

**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：**

**计算机科学与技术学院**

目 录

[**1流程控制实验 1**](#_Toc404837924)

[1.1程序改错 1](#_Toc404837925)

[1.2程序修改替换 2](#_Toc404837926)

[1.3程序设计 3](#_Toc404837926)

[1.4小结 18](#_Toc404837928)

[**2数组程序设计实验 19**](#_Toc404837934)

[2.1程序改错与跟踪调试 19](#_Toc404837935)

[2.2程序完善与修改替换 22](#_Toc404837936)

[2.3程序设计 27](#_Toc404837926)

[2.4小结 52](#_Toc404837937)

[**3结构与联合实验 54**](#_Toc404837938)

[3.1表达式求值的程序验证 54](#_Toc404837939)

[3.2源程序修改替换 56](#_Toc404837940)

[3.3程序设计 59](#_Toc404837926)

[3.4小结 95](#_Toc404837942)

[**参考文献 96**](#_Toc404837957)

# 

# 1 流程控制实验

## 程序改错

/\*合数判断器\*/

#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,以ctrl+Z结束\n");

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

{

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

{

if(！x%i) ※※此处应改为if（！（x%i））或改为if(x%i==0)

{

flag=1;

break;

}

}

if(flag=1) ※※此处应改为flag==1,一个“=”表示赋值

printf("%d是合数\n",x);

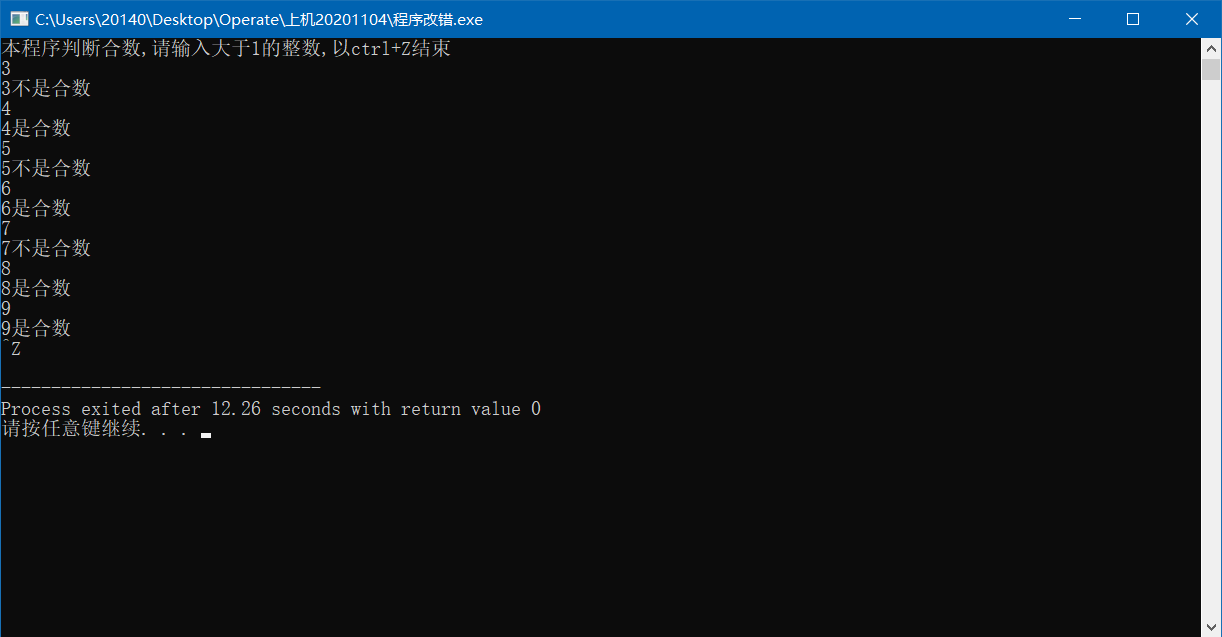
else

printf("%d不是合数\n",x);

※※此处应添加语句flag=0，否则无论是否是合数flag的值始终是1

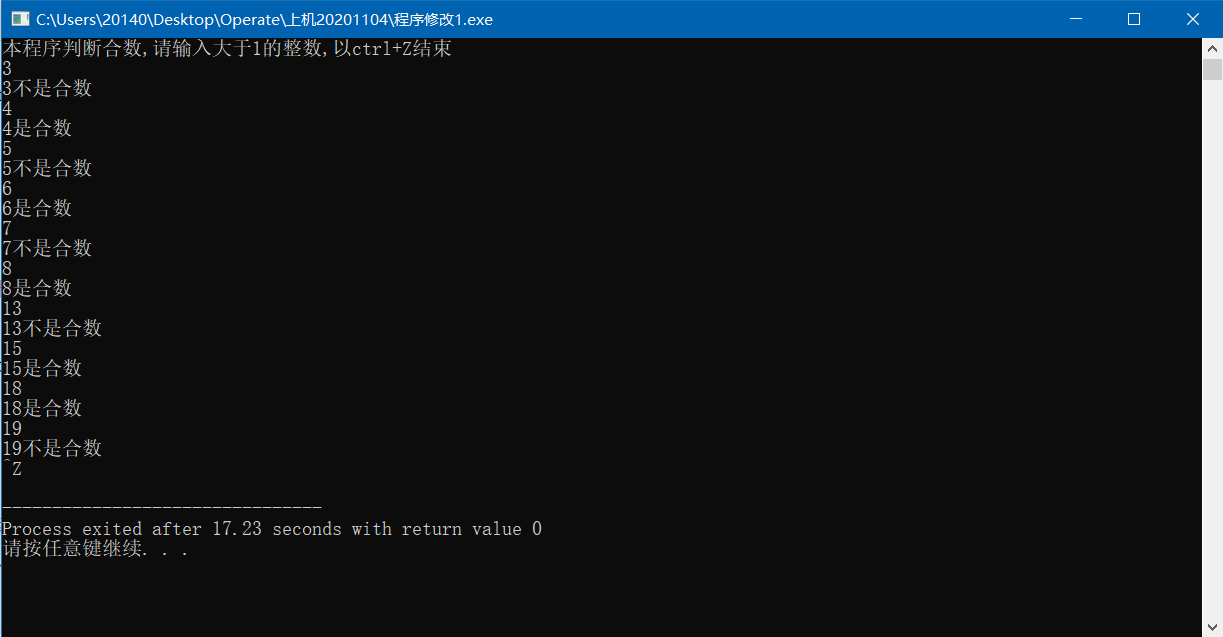
}

return 0;

}

## 1.2 程序修改替换

（1）修改方案：将原实验程序中if语句的break改为k=i;保证了if语句执行后退出循环。



（2）修改方案：将for循环改为如下代码：

int i=2，k;

k=x>>1;

do{

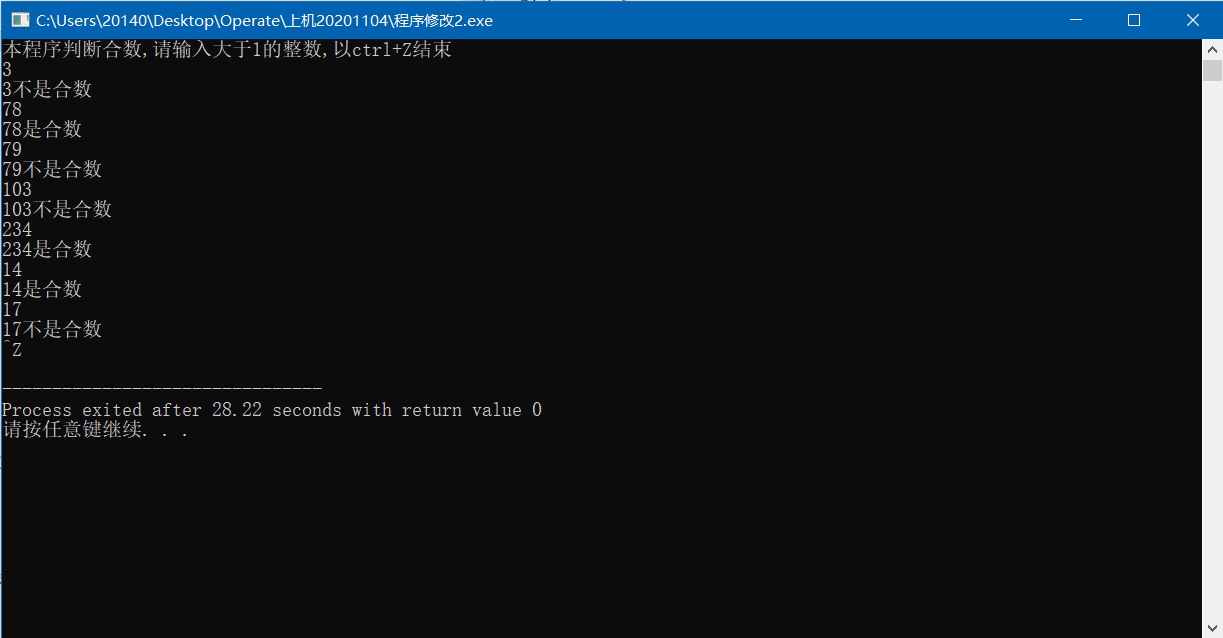
if(！（x%i）){

flag=1;

break;}

i++;

}while(i<=k);



（3）修改方案：定义两个函数，一个函数判断为合数，另一个判断是纯粹合数，并在main函数中调用判断纯粹合数的函数，输出所有的3位纯粹合数。

程序代码如下：

#include<stdio.h>

int NotPrime(int x) //判断是合数

{

int i,k,flag=0;

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

if(x%i==0){

flag=1;

break;

}

}

return flag;

}

//判断是纯粹合数

int pure(int x){

int flag=0,count=0;

for(int j=0;j<3;++j) {

if(NotPrime(x))

count++;

x/=10;

}

if(count==3)

flag=1;

return flag;

}

int main(void){

int num=0;

printf("纯粹合数有\n");

for(int x=100;x<=999;++x){

if(pure(x)){

printf("%d\t",x);

num++;

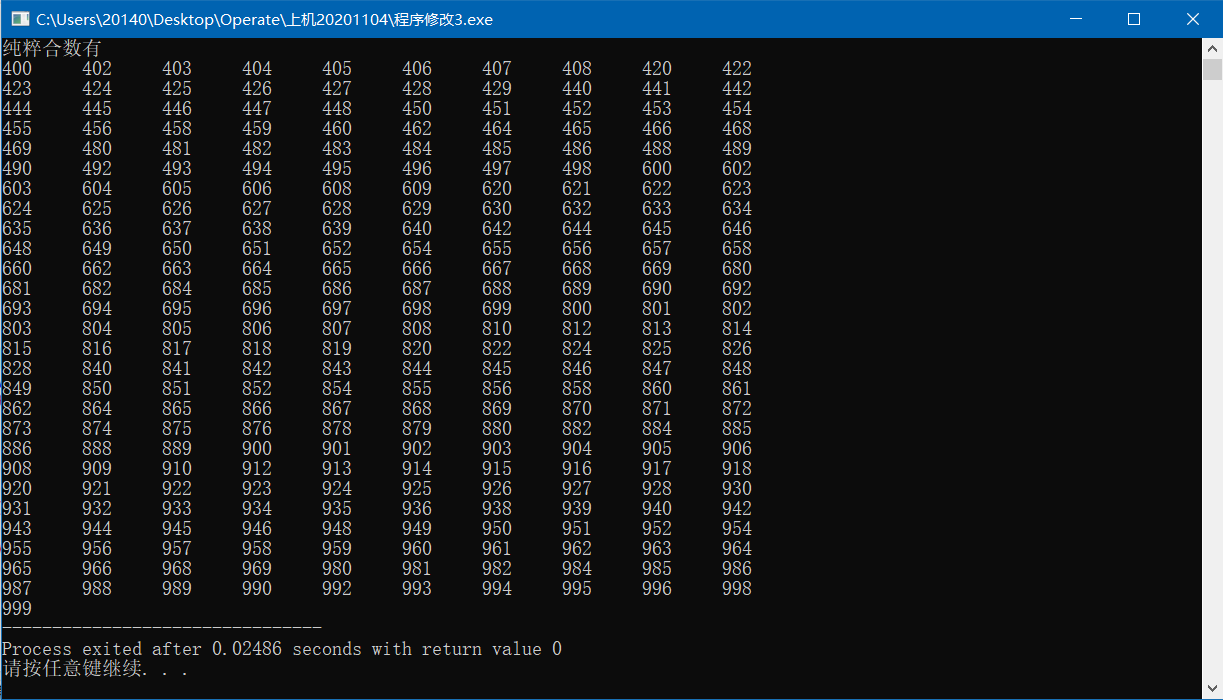
if(num%10==0)

printf("\n");

}

}

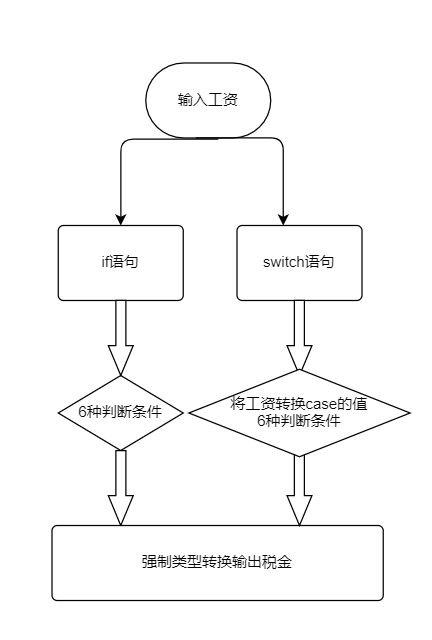
return 0; }



## 1.3 程序设计

1. 计算税金

1）解题思路：



2）程序源代码 ：

**①利用switch语句：**

#include<stdio.h>

int main(void){

int wage,tax,flag=1;

while(flag){

printf("\n请输入工资：\n") ;//提示输入工资

scanf("%d",&wage);

tax=(int)((wage)/1000);// 用于计算不同的工资范围

switch (tax){

case 0:printf("您不用交税金！");break;

case 1:printf("您应交税金：%.3f",(float)wage\*0.05);break;

case 2:printf("您应交税金：%.3f",(float)wage\*0.1);break;

case 3:printf("您应交税金：%.3f",(float)wage\*0.15);break;

case 4: printf("您应交税金：%.3f",(float)wage\*0.2);break;

default:printf("您应交税金：%.3f",(float)wage\*0.25);break;

} //强制类型转换，输出较为精确的浮点数

printf("\n是否需要继续计算？(0 or 1)\n\n");

scanf("%d",&flag);//用于决定是否继续循环的语句

}

return 0;

}

**② 利用if语句：**

#include<stdio.h>

int main(void){

int x,y,flag=1;

while(flag){

printf("\n请输入工资：\n") ;//提示输入工资

scanf("%d",&x);

if(x<0)

printf("请输入正整数！");

if(x<1000)

printf("您不用交税金！");

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

printf("您应交税金：%.3f",(float)x\*0.05);

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

printf("您应交税金：%.3f",(float)x\*0.1);

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

printf("您应交税金：%.3f",(float)x\*0.15);

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

printf("您应交税金：%.3f",(float)x\*0.2);

if(x>5000)

printf("您应交税金：%.3f",(float)x\*0.25);

//强制类型转换，输出较为精确的浮点数

printf("\n是否需要继续计算？(0 or 1)\n\n");

scanf("%d",&flag); //用于决定是否继续循环的语句

}

return 0;

}

3）测试：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | [0,1000) | [10000,2000) | [2000,3000) | [3000,4000) | [4000,5000) | [5000,+∞) |
| 输入值 | 999 | 1100 | 2400 | 3456 | 4312 | 5679 |
| 结果 | 不用交税金 | 55.000 | 240.000 | 518.400 | 862.400 | 1419.750 |

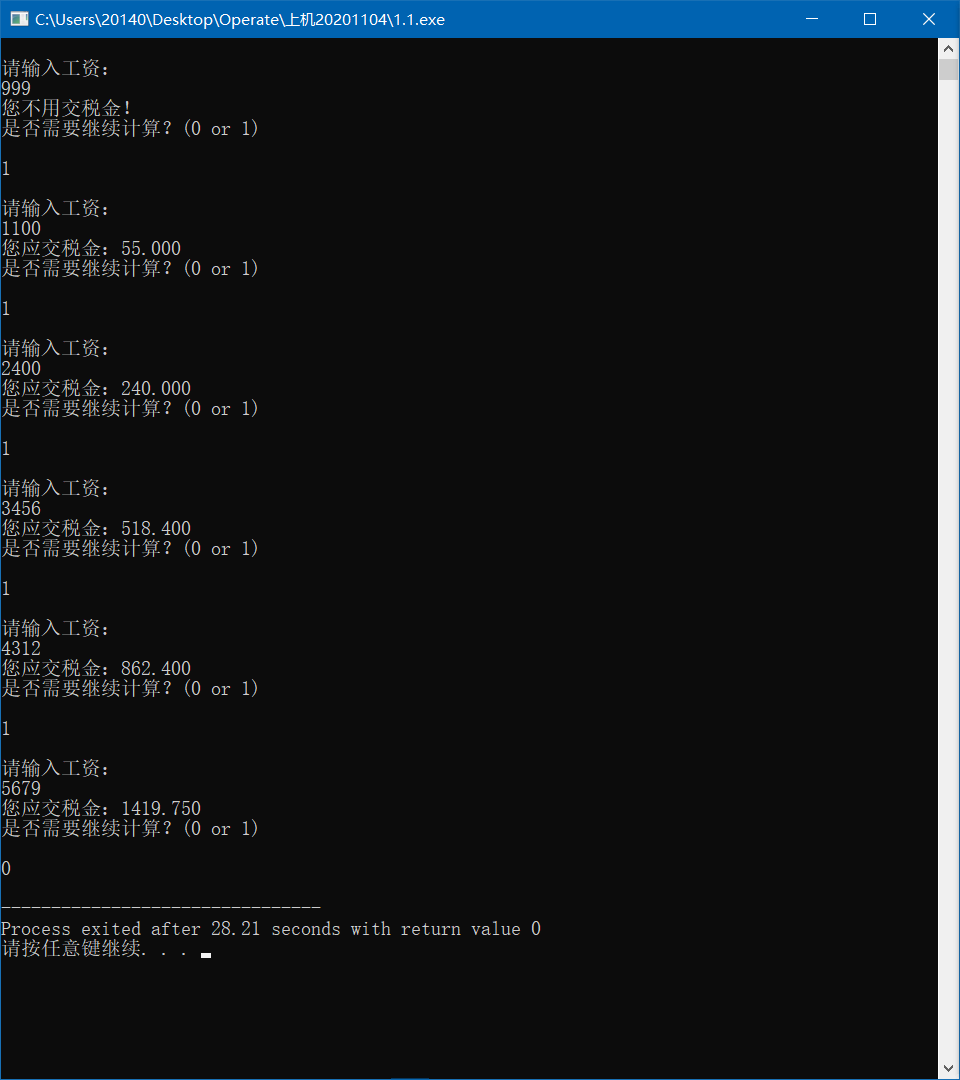
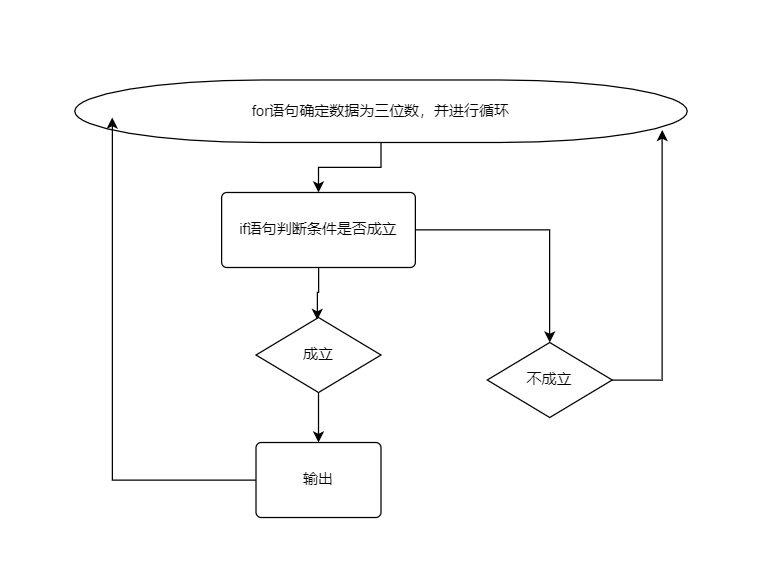


图1-1 程序设计题1的运行结果

2.输出满足条件的三位数

1）解题思路：



2）程序源代码 ：

#include<stdio.h>

int main(void){

for(int i=100;i<=999;i++){ //for循环语句确定是三位数

int t;

t=i\*i;

if(t%1000==i)//if判断该数平方末三位是否是该数本身

printf("%d\n",i);//若结果为真，输出该值。

}

return 0;

