

**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： U202015360**

**姓 名： 胡沁心**

**指导教师： 卢 萍**

**报告日期： 2021.1.5**

**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 5](#_Toc404837926)

[1.4小结 25](#_Toc404837928)

[**2数组程序设计实验 26**](#_Toc404837934)

[2.1程序改错与跟踪调试 26](#_Toc404837935)

[2.2程序完善与修改替换 27](#_Toc404837936)

[2.3程序设计 32](#_Toc404837926)

[2.4小结 53](#_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 55](#_Toc404837940)

[3.3程序设计 59](#_Toc404837926)

[3.4小结 84](#_Toc404837942)

[**参考文献 85**](#_Toc404837957)

# 

# 1 流程控制实验

## 1.1 程序改错

#include<stdio.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Ctr+2结束\n");  
  while(scanf ("%d", &x) !=EOF){

for(i=2,k=x>>1;i<=k;i++) /\*flag未初始化，加flag=0\*/  
{

if(!x%i) /\*!优先级高于%，x%i外加括号\*/  
  flag=1;  /\*if语句未加{ }\*/

break;

}  
if(flag==1)

printf("%d是合数\n",x)；

else printf("%d不是合数\n",x);

}

return 0;

}

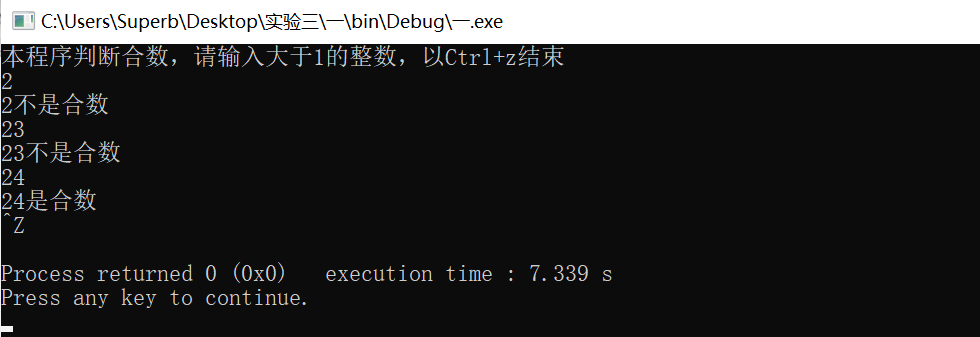


图1-1 程序改错的运行结果

## 1.2 程序修改替换

原程序：从2循环到x/2，用x除以i，如果有整除则为合数，flag=1，退出循环。flag=0则不是合数。

(1)

#include <stdio.h>

#include <stdlib.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Ctr+2结束\n");

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

for(i=2,k=x>>1,flag=0;i<=k;i++)

{

if(!(x%i))

flag+=1;

}

if(flag>=1)

printf("%d是合数\n",x);

else printf("%d不是合数\n",x);

}

return 0;

}

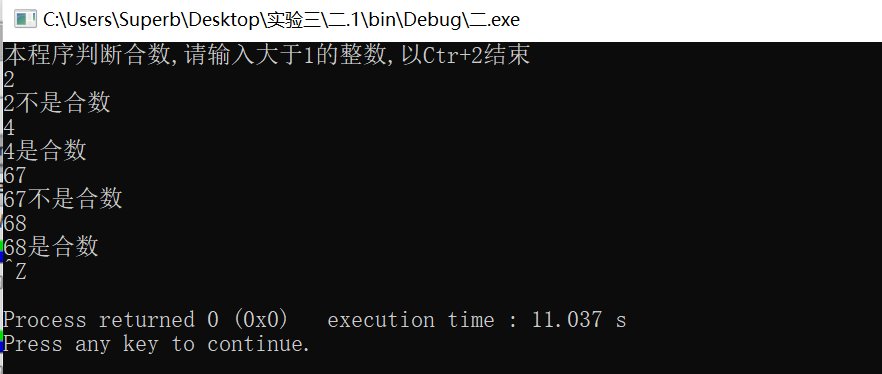


图1-2 程序替换修改1的运行结果

(2)

#include <stdio.h>

#include <stdlib.h>

int main()

{

int i,x,k,flag;

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){

k=x>>1;

i=2;

flag=0;

do{

if(!(x%i))

flag=1;

i++;

}while(i<=k);

if(x==2) printf("2不是合数\n");

else if(flag==1) printf("%d是合数\n",x);

else printf("%d不是合数\n",x);

}

return 0;

}

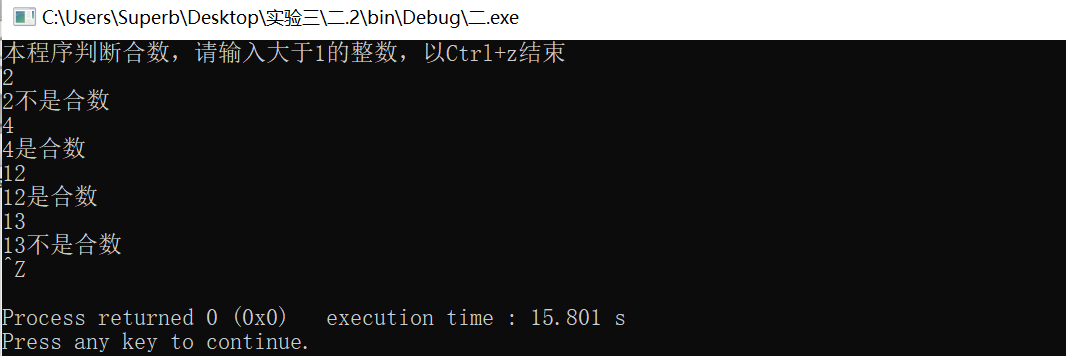


图1-3 程序替换修改2的运行结果

(3)

#include <stdio.h>

#include <stdlib.h>

int main()

{

int x,y,z,i,k,flag1,flag2,flag3;

printf("本程序判断合数,请输入大于1的整数,以Ctr+2结束\n");

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

{

y=x/10;

z=x/100;

for(i=2,k=x>>1,flag1=0;i<=k;i++)

{

if(!(x%i))

{

flag1=1;

break;

}

}

for(i=2,k=y>>1,flag2=0;i<=k;i++)

{

if(!(y%i))

{

flag2=1;

break;

}

}

for(i=2,k=z>>1,flag3=0;i<=k;i++)

{

if(!(z%i))

{

flag3=1;

break;

}

}

if(flag1==1&&flag2==1&&flag3==1)

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

}

return 0;

}

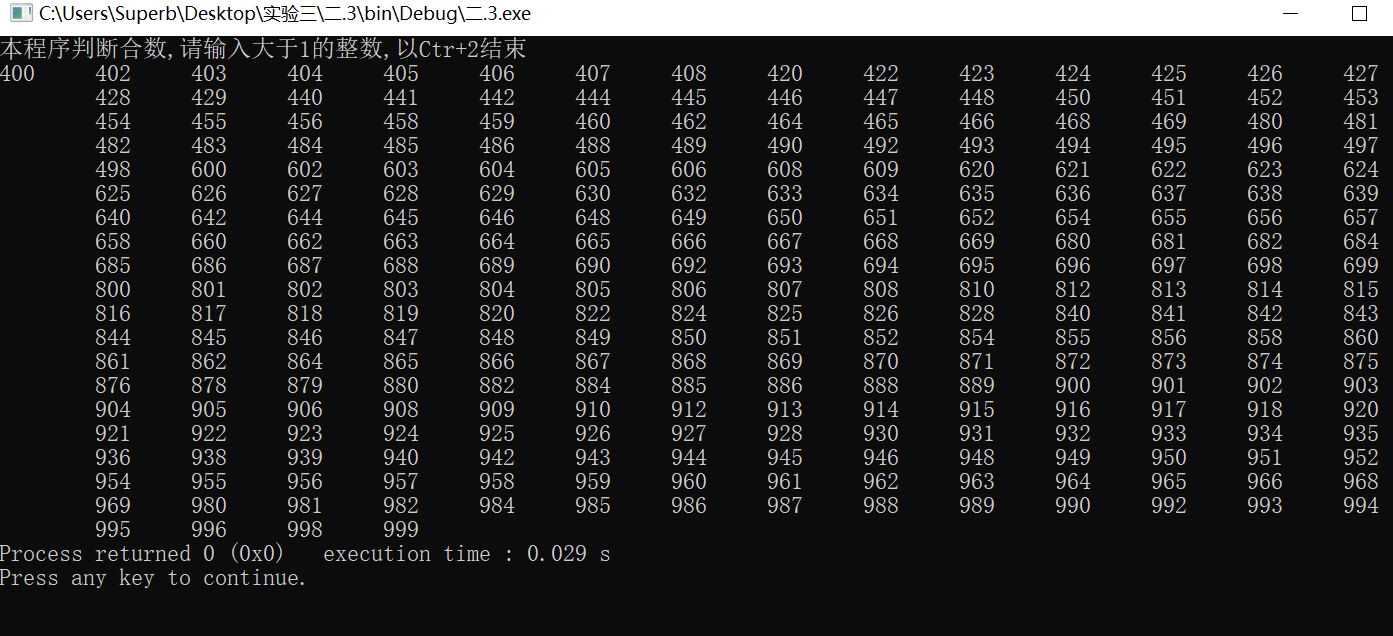


图1-4 程序替换修改3的运行结果

## 1.3 程序设计

1.

