

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 计算机1905班**

**学 号： U201915035**

**姓 名： 邹 雅**

**指导教师： 李 平**

**报告日期： 2019年12月03日**

**计算机科学与技术学院**

**目录**

**[实验三 流程控制实验 1](#_Toc2706)**

[3.1程序改错 1](#_Toc4709)

[3.2程序修改替换 1](#_Toc1272)

[3.3程序设计 3](#_Toc27529)

[3.4小结 14](#_Toc10150)

**[实验五 数组程序设计实验 15](#_Toc25554)**

[5.1程序改错与跟踪调试 15](#_Toc24055)

[5.2程序修改替换 15](#_Toc8956)

[5.3程序设计 19](#_Toc15166)

[5.4小结 31](#_Toc30578)

**[实验七 结构与联合实验 32](#_Toc17176)**

[7.1 表达式求值的实验验证 32](#_Toc29137)

[7.2 源程序修改替换 32](#_Toc14678)

[7.3 程序设计 34](#_Toc3494)

[7.4 小结 55](#_Toc24141)

**[参考文献 56](#_Toc22788)**

**实验三 流程控制实验**

**3.1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，Ctrl+z结束\n");

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

{

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

if(!x%i) 错误原因：!是单目运算符，优先级别高于%，应改if(!(x%i))

{

flag=1;

break;

}

if(flag=1) printf("%d是合数\n",x); 错误原因：判断的等号应该是==。

else printf("%d 不是合数\n",x);

}

return 0;

}

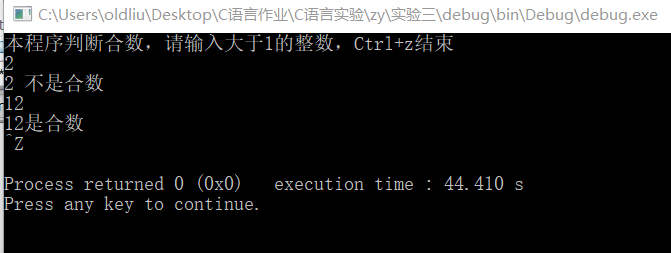


图1 改错题的修改后运行结果

**3.2程序修改替换**

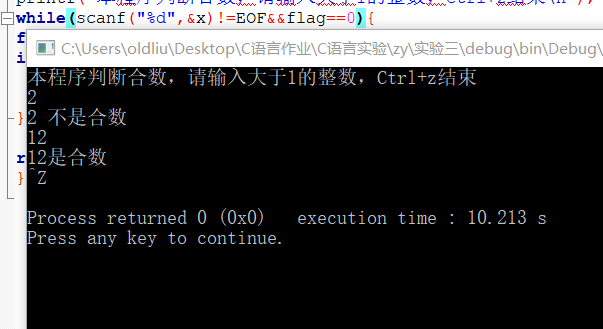
1. 原程序通过break语句在已经得到一个除1和其本身的除数之后退出循环。将程序改成单出口结构可以在删去break语句后将原while循环改成while(scanf("%d",&x)!=EOF&&flag==0) 。

图2（1） 修改题1的运行结果

（2）原程序通过for循环来找x的可能约数，也可以将这部分for循环改成 if(x==2) printf("%d 不是合数\n",x);

i=2; k=x/2+1;

do{

if(!(x%i))

flag=1;

i++; }

while(i<=k);

图2（2） 修改题2的运行结果

（3）此次修改要求改成纯粹合数求解器。可以使用原程序的判断合数方法。限定输入一个三位数，先对这个三位数进行合数判断，其次通过除法将其中的最后一位数去掉，在进行合数判断，最后再对最高位的单位数判断即可。以下给出代码。

#include <stdio.h>

/\*纯粹合数求解器\*/

int is(int x,int flag);/\*判断合数的函数\*/

int main()

{

int x,flag=0;/\*x是判断变量，用来输入百位数字，求解其中的纯粹合数\*/

printf("本程序判断纯粹合数\n");

for(x=100;x<1000;++x){

if(is(x,flag)==1) {flag=0;

if(is(x/10,flag)==1) {flag=0;

if(is(x/100,flag)==1) printf("%d ",x);}

}

}

return 0;

}

int is(int x,int flag){

int i,k;

i=2; k=x/2+1;

while(i<k&&flag==0){

if(!(x%i))

flag=1;

i++; }

return flag;

}

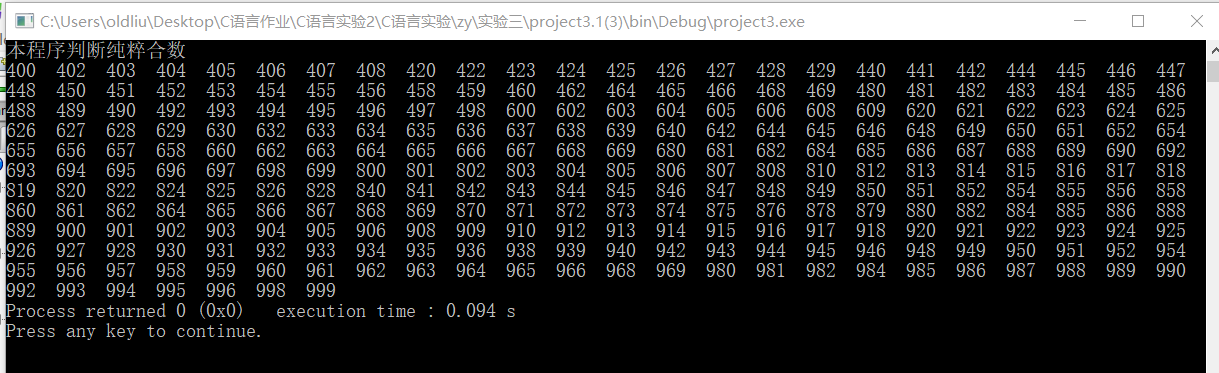


图2（3） 修改题3的运行结果

**3.3程序设计**

（1）先用分支语句判断x的所属范围，分为x<1000，1000<=x<2000，2000<=x<3000，3000<=x<4000，4000<=x<5000，x>5000六个范围。在每一个范围里都有相应的税金计算公式，即超过某个区间的工资乘以相应范围的税金比例再加上前一部分区间的所需税金。以下先给出if语句实现的程序，再给出switch语句实现的程序。

其中可输入的值可以是任意正数，故以下任意采用两个数据。

表1-1 设计题1的测试数据

|  |  |  |
| --- | --- | --- |
| 测试数据x | 2582 | 6850 |
| 应输出值 | 108.199997 | 962.500000 |

表1-1中测试用例的运行结果如图1-1、图1-2、图1-3和图1-4所示。

#include <stdio.h>

int main()

{

float x,taxx;

scanf("%f",&x);

if(x<1000) taxx=0;

else if(x>=1000&&x<2000) taxx=0.05\*(x-1000);

else if(x>=2000&&x<3000) taxx=0.10\*(x-2000)+50;

else if(x>=3000&&x<4000) taxx=0.15\*(x-3000)+150;

else if(x>=4000&&x<5000) taxx=0.20\*(x-4000)+300;

else if(x>5000) taxx=0.25\*(x-5000)+500;

printf("%f",taxx);

return 0;

}

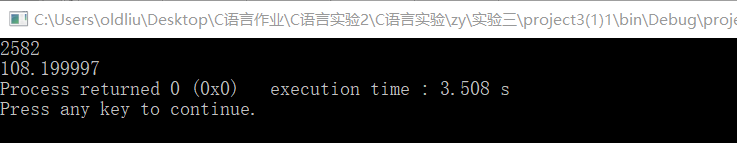
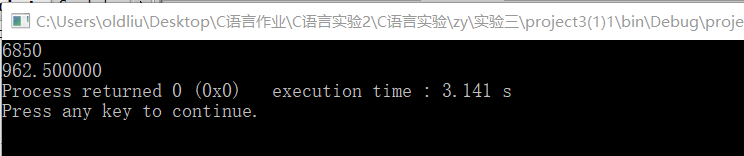


图1-1 设计题1的测试用例一的运行结果

图1-2 设计题1的测试用例二的运行结果

#include <stdio.h>

int main()

{

int x;

float taxx;

scanf("%d",&x);

switch(x/1000){

case 0:taxx=0;break;

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

case 2:taxx=0.10\*(x-2000)+50;break;

case 3:taxx=0.15\*(x-3000)+150;break;

case 4:taxx=0.20\*(x-4000)+300;break;

case 5:case 6:case 7:case 8:case 9:case 10:case 11:case 12:

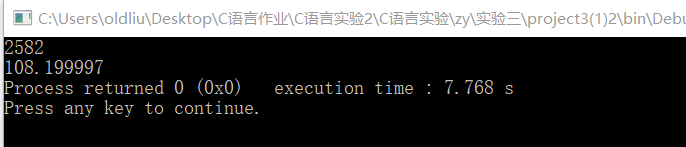
taxx=0.25\*(x-5000)+500;break;

}

printf("%f",taxx);

return 0;

}

图1-3 设计题1的测试用例一的运行结果

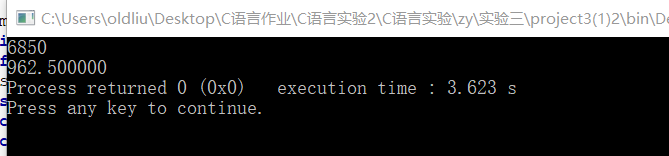


图1-4 设计题1的测试用例二的运行结果

（2）先输入变量n，再定义一组数组来输入每一天的收盘价格要找出股票的最大波动值，只需前后相减算出每一次的波动值（若值为负则乘以-1），然后将第一个值赋给一个判断变量maxp，之后每一次差值都和它比较，若任何差值大于这个判断变量，则将这个值赋给判断变量。直至最后可以找出最大的波动值。

以下给出代码。

#include <stdio.h>

int main()

{

int n,i,d,maxp,tem;

/\*n为天数，i、d都是循环判断变量，tem表示暂存所得波动值，maxp表示当前的最大波动值。\*/

scanf("%d",&n);

int pri[n];/\*记录每天的收盘价格\*/

for(i=0;i<n;++i) scanf("%d",&pri[i]);

for(d=0,tem=0,maxp=0;d<n-1;++d){

tem=pri[d+1]-pri[d];

if(tem<0) tem\*=-1;

if(tem>maxp) maxp=tem;

}

printf("%d",maxp);

return 0;

}

表2-1 设计题2的测试数据

|  |  |  |
| --- | --- | --- |
| 输入天数n | 输入每天的收盘价格pri[] | 输出最大波动值maxp |
| 7 | 1 2 2 3 8 5 2 | 5 |
| 9 | 10 12 20 35 64 23 85 96 32 |  |

表2-1中测试用例的运行结果如图2-1、图2-2所示。

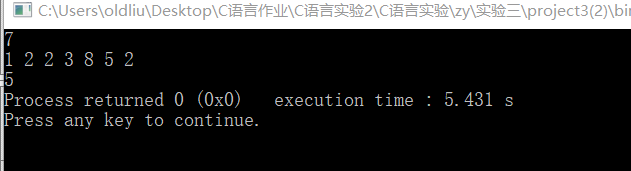


图2-1 设计题2的测试用例一的运行结果

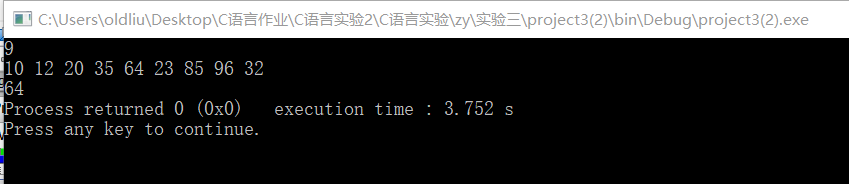


图2-2 设计题2的测试用例二的运行结果

（3）该程序的设计目的是将每行中超过一个以上的空格字符改为一个空格字符。因此定义两种状态：一种是copy，将输入的正文输出；一种是start，表示多个空格字符的开始，不再一一复制。用getchar（）函数输入字符，当遇到空格字符时，状态从copy更改为start，然后直至输入非空格字符，状态再改回copy。

以下给出代码。

#include <stdio.h>

enum {copy,start};

int main()

{

char c;

int state=copy;

c=getchar();

judge:

for(;c!='\n';c=getchar()){

switch(state){

case copy:

if(c==' ') {state=start;putchar(' ');}

else putchar(c);break;

case start:

if(c!=' ') {state=copy;putchar(c);}

else ;

}

}

if(c=='\n'){

c=getchar();

for(;c=='\n';)c=getchar();

if(c!='\n') goto judge;

} /\*这个if语句用于实现分行的判断\*/

return 0;

}

表3-1 设计题3的测试数据

|  |  |  |
| --- | --- | --- |
| 输入正文 | abc ddefg | ab cd ef |
| 输出正文 | abc ddefg | ab cd ef |

