

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： CS2003**

**学 号： U202015359**

**姓 名： 张庙松**

**指导教师： 卢萍**

**报告日期： 2021.1.2**

**计算机科学与技术学院**

目 录

**[1流程控制实验 1](#_1_流程控制实验)**

[1.1程序改错 1](#_程序改错)

[1.2程序修改替换 2](#_1.2_程序修改替换)

[1.3程序设计 6](#_1.3程序设计)

[1.4小结 24](#_1.4_小结)

**[2数组程序设计实验 27](#_2数组程序设计实验)**

[2.1程序改错与跟踪调试 27](#_2.1_程序改错与跟踪调试)

[2.2程序完善与修改替换 28](#_2.2_程序完善与修改替换)

[2.3程序设计 33](#_2.3_程序设计)

[2.4小结 65](#_2.4_小结)

[**3结构与联合实验 64**](#_3结构与联合实验)

[3.1表达式求值的程序验证 64](#_3.1表达式求值的程序验证)

[3.2源程序修改替换 64](#_3._2源程序修改替换)

[3.3程序设计 66](#_3.3程序设计)

[3.4小结 95](#_3.4小节)

[**参考文献 95**](#_参考文献)

# 

# 1 流程控制实验

## 程序改错

原程序如下：

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main()

{

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;i<=k;i++) 只要对x开方即可，而不是对x/2

if(!x%i){ 运算符优先级

flag=1;

break;

}

if(flag=1) if语句的判断形式

printf("%d是合数\n",x);

else printf("%d不是合数\n",x);

}

return 0;

}

修改过后的程序入下：

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main()

{

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=sqrt(x);i<k;i++) 只要对x开方即可，而不是对x/2

if(!(x%i)){ 运算符优先级

flag=1;

break;

}

if(flag==1) if语句的判断形式

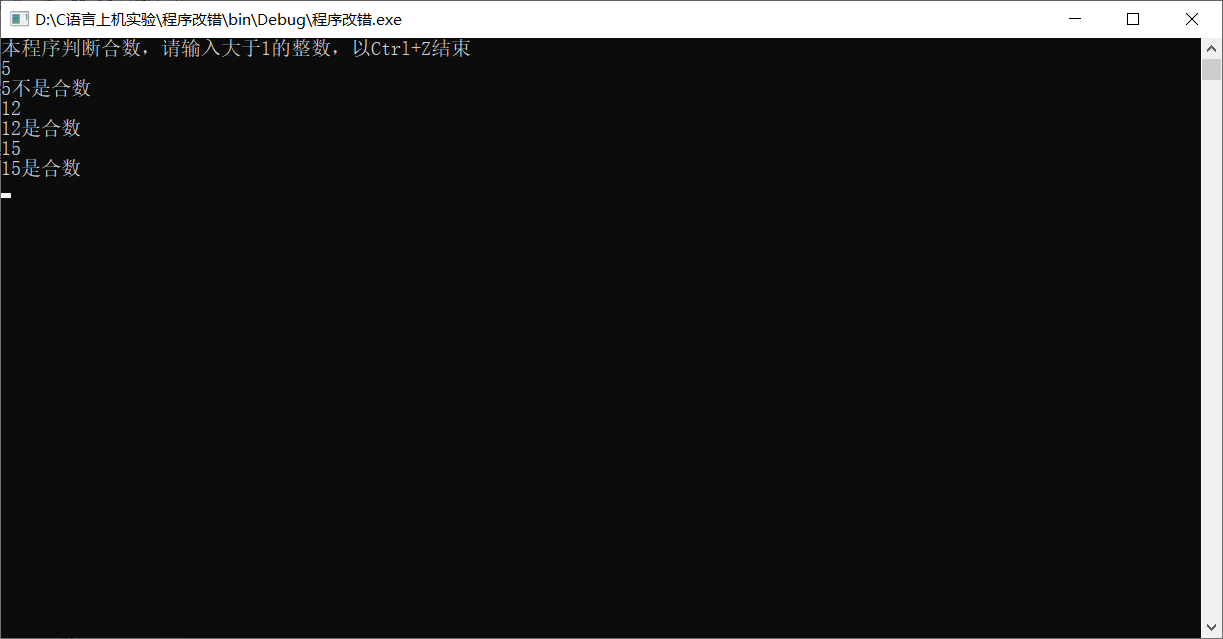
printf("%d是合数\n",x);

else printf("%d不是合数\n",x);

}

return 0;

}

图1.1程序改错题结果如图所示

## 1.2 程序修改替换

方法分析：原程序利用do-while语句循环历遍所有的三位数，利用合数判断方法进行判断

替换：利用dec函数将合数的判断进行简化

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int dec(int n)

{

int flag=0,k,i=2;

k=sqrt(n);

if(n!=2){

do{

if(n%i==0)

flag=1;

i++;

}while(i<k);

}

return flag;

}

int main()

{

int n=100,flag,a;

while(n<1000){

flag=dec(n);

if(flag==1)

{

a=n/10;

flag=dec(a);

if(flag==1){

a=a/10;

flag=dec(a);

if(flag==1){

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

}

}

}

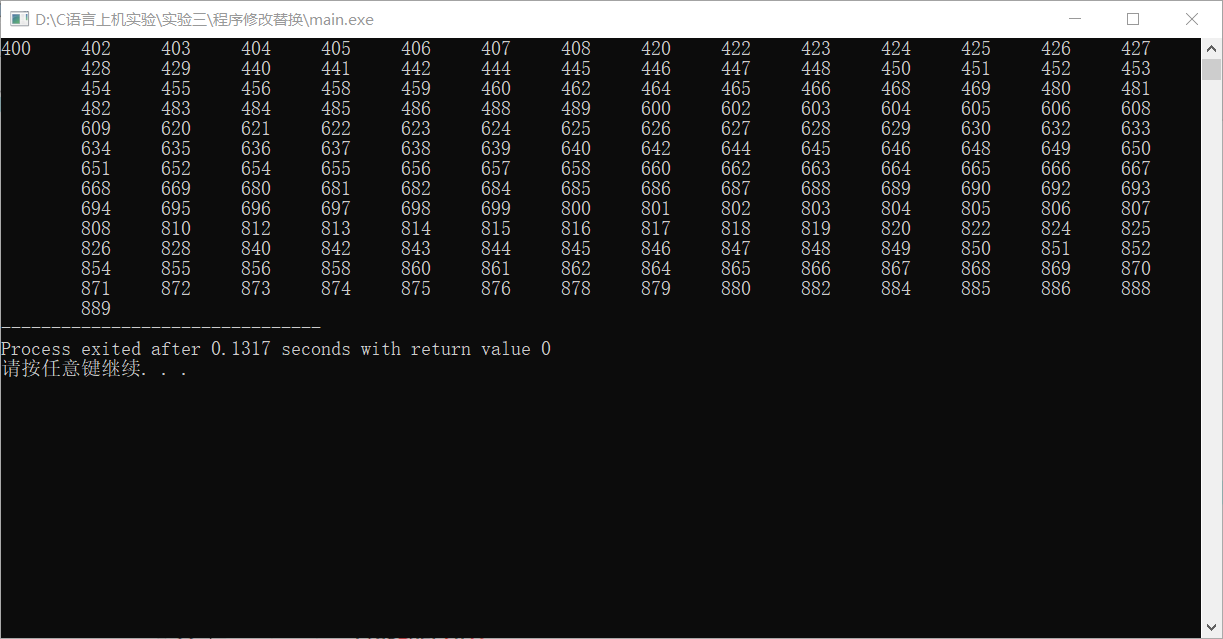
n++;

}

return 0;

}

图1.2程序修改替换结果如图所示



## 1.3程序设计

1

分析，（1）对于输入的工资，利用整除判断所在区间，由所在区间决定所占的税收

流程图1.1编程题一的流程草图



#include <stdio.h>

#include <stdlib.h>

int main()

{

int salary,x;

double tax;

printf("请输入工资");

scanf("%d",&salary);

x=salary/1000;//判断工资的区间

if(x<=4)

{

switch(x)//税收判断

{

case 0:

tax=0;

break;

case 1:

tax=(salary-1000)\*0.05;

break;

case 2:

tax=(salary-2000)\*0.1+50;

break;

case 3:

tax=(salary-3000)\*0.15+150;

break;

case 4:

tax=(salary-4000)\*0.2+300;

break;

}

}

else tax=(salary-5000)\*0.25+500;

printf("tax=%lf",tax);

return 0;

}

#include <stdio.h>

#include <stdlib.h>

int main()

{

int salary,x;

double tax;

printf("请输入工资");

scanf("%d",&salary);

x=salary/1000; //判断工资的区间

if(x==0) //税收判断

tax=0;

else if(x==1)

tax=(salary-1000)\*0.05;

else if(x==2)

tax=(salary-2000)\*0.1+50;

else if(x==3)

tax=(salary-3000)\*0.15+150;

else if(x==4)

tax=(salary-4000)\*0.2+300;

else if(x>=5)

tax=(salary-5000)\*0.25+500;

printf("tax=%lf",tax);

return 0;

}

表1.2编程题一的测试数据

|  |  |  |
| --- | --- | --- |
| 测试输入 | 程序输入 | 预期结果 |
| 例一 | 900 | 0 |
| 例二 | 2500 | 100 |
| 例三 | 6000 | 750 |

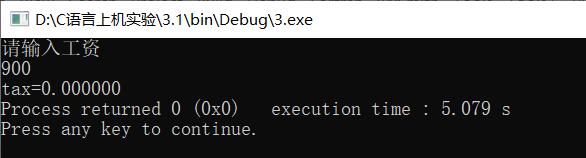
表1.2中测试用例的运行结果如图1.3、图1.4和图1.5所示

图1.3编程题一例一的运行结果

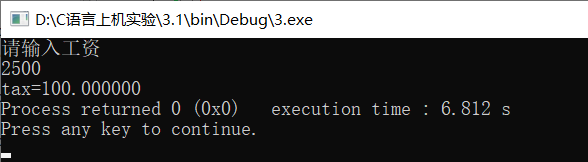


图1.4编程题一例二的运行结果

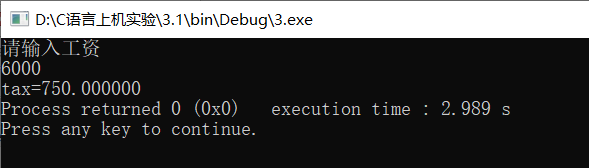


图1.5编程题一的例三的运行结果

2解题思路：对于符合条件的三位数，进行平方后取余，对于原三位数比较，通过循环完成。

流程图1.6编程题二的流程图



设计思路：对于所有三位数，设计循环，将其平方，平方数对1000取余，并判断余数是否与该三位数相同

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n=100,k;

for(n=100; n<1000; n++)

{

k=n\*n;

k=k%1000;

if(k==n)

{

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

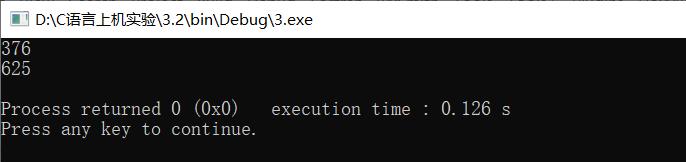
}

}

return 0;

}

图1.7编程题二的运行结果



3

建立数组，存储n天的数据，对于数据进行逐差，由题目所给的条件，采取函数的方法比较最大值，利用函数选出最大值

流程图1.8编程题三的运行结果



#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int max(int x,int y)

{

int z;

if(x>y)

z=x;

else

z=y;

return z;

}

int main()

{

int n;

long price[100],s[100],b,c,d;

printf("请输入天数\n");

scanf("%d",&n);

int i=0;

printf("请依次输入%d天的价格\n",n);

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

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

int a=0;

for(a=0; a<n-2; a++)

{

b=abs(price[a+1]-price[a]);

c=abs(price[a+2]-price[a+1]);//逐差

d=max(b,c);//取最大值

}

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

return 0;

}

图1.9编程题3的测试数据

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 测试用例 | 测试输入 | | | | | | 预期结果 |
| n | Day1 | Day2 | Day3 | Day4 | Day5 |
| 用例一 | 5 | 123 | 421 | 2115 | 651 | 122 | 1464 |
| 用例二 | 4 | 125 | 154 | 542 | 222 |  | 388 |
| 用例三 | 3 | 875 | 845 | 855 |  |  | 30 |

表1.9中测试用例的运行结果如图1.10、图1.11和图1.12所示

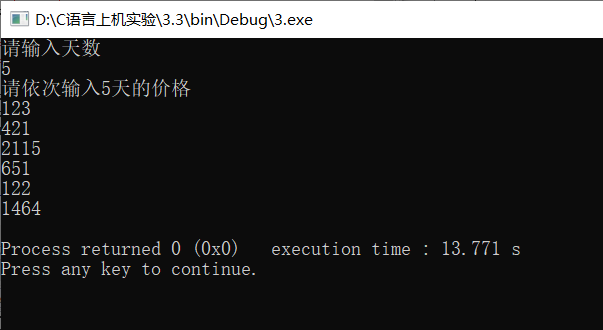


图1.10编程题三的用例一运行结果

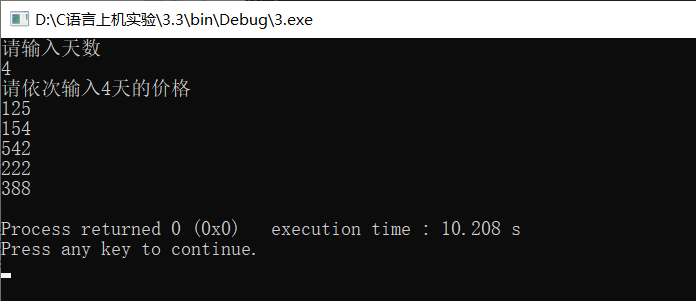


图1.11编程三用例二的运行结果

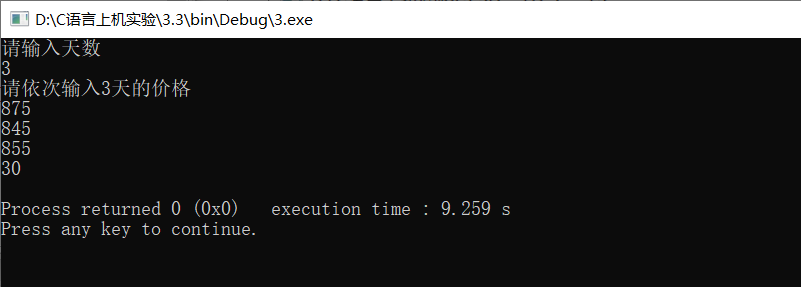


图1.12编程题三用例三的运行结果

4

解题思路：对于输入的字符开始循环，当遇到正常正文时，计数器清零，当遇到空格时，计数器+1，判断当计数器小于等于1时，输出字符。

流程图1.13



#include <stdio.h>

#include <stdlib.h>

int main()

{

char c;

int n=0;

printf("请输入正文\n");

while((c=getchar())!=EOF)

{

if(c!=' ')

n=0;

else n++;

if(n<=1)

putchar(c);

}

return 0;

}

图1.14编程题四的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期结果 |
| 用例一 | dfjlakj ajdfioaw slkdvj | dfjlakj ajdfioaw slkdvj |
| 用例二 | aldsj askdj asdk | aldsj askdj asdk |
| 用例三 | sldkfj sakldj aslkd | sldkfj sakldj aslkd |

图1.14编程四的三个用例的运行结果如下图1.15所示

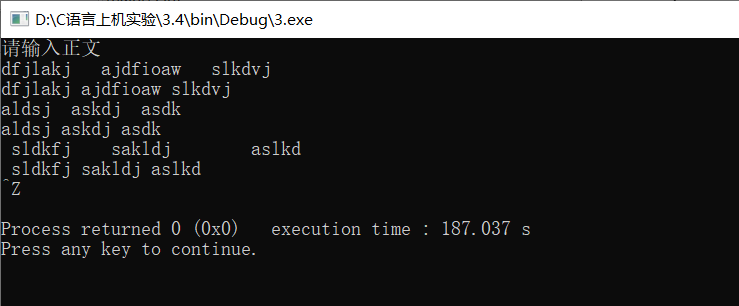


图1.15编程题四的三个用例结果

5

解题思路：观察该题，发现每行每列都有顺序及规律，即可以应用循环的，利用循环打印出行，利用循环打印出开头空格，利用循环以及递归打印每行的数，利用循环来换行

算法步骤：声明i，j，l，an

i用来控制行，对i<10进行循环

用l来输出空格，初始化l=40-2\*I,当l>1时，进行循环，输出空格

用j来表示列的下标当j<i+1时循环

当j=1时，表示第一列，输出1

当j!=1时，由公式，递归计算

缩进4格输出an

每一行结束输出换行符

#include <stdio.h>

#include <stdlib.h>

int main()

{

int i=0,j=1,l,an;

for(i=0; i<10; i++)//每行的控制

{

for(l=40-2\*i; l>1; l=l-2)//输出空格

{

printf(" ");

}

for(j=0; j<i+1; j++)//每列的控制

{

if(j==0)

an=1;

else

an=an\*(i-j+1)/j;

printf("%-4d",an);

