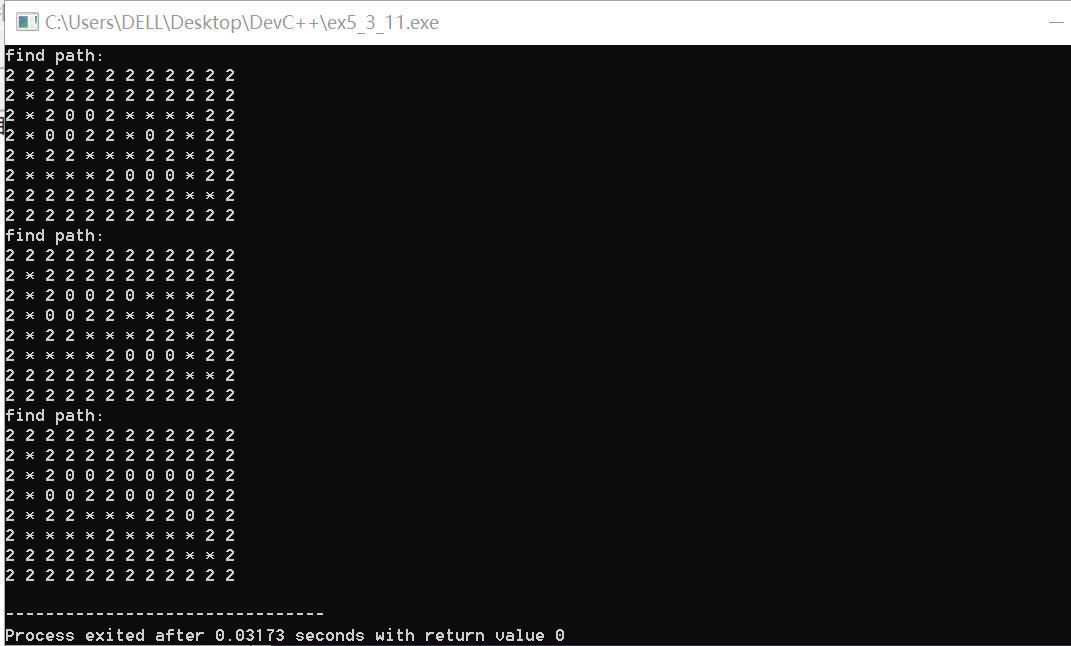
};

visited(a, 1, 1);

return 0;

}

测试



## 2.4小结

通过这次数组的实验，我了解到了C语言另一有力工具数组的功能，通过数组可以实现许多算法，包括处理字符串函数，在数字串与数之间转换等等函数功能，这使得我们的程序可以变得更加的灵活，也增添了我们程序的功能，可以解决更多问题。

一维数组用来冒泡排序是十分有用的，但是可能复杂度比较高，O(n^2)，而用字符数组来处理字符也是十分方便的，除了一维数组之外，用二维数组来处理一些有关联的量也是十分便捷的，并且处理后还能够很直观的表示相关变量的关系，但更多维的数组的用武之地可能就比较少了。

这次实验的第五题着实有一点费脑筋，要通过函数两个变量的关系来推出一个递归表达式，从来能够减少程序的运算量，使得程序运行更快。

而第六题走迷宫则运用了回溯递归这一算法，就是不停地去试探，在一个位置不停地去试探下一个可能的行走方向，一旦找到，就用相关的符号标记出来。

就我写的这些为数不多的程序来说，数组占了很多内容和处理效果，也是日后编写程序的一个有利武器。

# 3 结构与联合实验

# 3.1 表达式求值的程序验证

设有下面说明，请先自己计算表7-1中表达式的值，然后通过编程计算来加以验证。各表达式之间相互无关。

#include<stdio.h>

int main(void)

{

char u[]="UVWXYZ",v[]="xyz";

struct T{

int x;

char c;

char \*t;

}a[]={{11,'A',u},{100,'B',v}},\*p=a;

}

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 表达式 | 计算值 | 验证值 |
| 1 | (++p)->x | 100 | 100 |
| 2 | p++,p->c | B | B |
| 3 | \*p++->t,\*p->t | x | x |
| 4 | \*(++p)->t | x | x |
| 5 | \*++p->t | V | V |
| 6 | ++\*p->t | V | V |

## 3.2 程序完善与修改替换

给定一批整数，以0作为结束标志且不作为结点，将其建成一个先进先出的链表，先进先出链表的头指针始终指向最先创建的结点（链头），先建结点指向后建结点，后建结点始终是尾结点。

1. 源程序中存在什么样的错误（先观察执行结果）？对程序进行修改，调试，使之能够正确完成指定任务。

#include<stdio.h>

#include<stdlib.h>

struct s\_list{

int data;

struct s\_list \* next;

};

void create\_list(struct s\_list \*\* headp,int \*p); 注意这里要用二级指针

int main(void)

{

struct s\_list \* head =NULL, \*p;

int s[]={1,2,3,4,5,6,7,8,0};

create\_list(&head,s);

p=head;

while(p){

printf("%d\t",p->data);

p=p->next;

}

printf("\n");

return 0;

}

void create\_list(struct s\_list \*\* headp,int \*p)

{

struct s\_list \* loc\_head =NULL,\* tail;

if(p[0]==0)

;

else{

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++;

tail = loc\_head;

while(\*p){

tail->next = (struct s\_list \*)malloc(sizeof(struct s\_list));

tail = tail->next;

tail->data=\*p++;

}

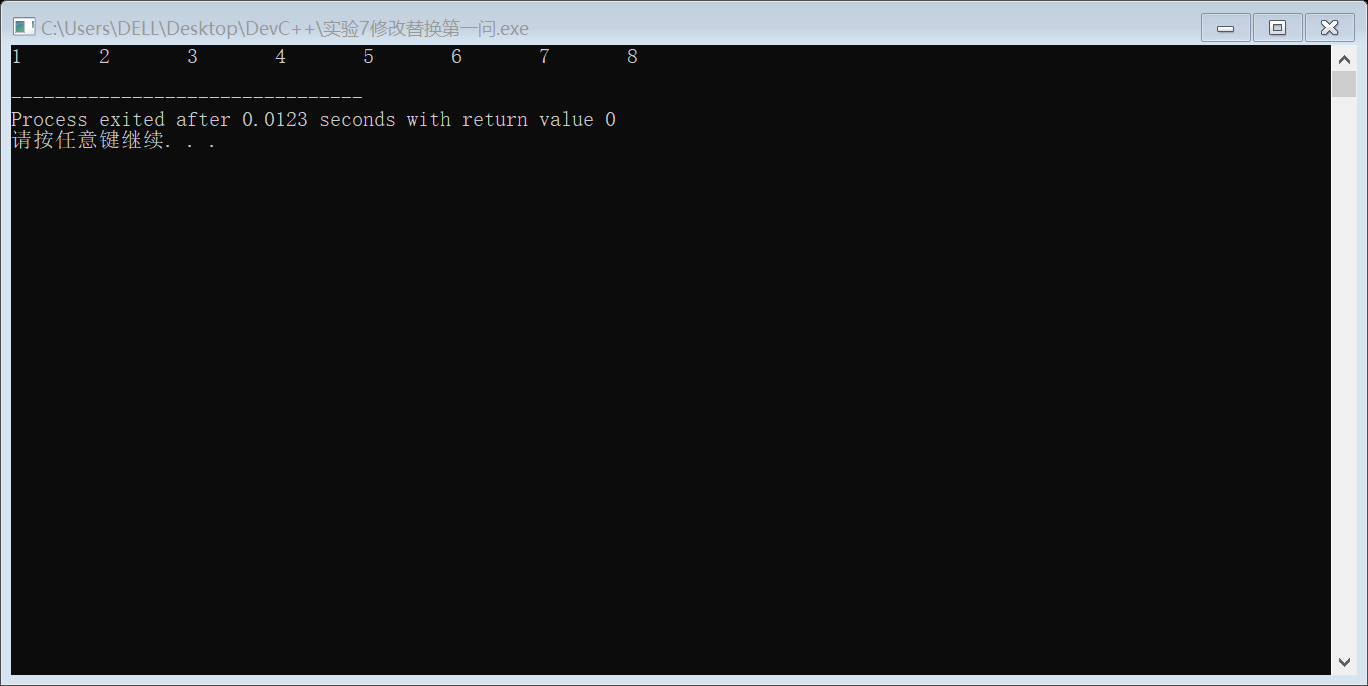
tail->next = NULL;

