

**课 程 实 验 报 告**

**课程名称： C语言程序设计实验**

**专业班级： 网络空间安全2105班**

**学 号： 网络空间安全**

**姓 名： 李翔**

**指导教师： 王同洋**

**报告日期： 2021.12.17**

**网络空间安全学院**

实验6 指针实验

6.1、实验目的

（1）熟练掌握指针的说明、赋值、使用。

（2）掌握用指针引用数组的元素，熟悉指向数组的指针的使用。

（3）熟练掌握字符数组与字符串的使用，掌握指针数组及字符指针数组的用法。

（4）掌握指针函数与函数指针的用法。

（5）掌握带有参数的main函数的用法。

6.2、实验题目及要求

### 6.2.1源程序改错题

在下面所给的源程序中，函数strcopy(t, s)的功能是将字符串s复制给字符串t，并且返回串t的首地址。请单步跟踪程序，根据程序运行时出现的现象或观察到的字符串的值，分析并排除源程序的逻辑错误，使之能按照要求输出如下结果：

Input a string:

programming↙ （键盘输入）

programming

Input a string again:

language↙ （键盘输入）

language

#include<stdio.h>

char \*strcopy(char \*, const char \*);

int main(void)

{

char \*s1, \*s2, \*s3;

printf("Input a string:\n", s2);

scanf("%s", s2);

strcopy(s1, s2);

printf("%s\n", s1);

printf("Input a string again:\n", s2);

scanf("%s", s2);

s3 = strcopy(s1, s2);

printf("%s\n", s3);

return 0;

}

/\*将字符串s复制给字符串t，并且返回串t的首地址\*/

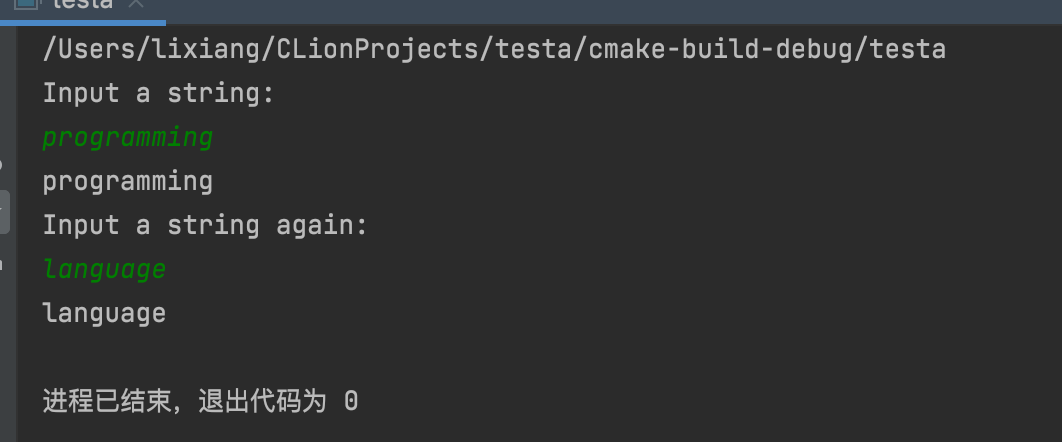
程序存在的错误：

1. strcopy函数返回的并不是串t的首地址而是末地址，应该另外设置一个指针存储头指针；
2. 字符串存在的区域是常量区，不可更改，如果想要修改，应该给字符串动态分配一个空间并使所设指针指向所申请的空间。

程序源代码：

#include<stdio.h>  
#include<stdlib.h>  
char \*strcopy(char \*, const char \*);  
int main(void)  
{  
 char \*s1, \*s2, \*s3;  
 printf("Input a string:\n");  
 s2=(char \*)malloc(100);  
 s1=(char \*)malloc(100);  
 scanf("%s", s2);  
 strcopy(s1, s2);  
 printf("%s\n", s1);  
 printf("Input a string again:\n");  
 scanf("%s",s2);  
 s3=strcopy(s1,s2);  
 printf("%s\n",s3);  
 return 0;  
}  
char \* strcopy(char \*t, const char \*s)  
{  
 char \*temp=t;  
 while((\*t++=\*s++));  
 return temp;  
}

程序运行截图：



### 6.2.2源程序完善、修改替换题

（1）下面程序中函数strsort用于对字符串进行升序排序，在主函数中输入N个字符串（字符串长度不超过49）存入通过malloc动态分配的存储空间，然后调用strsort对这N个串按字典序升序排序。

①请在源程序中的下划线处填写合适的代码来完善该程序。

#include<stdio.h>

#include<\_\_\_\_\_\_\_\_\_>

#include<string.h>

#define N 4

/\*对指针数组s指向的size个字符串进行升序排序\*/

void strsort(char \*s[], int size)

{

\_\_\_\_\_\_\_temp;

int i, j;

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

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

if (\_\_\_\_\_\_\_\_\_\_\_)

{

temp = s[j];

\_\_\_\_\_\_\_\_\_\_\_;

s[j+1] = temp;

}

}

int main()

{

int i;

char \*s[N], t[50];

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

{

gets(t);

s[i] = (char \*)malloc(strlen(t)+1);

strcpy(\_\_\_\_\_\_\_);

}

strsort(\_\_\_\_\_\_\_\_);

for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}

return 0;