}

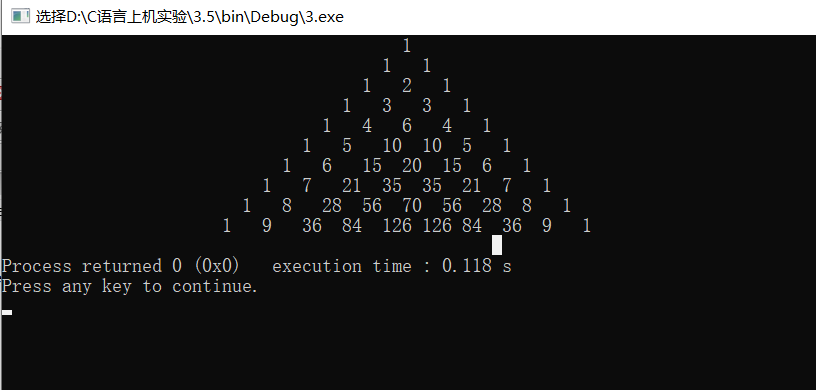
printf("\n");

}

return 0;

}

图1.16编程题五的运行结果



6

解题思路：观察该题，发现要进行次方运算和素数判断，利用函数来进行次方运算和素数判断，利用循环输出梅森素数

算法步骤：声明正整数m并输入m

声明i,a,b

初始化i=2，对于a=-1（利用函数实现）与m比大小，小于m时开始循环

判断此时的a是否为素数（利用函数实现）

如果是素数，而按格式输出

如果不是素数，则返回i的循环

#include <stdio.h>

#include <stdlib.h>

long cifang(int a, int b)

{

int c=1,i=0;

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

{

c=c\*a;

}

return c;

}

int dec(int n)

{

int flag=0,k,i=2;

k=n;

if(n!=2)

{

do

{

if(n%i==0)

flag=1;

i++;

}

while(i<k);

}

return flag;

}

int main()

{

long m;

printf("请输入正整数m\n");

scanf("%ld",&m);

int i,a,b;

for(i=2; (a=cifang(2,i)-1)<m; i++)

{

a=cifang(2,i)-1;

b=dec(a);//判断是否为素数

if(b==0)

{

printf("M(%d)=%d\n",i,a);

}

}

printf("");

return 0;

}

表1.17编程题六的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入m | 预期结果 |
| 用例一 | 97 | M(2)=3  M(3)=7  M(5)=31 |
| 用例二 | 53121 | M(2)=3  M(3)=7  M(5)=31  M(7)=127  M(13)=8191 |
| 用例三 | 427527527 | M(2)=3  M(3)=7  M(5)=31  M(7)=127  M(13)=8191  M(17)=131071  M(19)=524287 |

表1.17中测试用例的运行结果如图1.18、图1.19和图1.20所示

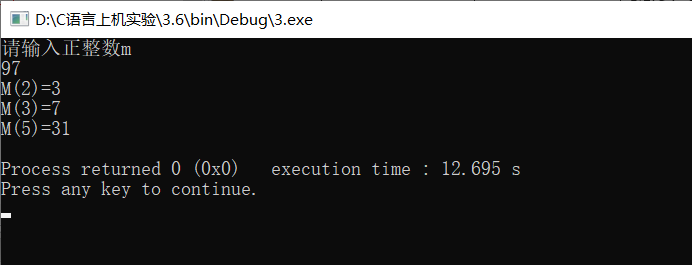


图1.18编程题六的用例一的运行结果

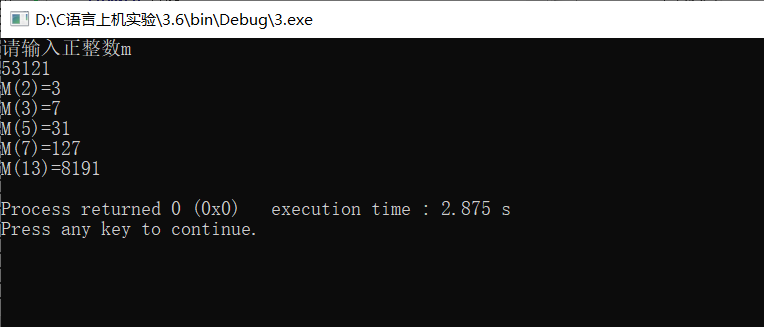


图1.19编程题六的用例二的运行结果

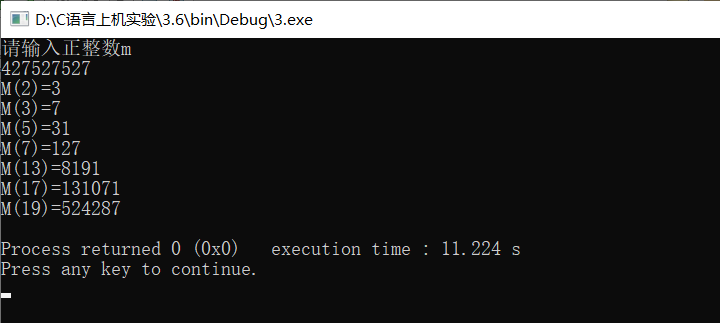


图1.20编程题六的用例三的运行结果

10

对于输入的m，利用筛选法将1000000以内的孪生素数都选出

声明数组b[1000000]

声明m并输入m

利用i=0,i<1000000,i++的循环使得b数组的每一个元素均为1

声明a，当a<1000000时开始循环，筛掉各个数的倍数

循环判断相隔2个数的数是否为孪生素数，是则输出

#include<stdio.h>

#include<stdlib.h>

int b[1000000];

int main()

{

long int m;

printf("请输入m\n");

scanf("%ld",&m);

int i;

for(i=0; i<1000000; i++)

{

b[i]=1;

}

int a=2;

while(a<1000000)

{

if(b[a])

{

for(i=2; i\*a<1000000; i++)

{

b[i\*a]=0;

}

}

a++;

}//筛法

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

{

if(b[i]&&b[i+2])

{

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

}

}

return 0;

}

表1.21编程题十的测试数据

|  |  |  |
| --- | --- | --- |
| 测试输入 | 程序输入m | 预期结果 |
| 用例一 | 84 | (3,5)  (5,7)  (11,13)  (17,19)  (29,31)  (41,43)  (59,61)  (71,73) |
| 用例二 | 154 | (3,5)  (5,7)  (11,13)  (17,19)  (29,31)  (41,43)  (59,61)  (71,73)  (101,103)  (107,109)  (137,139)  (149,151) |
| 用例三 | 21 | (3,5)  (5,7)  (11,13)  (17,19) |

表1.21中测试用例的运行结果如图1.22、图1.23图1..24所示

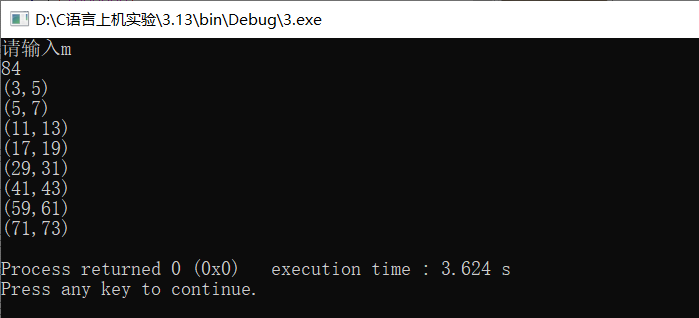


图1.22编程题十用例一的运行结果

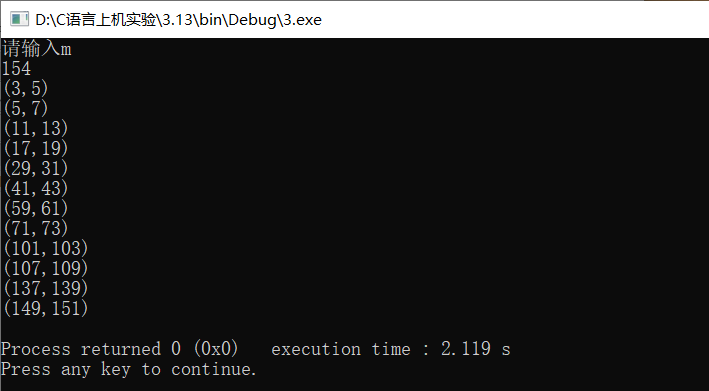


图1.23编程题十用例二的运行结果

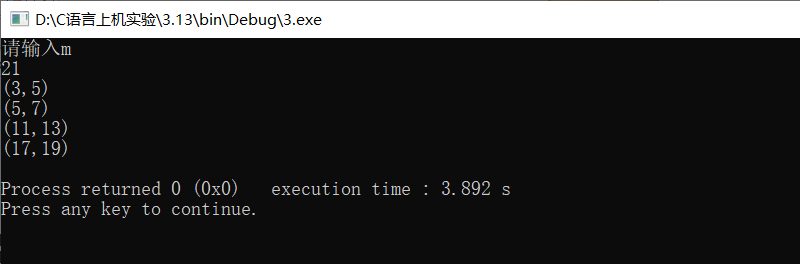


图1.24编程题十用例二的运行结果

11

解题思路：输入符合条件的数，由题意，除数，被除数，结果都是不同的字符，利用循环，由大到小进行排除，创建判断函数，对于除数与被除数进行判断，找出所有符合形式的表达式。

算法步骤：声明m,n,i并输入i

对于n=1234（符合条件的最小数）n<98765(符合条件的最大数)n++  
m=i\*n

当m大于五位数时，重新开始循环

否则，利用函数进行判断（函数的作用是判断是否有重复的数字）

符合则输出

#include<stdio.h>

int jug(int m, int n)

{

int ret=1;

int i=0;

int b;

char a[10];

while(m)

{

a[i++]=m%10;

m/=10;

}

while(n)

{

a[i++]=n%10;

n/=10;//将每一个出现的数存入数组

}

if(i==9)a[9]=0;

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

{

for(b=i+1; b<10&&ret; b++)

{

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

ret=0;//判断是否有重复

}

}

return ret;

}

int main()

{

int m,n,i;

printf("请输入2~79的值：\n");

scanf("%d",&i);

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

{

m=n\*i;

if(m>1e5)

break;

else if(jug(m,n))

printf("%d/%05d=%d\n",m,n,i);

}

return 0;

}  
表1.24编程题11的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入x | 预期结果 |
| 用例一 | 32 | 75168/02349=32 |
| 用例二 | 15 | 27945/01863=15  92745/06183=15 |
| 用例三 | 35 | 48265/01379=35  63945/01827=35  64295/01837=35  74865/02139=35  93485/02671=35 |

表1.24中测试用例的运行结果如图1.25、图1.26和图1.27所示

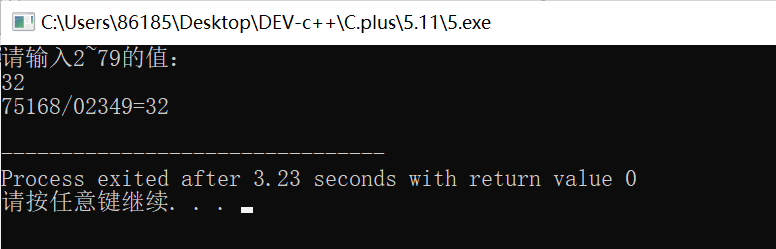


图1.25编程题11用例一的运行结果

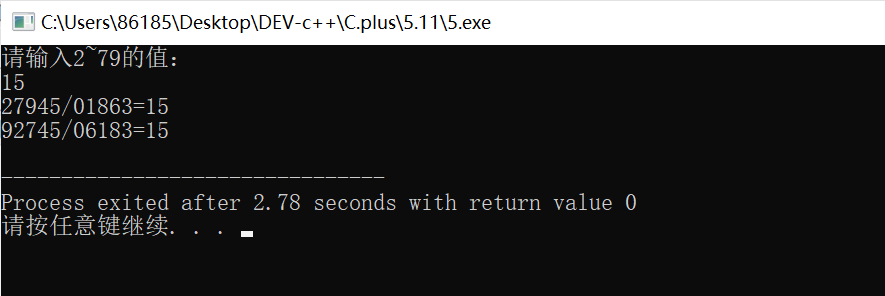


图1.26编程题11用例二的运行结果

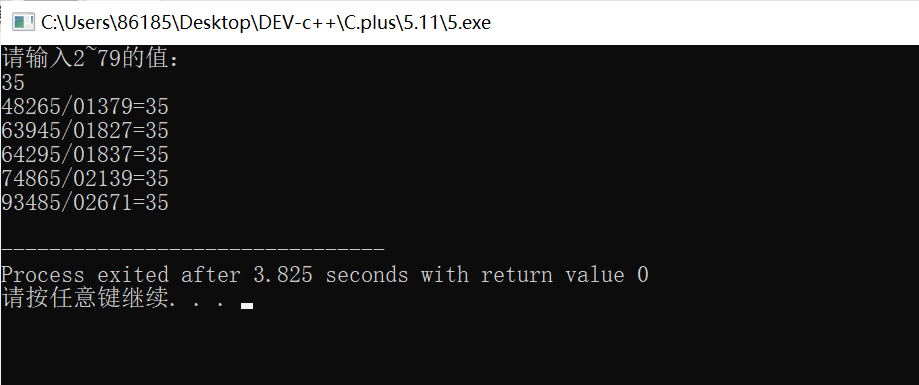


图1.27编程题11用例三的运行结果

## 1.4 小结

通过本次实验，，了解了程序执行时的流程控制可以用顺序结构，分支结构，循环结构来表示，对于表达式语句，复合语句，标号语句，选择语句，循环语句，转移语句有初步了解，初步了解了循环编程的使用和改进，在部分题目中发现的漏洞补上，有较大收获。但是对于编程及流程控制依然很不熟练，多次寻求老师和同学的帮助，难以独立完成。以后仍要多多练习。

# 

# 2数组程序设计实验

## 2.1 程序改错与跟踪调试

原程序如下：

#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'); 引用声明的字符，不是作为字符常量

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

return 0;

}

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

{

int i = 0, j = 0;

while (t[i++]);

while ((t[i++] = s[j++] != '0'));字符数组的结束判断错误

}

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];

缺少作为字符数组结束的标志

}

修改后的程序如下：

#include<stdio.h>

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

void strdelc(char[], char c);//声明字符

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');//作为字符

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

return 0;

}

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

{

int i = 0, j = 0;

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

while ((t[i++] = s[j++]));//字符数组结束的标志

}

void strdelc(char s[], char c)

{

int j=0, k=0;

while(s[j])

{

if (s[j] != c)

s[k++] = s[j];

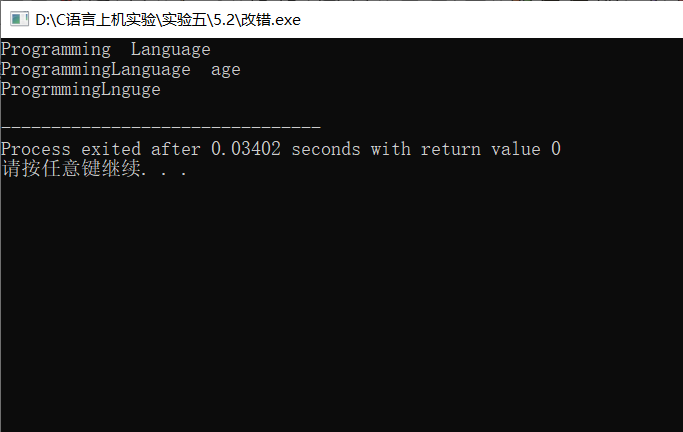
j++;

}

s[k] = '\0';

}

图2.1程序改错的结果



## 2.2 程序完善与修改替换

1

完善后的程序如下：

#include<stdio.h>

#include<string.h>

void RemoveDuplicate(char \*s);

int main()

{

char str[200];

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]!='\0')

{

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

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

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

s[i]='\0';

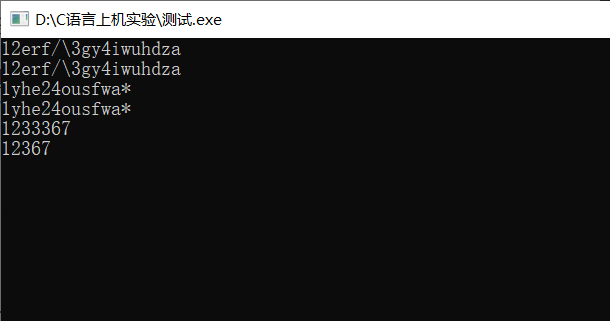
}

}

s[w]='\0';

}

图2.2程序完善题1-1的结果如图



修改程序

思路：原程序将重复的字符进行删除并且采用多层循环，现采用标记法，确定一个新的数组存放256个值，初始为0，当遇到字符时，对应元素+1，输出时，输出小于2时，输出改字符

修改后的程序如下：

#include<stdio.h>

#include<string.h>

void RemoveDuplicate(char\* s);

int main()

{

char str[200];

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);

char temp[256];

for(r=0;r<256;r++)

temp[r]=0;

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

{

temp[s[r]-1]++;

if (temp[s[r]-1] <2)

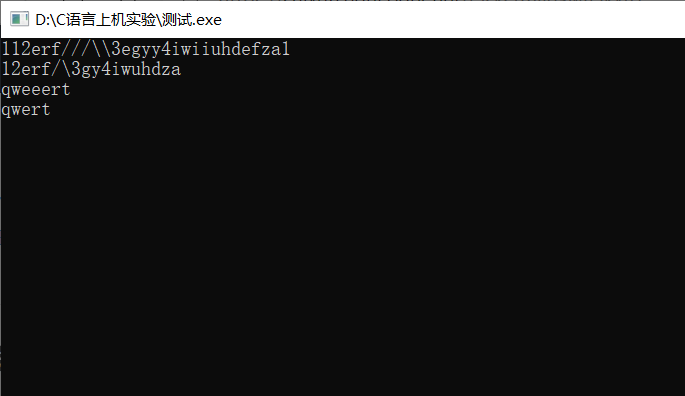
s[w++] = s[r];