}

\*headp=loc\_head;

}

运行截图



1. 修改替换create\_list函数，将其建成一个后进后出的链表，先进后出链表的头指针始终指向最后创建的结点（链头），后建结点指向先建结点，先建结点始终是尾结点。

#include<stdio.h>

#include<stdlib.h>

struct s\_list{

int data;

struct s\_list\*next;

};

void creat\_list(struct s\_list\*\*headp,int \*p);

int main(void)

{

struct s\_list \*head=NULL,\*p;

int s[]={1,2,3,4,5,6,7,8,0};

creat\_list(&head,s);

p=head;

while(p){

printf("%d\t",p->data);

p=p->next;

}

printf("\n");

return 0;

}

void creat\_list(struct s\_list \*\* headp,int \*p)

{

struct s\_list\*loc\_head=NULL,\*tail;

struct s\_list\*temp;

if(p[0]==0)

;

else

{

loc\_head=(struct s\_list \*)malloc(sizeof(struct s\_list));

loc\_head->data=\*p++;

tail=loc\_head;

while(\*p){

temp=(struct s\_list\*)malloc(sizeof(struct s\_list));

temp->next=loc\_head;

loc\_head=temp;

loc\_head->data=\*p++;

}

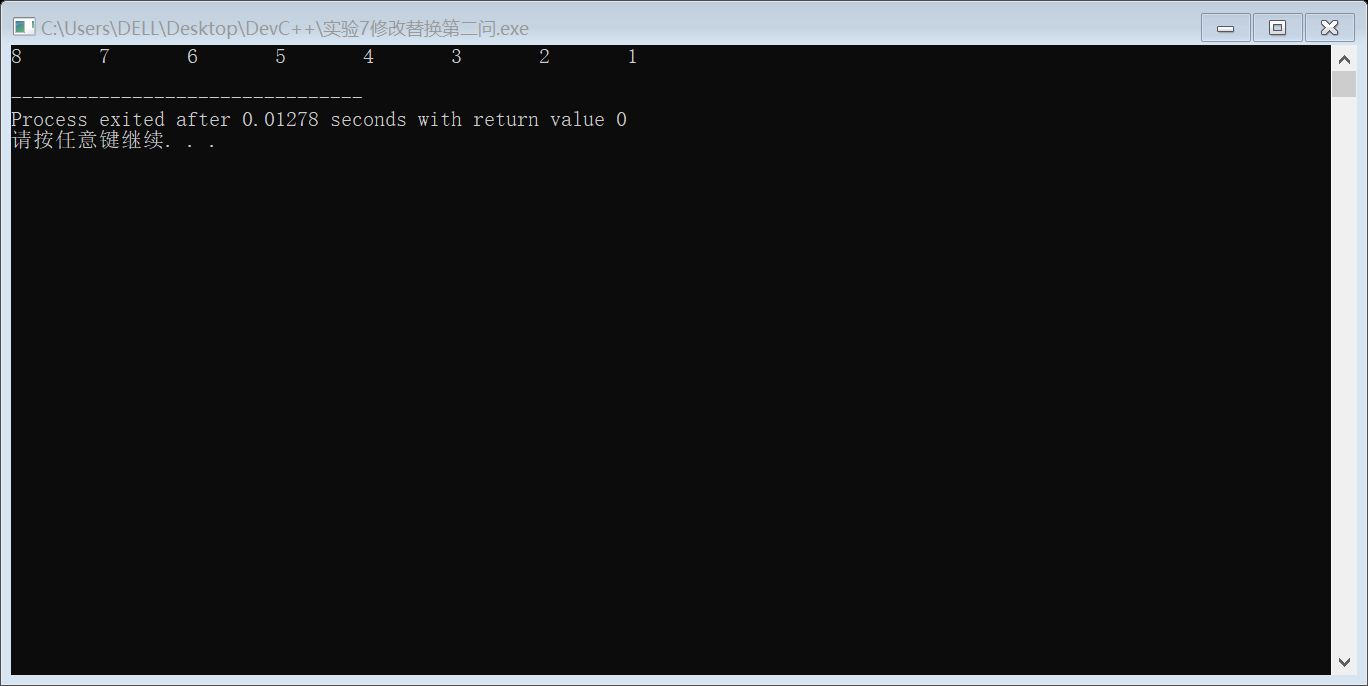
tail->next=NULL;

}

\*headp=loc\_head;

}

运行截图



## 3.3 程序设计

1.设计一个字段结构struct bits,它将一个8位无符号字节从最低位向最高位声明为8个字段，各字段依次为bit0,bit1,…,bit7，且bit0的优先级最高。同时，设计8个函数，第i个函数以biti(i=0,1,2,…,7)为参数，并且在函数体内输出biti的值。将8个函数的名字存入一个函数指针数组p\_fun。如果bit0为1，调用p\_fun[0]指向的函数。如果struct bits中有多位为1，则根据优先级从高到低顺序依次调用函数指针数组p\_fun中相应元素指向的函数。8个函数中的第0个函数可以设计为：

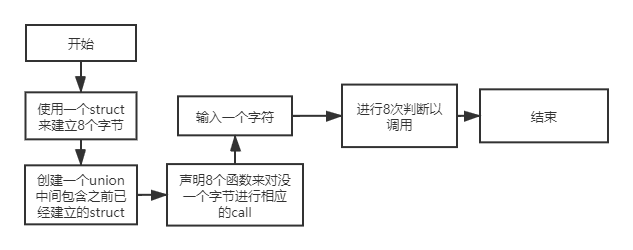
void f0(int b)

{

printf(“the function %d is called!\n”,b);

}

解题思路



程序清单

#include<stdio.h>

struct bits{

unsigned char bit0:1,bit1:1,bit2:1,bit3:1,bit4:1,bit5:1,bit6:1,bit7:1;

}a;

union w{

unsigned char t;

struct bits a;

}m;

void f0(struct bits b){

printf("the 1 function %d is called!\n",b.bit0);

}

void f1(struct bits b){

printf("the 2 function %d is called!\n",b.bit1);

}

void f2(struct bits b){

printf("the 3 function %d is called!\n",b.bit2);

}

void f3(struct bits b){

printf("the 4 function %d is called!\n",b.bit3);

}

void f4(struct bits b){

printf("the 5 function %d is called!\n",b.bit4);

}

void f5(struct bits b){

printf("the 6 function %d is called!\n",b.bit5);