}

补充之后的代码：

#include<stdio.h>  
#include<stdlib.h>  
#include<string.h>  
#define N 4  
/\*对指针数组s指向的size个字符串进行升序排序\*/  
void strsort(char \*s[], int size)  
{  
 char \*temp;  
 int i, j;  
 for(i=0; i<size-1; i++)  
 for (j=0; j<size-i-1; j++)  
 if (\*s[j]>\*s[j+1])  
 {  
 temp=s[j];  
 s[j]=s[j+1];  
 s[j+1]=temp;  
 }  
}  
  
int main()  
{  
 int i;  
 char \*s[N], t[50];  
 for (i=0; i<N; i++)  
 {  
 gets(t);  
 s[i] = (char \*)malloc(strlen(t)+1);  
 strcpy(s[i],t);  
 }  
 strsort(s,N);  
 for (i=0; i<N; i++) {puts(s[i]); free(s[i]);}  
 return 0;  
}

②数组作为函数参数其本质类型是指针。例如，对于形参char \*s[]，编译器将其解释为char \*\*s，两种写法完全等价。请用二级指针形参重写strsort函数，并且在该函数体的任何位置都不允许使用下标引用。

改写后的函数：

void strsort(char \*\*s, int size)  
{  
 char \*temp;  
 int i, j;  
 for(i=0; i<size-1; i++)  
 for (j=0; j<size-i-1; j++)  
 if (\*\*(s+j)>\*\*(s+j+1))  
 {  
 temp=\*(s+j);  
 \*(s+j)=\*(s+j+1);  
 \*(s+j+1)=temp;  
 }  
}

（2）下面源程序通过函数指针和菜单选择来调用库函数实现字符串操作；串复制strcpy、串连接strcat或串分解strtok。

①请在源程序中的下划线处填写合适的代码来完善该程序，使之能按照要求输出下面结果：

1 copy string.

2 connect string.

3 parse string.

4 exit.

input a number (1-4) please!

2↙ （键盘输入）

input the first string please!

the more you learn,↙ （键盘输入）

input the second string please!

the more you get. ↙ （键盘输入）

the result is the more you learn, the more you get.

# include<stdio.h>

# include<string.h>

int main (void)

{

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

char a[80], b[80], \*result;

int choice;

while(1)

{

do

{

printf("\t\t1 copy string.\n");

printf("\t\t2 connect string.\n");

printf("\t\t3 parse string.\n");

printf("\t\t4 exit.\n");

printf("\t\tinput a number (1-4) please.\n");

scanf("%d", &choice);

}while(choice<1 || choice>4);

switch(choice)

{

case 1: p = strcpy; break;

case 2: p = strcat; break;

case 3: p = strok; break;

case 4: goto down;

}

getchar();

printf("input the first string please!\n");

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

printf("input the second string please!\n");

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

result = \_\_\_\_\_\_\_\_\_\_\_(a, b);

printf("the result is %s\n", result);

}

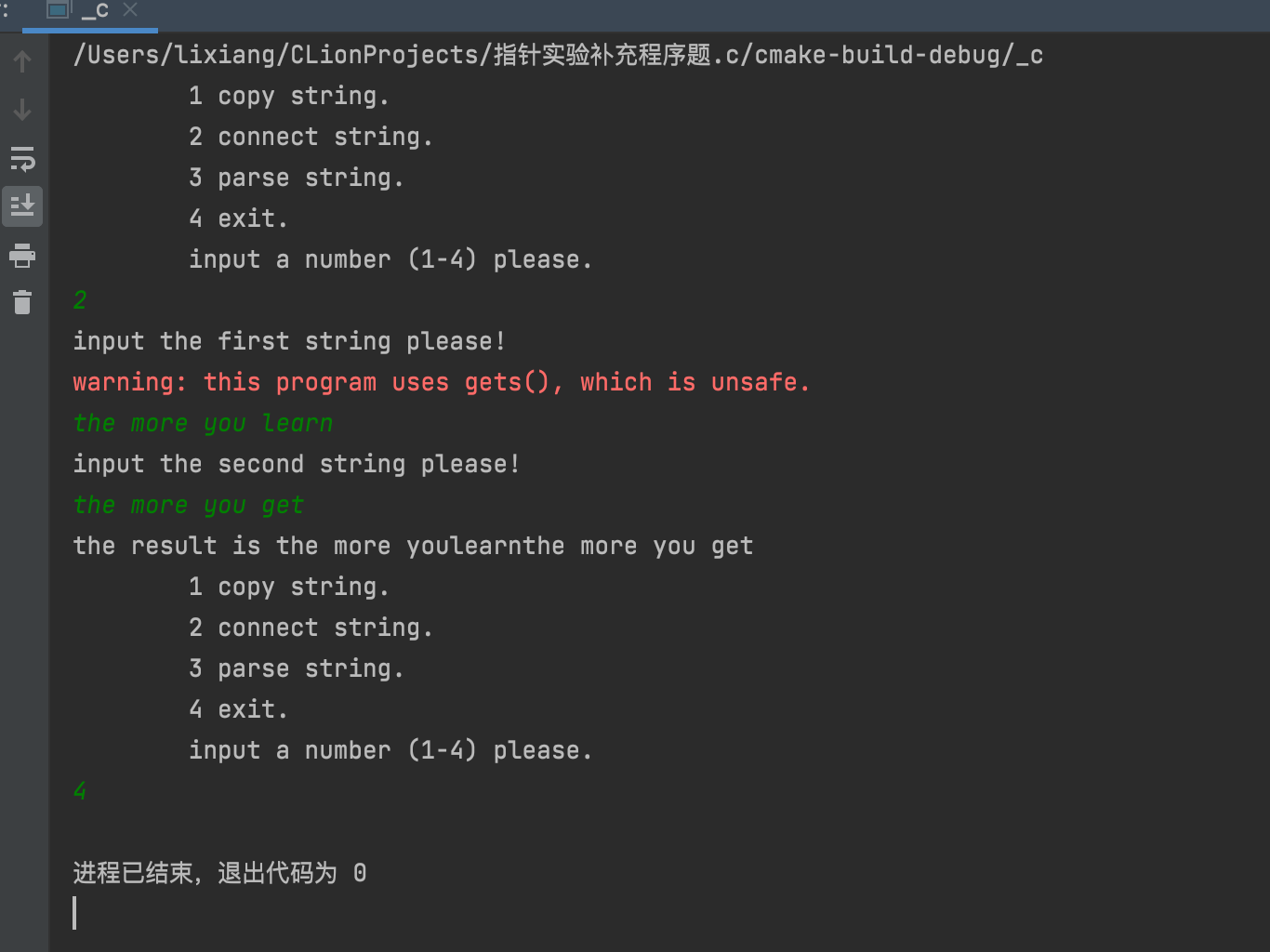
down:

return 0;