方法一) if语句

(1)算法步骤

1.输入工资x

2.判断x的范围

3.计算税y

4.输出税y

(2)解题清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

int x,y;

scanf("%d",&x);

if(x<1000)

y=0;

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

y=0.05\*(x-1000);

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

y=0.05\*1000+0.1\*(x-2000);

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

y=(0.05+0.1)\*1000+0.15\*(x-3000);

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

y=(0.05+0.1+0.15)\*1000+0.2\*(x-4000);

else

y=(0.05+0.1+0.15+0.2)\*1000+0.25\*(x-5000);

printf("%d",y);

return 0;

}

(3)测试

表1-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入x | 预期输出 |
| 用例一 | 1500（1000<x<2000） | 25 |
| 用例二 | 3500（3000<x<4000） | 225 |
| 用例三 | 5500（x>5000） | 625 |

方法二) switch语句

(1)算法步骤

1.输入工资x

2.计算x/1000的值

3.执行switch语句，计算税y的值

4.输出税y

(2)解题清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

int x,y;

scanf("%d",&x);

switch(x/1000)

{

case 0: y=0; break;

case 1: y=0.05 \* (x-1000); break;

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

case 3: y=(0.05+0.1) \* 1000+0.15 \* (x-3000); break;

case 4: y=(0.05+0.1+0.15) \* 1000+0.2 \* (x-4000); break;

default:y=(0.05+0.1+0.15+0.2) \* 1000+0.25 \* (x-5000); break;

}

printf("%d",y);

return 0;

}

(3)测试

表1-2 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入x | 预期输出 |
| 用例一 | 1500（x/1000=1） | 25 |
| 用例二 | 3500（x/1000=3） | 225 |
| 用例三 | 5500（x/1000>=5） | 625 |

表1-1，1-2中测试用例的运行结果如图1-5、图1-6和图1-7所示。



图1-5 编程题1的测试用例一的运行结果

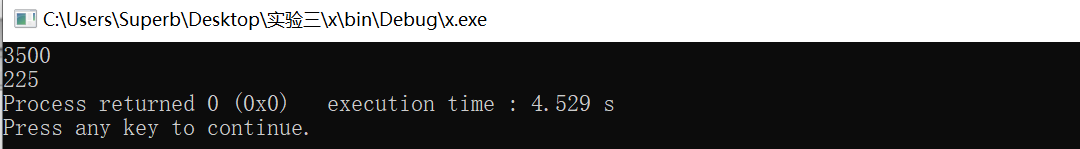


图1-6 编程题1的测试用例二的运行结果

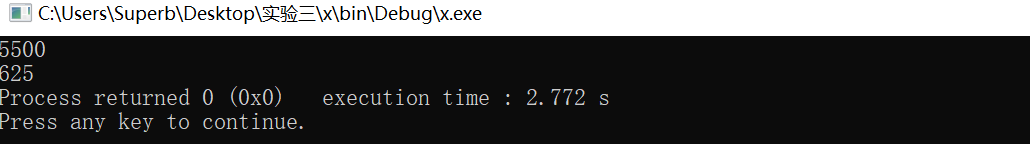


图1-7 编程题1的测试用例三的运行结果

2.

(1)流程图



(2)程序清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

int x,y,a,b,c;

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

{

y=x\*x;

a=y%10;

b=(y-a)%100;

c=(y-b-a)%1000;

if((a+b+c)==x)

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

}

return 0;

}

(3)测试

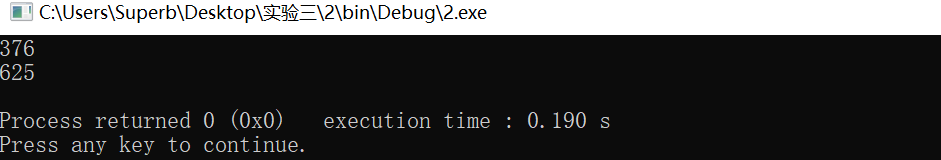


图1-8 编程题2的运行结果

3.

(1)算法步骤

1.输入n

2.输入数组a[n]

3.i从0循环到n-1，将a[i]和a[i+1]差的绝对值储存到数组b[i]中

4.i从0循环到n-2，将b[i]与b[i+1]进行比较，max赋值为更大值

5.输出max

(2)程序清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n,i,max;

scanf("%d",&n);

int a[n];

int b[n-1];

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

{

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

}

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

{

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

}

for(i=0,max=b[0];i<n-2;i++)

{

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

max=b[i+1];

}

printf("%d",max);

return 0;

}

(3)测试

表1-3 编程题3的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 测试输入 | | 预期输出 |
| n | a[n] |
| 用例一 | 4 | 1 3 6 8 | 3 |
| 用例二 | 5 | 1 6 3 8 2 | 6 |
| 用例三 | 3 | 2 5 1 | 4 |

表1-3中测试用例的运行结果如图1-9、图1-10和图1-11所示。

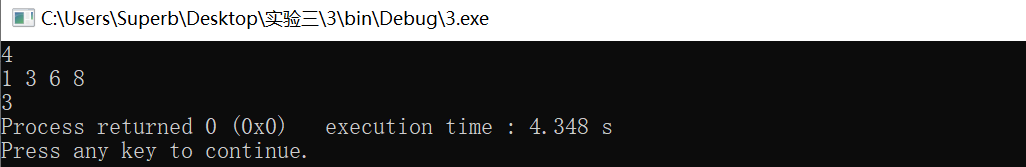


图1-9 编程题3的测试用例一的运行结果

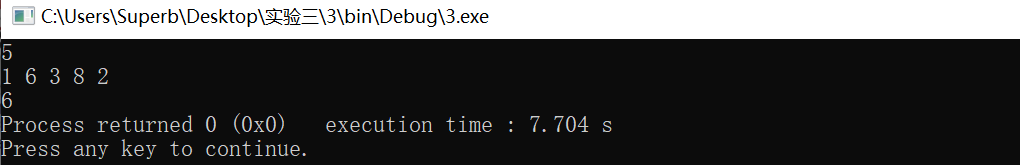


图1-10 编程题3的测试用例二的运行结果

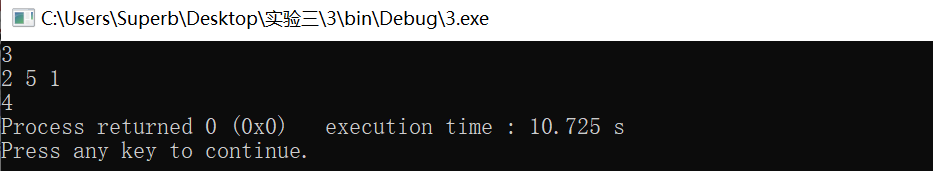


图1-11 编程题3的测试用例三的运行结果

4.

(1)流程图



(2)程序清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

char s[10000];

int i=0,j,count;

gets(s);

count=strlen(s);

while(i<count)

{

if(s[i]==32&&s[i+1]==32)

{

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

s[j]=s[j+1];

}

else i++;

}

printf("%s",s);

return 0;

}

(3)测试

表1-4 编程题4的测试数据

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 测试输入 | | 预期输出 |
| s | count |
| 用例一 | as df gh hj kl | 17 | as df gh hj kl |
| 用例二 | qw er ty we | 13 | qw er ty we |
| 用例三 | hello,world . | 18 | Hello, world . |

表1-4中测试用例的运行结果如图1-12、图1-13和图1-14所示

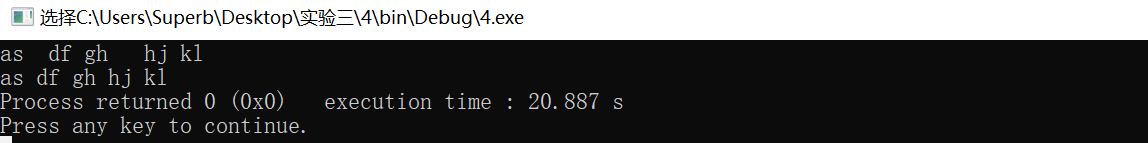


图1-11 编程题4的测试用例一的运行结果

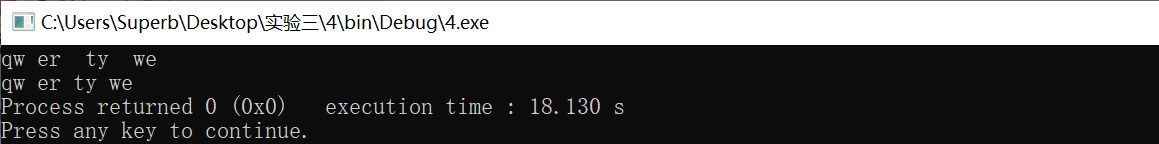


图1-12 编程题4的测试用例二的运行结果

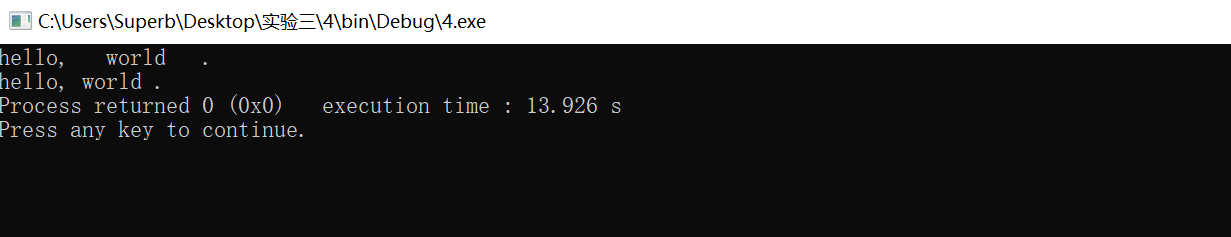


图1-13 编程题4的测试用例三的运行结果

5.