}

s[w] = '\0';

}

图2.3程序完善题1-2的测试结果



2

完善后的程序如下：

# include<stdio.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];

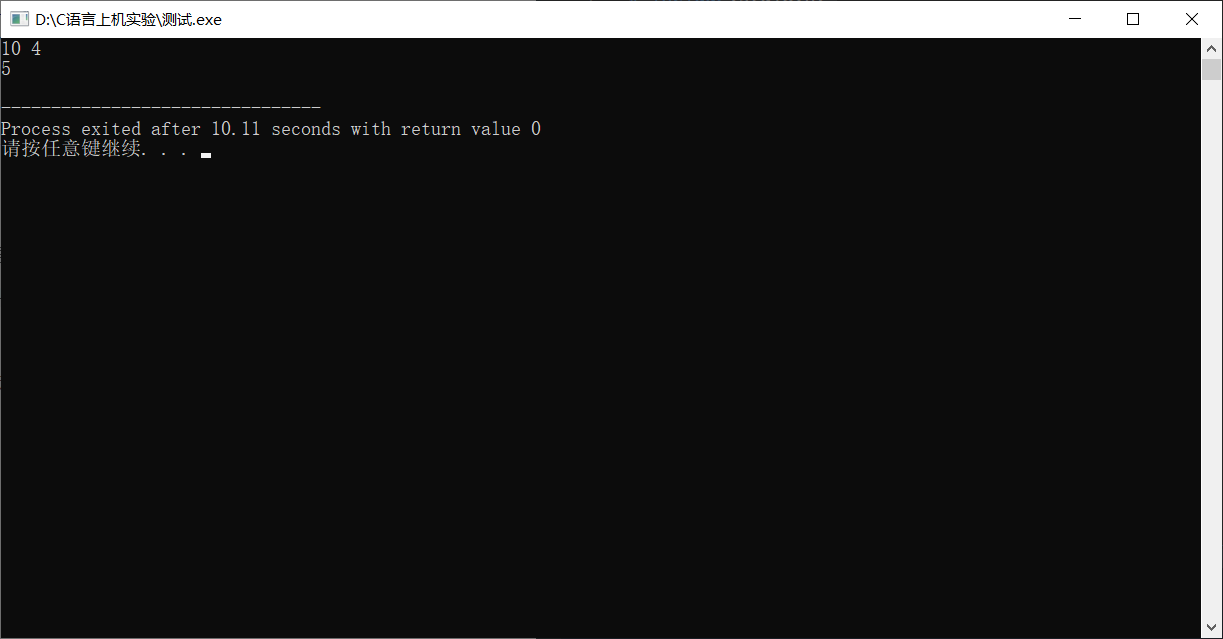
}

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

return 0;

}

图2.4程序完善题2-1的测试结果



改进程序：

思路：原程序采用压缩数组法，现采用标记处理法，对于刚开始的人，进行初始化为0，每一次报数到的时候变为1，下一次报数不再历遍值为1的元素，最后输出值为0的元素。

修改后的程序如下：

#include <stdio.h>

#include <stdlib.h>

int main()

{

int M, N;

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

int a[M];

int i=0;

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

a[i]=0;

int k=0,count=1,l=M;

i=0;

for(l=M;l>1;)

{

if (count==N)

{

k=(i)%M;

a[k]=1;

l--;

count=0;//因为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，故此处应将count置为0，带下标i遍历至下一个未出列元素时count再加1

}

i++;

if (a[i%M]!=1)

count++;

}

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

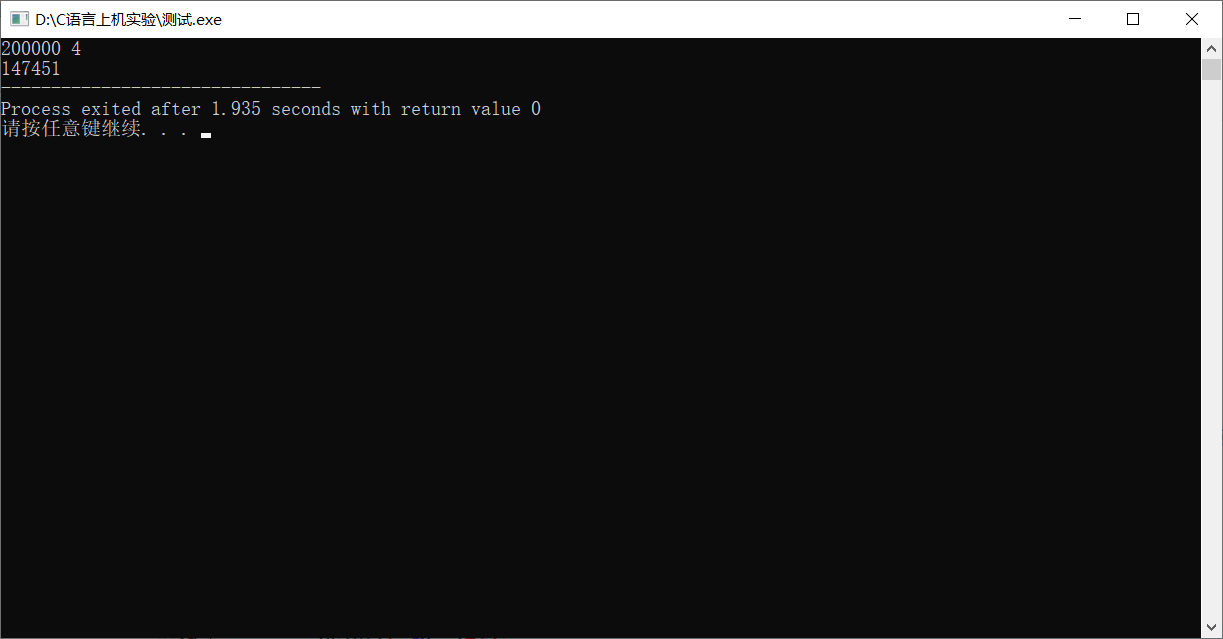
if(!a[i])

printf("%d",i+1);

}

}

图2.5程序完善题2-2的测试结果



## 2.3 程序设计

1

思路：运用字符数组来存储二进制位，对于输入的整数而言，进行二进制的转化，并将每一位都存储到数组中，对于存在的负数而言，利用强制转化，将负数转化为其在内存中表示的补码的十进制转化。

图2.6编程题一的流程图



源程序如下：

#include<stdio.h>

#include<string.h>

void reverse(char a[]);

int main()

{

unsigned int n;

int i=0;

char a[32];

for(i=0;i<32;i++)

{

a[i]='0';

}

scanf("%d", &n);

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

a[i] = n % 2+'0';

n /= 2;

}

reverse(a);

for (i = 0; i<32;i++){

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

if((i+1)%4==0)

printf(" ");

}

return 0;

}

void reverse(char a[])

{

int len =32;

int c, j, k;

for (j = 0, k =31;j < k;j++, k--)

c = a[j], a[j] = a[k], a[k] = c;

}

表2.7编程题一的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 程序输入 | 预期结果 |
| 用例一 | -1 | 1111 1111 1111 1111 1111 1111 1111 1111 |
| 用例二 | 123 | 0000 0000 0000 0000 0000 0000 0111 1011 |
| 用例三 | 2147483647 | 0111 1111 1111 1111 1111 1111 1111 1111 |

表2.7中的测试用例如下图2.8 2.9 2.10所示



图2.8编程题一用例一的运行结果

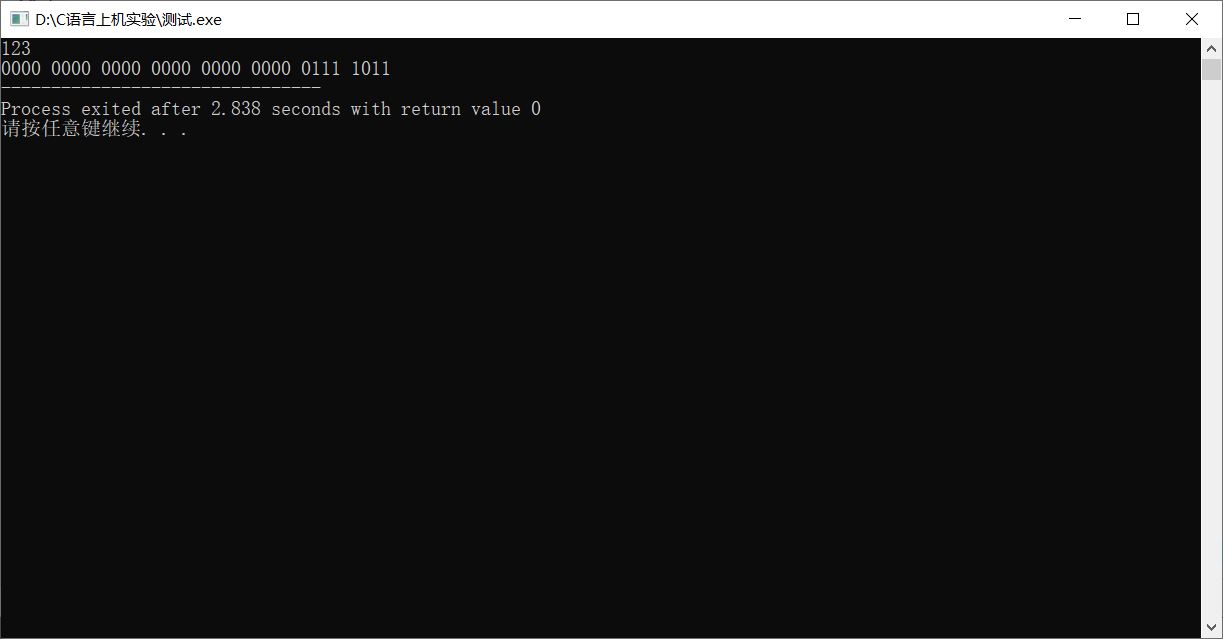


图2.9编程题一用例二的运行结果

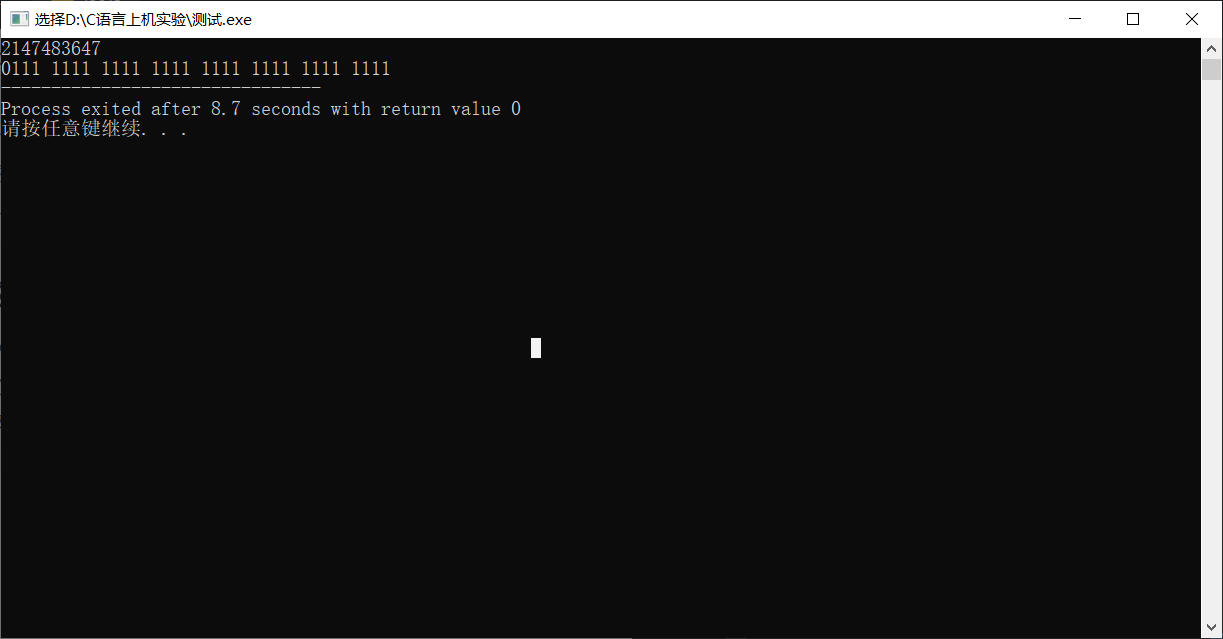


图2.10编程题一用例三的运行结果

2

思路：首先，要完成菜单的功能，用到函数，当满足条件的时候，转入函数。第一个函数，输入函数，采用二维数组输入学生姓名，方便后续的比较，采用数组存储成绩。第二个函数，排序函数，对于成绩的数组元素利用冒泡排序法进行排序，并交换对应位置的学生姓名，如果成绩相同，就按照姓名首字母和后续字母进行排序。第三个函数，输出函数，对应输出数组中的元素。

图2.11编程题二的流程图



图2.12编程题二的排序函数的流程图



源程序如下：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void sca();

void paixu();

void pri();

char a[100][100],b[100];

int n;

int main()

{

int c;

scanf("%d",&c);

while(c>0)//菜单选择

{

if(c==1)

sca();

else if(c==2)

paixu();

else if(c==3){

pri();

break;//输出之后没有菜单选择

}

scanf("%d",&c);

}

return 0;

}

void sca()

{

int i;

scanf("%d",&n);

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

{

scanf("%s",&a[i][0]);//每行输入一个名字

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

}

}

void paixu()

{

int i=0,j=0,t;

char c[20]={'\0'};//终止符

for(i=0;i<n-1;i++)//冒泡排序法

{

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

{

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

{

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

strcpy(c,a[j]);//将姓名交换

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

strcpy(a[j+1],c);

char c[20]={'\0'};

}

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

{

if(a[j][0]>a[j+1][0])//姓名首字母比较

{

strcpy(c,a[j]);

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

strcpy(a[j+1],c);

char c[20]={'\0'};

}

}

}

}

}

void pri()

{

int i;

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

{

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

}

}

表2.13编程题二的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | 1 3 a 90 b 70 c 85 3 0 | a 90  b 70  c 85 |
| 用例二 | 1 4 ming 80 jerry 89 hong 80 angel 87 2 3 0 | jerry 89  angel 87  hong 80  ming 80 |
| 用例三 | 1 4 xhong 80 jerry 89 xming 80 angel 87 2 3 0 | jerry 89  angel 87  xhong 80  xming 80 |