}

程序源代码及程序运行截图：

/\*  
1 copy string.  
2 connect string.  
3 parse string.  
4 exit.  
input a number (1-4) please!  
2↙ （键盘输入）  
input the first string please!  
the more you learn,↙ （键盘输入）  
input the second string please!  
the more you get. ↙ （键盘输入）  
the result is the more you learn, the more you get.\*/  
# include<stdio.h>  
# include<string.h>  
int main (void)  
{  
 char\* (\*p)(char \*,const char \*);  
 char a[80], b[80], \*result;  
 int choice;  
 while(1)  
 {  
 do  
 {  
 printf("\t\t1 copy string.\n");  
 printf("\t\t2 connect string.\n");  
 printf("\t\t3 parse string.\n");  
 printf("\t\t4 exit.\n");  
 printf("\t\tinput a number (1-4) please.\n");  
 scanf("%d", &choice);c  
 }while(choice<1 || choice>4);  
 switch(choice)  
 {  
 case 1: p=strcpy; break;  
 case 2: p=strcat; break;  
 case 3: p=strtok; break;  
 case 4: goto down;  
 }  
 getchar();  
 printf("input the first string please!\n");  
 gets(a);  
 printf("input the second string please!\n");  
 gets(b);  
 result=(\*p)(a, b);  
 printf("the result is %s\n", result);  
 }  
 down:  
 return 0;  
}



②函数指针的一个用途是用户散转程序，即通过一个转移表（函数指针数组）来实现多分枝函数处理，从而省去了大量的if语句或者switch语句。转移表中存放了各个函数的入口地址（函数名），根据条件的设定来查表选择执行相应的函数。请使用转移表而不是switch语句重写以上程序。

/\*  
1 copy string.  
2 connect string.  
3 parse string.  
4 exit.  
input a number (1-4) please!  
2↙ （键盘输入）  
input the first string please!  
the more you learn,↙ （键盘输入）  
input the second string please!  
the more you get. ↙ （键盘输入）  
the result is the more you learn, the more you get.\*/  
# include<stdio.h>  
# include<string.h>  
int main (void)  
{  
 char\* (\*p[3])(char \*,const char \*);  
 char a[80], b[80], \*result;  
 int choice;  
 while(1)  
 {  
 do  
 {  
 printf("\t\t1 copy string.\n");  
 printf("\t\t2 connect string.\n");  
 printf("\t\t3 parse string.\n");  
 printf("\t\t4 exit.\n");  
 printf("\t\tinput a number (1-4) please.\n");  
 scanf("%d", &choice);  
 }while(choice<1 || choice>4);  
 p[0]=strcpy;  
 p[1]=strcat;  
 p[2]=strtok;  
 if(choice==4) goto down;  
 getchar();  
 printf("input the first string please!\n");  
 gets(a);  
 printf("input the second string please!\n");  
 gets(b);  
 result=(\*p[choice-1])(a,b);  
 printf("the result is %s\n", result);  
 }  
 down:  
 return 0;  
}

### 6.2.3跟踪调试题

请按下面的要求对源程序进行操作，并回答问题和排除错误。

（1）单步执行。进入strcpy时watch窗口中s为何值？返回main时, watch窗口中s为何值？

Answer:进入strcpy时watch窗口中s为“”，即什么都没有。返回main时, watch窗口中s为"g\U00000002\U0000001f\x80z=\xec\xfe\U0000007f"

（2）排除错误，使程序输出结果为：there is a boat on the lake.

Answer:错误在于，这里strcpy函数的返回值应该是一个指针而最后返回一个s指针的话，这个是末地址，应该返回首地址所以应该用另一个指针先存着这个首地址，最后返回首地址。

其实这里我认为还有个错误就是这个a【20】，这个数组原本会溢出，但由于是连续的地址运算导致没有出错。但其实这里的数组根本就没有用到，传入strcpy函数的第一个参数，只要是个地址就行，把这个地址作为首地址，后面是若干连续的地址。如果输出这个字符数组a就会出错。

#include "stdio.h"

char \*strcpy(char \*,char \*);

void main(void)

{

char a[20],b[60]="there is a boat on the lake.";

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

}

char \*strcpy(char \*s,char \*t)

{

while(\*s++=\*t++)

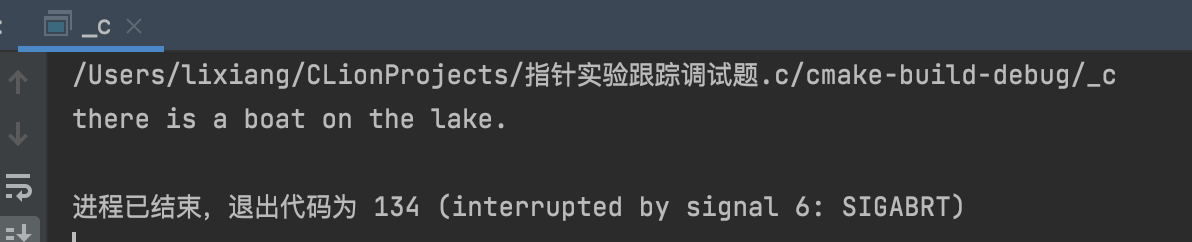
;

return (s);