}

void f6(struct bits b){

printf("the 7 function %d is called!\n",b.bit6);

}

void f7(struct bits b){

printf("the 8 function %d is called!\n",b.bit7);

}

int main(void)

{

void(\*p\_fun[8])(struct bits b);

printf("input char:");

m.t=getchar();

m.a.bit0=1;

p\_fun[0]=f0;

p\_fun[1]=f1;

p\_fun[2]=f2;

p\_fun[3]=f3;

p\_fun[4]=f4;

p\_fun[5]=f5;

p\_fun[6]=f6;

p\_fun[7]=f7;

if(m.a.bit0) p\_fun[0](m.a);

if(m.a.bit1) p\_fun[1](m.a);

if(m.a.bit2) p\_fun[2](m.a);

if(m.a.bit3) p\_fun[3](m.a);

if(m.a.bit4) p\_fun[4](m.a);

if(m.a.bit5) p\_fun[5](m.a);

if(m.a.bit6) p\_fun[6](m.a);

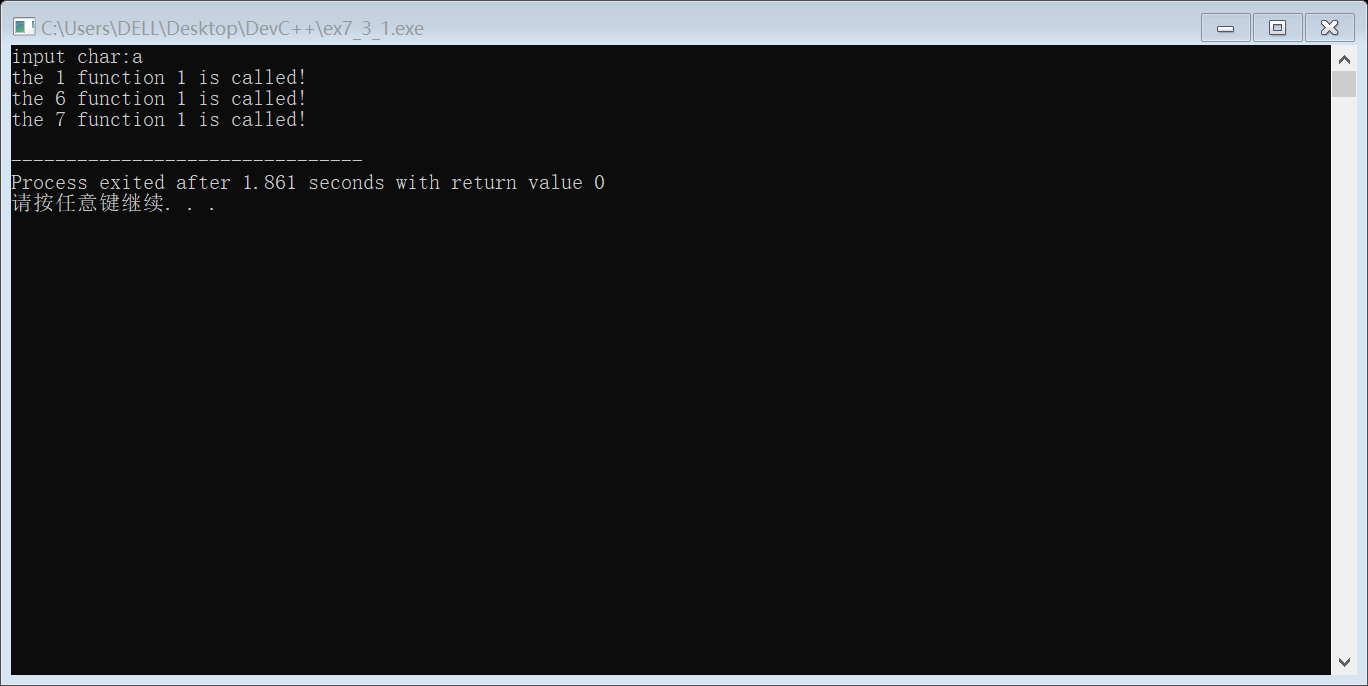
if(m.a.bit7) p\_fun[7](m.a);

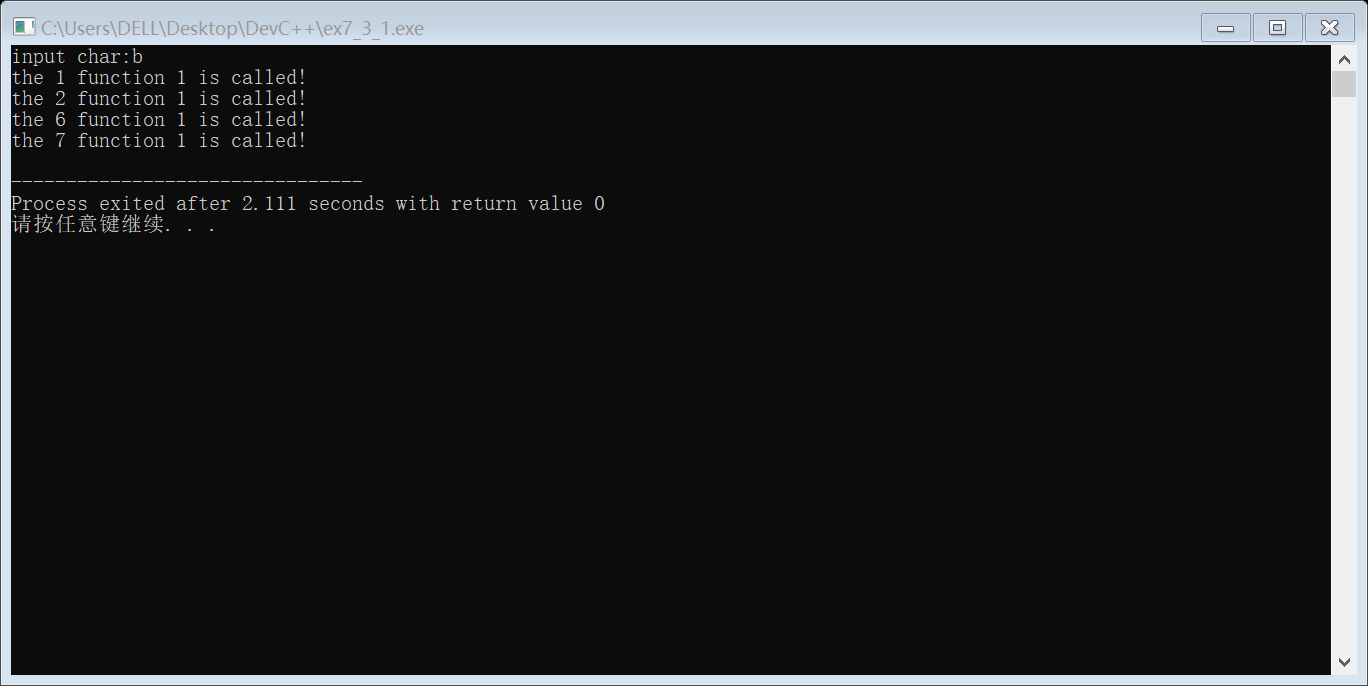
return 0;