}

3）测试：

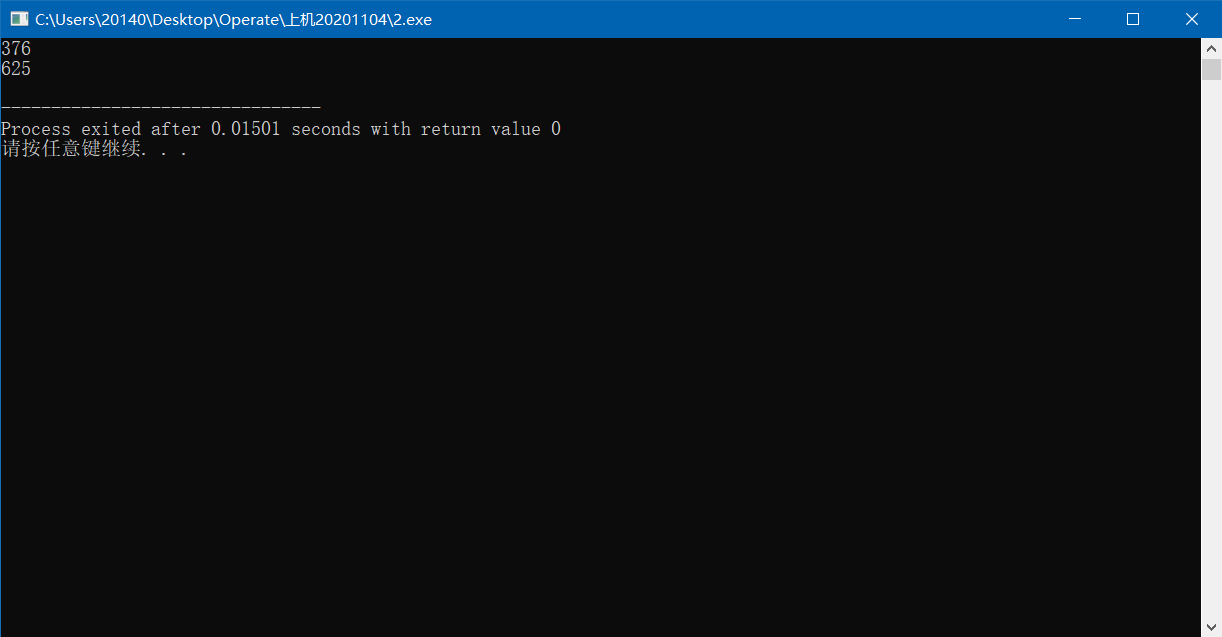
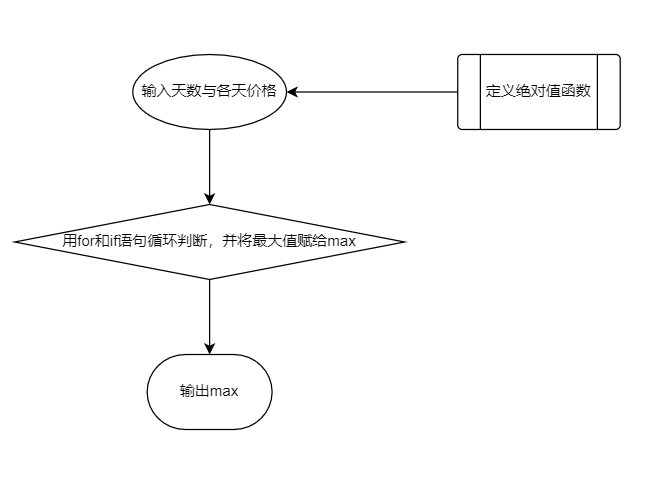


图1-2程序设计题2的运行结果

3.股票收盘价格波动值

1）解题思路：



2）程序源代码 ：

#include<stdio.h>

double abs(double a) {

if(a<0)

a=-a;

return a;

}//定义绝对值函数

int main(void){

double a,b,max;

int day;

printf("请输入天数:\n");

scanf("%d",&day);

printf("请输入这些天的收盘价格:\n");

scanf("%lf",&b);//提示输入收盘天数与每天收盘价格

for(int i=1;i<day;++i){

a=b;

scanf("%lf",&b);

if(abs(a-b)>max)//if语句将每次的最大波动值赋给max

max=abs(a-b);

}

printf("%.2lf\n",max);//输出最大波动值

return 0;

}

3)测试：

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出 |
| 1 | 1 5 3 9 | 6.00 |
| 2 | 1.23 1.78 2.57 1.59 | 0.98 |
| 3 | 200 267.45 345.89 | 78.44 |

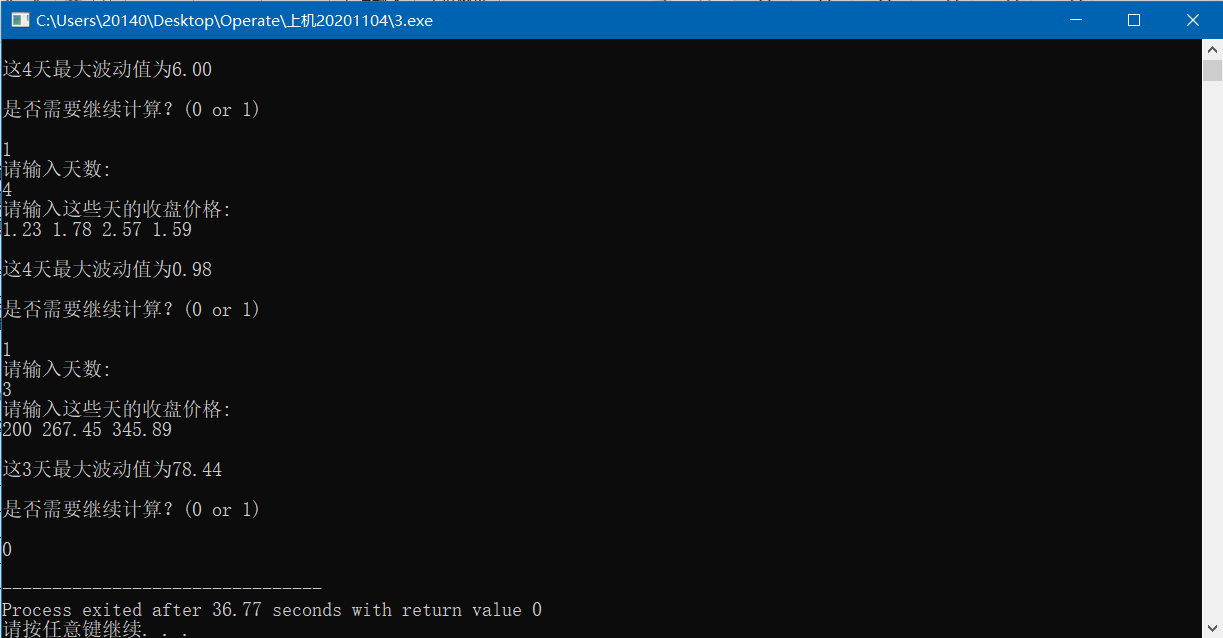
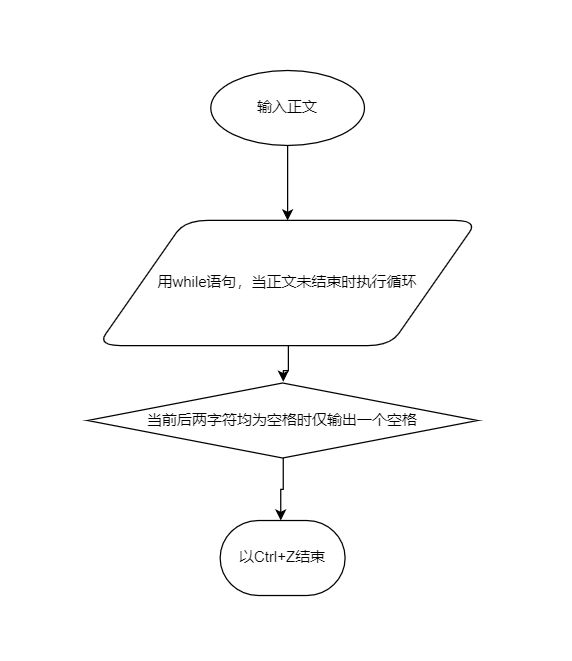


图1-3程序设计题3的运行结果

4.将正文的多个空格用一个代替

1）解题思路：



2）程序源代码 ：

#include<stdio.h>

int main(void) {

char a,b;

printf("请输入一段正文，并以Ctrl+Z结束:\n") ;

while((b=getchar())!=EOF){

if(!(a==32&&b==32))//如果前后两个字符均为空格，则只输出一个

putchar(b);

a=b;//将后一个字符赋给前一个字符

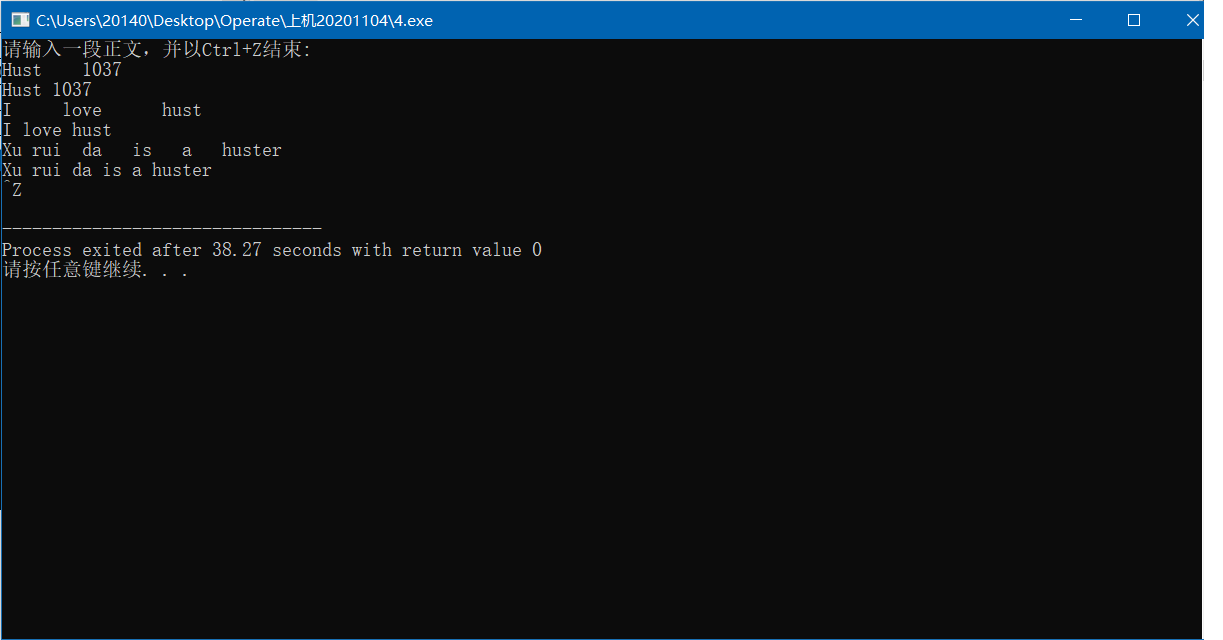
}

return 0;

}

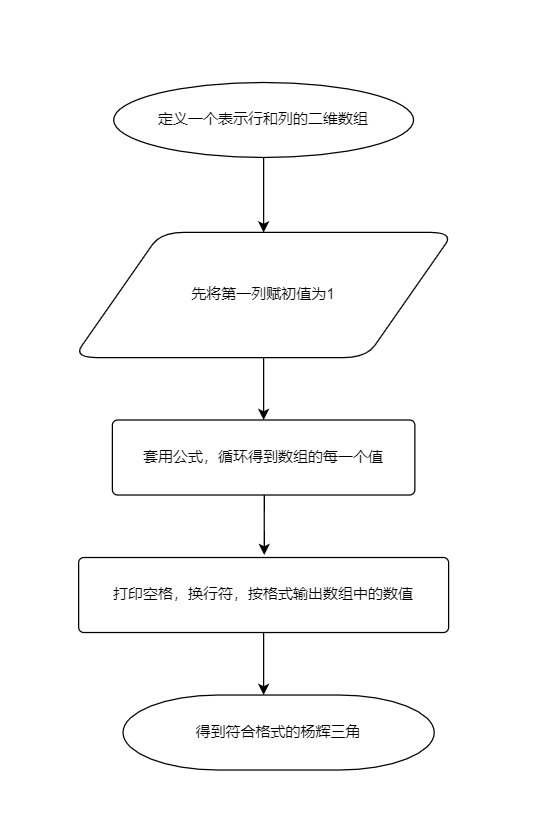
3）测试：

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出 |
| 1 | Hust 1037 | Hust 1037 |
| 2 | I love hust | I love hust |
| 3 | Xu rui da is a huster | Xu rui da is a huster |

图1-4程序设计题4的运行结果

5.输出10行杨辉三角

1）解题思路：



2）程序源代码 ：

#include <stdio.h>

int main(){

int i,j,a[100][100];//定义一个二维数组，代表行和列

for(i=0;i<10;++i)

a[i][0]=1;//通过遍历，将第一列的值赋为1

for(i=0;i<10;++i)

{

for(j=1;j<10;++j)

a[i][j]=(a[i][j-1])\*(i-j+1)/j;//应用公式将值存入二维数组中

}

for (i=0;i<10;++i)

{

for(j=i;j<10;++j)

printf(" ");//打印空格，形成三角格式

for(j=0;j<10;++j)

{

if(a[i][j]!=0)

printf("%4d",a[i][j]);//按一个数字占四个字符的域宽输出

}

printf("\n");//确保换行输出

}

return 0;

}

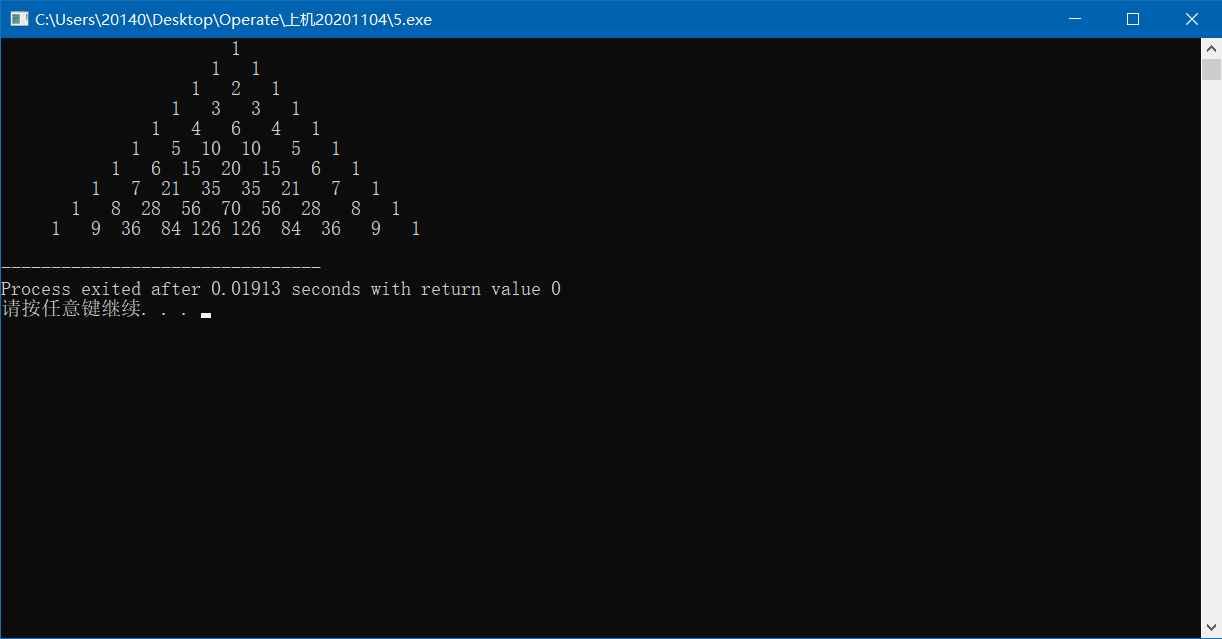
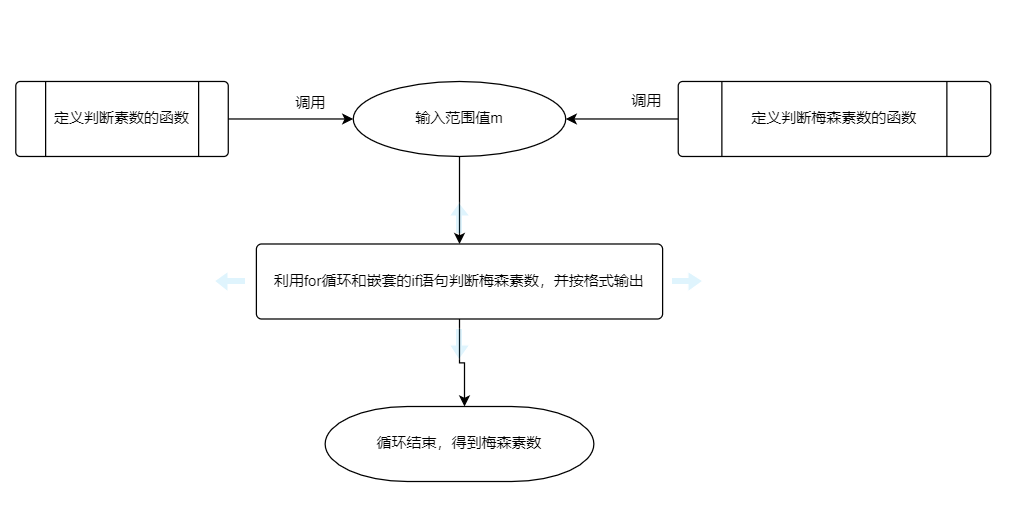
3）输出结果：

图1-5程序设计题5的运行结果

6.输出梅森素数

1）解题思路：



2）程序源代码 ：

#include<stdio.h>

#include<math.h>

long isprime(long n)//定义一个函数判断是素数

{

int flag=1;

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

{

if(n%i==0)

{

fl

ag=0;

break;

}

}

return flag;

}

long isMasonNumber(long num)//定义一个函数判断是梅森数

{

int flag=0;

if (((num+1) & (num)) == 0)

flag=1;

return flag;

}

int count(long x)//定义一个求某数是2的几次方的函数

{

int count=0;

int total=1;

while(total<x)

{

total = total \* 2;

++count;

}

if (total == x)

return count;

}

int main(void)

{

long i=2,m;

printf("请输入一个长度大于三的长整数：\n");

scanf("%d",&m);

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

{

if(isMasonNumber(i))

//if循环的嵌套，先判断是梅森数，再判断是否为梅森素数

{

if(isprime(i))

printf("M(%d)=%ld\n",count(i+1),i);//按示例输出

}

}

return 0;