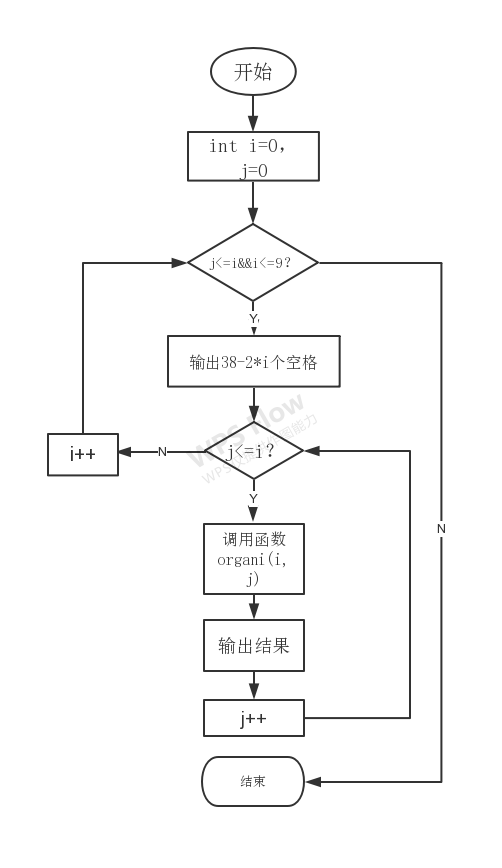
表3-1中测试用例的运行结果如图3-1、图3-2所示。



图3-1 设计题3的测试用例一的运行结果

图3-2 设计题3的测试用例二的运行结果

（4）设计一个函数int organi(int ,int )利用阶乘思想来计算相应位置的组合数，在每次运行到一个位置时都调用这个函数。

以下先给出流程图。

流程图3-1

以下给出代码。运行结果如图4-1所示。

#include <stdio.h>

int organi(int ,int );

int main()

{

int i,j,k;

for(i=0,j=0;j<=i&&i<=9;++i){

for(k=1;k<=38-2\*i;k++) putchar(' ');

for(;j<=i;j++) printf("%4d",organi(i,j));

printf("\n");

j=0;

}

return 0;

}

int organi(int i,int j){

int a,b,c;

int a1,b1,c1;

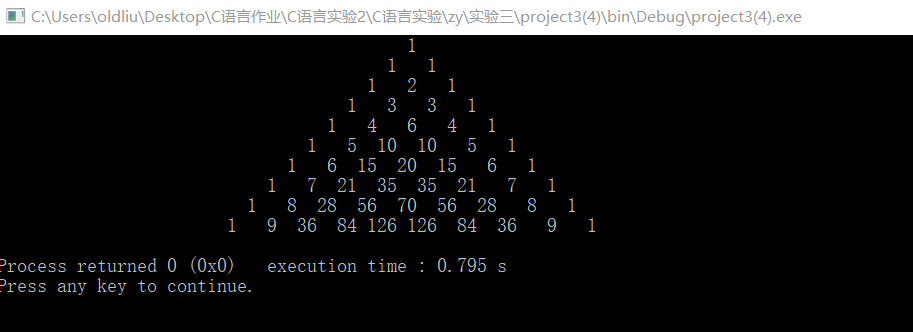
for(a1=1,a=1;a1<=j;++a1) a\*=a1;

for(b1=1,b=1;b1<=i;++b1) b\*=b1;

for(c1=1,c=1;c1<=i-j;++c1) c\*=c1;

return b/(a\*c);

}

图4-1 设计题4的运行结果

1. 利用两层for循环来分别确定该四位数的前两位数的数字和后两位数的数字，在用sqrt（）函数结合一个if语句判断该四位数是否是一个整数的平方，最后输出满足条件的数即可。

以下给出代码。

运行结果如图5-1所示。

#include <stdio.h>

#include <math.h>

int main()

{

int number;

int front,after;

int x;

for(front=1;front<=9;front++){

for(after=1;after<=9;after++){

if(front==after) continue;

number=front\*1100+after\*11;

x=sqrt(number);

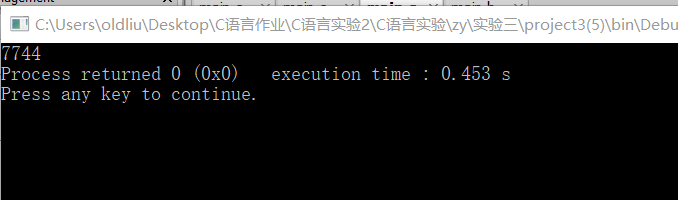
if(x\*x==number) printf("%d",number);

}

}

return 0;

}

图5-1 设计题5的运行结果

1. 该程序要计算四个回合后体力值小于1000的理论概率。首先根据输赢情况我们可以知道一共有十六种等概率情况，每种情况的概率为0.0625。假定将甲赢的情况定义为1，输的情况定义为0，那么用0到15这十六个数字的前四位二进制数字的0、1情况即可得到十六种不同的比赛情况。比如数字3的二进制表示为0011，表示前两场甲赢，后两场乙赢。通过内外层循环，即可得到每一次比赛后双方的体力值，将其赋给两个数组来保存。最后进行比较得到满足条件的情况数，乘以概率，再输出概率即可。

以下给出代码。运行结果如图6-1所示。

#include <stdio.h>

enum {lose,win};

int main()

{

float j=1000,y=2000;

float m[16],n[16];/\*保存每种情况的体力值的数组\*/

int a,b,c,d;/\*循环变量\*/

float number=0;/\*最后有多少种情况满足体力值只差小于1000\*/

unsigned int t;

int x[4];

for(a=0,t=0;a<=15&&t<=15;a++,t++){

for(b=0;b<=3;b++){

x[b]=(t>>b)&1;

}/\*保存每一种情况四个回合中甲的输赢情况\*/

for(c=0;c<=3;c++){

if(x[c]==0) {j+=y/4;y=3\*y/4;}

if(x[c]==1) {y+=j/4;j=3\*j/4;}

}

m[a]=j;n[a]=y;

j=1000;y=2000;/\*每种情况计算之后重新初始化甲乙的体力值\*/

}

for(d=0;d<=15;d++){

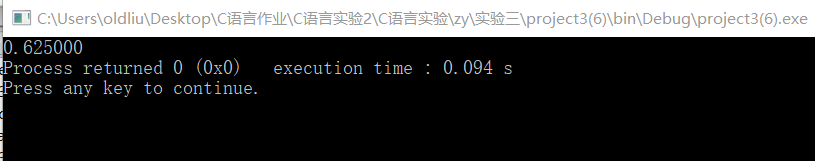
if(m[d]-n[d]>-1000&&m[d]-n[d]<1000) number++;

}

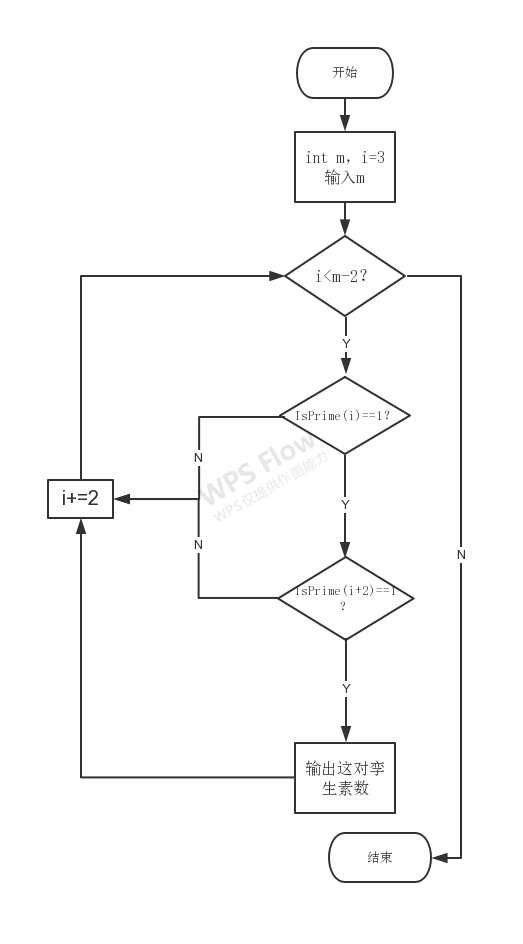
printf("%f",number/16);

return 0;

}

图6-1 设计题6的运行结果

1. 构造一个函数int IsPrime(int )来判断一个数是否是素数。

以下先给出流程图。

流程图3-7

以下给出代码。

#include <stdio.h>

int IsPrime(int );

int main()

{

int m,i;

scanf("%d",&m);

for(i=3;i<m-2;i+=2){

if(IsPrime(i)==1&&IsPrime(i+2)==1) printf("(%d,%d) ",i,i+2);

}

return 0;

}

int IsPrime(int n){

int k,limit;

if(n==2) return 1;

if(!(n%2)) return 0;

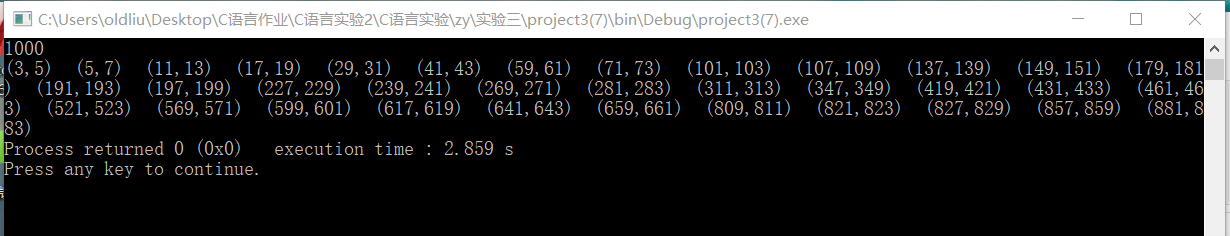
limit=n/2;

for(k=3;k<=limit;k+=2)

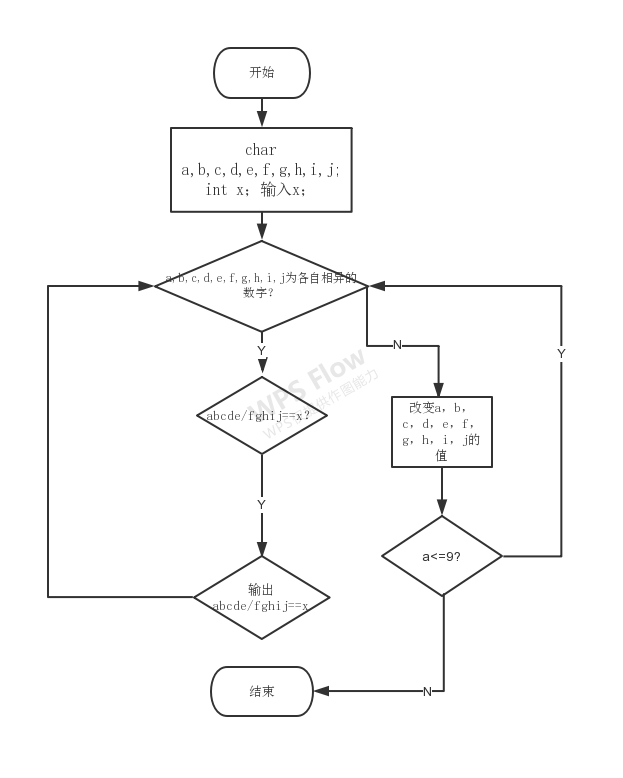
if(!(n%k)) return 0;

return 1;

}

图7-1给出以m=1000为例的所有孪生素数。图7-1 设计题7的运行结果

1. 定义10个字符型数字a，b，c，d，e，f，g，h，i，j，通过十个for循环以及结合if和continue语句来分别输入不同的0~9十个数字。输入x，判断abcde/fghij的值是否等于x，若等于则输出abcde/fghij=x。

以下给出流程图。

流程图3-8

以下给出代码。

#include <stdio.h>

int main()

{

char a,b,c,d,e,f,g,h,i,j;

int x;

int m,n;

scanf("%d",&x);

for(a='0';a<='9';a++)

{

for(b='0';b<='9';b++)

{

if((b-a)==0) continue;

for(c='0';c<='9';c++)

{

if((b-c)\*(c-a)==0) continue;

for(d='0';d<='9';d++)

{

if((d-c)\*(d-a)\*(d-b)==0) continue;

for(e='0';e<='9';e++)

{

if((e-c)\*(e-a)\*(e-b)\*(e-d)==0) continue;

for(f='0';f<='9';f++)

{

if((f-c)\*(f-a)\*(f-b)\*(f-d)\*(f-e)==0) continue;

for(g='0';g<='9';g++)

{

if((g-c)\*(g-a)\*(g-b)\*(g-d)\*(g-e)\*(g-f)==0) continue;

for(h='0';h<='9';h++)

{

if((h-c)\*(h-a)\*(h-b)\*(h-d)\*(h-e)\*(h-f)\*(h-g)==0) continue;

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

{

if((i-c)\*(i-a)\*(i-b)\*(i-d)\*(i-e)\*(i-f)\*(i-g)\*(i-h)==0) continue;

for(j='0';j<='9';j++)

{

if((j-c)\*(j-a)\*(j-b)\*(j-d)\*(j-e)\*(j-f)\*(j-g)\*(j-h)\*(j-i)==0) continue;

m=(a-'0')\*10000+(b-'0')\*1000+(c-'0')\*100+(d-'0')\*10+(e-'0');

n=(f-'0')\*10000+(g-'0')\*1000+(h-'0')\*100+(i-'0')\*10+(j-'0');

if(m/n==x&&m%n==0) printf("%c%c%c%c%c/%c%c%c%c%c=%d\n",a,b,c,d,e,f,g,h,i,j,x);