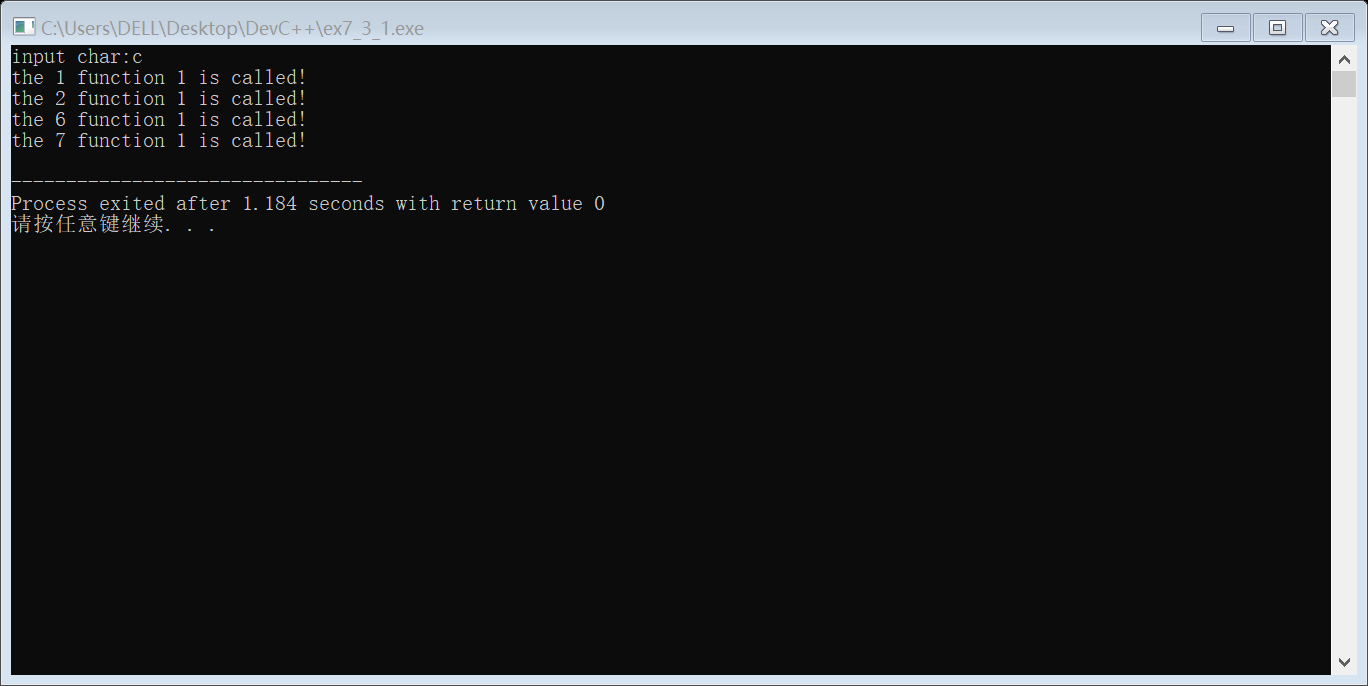
}测试

表2-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例一 | a | 如图 |
| 用例二 | b | 如图 |
| 用例三 | c | 如图 |







2.用单向链表建立一张班级成绩单，包括每个学生的学号，姓名，英语，高等数学，普通物理，C语言程序设计4门课程的成绩。用函数编程实现下列功能，并提供菜单选项。

（1）输入每个学生的各项信息

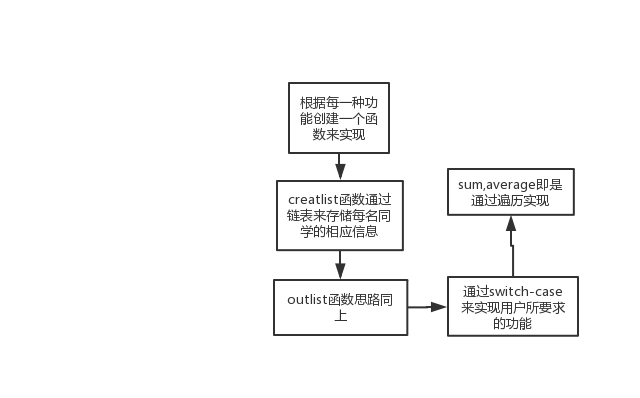
（2）输出每个学生的各项信息

（3）修改指定学生的指定数据项的内容

（4）统计每个学生的平均成绩（保留2位小数）

（5）输出各位学生的学号，姓名，4门课程的总成绩和平均成绩

解题思路



程序清单

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct stu{

long num;

char s[10];

int c1;

int c2;

int c3;

int c4;

struct stu\*next;

};

void creatlist(struct stu \*\*headp);

void outputlist(struct stu \*\*headp);

void correctlist(struct stu \*\*headp);

void avstu(struct stu \*\*headp);

void sumstu(struct stu \*\*headp);

int main(void)

{

struct stu \*head=NULL;

creatlist(&head);

outputlist(&head);

correctlist(&head);

avstu(&head);

sumstu(&head);

return 0;

}

void creatlist(struct stu \*\*headp)

{

\*headp=(struct stu \*)malloc(sizeof(struct stu));

struct stu \*tail=\*headp,\*p=\*headp;

printf("please input the following item divided by squre\n");

printf("ID\tname\tEnglishgrade\tmathgrade\tphysicsgrade\tC language grade");

printf("\nEnd with 0\n");

for(;;)

{

scanf("%ld",&p->num);

if(p->num==0) break;

scanf("%s %d %d %d %d",p->s,&p->c1,&p->c2,&p->c3,&p->c4);

p=(struct stu \*)malloc(sizeof(struct stu));

tail->next=p;

tail=p;

}

tail->next=NULL;

}

void outputlist(struct stu \*\*headp)

{

struct stu \*p=\*headp;

while(p->next!=NULL)

{

printf("ID %ld NAME %s ENG %d MATH %d PHY %d C %d\n"

,p->num,p->s,p->c1,p->c2,p->c3,p->c4);

p=p->next;

}

}

void correctlist(struct stu \*\*headp)

{

char c;

struct stu \*p;

p=\*headp;

printf("1:look up by name\n");

printf("2:look up by number\n");

c=getchar();

c=getchar();

switch(c)

{

case '1':printf("please input the name\n");

char t[10];

scanf("%s",t);

for(;p->next!=NULL&&strcmp(p->s,t);p=p->next)

;

if(!strcmp(p->s,t))

printf("FIND ID %ld NAME %s ENG %d MATH %d PHY %d C %d\n"

,p->num,p->s,p->c1,p->c2,p->c3,p->c4);

else if(p->next==NULL)

{

printf("NOT FOUND\n");

return ;

}

break;

case '2':printf("please input the number\n");

long tmp;

scanf("%ld",&tmp);

for(;p->next!=NULL&&tmp!=p->num;p=p->next);

if(tmp==p->num)

printf("FIND ID %ld NAME %s ENG %d MATH %d PHY %d C %d\n"

,p->num,p->s,p->c1,p->c2,p->c3,p->c4);

else if(p->next==NULL)

{

printf("NOT FOUND\n");

return ;

}

break;

}

printf("which one to change?\n");

printf("1:number\n");

printf("2:name\n");

printf("3:Grade of English\n");

printf("4:Grade of Math\n");

printf("5:Grade of Physics\n");

printf("6.Grade of C Language\n");

printf("7:I don't want to change anything\n");

c=getchar();

c=getchar();

printf("please reinput\n");

switch(c){

case '1':scanf("%ld",&p->num);break;

case '2':scanf("%s",p->s);break;

case '3':scanf("%d",&p->c1);break;

case '4':scanf("%d",&p->c2);break;

case '5':scanf("%d",&p->c3);break;

case '6':scanf("%d",&p->c4);

default: return ;