}

3）测试：

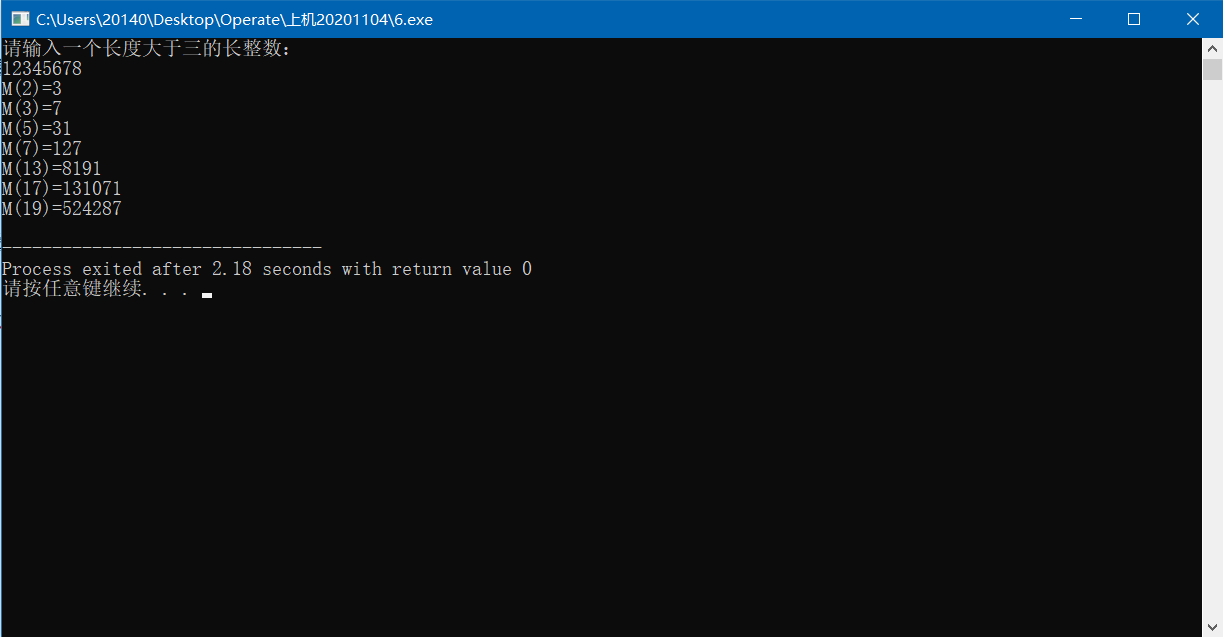
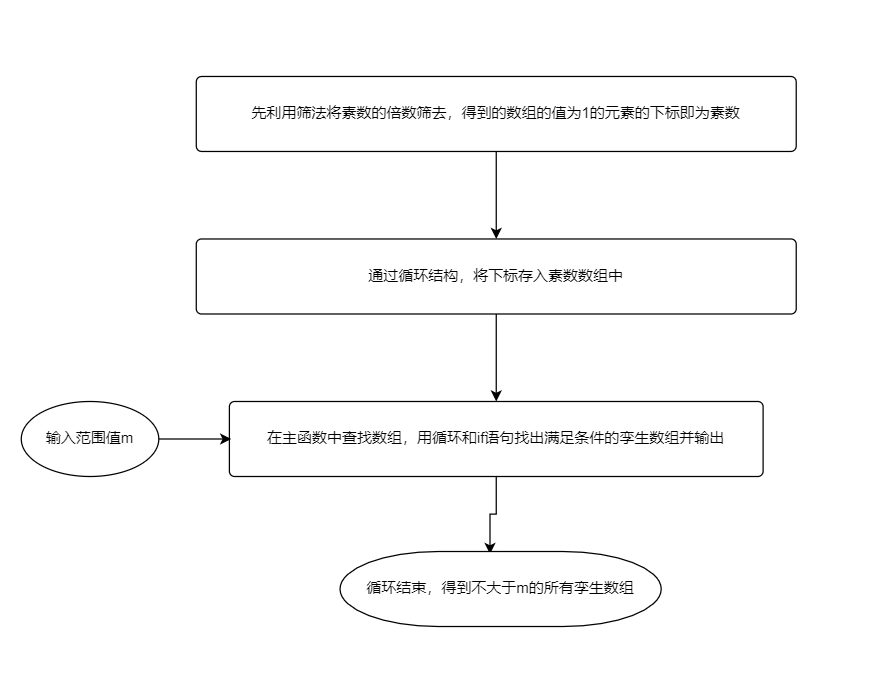
输入一个较大的m可得到结果如图

图1-6程序设计题6的运行结果

7.筛法构造素数表

1）解题思路：



2）程序源代码 ：

#include<stdio.h>

#define scale 100000//可改变素数表输出范围

int main()

{ long long i, j, a[scale],prime[scale];

for (a[0]=a[1]=0,i=2;i<scale;i++)

a[i]=1;

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

{

if(a[i]){

for (j=i\*2;j<scale;j+=i) a[j]=0;//筛掉素数的倍数

} }

long long n=0;

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

{

if(a[i])

{ prime[n]=i; ++n;//将表中的下标（素数）存入素数表中

} }

long long m; scanf("%lld",&m);

long long q=0;

for (;prime[q+1]<m;q++)//查找素数表输出孪生素数

{

if(prime[q+1]-prime[q]==2)

printf("(%lld,%lld)\n",prime[q],prime[q+1]);

}

return 0; }

3）测试输出：

分别输入m的值为50,150,500可得如图结果：

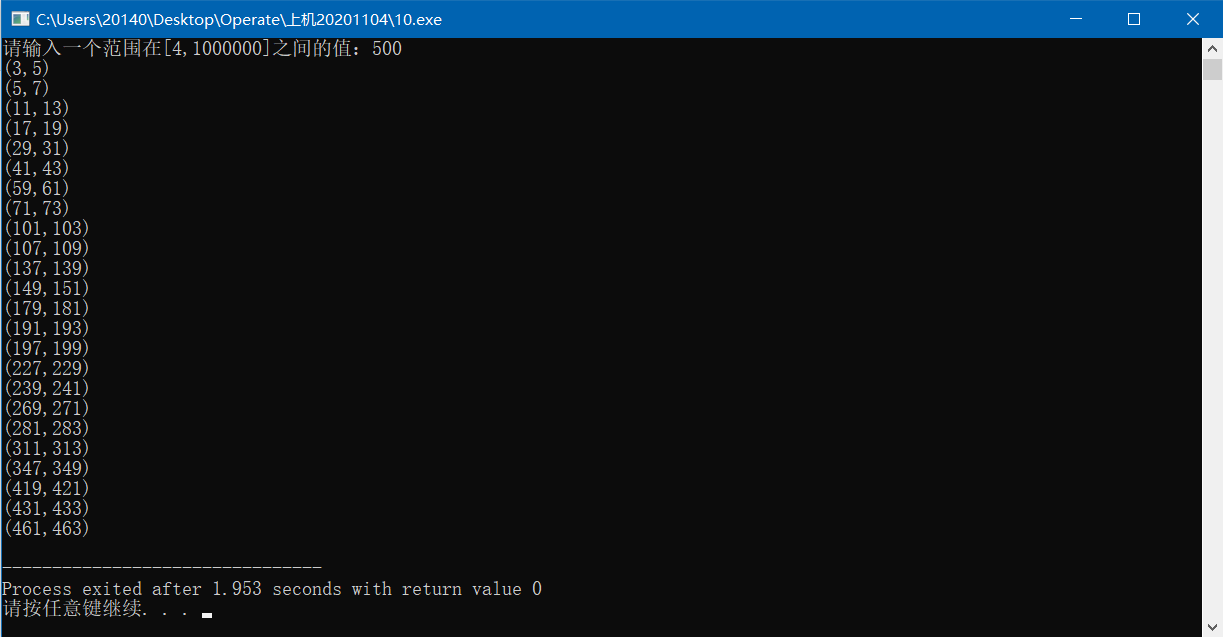
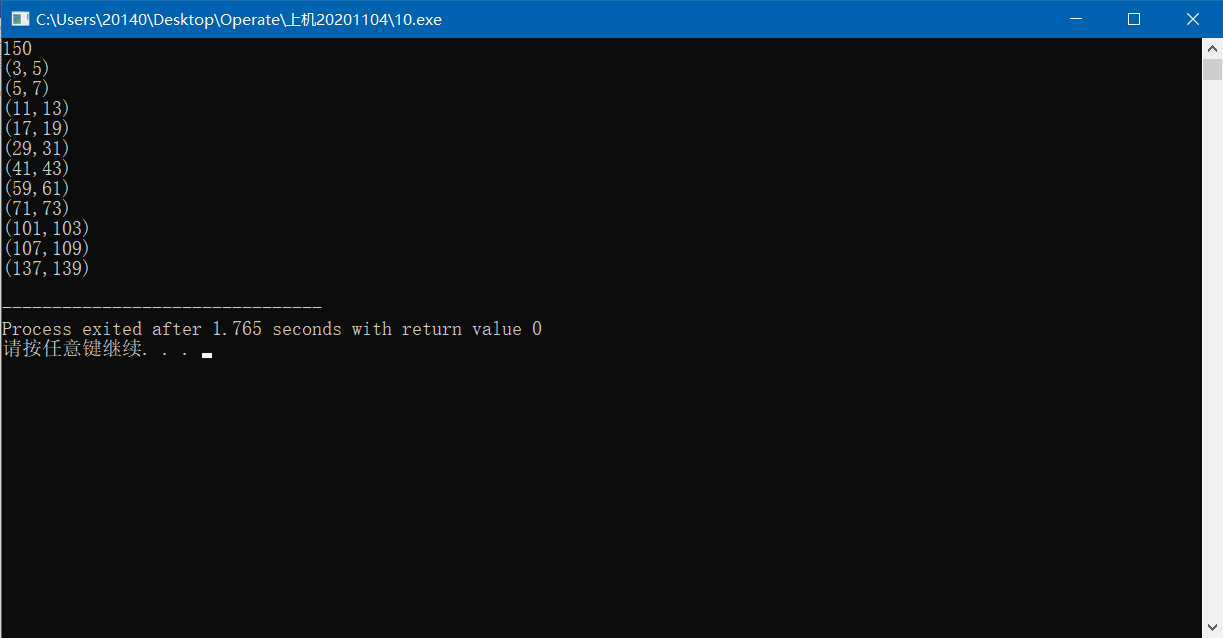
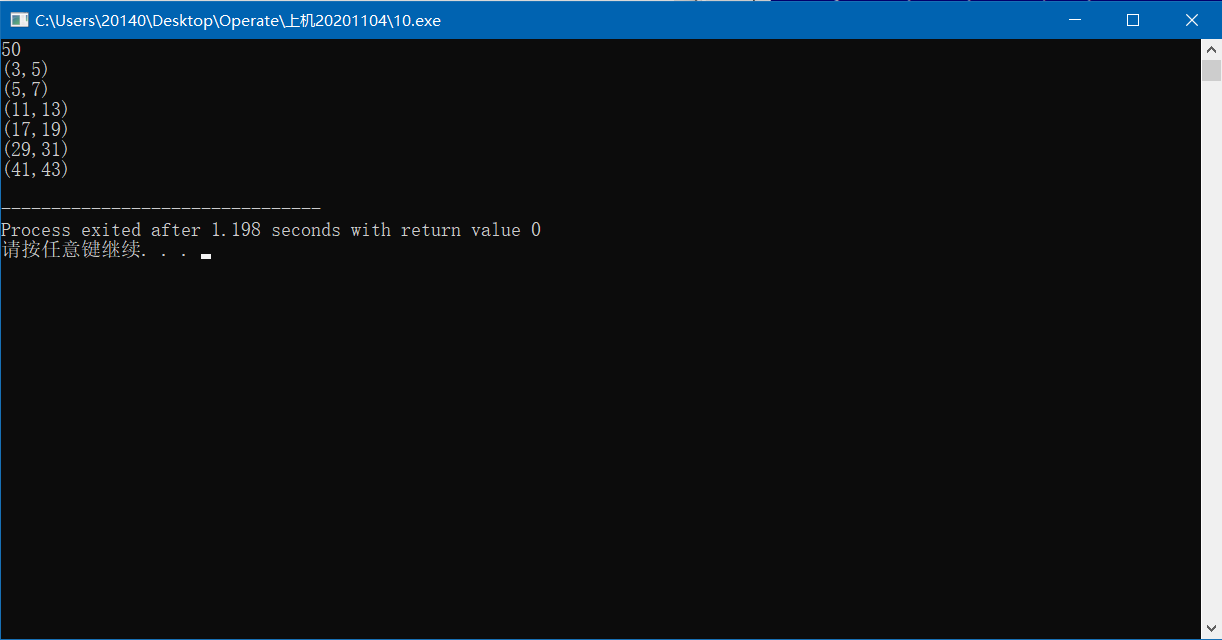
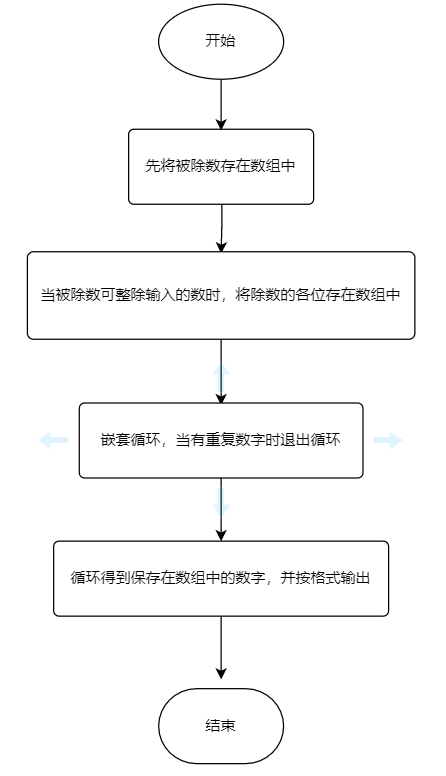


图1-7 程序设计题7的运行结果

8.输出符合形式的表达式

1）解题思路：



2）程序源代码 ：

#include<stdio.h>

int main(){

int p,i=0,j=0,k=0,A[10],x;

long m=0,n=0;

scanf("%d",&p);

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

x=0;

m=n;

for(i=4;i>=0;i--){ //将被除数各位存在数组中

A[i]=m%10;

m=m/10;

}

if((n%p)!=0)

continue;

else { //当被除数可整除输入的数时，将除数的各位存在数组中

m=n/p;

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

A[i]=m%10;

m=m/10;

} }

for(k=0;k<=8;k++){ //嵌套循环，当有重复数字时退出循环

for(j=k+1;j<=9;j++){

if(A[k]==A[j]){

x=1;

break;}

}

if(x==1)

break;

}

if(x==0) { //循环得到保存在数组中的数字，并按格式输出

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

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

printf("/");

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

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

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

}

}

return 0;

}

3）测试：

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出 |
| 1 | 32 | 75168/02349=32 |
| 2 | 24 | 39528/01647=24  46872/01953=24 |
| 3 | 68 | 98736/01452=68 |

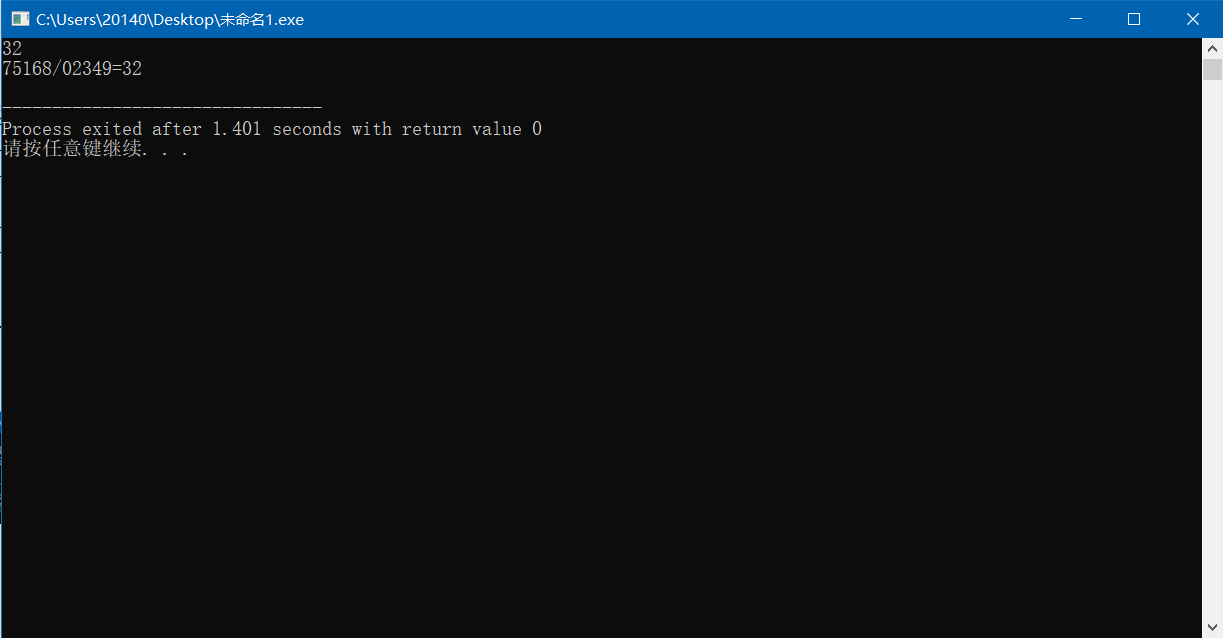
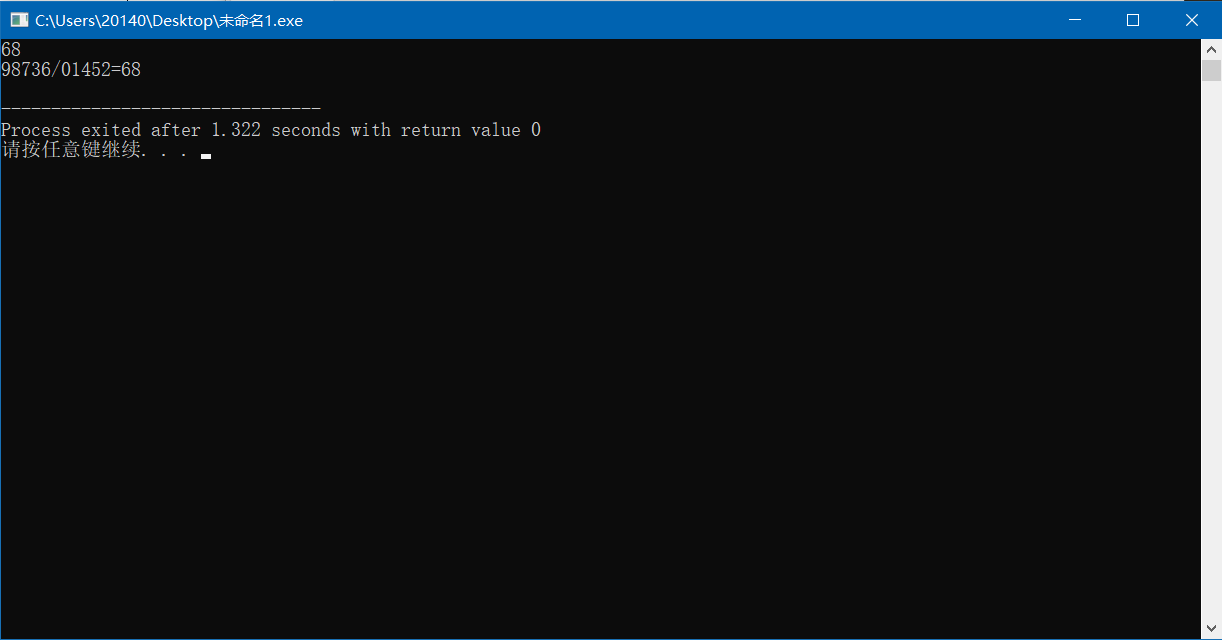
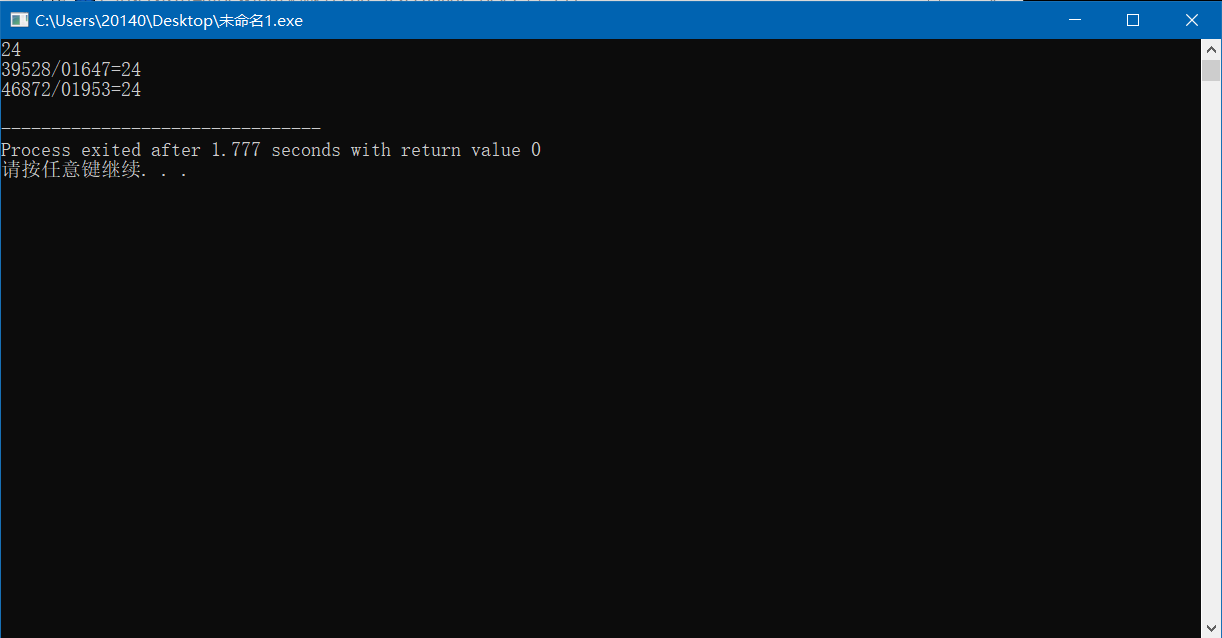