(1)算法步骤

1.i从1循环到11，j从1 循环到i，确定等于1 的项

2.下一行的数等于两肩上数的和

3.在数字之前打出10-i个空格

4.打印出数字，每个数字占4个字符位置

(2)程序清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

int i,j,a[11][11],k;

for(i=1;i<11;i++)

{

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

if(j==1||i==j)

a[i][j]=1;

else

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

}

for(i=1;i<11;i++)

{

for(k=1;k<=10-i;k++)

printf(" ");

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

printf("%4d",a[i][j]);

printf("\n");

}

return 0;

}

(3)测试

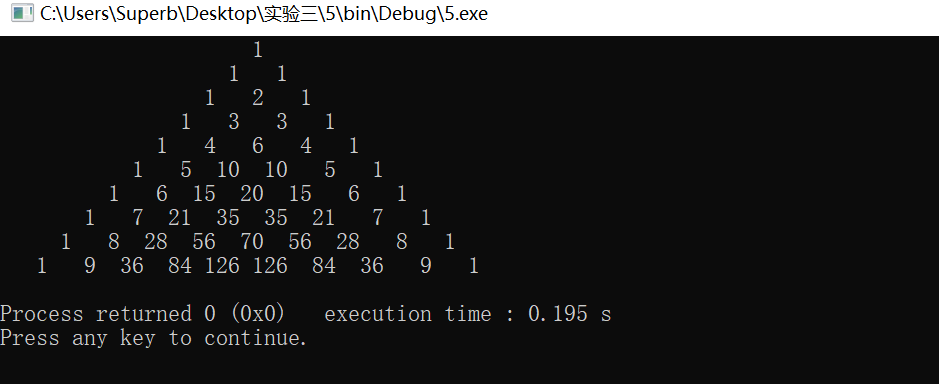


图1-14 编程题5的运行结果

6.

(1)算法步骤

1.输入m

2.判断m的大小，大于3则进入for循环

3.从i=2循环到2^i-1大于m

4.判断2的i次方减1是否为质数

5.是质数则打印M[i]=2^i-1

(2)程序清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

int i,j,k,x=2,flag;

long m;

scanf("%d",&m);

if(m>3)

{

for(i=2,k=0;(x\*2-1)<=m;i++)

{

x\*=2;

k=(x-1) >>1;

for(j=2,flag=0;j<=k;j++)

{

if((x-1)%j==0)

flag+=1;

}

if(flag==0)

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

}

}

else printf("错误");

return 0;

}

(3)测试

表1-5 编程题6的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入m | 预期输出 |
| 用例一 | 97 | M(2)=3  M(3)=7  M(5)=31 |
| 用例二 | 130 | M(2)=3  M(3)=7  M(5)=31  M(7)=128 |
| 用例三 | 2 | 错误 |

表3-6中测试用例的运行结果如图1-15、图1-16和图1-17所示。

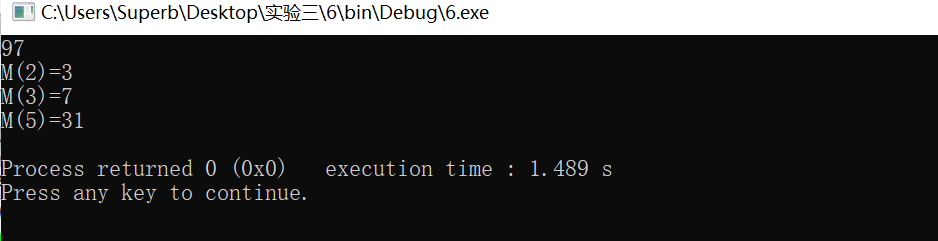


图1-15编程题6的用例一的运行结果

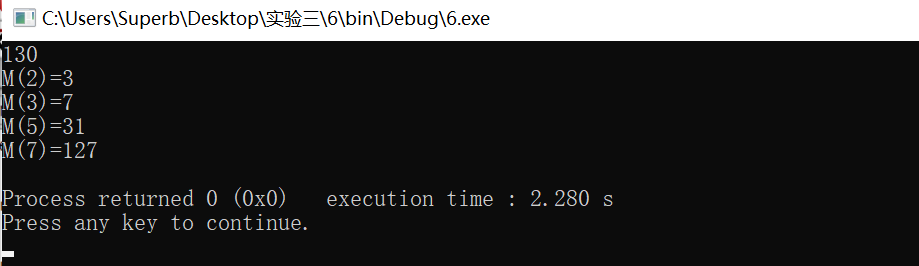


图1-16 编程题6的用例二的运行结果

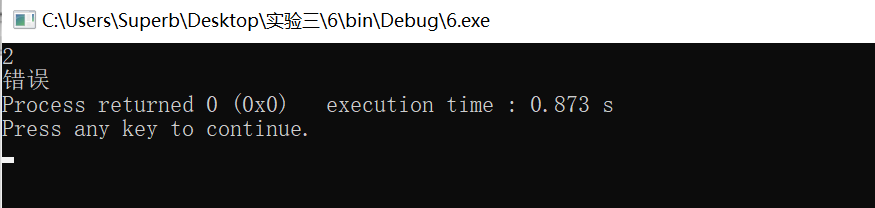


图1-17 编程题6的用例三的运行结果

10.

(1)算法步骤

1.输入m

2.判断m的范围，符合要求则进入while循环

3.x从1循环到m-2，赋值y为x+2

4.i从2分别循环到x/2，y/2，分别用x，y分别除以i，判断是否为质数

5.若x，y均为质数，输出(x,y)

6.退出while循环

(2)程序清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

long m,x,y;

int i,k,flag1,flag2;

scanf("%d",&m);

while(m>=4&&m<=1000000){

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

{

y=x+2;

for(i=2,k=x>>1,flag1=0;i<=k;i++)

{

if(!(x%i)){

flag1=1;

break;

}

}

for(i=2,k=y>>1,flag2=0;i<=k;i++)

{

if(!(y%i))

{

flag2=1;

break;

}

}

if(flag1==0&&flag2==0)

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

}

break;

}

return 0;

}

(3)测试

表1-6 编程题10的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入m | 预期输出 |
| 用例一 | 17 | (3,5)  (5,7)  (11,13) |
| 用例二 | 77 | (3,5)  (5,7)  (11,13)  (17,19)  (29,31)  (41,43)  (59,61)  (71,73) |
| 用例三 | 2 |  |

表1-6中测试用例的运行结果如图1-17、图1-18和图1-19所示

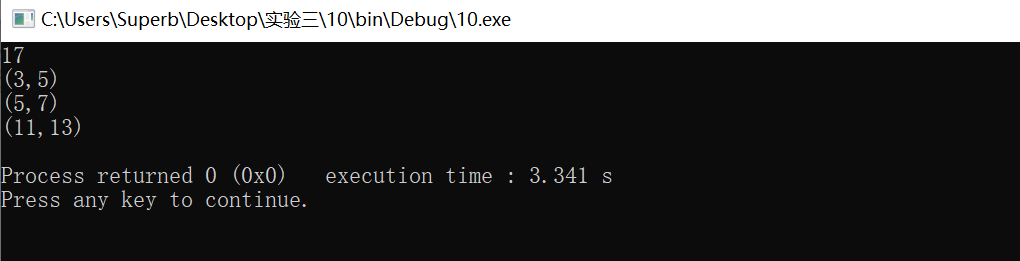


图1-17 编程题10的用例一的运行结果

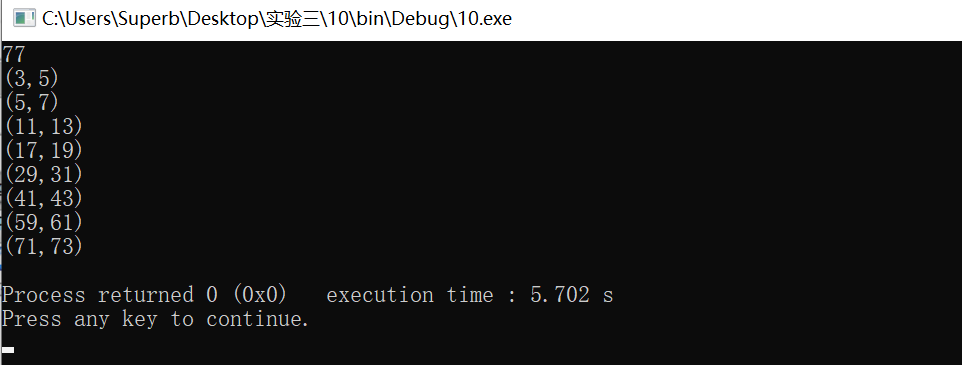


图1-18 编程题10的用例二的运行结果

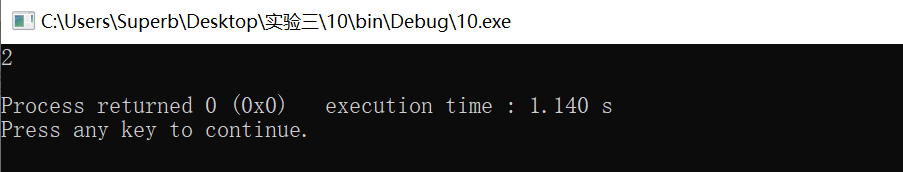


图1-19编程题10的用例三的运行结果

11.