}

}

}

}

}

}

}

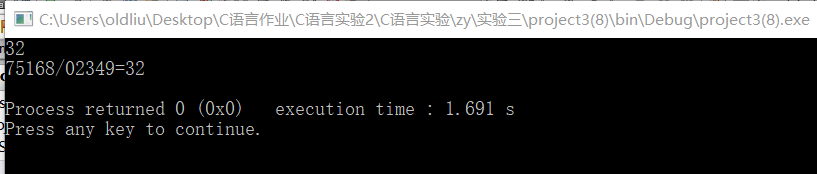
}

}

}

return 0;

}

图8-1给出x=32时应输出75168/02349=32的运行结果。图8-1 设计题8的运行结果

**3.4小结**

通过本次流程控制实验，我更加深入地体会到了流程控制的顺序、循环、分支等多种思想，同时在设计实验程序和反复地调试中，我对C语言的语法和其注意事项等更加熟悉，在多次的错误中我发现了自己的知识盲区和思维漏洞。

比如我之前将“\0”当做空格符使用，而在实际的程序编写中，直接用“ ”代替即可。比如我学会了如何判断一个数是否是一个整数的平方，学会使用sqrt（）函数和#include<math.h>。比如我更加深刻地体会到switch语句中case只能等于一个值而不是一个范围这个点，让我以后使用switch语句时加强注意。对于第六个程序编写题，我第一次编写时忘记了回合输赢情况的顺序问题，导致程序需要大的修改，而在第二次编写时采用了0~15的前四位二进制表示输赢情况，这个创新也是在我的反复思考中得来的；而在第二次的书写中，我采用了将变量乘以3/4的方式来更新体力值，结果导致变量值变成0，而在采用了输出函数来打印之后才发现这个错误，我感悟到在程序编写中一定要注意到每一个细节点，不然一点点的偏差都会导致失败，而且debug的方法也很重要。再比如我对于初始化变量也犯了一些错误，一次是忘记初始化变量导致输出很大的任意值，还有几次是反复地初始化变量导致了变量的值改变不恰当。这些错误都是经验的累积，经历过了以后就会注意或者更好更快地发现错误。再比如对数组的提前预习后在这次实验中多次对数组的使用，感受到了数组对于循环语句的便捷性。

特别地，我在这次的实验中尝试了多次break、continue、goto语句和自定义函数。实践出真知，这些尝试让我更好地掌握了它们并能更好地使用。

总之，在刚开始C语言的这个阶段，很多语法和思想都没有学，所以我还有很多的知识要学习，还有很大的进步空间去发展。之后我将多看书，多复习，多实践，争取之后实验避免耗费大量的debug时间。

实验五 数组程序设计实验

5.1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[50]="Language",b[50]="Programming"; 错误：原题中未规定a和b的数组长度，则当b连接到a后时，a中存放不下多余的数据就会挤占a后b数组的存储空间。故应给两个数组足够大的空间。

    printf("%s   %s\n",b,a);

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

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

    return 0;

}

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

{

    int i=0,j=0;

    while(t[i++]) ;i--;

    while((t[i++]=s[j++])!='\0'); 错误：原题处无i--，则当第一个while循环结束进入第二个while循环时，i指向的是t数组原空白字符后一个字符，那么当输出t时仍在原处结束。

}

void strdelc(char s[],char c)

{

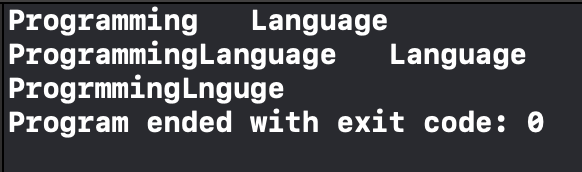
    int j,k;

    for(j=k=0;s[j]!='\0';j++)

        if(s[j]!=c)  s[k++]=s[j];

    s[k]='\0'; 错误：加上'\0'表示字符串的结束符

}



**图1 改错题的修改后运行结果**

5.2程序修改替换

1. (1)

#include <stdio.h>

#include<string.h>

void RemoveDuplicate(char \*s); 原因：必须加入函数的声明后才能调用。

int main()

{

    char str[200];

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

    while(fgets(str,200,stdin)!=NULL)

    {

        RemoveDuplicate(str);

        printf("%s",str);

    }

    return 0;

}

void RemoveDuplicate(char \*s)

{

    int r,w,i,len;

    len=strlen(s);

    for(r=w=0;r<len;r++)

    {

        if(s[r]!='\0') 原因：当字符不为'\0'时，进入执行语句。

        {

          s[w++]=s[r]; 原因：目的是去掉'\0'的数组元素。

          for(i=r+1;i<len;i++)

            if(s[i]==s[r])

            s[i]='\0';

        }

    }

    s[w]='\0'; 原因：要形成字符串，故加上'\0'

}

以下以数据1212eeer为例。

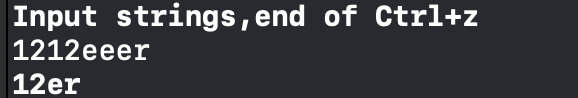


图2 修改题1.(1)的修改后运行结果

1. (2)

此次的程序修改设置一个有256个元素ascii数组，表示256个ASCII码代表的字符，当遇到一个字符，根据该字符当ASCII码值对应的下标的元素设为1，则下一次遇到字符时先判断该字符ASCII码值对应下标的元素是否为1，若为1，则跳过。（反思：这种设计方法导致一次程序启动没法多次进行删除重复的操作，因为可能上一次已经改变了ascii数组内的值。）

#include <stdio.h>

#include<string.h>

void RemoveDuplicate(char \*s);

int ascii[256];

int main()

{

    char str[200];

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

    while(fgets(str,200,stdin)!=NULL)

    {

        RemoveDuplicate(str);

        printf("%s",str);

    }

    return 0;

}

void RemoveDuplicate(char \*s)

{

    int r,w,len;

    len=strlen(s);

    for(r=w=0;r<len;r++)

    {

        if(ascii[(int)s[r]]!=1)

        {

          s[w++]=s[r];

            ascii[(int)s[r]]=1;

        } }

    s[w]='\0';

}

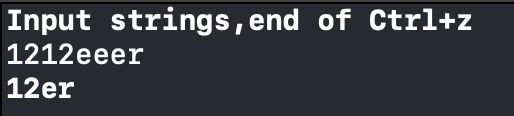


图3 修改题1.(2)的修改后运行结果

1. (1)

#include <stdio.h>

#define M 10

#define N 3

int main(void)

{

    int a[M],b[M];

    int i,j,k;

    for(i=0;i<M;i++)

        a[i]=i+1;

    for(i=M,j=0;i>1;i--){

        for(k=1;k<=N;k++)

            if(++j>i-1) j=0;

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

        if(j)

            for(k=--j;k<i;k++)

                a[k]=a[k+1];

    }

    for(i=0;i<M-1;i++)

        printf("%6d",b[i]);

    printf("%6d\n",a[0]);

    return 0;

}图4 修改题2.(1)的修改后运行结果

1. (2)

改写之后，让表示已经出圈的人的数组元素等于0。然后在之后的报数循环中，跳过所有已经变为0的数组元素即可。

#include <stdio.h>

#define M 10

#define N 3

int main(void)

{

    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++)

        {

            while (!a[j]) {

                j++;

            }

            if(++j>M-1) j=0;

        }

        if(j){

            b[M-i]=a[j-1];

            a[j-1]=0;

        }

        else {

            b[M-i]=a[M-1];

            a[M-1]=0;

        }

    }

    for(i=0;i<M-1;i++)

        printf("%6d",b[i]);

    for(i=0;i<M;i++)

        if (a[i]!=0) {

            printf("%6d",a[i]);

        }

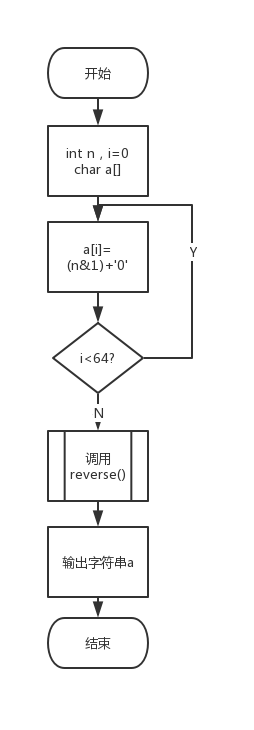
    return 0;

}

图5 修改题2.(2)的修改后运行结果

**5.3程序设计**

1.

#include <stdio.h>

#include <string.h>

void reverse(char s[])

{

    int c,j;

    unsigned long k;

    for(j=0,k = strlen(s)-1;j<k;j++,k--)

    { c=s[j];s[j]=s[k];s[k]=c;}

}

int main(void)

{

    int n;

    char a[100];

    int i,t;

    printf("Input a interger");

    scanf("%d",&n);

    for(i=0;i<64;i++){

        a[i]=(n&1)+'0';

        n=n>>1;

    }

    reverse(a);

    for(t=0;t<strlen(a);t++) printf("%c",a[t]);

    return 1;

}

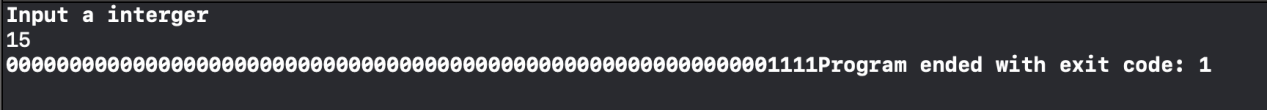
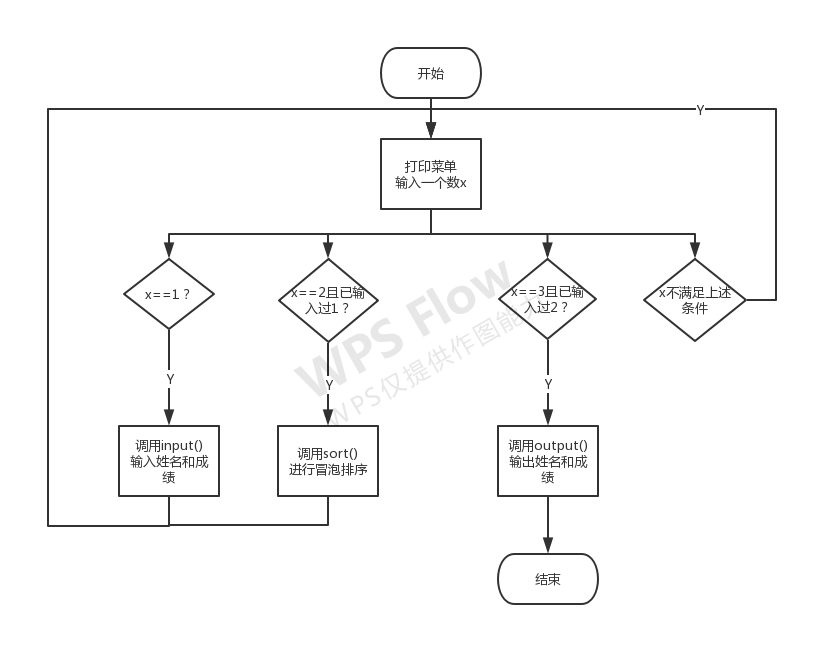


图1-1 设计题1的运行结果 流程图5-1

2.



流程图5-2

#include <stdio.h>

int flag1=0,flag2=0;

char name[100][10];

int score[100];

int number;

void input(void);

void sort(char name[100][10],int score[]);

void output(void);

int main(void)

{

    int x;

start:

        printf("---menu---\n");

        printf("1:Input the information\n");

        printf("2:Sort\n");

        printf("3:Output\n");

        printf("----------\n");

        printf("Input a number: ");

        scanf("%d",&x);

        printf("\n");

    if(x==1&&flag1==0) {input();flag1=1;goto start;}

    else if(x==2&&flag1==1) {

        sort(name, score);flag2=1;

        printf("sort is finished\n");

        goto start;

    }

    else if(x==3&&flag1==1&&flag2==1) output();

    else {printf("error\n");goto start;}

    return 1;

}

 void input()

{

    int i,k;

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

    scanf("%d",&number);

    for(i=0;i<number;i++){

         scanf("%s",name[i]);

        }

    for(k=0;k<number;k++){

        scanf("%d",&score[k]);

    }

}

void sort(char name[100][10],int score[100])

{

    int i,j,t;

    char tem[100][10];

    for(i=0;i<number-1;i++){

        for(j=0;j<number-i-1;j++){

            if(score[j]>score[j+1])

            {t=score[j];score[j]=score[j+1];score[j+1]=t;

                \*tem[j]=\*name[j];\*name[j]=\*name[j+1];\*name[j+1]=\*tem[j];

            }

        }

    }

}

void output(void)

{

    int i;

    for(i=number-1;i>=0;i--)

    {

        printf("%s  ",name[i]);

        printf("%d\n",score[i]);