}

修改过后的代码及程序运行截图

#include "stdio.h"  
char \*strcpy(char \*,char \*);  
int main(void)  
{  
 char a[20],b[60]="there is a boat on the lake.";  
 printf("%s\n",strcpy(a,b));  
 return 0;  
}  
char \*strcpy(char \*s,char \*t)  
{  
 char \*temp=s;  
 while((\*s++=\*t++));  
 return temp;  
}



### 6.2.4编程设计题

（1）指定main函数的参数

在IDE（比如DevC++）中，选择“**运行**”｜“**参数**”菜单，在 “**传递给主程序的参数**”文本框中输入main函数的参数arg1 arg2 arg3，只输入命令行中文件名后的参数，文件名不作为参数输入，参数间以空格隔开。编写程序在命令行输出这三个参数。（注意不同IDE输入参数的方式不相同，可参考各个IDE的使用手册。）

/\*在IDE（比如DevC++）中，选择“运行”｜“参数”菜单，  
 \* 在 “传递给主程序的参数”文本框中输入main函数的参数arg1 arg2 arg3，  
 \* 只输入命令行中文件名后的参数，文件名不作为参数输入，参数间以空格隔开。  
 \* 编写程序在命令行输出这三个参数。（  
 \* 注意不同IDE输入参数的方式不相同，可参考各个IDE的使用手册。）\*/  
  
#include <stdio.h>  
  
int main(int argc,char \*argv[])//指针数组，每个元素都是一个指针，指针的值就是首地址  
{  
 int n;  
 for(n=1;n<argc;n++)  
 {  
 printf("%s\n",argv[n]);  
 }  
 return 0;  
}



### 6.2.5头歌实验

实验6-1：

一个长整型变量占4个字节，其中每个字节又分成高4位和低4位。输入一个长整型变量，要求从高字节开始，依次取出每个字节的高4位和低4位并以十六进制数字字符的形式进行显示，通过指针取出每字节。