图1-8 程序设计题8的运行结果

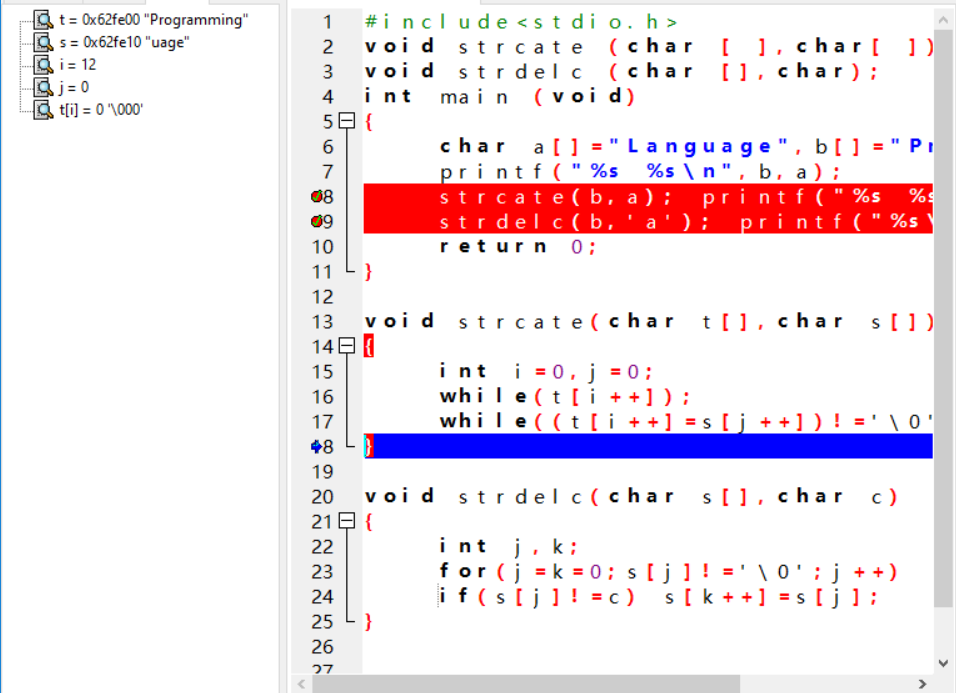
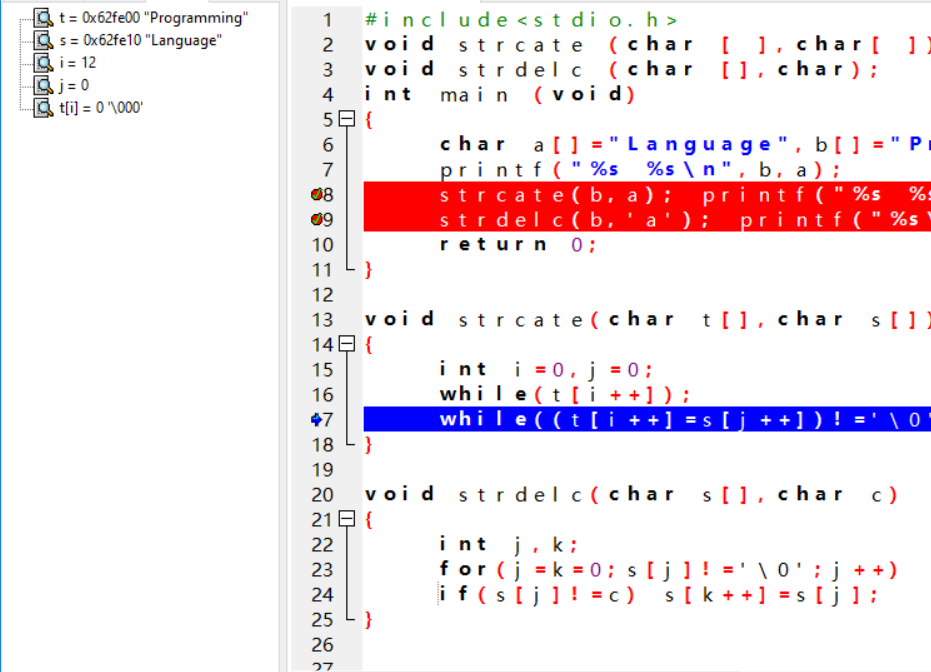
## 1.4 小结

通过本节有关程序设计中流程控制的程序设计题目，我进一步地练习了如何运用嵌套循环，掌握了用筛法求素数表，但是在按照格式输出结果，算法的实现上尚还欠缺能力。

# 2 数组程序设计实验

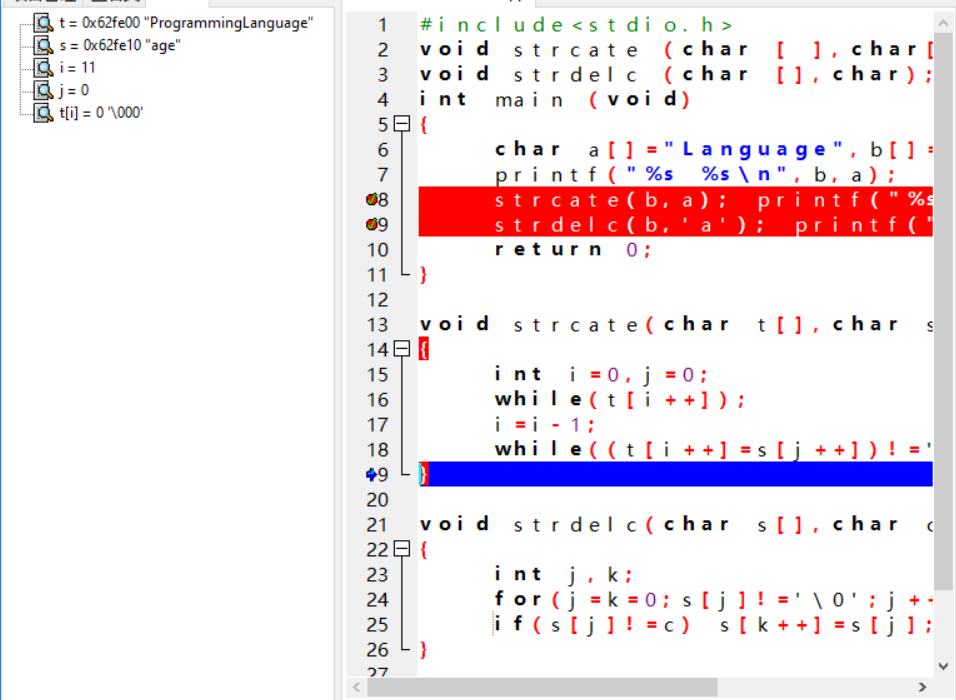
## 2.1 程序改错与跟踪调试

（1）单步执行源程序的跟踪调试

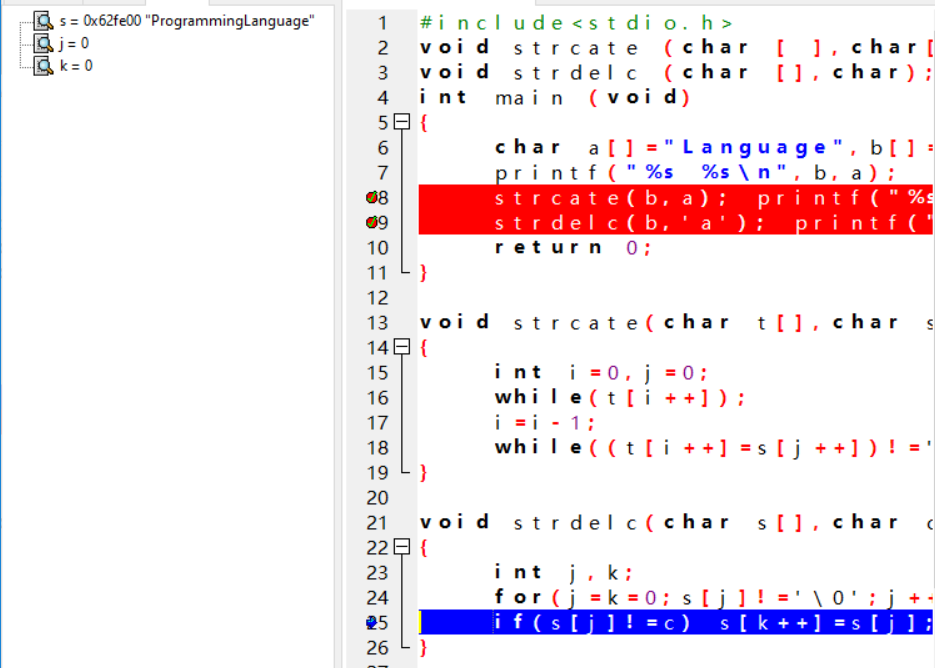


由调试结果可知，当其进入strcate函数时，字符数组t和s的内容都是正确的。当处在第二个while语句时i=12，但是由于“Programming”的字符个数是11，因此此时应该是在t[11]的位置，所以i应该减去1;才能顺利完成后续的拼接操作。而当执行到花括号时，发现按预料的一样并未执行正确。因为t[11]的值也为‘\0’，所以此时字符数组t的结果就像是没有拼接一样，其实是从t[12]开始进行了复制操作。我们添加代码i=i-1;后得到如下结果:

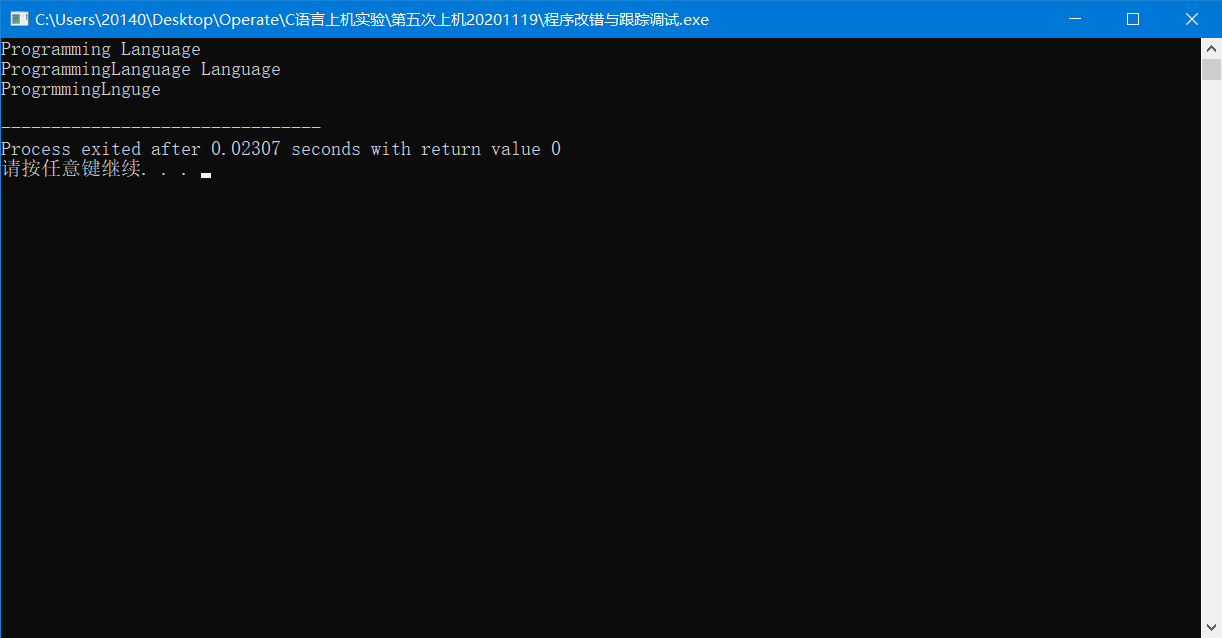
我们发现此时s的值输出并不正确。



（2）跟踪进入strdelc函数



我们看到，初始进入函数时值是正确的，然而最终得到的结果却是ProgrmmingLngugeage，正确结果是ProgrmmingLnguge，可见数组末尾多了“age”三个字符。这是由于数组本该在s[k]处终止，因此，要将其值赋为‘\0’。最后得到结果如图：



最终代码修改方案如下：

#include<stdio.h>

void strcate (char [ ],char[ ]);

void strdelc (char [],char);

int main (void)

{

char a[]="Language",b[]="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[])

{

int i=0,j=0;

while(t[i++]);

此处添加代码i=i-1;

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

}

void strdelc(char s[],char c)

{

int j,k;

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

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

此处添加代码s[k]='\0';

}

## 2.2 程序完善与修改替换

（1）文本去重

①完善后代码：

# include<stdio.h>

# include<string.h>

void RemoveDuplicate(char \*s);

int main( ){

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]) {

s[w++]=s[r];

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

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

s[i]='\0';

}

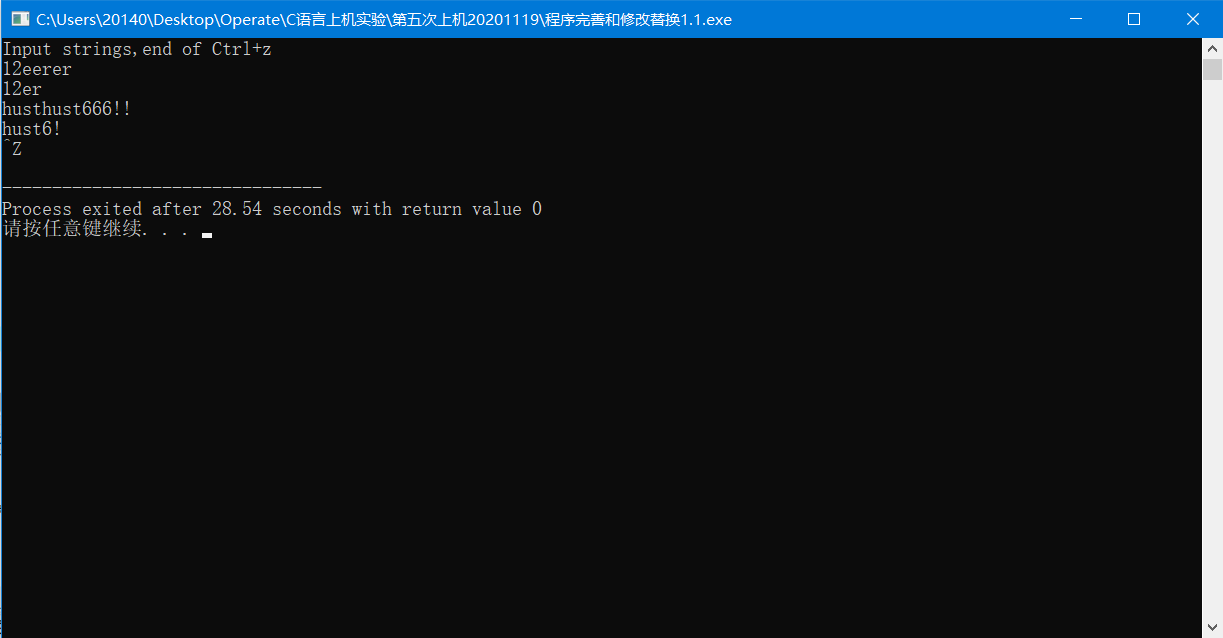
}

}

s[w]='\0';

}

运行结果如下：



② 空间换时间提高效率：

# include<stdio.h>

# include<string.h>

void RemoveDuplicate(char \*s);

int main( ){

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 i, j, len, remainder;

int check[8] = {0};

len = strlen(s);

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

remainder = s[i] % 32;

if((check[s[i] >> 5] & (1 << remainder)) == 0) {

s[j++] = s[i];

check[s[i] >> 5] |= (1 << remainder);

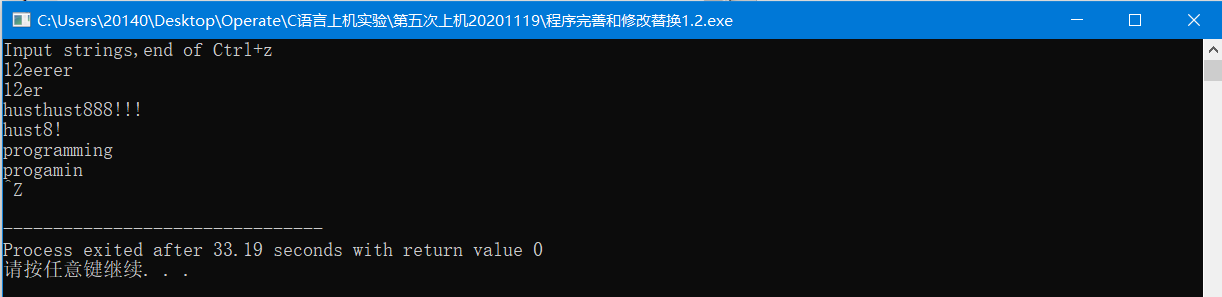
}

}

s[j] = '\0';

}

运行后截图：



（2）约瑟夫问题

①完善代码：

# include<stdio.h>

# include<string.h>

int main (void)

{

int M,N;

scanf("%d%d",&M,&N);

int a[M],b[M];

int i,j,k;

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

a[i]=i+1;

for(i=M,j=0;i>1;i--)

{

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

if(++j>i-1) j=0;

b[M-i]=(j?a[j-1]:a[i-1]);

if(j)

for(k=--j;k<i;k++)

a[k]=a[k+1];

}

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

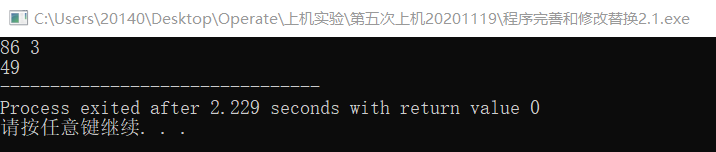
printf("%d\n",b[i]);

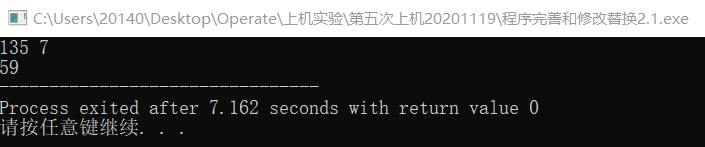
printf("%d",a[0]);

return 0;

}

运行截图如下





②提高效率：

#include <stdio.h>

int main()

{

int j,m,i=0;

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

int check[j];

for(;i<j;i++)

check[i]=0;

int k=0,count=1,n=j;

for(i=0;n>=1;)

{

if(m==count)

{

k=i%j;

check[k]=1;

n--;

count=0;

//此时遍历下标还在出列元素，而下一个元素才是从1开始报数，

//故此处将count置为0，当下标i遍历至下一个未出列元素时count再加1

}

i++;

if (check[i%j]!=1)

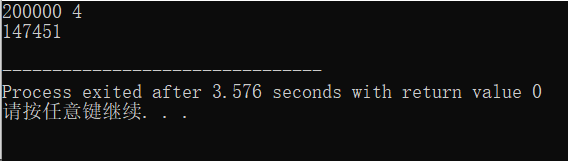
count++;

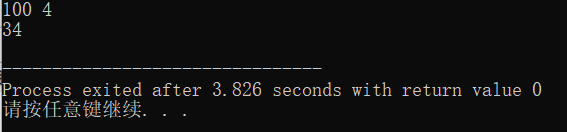
}

printf("%d\n",k+1);

}

运行截图如下

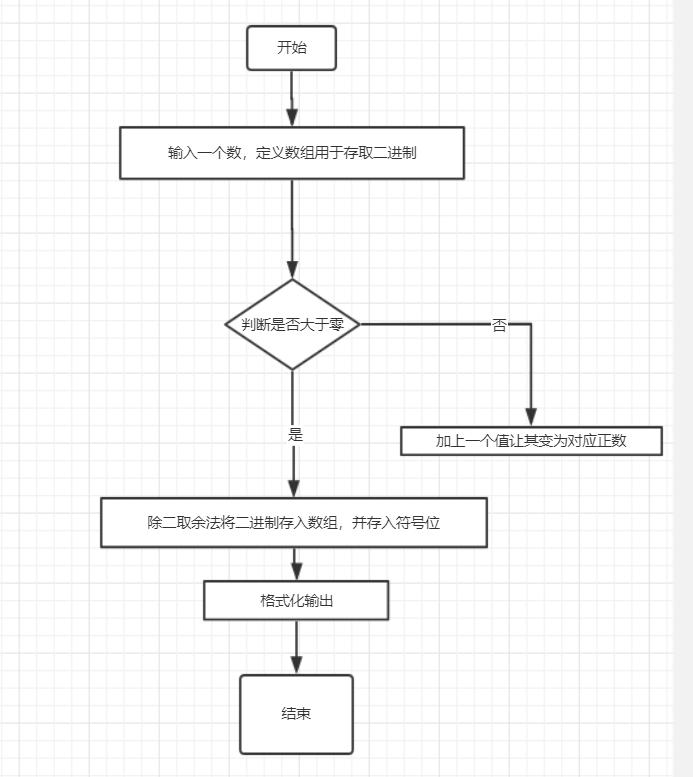




## 2.3 程序设计

1.输出二进制

1）解题思路：



1. 源代码：

#include<stdio.h>

#include<math.h>

#include<string.h>

int main(void){

int m,n,flag,count=0;

scanf("%d",&m);

int a[50];

if(m>=0) n=m;

//判断正负，如果是正数不变，如果是负数则根据二进制原理加上相应值

Else n=m+(int)pow(2,32)+1;

for(int j=0;j<32;j++){

//采用除二取余法依次将二进制从低位到高位存入数组

flag=n%2; n=(int)(n/2); count++;

a[j]=flag;