}

}

void avstu(struct stu \*\*headp)

{

struct stu \*p=\*headp;

for(;p->next!=NULL;p=p->next)

{

double tp;

tp=(p->c1+p->c2+p->c3+p->c4)/4.0;

printf("ID %ld Name %s average %.2lf\n",p->num,p->s,tp);

}

}

void sumstu(struct stu \*\*headp)

{

struct stu \*p;

p=\*headp;

for(;p->next!=NULL;p=p->next)

{

double tp;

int sum;

sum=p->c1+p->c2+p->c3+p->c4;

tp=sum/4.0;

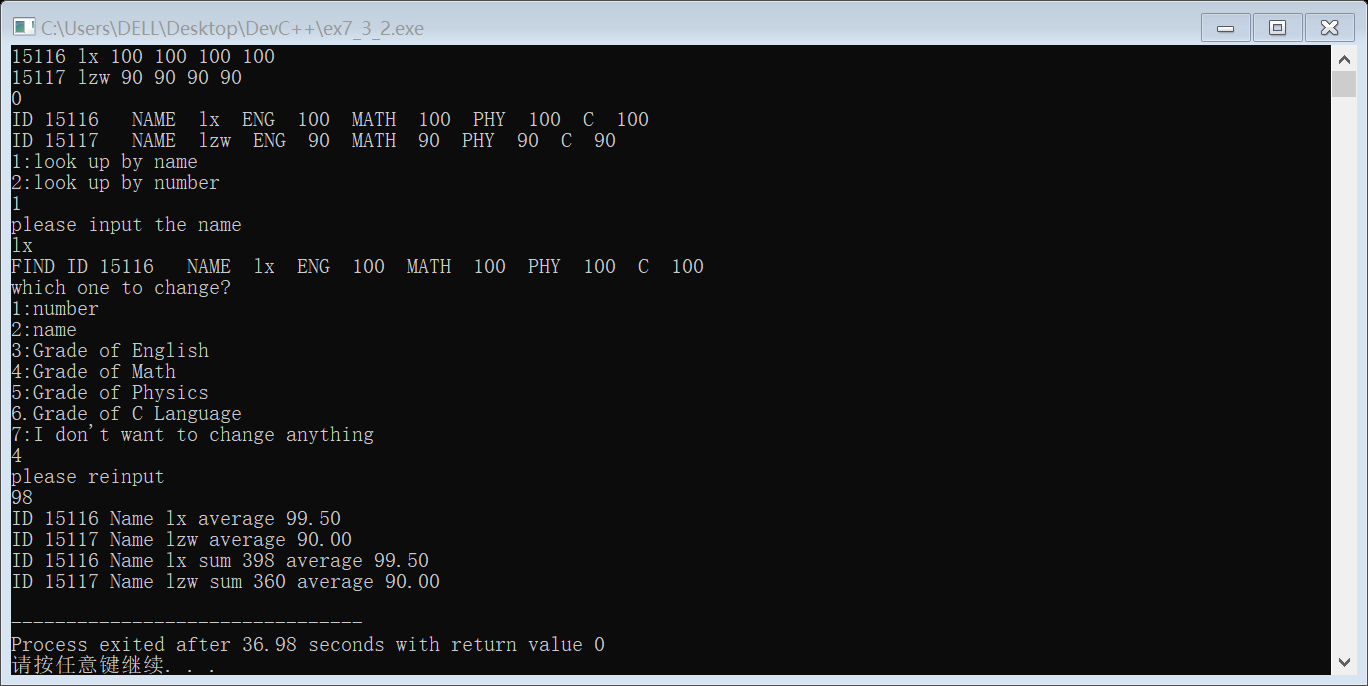
printf("ID %ld Name %s sum %d average %.2lf\n",p->num,p->s,sum,tp);

}

}

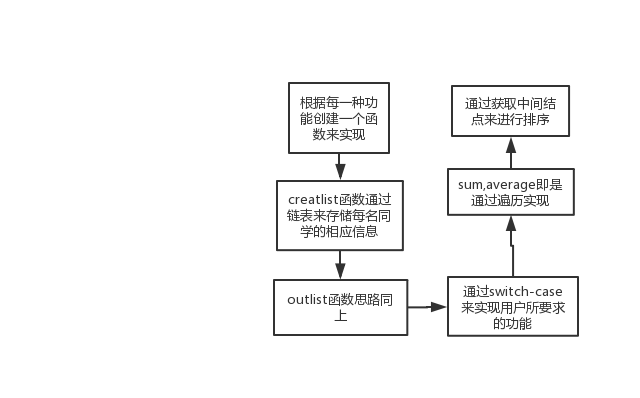
}

测试



3对程序设计第（2）题的程序，增加按照平均成绩进行升序排序的函数，写出用交换结点数据域的方法升序排序的函数，排序可用选择法或冒泡法。

解题思路



程序清单

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define null NULL

#define val tp

struct stu{

long num;

char s[10];

int c1;

int c2;

int c3;

int c4;

double tp;

int sum;

struct stu\*next;

};

void creatlist(struct stu \*\*headp);

void outputlist(struct stu \*headp);

void correctlist(struct stu \*\*headp);

//void avstu(struct stu \*headp);

//void sumstu(struct stu \*headp);

struct stu\* sortList(struct stu \*\*head);

struct stu\* merge(struct stu \*left,struct stu \*right);

struct stu\* getMidNode(struct stu\* node);

void Free\_list(struct stu \* pHead) //释放链表

{

struct stu \* p;

while(pHead != NULL)

{

p = pHead;

pHead = pHead->next;

free(p);

p = NULL;

}

}

int main(void)

{

struct stu \*head=NULL;

creatlist(&head);

outputlist(head);

correctlist(&head);

//avstu(head);

//sumstu(head);

outputlist(head);

outputlist(sortList(&head));

Free\_list(head);

return 0;

}

void creatlist(struct stu \*\*headp)

{

long n;

struct stu \*tail,\*temp;

\*headp=(struct stu \*)malloc(sizeof(struct stu));

tail=\*headp;

printf("please input the following item divided by squre\n");

printf("ID\tname\tEnglishgrade\tmathgrade\tphysicsgrade\tC language grade");

printf("\nEnd with 0\n");

while(1)

{

scanf("%ld",&n);

if(n==0) break;

tail->num=n;

scanf("%s %d %d %d %d",tail->s,&tail->c1,&tail->c2,&tail->c3,&tail->c4);

tail->sum=tail->c1+tail->c2+tail->c3+tail->c4;

tail->tp=tail->sum/4.0;

tail->next=(struct stu \*)malloc(sizeof(struct stu));

temp=tail;

tail=tail->next;

}

free(tail);

temp->next=NULL;

}

void outputlist(struct stu \*p)

{

while(p!=NULL)

{

printf("ID %ld NAME %s ENG %d MATH %d PHY %d C %d SUM %d AVERAGE %.2f\n"

,p->num,p->s,p->c1,p->c2,p->c3,p->c4,p->sum,p->tp);

p=p->next;

}

}

void correctlist(struct stu \*\*headp)

{

char c;

struct stu \*p,\*temp;

int jud=1;

p=\*headp;

temp=\*headp;

printf("1:look up by name\n");

printf("2:look up by number\n");

c=getchar();

c=getchar();

switch(c)