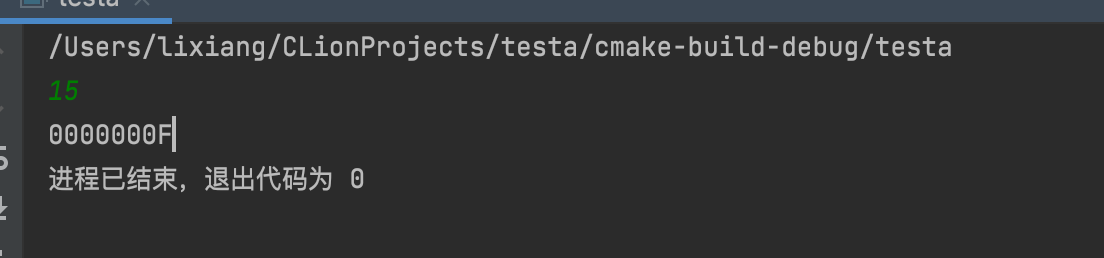
样例输入：15

样例输出：0000000F

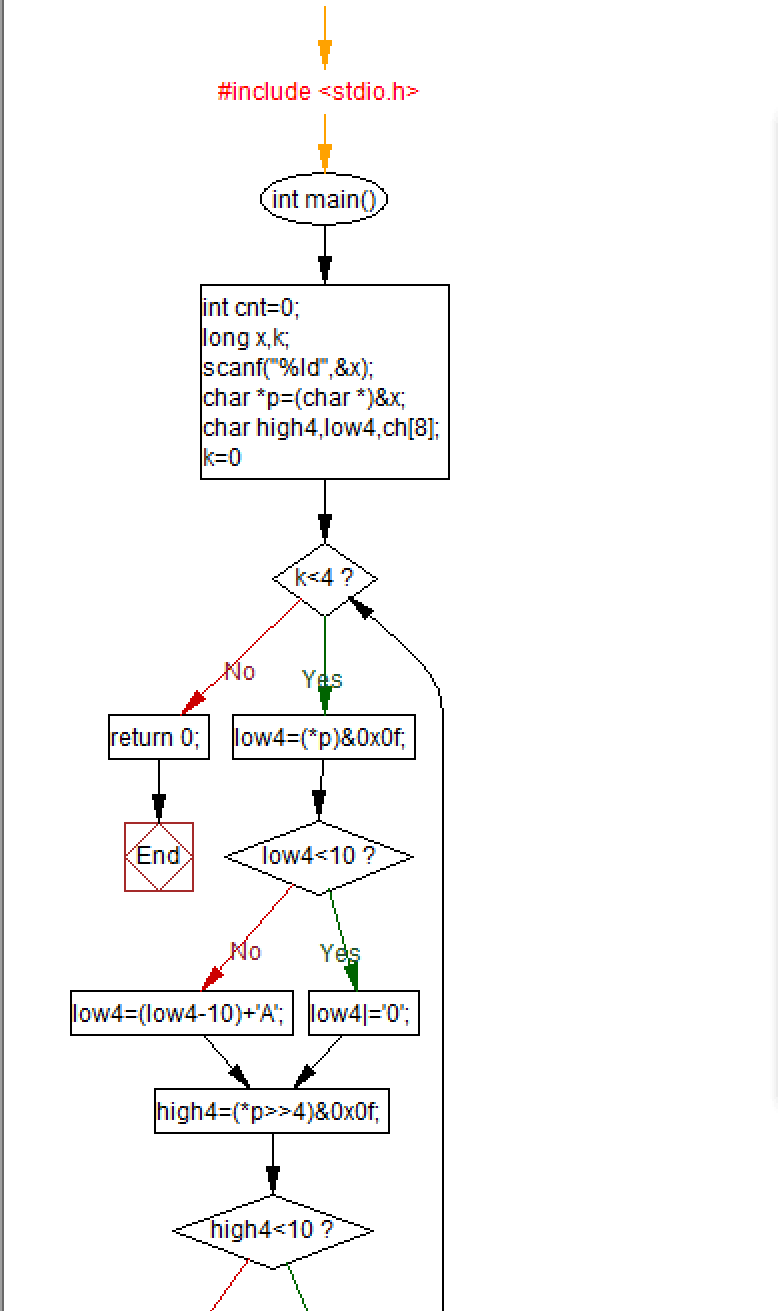
程序源代码：

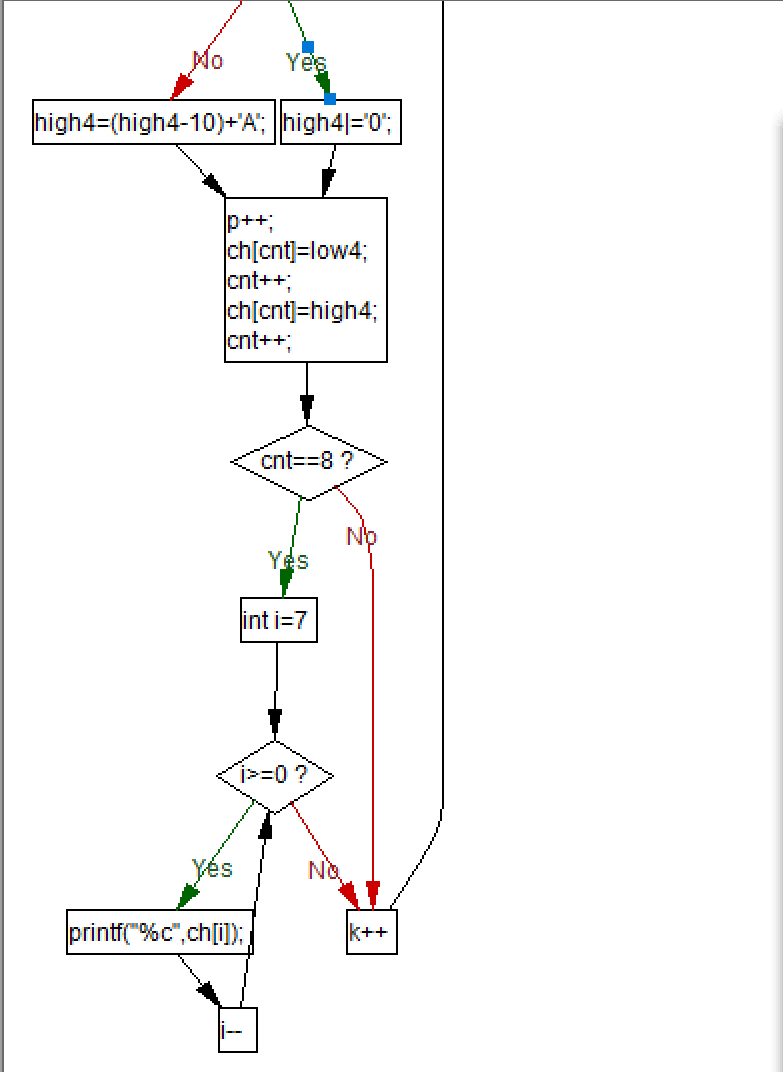
#include <stdio.h>  
int main()  
{  
 int cnt=0;  
 long x,k;  
 scanf("%ld",&x);  
 char \*p=(char \*)&x;  
 char high4,low4,ch[8];  
 for(k=0;k<4;k++)  
 {  
 low4=(\*p)&0x0f;  
 if(low4<10)  
 {  
 low4+='0';  
 }  
 else  
 {  
 low4=(low4-10)+'A';  
 }  
 high4=(\*p>>4)&0x0f;  
 if(high4<10)  
 {  
 high4+='0';  
 }  
 else  
 {  
 high4=(high4-10)+'A';  
 }  
 p++;  
 ch[cnt]=low4;  
 cnt++;  
 ch[cnt]=high4;  
 cnt++;  
 if(cnt==8)  
 {  
 for(int i=7;i>=0;i--)  
 {  
 printf("%c",ch[i]);  
 }  
 }  
 }  
 return 0;  
}

程序运行结果：



程序流程图：





实验6-2:

旋转是图像处理的基本操作，编程实现一个将一个图像逆时针旋转90°。提示：计算机中的图像可以用一个矩阵来表示，旋转一个图像就是旋转对应的矩阵。将旋转矩阵的功能定义成函数，通过使用指向数组元素的指针作为参数使该函数能处理任意大小的矩阵。要求在main函数中输入图像矩阵的行数n和列数m，接下来的n行每行输入m个整数，表示输入的图像。输出原始矩阵逆时针旋转90°后的矩阵。

样例输入：

2 3

1 5 3

3 2 4

样例输出：

3 4

5 2

1 3

程序源代码：

#include<stdio.h>

void spinning(int m,int n,int a[],int b[][n]);

int main()

{

int i,j,m,n;

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

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

for(i=0;i<m\*n;i++)

{

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

}

spinning(m,n,a,b);

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

{

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

{

if(j==m-1) printf("%d",b[i][j]);

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

}

printf("\n");

}

return 0;

}

void spinning(int m,int n,int a[],int b[][n])

{

int \*p=a;

int i,j;

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

{

for(j=n-1;j>=0;j--,p++)