(1)算法步骤

1.输入x

2.判断x的范围，符合要求则进入for循环

3.y从1234循环到98765

4.z=x\*y，判断z的范围是否符合要求，符合则将y赋给m，z赋给n

5.将y,z储存到数组a

6.判断a内有无数字重复

7.无重复则输出a的前五位/后五位=x

(2)程序清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

int x, i, j, a[10];

unsigned long y, z;

scanf("%d", &x);

if(x >= 2 && x <= 79)

{

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

{

a[0] = '\0';

int k = 1;

z = x \* y;

if (z >= 1234 && z <= 98765)

{

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

{

a[i] = m % 10;

m /= 10;

}

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

{

a[i] = n % 10;

n /= 10;

}

for (i = 0; i < 9; i++)

{

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

{

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

k = 0;

}

}

if (k == 1)

{

for (i = 5; i < 10; i++)

{

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

}

printf("/");

for (i = 0; i < 5; i++)

{

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

}

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

}

}

}

}

return 0;

}

(3)测试

表1-7 编程题11的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入x | 预期输出 |
| 用例一 | 32 | 75168/02349=32 |
| 用例二 | 3 | 17469/05823=3  17496/05832=3  50382/16794=3  53082/17694=3  61749/20583=3  69174/23058=3  91746/30582=3  96174/32058=3 |
| 用例三 | 56 |  |

表1-7中测试用例的运行结果如图1-20、图1-21和图1-22所示。

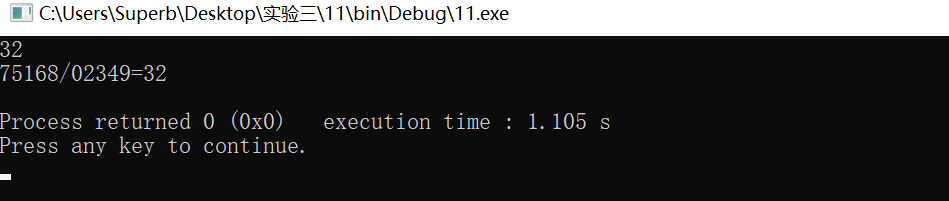


图1-20 编程题11的用例一的运行结果

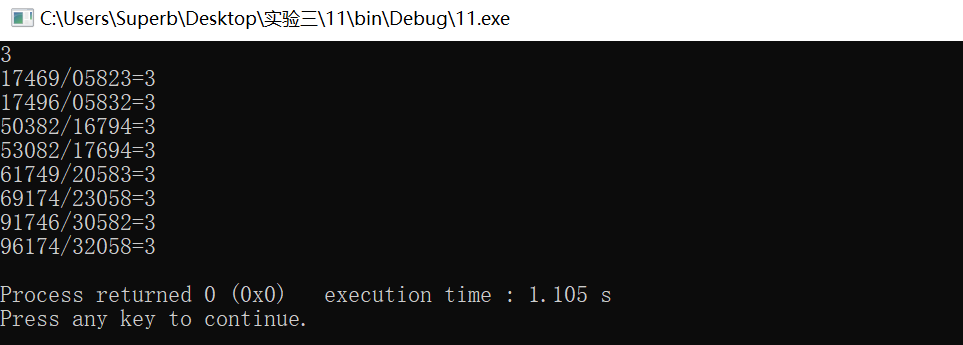


图1-21 编程题11的用例二的运行结果

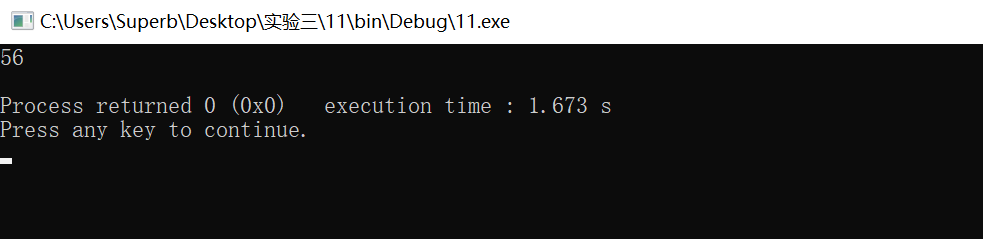


图1-22 编程题11的用例三的运行结果

## 1.4 小结

通过这次实验作业我学会了如何改变输出数据的排布形状、如何使用绝对值和画流程图。**2 数组程序设计实验**

## 2.1 程序改错与跟踪调试

#include<stdio.h>

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

void strdelc(char[], char);

int main(void)

{

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

scanf("%s%s",b,a); /\*输入字符串\*/

scanf("%c",&x); /\*输入要删除的字符\*/

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

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

return 0;

}

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

{

int i = 0, j = 0;

while (t[i] != '\0') /\*计算字符串t的长度\*/

i++;

while (s[j] != '\0') /\*计算字符串s的长度\*/

{

t[i] = s[j]; /\*s接在t后\*/

i++;

j++;

}

t[i] = '\0'; /\*字符串t最后一位为null\*/

}

void strdelc(char t[], char c)

{

int j,k;

for (j = 0; t[j] != '\0'; j++)

{

if (t[j] == c)

{

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

t[k] = t[k + 1]; /\*删去指定字符，后一个补上\*/

}

}

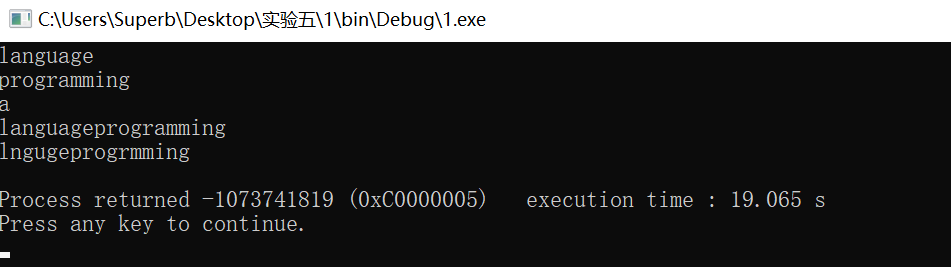


图2-1 程序改错的运行结果

## 2.2 程序完善与修改替换

原程序：从2循环到x/2，用x除以i，如果有整除则为合数，flag=1，退出循环。flag=0则不是合数。

(1)

#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); /\*计算字符串长度\*/

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

{

if(s[r]!='\0')

{

s[w++]=s[r]; /\*保存字符串\*/

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

{

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

s[i] = '\0'; /\*删去相同字符\*/

}

}

}

s[w]=’\0’;

}

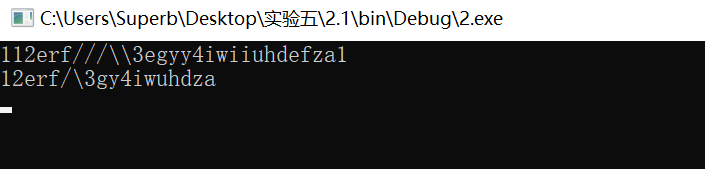


图2-2 程序替换修改1的运行结果

(2)

#include <stdio.h>

#include <stdlib.h>

int main()

{

char s[200];

scanf("%s",s);

int i=0,j,k,len;

len=strlen(s);

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

{

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

{

for(k=j+1;k<len;k++)

{

if(s[j]==s[k])

s[k]='\0';

}

}

i++;

}

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

{

if(s[i]!='\0')

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

}

return 0;

}

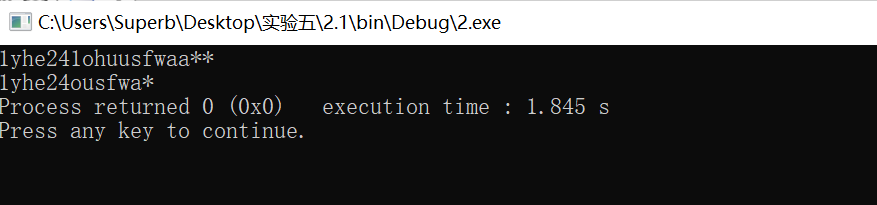


图2-3 程序替换修改1的运行结果

(3)

# 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("%6d\n", a[0]);

return 0;

}

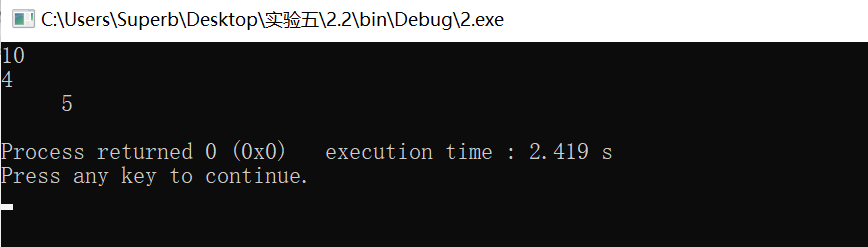


图2-4 程序替换修改2的运行结果

(4)

#include <stdio.h>

#include <stdlib.h>

int main()

{

int m, n, i, j, k, x;

scanf("%d%d", &m, &n); /\*输入m,n\*/

int a[1000];

x = m; /\*人数储存到x中\*/

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

a[i] = i + 1;

i = 1; /\*i赋值为1\*/

while (x > 1)

{

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

{

if (a[j] != '\0') /\*如果j+1未出局\*/

{

if (i % n == 0) /\*如果i整除n\*/

{

a[j] = '\0'; /\*第j+1个人出局\*/

x--; /\*人数减1\*/

}

i++; /\*i加1\*/

}

else

continue; /\*如果已出局则跳过\*/

}

}

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

{

if (a[i] != 0)

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

}

return 0;

}

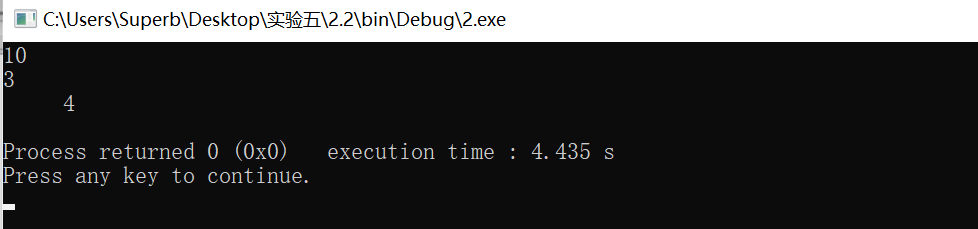


图2-5 程序替换修改2的运行结果

## 2.3 程序设计

1.