    }

}

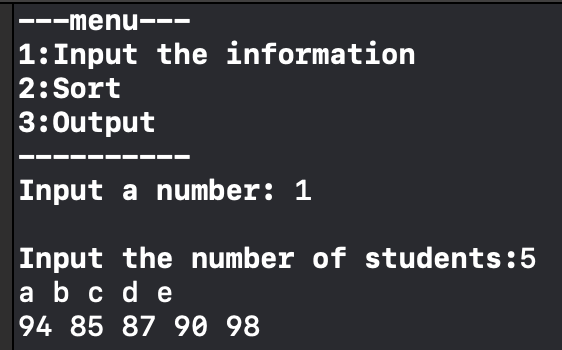


图2-1 设计题2的运行结果

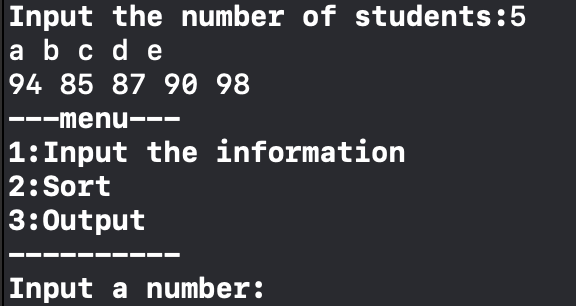


图2-2 设计题2的运行结果

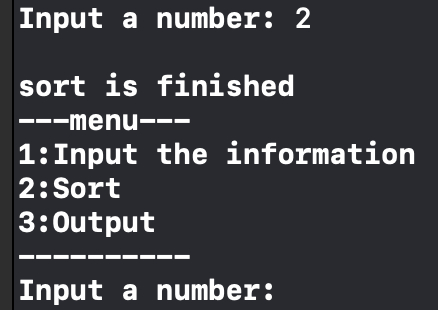


图2-3 设计题2的运行结果

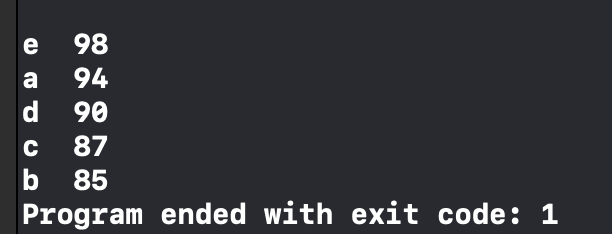
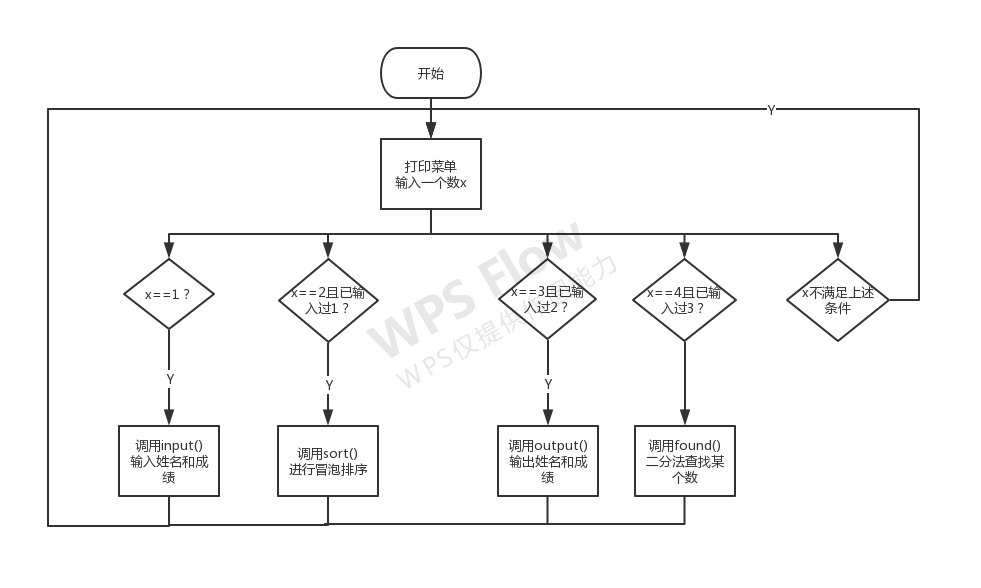


图2-4 设计题2的运行结果

3.



流程图5-3

#include <stdio.h>

int flag1=0,flag2=0;

char name[100][10];

int score[100];

int number;

void input(void);

void sort(char name[100][10],int score[]);

void output(void);

void found(void);

int main(void)

{

    int x;

start:

        printf("---menu---\n");

        printf("1:Input the information\n");

        printf("2:Sort\n");

        printf("3:Output\n");

        printf("4:found\n");

        printf("----------\n");

        printf("Input a number: ");

        scanf("%d",&x);

        printf("\n");

    if(x==1&&flag1==0) {input();flag1=1;goto start;}

    else if(x==2&&flag1==1) {

        sort(name, score);flag2=1;

        printf("sort is finished\n");

        goto start;

    }

    else if(x==3&&flag1==1&&flag2==1) {output();goto start;}

    else if(x==4&&flag1==1&&flag2==1) {found();goto start;}

    else {printf("error\n");goto start;}

    return 1;

}

 void input()

{

    int i,k;

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

    scanf("%d",&number);

    for(i=0;i<number;i++){

         scanf("%s",name[i]);

        }

    for(k=0;k<number;k++){

        scanf("%d",&score[k]);

    }

}

void sort(char name[100][10],int score[100])

{

    int i,j,t;

    char tem[100][10];

    for(i=0;i<number-1;i++){

        for(j=0;j<number-i-1;j++){

            if(score[j]>score[j+1])

            {t=score[j];score[j]=score[j+1];score[j+1]=t;

                \*tem[j]=\*name[j];\*name[j]=\*name[j+1];\*name[j+1]=\*tem[j];

            }

        }

    }

}

void output(void)

{

    int i;

    for(i=number-1;i>=0;i--)

    {

        printf("%s  ",name[i]);

        printf("%d\n",score[i]);

    }

}

void found(void)

{

    int s,result=-1,flag=0;

    int front=0,back=number-1,middle;

printf("Input a score you want to find:");

    scanf("%d",&s);

    while(front<=back&&flag==0){

        middle=(front+back)/2;

        if(s<score[middle]) back=middle-1;

        else if(s>score[middle]) front=middle+1;

        else {result=middle;flag=1;}

    }

    if(result==-1) printf("not found!");

    else {printf("%s   ",name[result]);

    printf("%d\n",score[result]);

    }

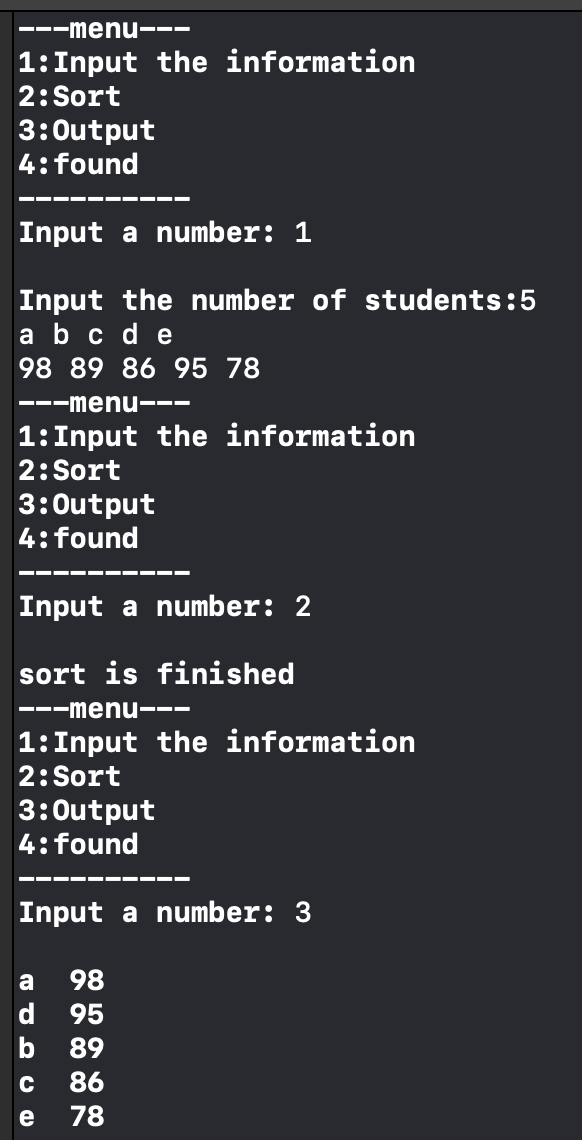
}

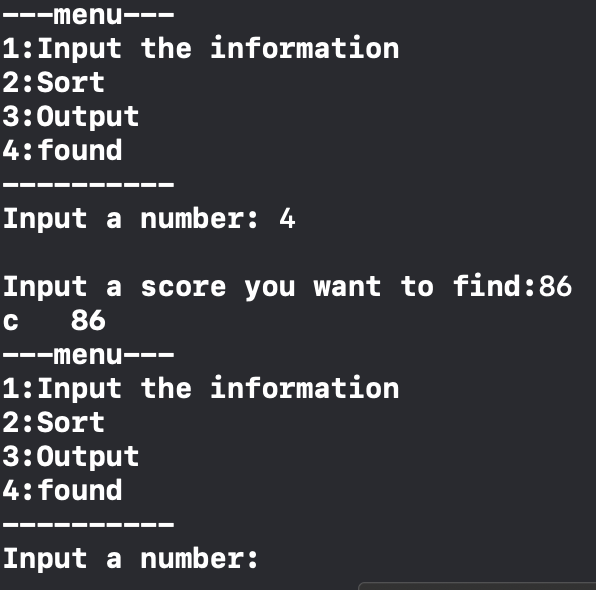
图3-1 设计题3的运行结果

图3-2 设计题3的运行结果

1. 先输入n、t、s，然后调用函数strnins(t,s,n)。该函数将t数组第n个字符后字符先赋给一个中间数组，然后将s中的字符赋值入t数组第n个字符后，再将中间数组中保存的t原来的字符接在改变后的t后，最后输出t。

#include <stdio.h>

#include <string.h>

void strnins(char [],char [],int );

int main(void){

    char t[100],s[100];

    int n;

    scanf("%d",&n);

    scanf("%s",t);

    scanf("%s",s);

    strnins(t,s,n);

}

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

    unsigned long i,u;

    int k,j,a,b,c,d;

    i=strlen(s);

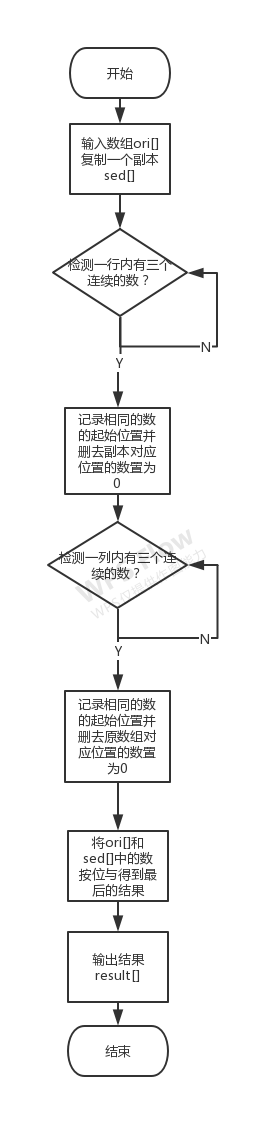
    u=strlen(t);

    char tem[u-n];

    for(k=n,j=0;k<u;k++,j++){

        tem[j]=t[k];

    }

    for(a=n,b=0;b<i;a++,b++){

        t[a]=s[b];

    }

    for (c=0,d=n+i; c<j; c++,d++) {

        t[d]=tem[c];

    }

    printf("%s\n",t);

}

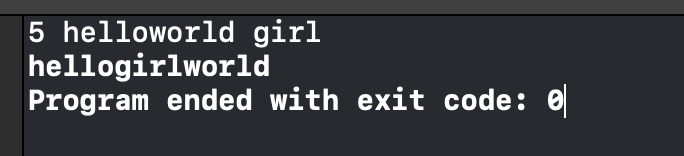


图5-1 设计题5的运行结果

6.流程图如右图所示。

#include <stdio.h>

int main(void)

{

    int n,m,i,j;

    printf("Input n and m:");

    scanf("%d%d",&n,&m);

    int ori[n][m],sed[n][m],result[n][m];

    int first=0,end=0,a,b,flag=0;

    for(i=0;i<n;i++){

        for(j=0;j<m;j++){

            scanf("%d",&ori[i][j]);

        sed[i][j]=ori[i][j];

    }

    }

    for(i=0;i<n;i++){

        for(j=0;j<m-2;j++,flag=0)

        {

        if(sed[i][j]==sed[i][j+1]&&sed[i][j+1]==sed[i][j+2])

        {

            first=j;end=j+2;

            for(a=3;a+j<m;a++){

                if(sed[i][j+a]==sed[i][j]) end=j+a;

                else break;

            }

            flag=1;

        }

            if(flag==1){

                for(b=first;b<=end;b++)

                    sed[i][b]=0;

            }

    }

    }

    for(j=0;j<m;j++){

        for(i=0;i<n-2;i++,flag=0)

        {

        if(ori[i][j]==ori[i+1][j]&&ori[i+1][j]==ori[i+2][j])

        {

            first=i;end=i+2;

            for(a=3;a+j<m;a++){

                if(ori[i+a][j]==ori[i][j]) end=j+a;

                else break;

            }

            flag=1;

        }

        if(flag==1){

            for(b=first;b<=end;b++)

                ori[b][j]=0;

        }

    }

    }

    for(i=0;i<n;i++){

        for(j=0;j<m;j++){

        result[i][j]=ori[i][j]&sed[i][j];

    }

    }

    for(i=0;i<n;i++){

        for(j=0;j<m;j++){

        printf("%d  ",result[i][j]);

    }

        printf("\n");

    }

    return 1;

}

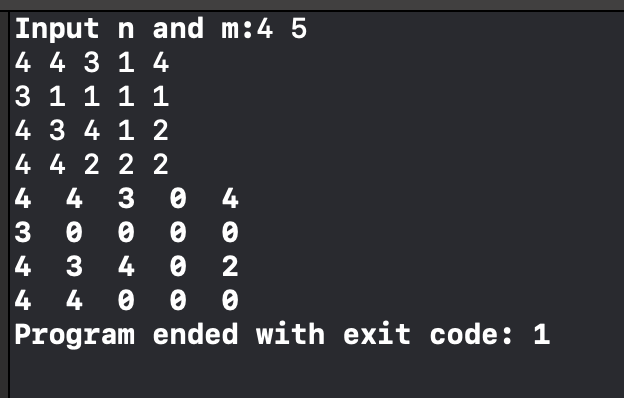


图6-1 设计题6的运行结果

9.