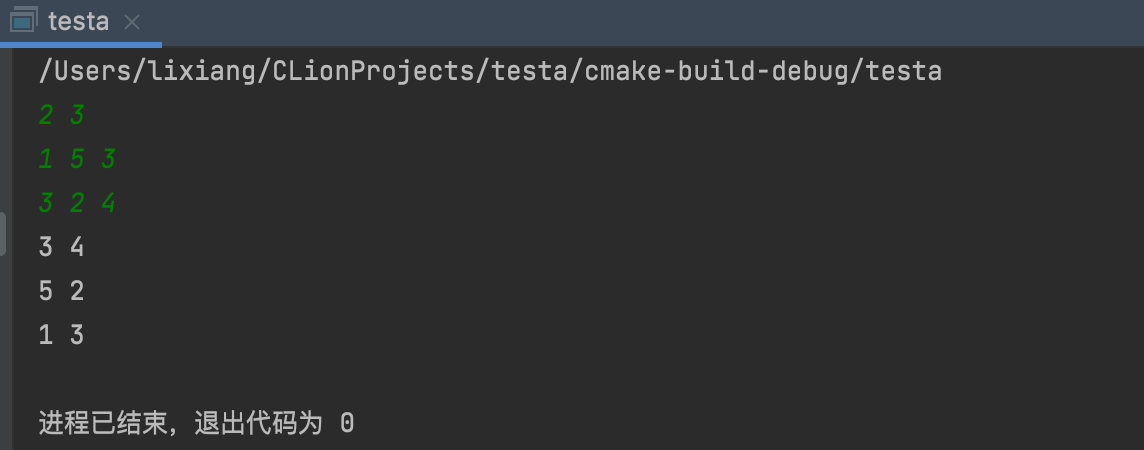
{

b[j][i]=\*p;

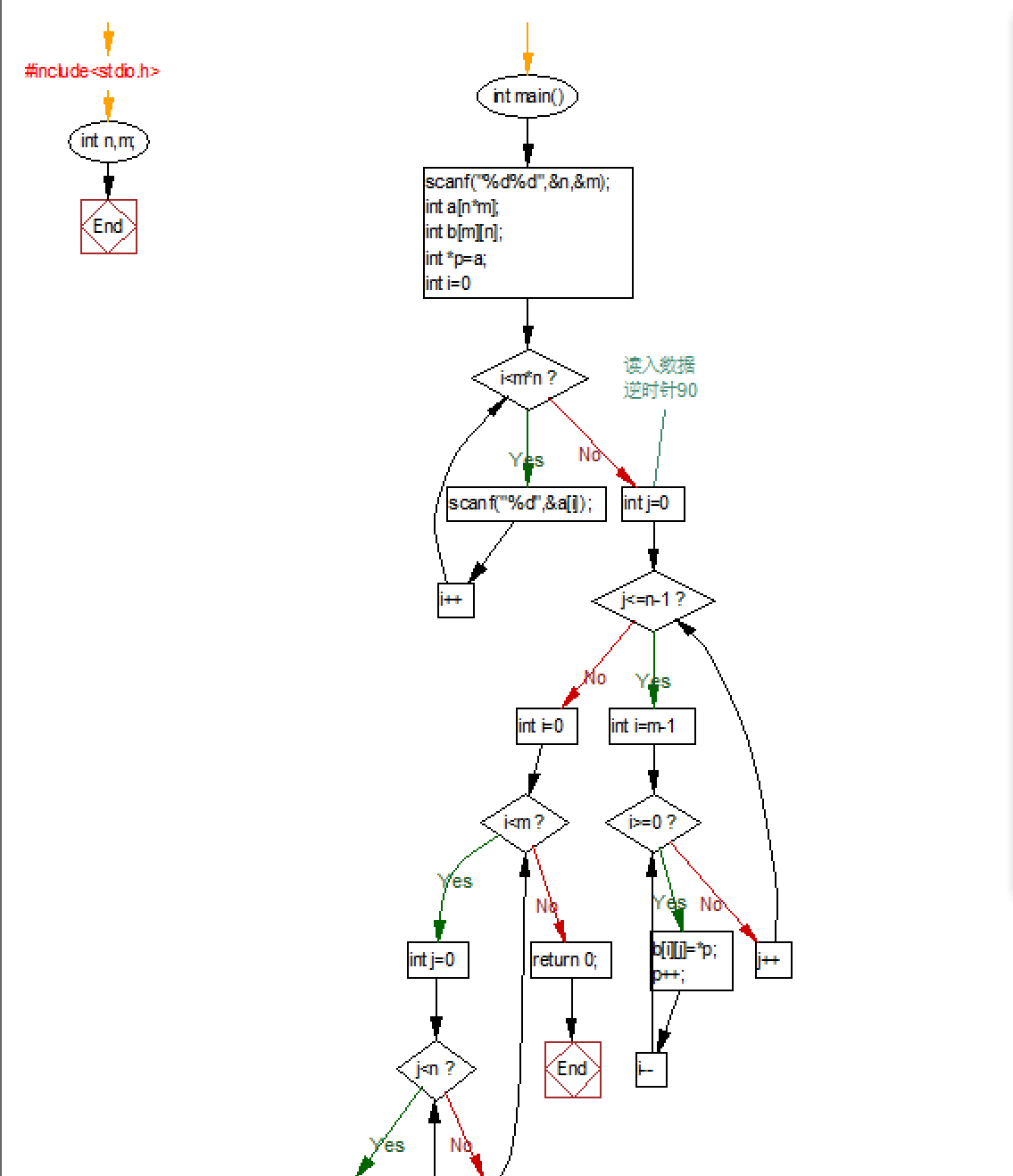
}

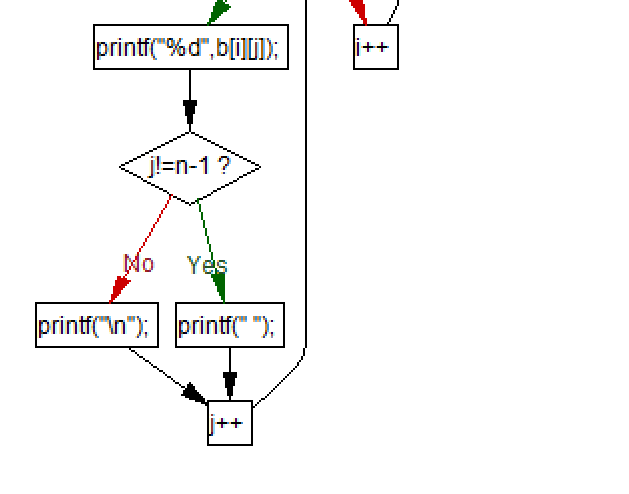
}

}程序运行截图：



程序流程图：





实验6-3:

输入n行文本，每行不超过80个字符，用字符指针数组指向键盘输入的n行文本（n不作为输入，可理解为循环输入多行，以CTRL+Z结束），删除每一行中的前置空格（' '）和水平制表符（'\t'）。

要求：将删除一行文本中前置空格和水平制表符的功能定义成函数，在main函数中输出删除前置空格符的各行。（并在最后输出换行符。）

程序源代码：

#include <stdio.h>

#include "string.h"

#include "stdlib.h"

#define MAXLINES 2

#define MAXLEN 80

int readlines(char \*pline[],int maxlines)

{

int nlines=0;

char temp[MAXLEN];

for(int i=0;i<maxlines&&gets(temp);i++)

{

pline[i]=(char \*)malloc(strlen(temp)+2);

if(pline[i]==NULL) return -1;

nlines++;

strcpy(pline[i],temp);

pline[i][strlen(temp)]='\n';

pline[i][strlen(temp)+1]='\0';

}

return (nlines);

}

void delete(char \*s[],int n)

{

int i,j,k,flag,sign;

char temp[81];

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

{

strcpy(temp,s[i]);

flag=1;

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

{

sign=0;

if(flag==1&&temp[j]==' ')

{

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

{

temp[k]=temp[k+1];

}

sign=1;

}

if(temp[j]!=' ') flag=0;

if(temp[j]=='\t')

{

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

{

temp[k]=temp[k+1];

}

sign=1;

}

if(sign==1) j--;

}

strcpy(s[i],temp);

}

}

int main()

{

int nlines;

char \*s[MAXLINES];

nlines=readlines(s,MAXLINES);

delete(s,nlines);

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

{

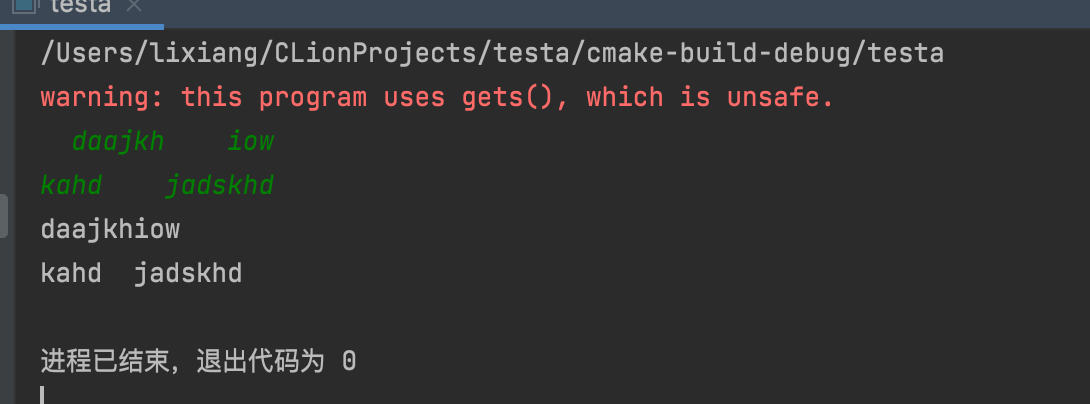
printf("%s",s[i]);

}

return 0;