}

if(m<0) a[31]=1; //如果小于零，则把符号位置为1

for(int j=count-1;j>=0;j--){ //从高位到低位输出二进制

if(j!=0&&(j+1)%4==0&&(j+1)!=32)

printf(" ");//实现按格式输出

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

}

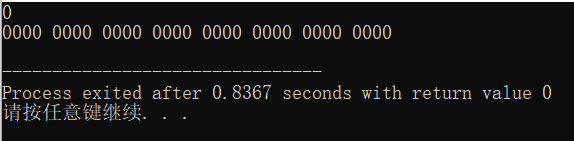
printf("\n");

return 0;

}

3）测试：

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出 |
| 1 | 0 | 0000 0000 0000 0000 0000 0000 0000 0000 |
| 2 | -1 | 1111 1111 1111 1111 1111 1111 1111 1111 |
| 3 | 123 | 0000 0000 0000 0000 0000 0000 0111 1011 |



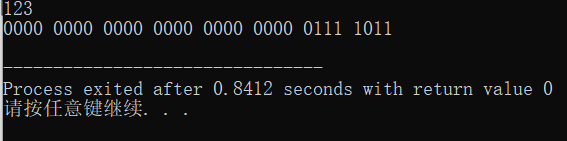
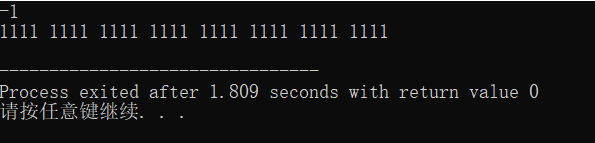
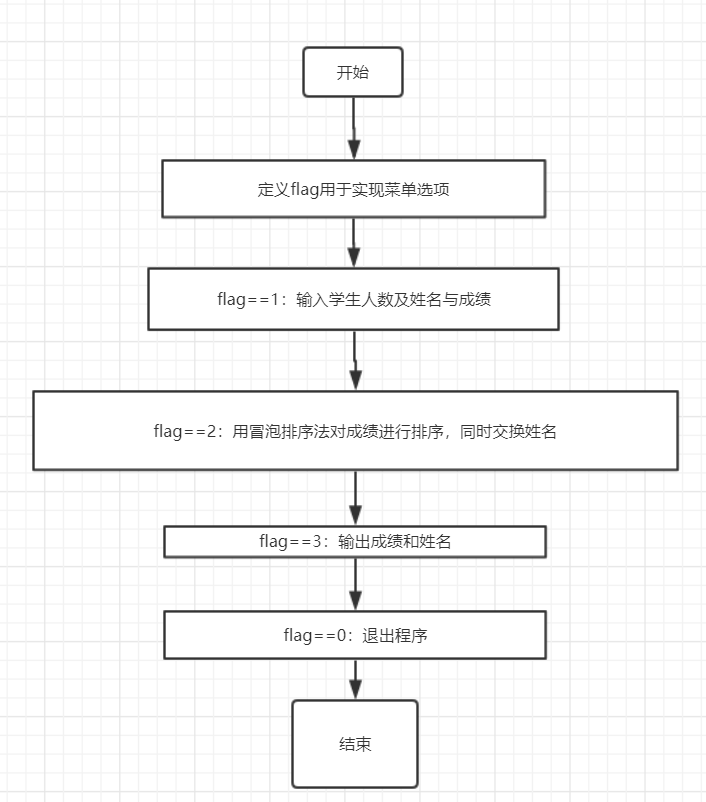


图2-1程序设计题1的运行结果

1. 将成绩与对应姓名进行排序
2. 解题思路：



1. 源代码：

#include<stdio.h>

//声明冒泡排序函数

void bubble\_sort(char a[][10],int b[],int s);

int main(void){

int flag=1;

int score[1000];

int number;

char name[1000][10];

while(flag!=0) {//使用while函数实现多次输入指令

scanf("%d",&flag);

if(flag==1) {

scanf("%d",&number);

for(int i=0;i<number;i++){//当flag的值为1时，提示输入姓名与成绩

scanf("%s",&name[i]);

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

}

}

if(flag==2)

//值为2时，进行排序但并不输出

bubble\_sort(name,score,number);

if(flag==3){//值为3时，进行输出

for(int i=0;i<number;i++) {

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

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

} } }

return 0;

}

void bubble\_sort(char a[][10],int b[],int s){

int i,j,t;

char m[1000][10];

for (i=0;i<s-1;i++) {

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

if(b[j]<b[j+1]){

t=b[j],b[j]=b[j+1],b[j+1]=t;

for(int k=0;k<10;k++)

{//在冒泡排序成绩时，同时对成绩对应的姓名进行交换操作

m[j][k]=a[j][k];

a[j][k]=a[j+1][k];

a[j+1][k]=m[j][k];

}

}

if(b[j]==b[j+1])

{//如果成绩相等，那么就比较姓名首字母，若仍相同比较下一个字母，以此类推进行排序

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

if(a[j][i]>a[j+1][i]){

for(int k=0;k<10;k++){

m[j][k]=a[j][k];

a[j][k]=a[j+1][k];

a[j+1][k]=m[j][k];

}

break;

}

if(a[j][i]==a[j+1][i]) continue;

//当首字母相同时，继续下一个循环

else break;

} }

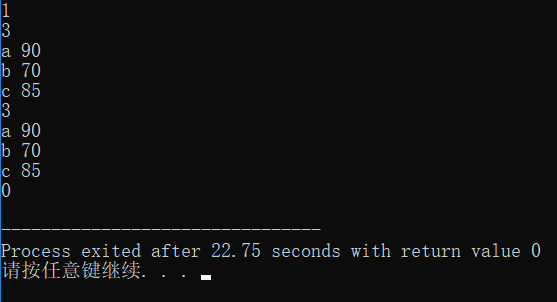
}

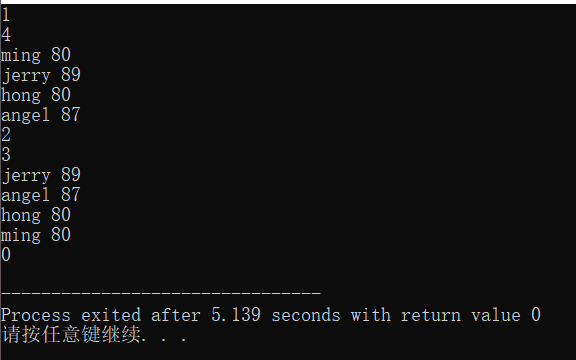
}

}

1. 测试：

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出(换行符在输出截图中展示) |
| 1 | 1 3 a 90 b 70 c 85 3 0 | a 90 b 70 c 85 |
| 2 | 1 4 ming 80 jerry 89 hong 80 angel 87 2 3 0 | jerry 89 angel 87  hong 80 ming 80 |
| 3 | 1 4 xhong 80 jerry 89  xming 80 angel 87 2 3 0 | jerry 89 angel 87  xhong 80 xming 80 |





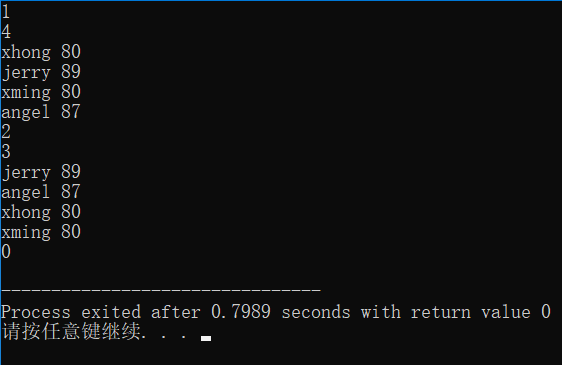
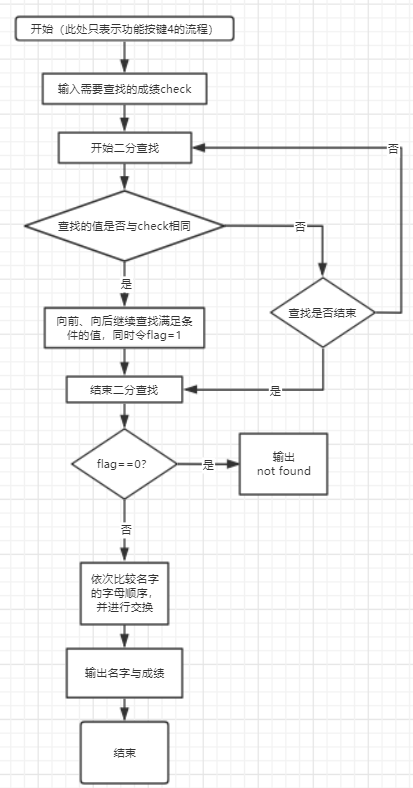


图2-2程序设计题2的运行结果

1. 对第2题增加查找功能
2. 解题思路：只需在第二题的基础上添加功能按键4,，查找功能的流程图如下



1. 源代码：以下只在功能按键4处添加注释

#include<stdio.h>

#include<string.h>

void bubble\_sort(char a[][10],int b[],int s);

int search(int score[],int check,int number);

int main(void)

{

int flag=1;

int score[1000];

int number;

char name[1000][10];

while(flag!=0)

{

scanf("%d",&flag);

if(flag==1)

{

scanf("%d",&number);

for(int i=0;i<number;i++)

{

scanf("%s",&name[i]);

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

}

}

if(flag==2)

bubble\_sort(name,score,number);

if(flag==3)

{

for(int i=0;i<number;i++)

{

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

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

}

}

if(flag==4)

{

int check;

scanf("%d",&check);

int i=0,j=number-1,flag=0,k,result[10],p,q;

while(i<=j)

{//二分查找

k=(i+j)/2;

if(check>score[k]) j=k-1;

else if(check<score[k]) i=k+1;

else{

//当找到第一个满足的值时，向前向后继续寻找满足条件的值

q=p=k;

while(score[--p]==check);

while(score[++q]==check);

flag=1;

break;

} }

if(flag==0) printf("not found");

//flag==0，也即没有找到该成绩

else

{

for(int j=p+1;j<q;j++)

{//下面比较名字的字母顺序，并将字母顺序靠前的姓名通过交换数组来移动

for(int i=0;i<4&&j<q-1;i++)

{

if(name[j][i]>name[j+1][i])

{

printf("enter!\n");

char m[20];

strcpy(m,name[j]);

strcpy(name[j],name[j+1]);

strcpy(name[j+1],m);

break;

}

else break;

}

//输出符合条件的成绩和姓名

printf("%s %d\n",name[j],score[j]);

}

}

}

}

return 0;

}

void bubble\_sort(char a[][10],int b[],int s)

{

int i,j,t;

char m[1000][10];

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

{

for(j=0;j<s-i-1;j++)

{

if(b[j]<b[j+1])

{

t=b[j],b[j]=b[j+1],b[j+1]=t;

for(int k=0;k<10;k++)

{

m[j][k]=a[j][k];

a[j][k]=a[j+1][k];

a[j+1][k]=m[j][k];

}

}

if(b[j]==b[j+1])

{

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

{

if(a[j][i]>a[j+1][i])

{

for(int k=0;k<10;k++)

{

m[j][k]=a[j][k];

a[j][k]=a[j+1][k];

a[j+1][k]=m[j][k];

}

break;

}

else if(a[j][i]==a[j+1][i]) continue;

else break;

}

}

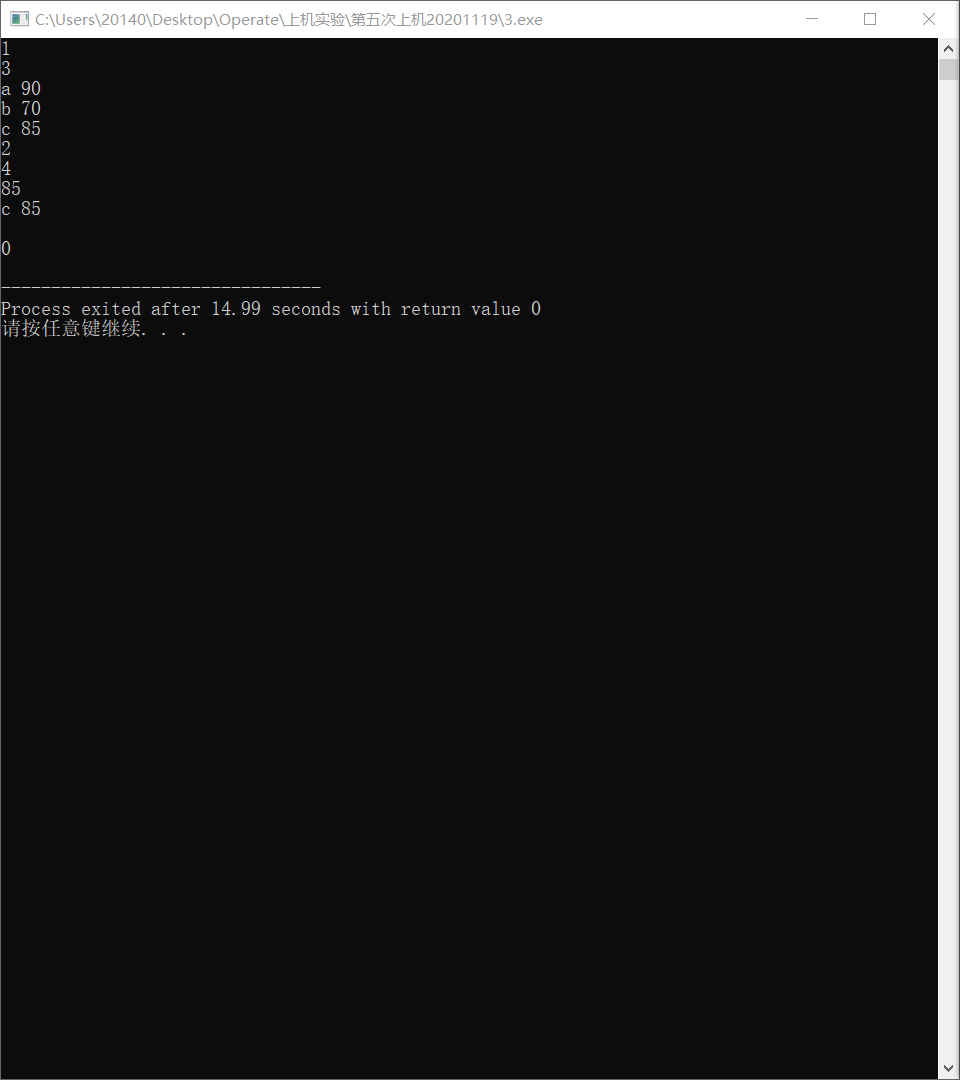
}

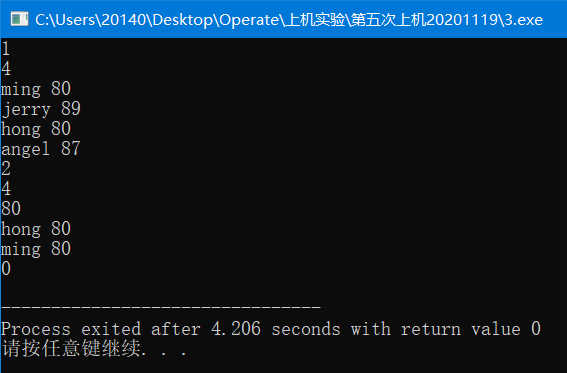
}

}

1. 测试：

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出 |
| 1 | 1 3 a 90 b 70 c 85 2 4 85 0 | c 85 |
| 2 | 1 4 ming 80 jerry 89 hong 80  angel 87 2 4 80 0 | hong 80  ming 80 |
| 3 | 1 4 xhong 80 jerry 89 xming 80  angel 87 2 4 80 0 | xhong 80  xming 80 |





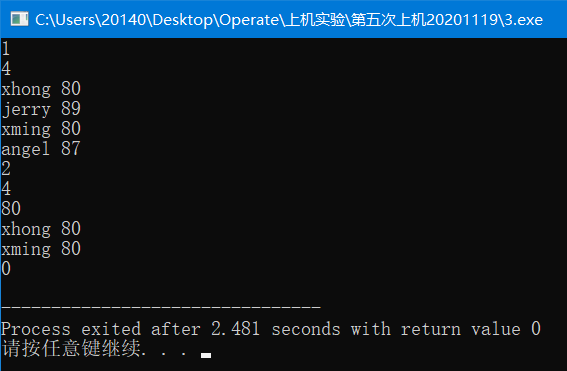
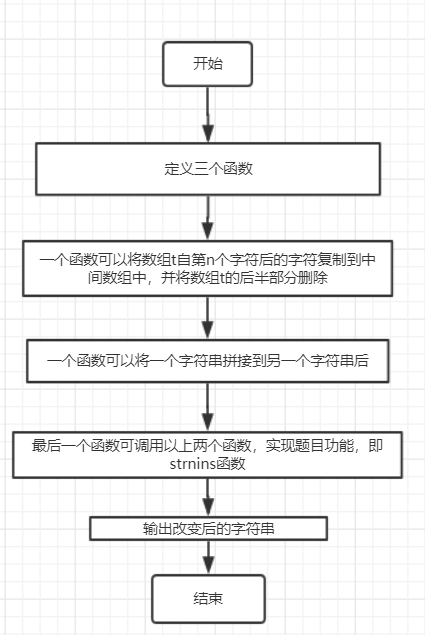


图2-3程序设计题3的运行结果

5.编写插入字符串函数

1）解题思路：



2）代码：

#include<stdio.h>

//声明函数

void codel(char t[],char mid[],int n);

void strcate(char t[],char s[]);

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

int main(void){

char t[100],s[100];

int n;

scanf("%s%s%d",t,s,&n);

strnins(t,s,n);

printf("%s",t);

}

void codel(char t[],char mid[],int n)

{//将数组t自第n个字符后的字符复制到中间数组中，并将数组t的后半部分删除

int i=0,k=n;

for(;t[k]!='\0';i++,k++)

mid[i]=t[k];

mid[i]='\0';

t[n]='\0';

}

void strcate(char t[],char s[])

{//将一个字符串拼接到另一个字符串后

int i=0,j=0,k=-1;

while(t[i++]); i--;

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

}

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

{//调用以上两个函数，实现插入字符串功能，

char mid[100];

codel(t,mid,n);

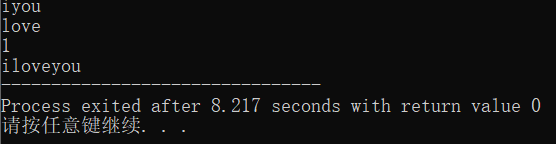
strcate(t,s);

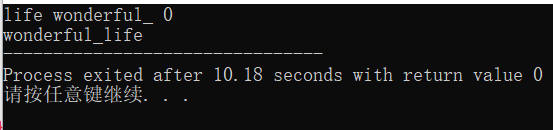
strcate(t,mid);

}

1. 测试

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入（依次为t,s,n） | 输出 |
| 1 | iyou love 1 | iloveyou |
| 2 | life wonderful\_ 0 | wonderful\_life |
| 3 | xuda rui 2 | xuruida |





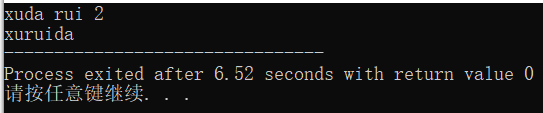
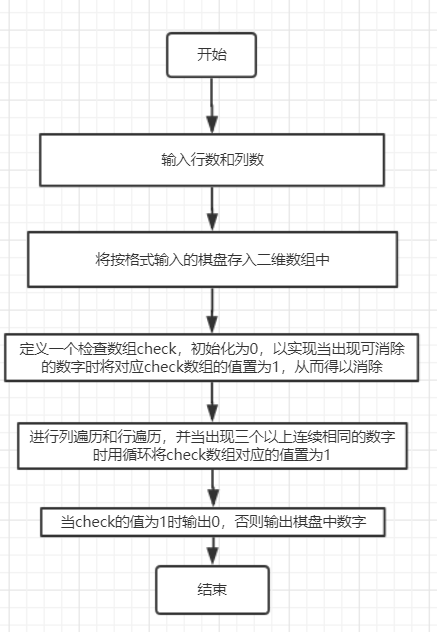


图2-4程序设计题5的运行结果

1. 消除类游戏
2. 解题思路



1. 代码：

#include<stdio.h>

int main(void){

int line,raw,count=1,flag=0;

scanf("%d%d",&line,&raw);

int chess[100][100];

int check[100][100]={0};

//定义一个检查数组check，初始化为0，以实现当出现可消除的数字时将对应check数组的值置为1，从而得以消除

int j=0,k=0;

//将棋盘存入数组中

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

for(int j=0;j<raw;j++)

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

}

//行遍历，当出现三个以上连续相同的数字时用循环将check数组对应的值置为1

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

count=1;

flag=chess[i][0];

for(j=1;j<=raw;j++){

if(flag==chess[i][j])

count++;

else{

if(count>=3) {

while(count){

check[i][j-count]=1;

count--;

}

}

count=1;

flag=chess[i][j];