(1)算法步骤

1.输入n

2.如果n为非负数，除以2取余数，n再除2，余数存入数组

3.如果n为负数，取反减1，重复步骤2

4.输出n的二进制码

(2)解题清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

int m,n,i,a[32]={0}; /\*定义数组a全为0\*/

scanf("%d",&n);

if(n>=0) /\*若n为非负数\*/

{

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

{

a[i]=n%2; /\*除以2取余二进制存入数组a\*/

n/=2;

}

for(i=31;i>=0;i--){

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

if(i%4==0)

printf(" ");

}

}

else /\*若n为负数\*/

{

m=-n-1; /\*取补码\*/

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

{

a[i]=m%2;

m/=2;

}

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

{

if(i==31)

a[i]=1; /\*首位为1表示负数\*/

else if(a[i]==0) /\*各位取反\*/

a[i]=1;

else a[i]=0;

}

for(i=31;i>=0;i--)

{

printf("%d",a[i]); /\*输出二进制码\*/

if(i%4==0&&i>0)

printf(" ");

}

}

return 0;

}

(3)测试

表2-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入x | 预期输出 |
| 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 | 5 | 0000 0000 0000 0000 0000 0000 0000 1001 |

表2-1中测试用例的运行结果如图2-6、图2-7和图2-8所示

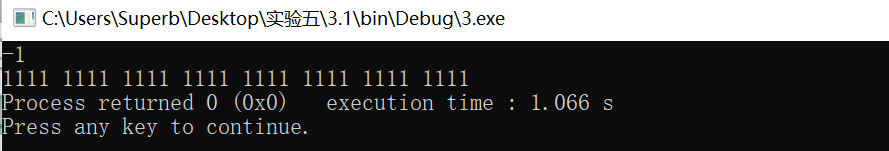


图2-6 编程题2的运行结果1

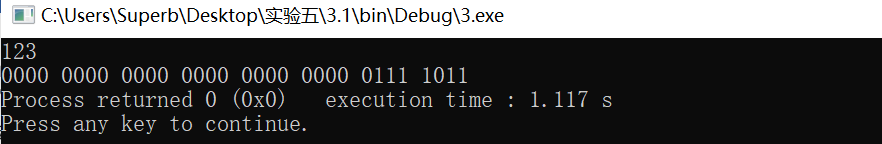


图2-7 编程题2的运行结果2

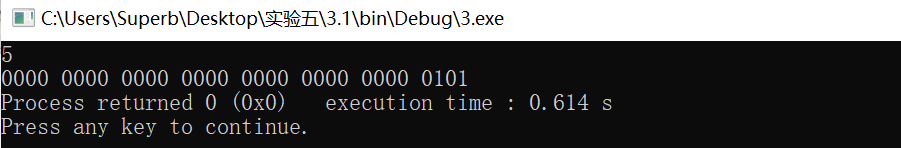


图2-8 编程题2的运行结果3

2.

(1)算法步骤

1.定义结构student，内容为姓名和成绩。

2.定义结构数组

3.输入功能编号依次

4.功能编号1: 输入人数，姓名,成绩

5.功能编号2: 需要排序，则将成绩和姓名字母顺序进行比较，将结构数组交换。

6.功能编号3: 输出结果。

(2)程序清单

#include <stdio.h>

#include <stdlib.h>

struct student

{

char name[10];

int score;

};

int main()

{

struct student s[10];

int i,j,n,x,y;

while(1)

{

scanf("%d",&x);

switch(x) /\*switch菜单\*/

{

case 1:

{

scanf("%d",&n);

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

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

break;

}

case 2:

{

struct student temp;

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

{

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

{

if(s[j].score<s[j+1].score) /\*比较成绩\*/

{

temp=s[j];

s[j]=s[j+1];

s[j+1]=temp; /\*交换\*/

}

}

}

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

{

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

{

int p=0;

if(s[j].score==s[j+1].score)

{

while(s[j].name[p]!='\0')

{

if(s[j].name[p]>s[j+1].name[p])/\*比较字母顺序\*/

{

temp=s[j];

s[j]=s[j+1];

s[j+1]=temp; /\*交换\*/

goto a;

}

else if(s[j].name[p]<s[j+1].name[p])

goto a;

p++; /\*字母往后移一位\*/

}

}

}

}

a:

break;

}

case 3:

{

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

printf("%s %d\n",s[i].name,s[i].score); /\*输出\*/

}

case 0:goto down;

}

}

down:

return 0;

}

(3)测试

表2-2 编程题2的测试数据

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 测试用例 | 测试输入 | | | | 预期输出 |
| x | n | name&score | y |
| 用例一 | 1 | 3 | aaa 90 b 100 cc 80 | 2 | b 100  a 90  c 80 |
| 用例二 | 1 | 3 | jerry 80 angle 89  hong 80 ming 87 | 2 | jerry 89  angle 87  hong 80  ming 80 |
| 用例三 | 1 | 3 | a 90 b 70 c 85 | 3 | a 90  b 70  c 80 |

表2-2中测试用例的运行结果如图2-9、图2-10和图2-11所示

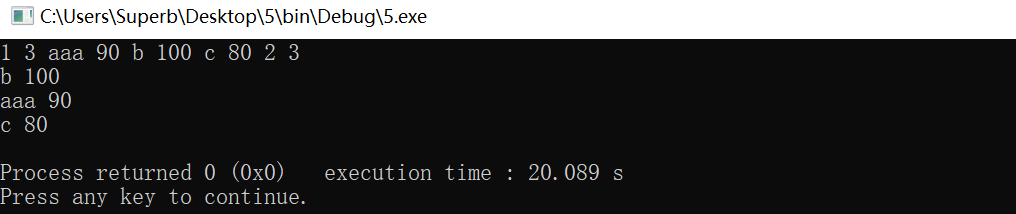


图2-9 编程题2的运行结果

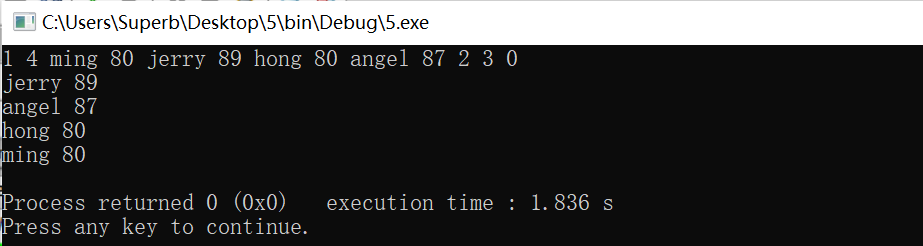


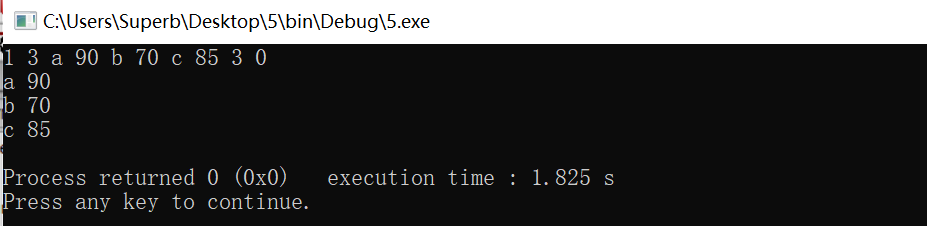
图2-10 编程题2的运行结果

图2-11 编程题2的运行结果

3.

(1)算法步骤

1.定义结构student，内容为姓名和成绩。

2.定义结构数组

3.输入功能编号依次

4.功能编号1: 输入人数，姓名,成绩

5.功能编号2: 需要排序，则将成绩和姓名字母顺序进行比较，将结构数组交换。

6.功能编号4: 二分查找。

(2)程序清单

#include <stdio.h>

#include <stdlib.h>

struct student

{

char name[10];

int score;

};

int main()

{

struct student s[10];

int i,j,n,x,y;

while(1){

scanf("%d",&x);

switch(x)

{

case 1:

{

scanf("%d",&n);

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

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

break;

}

case 2:

{

struct student temp;

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

{

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

{

if(s[j].score<s[j+1].score) /\*比较成绩\*/

{

temp=s[j];

s[j]=s[j+1];

s[j+1]=temp; /\*交换\*/

}

}

}

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

{

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

{

int p=0;

if(s[j].score==s[j+1].score)

{

while(s[j].name[p]!='\0')

{

if(s[j].name[p]>s[j+1].name[p])/\*比较字母顺序\*/

{

temp=s[j];

s[j]=s[j+1];

s[j+1]=temp; /\*交换\*/

goto a;

}

else if(s[j].name[p]<s[j+1].name[p])

goto a;

p++; /\*字母往后移一位\*/

}

}

}

}

a:

break;

}

case 4:

{

scanf("%d",&y);

int low=0, high=n-1, mid,flag=n/2; /\*flag为最大查找次数\*/

while(low<=high&&flag>0)

{

mid=(low+high)/2;

if(s[mid].score>y)

low=mid+1; /\*成绩大于查找值，low=mid+1\*/

else if(s[mid].score<y)

high=mid-1; /\*成绩大于查找值，high=mid-1\*/

else

{

printf("%s %d\n",s[mid].name,s[mid].score); /\*找到了\*/

low=mid+1;