表2.13中的测试用例如图2.14 2.15 2.16所示

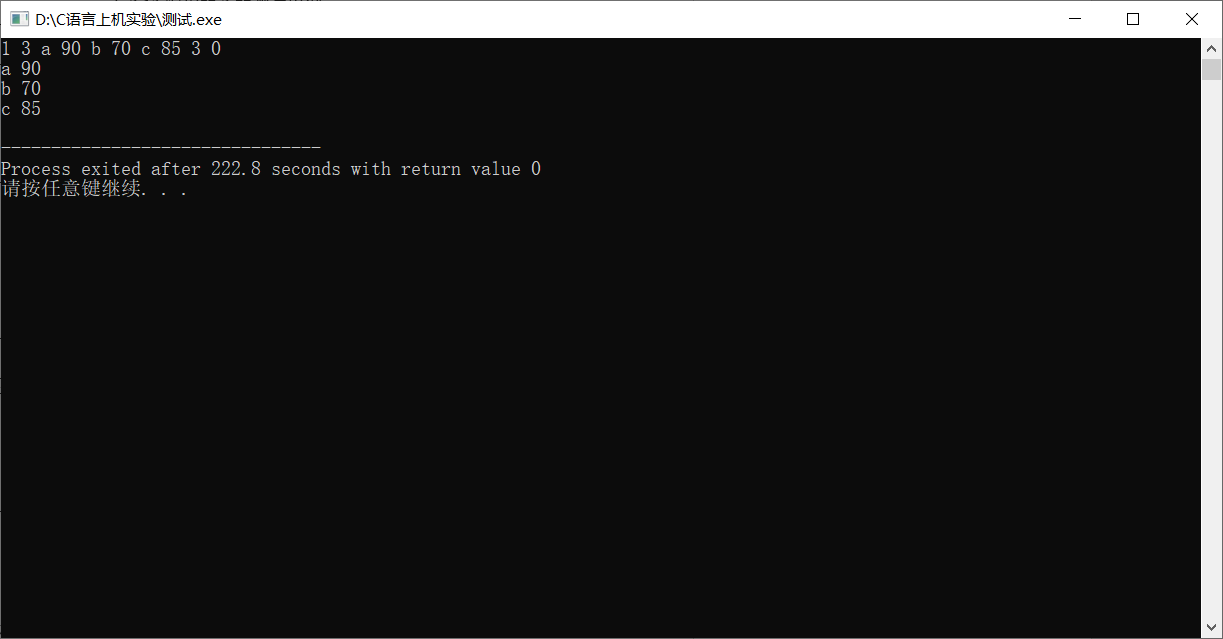


图2.14编程题二用例一的运行结果

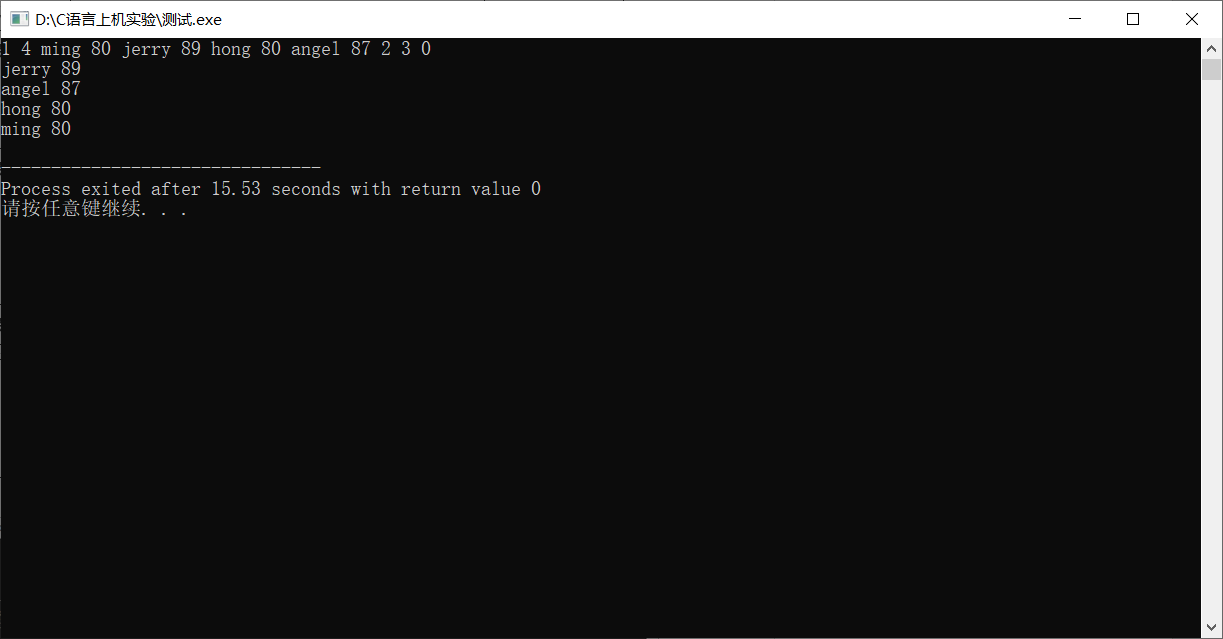


图2.15编程题二用例二的运行结果

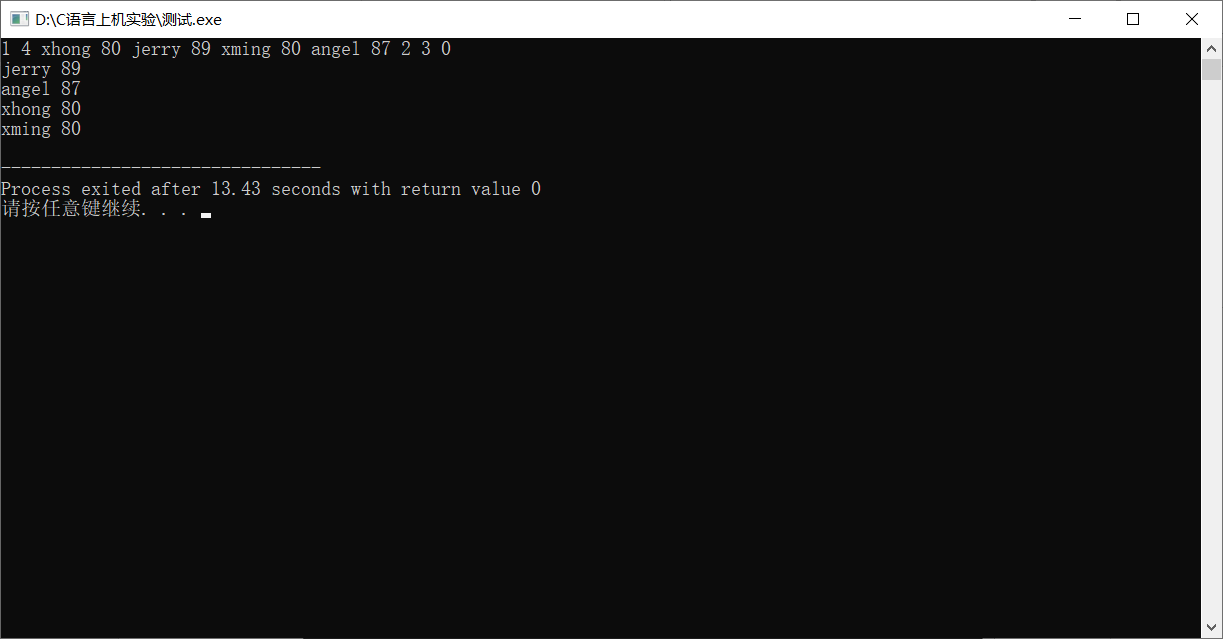


图2.16编程题二用例三的运行结果

3

思路：增加查找功能，定义查找函数，利用二分法，首先，在排序完成的基础上选取分数最高和最低值，再将最低和最高中间的值作为中间值，比对输入值和中间值，如果中间值小了，则最低值改为中间值之后的数，如果中间值大了，则最高值改为中间值之前的值，重新找到中间值，并且重新比对，如果发现有相等的情况，则输出，并返回1，如果最低值高于最高值，则返回0，如果函数返回了0，则输出not found

图2.17编程题三的流程图



图2.18编程题三的查找函数的流程图



源程序如下：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void sca();

void paixu();

void pri();

int cha();

char a[100][100],b[100];

int n;

int main()

{

int c;

scanf("%d",&c);

while(c>0)//菜单选择

{

if(c==1)

sca();

else if(c==2)

paixu();

else if(c==4)

{

int temp;

temp=cha();

if(temp==0)

printf("not found");

break;

}

else if(c==3)

{

pri();

break;//输出之后没有菜单选择

}

scanf("%d",&c);

}

return 0;

}

void sca()

{

int i;

scanf("%d",&n);

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

{

scanf("%s",&a[i][0]);//每行输入一个名字

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

}

}

void paixu()

{

int i,j,t;

char c[20]={'\0'};//终止符

for(i=0;i<n-1;i++)//冒泡排序法

{

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

{

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

{

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

strcpy(c,a[j]);//将姓名交换

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

strcpy(a[j+1],c);

char c[20]={'\0'};

}

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

{

if(a[j][0]>a[j+1][0])//姓名首字母比较

{

strcpy(c,a[j]);

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

strcpy(a[j+1],c);

char c[20]={'\0'};

}

else if(a[j][0]=a[j][0])

{

if(a[j][1]>a[j][2])

{

strcpy(c,a[j]);

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

strcpy(a[j+1],c);

char c[20]={'\0'};

}

}

}

}

}

}

void pri()

{

int i;

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

{

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

}

}

int cha()

{

int m,mid,fro=0,bac=n-1;

scanf("%d",&m);

//二分法

while(fro<=bac){

mid=(fro+mid)/2;

if(m<b[mid]){

bac=mid-1;

}

else if(m>b[mid]){

fro=mid+1;}

else {

if(b[mid]==b[mid+1]){

printf("%s %d",a[mid],b[mid]);

printf("\n");

printf("%s %d",a[mid+1],b[mid+1]);

return 1;

}

else {

printf("%s %d",a[mid],b[mid]) ;

return 1;

}

}

}

return 0;

}

表2.19编程题三的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | 1 3 xiaoming 90 xiaohong 79 zhangsan 83 2 4 83 0 | zhangsan 83 |
| 用例二 | 1 4 lisi 78 xiaoming 90 zhangsan 99 xiaohong 90 2 4 89 0 | not found |
| 用例三 | 1 4 lisi 78 xming 90 zhangsan 99 xhong 90 2 4 90 0 | xhong 90  xming 90 |

表2.19中的测试结果如图2.20 2.21 2.22所示

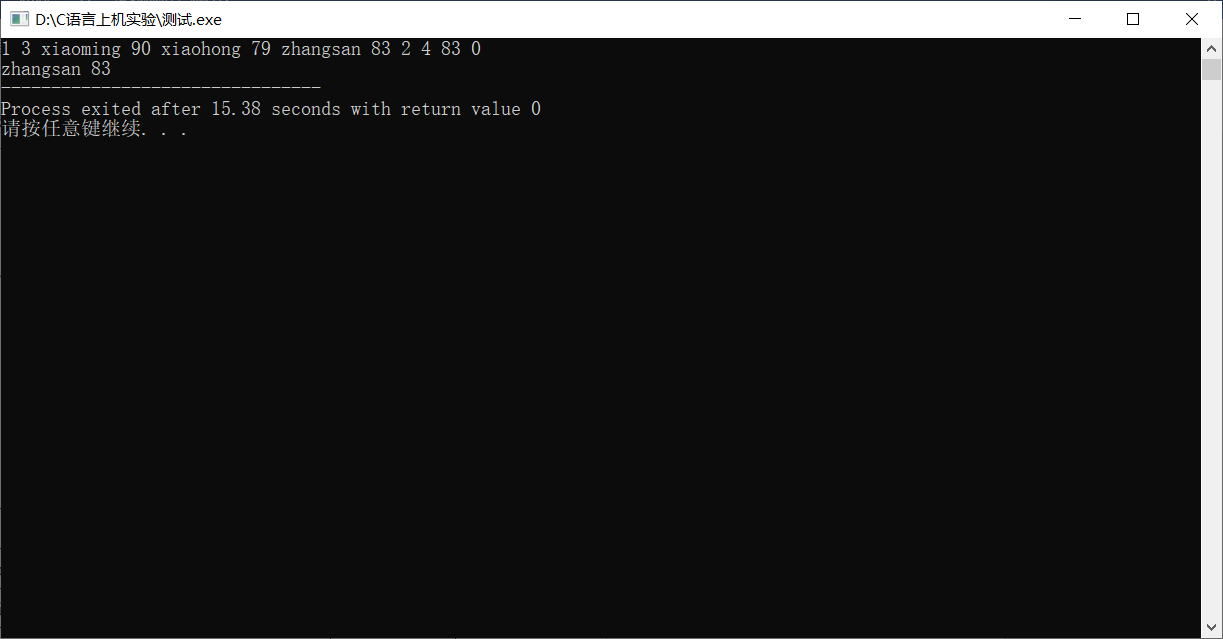


图2.20编程题三用例一的运行结果

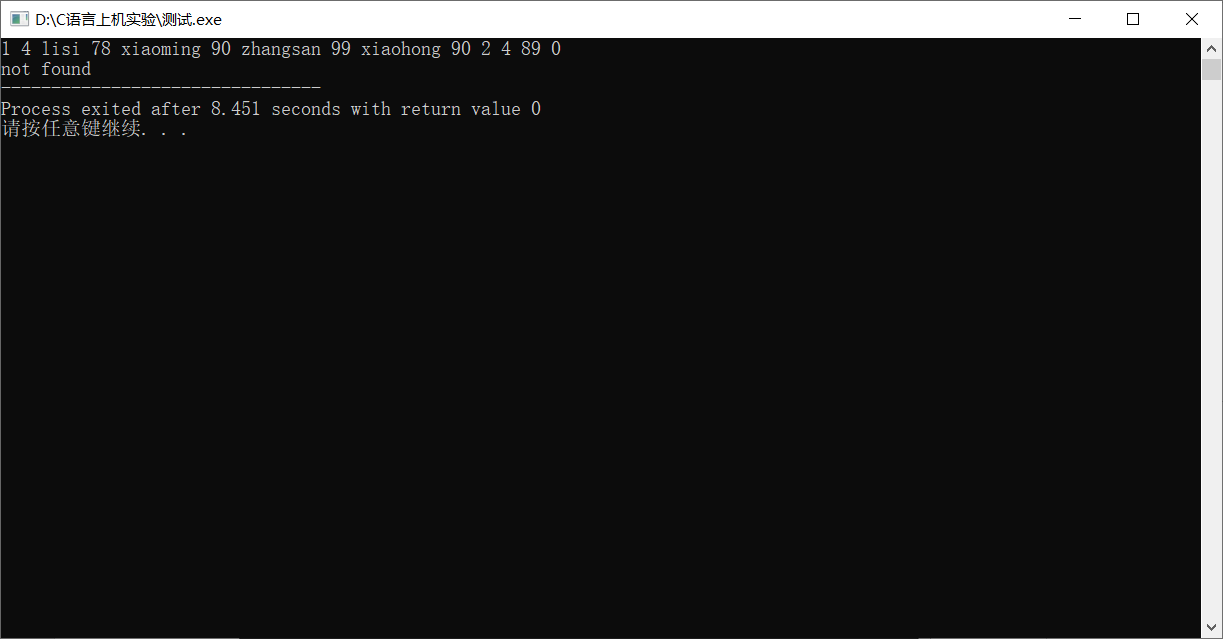


图2.21编程题三用例二的运行结果

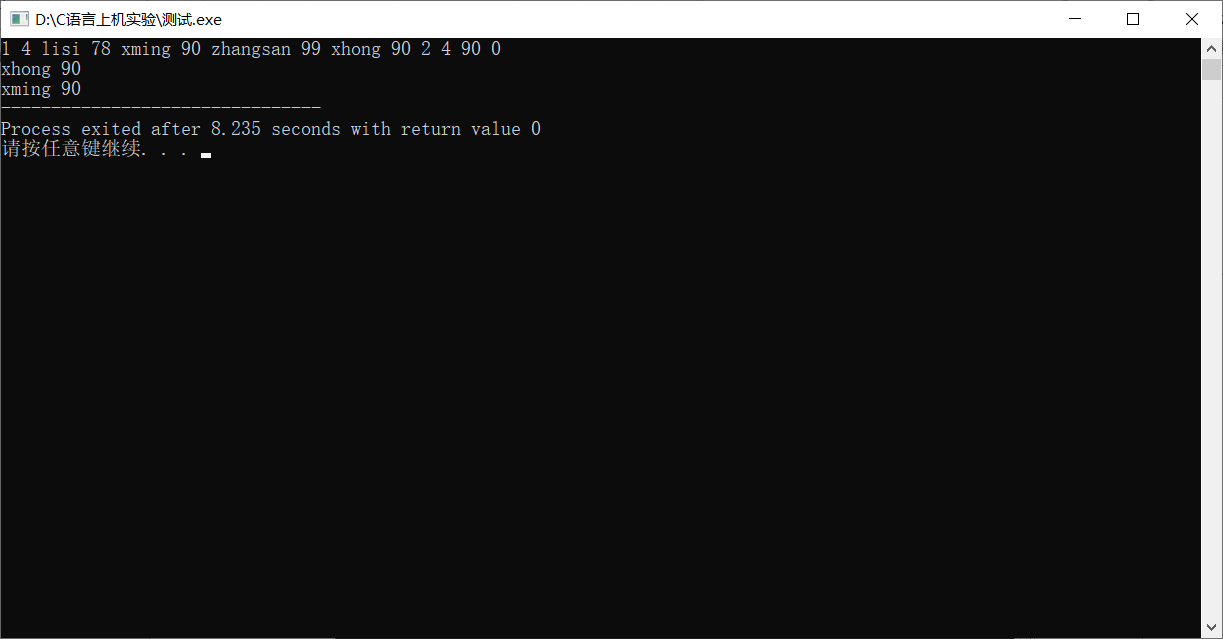


图2.22编程题三用例三的运行结果

5

思路：首先统计字符数组t与插入数组s的长度为pq，并比较n与q的大小关系如果n<q，利用temp数组将s中的元素储存起来，并将t中n位以后的元素在temp后存储，最后将t中n位以后的元素替换为temp中的元素。如果n=q，直接将t与s连接。

图2.23编程题五的strnins函数的流程图



源程序如下

# include<stdio.h>

# include<string.h>

# include<math.h>

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

int main(){

char s[200];

char t[200];

int n,i=0;

scanf("%s",t);

scanf("%s",s);

scanf("%d",&n);

strnins(t,s,n);

printf("%s",t);

return 0;

}

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

{

int p=strlen(t),q=strlen(s);

if(n<q){

char temp[100];

int i,j;

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

temp[i]=s[i];

for(j=0;j<p-n;j++)

temp[i++]=t[j+n];

for(j=0,i=n;i<100;i++)

t[i]=temp[j++];

}

else strcat(t,s);

}

表2.24编程题五的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | iyou love 1 | iloveyou |
| 用例二 | life wonderful\_ 0 | wonderful\_life |
| 用例三 | Sweet Candy 5 | SweetCandy |