} } }

//列遍历，当出现三个以上连续相同的数字时用循环将check数组对应的值置为1

for (int i=0;i<raw;i++){

flag=chess[0][i];

count=1;

for(j=1;j<=line;j++){

if(flag==chess[j][i])

count++;

else{

if(count>=3) {

while(count){

check[j-count][i]=1;

count--;

} }

count=1;

flag=chess[j][i];

} } }

printf("\n\n");

//格式化输出——当check的值为1时输出0，否则输出棋盘中数字

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

for(int j=0;j<raw;j++){

if(check[i][j]==1){

printf("0");

if(j<raw-1) printf(" ");

}

else{

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

if(j<raw-1) printf(" ");

} }

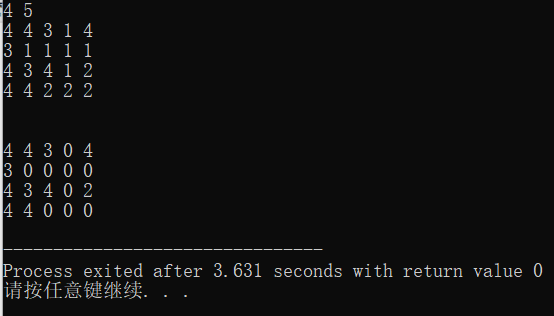
printf("\n"); }

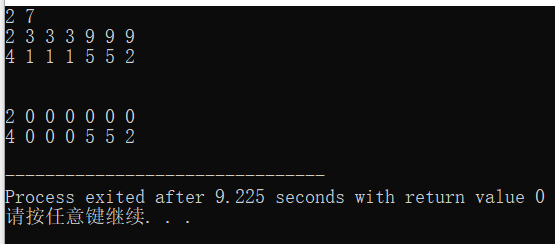
return 0;

}

1. 测试

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出 |
| 1 | 4 4 3 1 4  3 1 1 1 1  4 3 4 1 2  4 4 2 2 2 | 4 4 3 0 4  3 0 0 0 0  4 3 4 0 2  4 4 0 0 0 |
| 2 | 2 3 3 3 9 9 9  4 1 1 1 5 5 2 | 2 0 0 0 0 0 0  4 0 0 0 5 5 2 |
| 3 | 3 4 5 6 6 6  2 3 5 6 6 7  1 8 5 9 6 8  4 8 5 7 7 7  4 3 5 2 7 8  4 5 4 3 7 8 | 3 4 0 0 0 0  2 3 0 6 0 7  1 8 0 9 0 8  0 8 0 0 0 0  0 3 0 2 0 8  0 5 4 3 0 8 |





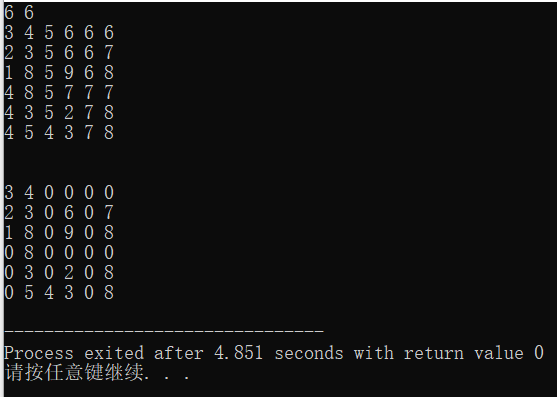
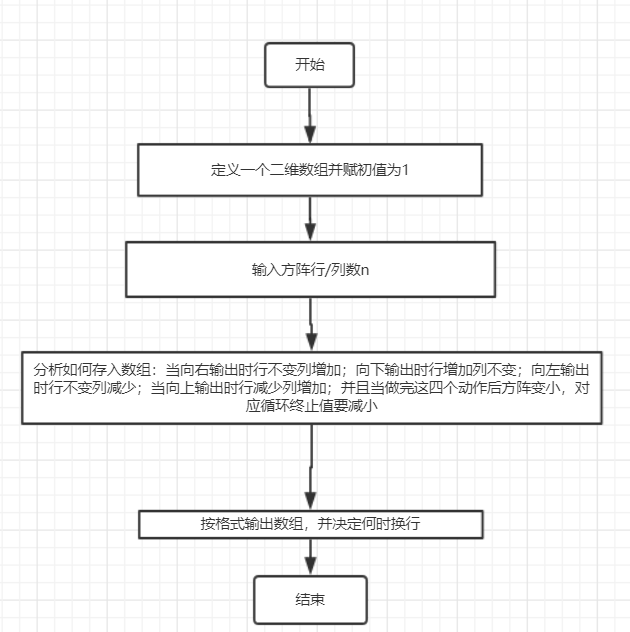


图2-5程序设计题6的运行结果

1. n阶顺转方阵
2. 解题思路



1. 代码

#include<stdio.h>

int main(){

int i,j,m,base=1,n;

int a[10][10]={1};

scanf("%d",&n);

for(m=0;m<n/2;m++) {

//当做完这四个动作后方阵变小，对应循环终止值要减小

for(i=m;i<=n-1-m;i++)

//当向右输出时行不变列增加

a[m][i]=base++;

for(j=m+1;j<n-1-m;j++)

//向下输出时行增加列不变

a[j][n-1-m]=base++;

for(i=n-1-m;i>m;i--)

//向左输出时行不变列减少

a[n-1-m][i]=base++;

for(j=n-1-m;j>m;j--)

//当向上输出时行减少列增加

a[j][m]=base++;

if(n%2==1)

//到最后一个值时若行数为奇数要单独输出

a[(n-1)/2][(n-1)/2]=base;

}

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

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

printf("%2d",a[i][j]);

if(j!=(n-1))

printf(" ");

}

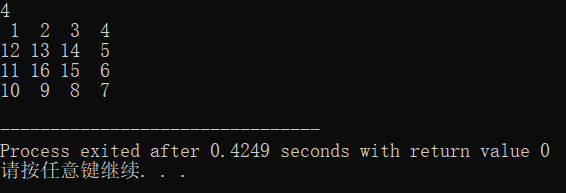
printf("\n");

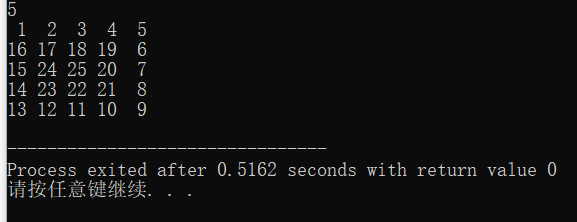
}

return 0;

}

1. 测试（为方便检验，按对齐格式输出）





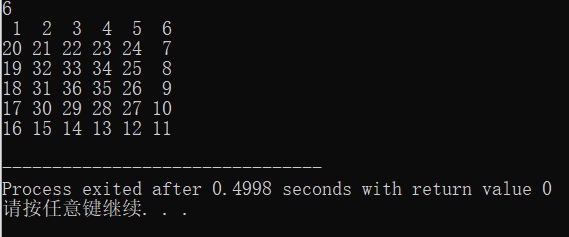
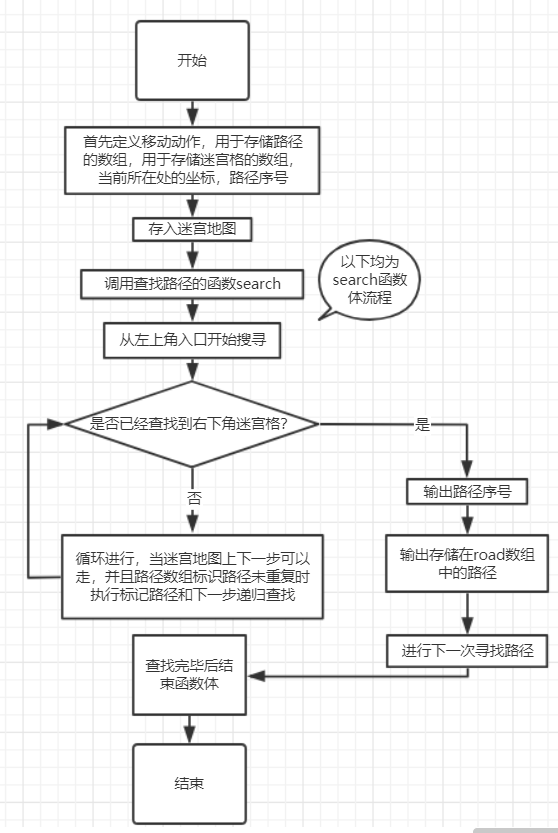


图2-6程序设计题10的运行结果

1. 迷宫问题
2. 解题思路：



1. 代码：

#include<stdio.h>

#define ph1 x+moveX[i]>=0

#define ph2 x+moveX[i]<position\_x

#define ph3 y+moveY[i]>=0

#define ph4 y+moveY[i]<position\_y

#define ph5 road[x+moveX[i]][y+moveY[i]]==0

#define ph6 map[x+moveX[i]][y+moveY[i]]==1

#define ph7 road[x+moveX[i]][y+moveY[i]]=1

#define ph8 search(x+moveX[i],y+moveY[i])

#define ph9 road[x+moveX[i]][y+moveY[i]]=0

//宏定义运算表达式

int map[20][20]={};

int road[20][20]={};

void search(int,int);

int moveX[4]={1,0,0,-1};

int moveY[4]={0,1,-1,0};

//定义在x和y方向的移动动作，即二者结合可以进行四个方向的移动，从而可以搜寻四个方向的路径

int position\_x,position\_y,num=0;

//定义当前的位置和路径序号

int main(){

scanf("%d %d",&position\_x,&position\_y);

int i,j;

for(i=0;i<position\_x;i++){

for(j=0;j<position\_y;j++)

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

}//将迷宫图存入数组中

search(0,0);//调用迷宫进行查找路径

return 0;

}

void search(int x,int y){

int i,j;

if(x==position\_x-1&&y==position\_y-1){

//当已经查找到迷宫右下角时，输出已经查找到的路径

num++;//标记已经找到的路径序号

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

for(i=0;i<position\_x;i++){

for(j=0;j<position\_y;j++){

//输出已经存储在road中的路径

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

if(j<position\_y-1) printf(" ");

}

printf("\n");

}

return;//输出完成后进行下一次查找路径

}

else{

road[0][0]=1;

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

if(ph1&&ph2&&ph3&&ph4&&ph5&&ph6){

//当迷宫地图上下一步是1，即可以走并满足不与已找到的路径重复时时继续执行

ph7;//标记自己已经走过的迷宫格为1

ph8;//递归进行下一步查找

ph9;

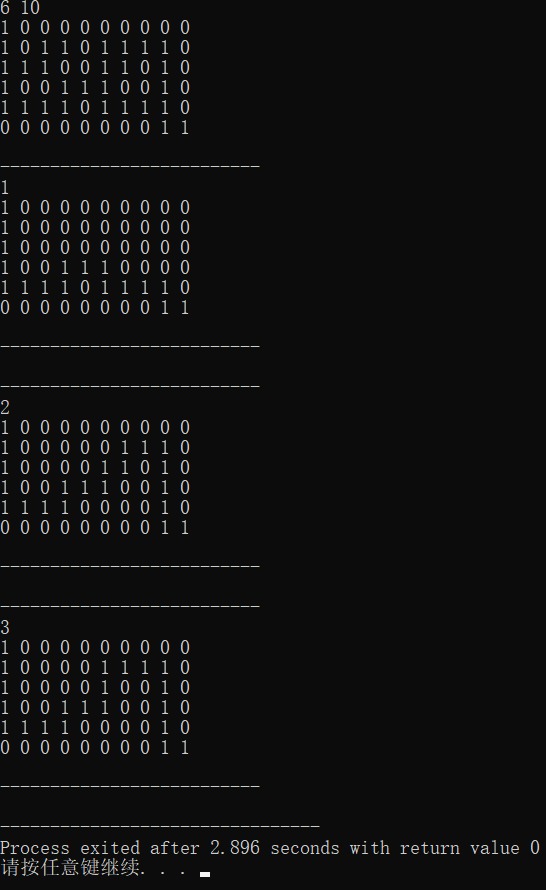
}

}

}

}

1. 测试：（直接在截图中展示，为方便观测，添加了分隔符）



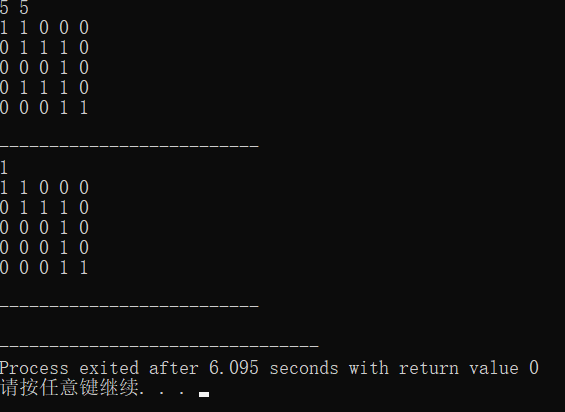


图2-7程序设计题11的运行结果

## 2.4小结

通过本节对数组的问题的解决，我意识到了自己对于算法改进方面的不足。

在程序调试中，我一直对数组s的内容明明没有被改变，但是在调试时却一直有数组内容被修改的问题感到困惑不解，在对观察变量的下标范围进行改变后，才意识到问题所在。

在程序修改替换一题中，标记数组法是降低算法时间复杂度的重要方法。尽管在处理文本去重的改进中较为顺利，但是在约瑟夫问题中却遇到了困难，由于一开始对约瑟夫问题的算法没有理解透彻，以致于我在标记数组后的下一步循环报数过程中处理得不顺利。

在程序设计中，我在位运算方面掌握能力的欠缺导致处理二进制输出这一题时的处理方法不够高级。而针对成绩输入与查找方面，我认为在把相同成绩的人的名字按顺序输出这方面耗费了大量的编程时间。最终，通过添加了一行代码才解决了始终无法交换成绩相同的人的名字的代码中的漏洞。

消除类游戏让我建立了利用标记数组法解决问题的观念。在最初写代码时，由于需要进行行遍历和列遍历来进行消除，导致在进行行遍历后列遍历无法正常进行。因为此时原来的游戏棋盘已经被改变。而解决这一问题的有效手段便是另外定义一个检查数组，通过标记对应元素的方法实现消除功能。

N阶顺转方阵在掌握算法后并不难以解决。而在最后一题——迷宫问题中，在开始时我已经有了向各个方向查找，存储路径的想法，但是却忽略了当路走不通时回溯和路径重复的问题，导致写代码时总是无法正确输出结果。最后在查找相关的迷宫问题算法后才得到了解决，同时又借鉴了其将查找功能封装为函数和

递归的算法。

总之在这次实践中，我掌握了标记数组法，空间换时间等方法，但在递归，封装函数，调试查找漏洞，确定算法上还尚待提高。

# 3 结构与联合实验

## 3.1 表达式求值的程序验证

1）分析表达式求值过程：

首先，“.”和“->”优先级最高，左结合，“\*”“++”“--”优先级次之，右结合。同时，先计算括号中表达式。

第一问，指针自增后指向结构数组中的第二个元素，故（++p）->x=100;

第二问，指针自增后指向结构数组中的第二个元素，故p->c=‘B’;

第三问，指针自增后指向结构数组中的第二个元素也即数组v，而p->t是指向了数组v的第一个元素，故\*p->t为x;

第四问，指针自增后指向结构数组中的第二个元素，而p->t是指向了数组v的第一个元素，故\*p->t为x;

第五问，指针指向了第一个元素的数组u的第一个元素,接着自增指向数组u中的第二个元素也就是字符‘V’;

第六问，指针指向第一个元素中的数组u的第一个元素,\*p->t的结果是U，然后自增得到V。

2）编程代码：

#include<stdio.h>

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

struct T

{

int x;

char c;

char \*t;

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

int main(void)

{

int flag=0;

scanf("%d",&flag);

if(flag==1) printf("%d",(++p)->x);

else if(flag==2)printf("%c",(p++,p->c));

else if(flag==3) printf("%c",(\*p++->t,\*p->t));

else if(flag==4)printf("%c",\*(++p)->t);

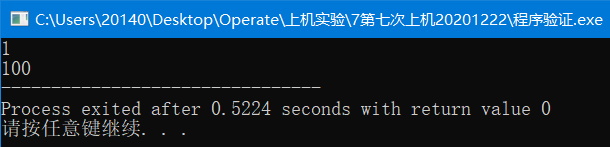
else if(flag==5)printf("%c",\*++p->t);

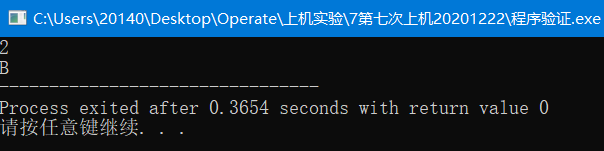
else if(flag==6)printf("%c",++(\*p->t));

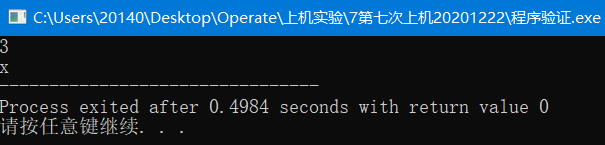
return 0;

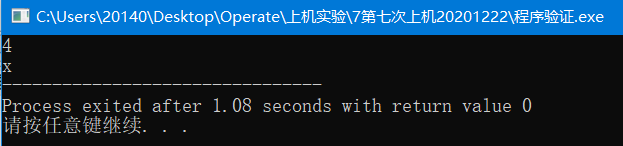
}

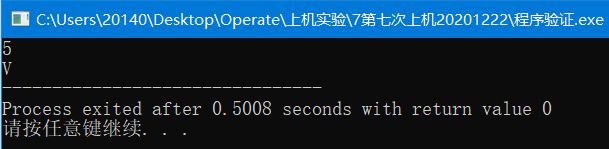
3）截图验证

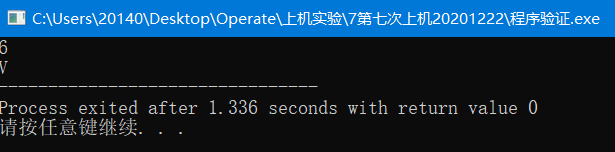












## 3.2 源程序修改替换

1）代码修改与截图验证：

# include<stdio.h>

#include<stdlib.h>

struct s\_list

{

int data;

struct s\_list \* next;

};

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

//此处改为void create\_list(struct s\_list \*\*headp, int \*p);

int main(void)

{

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

int s[100];

for(int i=0;;i++)

{

scanf("%d",&s[i]);//此处根据测试需求改为输入数据

if(s[i]==0) break;

}

create\_list(head,s);//此处改为 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)

{ //此处改为 struct s\_list \*\*headp

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++;

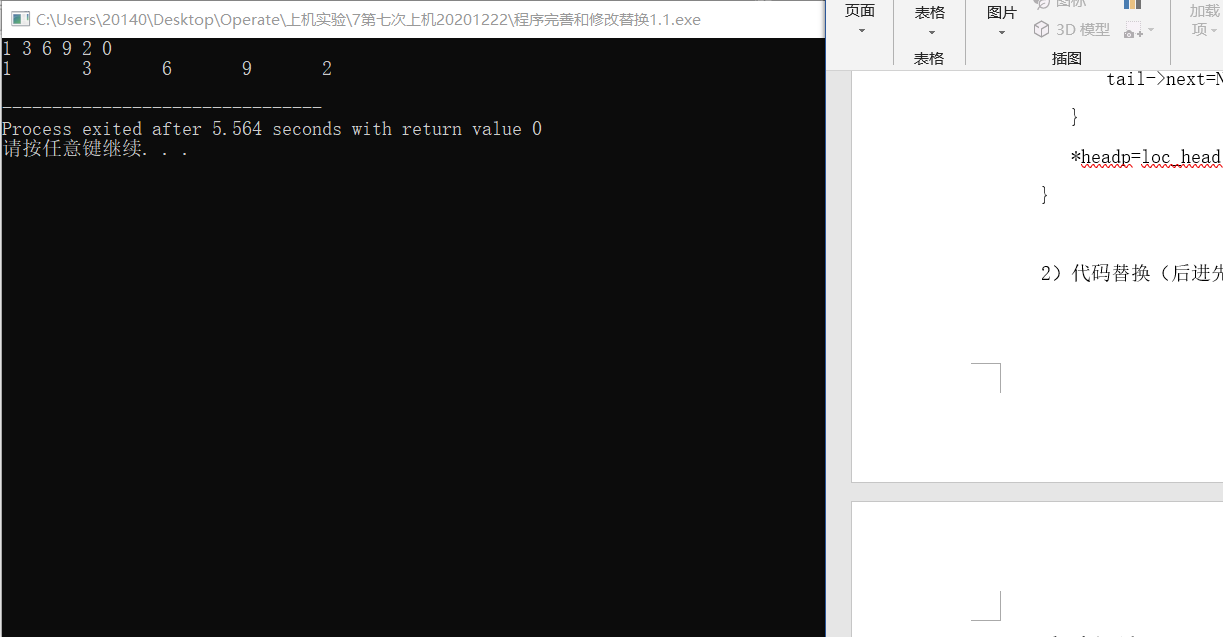
}

tail->next=NULL;