}

flag--; /\*次数减1\*/

}

if(s[mid].score!=w)

printf("not found"); /\*查找完后不相等，没找到\*/

}

case 0:goto down;

}

}

down:

return 0;

}

(3)测试

表2-3 编程题3的测试数据

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 测试用例 | 测试输入 | | | | | | | 预期输出 |
| x | n | name &score | y | z | w | t |
| 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2-3中测试用例的运行结果如图2-12、图2-13和图2-14所示

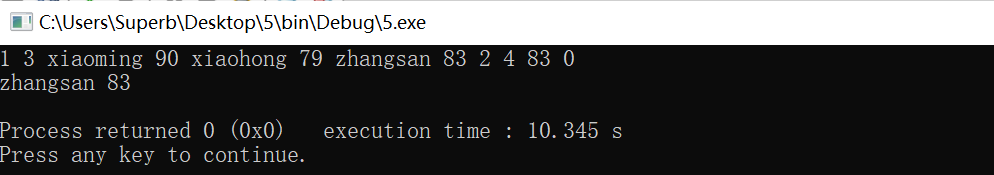


图2-12 编程题3的测试用例一的运行结果

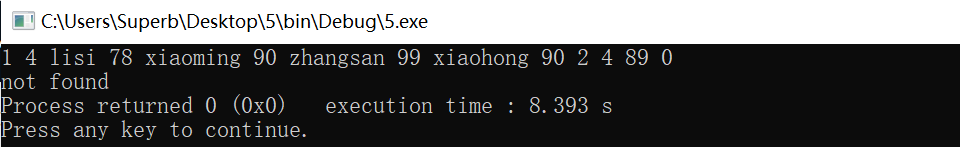


图2-13 编程题3的测试用例二的运行结果

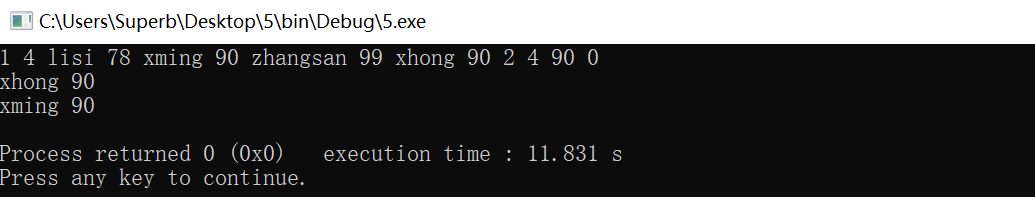


图2-14 编程题3的测试用例三的运行结果

5．

(1)算法步骤

1.输入字符串t，s

2.输入n

3.调用函数

3.将s插入位置的字符复制到t的末尾

4.将s要插入的位置替换位s的字符

5.输出t

(2)程序清单

#include <stdio.h>

#include <stdlib.h>

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

int main()

{

char t[100],s[100];

int n;

scanf("%s%s",t,s); /\*输入字符串t，s\*/

scanf("%d",&n); /\*输入插入的位置\*/

strnins(t,s,n);

printf("%s",t);

return 0;

}

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

int len1=0,len2=0,i;

while (t[len1++]); /\*计算字符串t的长度\*/

while (s[len2++]); /\*计算字符串s的长度\*/

len1--; /\*减去最后一位null\*/

len2--;

for (i=len1-1;i>n-1;i--)

t[i+len2]=t[i]; /\*将t从第n-1位开始的len1位复制到t末尾\*/

for(i=0;i<len2;i++)

t[i+n]=s[i]; /\*将t从第n-1位开始的len1位替换成s\*/

t[len1+len2]='\0'; /\*t最后一位封口\*/

return t;

}

(3)测试

表2-4 编程题5的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例 | 测试输入 | | | 预期输出 |
| t | s | n |
| 用例一 | iyou | love | 1 | iloveyou |
| 用例二 | comter | pu | 3 | computer |
| 用例三 | world | hello | 0 | helloworld |

表2-4中测试用例的运行结果如图2-15、图2-16和图2-17所示。

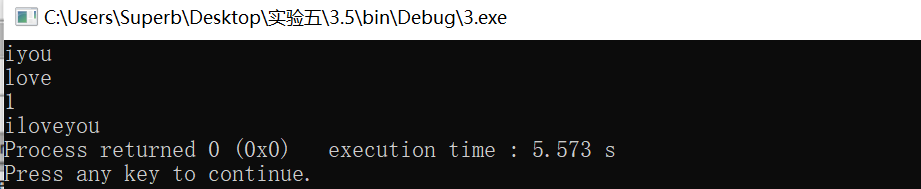


图2-15 编程题5的测试用例一的运行结果

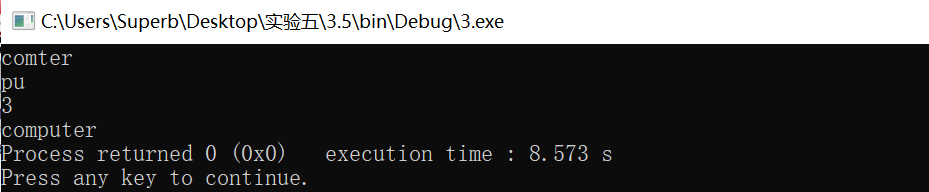


图2-16 编程题5的测试用例二的运行结果

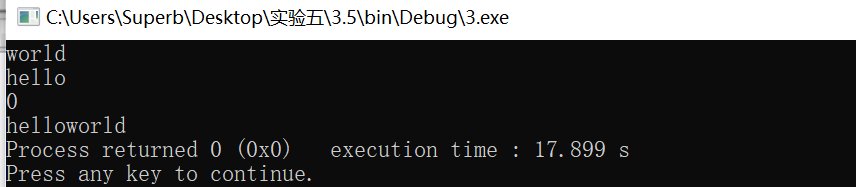


图2-17 编程题5的测试用例三的运行结果

6.

(1)流程图



(2)程序清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

int i, j, a[10][10], b[10][10], m, n;

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

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

{

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

{

scanf("%d", &a[i][j]); /\*输入二维数组a\*/

b[i][j] = a[i][j]; /\*将a储存到二维数组b\*/

}

}

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

{

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

{ /\*如果同行3个

if(a[i][j] == a[i][j+1]&&a[i][j+1]==a[i][j+2]) 数连续相等\*/

b[i][j] = b[i][j + 1] = b[i][j + 2] = 0; /\*全部消为0\*/

}

}

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

{

for (i = 0; i < m - 2; i++)

{ /\*如果同行3个

if(a[i][j] == a[i+1][j] && a[i+1][j]==a[i+2][j]) 数连续相等\*/

b[i][j] = b[i + 1][j] = b[i + 2][j] = 0; /\*全部消为0\*/

}

}

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

{

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

{

printf("%d", b[i][j]); /\*输出数组b\*/

if(j!=n-1)

printf(" ");

}

printf("\n");

}

return 0;

}

(3)测试

表2-5 编程题6的测试数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例 | 测试输入 | | | 预期输出 |
| m | n | a (b) |
| 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 | 3 | 4 3 1 2 3 4 5 3 6 7 3 2 4 5 | 1 2 0  4 5 0  6 7 0  2 4 5 |

表2-5中测试用例的运行结果如图2-18、图2-19和图2-20所示。

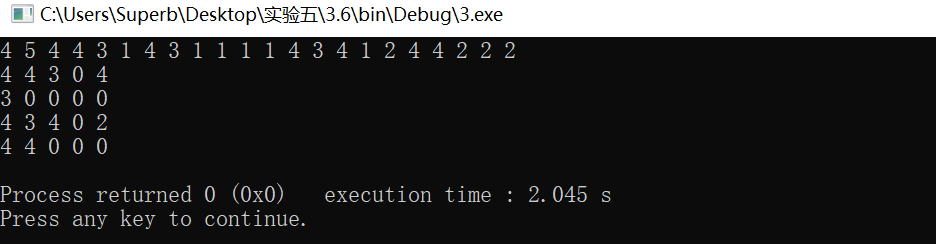


图2-18编程题6的用例一的运行结果

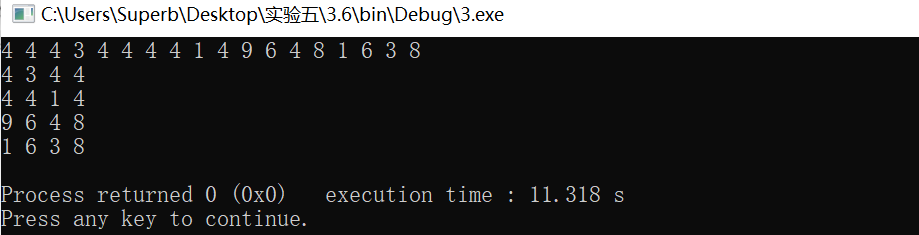


图2-19 编程题6的用例二的运行结果

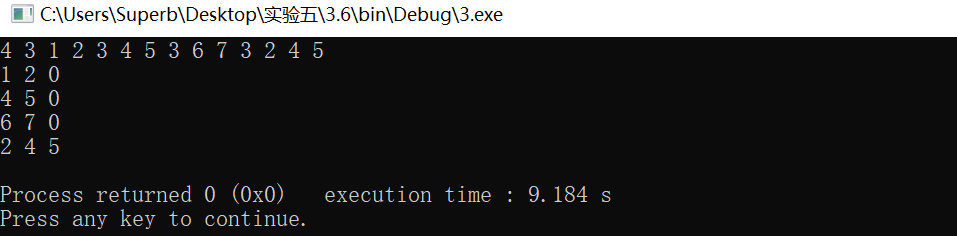


图2-20 编程题6的用例三的运行结果

10.