表2.24的测试结果如图2.25 2.26 2.27所示

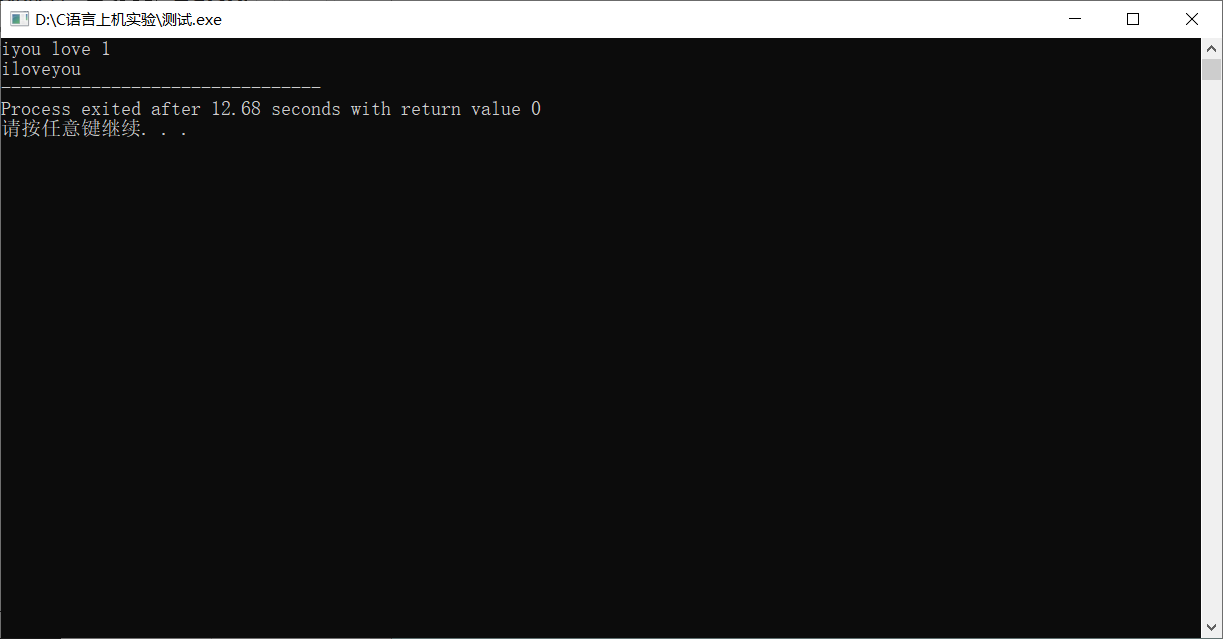


图2.25编程题五用例一的测试结果

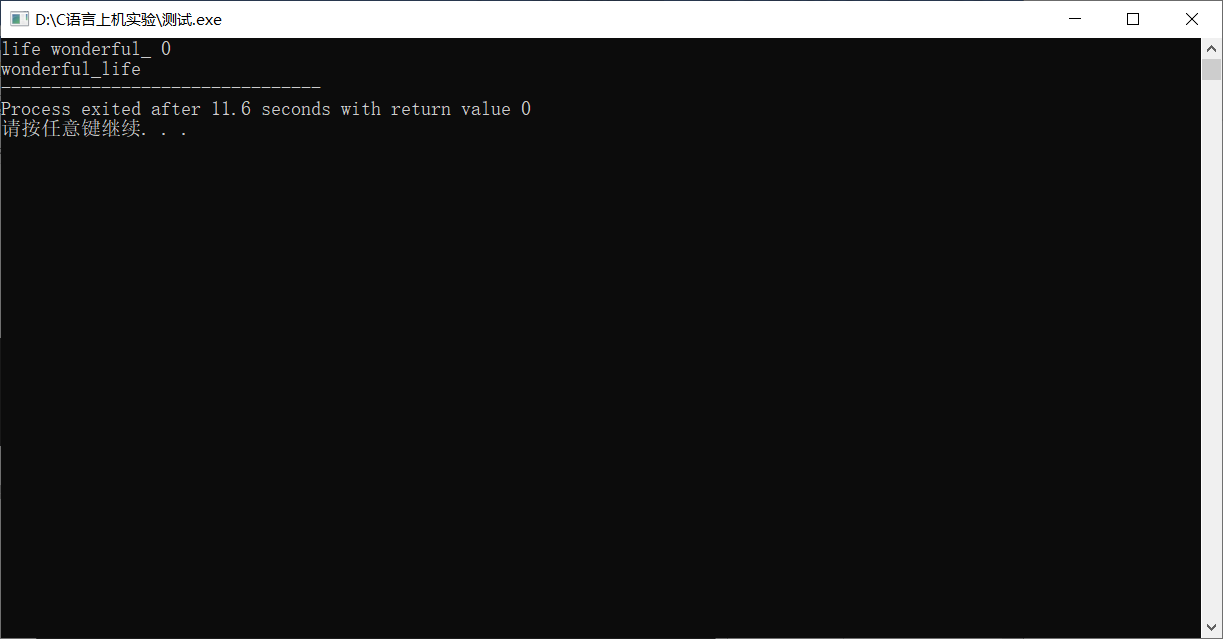


图2.26编程题五用例二的测试结果

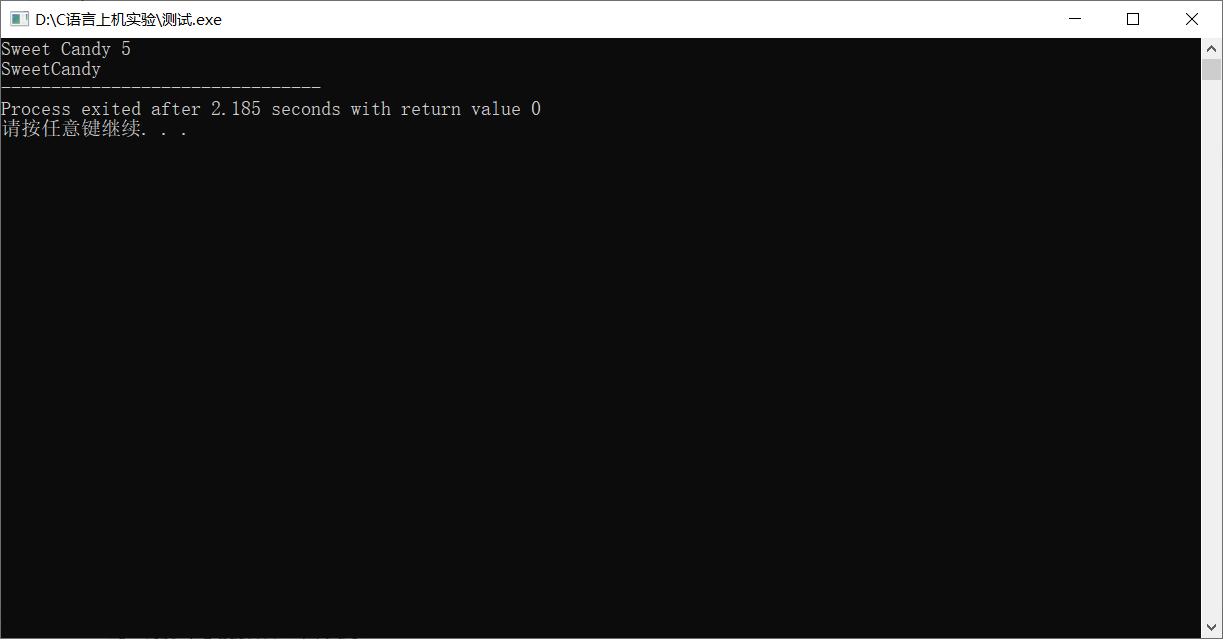


图2.27编程题五用例三的测试结果

6

思路：输入棋盘所表示的数组，并利用temp数组来表示出现重复的元素位置。首先比较行方向上的重复出现的元素：利用p进行标记，用k来表示第一个出现的元素，逐列比较，每出现重复的元素，p++，如果没有出现重复的元素，判断p的大小，如果p大于等于3，将temp数组中的对应元素标记为1，此后k重新初始化为之前未曾重复的元素，再次进行比较，直至每行结束。之后，用同样的方法比对每一列。

图2.28编程题六的流程图



源程序如下：

#include<stdio.h>

int main()

{

int n,m;

int i,j,l;

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

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

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

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

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

scanf("%d",&a[i][j]);//输入

// 计算横向的重复三次及以上的数字

for( i=0;i<n;i++)//对于每行而言

{

int p=1;//作为标记出现的

int k = a[i][0];//对于第i+1行而言

for( j=1;j<m;j++)//从第二列开始比较

{

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

{

p++;//p标记重复的个数

}

else if(p >= 3)//当出现不同的数如果满足消去条件 ，开始循环

{

int q = j;//不能占用标记列的字符

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

{

temp[i][q-1] =1;//重新定义一个数组，避免出现交叉情况 ，并且，将满足条件的标记为1

q--;

}

k = a[i][j];//重新开始比对计数

p = 1;

}

else//出现了不同的数，且不满足消去条件，重新开始比对

{

k = a[i][j];

p = 1;

}

if(j == m-1 && p>=3)//对于最后的数进行额外判断

{

int q = j;

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

{

temp[i][q] = 1;

q--;

}

}

}

}

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

{

int p=1;

int k = a[0][j];

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

{

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

{

p++;

}

else if(p >= 3)

{

int q = i;

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

{

temp[q-1][j] = 1;

q--;

}

k = a[i][j];

p = 1;

}

else

{

k =a[i][j];

p = 1;

}

if(i == n-1 && p>=3)

{

int q = i;

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

{

temp[q][j] = 1;

q--;

}

}

}

}

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

{

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

{

if(temp[i][j]==1)

a[i][j]=0;

if(j<m-1)

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

else

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

}

printf("\n");

}

return 0;

}

图2.29编程题六的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | 4 5 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 |
| 用例二 | 4 4 4 3 4 4 4 4 1 4 9 6 4 8 1 6 3 8 | 4 3 4 4  4 4 1 4  9 6 4 8  1 6 3 8 |
| 用例三 | 4 5 1 1 1 2 3 1 5 4 2 3 1 5 4 2 3 1 5 4 4 4 | 0 0 0 0 0  0 0 0 0 0  0 0 0 0 0  0 0 0 0 0 |

图2.29的测试结果如图2.30 2.31 2.32所示

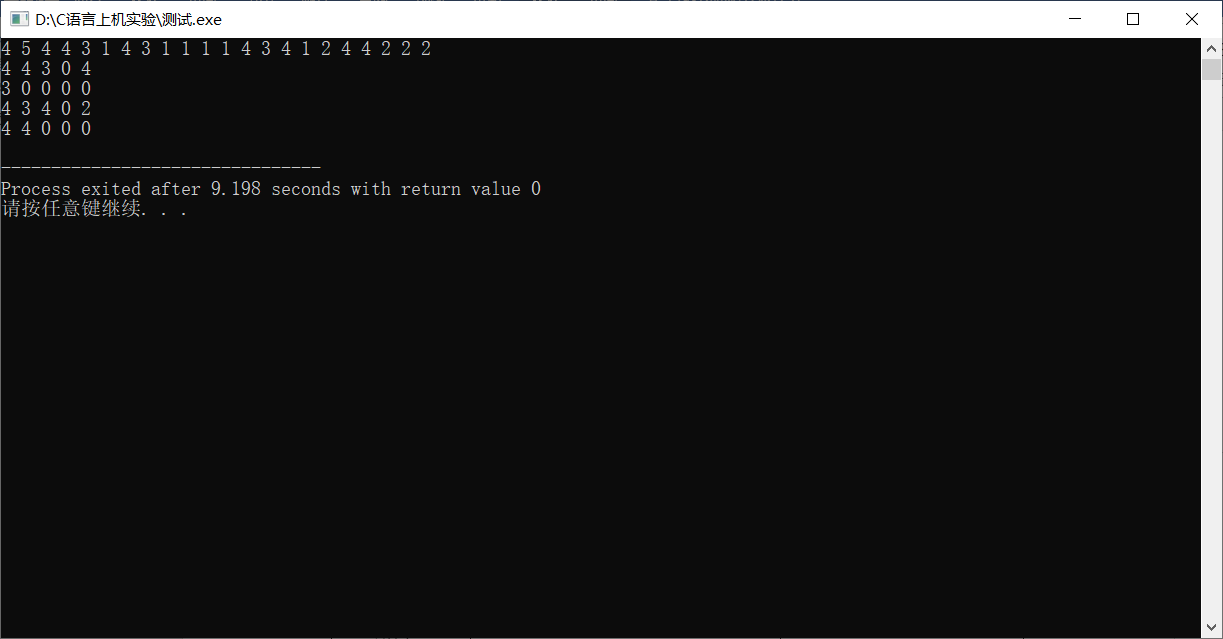


图2.30编程题六用例一的运行结果

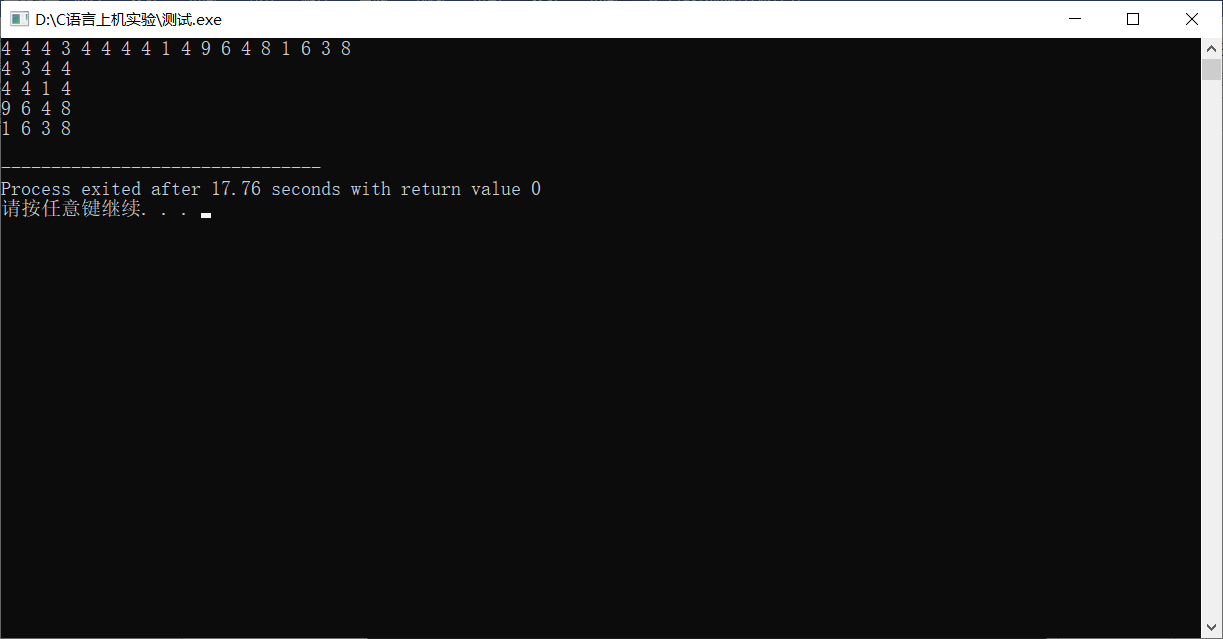


图2.31编程题六用例二的运行结果

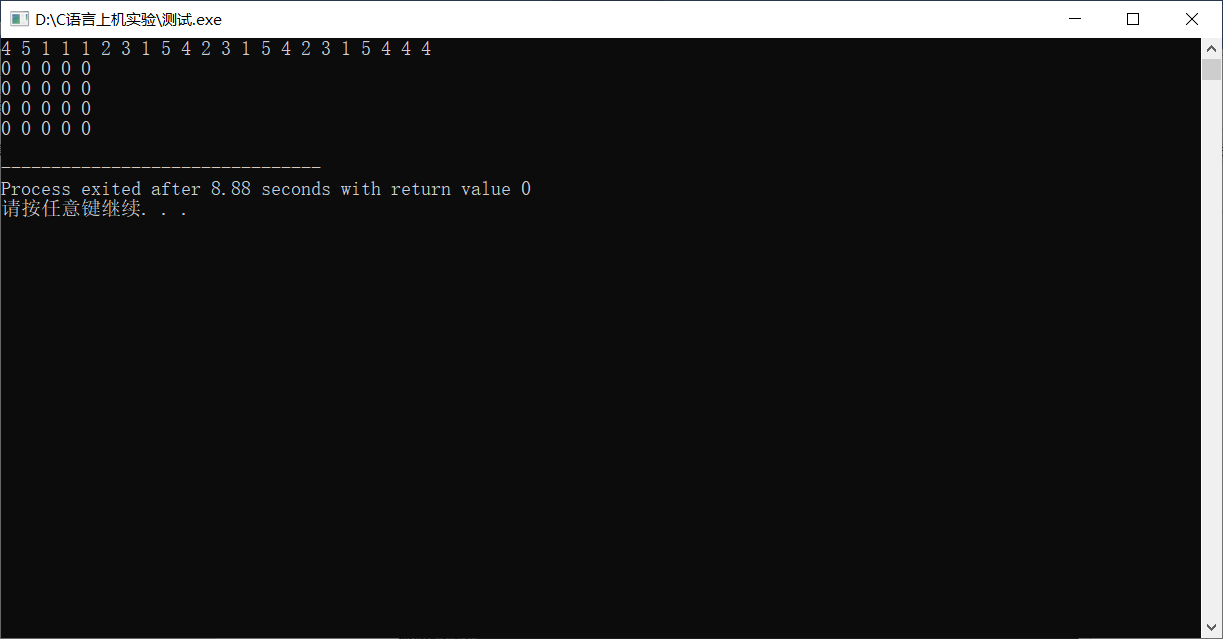


图2.32编程题六用例三的运行结果

10

思路：出现顺转的方阵，那么，利用循环，每一次完成外围一圈。输入阶数n，方阵为n\*n型，最外围循环只需完成（n+1）/2次，首先完成该循环内外围上行的数据，然后转到列处，完成循环内外围右列的数据，再转至行，完成循环内外围下行的数据，再转至列处，完成外围左行的数据，最终完成。

图2.33编程题十的流程图



源程序如下：

#include <stdio.h>

int main()

{

int i,j,n,count;

int a[100][100]={0}; //利用二维数组存储

scanf("%d",&n);

count=1; //作为表格中出现的数

for(i=0;i<(n+1)/2;i++) //在每一次循环的时候，都会减少上下两行只要循环一半的行数

{

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

{

a[i][j] = count++;