利用递归函数void breaks(char s[],int start ,int rest)来实现数的拆分。输入一个数n，在main（）函数中通过一个for循环，首先让变量i等于n，输出n，然后i自减，直至i=1为止。每次循环中都调用函数breaks(s,i,n-i),在该函数中，从rest开始再次进行拆分，直至rest拆分至0，输出。

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void  breaks(char s[],int op ,int rest);

int main()

{

    int n,i;

    char s[20];

    printf("Input an interger:\n");

    scanf("%d",&n);

    for(i=n;i>=1;i--)

    {

        if(i==n) printf("%d\n",n);

        else{

        s[0]=i+'0';

        breaks(s,i,n-i);

        printf("\n");

        }

    }

    return 0;

}

void  breaks(char s[],int op ,int rest)

{

    int k;

    static char t[20]={0};

    char res[20]={0},str[3]={0};

    if(rest==0)

    {

        if(s[0]==t[0]) printf(",%s",s);

        else printf("%s",s);

        strcpy(t,s);

    }

        for(k=rest;k>0;k--)

        {

            if(k>op) continue;

            else

            {

                str[0]='+';

                str[1]=k+'0';

                strcpy(res,s);strcat(s,str);

                breaks(s,k,rest-k);strcpy(s,res);

            }

        }

}

10.

定义一个外部二维数组s[M][M],则其中的初值自动赋为0.定义一个一直在自增变量k，并赋初值1，自增至M\*M为止。二维数组向右、向下、向左再向上不断赋值，每个过程都要检测到前面一个数为0再赋值，不然则退出循环向下进行。最后输出该二维数组。

#include <stdio.h>

#define M 6

int s[M][M];

int main(void)

{

    int i=0,j=0,k=1;

    s[0][0]=1;

    while (k<M\*M) {

        while (j<M-1&&!s[i][j+1]) {

            s[i][++j]=++k;

        }

        while (i<M-1&&!s[i+1][j]) {

            s[++i][j]=++k;

        }

        while (j>0&&!s[i][j-1]) {

            s[i][--j]=++k;

        }

        while (i>0&&!s[i-1][j]) {

            s[--i][j]=++k;

        }

    }

    for (i=0; i<M; i++) {

        for (j=0; j<M; j++) {

            printf("%6d",s[i][j]);

        }

        printf("\n");

    }

    return 0;

}

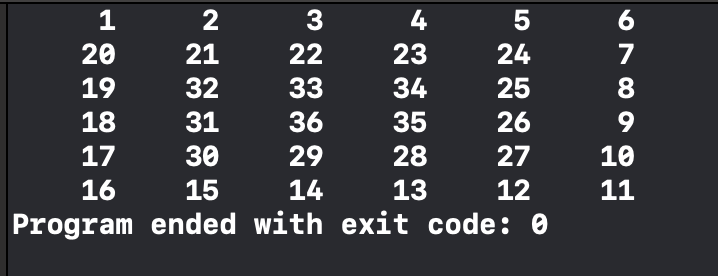


图10-1 设计题10的运行结果

**5.4小结**

在本次实验数组程序设计实验中，其中运用了大量与数组有关的程序设计，让我在这过程中，更好地理解了数组的运用，以及初步运用了二维数组。

在第一题程序修改题中，我调试多次仍出错后，老师告诉我是因为数组初始化时固定了数组的大小，之后将字符串拼接，导致数组中存放不下过多的字符，便挤占了其后另一个数组的存储空间。这个过程让我更深刻地体会了数组在内存中的存放。

在程序修改的转圈报数题中，原程序编写采用的方法让我觉得十分巧妙，体会到自己离写出优秀的程序之间还有一定的差距。之后的改写时，我想用跳过数组元素置为0的元素，但是忽略了数组元素置为0可能连续，应该都跳过。这个debug过程中我体会到了自己思维的局限。

在程序设计第二题中，我学会了编写菜单，由根据用户的值执行相应的程序，并且学会了若需按顺序执行时，如何解决这个问题。并且在这题中我也学会了用二维数组表示一系列字符串。在第三题中，我通过编写体会到了二分法查找。第六题中我学会了，当需要保留一个原值数据时，可以建立与原值相同的副本来防止数据的丢失。第九题我思考了很久，确定了大致方向为递归拆分，但是很难处理其中的加号和逗号的输出细节，于是通过查找资料以及和同学一起交流研究，通过使用一系列的字符串函数，完成了这个程序。第十题顺时针螺旋数，我也思考了很久，一开始想用行数减1，列数减1的方法来判断何时改变方向，也是在查阅资料之后，明白了将数组初值赋为0，判断前方的值是否为0来决定是否改变方向。这些设计的精巧是我需要学习的。

我还是希望自己多看书，多实践，遇到难题多钻研，提升自己的程序设计思想。

实验七 结构与联合实验

7.1 表达式求值的实验验证

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 表达式 | 计算值 | 验证值 |
| 1 | （++p）->x | 100 | 100 |
| 2 | p++,p->c | ‘B’ | ‘B’ |
| 3 | \*p++->t,\*p->t | ‘U’ | ‘U’ |
| 4 | \*(++p)->t | ‘x’ | ‘x’ |
| 5 | \*++p->t | ‘V’ | ‘V’ |
| 6 | ++\*p->t | ‘V’ | ‘V’ |

表7-1 表达式求值及验证

7.2 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,\*pp;

    int s[] = {1,2,3,4,5,6,7,8,0};

    int i;

    create\_list(&head, s);

    p = head;

    while (p) {

        printf("%d\t",p->data);

        p = p->next;

    }

    printf("\n");

    p = head;

    for (i=0; i<8; i++) {

        pp=p->next;

        free(p);

        p=pp;

    }

    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;

}

修改部分：将create\_list函数原来的参数struct s\_list \*headp改成了struct s\_list \*\*headp，因为需要通过参数headp传出head的值，如果不修改则head的值在函数外不会被改变。

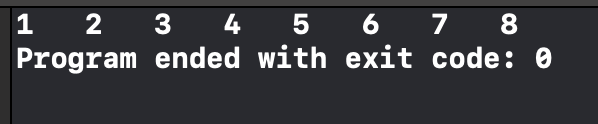


图7.2.1 修改题（1）的运行结果

（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,\*pp;

    int s[] = {1,2,3,4,5,6,7,8,0};

    int i;

    create\_list(&head, s);

    p = head;

    while (p) {

        printf("%d\t",p->data);

        p = p->next;

    }

    printf("\n");

    p = head;

    for (i=0; i<8; i++) {

        pp=p->next;

        free(p);

        p=pp;

    }

    return 0;

}

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

{

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

    if (p[0]==0);

    else {

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

        loc\_head->data = \*p++;

        loc\_head->next = NULL;

        current = loc\_head;

        while (\*p) {

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

            loc\_head->data = \*p++;

            loc\_head->next=current;

            current=loc\_head;

        }

    }

    \*headp=loc\_head;

}

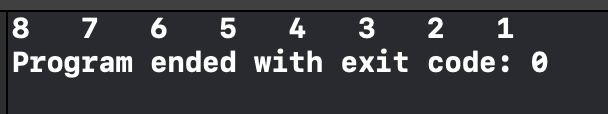


图7.2.2 修改题（2）的运行结果

7.3 程序设计

1.首先设计结构bits，在这个结构中将一个8位无符号字节从最低位向最高位声明为八个字段。然后构造了八个函数，可以输出相应biti的值。在main函数中，将struct bits实例化，然后进行初始化，并对其中一部分（bit2和bit4）赋值为1.接着一个个检测biti（i取0至7）的值，若为1，则调用相应函数。

#include <stdio.h>

struct bits{

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

};

void f0(int b)

{

    printf("the function0 %d is called!\n",b);

}

void f1(int b)

{

    printf("the function1 %d is called!\n",b);

}

void f2(int b)

{

    printf("the function2 %d is called!\n",b);

}

void f3(int b)

{

    printf("the function3 %d is called!\n",b);

}

void f4(int b)

{

    printf("the function4 %d is called!\n",b);

}

void f5(int b)

{

    printf("the function5 %d is called!\n",b);

}

void f6(int b)

{

    printf("the function6 %d is called!\n",b);

}

void f7(int b)

{

    printf("the function7 %d is called!\n",b);

}

int main(void) {

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

    struct bits bit;

    bit.bit0=0;bit.bit1=0;bit.bit2=0;bit.bit3=0;

    bit.bit4=0;bit.bit5=0;bit.bit6=0;bit.bit7=0;//初始化

    bit.bit2=1;

    bit.bit4=1;

    if (bit.bit0==1) {

        p\_fun[0](bit.bit0);

    }

    if (bit.bit1==1) {

        p\_fun[1](bit.bit1);

    }

    if (bit.bit2==1) {

        p\_fun[2](bit.bit2);

    }

    if (bit.bit3==1) {

        p\_fun[3](bit.bit3);

    }

    if (bit.bit4==1) {

        p\_fun[4](bit.bit4);

    }

    if (bit.bit5==1) {

        p\_fun[5](bit.bit5);

    }

    if (bit.bit6==1) {

        p\_fun[6](bit.bit6);

    }

    if (bit.bit7==1) {

        p\_fun[7](bit.bit7);

    }

    return 0;

}

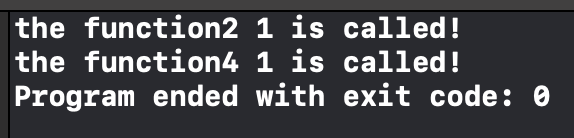
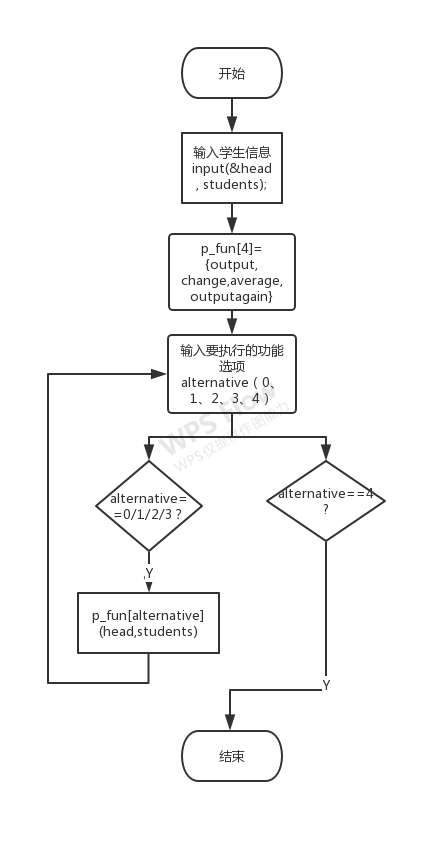


图7.3.1 程序1的运行结果

2. 