{

case '1':printf("please input the name\n");

char t[10];

scanf("%s",t);

while(p!=NULL){

if(!strcmp(p->s,t)){

printf("FIND ID %ld NAME %s ENG %d MATH %d PHY %d C %d\n"

,p->num,p->s,p->c1,p->c2,p->c3,p->c4);

jud=0;

}

temp=p;

p=p->next;

}

if(jud){

printf("NOT FOUND\n");

return ;

}

p=temp;

break;

case '2':printf("please input the number\n");

long tmp;

scanf("%ld",&tmp);

while(p!=NULL){

if(tmp==p->num){

printf("FIND ID %ld NAME %s ENG %d MATH %d PHY %d C %d\n"

,p->num,p->s,p->c1,p->c2,p->c3,p->c4);

jud=0;

}

temp=p;

p=p->next;

}

if(jud){

printf("NOT FOUND\n");

return ;

}

p=temp;

break;

}

printf("which one to change?\n");

printf("1:number\n");

printf("2:name\n");

printf("3:Grade of English\n");

printf("4:Grade of Math\n");

printf("5:Grade of Physics\n");

printf("6.Grade of C Language\n");

printf("7:I don't want to change anything\n");

c=getchar();

c=getchar();

printf("please re-input\n");

switch(c){

case '1':scanf("%ld",&p->num);break;

case '2':scanf("%s",p->s);break;

case '3':scanf("%d",&p->c1);break;

case '4':scanf("%d",&p->c2);break;

case '5':scanf("%d",&p->c3);break;

case '6':scanf("%d",&p->c4);

default: return ;

}

}

/\*void avstu(struct stu \*p)

{

for(;p!=NULL;p=p->next)

{

printf("ID %ld Name %s average %.2lf\n",p->num,p->s,p->tp);

}

}

\*/

/\*void sumstu(struct stu \*p)

{

for(;p!=NULL;p=p->next)

{

int sum;

sum=p->c1+p->c2+p->c3+p->c4;

printf("ID %ld Name %s sum %d average %.2lf\n",p->num,p->s,sum,p->tp);

}

}

\*/

struct stu\* sortList(struct stu \*\*head) {

printf("\n\n");

if(\*head == null || (\*head)->next == null) return \*head;

struct stu\* mid = getMidNode(\*head);// 获取中间节点

struct stu\* left = \*head;

struct stu\* right = mid->next;

mid->next = null; // 这一点很重要，不然可能出现循环链表

\*head=merge(sortList(&left),sortList(&right));

return \*head;

}

// 合并链表，代码比较长，但是很好理解，头节点需要单独考虑一下

struct stu\* merge(struct stu \*left,struct stu \*right){

if(left == null) return right;

if(right == null) return left;

struct stu\* head = null, \*p = null; // head 头节点，p用于遍历操作

while(left != null && right != null){

if(head == null){

if(left->val < right->val){

head = left;

left = left->next;

}else{

head = right;

right = right->next;

}

p = head;

}else{

if(left->val < right->val){

p->next = left;

left = left->next;

}else{

p->next = right;

right = right->next;

}

p = p->next;

}

}

// 对剩下的节点进行merge

if(left != null) p->next = left;

else p->next = right;

return head;

}

// 使用快慢指针快速找到中间节点

struct stu\* getMidNode(struct stu\* node){

if(node == null || node->next == null) return node;

struct stu\* low = node;

struct stu\* fast = node;

while(fast->next != null && fast->next->next != null){

low = low->next;

fast = fast->next->next;

}

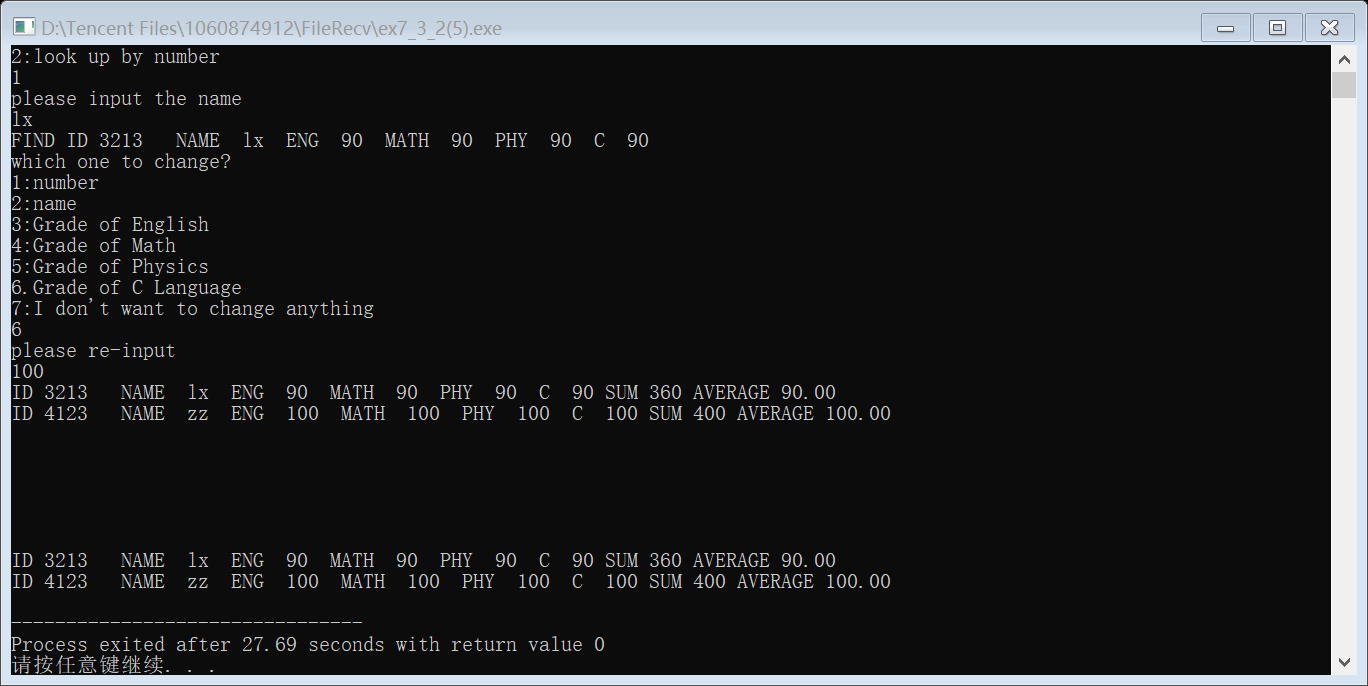
return low;

}

测试

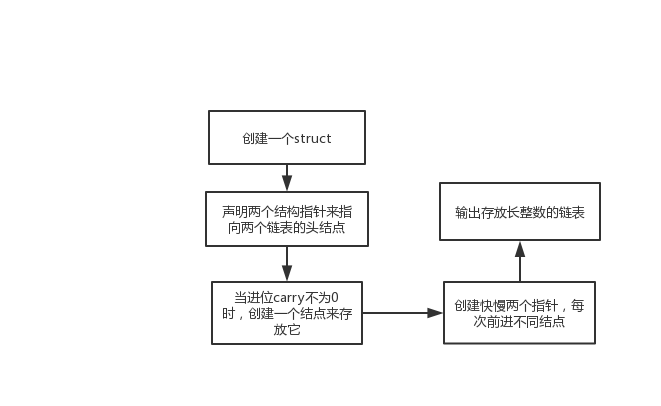
表2-3 编程题5的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例一 | 见程序清单 |  |