}

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

{

a[j][n-i-1] = count++;

}

for (j=n-i-2;j>=i;j--)

{

a[n-i-1][j] = count++;

}

for (j=n-i-2;j>i;j--)

{

a[j][i] = count++;

}

}

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

{

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

{ if(j<n-1)

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

else

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

}

printf("\n");

}

return 0;

}

图2.34编程题十的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | 3 | 1 2 3  8 9 4  7 6 5 |
| 用例二 | 1 | 1 |
| 用例三 | 5 | 1 2 3 4 5  16 17 18 19 6  15 24 25 20 7  14 23 22 21 8  13 12 11 10 9 |

图2.34的测试用例如图2.35 2.36 2.37所示

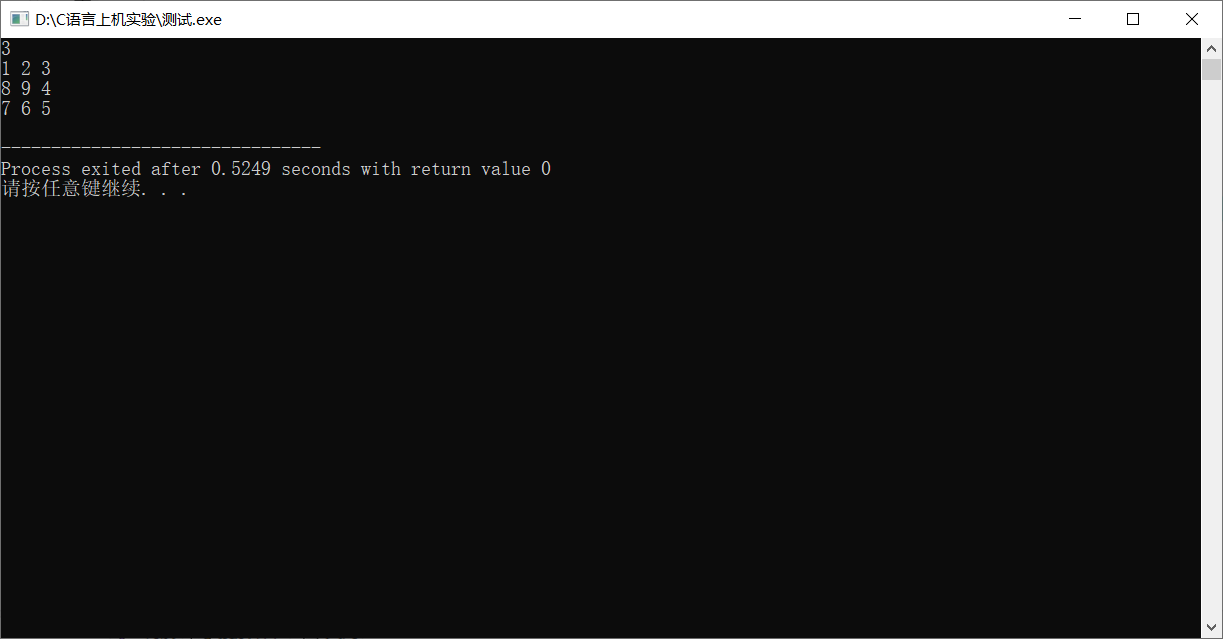


图2.35编程题十用例一的运行结果

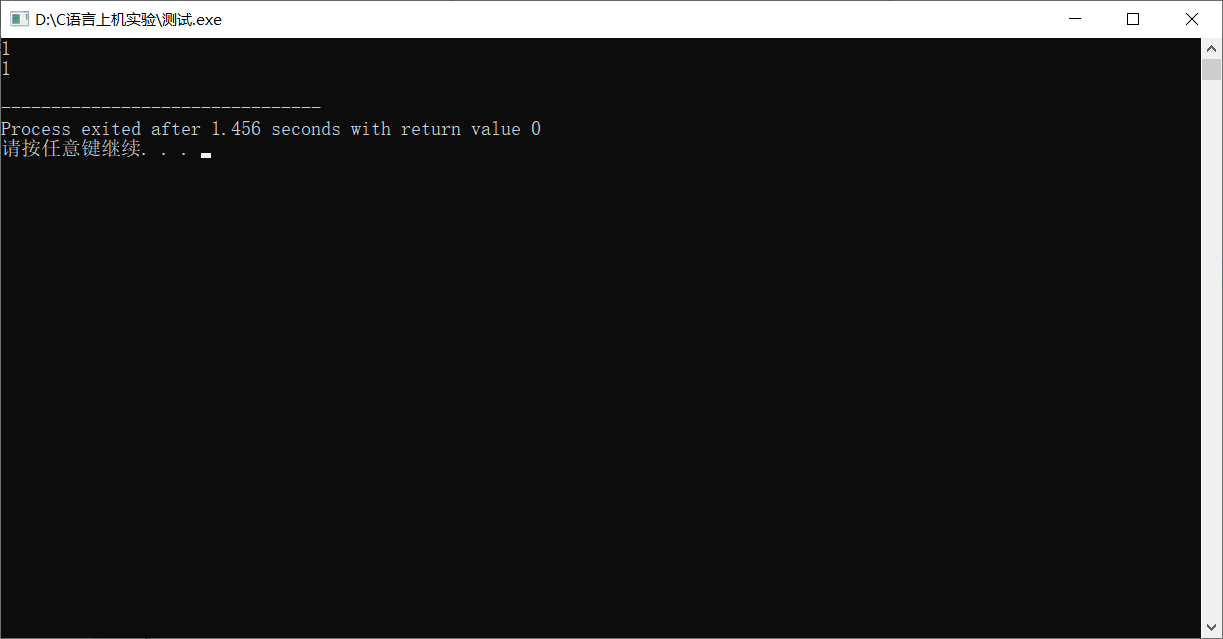


图2.36编程题十用例二的运行结果

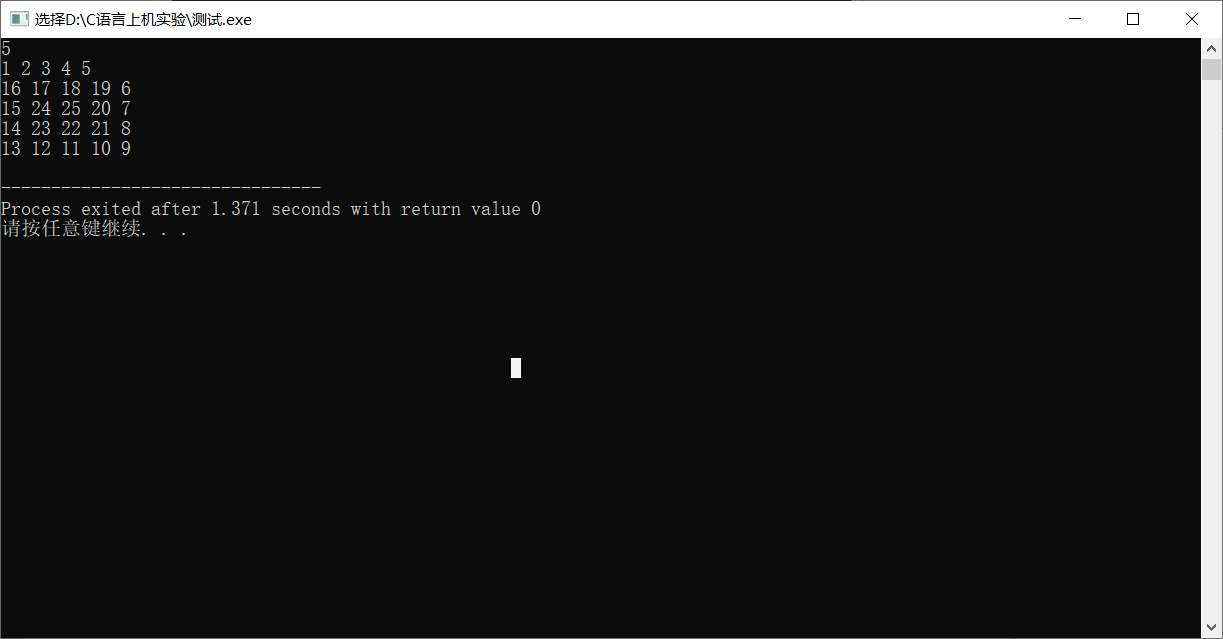


图2.37编程题十用例三的运行结果

11

思路：根据输入的行列数，将迷宫多余的行列进行初始化，避免影响后续判断，输入迷宫，利用函数对迷宫进行历遍,将符合条件的情况进行输出

源程序如下：

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void visited(int pa[20][20], int i, int j,int x,int y);

int main()

{

int a[20][20];

int x,y,i,j;

scanf("%d%d",&x,&y);

for(i=0;i<x+2;i++)

{

for(j=0;j<y+2;j++)

{

if((i==0)||(j==0)||(i==x+1)||(j==y+1)) a[i][j]=-1;//避免影响

}

}

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

{

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

{

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

}

}

visited(a, 1, 1, x, y);

return 0;

}

void visited(int pa[20][20], int i, int j,int x,int y)//利用函数对迷宫进行历遍 ，当出现符合条件的情况进行输出

{

int n, m;

pa[i][j] = 2;

if(i == x && j == y){

printf("\n");

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

for(m=1; m<y+1; m++){

if(m<y){

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

printf("1 ");

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

printf("0 ");

else

printf("0 ");}

else{

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

printf("1");

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

printf("0");

else

printf("0");}

}

printf("\n");

}

}

//再对于后续的元素进行递归，达到历遍的目的

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

visited(pa, i-1, j, x, y);

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

visited(pa, i+1, j, x, y);

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

visited(pa, i, j-1, x, y);

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

visited(pa, i, j+1, x, y);

pa[i][j] = 1;

}

表2.38实验题十一的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | 6 10 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 | 1 0 0 0 0 0 0 0 0 0  1 0 0 0 0 1 1 1 1 0  1 0 0 0 0 1 0 0 1 0  1 0 0 1 1 1 0 0 1 0  1 1 1 1 0 0 0 0 1 0  0 0 0 0 0 0 0 0 1 1  1 0 0 0 0 0 0 0 0 0  1 0 0 0 0 0 1 1 1 0  1 0 0 0 0 1 1 0 1 0  1 0 0 1 1 1 0 0 1 0  1 1 1 1 0 0 0 0 1 0  0 0 0 0 0 0 0 0 1 1  1 0 0 0 0 0 0 0 0 0  1 0 0 0 0 0 0 0 0 0  1 0 0 0 0 0 0 0 0 0  1 0 0 1 1 1 0 0 0 0  1 1 1 1 0 1 1 1 1 0  0 0 0 0 0 0 0 0 1 1 |
| 用例二 | 5 5 1 0 0 1 0 1 1 1 1 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 1 | 1 0 0 0 0  1 1 0 0 0  0 1 0 0 0  0 1 1 0 0  0 0 1 1 1  1 0 0 0 0  1 1 1 1 1  0 0 0 0 1  0 0 0 0 1  0 0 0 0 1 |

表2.38中的测试结果如图2.39 2.40所示

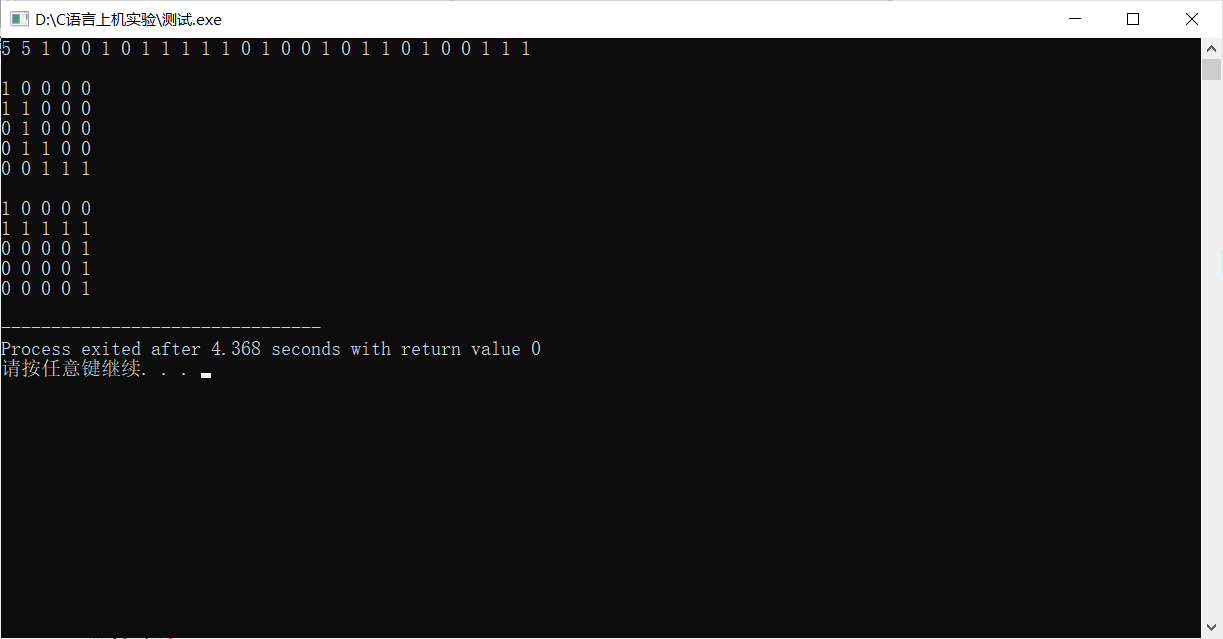


图2.39编程题十一用例一的运行结果

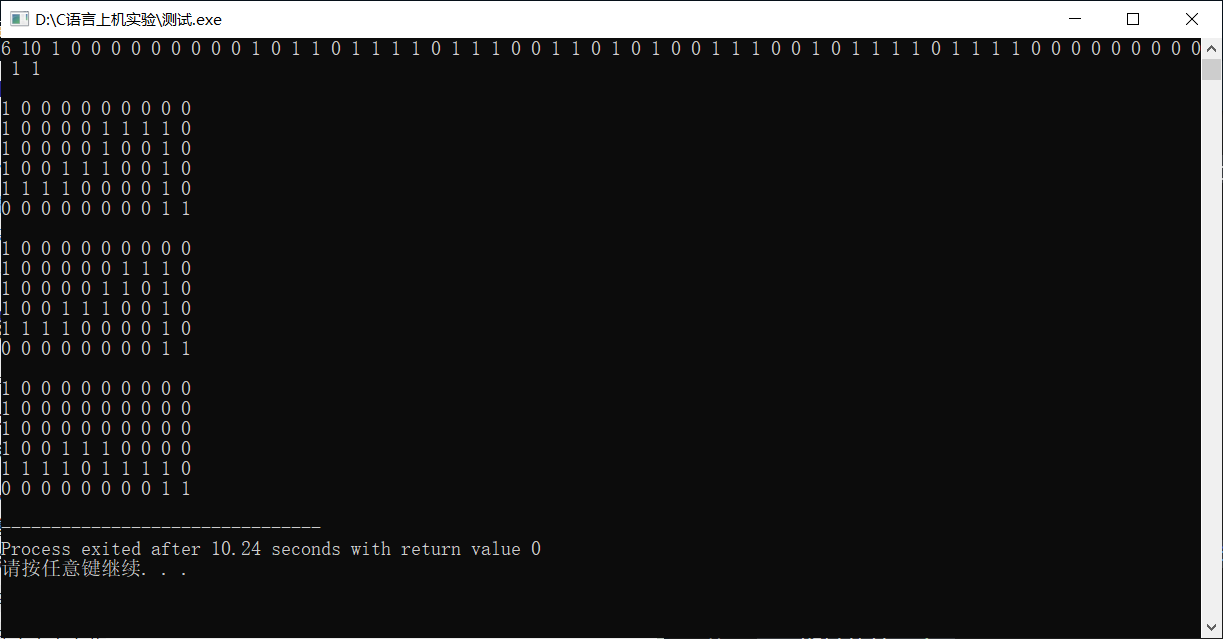


图2.40编程题十一用例二的运行结果

## 2.4 小结

通过本次的实验，加强了对于数组及其应用的深刻认识，同时深化对于函数以及循环的运用，但是仍然有很多不足，对于数组的运用不够熟练，对于程序的设计不够精简，对于程序的思路没有头绪，同时，在对于流程图的设计也有诸多的问题，在老师和同学的帮助下，仍然存在不够完善之处。

# 3结构与联合实验

## 3.1表达式求值的程序验证

源程序如下：

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

struct T{

int x;

char c;

char \*t;

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

表达式1：(++p)->x，先计算++p的值，即为a[1]，其中的x即为100，故计算值为100

表达式2：p++，p->c，逗号表达式，值为最后一个表达式的值，故为B

表达式3：\*p++->t,\*p->t,同为逗号表达式，p进行+1运算后为a[1]，则最终结果是v[0],即为x

表达式4：\*(++p)->t，对于p进行+1操作并访问结构体中的t，对t进行取值操作，故为x

表达式5：\*++p->t,对于p进行访问结构体中t，对t+1进行取值为V

表达式6：++\*p->t,对于p进行访问结构体中的t，并取值，值+1为V

## 3. 2源程序修改替换

原程序中错误在于形参是链表头部的的地址无法修改，最后程序无法实现，应该采用二级指针的形参，修改链表头部的指针。

修改后的程序如下：

#include <stdio.h>

#include <stdlib.h>

struct s\_list{

int data;

struct s\_list \*next;