流程图7-2

#include <stdio.h>

#include <stdlib.h>

struct infomation{

    int number;

    char name[9];

    int es,ms,ps,cs;

    struct infomation \*next;

};

#define students 2 //此处修改学生人数

float averages[students];

void input(struct infomation \*\*headp,int n);

void output(struct infomation \*headp,int n);

void change(struct infomation \*headp,int n);

void average(struct infomation \*headp,int n);

void outputagain(struct infomation \*headp,int n);

int main(void) {

    struct infomation \*head,\*p,\*pp;

    int alternative;

void(\*p\_fun[4])(struct infomation \*headp,int n)={output,change,average,outputagain};

    printf("Please input students' information");

    input(&head, students);

    printf("0:output students'information\n");

    printf("1:change information\n");

    printf("2:average scores\n");

    printf("3:output again\n");

    printf("4:exit");

    scanf("%d",&alternative);

    while (alternative==0||alternative==1|| alternative==2||alternative==3) {

        p\_fun[alternative](head,students);

        printf("input your choice again");

        scanf("%d",&alternative);

    }

    p=head;

    for (int i=0; i<students;i++) {

        pp=p->next;

        free(p);

        p=pp;

    }

    return 0;

}

void input(struct infomation \*\*headp,int n){

    struct infomation \*loc\_head,\*tail;

    int i;

    loc\_head = (struct infomation \*)malloc(sizeof(struct infomation));

    scanf("%d",&loc\_head->number);

    scanf("%s",loc\_head->name);

    scanf("%d%d%d%d",&loc\_head->es,&loc\_head->ms,&loc\_head->ps,&loc\_head->cs);

    tail=loc\_head;

    for (i=1; i<n; i++) {

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

        tail=tail->next;

        scanf("%d",&tail->number);

        scanf("%s",tail->name);

        scanf("%d%d%d%d",&tail->es,&tail->ms,&tail->ps,&tail->cs);

    }

    tail->next=NULL;

    \*headp=loc\_head;

}

void output(struct infomation \*headp,int n){

    struct infomation \*p=headp;

    while (p) {

        printf("%d\t",p->number);

        printf("%s\t",p->name);

        printf("%d\t%d\t%d\t%d\n",p->es,p->ms,p->ps,p->cs);

        p=p->next;

    }

}

void change(struct infomation \*headp,int n){

    printf("change number input1\nchange name input 2\n");

    printf("change English score input 3\n");

    printf("change math score input 4\n");

    printf("change physics score input 5\n");

    printf("change Clanguage score input 6\n");

    int choice,num;

    printf("input the number of student you want to change");

    scanf("%d",&num);

    struct infomation \*p=headp;

    while (p->number!=num) {

        p=p->next;

    }

    scanf("%d",&choice);

    printf("Please input information");

    switch (choice) {

        case 1:

            scanf("%d",&p->number);break;

       case 2:

            scanf("%s",p->name);break;

        case 3:

            scanf("%d",&p->es);break;

        case 4:

            scanf("%d",&p->ms);break;

        case 5:

            scanf("%d",&p->ps);break;

        case 6:

            scanf("%d",&p->cs);break;

    }

}

void average(struct infomation \*headp,int n){

    struct infomation \*p=headp;

    int i;

    for (i=0; i<n; i++) {

        averages[i]=(p->es+p->ms+p->ps+p->cs)\*1.0/4.0;

        p=p->next;

    }

    for (i=0; i<n; i++) {

        printf("%.2f\t",averages[i]);

    }

    printf("\n");

}

void outputagain(struct infomation \*headp,int n){

    int sum[n],k;

    struct infomation \*p=headp;

    for (k=0; k<n; k++) {

        sum[k]=p->es+p->ms+p->ps+p->cs;

        p=p->next;

    }

    p=headp;

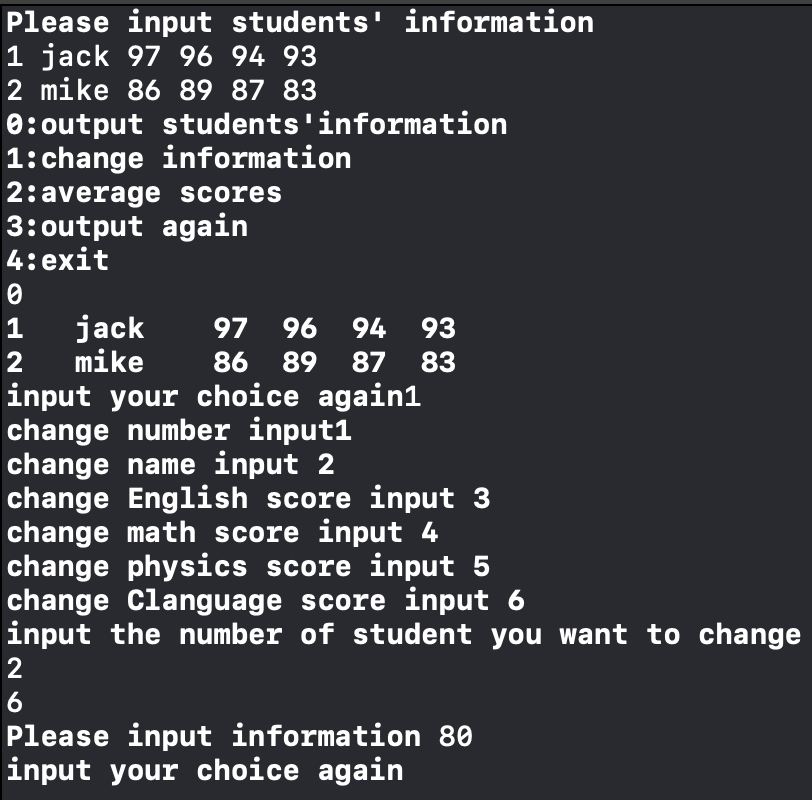
    for (k=0; k<n; k++) {

        printf("%d\t%s\t%d\t%.2f\n",p->number,p->name,sum[k],averages[k]);

        p=p->next;

    }

}

图7.3.2 程序2的运行结果

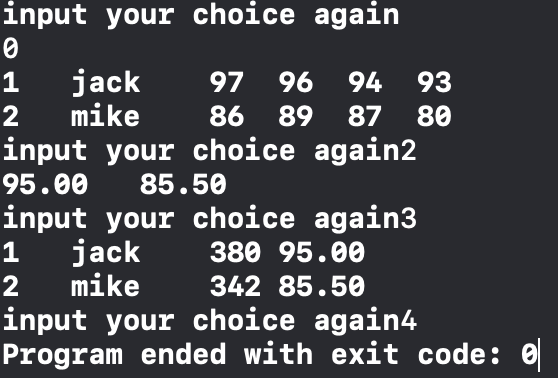
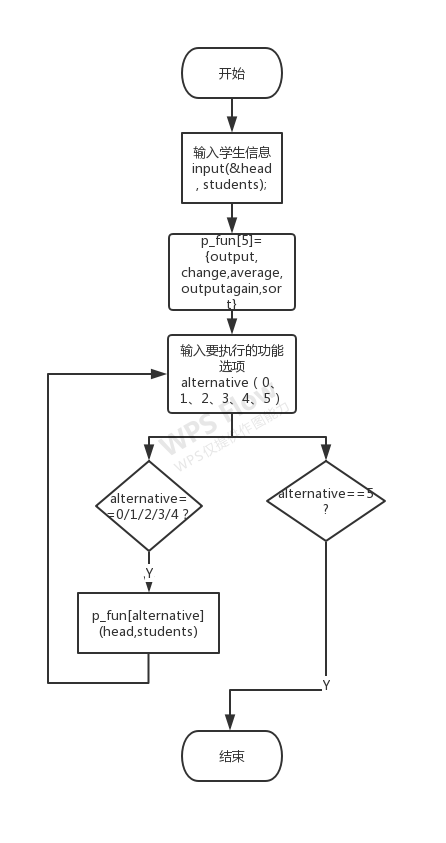


图7.3.3 程序2的运行结果

1. 

流程图7-3

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct infomation{

    int number;

    char name[9];

    int es,ms,ps,cs;

    struct infomation \*next;

};

#define students 3 //此处修改学生人数

float averages[students];

void input(struct infomation \*\*headp,int n);

void output(struct infomation \*headp,int n);

void change(struct infomation \*headp,int n);

void average(struct infomation \*headp,int n);

void outputagain(struct infomation \*headp,int n);

void sort(struct infomation \*headp,int n);

int main(void) {

    struct infomation \*head,\*p,\*pp;

    int alternative;

void (\*p\_fun[5])(struct infomation \*headp,int n)={output,change,average,outputagain,sort};

    printf("Please input students' information\n");

    input(&head, students);

    printf("0:output students'information\n");

    printf("1:change information\n");

    printf("2:average scores\n");

    printf("3:output again\n");

    printf("4:sort\n");

    printf("5:exit\n");

    scanf("%d",&alternative);

    while (alternative==0||alternative==1|| alternative==2||alternative==3||alternative==4) {

        p\_fun[alternative](head,students);

        printf("input your choice again\n");

        scanf("%d",&alternative);

    }

    p=head;

    for (int i=0; i<students;i++) {

        pp=p->next;

        free(p);

        p=pp;

    }

    return 0;

}

void input(struct infomation \*\*headp,int n){

    struct infomation \*loc\_head,\*tail;

    int i;

    loc\_head = (struct infomation \*)malloc(sizeof(struct infomation));

    scanf("%d",&loc\_head->number);

    scanf("%s",loc\_head->name);

    scanf("%d%d%d%d",&loc\_head->es,&loc\_head->ms,&loc\_head->ps,&loc\_head->cs);

    tail=loc\_head;

    for (i=1; i<n; i++) {

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

        tail=tail->next;

        scanf("%d",&tail->number);

        scanf("%s",tail->name);

        scanf("%d%d%d%d",&tail->es,&tail->ms,&tail->ps,&tail->cs);

    }

    tail->next=NULL;

    \*headp=loc\_head;

}

void output(struct infomation \*headp,int n){

    struct infomation \*p=headp;

    while (p) {

        printf("%d\t",p->number);

        printf("%s\t",p->name);

        printf("%d\t%d\t%d\t%d\n",p->es,p->ms,p->ps,p->cs);

        p=p->next;

    }

}

void change(struct infomation \*headp,int n){

    printf("change number input 1\nchange name input 2\n");

    printf("change English score input 3\n");

    printf("change math score input 4\n");

    printf("change physics score input 5\n");

    printf("change Clanguage score input 6\n");

    int choice,num;

    printf("input the number of student you want to change");

    scanf("%d",&num);

    struct infomation \*p=headp;

    while (p->number!=num) {

        p=p->next;

    }

    scanf("%d",&choice);

    printf("Please input information");

    switch (choice) {

        case 1:

            scanf("%d",&p->number);break;

       case 2:

            scanf("%s",p->name);break;

        case 3:

            scanf("%d",&p->es);break;

        case 4:

            scanf("%d",&p->ms);break;

        case 5:

            scanf("%d",&p->ps);break;

        case 6:

            scanf("%d",&p->cs);break;

    }

}

void average(struct infomation \*headp,int n){

    struct infomation \*p=headp;

    int i;

    for (i=0; i<n; i++) {

        averages[i]=(p->es+p->ms+p->ps+p->cs)\*1.0/4.0;

        p=p->next;

    }

    for (i=0; i<n; i++) {

        printf("%.2f\t",averages[i]);

    }

    printf("\n");

}

void outputagain(struct infomation \*headp,int n){

    int sum[n],k;

    struct infomation \*p=headp;

    for (k=0; k<n; k++) {

        sum[k]=p->es+p->ms+p->ps+p->cs;

        p=p->next;

    }

    p=headp;

    for (k=0; k<n; k++) {

        printf("%d\t%s\t%d\t%.2f\n",p->number,p->name,sum[k],averages[k]);

        p=p->next;

    }

}

void sort(struct infomation \*headp,int n){

    int i,j,temp;

    char tempname[9];

    struct infomation \*p=headp,\*pp=headp->next;

    for (i=0; i<n-1; i++,p=p->next,pp=p->next) {

        for (j=i+1; j<n; j++) {

            if (averages[i]>averages[j]) {

                temp=p->number;p->number=pp->number;pp->number=temp;

            strcpy(tempname,p->name);strcpy(p->name,pp->name);strcpy(pp->name,tempname);

                temp=p->es;p->es=pp->es;pp->es=temp;

                temp=p->ms;p->ms=pp->ms;pp->ms=temp;

                temp=p->ps;p->ps=pp->ps;pp->ps=temp;

                temp=p->cs;p->cs=pp->cs;pp->cs=temp;

            }

            pp=pp->next;

        }

    }

    printf("Already sort!\n");