4.输入两个十进制超大正整数（数的十进制位数非常大且在程序运行前无法预估），计算他们的和并输出。要求用单向链表来存放超大正整数，链表的每个结点上存放一位十进制数字。

解题思路



程序清单

#include <stdio.h>

#include <malloc.h>

#include <string.h>

#define MAX\_SIZE 50

typedef struct node {

int data;

struct node \*next;

} NodeType;

static void create\_list(NodeType \*&h, char a[], int n) {

NodeType \*p, \*r;

int i = 0;

h = (NodeType \*)malloc(sizeof(NodeType));

r = h;

while(i<n) {

p = (NodeType \*)malloc(sizeof(NodeType));

p->data = a[n - i - 1] - '0';

r->next = p;

r = p;

i++;

}

r->next = NULL;

}

static void display\_list(NodeType \*h) {

NodeType \*p = h->next;

while(p != NULL) {

printf("%d", p->data);

p = p->next;

}

printf("\n");

}

static void add\_list(NodeType \*h1, NodeType \*h2, NodeType \*&h) { // 指针的引用

NodeType \*p1 = h1->next; // p1指向单链表h1中的第一个数据结点

NodeType \*p2 = h2->next; // p2指向单链表h2中的第一个数据结点

NodeType \*p, \*r;

int carry = 0;

// 创建头结点

h = (NodeType \*)malloc(sizeof(NodeType));

// r指向新创建头结点

r = h;

while(p1 != NULL && p2 != NULL) {

// 创建新结点p

p = (NodeType \*)malloc(sizeof(NodeType));

p->data = (p1->data + p2->data + carry) % 10; // 求余

// 将新结点p插入到r指向的头结点之后

r->next = p;

// r后移一个结点

r = p;

carry = (p1->data + p2->data + carry) / 10; // 求商

// p1和p2指向下一个结点

p1 = p1->next;

p2 = p2->next;

}

if(p1 == NULL)

p1 = p2;

while(p1 != NULL) {

p = (NodeType \*)malloc(sizeof(NodeType));

p->data = (p1->data + carry) % 10; // 数据域

// 将新结点p插入到r指向的头结点之后

r->next = p;

// r后移一个结点

r = p;

carry = (p1->data + carry) / 10;

p1 = p1->next;

}

// 最后carry不为0时,创建一个结点存放它

if(carry > 0) {

p = (NodeType \*)malloc(sizeof(NodeType));

p->data = carry;

// 将新结点p插入到r指向的头结点之后

r->next = p;

// r后移一个结点

r = p;

}

r->next = NULL;

}

/\*------------------------逆置整数单链表h--------------------------\*/

static void reverse\_list(NodeType \*&h) {

NodeType \*p = h->next, \*q;

h->next = NULL;

while(p != NULL) {

q = p->next;

p->next = h->next;

h->next = p;

p = q;

}

}

static int mid\_list(NodeType \*h) {

NodeType \*slow = h; // 定义慢指针

NodeType \*quick = h; // 定义快指针

while(quick != NULL && quick->next != NULL) {

slow = slow->next; // 慢指针slow前进一个结点

quick = quick->next->next; // 快指针quick前进两个结点

}

return slow->data;

}

int main(int argc, char \*argv[]) {

NodeType \*h1, \*h2, \*h;

char s[MAX\_SIZE], t[MAX\_SIZE];

printf("输入整数1: ");

scanf("%s", s);

printf("输入整数2: ");

scanf("%s", t);

create\_list(h1, s, strlen(s));

create\_list(h2, t, strlen(t));

add\_list(h1, h2, h);

reverse\_list(h);

printf("结果为:\n");

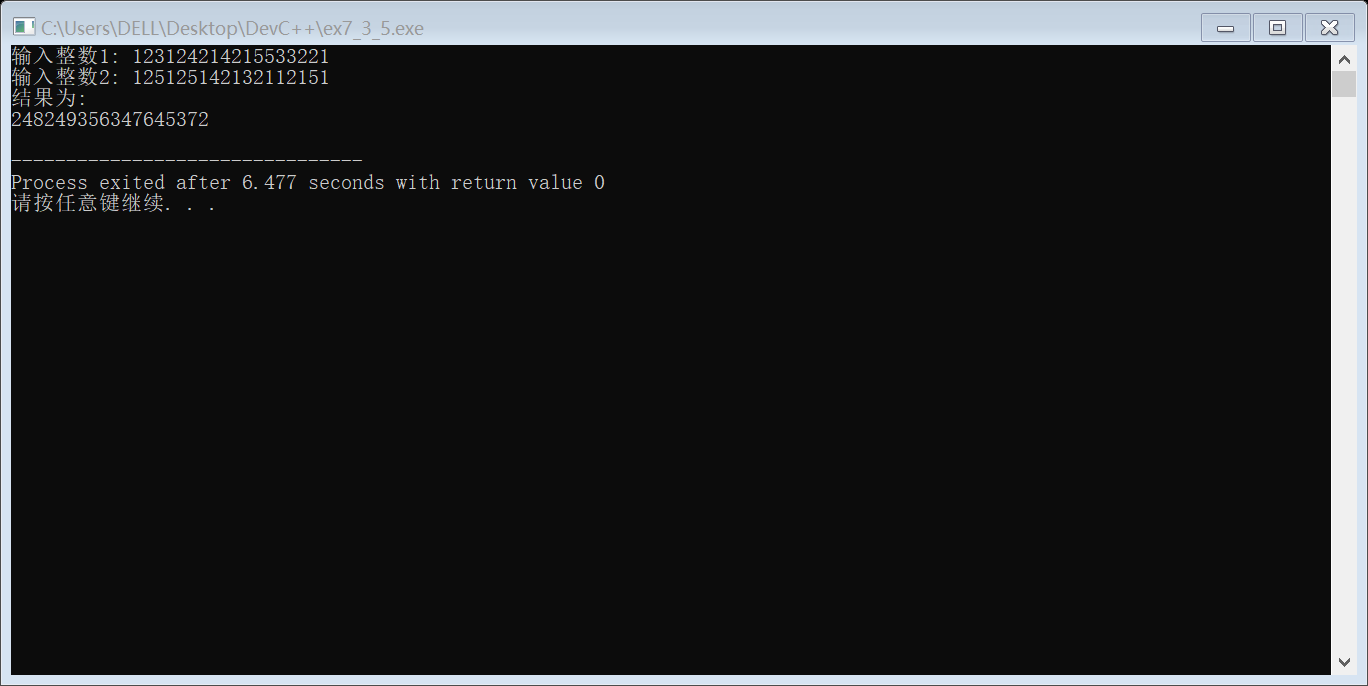
display\_list(h);

return 0;

}测试

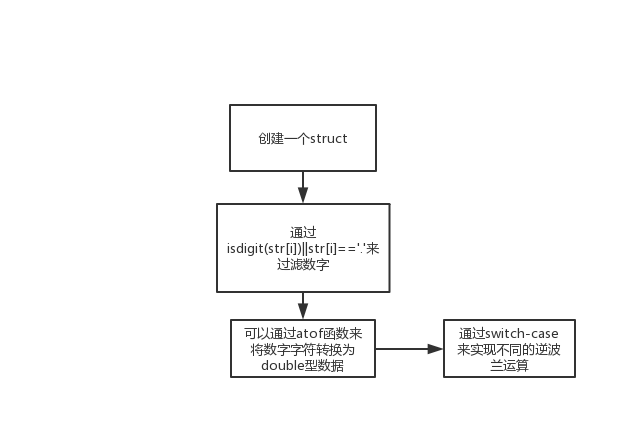
表2-4 编程题6的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例一 | 见下图 | 见下图 |