};

void create\_list(struct s\_list \*\*headp,int \*p);//利用二级指针修改指针的地址

int main(void){

struct s\_list \*head=NULL,\*p;//head作为指针

int s[10],i=0;

do

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

while (s[i++]);

create\_list(&head,s);//修改指针地址使得指针指向表头

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)

\*headp=NULL;//直接将指针指向空

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;//修改指针的地址，

}

程序修改：原程序是一个先进先出的链表，现改为一个后进先出的链表，使新节点的指针指向链表的第一个节点，再使新节点成为链表的第一个节点，实现对于输入次序的逆序

修改后的程序如下：

#include <stdio.h>

#include <stdlib.h>

struct s\_list{

int data;

struct s\_list \*next;

};

void create\_list(struct s\_list \*\*headp,int \*p);//利用二级指针修改指针的地址

int main(void){

struct s\_list \*head=NULL,\*p;//head作为指针

int s[10],i=0;

do

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

while (s[i++]);

create\_list(&head,s);//修改指针地址使得指针指向表头

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 \*h=NULL,\*f;

while (\*p)

{

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

f->data=\*p++;

f->next=h; //新结点的指针指向链表的第一个结点

h=f; //使新结点成为链表的第一个结点

}

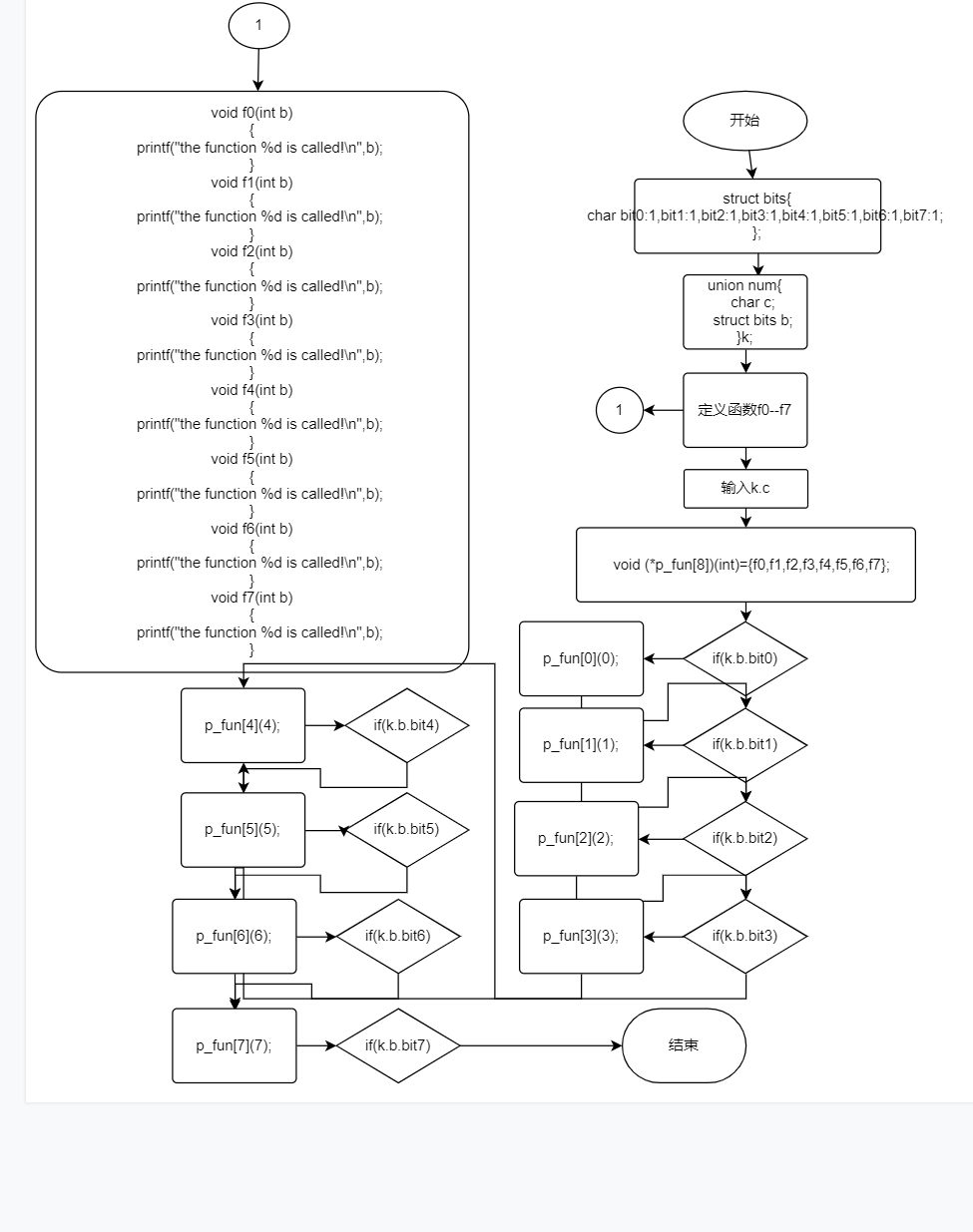
\*headp=h;

}

## 3.3程序设计

1分析：按照题目要求，设计8个函数，以biti为形参，用函数指针数组来存放8个函数，设计字段结构，按照题目要求每个字符为一个字节，对于输入的无符号整型数，按照二进制形式，转化并存储到数组，再根据该整型数的二进制形式进行函数调用。

图3.1编程题一的流程图



#include<stdio.h>

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);

}

int main()

{

struct bits{

unsigned char bit0:1;

unsigned char bit1:1;

unsigned char bit2:1;

unsigned char bit3:1;

unsigned char bit4:1;

unsigned char bit5:1;

unsigned char bit6:1;

unsigned char bit7:1;

}fun;

void (\*p\_fun[8])(int b);//函数指针数组来存放函数

int i;

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;

unsigned int tell;

scanf("%d",&tell);

int a[8]={0};

for (i = 0;tell > 0;i++) {

a[i] = tell % 2;

tell/= 2;

}

fun.bit0=a[0];

fun.bit1=a[1];

fun.bit2=a[2];

fun.bit3=a[3];

fun.bit4=a[4];

fun.bit5=a[5];

fun.bit6=a[6];

fun.bit7=a[7];

if(fun.bit0)

p\_fun[0](0);

if(fun.bit1)

p\_fun[1](1);

if(fun.bit2)

p\_fun[2](2);

if(fun.bit3)

p\_fun[3](3);

if(fun.bit4)

p\_fun[4](4);

if(fun.bit5)

p\_fun[5](5);

if(fun.bit6)

p\_fun[6](6);

if(fun.bit7)

p\_fun[7](7);

return 0;

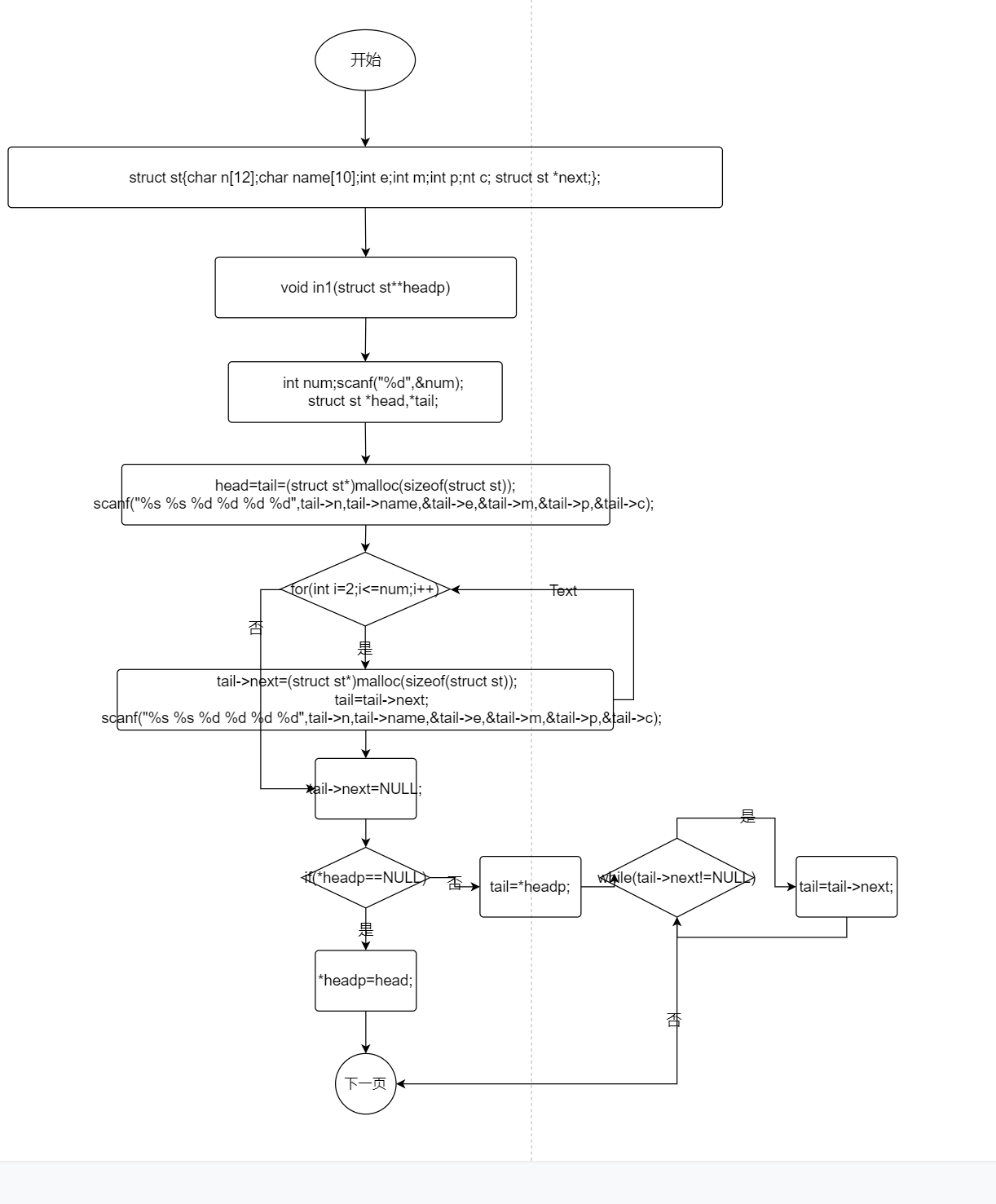
}

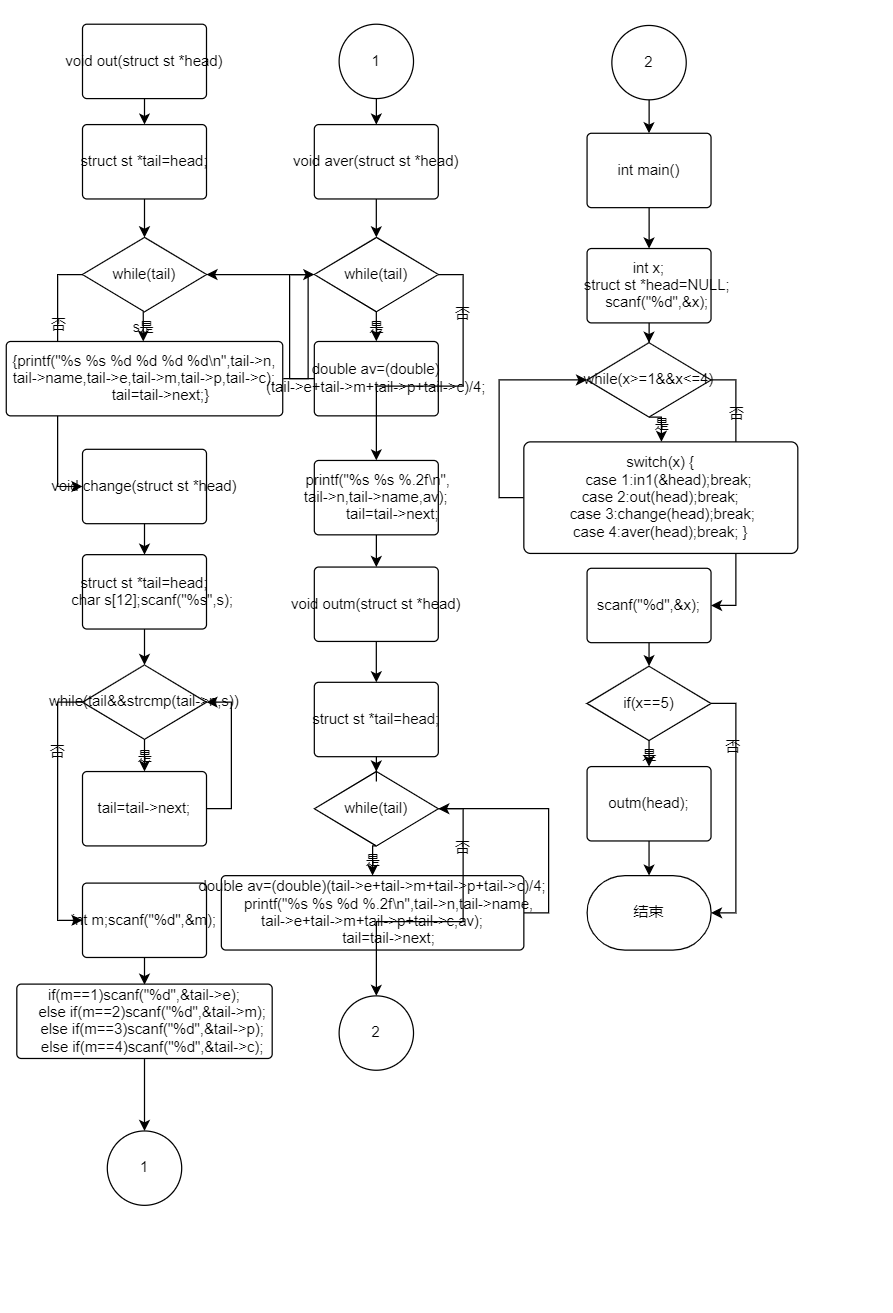
图3.2编程题一的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | 123 | the function 0 is called!  the function 1 is called!  the function 3 is called!  the function 4 is called!  the function 5 is called!  the function 6 is called! |
| 用例二 | 255 | the function 0 is called!  the function 1 is called!  the function 2 is called!  the function 3 is called!  the function 4 is called!  the function 5 is called!  the function 6 is called!  the function 7 is called! |
| 用例三 | 0 |  |

2思路：要实现菜单选择功能，采用函数，利用输入，输出，修改，计算，最终输出四个函数，对于输入函数，先在主函数内创建链头，调用输入函数，判断链头后面是否为空，如果为空，则在链头的下一处进行输入，不为空，则转至最后一个不空的链节，再进行输入，对于输入的最后一个学生进行特殊化，使得该学生的后一个链节为空。输出函数以链头指针为形参，逐个进行输出。修改函数以链头指针作为形参，历遍找到符合条件的学生，对于数据进行修改，计算函数以链头指针作为形参，逐个计算学生平均分总分，对于成绩保留两位小数，最终输出以链头指针为形参，进行输出

图3.3编程题二的流程图





源程序如下：

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

struct st{

char n[12];

char name[10];

int e;

int m;

int p;

int c;

int sum;

struct st \*next;

};//定义结构型

void in1(struct st\*\*headp)

{

int num;scanf("%d",&num);//输入要输入信息的个数

struct st \*head,\*tail;//确定新建的头尾链表指针

head=tail=(struct st\*)malloc(sizeof(struct st));//为链表头指针分配动态空间

scanf("%s %s %d %d %d %d",tail->n,tail->name,&tail->e,&tail->m,&tail->p,&tail->c);//输入每个学生的学号、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

head->sum=tail->e+tail->m+tail->p+tail->c;

for(int i=2;i<=num;i++)

{

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

tail=tail->next;//指针指向新建链表

scanf("%s %s %d %d %d %d",tail->n,tail->name,&tail->e,&tail->m,&tail->p,&tail->c); //输入每个学生的学号、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

tail->sum=tail->e+tail->m+tail->p+tail->c;

}

tail->next=NULL;

if(\*headp==NULL)\*headp=head;//若此链表为空

else

{

tail=\*headp;

while(tail->next!=NULL)

{

tail=tail->next;

}

tail->next=head;

}//若不为空，则把新建头指针的值赋给先前最后一个结构的next

}

void out(struct st \*head)

{

struct st \*tail=head;

while(tail)

{

printf("%s %s %d %d %d %d\n",tail->n,tail->name,tail->e,tail->m,tail->p,tail->c);

//输出每个学生的学号、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

tail=tail->next;

}

}

void change(struct st \*head)

{

struct st \*tail=head;

char s[12];scanf("%s",s);//寻找要更改的对象

while(tail&&strcmp(tail->n,s)) //寻找要更改的对象

{

tail=tail->next;//下一个结构

}

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

if(m==1)scanf("%d",&tail->e);

else if(m==2)scanf("%d",&tail->m);

else if(m==3)scanf("%d",&tail->p);

else if(m==4)scanf("%d",&tail->c);//做相应的更改

tail->sum=tail->e+tail->m+tail->p+tail->c;

}

void aver(struct st \*head)

{

struct st \*tail=head;

while(tail)

{

printf("%s %s %.2f\n",tail->n,tail->name,(double)tail->sum/4);//输出每一个结果

tail=tail->next; //寻找直到最后一个结构

}

}

void outm(struct st \*head)