}

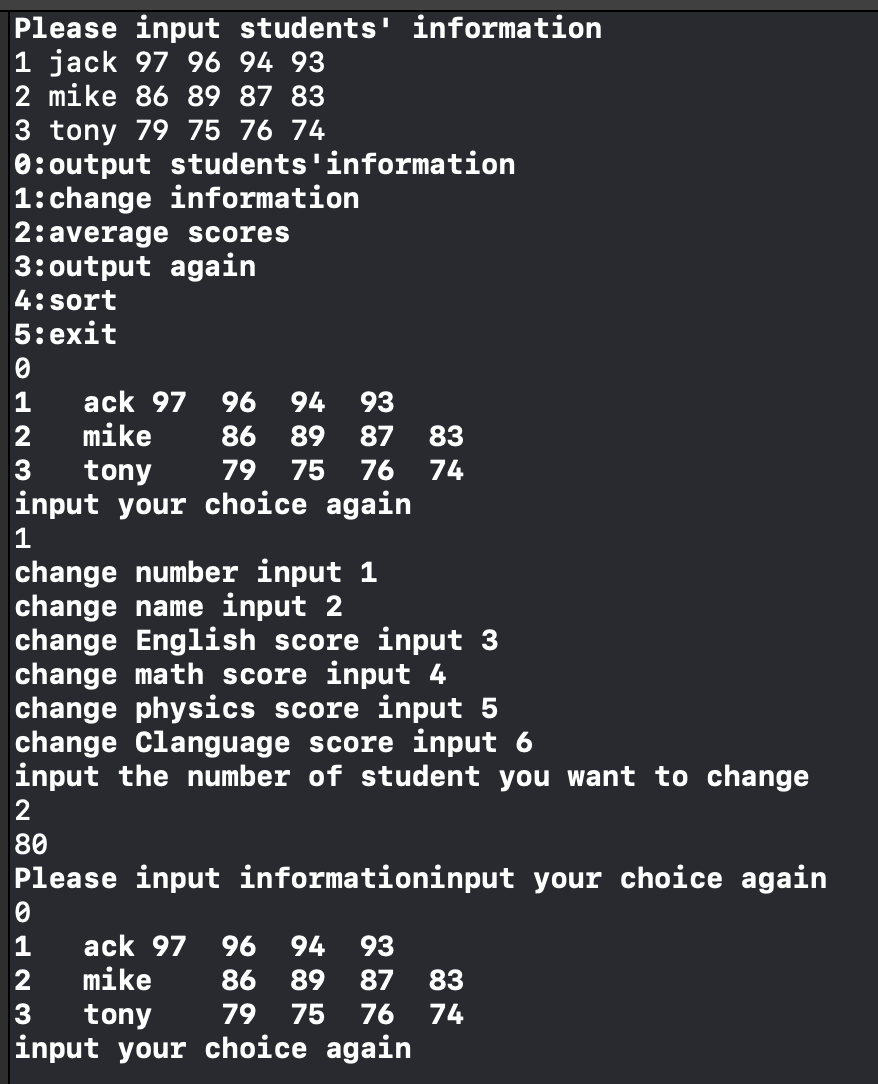


图7.3.4 程序3的运行结果

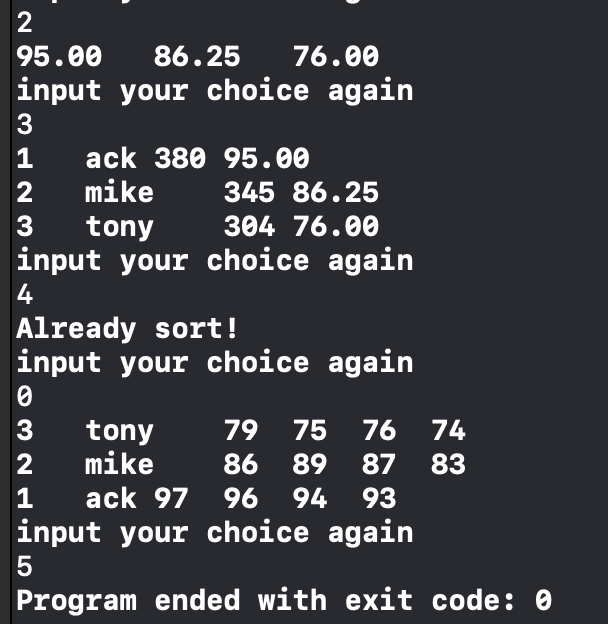
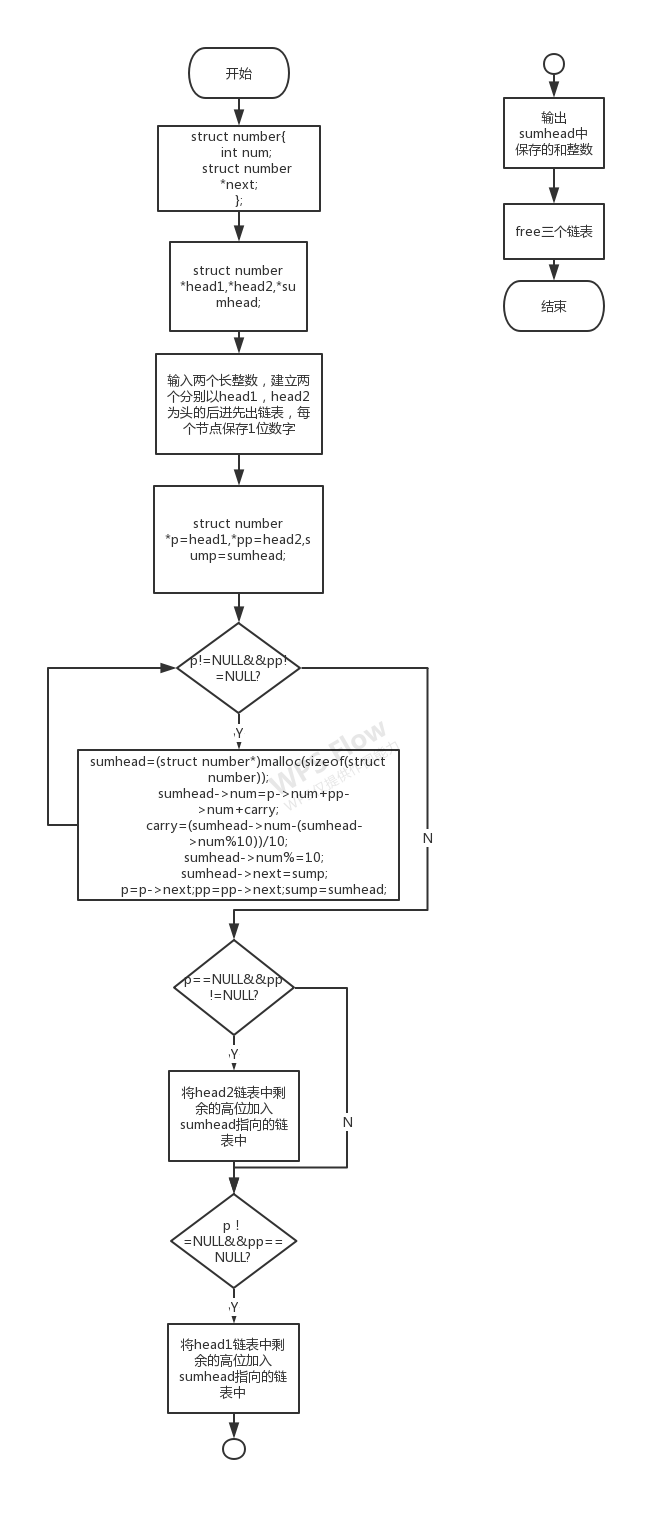


图7.3.5 程序3的运行结果

5.

流程图7-5

#include <stdio.h>

#include<stdlib.h>

#include <ctype.h>

struct number{

    int num;

    struct number \*next;

};

int main(void){

    char c;

    struct number \*head1,\*head2,\*sumhead;

    struct number \*p,\*pp,\*sump;

    int carry=0;

    //输入部分

    head1=(struct number\*)malloc(sizeof(struct number));

    head2=(struct number\*)malloc(sizeof(struct number));

    c=getchar();

    head1->num=c-'0';head1->next=NULL;

    for (p=head1,c=getchar(); isdigit(c); p=head1,c=getchar()) {

        head1=(struct number\*)malloc(sizeof(struct number));

        head1->num=c-'0';

        head1->next=p;

    }

    c=getchar();

    head2->num=c-'0';head2->next=NULL;

    for (p=head2,c=getchar(); isdigit(c); p=head2,c=getchar()) {

        head2=(struct number\*)malloc(sizeof(struct number));

        head2->num=c-'0';

        head2->next=p;

    }

    //求和部分

    p=head1;pp=head2;

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

    sumhead->num=p->num+pp->num+carry;

    carry=(sumhead->num-(sumhead->num%10))/10;

    sumhead->num%=10;

    sumhead->next=NULL;

    p=p->next;pp=pp->next;

    for (sump=sumhead;p!=NULL&&pp!=NULL; ) {

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

        sumhead->num=p->num+pp->num+carry;

        carry=(sumhead->num-(sumhead->num%10))/10;

        sumhead->num%=10;

        sumhead->next=sump;

        p=p->next;pp=pp->next;sump=sumhead;

    }

    if (p==NULL&&pp!=NULL) {

        while (pp) {

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

            sumhead->num=pp->num+carry;

            carry=(sumhead->num-(sumhead->num%10))/10;

            sumhead->num%=10;

            sumhead->next=sump;

            pp=pp->next;sump=sumhead;

        }

    }

    if (p!=NULL&&pp==NULL) {

        while (p) {

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

            sumhead->num=p->num+carry;

            carry=(sumhead->num-(sumhead->num%10))/10;

            sumhead->num%=10;

            sumhead->next=sump;

            p=p->next;sump=sumhead;

        }

    }

    if (carry) {

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

        sumhead->num=carry;

        sumhead->next=sump;

    }

    //print sum

    p=sumhead;

    while (p) {

        printf("%d",p->num);

        p=p->next;

    }

    printf("\n");

    //free部分

    p=head1;

    while (p) {

        pp=p->next;

        free(p);

        p=pp;

    }

   p=head2;

    while (p) {

        pp=p->next;

        free(p);

        p=pp;

    }

    p=sumhead;

    while (p) {

        pp=p->next;

        free(p);

        p=pp;

    }

    return 0;

}

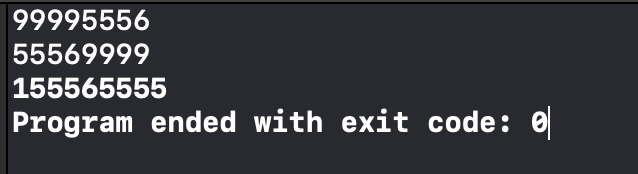
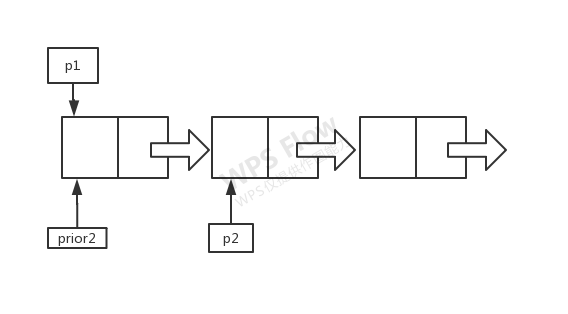
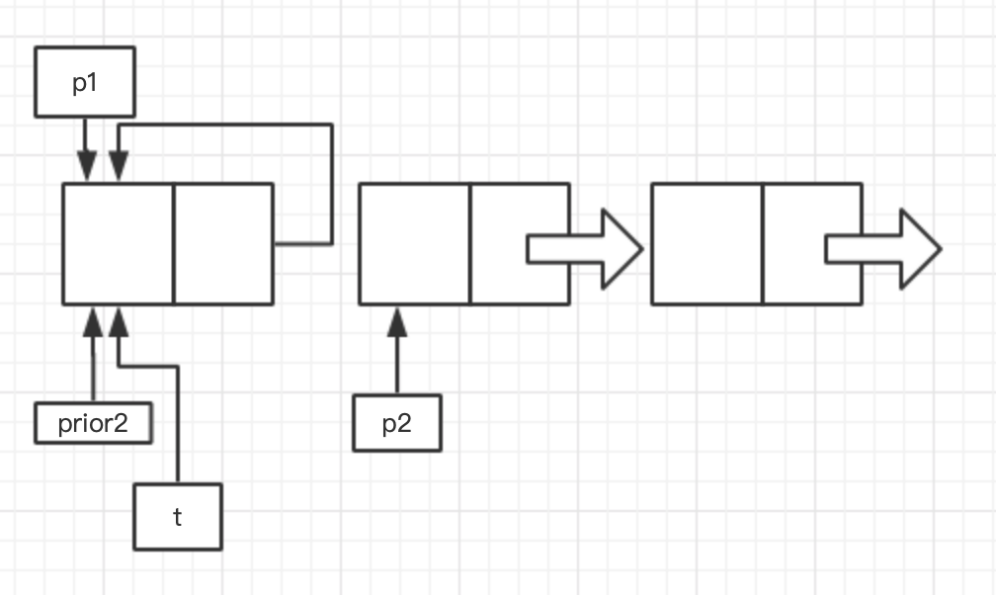