5.利用值栈对逆波兰表达式进行求值。逆波兰表达式从键盘输入，其中的运算符仅包含加，减，乘，除4种运算，表达式中的数都是十进制数，用换行符结束输入。由于逆波兰表达式的长度不限，所以值栈要用后进先出链表实现。

解题思路



程序清单

#include<stdio.h>

#include<stdlib.h>

#include<assert.h>

#include<ctype.h>

#define INITSIZE 20

#define INCREMENT 10

#define MAXBUFFER 10

#define LEN sizeof(Elemtype)

/\*栈的动态分配顺序存储结构\*/

typedef double Elemtype;

typedef struct{

Elemtype \*base;

Elemtype \*top;

int StackSize;

}SqStack;

void InitStack(SqStack \*S)

{

S->base=(Elemtype\*)malloc(LEN\*INITSIZE);

assert(S->base != NULL);

S->top=S->base;

S->StackSize=INITSIZE;

}

void PushStack(SqStack \*S,Elemtype e)

{

if(S->top - S->base >= S->StackSize)

{

S->base=(Elemtype\*)realloc(S->base,(S->StackSize+INCREMENT)\*LEN);

assert(S->base !=NULL);

S->top=S->base+S->StackSize;

S->StackSize+=INCREMENT;

}

\*S->top =e;

S->top++;

}

void PopStack(SqStack \*S,Elemtype \*e)

{

\*e=\*--S->top;

}

void CalFunction(SqStack \*S,char str[])

{

Elemtype number,e,d;

char arr[MAXBUFFER];

int i=0,j=0;

InitStack(S);

while(str[i]!='\0')

{

while(isdigit(str[i])||str[i]=='.') //过滤数字

{

arr[j++]=str[i++];

arr[j]='\0';

if( j >= MAXBUFFER )

{

printf("输入单个数据过大！\n");

return ;

}

if(str[i]==' ')

{

number=atof(arr); //利用atof函数将数字字符转化为double型数据

PushStack(S,number); //将转换的数进行压栈

j=0;

break;

}

}

switch(str[i])

{

case '+':

PopStack(S,&e);

PopStack(S,&d);

PushStack(S,d+e);

break;

case '-':

PopStack(S,&e);

PopStack(S,&d);

PushStack(S,d-e);

break;

case '\*':

PopStack(S,&e);

PopStack(S,&d);

PushStack(S,d\*e);

break;

case '/':

PopStack(S,&e);

PopStack(S,&d);

if(e == 0)

{

printf("输入出错，分母为零！\n");

return ;

}

PushStack(S,d/e);

break;

}

i++;

}

PopStack(S,&e);

printf("计算结果为：%.2f",e);

}

int main()

{

char str[100];

SqStack S;

printf("请按逆波兰表达式输入数据,每个数据之间用空格隔开:\n");

gets(str);

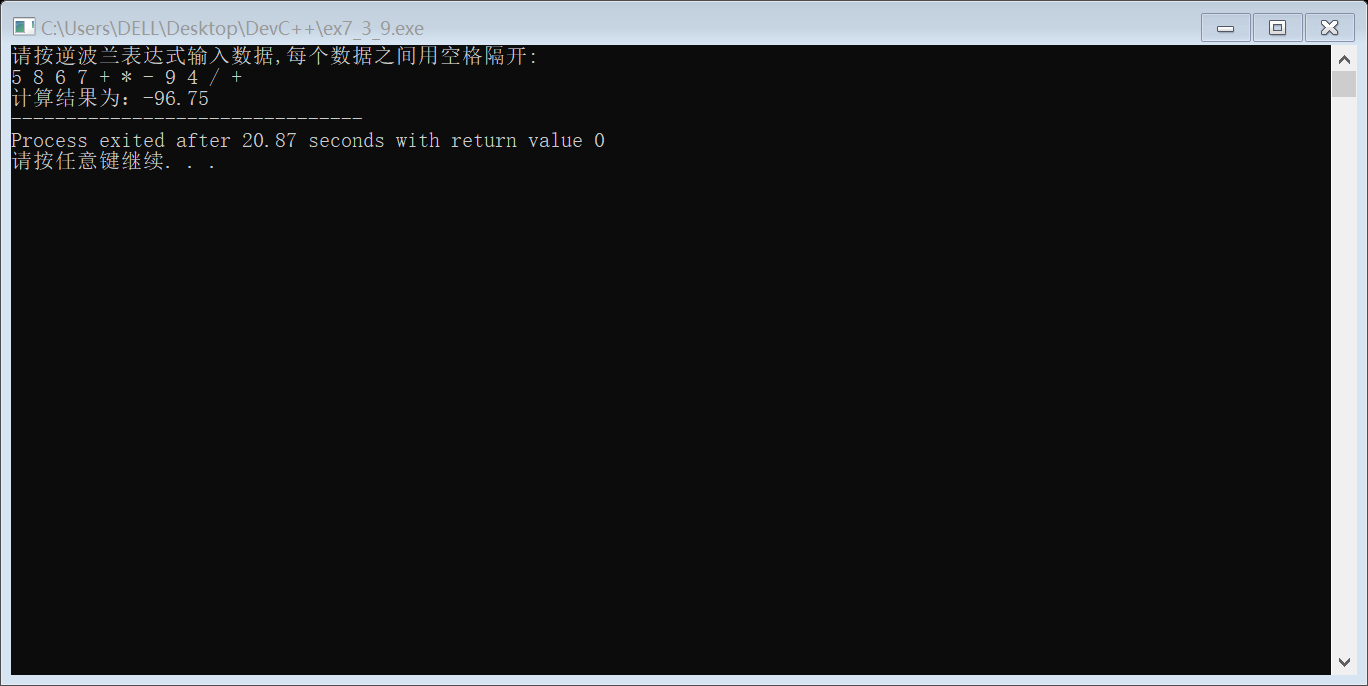
CalFunction(&S,str);

return 0;

}测试

表2-5 编程题9的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 |
| 用例一 | 见下图 | 见下图 |



## 3.4小结

结构和联合作为C语言的另外两种构造类型，在处理一些有关联的数据的时候，比二维数组乃至多维数组更加的方便，也更加地便于后续的修改，就做过的编程题目来说，结构和联合除以比如一个班级的学生信息有显著的优势。而通过结构，可以建立链表这一工具，包括单向链表，双向链表等等，链表的使用配上指针能够让各个数据之间的联系更为紧密。

通过结构和联合，我知道了可以用另外的方法来实现之前的输入日期判断天数这一功能，或者实现大数的相加和相乘等等，链表在这些方面都起到了极大的作用。

一旦开始用链表解决一些问题，代码的行数和变量的数目都增加了许多，但其实在脑中有了思路之后写起程序来就并不困难，至此掌握了一些C语言的基本工具和基本算法之后，我觉得参与一些项目的编写也就并不困难了。

这次实验中的逆波兰表达式求值也确实让我费了点脑筋，它要求我不仅仅单单的读取相应的运算符号，在一些很长的算式，也能计算中正确的结果，确实对我的能力提出了极大的挑战。

# 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

........