(1)流程图



(2)程序清单

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n,i,j,k=1,row,line;

scanf("%d",&n); /\*输入阶数n\*/

int a[n][n];

for(i=0,j=0;k<=n\*n;i++,j++)

{

for(row=j;row<n-j;row++,k++)

a[i][row]=k; /\*第i+1层上行赋值\*/

for(line=i+1;line<n-i;line++,k++)

a[line][n-j-1]=k; /\*第i+1层右行赋值\*/

for(row=n-j-2;row>=j;row--,k++)

a[n-i-1][row]=k; /\*第i+1层下行赋值\*/

for(line=n-i-2;line>=i+1;line--,k++)

a[line][j]=k; /\*第i+1层左行赋值\*/

}

if(n%2!=0) /\*假如是n为奇数,最中心的一个数要赋值\*/

a[(n+1)/2][(n+1)/2]=n\*n;

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

{

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

{

printf("%d",a[i][j]); /\*输出方阵\*/

if(j!=n-1)

printf(" ");

else

printf("\n");

}

}

return 0;

}

(3)测试

表2-7 编程题10的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入n | 预期输出 |
| 用例一 | 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 |
| 用例二 | 3 | 1 2 3  8 9 4  7 6 5 |
| 用例三 | 6 | 1 2 3 4 5 6  20 21 22 23 24 7  19 32 33 34 25 8  18 31 36 35 26 9  17 30 29 28 27 10  16 15 14 13 12 11 |

表2-7中测试用例的运行结果如图2-21、图2-22和图2-23所示

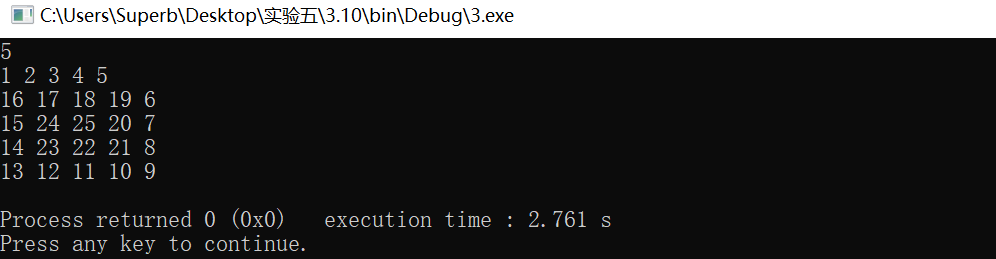


图2-20 编程题10的用例一的运行结果

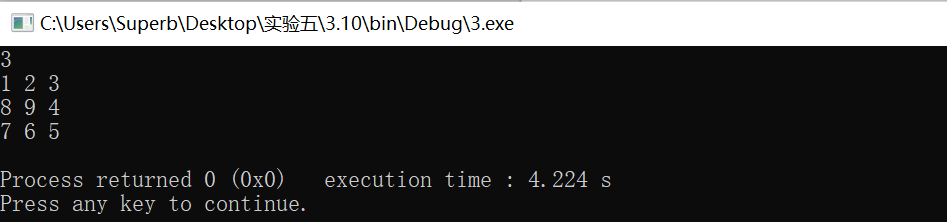


图2-21 编程题10的用例二的运行结果

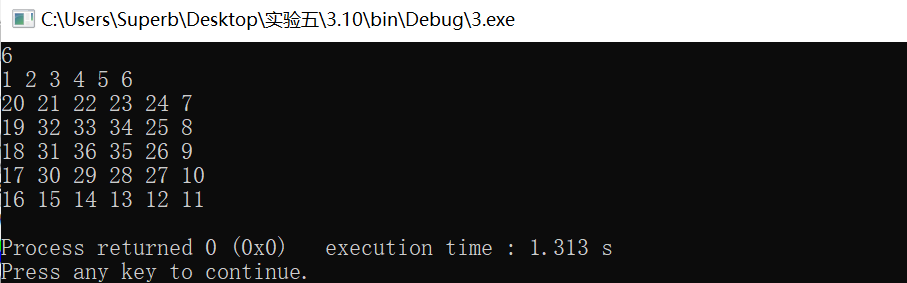


图2-22 编程题10的用例三的运行结果

## 2.4小结

通过这次实验作业我学会了如何改变输出数据的排布形状、如何使用绝对值和画流程图。自学并运用了结构数组的定义和使用。**3 结构与联合实验**

## 3.1 表达式求值的程序验证

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

int main()

{

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

struct T{

int x;

char c;

char \*t;

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

int i;

scanf("%d",&i);

switch(i)

{

case 1:printf("%d",(++p)->x);break;

case 2:p++,printf("%c",p->c);break;

case 3:\*p++->t,printf("%c",\*p->t);break;

case 4:printf("%c",\*(++p)->t);break;

case 5:printf("%c",\*++p->t);break;

case 6:printf("%c",++\*p->t);break;

}

return 0;

}

## 3.2 源程序修改替换

(1)

#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[10],i;

for(i=0;i<10;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;

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)

#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[10],i;

for(i=0;i<10;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;

if(p[0]==0);

else

{

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

tail=loc\_head;

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

while (\*p)

{

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

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

loc\_head->next=tail;

tail=loc\_head;

}

}

\*headp=loc\_head;

}

## 3.3 程序设计

1.

(1)算法步骤



(2)解题清单

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

void f0(int b); /\*声明8个函数\*/

void f1(int b);

void f2(int b);

void f3(int b);

void f4(int b);

void f5(int b);

void f6(int b);

void f7(int b);

struct bits{

int bit;

}s[8]; /\*定义结构体数组\*/

int main()

{

int n,i;

scanf("%d",&n); /\*输入十进制数字\*/

void(\*p\_fun[8])(int b); /\*定义函数指针数组，存入函数名\*/

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;

for(i=0;i<8;i++)

{

s[i].bit=n%2; /\*转化为二进制数，储存在结构体数组中\*/

n/=2;

if(s[i].bit) /\*当二进制位为1时，调用相应函数\*/

p\_fun[i](i);

}

return 0;

}

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);

}

(3)测试

表3-1 编程题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 测试输入 | 预期输出 |
| 用例一 | 123  (0111 1011) | 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  (1111 1111) | 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  (0000 0000) |  |

2.

(1)算法步骤

1.声明结构体链表

2.声明各功能编号的函数

3.给链表分配内存

4.输入功能编号，switch菜单选择

5.功能编号1: 1)输入人数n

2)指针指向头

3)输入数据到结构体中

4)指针指向下一个结点，分配内存

5)链表尾为NULL

6.功能编号2: 调用函数output

1) 指针遍历链表

2) 输出结点中的数据

7.功能编号3: 调用函数correct

1) 输入要改正的学生学号，科目，成绩

2) 指针遍历链表,查找相同学号

3) 若找到相同的学号，改正相应科目成绩

8.功能编号4: 调用函数average

1) 指针遍历链表

2）计算每位同学的平均分

3）遍历链表，输出学号，姓名，平均分

9.功能编号5: 调用函数sum

1) 遍历链表

2) 计算总分和平均分

3) 输出各项数据

(2)程序清单

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct stu{

char num[11];

char name[10];

int a;

int b;

int c;

int d;

struct stu \*next;

};

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

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

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

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

int main()

{

struct stu \*head=NULL;

int x;

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

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

while(1){

scanf("%d", &x);

switch(x) /\*菜单选择\*/

{

case 1:

{

int n,i;

scanf("%d",&n);

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

{

scanf("%s%s%d%d%d%d",p->num, p->name ,&p->a, &p->b, &p->c, &p->d);

p=(struct stu \*)malloc(sizeof(struct stu));/\*分配内存\*/

tail->next=p;

tail=p; /\*指向下一个结点形成链表\*/

}

tail->next=NULL; /\*结束链表\*/

break;

}

case 2:output(&head);break;

case 3:correct(&head);break;

case 4:average(&head);break;

case 5:sum(&head);goto down;

}

}

down:

return 0;

}

void output(struct stu \*\*headp){

struct stu \*p=\*headp;

while(p->next!=NULL){

printf("%s %s %d %d %d %d\n",p->num,p->name,p->a,p->b,p->c,p->d);

p=p->next; /\*指针指向下一个结点\*/

}

}

void correct(struct stu \*\*headp){

struct stu \*p=\*headp;

char num0[11];

int x,score;

scanf("%s",num0);

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

while(p->next!=NULL){

if(strcmp(num0,p->num)==0) /\*判断学号相等\*/

{

switch(x) /\*菜单选择科目\*/

{

case 1:p->a=score;break; /\*改正分数\*/

case 2:p->b=score;break;

case 3:p->c=score;break;

case 4:p->d=score;break;

}

}

p=p->next; /\*指针指向下一个结点\*/

}

p=\*headp; /\*指针指向头地址\*/

}

void average(struct stu \*\*headp){

struct stu \*p=\*headp;

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

{

int av=p->a+p->b+p->c+p->d; /\*计算总分\*/

printf("%s %s %.2f\n",p->num,p->name,(double)av/4);

}

}

void sum(struct stu \*\*headp){

struct stu \*p=\*headp;

while(p->next!=NULL){

printf("%s %s %d %.2f\n",p->num,p->name,p->a+p->b+p->c+p->d,(double)(p->a+p->b+p->c+p->d)/4);

p=p->next; /\*指针指向下一个结点\*/

}

}

(3)测试

表3-2 编程题2的测试数据

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 |  | 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 |
| 1 3 U987654321 Gray 0 0 0 0 U000000000 Black 100 100 100 100 U202073456 Red 45 34 67 99 |  |  |  |

3.