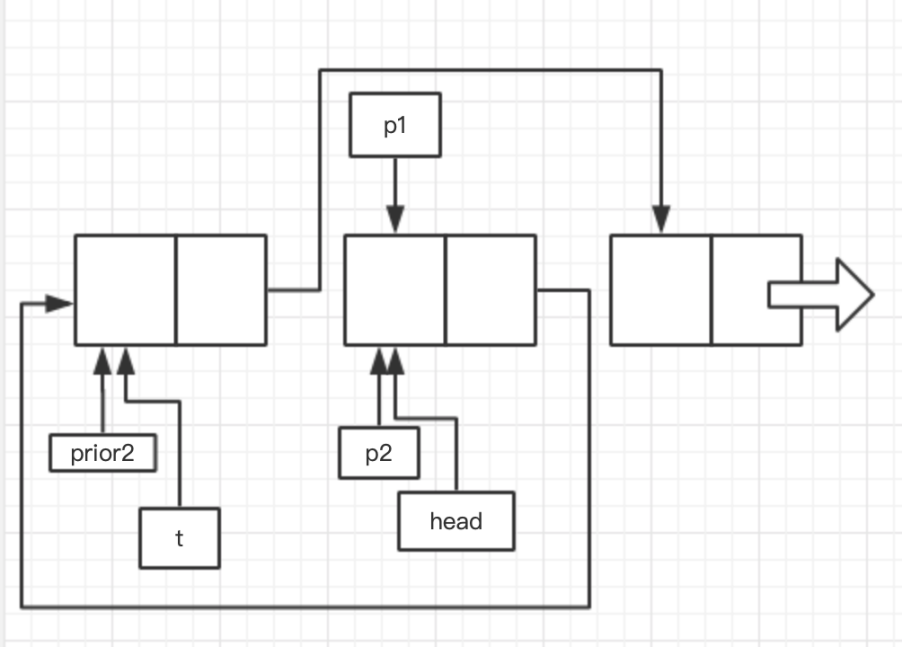
图7.3.6 程序5的运行结果



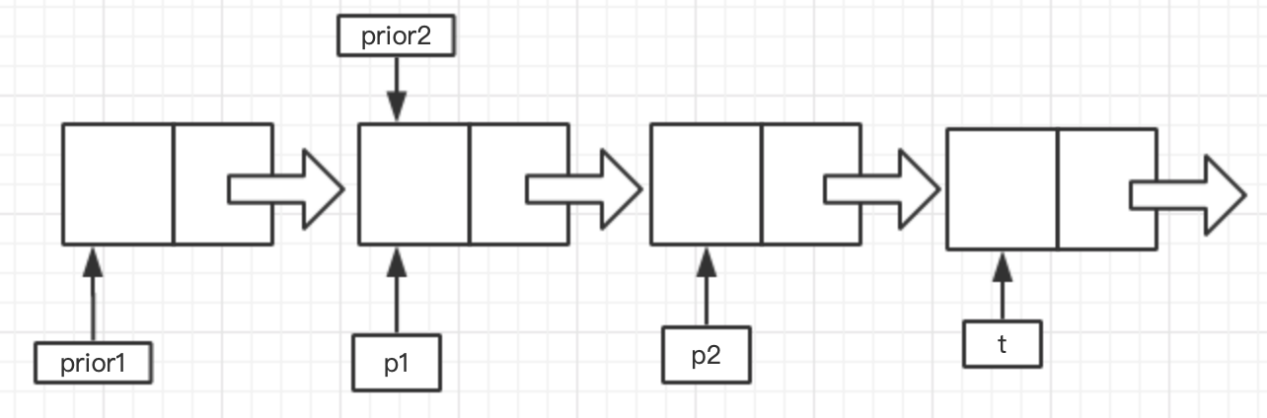
对头节点处理步骤1

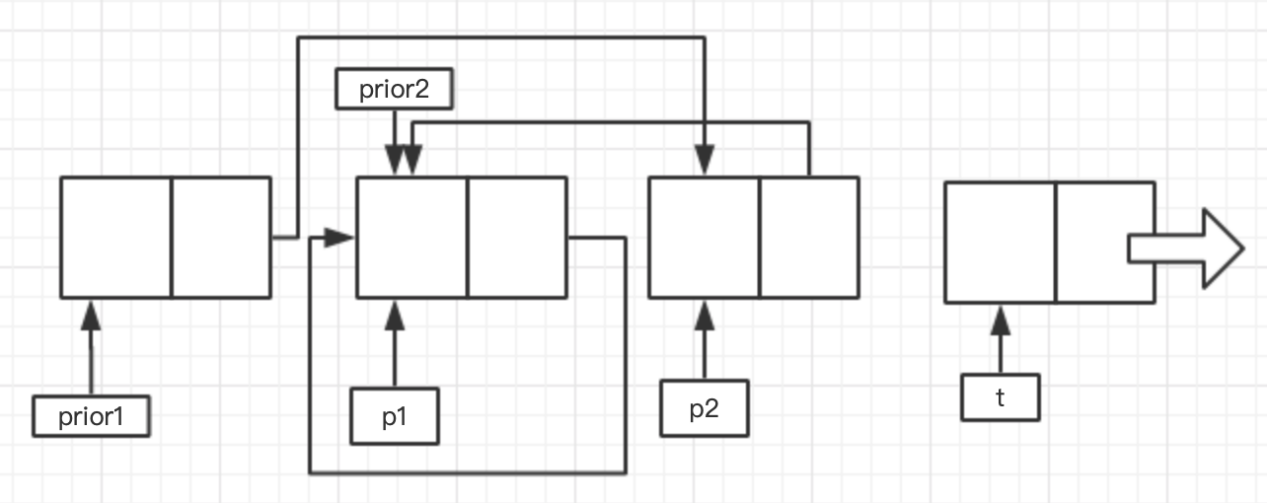


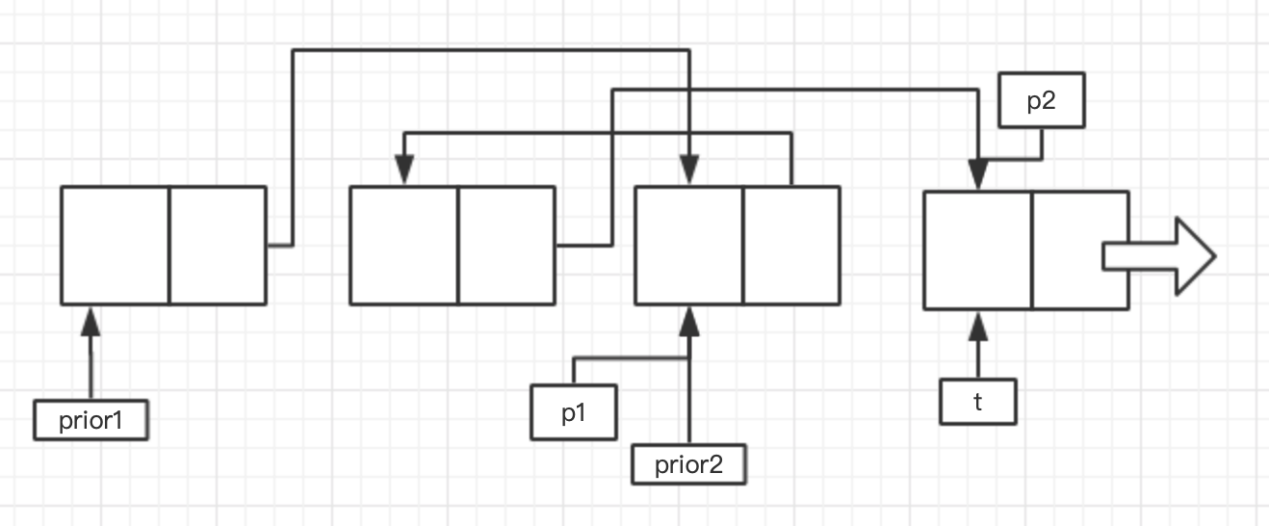
对头节点处理步骤2



对头节点处理步骤3

对后续节点交换处理步骤1

对后续节点交换处理步骤2



对后续节点交换处理步骤3

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

    struct infomation \*prior1,\*prior2,\*p1,\*p2,\*t;

    int i,j;

    float temp;

    p1=\*headp;

    for (p2=p1->next,prior2=p1,i=1; p2!=NULL&&i<students; prior2=p2,p2=p2->next,i++) {

        if (averages[0]>averages[i]) {

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

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

            p2->next=t;p1=p2;

            temp=averages[0];averages[0]=averages[i];averages[i]=temp;

        }

    }

    for (prior1=\*headp,p1=prior1->next,i=1; p1->next!=NULL; prior1=p1,p1=p1->next,i++) {

        for (p2=p1->next,prior2=p1,j=i+1; p2!=NULL&&j<students; prior2=p2,p2=p2->next,j++) {

            if (averages[i]>averages[j]) {

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

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

                p1->next=t;p1=p2;

                temp=averages[i];averages[i]=averages[j];averages[j]=temp;

            }

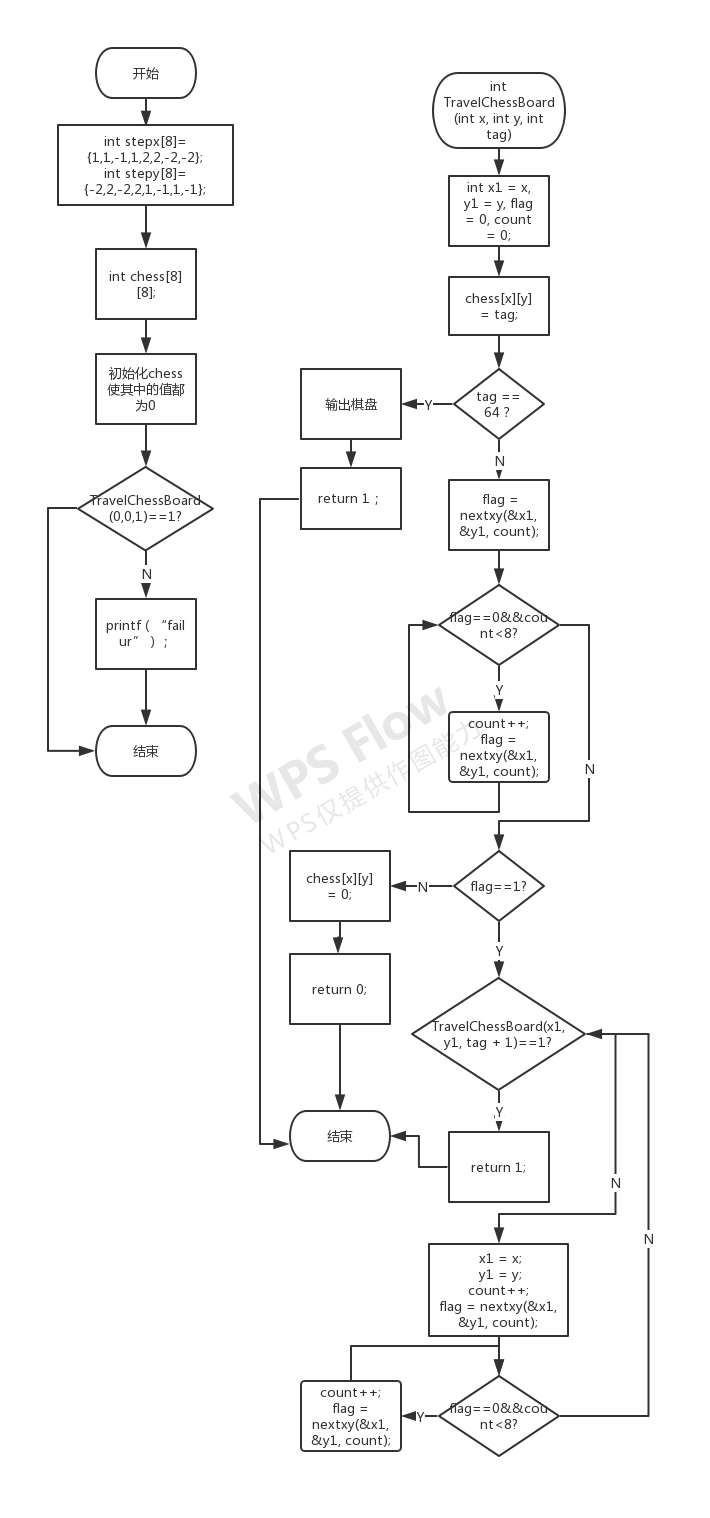
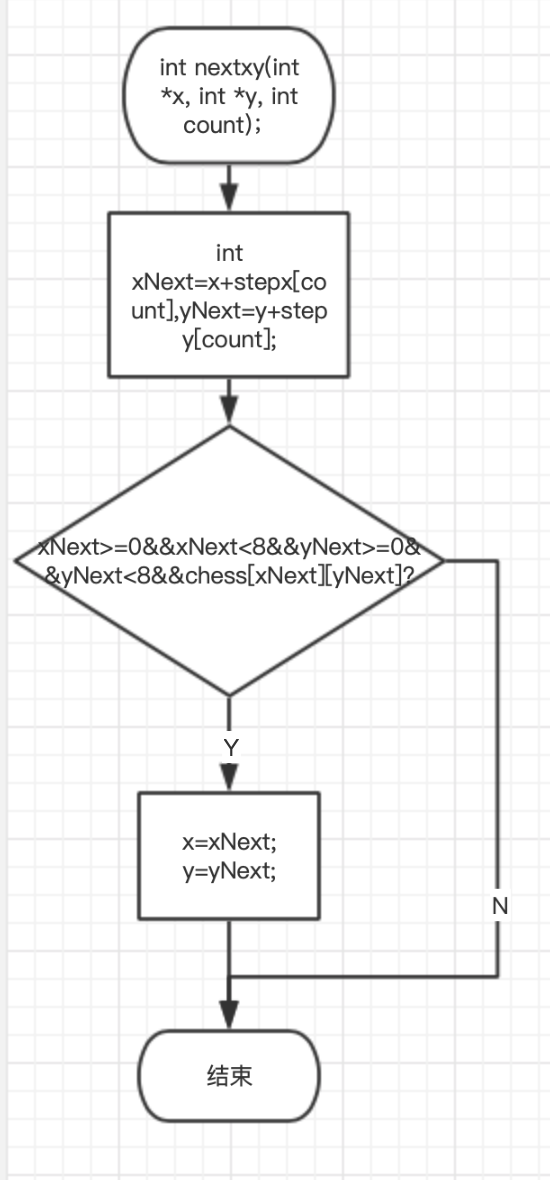
        }

    }

    printf("Already sort!\n");

}

1. 马踏棋盘问题
2. 深度优先遍历和递归函数的结合



1. 结合贪心算法的深度优先遍历

利用贪心算法思想，先对每次要走的下一步进行排序。统计所要走的下一步的位置拥有多少种再继续走下一步的方向，方向越少的越优先。然后针对每一步走出后进行深度优先遍历，若遇到不对的情况，再依次出栈接着走下一种方向。

7.4 小结

本次实验意在熟悉和掌握结构的说明和引用、结构的指针、结构数组以及函数使用结构的方法，掌握动态存储分配函数的用法，掌握自引用结构以及单向链表的创建、遍历、结点的增删和交换、查找等，了解字段结构和联合的用法。

程序设计题1让我更加深刻地体会到了字段结构的编写和使用。而在程序设计题2、3、7中，分别熟悉了单向链表的创建、遍历、查找、数据域的修改和交换、交换结点指针域等。这对深入理解单向链表是非常有效的。在程序设计题5中，学会使用后进先出的单向链表以及对十进制超大整数加法的设计。最后的马踏棋盘问题，我还没有解决，但是也深入学习了一下关于栈和深度优先遍历、队列和广度优先遍历、贪心算法等有关概念和方法。我相信随着学习的深入以及对数据结构的更深的理解，我可以解决这个问题。

# 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

........