{

struct st \*tail=head;

while(tail)

{

printf("%s %s %d %.2f\n",tail->n,tail->name,tail->e+tail->m+tail->p+tail->c,(double)tail->sum/4);//输出

tail=tail->next;

}

}

int main()

{

int x;

struct st \*head=NULL;

scanf("%d",&x);//输入选项

while(x>=1&&x<=4)

{

switch(x)//根据不同选项做出选择

{

case 1:in1(&head);break;

case 2:out(head);break;

case 3:change(head);break;

case 4:aver(head);break;

}

scanf("%d",&x);

}

if(x==5)outm(head);

}

图3.4编程题二的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | 1 2 U202012345 Jack 99 100 80 96 U202054321 Rose 89 94 85 100 2 3 U202054321 1 66 4 5 | U202012345 Jack 99 100 80 96  U202054321 Rose 89 94 85 100  U202012345 Jack 93.75  U202054321 Rose 86.25  U202012345 Jack 375 93.75  U202054321 Rose 345 86.25 |
| 用例二 | 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 | U202012345 Jack 99 100 80 96  U202054321 Rose 89 94 85 100  U202056789 Tom 12 34 56 78  U202098765 Jerry 98 76 54 32  U202012345 Jack 93.75  U202054321 Rose 86.25  U202056789 Tom 45.00  U202098765 Jerry 65.00  U202012345 Jack 375 93.75  U202054321 Rose 345 86.25  U202056789 Tom 180 45.00  U202098765 Jerry 260 65.00 |
| 用例三 | 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 | U123456789 Elio 34 56 78 90  U123456789 Elio 301 75.25  U987654321 Gray 0 0.00  U000000000 Black 400 100.00  U202073456 Red 245 61.25 |

3思路：增加排序函数， 利用冒泡排序法，只改变节点的数据域，不改变节点的连接顺序

图3.5编程题三的流程图



源程序如下：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct information{

char num[15];

char name[10];

int yinyu;

int shuxue;

int wuli;

int cyuyan;

float average;

int sum;

struct information \*next;

};

void paixu(struct information \*head)//利用平均分排序

{

struct information \*p;

p=head;

while(p->next!=NULL)

{

p->sum=p->yinyu+p->shuxue+p->wuli+p->cyuyan;

p->average=p->sum\*25;

p->average+=0.5;

p->average=(int )p->average;

p->average/=100;

p=p->next;

}

struct information \*q=head;

int i=0;

p=head;

while(p->next!=NULL)

{

p=p->next;i++;

}

p=head->next;

for(int i1=0;i1<i-1;i1++)

{

int change=1;

for(int i2=0;i2<i-i1-1;i2++)

{

if(q->sum>p->sum)

{

char s[20];strcpy(s,q->num);strcpy(q->num,p->num);strcpy(p->num,s);

strcpy(s,q->name);strcpy(q->name,p->name);strcpy(p->name,s);

int m=p->yinyu;p->yinyu=q->yinyu;q->yinyu=m;

m=p->shuxue;p->shuxue=q->shuxue;q->shuxue=m;

m=p->wuli;p->wuli=q->wuli;q->wuli=m;

m=p->cyuyan;p->cyuyan=q->cyuyan;q->cyuyan=m;

m=p->sum;p->sum=q->sum;q->sum=m;

change=0;

}

q=p;p=p->next;

}

if(change==1)break;

p=head->next;q=head;

}

}

void shuru(struct information \*head,int n)

{

int i;

struct information \*p;

p=head;

while(p->next!=NULL)

{

p=p->next;

}

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

{

if(i<n-1)

{

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

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

scanf("%d %d %d %d",&(\*p).yinyu,&(\*p).shuxue,&(\*p).wuli,&(\*p).cyuyan);

p->next=(struct information\*)malloc(sizeof(struct information));

p=p->next;

}

else {//最后一个学生特殊化

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

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

scanf("%d %d %d %d",&(\*p).yinyu,&(\*p).shuxue,&(\*p).wuli,&(\*p).cyuyan);

p->next=(struct information\*)malloc(sizeof(struct information));

p=p->next;

p->next=NULL;

}

}

}

void shuchu(struct information \*head)

{

// paixu(head);

struct information \*p;

p=head;

while(p->next!=NULL)

{

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

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

printf("%d %d %d %d\n",p->yinyu,p->shuxue,p->wuli,p->cyuyan);

p=p->next;

}

}

void xiugai(struct information \*head)

{

struct information \*p;

struct information tell;//修改的学生信息

scanf("%s",tell.num);

int xuan;

scanf("%d",&xuan);//修改的科目

int gai;

scanf("%d",&gai);//最终分数

p=head;

while(strcmp(p->num,tell.num)!=0)//字符串的比较

{

p=p->next;

}

switch(xuan)

{

case 1:p->yinyu=gai;break;

case 2:p->shuxue=gai;break;

case 3:p->wuli=gai;break;

case 4:p->cyuyan=gai;break;

}

}

void jisuan(struct information \*head)

{

struct information \*p;

p=head;

while(p->next!=NULL)

{

p->sum=p->yinyu+p->shuxue+p->wuli+p->cyuyan;

p->average=p->sum\*25;

p->average+=0.5;

p->average=(int )p->average;

p->average/=100;

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

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

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

p=p->next;

}

}

void shuchu2(struct information \*head)

{

struct information \*p;

// paixu(head);

p=head;

while(p->next!=NULL)

{

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

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

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

p=p->next;

}

}

int main()

{

int choice;

scanf("%d",&choice);//菜单选项

int n;

struct information \*head,\*p;

head=(struct information\*)malloc(sizeof(struct information));//动态存储

head->next=NULL;

while(choice>0)//实现菜单选择功能

{

if(choice==1)//实现对于学生信息输入

{

scanf("%d",&n);//输入学生人数

shuru(head,n);

paixu(head);

}

else if(choice==2)//实现对于学生信息输出

{

shuchu(head);

}

else if(choice==3)//实现成绩修改

{

xiugai(head);

}

else if(choice==4)

{

jisuan(head);

}

else if(choice==5)

{

shuchu2(head);

break;

}

scanf("%d",&choice);

}

return 0;

}

图3.6编程题三的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | 1 2 U202054321 Rose 89 94 85 100 U202012345 Jack 99 100 80 96 2 3 U202054321 1 66 4 5 | U202054321 Rose 89 94 85 100  U202012345 Jack 99 100 80 96  U202054321 Rose 86.25  U202012345 Jack 93.75  U202054321 Rose 345 86.25  U202012345 Jack 375 93.75 |
| 用例二 | 1 4 U202054321 Rose 89 94 85 100 U202056789 Tom 12 34 56 78 U202012345 Jack 99 100 80 96 U202098765 Jerry 98 76 54 32 2 3 U202054321 1 66 4 5 | U202056789 Tom 12 34 56 78  U202098765 Jerry 98 76 54 32  U202054321 Rose 89 94 85 100  U202012345 Jack 99 100 80 96  U202056789 Tom 45.00  U202098765 Jerry 65.00  U202054321 Rose 86.25  U202012345 Jack 93.75  U202056789 Tom 180 45.00  U202098765 Jerry 260 65.00  U202054321 Rose 345 86.25  U202012345 Jack 375 93.75 |
| 用例三 | 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 | U123456789 Elio 34 56 78 90  U987654321 Gray 0 0.00  U202073456 Red 245 61.25  U123456789 Elio 301 75.25  U000000000 Black 400 100.00 |

4思路：设计反转函数，使得链表进行反转，分奇偶取出原链表的一半，反转后半链表，与前半链表的数据进行比对

图3.7编程题四的流程图





源程序如下：

typedef struct c\_node

{

char data;

struct c\_node \*next;

} C\_NODE;

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

{

C\_NODE \*h=NULL, \*tail;

int i;

h = tail = (C\_NODE \*)malloc(sizeof(C\_NODE));

h->data = s[0];

for (i = 1; s[i] != '\0'; i++)

{

tail->next = (C\_NODE \*)malloc(sizeof(C\_NODE));

tail = tail->next;

tail->data = s[i];

}

tail->next = NULL;

\*headp = h;

}

C\_NODE \*reverse\_(C\_NODE \*head)//对于链表进行反转

{

C\_NODE \*newHead = NULL;

C\_NODE \*node;

while (head != NULL)

{//对之前的链表做头删

node = head;

head = head->next;

//对新链表做头插

node->next = newHead;

newHead = node;

}

return newHead;

}

void judgePalindrome(C\_NODE \*head)

{

int len = 0, i;

C\_NODE \*p, \*head1 = head, \*head2, \*a, \*b;

int flag = 1;

for (p = head; p; p = p->next)//判断出链表的长度

len++;

a = head1;

for (i = 0; i < len / 2 - 1; i++)//取一半的链表

a = a->next;

if (!(len % 2))

b = a->next;

else

b = a->next->next;

head2 = b;

a->next = NULL;

head2 = reverse\_(head2);//反转一半链表

for (i = 0, a = head1, b = head2; i < len / 2; i++, a = a->next, b = b->next)

if (a->data != b->data)

flag = 0;

if (flag)

printf("true");

else

printf("false");

}

图3.8编程题四的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | ppoqq | false |
| 用例二 | A man a plan aca nalp a nam A | true |
| 用例三 | ======== | true |

7思路：采用交换节点指针域的方法进行排序，排序采用选择排序法，采用二重循环，增加一个头节点，对于头节点空，通过循环找到最小的节点，然后使头节点指向该节点，循环到最后，完成排序。

图3.9编程题七的流程图



源程序如下：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct information{

char num[15];

char name[10];

int yinyu;

int shuxue;

int wuli;

int cyuyan;

float average;

int sum;

struct information \*next;

};

void paixu(struct information \*\*head)//采用交换节点的方式进行排序

{

struct information \*c,\*b,\*a,\*t,\*p,\*q;

p=\*head;

while(p->next!=NULL)//计算出该学生的总成绩和平均成绩

{

p->sum=p->yinyu+p->shuxue+p->wuli+p->cyuyan;

p->average=p->sum\*25;

p->average+=0.5;

p->average=(int )p->average;

p->average/=100;

p=p->next;

}

int i=0;

p=\*head;

while(p->next!=NULL)

{

p=p->next;i++;

}

struct information \*d;

d=(struct information\*)malloc(sizeof(struct information));

d->next=\*head;//增加一个头节点

for(int i1=0;i1<i-1;i1++)

{

a=d;

p=d->next;

for(int i2=0;i2<i-i1-1;i2++)

{

q=p->next;

b=q->next;

if(p->sum>q->sum)

{

q->next=p;

p->next=b;

c=p;

p=q;

q=c;

a->next=p;

}

if(q->next==NULL)

break;

else

{

a=a->next;

p=a->next;

}

}

}

\*head=d->next;

}

void shuru(struct information \*head,int n)

{

int i;

struct information \*p;

p=head;

while(p->next!=NULL)

{

p=p->next;

}

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

{

if(i<n-1)

{

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

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

scanf("%d %d %d %d",&(\*p).yinyu,&(\*p).shuxue,&(\*p).wuli,&(\*p).cyuyan);

p->next=(struct information\*)malloc(sizeof(struct information));

p=p->next;

}

else {//最后一个学生特殊化

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

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

scanf("%d %d %d %d",&(\*p).yinyu,&(\*p).shuxue,&(\*p).wuli,&(\*p).cyuyan);

p->next=(struct information\*)malloc(sizeof(struct information));

p=p->next;

p->next=NULL;

}

}

}

void shuchu(struct information \*head)

{

struct information \*p;

p=head;

while(p->next!=NULL)

{

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

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

printf("%d %d %d %d\n",p->yinyu,p->shuxue,p->wuli,p->cyuyan);

p=p->next;

}

}

void xiugai(struct information \*head)

{

struct information \*p;

struct information tell;//修改的学生信息

scanf("%s",tell.num);

int xuan;

scanf("%d",&xuan);//修改的科目

int gai;

scanf("%d",&gai);//最终分数

p=head;

while(strcmp(p->num,tell.num)!=0)//字符串的比较

{

p=p->next;

}

switch(xuan)

{

case 1:p->yinyu=gai;break;

case 2:p->shuxue=gai;break;

case 3:p->wuli=gai;break;

case 4:p->cyuyan=gai;break;

}

}

void jisuan(struct information \*head)

{

struct information \*p;

p=head;

while(p->next!=NULL)

{

p->sum=p->yinyu+p->shuxue+p->wuli+p->cyuyan;

p->average=p->sum\*25;

p->average+=0.5;

p->average=(int )p->average;

p->average/=100;

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

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

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

p=p->next;

}

}

void shuchu2(struct information \*head)

{

struct information \*p;

p=head;

while(p->next!=NULL)

{

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

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

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

p=p->next;

}

}

int main()

{

int choice;

scanf("%d",&choice);//菜单选项

int n;

struct information \*head,\*p;

head=(struct information\*)malloc(sizeof(struct information));//动态存储

head->next=NULL;

while(choice>0)//实现菜单选择功能

{

if(choice==1)//实现对于学生信息输入

{

scanf("%d",&n);//输入学生人数

shuru(head,n);

paixu(&head);

}

else if(choice==2)//实现对于学生信息输出

{

shuchu(head);

}

else if(choice==3)//实现成绩修改

{

xiugai(head);

}

else if(choice==4)

{

jisuan(head);

}

else if(choice==5)

{

shuchu2(head);

break;

}

scanf("%d",&choice);

}

return 0;

}

图3.10编程题七的测试用例

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | 1 2 U202054321 Rose 89 94 85 100 U202012345 Jack 99 100 80 96 2 3 U202054321 1 66 4 5 | U202054321 Rose 89 94 85 100  U202012345 Jack 99 100 80 96  U202054321 Rose 86.25  U202012345 Jack 93.75  U202054321 Rose 345 86.25  U202012345 Jack 375 93.75 |
| 用例二 | 1 4 U202054321 Rose 89 94 85 100 U202056789 Tom 12 34 56 78 U202012345 Jack 99 100 80 96 U202098765 Jerry 98 76 54 32 2 3 U202054321 1 66 4 5 | U202056789 Tom 12 34 56 78  U202098765 Jerry 98 76 54 32  U202054321 Rose 89 94 85 100  U202012345 Jack 99 100 80 96  U202056789 Tom 45.00  U202098765 Jerry 65.00  U202054321 Rose 86.25  U202012345 Jack 93.75  U202056789 Tom 180 45.00  U202098765 Jerry 260 65.00  U202054321 Rose 345 86.25  U202012345 Jack 375 93.75 |
| 用例三 | 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 | U123456789 Elio 34 56 78 90  U987654321 Gray 0 0.00  U202073456 Red 245 61.25  U123456789 Elio 301 75.25  U000000000 Black 400 100.00 |

9思路：对于逆波兰表达式，创建一个栈push，用来存储输入的数据，将数字压入栈中，并且考虑数据为负数的情况，依次将输入的数据压入栈中，如果遇到空格，继续程序，当数据流中为-时，判断-后面的情况，如果为数字，则表明还是一个数据，压入栈中，如果-后面为空格，则表明是运算，调用pop函数，释放符号前的数据，将运算后的数据重新存储到栈中，同理遇到其他运算的符号做同样处理，最后栈中只有一个数据，即为运算的最终结果。

源程序代码如下：

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

struct number{

int x;

struct number \*next;

};

void push(struct number \*\*head,int n)//创建链表作为栈,将数字压入栈中

{

struct number \*h,\*tail;

h=tail=(struct number \*)malloc(sizeof(struct number));

if(\*head==NULL)

{

h=NULL;

tail->x=n;

tail->next=h;

h=tail;

\*head=h;

}

else

{

h=\*head;

tail->x=n;

tail->next=h;

h=tail;

\*head=h;

}

}

int pop(struct number \*\*head,char c)

{

int m,n,t;

struct number \*p=\*head,\*q;

m=p->x,n=p->next->x;

if(c=='+') t=n+m;

else if(c=='-') t=n-m;

else if(c=='\*') t=n\*m;

else if(c=='/') t=n/m;

q=p->next;

\*head=q->next;

free(p),free(q);

return t;

}

void display(struct number \*head)

{

struct number \*p;

p=head;

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

}

int main()

{

struct number \*head=NULL;

char c,c1;

int n,t;

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

{

if(c==' ')continue;

if(isdigit(c)||c=='-')

{

if(c=='-')

{

c1=getchar();

ungetc(c1,stdin);

if(!isdigit(c1))

{

t=pop(&head,c);

push(&head,t);

break;

}

}

ungetc(c,stdin);

scanf("%d",&n);

push(&head,n);

continue;

}

else if(c=='+'||c=='-'||c=='\*'||c=='/')

{

t=pop(&head,c);

push(&head,t);

}

}

display(head);

return 0;

}

图3.11编程题九的测试用例

|  |  |  |
| --- | --- | --- |
| 测试输入 | 测试输入 | 预期输出 |
| 用例一 | 2 1 + 3 \* | 9 |
| 用例二 | 4 13 5 / + | 6 |
| 用例三 | 10 6 9 3 + -11 \* / \* 17 + 5 + | 22 |

## 3.4小节

这次的实验，初步掌握了结构的说明与应用，结构的指针，结构数组以及基于结构的链表的使用，加强了对于动态分配的运用，掌握自引用结构，单向结构链表的创建，历遍，增加，删除，修改的方法，但是在运用的时候还有所不熟练，需要向同学老师请教，在完善程序后，进行多次修改，推算。但是仍然有不完善之处。

# 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

........