(1)算法步骤

1.声明结构体链表

2.声明各功能编号的函数

3.给链表分配内存

4.输入功能编号，switch

5.功能编号1: 1)输入人数n

2)指针指向头

3)输入数据到结构体中

4)指针指向下一个结点，分配内存

5)链表尾为NULL

6.功能编号2: 调用函数sort，output

1) 指针遍历链表

2) 输出结点中的数据

7.功能编号3: 调用函数correct

1) 输入要改正的学生学号，科目，成绩

2) 指针遍历链表,查找相同学号

3) 若找到相同的学号，改正相应科目成绩

8.功能编号4: 调用函数sort，average

1) 指针遍历链表

2）计算每位同学的平均分

3）遍历链表，输出学号，姓名，平均分

9.功能编号5: 调用函数sort，sum

1) 遍历链表

2) 计算总分和平均分

3) 输出各项数据

10.函数sort: 1)指针遍历链表，比较前后两项平均成绩的大小

2)冒泡排序，将节点中的每项数据交换

(2)程序清单

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct stu{

char num[11];

char name[10];

int a;

int b;

int c;

int d;

int av;

struct stu \*next;

};

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

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

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

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

void sort(struct stu \*head);

int main()

{

struct stu \*head=NULL;

int x;

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

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

while(1)

{

scanf("%d",&x);

switch(x)

{

case 1:

{

int n,i;

scanf("%d",&n);

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

{

scanf("%s%s%d%d%d%d",p->num,p->name,&p->a,&p->b,&p->c,&p->d);

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

tail->next=p;

tail=p;

}

p=head;

while(p->next!=NULL)

{

p->av=p->a+p->b+p->c+p->d;

p=p->next;

}

tail->next=NULL;

break;

}

case 2:sort(head);output(&head);break;

case 3:correct(&head);break;

case 4:sort(head);average(&head);break;

case 5:sort(head);sum(&head);goto down;

}

}

down:

return 0;

}

void output(struct stu \*\*headp){

struct stu \*p=\*headp;

while(p->next!=NULL)

{

printf("%s %s %d %d %d %d\n",p->num,p->name,p->a,p->b,p->c,p->d);

p=p->next;

}

}

void correct(struct stu \*\*headp){

struct stu \*p=\*headp;

char num0[11];

int x,score;

scanf("%s",num0);

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

while(p->next!=NULL){

if(strcmp(num0,p->num)==0)

{

switch(x)

{

case 1:p->a=score;break;

case 2:p->b=score;break;

case 3:p->c=score;break;

case 4:p->d=score;break;

}

}

p=p->next;

}

p=\*headp;

while(p->next!=NULL)

{

p->av=p->a+p->b+p->c+p->d;

p=p->next;

}

}

void average(struct stu \*\*headp){

struct stu \*p=\*headp;

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

{

printf("%s %s %.2f\n",p->num,p->name,(double)p->av/4);

}

}

void sum(struct stu \*\*headp){

struct stu \*p=\*headp;

while(p->next!=NULL){

printf("%s %s %d %.2f\n",p->num,p->name,p->a+p->b+p->c+p->d,(double)(p->a+p->b+p->c+p->d)/4);

p=p->next;

}

}

void sort(struct stu \*head){

struct stu \*p=head,\*p0;

char c[11];

char d[10];

int l,q,r,s,t,len=0,i,j;

while(p->next!=NULL){

len++; /\*计算链表长度\*/

p=p->next;

}

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

{

for(j=i+1,p0=p->next;j<len;j++,p0=p0->next)

{

if(p->av>p0->av) /\*遍历链表，比较\*/

{

strcpy(c,p->num);strcpy(p->num,p0->num);strcpy(p0->num,c);

strcpy(d,p->name);strcpy(p->name,p0->name);strcpy(p0->name,d);

l=p->a;p->a=p0->a;p0->a=l;

q=p->b;p->b=p0->b;p0->b=q;

r=p->c;p->c=p0->c;p0->c=r;

s=p->d;p->d=p0->d;p0->d=s;

t=p->av;p->av=p0->av;p0->av=t; /\*交换数据\*/

}

}

}

}

(3)测试

表3-3 编程题3的测试数据

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 | 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 |  | 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 |
| 1 3 U987654321 Gray 0 0 0 0 U000000000 Black 100 100 100 100 U202073456 Red 45 34 67 99 |  |  |  |

5.

(1)流程图

createlist函数：



judgePalindrome函数：



main函数：



(2)程序清单

main.c:

#include "stdio.h"

#include "string.h"

typedef struct c\_node{

char data;

struct c\_node \*next;

} C\_NODE;

#include "stu.h"

int main()

{

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

int len=0;

C\_NODE \*head,\*p;

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

createLinkList(&head,s);

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;

}

stu.h:

typedef struct c\_node{

char data;

struct c\_node \*next;

} C\_NODE;

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

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

char \*p=s;

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

head->data=\*p++;

tail=head;

while(\*p){

tail->next=(C\_NODE \*)malloc(sizeof(C\_NODE)); /\*分配内存\*/

tail=tail->next; /\*tail指向下一个结点\*/

tail->data=\*p++; /\*给节点赋值\*/

}

tail->next=NULL; /\*链表结束\*/

\*headp=head; /\*指针指向头\*/

}

void judgePalindrome(C\_NODE \*head){

struct c\_node \*x=head,\*y=head,\*z=head; /\*将头地址存到三个指针中\*/

int i,j,len=0,flag=1;

while(head!=NULL){

head=head->next;

len++; /\*计算链表长度\*/

}

int a=len; /\*长度存到a中\*/

for(j=1;j<=(a/2);j++)

{

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

{

x=x->next; /\*x指向未判断的最后一个指针\*/

}

if(y->data==x->data) /\*判断未判断的第一个指针和最后

{ 一个指针指向的字符是否相等\*/

y=y->next; /\*指向下一个未判断的指针\*/

len--; /\*判断完的链表移出\*/

}

else

{

flag=0; /\*不相等，退出循环\*/

goto down;

}

x=z; /\*x指向头\*/

}

down:

if(flag==1)

{

printf("true");

}

else

{

printf("false");

}

}

(3)测试

表3-4 编程题4的测试数据

|  |  |  |
| --- | --- | --- |
| 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 | acca | true |

7.

(1)算法清单

1.计算总分

2.分配内存，形成单链表

3.指针遍历列表，判断前后两项分数高低

4.交换节点

5.输出结果

(2)程序清单

void sort(struct stu \*\*headp){

struct stu \*p1,\*p2,\*pr1,\*pr2;

struct stu \*temp;

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

{

p2->av=p2->c1+p2->c2+p2->c3+p2->c4; /\*计算总分\*/

}

p1=(struct stu\*)malloc(sizeof(struct stu));

p1->next=\*headp;

(\*headp)=pr1=p1;

for(p1=pr1->next;p1->next;pr1=p1,p1=p1->next)

{

for(p2=p1->next,pr2=p1;p2;pr2=p2,p2=p2->next)

{

if(p1->av>p2->av) /\*判断分数高低\*/

{

temp=p2->next; /\*交换节点\*/

pr1->next=p2;

pr2->next=p1;

p2->next=p1->next;

p1->next=temp;

temp=p1;p1=p2;p2=temp;

}

p1=(\*headp);

(\*headp)=p1->next;

free(p1); /\*释放内存\*/

}

}

}

(3)测试

表2-5 编程题7的测试数据

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 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 | 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 |  | 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 |
| 1 3 U987654321 Gray 0 0 0 0 U000000000 Black 100 100 100 100 U202073456 Red 45 34 67 99 |  |  |  |

9.

(1)算法步骤

1.定义结构体，函数

2.分配内存，形成链表

3.从后向前进行运算

4.输出答案

(2)程序清单

#include<stdio.h>

#include<stdlib.h>

#define N 100

struct nbl{

int num;

int op[N];

};

void push(struct nbl \*st, int n){

st->num++;

st->op[st->num]=n; /\*推入\*/

}

void pull(struct nbl \*st, int \*n){

\*n=st->op[st->num];

st->num--; /\*推出\*/

}

int main()

{

char s[100],c;

int flag=0,i=0,n,m,b;

gets(s);

int result=0;

struct nbl \*number;

number=(struct nbl\*)malloc(sizeof(struct nbl));

number->num=-1;

i=0;

while(s[i]){

flag=0;

if(s[i]=='-') /\*从后向前进行运算\*/

{

if(s[i+1]>='0'&&s[i+1]<='9') /\*判断是数字还是符号\*/

{

flag=1;

i++;

}

}

if(s[i]>='0'&&s[i]<='9')

{

b=0;

b+=s[i]-'0';

i++;

while(s[i]>='0'&&s[i]<='9'){

b\*=10;

b+=s[i]-'0';

i++;

}

if(flag==1)

b\*=(-1);

push(number,b);

}

else if(s[i]=='+'||s[i]=='-'||s[i]=='\*'||s[i]=='/')

{

pull(number,&n);

pull(number,&m);

if(s[i]=='+')

result=m+n;

else if(s[i]=='-')

result=m-n;

else if(s[i]=='\*')

result=m\*n;

else if(s[i]=='/')

result=m/n;

push(number,result);

i++;

}

else

i++; /\*下一个数字或符号\*/

}

printf("%d",result);

}

(3)测试

表2-6 编程题9的测试数据

|  |  |  |
| --- | --- | --- |
| 测试用例 | 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 | 1 2 + | 3 |

## 3.4小结

链表的题目结点容易搞混，需要画图来确定步骤的先后。需要循环输入形成链表时最好不调用函数，直接在主函数中输入更方便。学习了逆波兰式中栈的应用。**参考文献**

[1] 曹计昌,卢萍,李开. C语言程序设计,北京：科学出版社,2013

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