}

\*headp=loc\_head;

}



2）代码替换（后进先出）

# include<stdio.h>

#include<stdlib.h>

struct s\_list

{

int data;

struct s\_list \* next;

};

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

int main(void)

{

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

int s[100];

for(int i=0;;i++)

{

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

if(s[i]==0) break;

}

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;

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++;

}

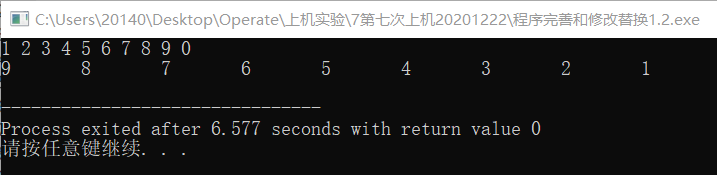
tail->next=NULL;

}

\*headp=loc\_head;

}

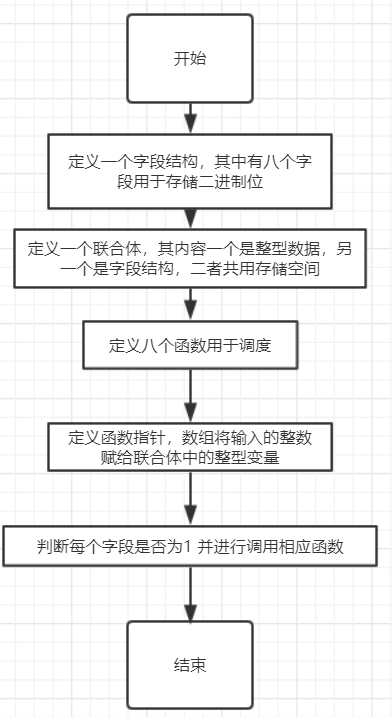
//在第一问修改的基础上对create\_list函数进行修改



## 3.3 程序设计

1.设计字段结构

1）实验思路



2）源代码：

#include<stdio.h>

#include<math.h>

#include<string.h>

typedef struct bits{

unsigned int b0:1,b1:1,b2:1,b3:1,b4:1,b5:1,b6:1,b7:1;

// unsigned int b8:1,b9:1,b10:1,b11:1,b12:1,b13:1,b14:1,b15:1;

}Abit; //构建字段结构

typedef union store{

int m;

Abit intb;

}STORE;//构建联合体，其内容一个是整型数据，另一个是字段结构

void f0(int b)

{

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

}

void f1(int b)

{

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

}

void f2(int b)

{

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

}

void f3(int b)

{

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

}

void f4(int b)

{

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

}

void f5(int b)

{

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

}

void f6(int b)

{

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

}

void f7(int b)

{

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

}//定义八个函数

int main(void)

{

void(\*p\_fun[8])(int b)={f0,f1,f2,f3,f4,f5,f6,f7};

//定义一个函数指针数组，共八个元素，分别指向八个函数

int m,n,flag,count=0;

scanf("%d",&m);//读取输入的整数

STORE usint;

Abit \*bitp;

usint.m=m;//将输入的整数赋给联合体中的整型变量；

// printf("%u,%u,%u",sizeof(int),sizeof(STORE),sizeof(Abit));

if(usint.intb.b0==1) p\_fun[0](0);

if(usint.intb.b1==1) p\_fun[1](1);

if(usint.intb.b2==1) p\_fun[2](2);

if(usint.intb.b3==1) p\_fun[3](3);

if(usint.intb.b4==1) p\_fun[4](4);

if(usint.intb.b5==1) p\_fun[5](5);

if(usint.intb.b6==1) p\_fun[6](6);

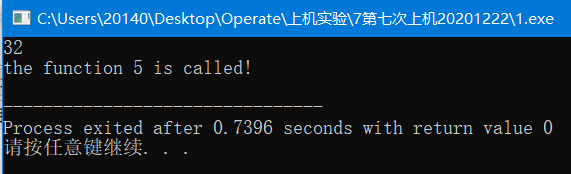
if(usint.intb.b7==1) p\_fun[7](7);//依次判断并调用函数

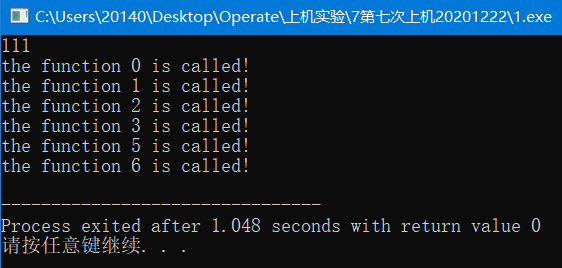
return 0;

}

3）测试结果

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入（括号内为对应二进制） | 输出（只标出调用的函数） |
| 1 | 32（00100000） | 5 |
| 2 | 111（011001111） | 0 1 2 3 5 6 |
| 3 | 255（11111111） | 0 1 2 3 4 5 6 7 |





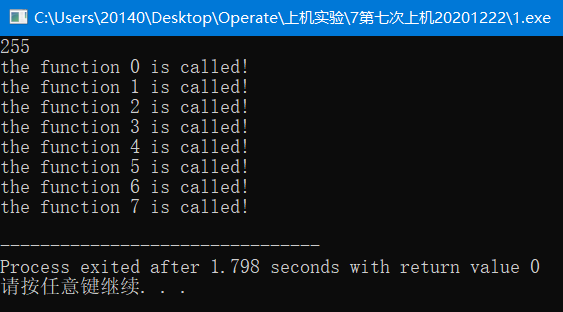
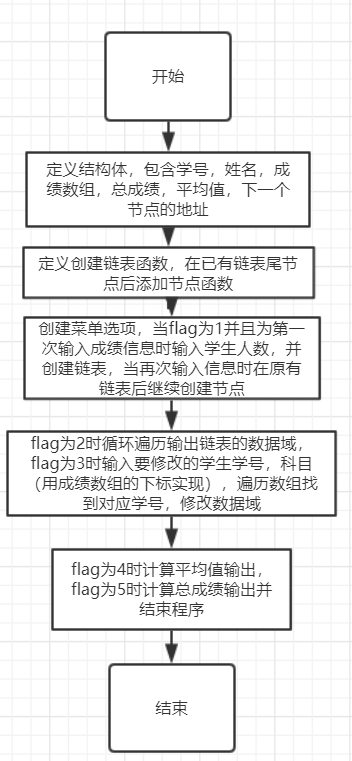


图3-1程序设计题1的运行结果

2.录入成绩单

1）实验思路：



2）源代码：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct s\_list

{

char num[20];

char name[10];

int score[4];

int sum;

float average;

struct s\_list \*next;

}list;

void create\_list(list \*\*headp,int number);

void AddToTail(list \*\* pHead);

int main(void)

{

list \*head=NULL,\*p,\*before;

int number,flag=1,check\_count=0,check\_new,count=1;

char check\_num[20];

while(flag!=5)

{

scanf("%d",&flag);

if(flag==1)

{

scanf("%d",&number);

if(count!=1)//当多次输入信息时第一次以后通过增加节点来录入信息

{

for(int i=0;i<number;i++)

AddToTail(&head);

}

else create\_list(&head,number);

p=head;

while(p)

{

p->sum=p->score[0]+p->score[1]+p->score[2]+p->score[3];

//计算成绩总和

p->average=(float)((p->sum)/4.0);

p=p->next;//计算平均值

}

count++;//记录输入信息次数

}

if(flag==2)

{

p=head;

while(p){

printf("%s %s ",p->num,p->name);

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

{

printf("%d",p->score[i]);

if(i<3) printf(" ");

}

printf("\n");

p=p->next;

}

}

if(flag==3)

{

scanf("%s%d%d",check\_num,&check\_count,&check\_new);

p=head;

while(p)

{

if(strcmp(p->num,check\_num)==0)//遍历比较学号从而查找

{

p->score[check\_count-1]=check\_new;

break;

}

else p=p->next;

}

p=head;

while(p)

{

p->sum=p->score[0]+p->score[1]+p->score[2]+p->score[3];

p->average=(float)((p->sum)/4.0);

p=p->next;//重新计算总和和平均值

}

}

if(flag==4)

{

p=head;

while(p)

{

printf("%s %s %.2f\n",p->num,p->name,p->average);

p=p->next;

}

}

if(flag==5)

{

p=head;

while(p)

{

printf("%s %s %d %.2f\n",p->num,p->name,p->sum,p->average);

p=p->next;

}

return 0;

} } }

void create\_list(list \*\*headp,int number)

{ list \*loc\_head=NULL,\*tail;

char num0[20];

char name0[10];

int score0[4];//依次为English,math,physics,C成绩

scanf("%s%s%d%d%d%d",num0,name0,&score0[0],&score0[1],&score0[2],&score0[3]);

//将录入的信息存入临时数据域中

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

strcpy(loc\_head->num,num0);

strcpy(loc\_head->name,name0);

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

loc\_head->score[i]=score0[i];

//依次赋值

tail=loc\_head;

for(int i=1;i<number;i++){

scanf("%s%s%d%d%d%d",num0,name0,&score0[0],&score0[1],&score0[2],&score0[3]);

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

tail=tail->next;

strcpy(tail->num,num0);

strcpy(tail->name,name0);

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

tail->score[i]=score0[i];

//依次赋值

}

tail->next=NULL;

\*headp=loc\_head;

}

void AddToTail(list \*\*pHead)//在已有链表尾节点后添加新节点

{

list \* pNew=(list \*)malloc(sizeof(struct s\_list));

char num0[20];

char name0[10];

int score0[4];//依次为English,math,physics,C成绩

scanf("%s%s%d%d%d%d",num0,name0,&score0[0],&score0[1],&score0[2],&score0[3]);

strcpy(pNew->num,num0);

strcpy(pNew->name,name0);

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

pNew->score[i]=score0[i];

pNew->next=NULL;

if(\*pHead==NULL)

\*pHead=pNew;

else

{

list \*pNode=\*pHead;

while(pNode->next != NULL)

pNode=pNode->next;

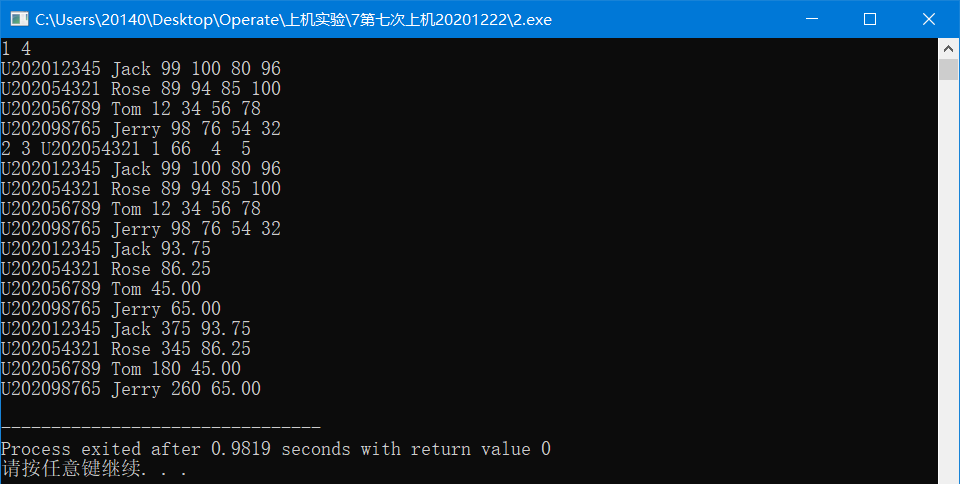
pNode->next=pNew;

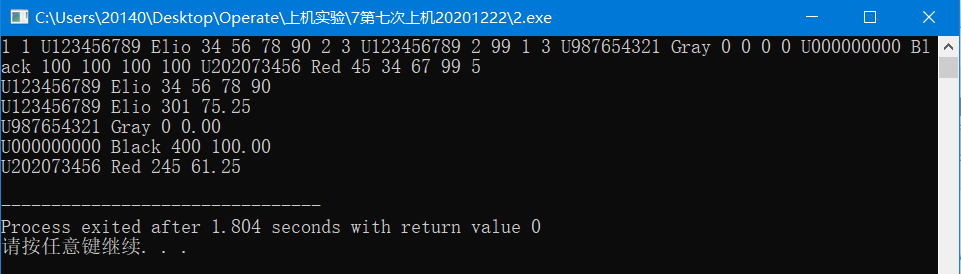
}

}

3）测试结果

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出（均见测试截图） |
| 1 | 1 4  U202012345 Jack 99 100 80 96  U202054321 Rose 89 94 85 100  U202056789 Tom 12 34 56 78  U202098765 Jerry 98 76 54 32  2 3 U202054321 1 66 4 5 |  |
| 2 | 1 1 U123456789 Elio 34 56 78 90  2 3 U123456789 2 99 1 3  U987654321 Gray 0 0 0 0  U000000000 Black 100 100 100 100  U202073456 Red 45 34 67 99 5 |  |
| 3 | 1 2 U202012345 Jack 99 100 80 96 U202054321 Rose 89 94 85 100 2 3 U202054321 1 66 4 5 |  |





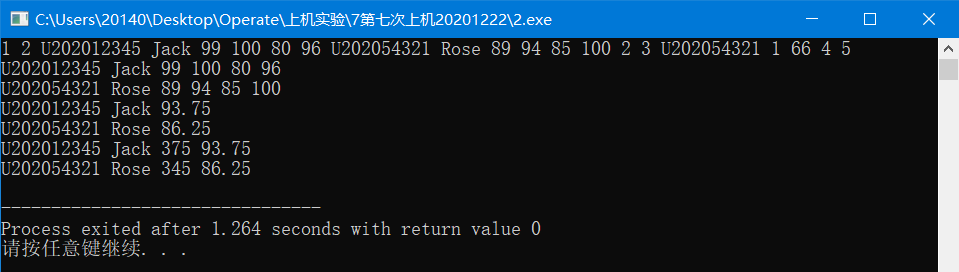
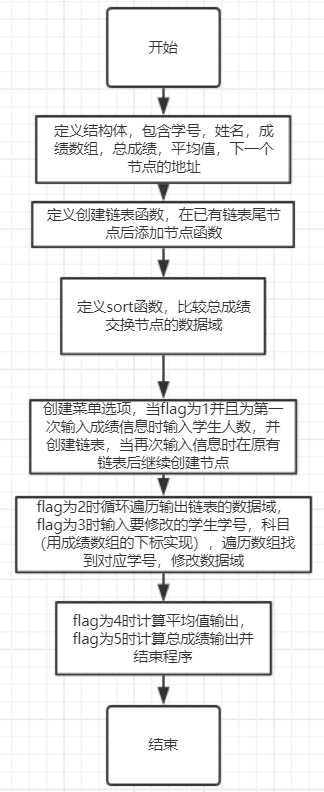


图3-2程序设计题2的运行结果

3.对成绩用交换数据域的方法排序

1）实验思路：

2）源代码（此处对有关排序的代码进行注释）：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct s\_list

{

char num[20];

char name[10];

int score[4];

int sum;

float average;

struct s\_list \*next;

}list;

void create\_list(list \*\*headp,int number);

void AddToTail(list \*\* pHead);

void sort\_lists(list \*head);

int main(void)

{

list \*head=NULL,\*p,\*before;

int number,flag=1,check\_count=0,check\_new,count=1;

char check\_num[20];

while(flag!=5)

{

scanf("%d",&flag);

if(flag==1)

{

scanf("%d",&number);

if(count!=1)

{

for(int i=0;i<number;i++)

AddToTail(&head);

}

else create\_list(&head,number);

p=head;

while(p)

{

p->sum=p->score[0]+p->score[1]+p->score[2]+p->score[3];

p->average=(float)((p->sum)/4.0);

p=p->next;

}

sort\_lists(head);//调用函数进行排序

count++;

}

if(flag==2)

{

p=head;

while(p){

printf("%s %s ",p->num,p->name);

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

{

printf("%d",p->score[i]);

if(i<3) printf(" ");

}

printf("\n");

p=p->next;

}

}

if(flag==3)

{

scanf("%s%d%d",check\_num,&check\_count,&check\_new);

p=head;

while(p)

{

if(strcmp(p->num,check\_num)==0)

{

p->score[check\_count-1]=check\_new;

break;

}

else p=p->next;

}

p=head;

while(p)

{

p->sum=p->score[0]+p->score[1]+p->score[2]+p->score[3];

p->average=(float)((p->sum)/4.0);

p=p->next;

}

}

if(flag==4)

{

p=head;

while(p)

printf("%s %s %.2f\n",p->num,p->name,p->average),p=p->next;

}

if(flag==5)

{

p=head;

while(p)

printf("%s %s %d %.2f\n",p->num,p->name,p->sum,p->average),p=p->next;

return 0;

}

}

}

void create\_list(list \*\*headp,int number)

{ list \*loc\_head=NULL,\*tail;

char num0[20];

char name0[10];

int score0[4];//English,math,physics,C

scanf("%s%s%d%d%d%d",num0,name0,&score0[0],&score0[1],&score0[2],&score0[3]);

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

strcpy(loc\_head->num,num0);

strcpy(loc\_head->name,name0);

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

loc\_head->score[i]=score0[i];

tail=loc\_head;

for(int i=1;i<number;i++){

scanf("%s%s%d%d%d%d",num0,name0,&score0[0],&score0[1],&score0[2],&score0[3]);

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

tail=tail->next;

strcpy(tail->num,num0);

strcpy(tail->name,name0);

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

tail->score[i]=score0[i];

}

tail->next=NULL;

\*headp=loc\_head;

}

void AddToTail(list \*\*pHead)

{

list \* pNew=(list \*)malloc(sizeof(struct s\_list));

char num0[20];

char name0[10];

int score0[4];//English,math,physics,C

scanf("%s%s%d%d%d%d",num0,name0,&score0[0],&score0[1],&score0[2],&score0[3]);

strcpy(pNew->num,num0);

strcpy(pNew->name,name0);

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

pNew->score[i]=score0[i];

pNew->next=NULL;

if(\*pHead==NULL)

\*pHead=pNew;

else

{

list \*pNode=\*pHead;

while(pNode->next != NULL)

pNode=pNode->next;

pNode->next=pNew;

}

}

void sort\_lists(struct s\_list \*head)

{ struct s\_list \*p1=head,\*p2;

int len=0,i,j;

float t;

char temp[100];

while(p1){len++; p1=p1->next;}/\* 计算链表的长度 \*/

for(i=0,p1=head;i<len-1;i++,p1=p1->next)

for(j=i+1,p2=p1->next;j<len;j++,p2=p2->next)//进行选择排序

if(p1->sum>p2->sum)

{

t=p1->sum; p1->sum=p2->sum; p2->sum=t;

t=p1->average;p1->average=p2->average;p2->average=t;

for(int k=0;k<4;k++)

t=p1->score[k],p1->score[k]=p2->score[k],p2->score[k]=t;

strcpy(temp,p1->num),strcpy(p1->num,p2->num),strcpy(p2->num,temp);

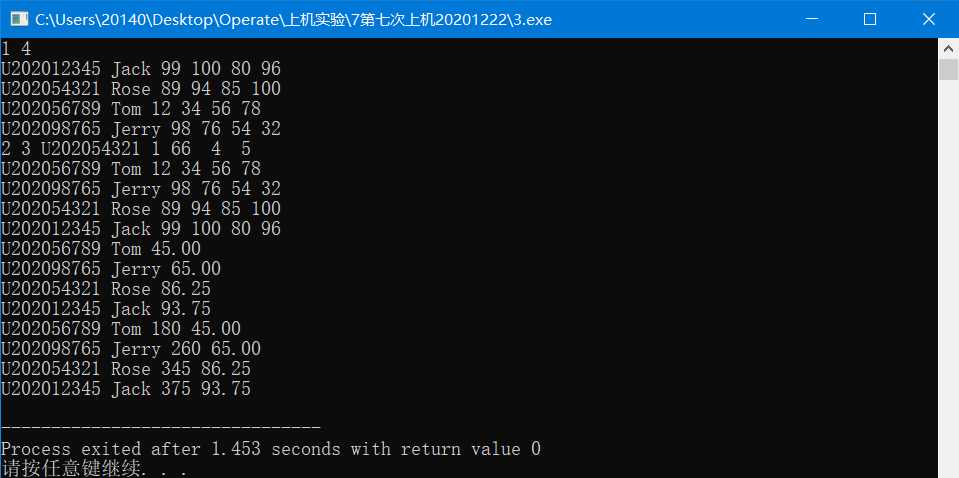
strcpy(temp,p1->name),strcpy(p1->name,p2->name),strcpy(p2->name,temp);

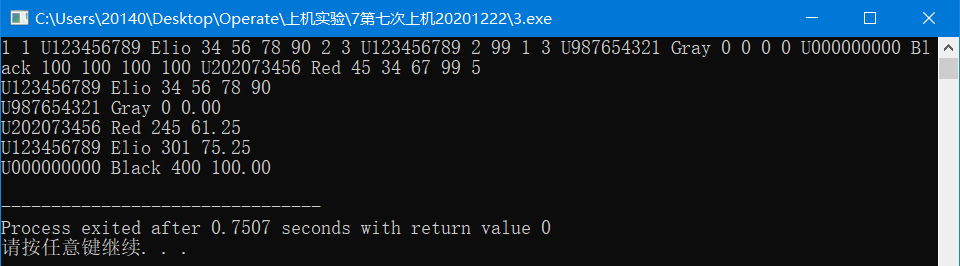
/\* 交换数据域中的各项内容 \*/

} }

3）测试结果

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出（均见测试截图） |
| 1 | 1 4  U202012345 Jack 99 100 80 96  U202054321 Rose 89 94 85 100  U202056789 Tom 12 34 56 78  U202098765 Jerry 98 76 54 32  2 3 U202054321 1 66 4 5 |  |
| 2 | 1 1 U123456789 Elio 34 56 78 90  2 3 U123456789 2 99 1 3  U987654321 Gray 0 0 0 0  U000000000 Black 100 100 100 100  U202073456 Red 45 34 67 99 5 |  |
| 3 | 1 2 U202012345 Jack 99 100 80 96 U202054321 Rose 89 94 85 100 2 3 U202054321 1 66 4 5 |  |





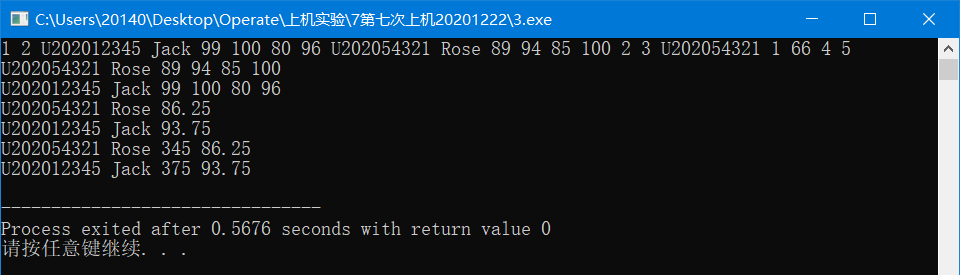


图3-3程序设计题3的运行结果

4.链表判断回文串

1）实验思路

利用scanf("%[^\n]",s);接收一段很长的字符串，回车后结束输入，利用该字符串构建链表，调用函数判断是否回文，判断方式是通过函数position将指针定位到相应节点再比较数据域。

2）源代码

#include "stdio.h"

#include "string.h"

#include<stdlib.h>

typedef struct c\_node{

char data;

struct c\_node \*next;

} C\_NODE;

void createLinkC\_NODE(C\_NODE \*\*headp, char s[]) ;

void judgePalindrome(C\_NODE \*head);

int main()

{

char s[1000],\*pc=s;

int len=0;

C\_NODE \*head,\*p;

scanf("%[^\n]",s);

createLinkC\_NODE(&head,s);

ad;

// while(p)

// {

// printf("%c",p->data);

// p=p->next;

// }

for(p=head;p;p=p->next) len++;

if (len!=strlen(s)) { printf("单链表长度不正确");return 1;}

else for(p=head;p;p=p->next)

if (p->data!=\*pc++) { printf("单链表有错误结点");return 1;}

judgePalindrome(head);

return 1;

}

/\*void createLinkC\_NODE(C\_NODE \*\*headp, char s[])

根据字符串s构造单链表，得到一个以单链表表示的字符串，单链表结点定义见右侧编辑器。

void judgePalindrome(C\_NODE \*head) 判断以head为头指针的单链表表示的字符串是否为回文，

是则显示true，否则显示false。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char position(C\_NODE \*\*pHead,int n)

{

C\_NODE \*p=\*pHead;

int count1=0;

while(p)

{

count1++;

if(count1==n) return p->data;

p=p->next;

}

}

void createLinkList(C\_NODE \*\*headp, char s[])

{

\*headp=NULL;

for(int i=0;i<strlen(s);i++)

{

C\_NODE \*pNew=(C\_NODE \*)malloc(sizeof(C\_NODE));

C\_NODE \*loc\_head=\*headp;

pNew->data=s[i];

pNew->next=NULL;

if(\*headp==NULL)

\*headp=pNew;

else

{

C\_NODE \*loc\_head=\*headp;

while(loc\_head->next)

loc\_head=loc\_head->next;

loc\_head->next=pNew;

}

}

}

void judgePalindrome(C\_NODE \*head)

{

C\_NODE \*p=head;

int count=0,flag=1;

while(p) {count++;p=p->next;}

for(int i=0;i<(count/2);i++)

{

if(position(&head,i+1)==position(&head,count-i)) continue;

else {

flag=0;

break;

}

}

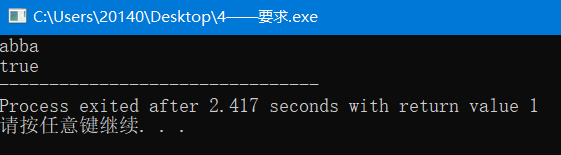
if(flag) printf("true");

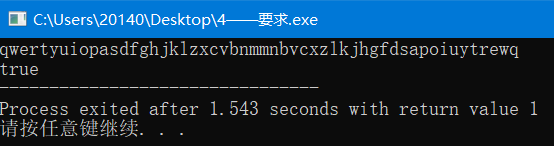
else printf("false");

}

3）测试结果

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出 |
| 1 | abba | true |
| 2 | qwertyuiopasdfghjklzxcvbnmmnbvcxzlkjhgfdsapoiuytrewq | true |
| 3 | Qwertyuiolmnbvcsaertyuijhertyujbv  drtyunbdtyujbvuiookjhwsxcvb | false |





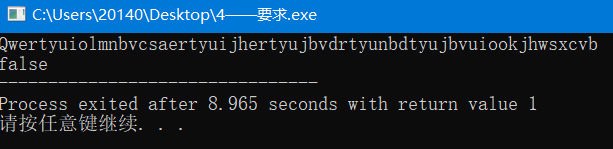
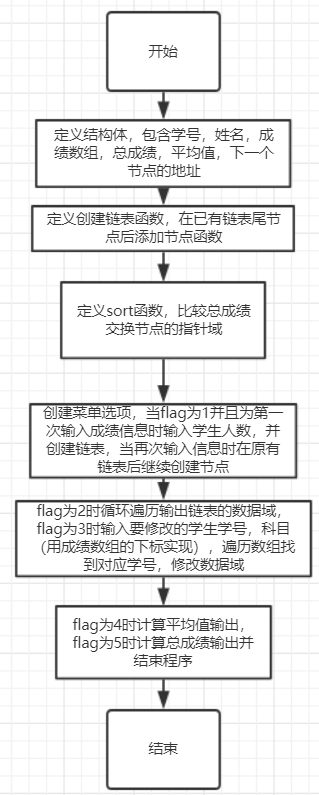


图3-4程序设计题4的运行结果

5. 对成绩用交换指针域的方法排序

1）实验思路：



2）源代码（仅对有关排序的代码进行注释）：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct s\_list

{

char num[20];

char name[10];

int score[4];

int sum;

float average;

struct s\_list \*next;

}list;

void create\_list(list \*\*headp,int number);

void AddToTail(list \*\* pHead);

void sort\_lists(list \*\*headp,int number);

int main(void)

{

list \*head=NULL,\*p,\*before;

int number,flag=1,check\_count=0,check\_new,count=1;

char check\_num[20];

while(flag!=5)

{

scanf("%d",&flag);

if(flag==1)

{

scanf("%d",&number);

if(count!=1)

{

for(int i=0;i<number;i++)

AddToTail(&head);

}

else create\_list(&head,number);

p=head;

while(p)

{

p->sum=p->score[0]+p->score[1]+p->score[2]+p->score[3];

p->average=(float)((p->sum)/4.0);

p=p->next;

}

sort\_lists(&head,number);//传递头指针的地址和输入数据的个数

count++;

}

if(flag==2)

{

p=head;

while(p){

printf("%s %s ",p->num,p->name);

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

{

printf("%d",p->score[i]);

if(i<3) printf(" ");

}

printf("\n");

p=p->next;

}

}

if(flag==3)

{

scanf("%s%d%d",check\_num,&check\_count,&check\_new);

p=head;

while(p)

{

if(strcmp(p->num,check\_num)==0)

{

p->score[check\_count-1]=check\_new;

break;

}

else p=p->next;

}

p=head;

while(p)

{

p->sum=p->score[0]+p->score[1]+p->score[2]+p->score[3];

p->average=(float)((p->sum)/4.0);

p=p->next;

}

}

if(flag==4)

{

p=head;

while(p)printf("%s %s %.2f\n",p->num,p->name,p->average),p=p->next;

}

if(flag==5)

{

p=head;

while(p)

printf("%s %s %d %.2f\n",p->num,p->name,p->sum,p->average),p=p->next;

return 0;

} } }

void create\_list(list \*\*headp,int number)

{ list \*loc\_head=NULL,\*tail;

char num0[20];

char name0[10];

int score0[4];//English,math,physics,C

scanf("%s%s%d%d%d%d",num0,name0,&score0[0],&score0[1],&score0[2],&score0[3]);

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

strcpy(loc\_head->num,num0);

strcpy(loc\_head->name,name0);

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

loc\_head->score[i]=score0[i];

tail=loc\_head;

for(int i=1;i<number;i++){

scanf("%s%s%d%d%d%d",num0,name0,&score0[0],&score0[1],&score0[2],&score0[3]);

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

tail=tail->next;

strcpy(tail->num,num0);

strcpy(tail->name,name0);

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

tail->score[i]=score0[i];

}

tail->next=NULL;

\*headp=loc\_head;

}

void AddToTail(list \*\*pHead)

{

list \* pNew=(list \*)malloc(sizeof(struct s\_list));

char num0[20];

char name0[10];

int score0[4];//English,math,physics,C

scanf("%s%s%d%d%d%d",num0,name0,&score0[0],&score0[1],&score0[2],&score0[3]);

strcpy(pNew->num,num0);

strcpy(pNew->name,name0);

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

pNew->score[i]=score0[i];

pNew->next=NULL;

if(\*pHead==NULL)

\*pHead=pNew;

else

{

list \*pNode=\*pHead;

while(pNode->next != NULL)

pNode=pNode->next;

pNode->next=pNew;

}

}

void sort\_lists(list \*\*headp,int number)

{ struct s\_list \*prior1,\*prior2,\*p1,\*p2,\*t;

int i=0;

p1=\*headp;//此处用先单独处理链头的方法交换指针域

if(number==1) return;//当仅有一个输入信息时，无需排序，结束函数

else

{

for(p2=p1->next,prior2=p1;p2!=NULL;prior2=p2,p2=p2->next)

if(p1->sum>p2->sum){

prior2->next=p1; t=p1->next;

p1->next=p2->next; (\*headp)=p2;

p2->next=t; p1=p2;

}

for(prior1=\*headp,p1=prior1->next;p1->next!=NULL;prior1=p1,p1=p1->next)

for(p2=p1->next,prior2=p1;p2!=NULL;prior2=p2,p2=p2->next)

if(p1->sum>p2->sum){

t=p2->next; prior1->next=p2;

prior2->next=p1; p2->next=p1->next;

p1->next=t; p1=p2;

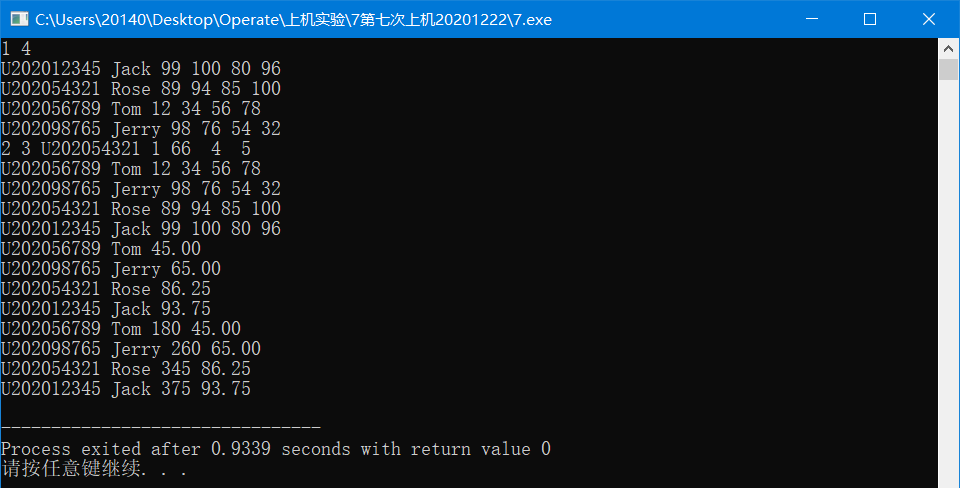
}

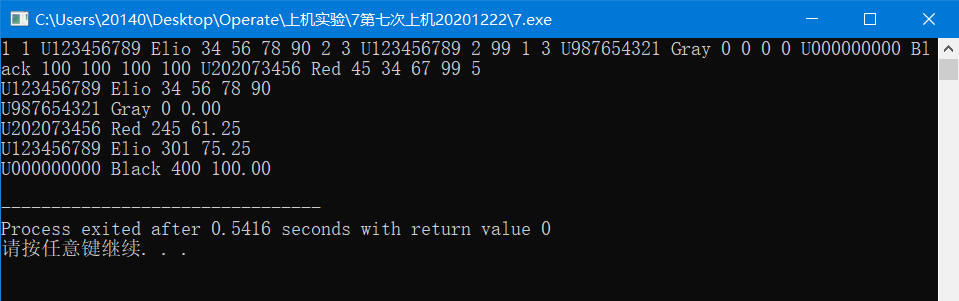
}

}

3）测试结果

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出（均见测试截图） |
| 1 | 1 4  U202012345 Jack 99 100 80 96  U202054321 Rose 89 94 85 100  U202056789 Tom 12 34 56 78  U202098765 Jerry 98 76 54 32  2 3 U202054321 1 66 4 5 |  |
| 2 | 1 1 U123456789 Elio 34 56 78 90 2 3 U123456789 2 99 1 3 U987654321 Gray 0 0 0 0 U000000000 Black 100 100 100 100 U202073456 Red 45 34 67 99 5 |  |
| 3 | 1 2 U202012345 Jack 99 100 80 96 U202054321 Rose 89 94 85 100 2 3 U202054321 1 66 4 5 |  |





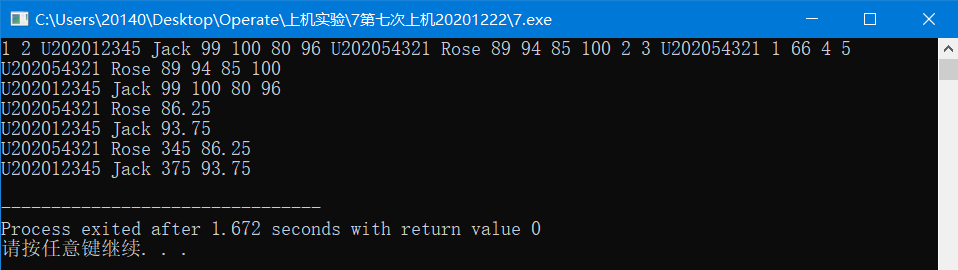


图3-5程序设计题5的运行结果

6.利用值栈对逆波兰表达式进行求值

1）实验思路

存储数据到后进先出链表中->遇到运算符时进行计算，把数据顶出栈，计算结果再存入栈中->得出最终结果

2）源代码：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct list {

int data;

struct list\* next;

}list;

typedef struct

{

list\* list1;

int scale;

list\* last;

}list1;

void add (list1\* list1,int x);

int positon (list1\* list1,int check);

list\* new\_position (list1\* list1,int check);

void new\_free(list1\* list1);

int main()

{

list1 list1= {NULL,0,NULL};

char c='a';

int flag=0;

int t=0;

while(c!=EOF&&c!='\n')

{

c=getchar();

if(c>='0'&&c<='9'){

t\*=10;

t+=c-'0';

flag=1;

}

else if(c==' '){

if(flag){

add(&list1,t);

t=0;

flag=0;

}

}

else if(c=='+')

{

int y=positon(&list1,1)+positon(&list1,0);

new\_free(&list1);

new\_free(&list1);

add (&list1,y);

}

else if(c=='-'){

if((c=getchar())==' '||c==EOF||c=='\n')

{

int y=positon(&list1,1)-positon(&list1,0);

new\_free(&list1);

new\_free(&list1);

add(&list1,y);

}

else

{

t\*=10;

t+=c-'0';

t\*=-1;

flag=1;

}

}

else if(c=='\*'){

int y=positon(&list1,1)\*positon(&list1,0);

new\_free(&list1);

new\_free(&list1);

add(&list1,y);

}

else if(c=='/'){

int y=positon(&list1,1)/positon(&list1,0);

new\_free(&list1);

new\_free(&list1);

add (&list1,y);

}

}

printf("%d",positon(&list1,0));

return 0;

}

void add (list1\* list1, int x)

{//在链表后添加节点

list1->scale++;

list\* np=(list\*) malloc(sizeof(list));

np->data=x;

if(list1->scale==1) list1->last=np;

else np->next=list1->list1;

list1->list1=np;

}

int positon (list1\* list1,int check)

{//定位节点所在处，返回数据域

list\* np=list1->list1;

for(int i=0;i<check;i++) np=np->next;

return (np->data);

}

list\* new\_position(list1\* list1,int check)

{//返回定位节点处的地址

list\*np=list1->list1;

for(int i=0;i<check;i++) np=np->next;

return np;

}

void new\_free(list1\* list1){

//释放节点

list\*temp=list1->list1->next;

free(list1->list1);

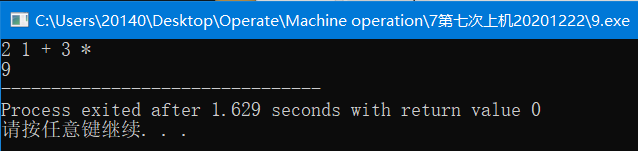
list1->list1=temp;

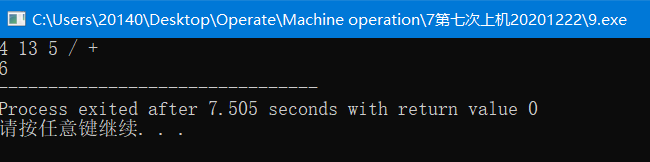
list1->scale--;

}

3）测试结果

|  |  |  |
| --- | --- | --- |
| 项目 | 测试输入 | 输出 |
| 1 | 2 1 + 3 \* | 9 |
| 2 | 4 13 5 / + | 6 |
| 3 | 22 13 / 34 + 78 \* 56 – 24 + | 2698 |





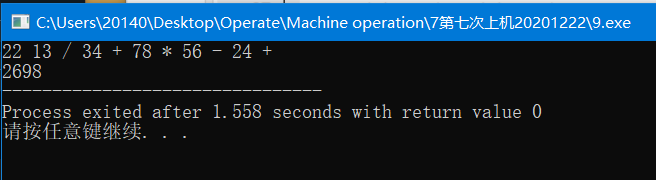


图3-6程序设计题6的运行结果

## 3.4小结

通过这次编程实践，我更加深入的了解和掌握了链表的使用。

在程序表达式验证中我掌握了指针和结构体的关系，在程序修改替换中我掌握了如何创建链表。

在程序设计第一题中，我认识到了联合体对于处理数据各字节的好处。同时又掌握了字段结构的应用。

在程序设计第二题中，由于有一种输入情况是增加成绩信息，而假如此时再次调用create函数创建链表时，之前已经录入的成绩信息会被覆盖，因此要使用addtotail函数在先前链表后添加节点，在这一点我遇到了问题，经过多次调试和输出后才解决问题。而在用两种方式进行排序时，我认识到了交换指针域排序的算法优越性，交换数据域的方法在链表节点很多或者节点数据域复杂时尤为不方便。

在处理回文数这一题时，我遇到了如何任意输入字符串的问题，在educoder上得到提示后才意识到要用单向链表存储，而之后用链表来判断回文串又遇到了麻烦，在网上查找资料后，采用了定位指针所指向的节点然后比较数据域的方法。

在最后一题中，我了解了逆波兰表达式（后缀表示法），并且对值栈有个更深一步的了解，但是在创建后进后出链表处理时却总漏洞百出。而且在处理多个运算符的运算符时不清楚算法。之后查阅资料了解到要将数据顶出栈，将计算结果存入栈。

# 参考文献

[1] 曹计昌,卢萍,李开. C语言与程序设计,北京：电子工业出版社,2013

[2] 卢萍,李开,王多强，甘早斌. C语言程序设计典型题解与实验指导,北京：清华大学出版社,2019

......