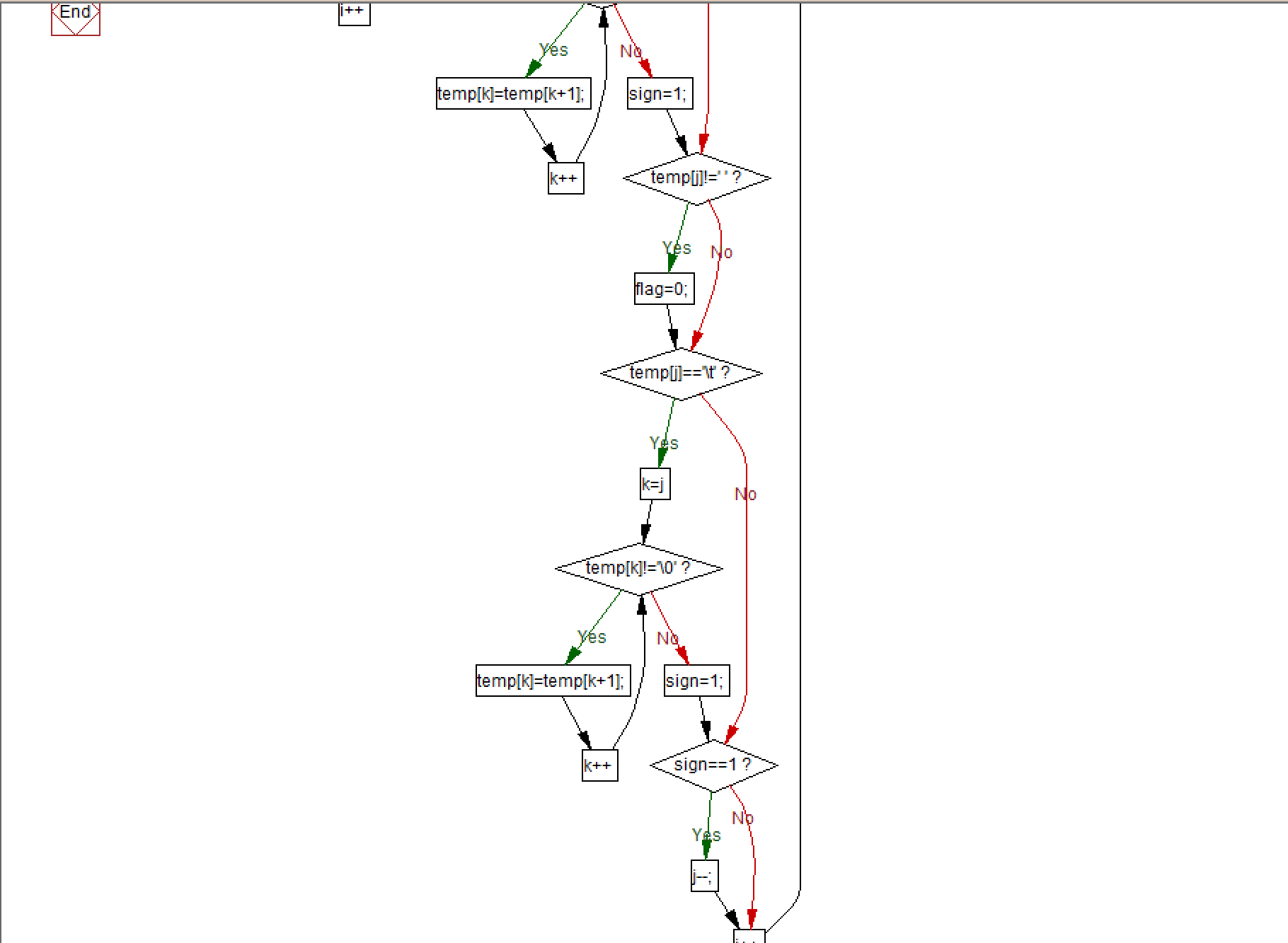
}

程序运行截图：



程序流程图：

CD4D71869E3268E91D2055D93D9652EB



实验6-4:

编写8个任务函数，一个scheduler调度函数和一个execute执行函数。仅在main函数中调用scheduler函数，scheduler函数要求用最快的方式调度执行用户指定的任务函数。

①先设计task0, task1, task2, task3, task4, task5, task6, task7共8个任务函数，每个任务函数的任务就是输出该任务被调用的字符串。例如，第0个任务函数输出“task0 is called!”，第1个任务函数输出“task1 is called!”，以此类推。

②scheduler函数根据键盘输入的数字字符的先后顺序，一次调度选择对应的任务函数。例如，输入：1350并回车，则scheduler函数一次调度选择task1, task3, taks5, task0，然后以函数指针数组和任务个数为参数将调度选择结果传递给execute函数并调用execute函数。

③execute函数根据scheduler函数传递的指针数组和任务个数为参数，按照指定的先后顺序依此调用执行选定的任务函数。

当输入13607122并回车，程序运行结果应当如下：

task1 is called!

task3 is called!

task6 is called!

task0 is called!

task7 is called!

task1 is called!

task2 is called!

task2 is called!

程序源代码：

#include <stdio.h>

void task0()

{

printf("task0 is called!\n");

}

void task1()

{

printf("task1 is called!\n");

}

void task2()

{

printf("task2 is called!\n");

}

void task3()

{

printf("task3 is called!\n");

}

void task4()

{

printf("task4 is called!\n");

}

void task5()

{

printf("task5 is called!\n");

}

void task6()

{

printf("task6 is called!\n");

}

void task7()

{

printf("task7 is called!\n");

}

void execute(int a[],int n)

{

void (\*t[8])()={task0,task1,task2,task3,task4,task5,task6,task7};

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

{

(\*t[a[i]])();

}

}

void scheduler()

{

int i=0,a[8];

char c;

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

{

a[i]=c-'0';

i++;

}

execute(a,i);

}

int main()

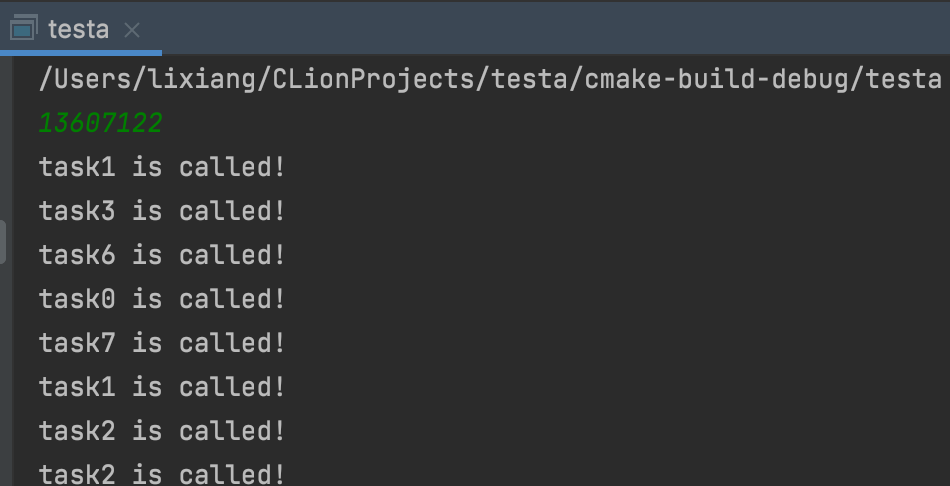
{

scheduler();

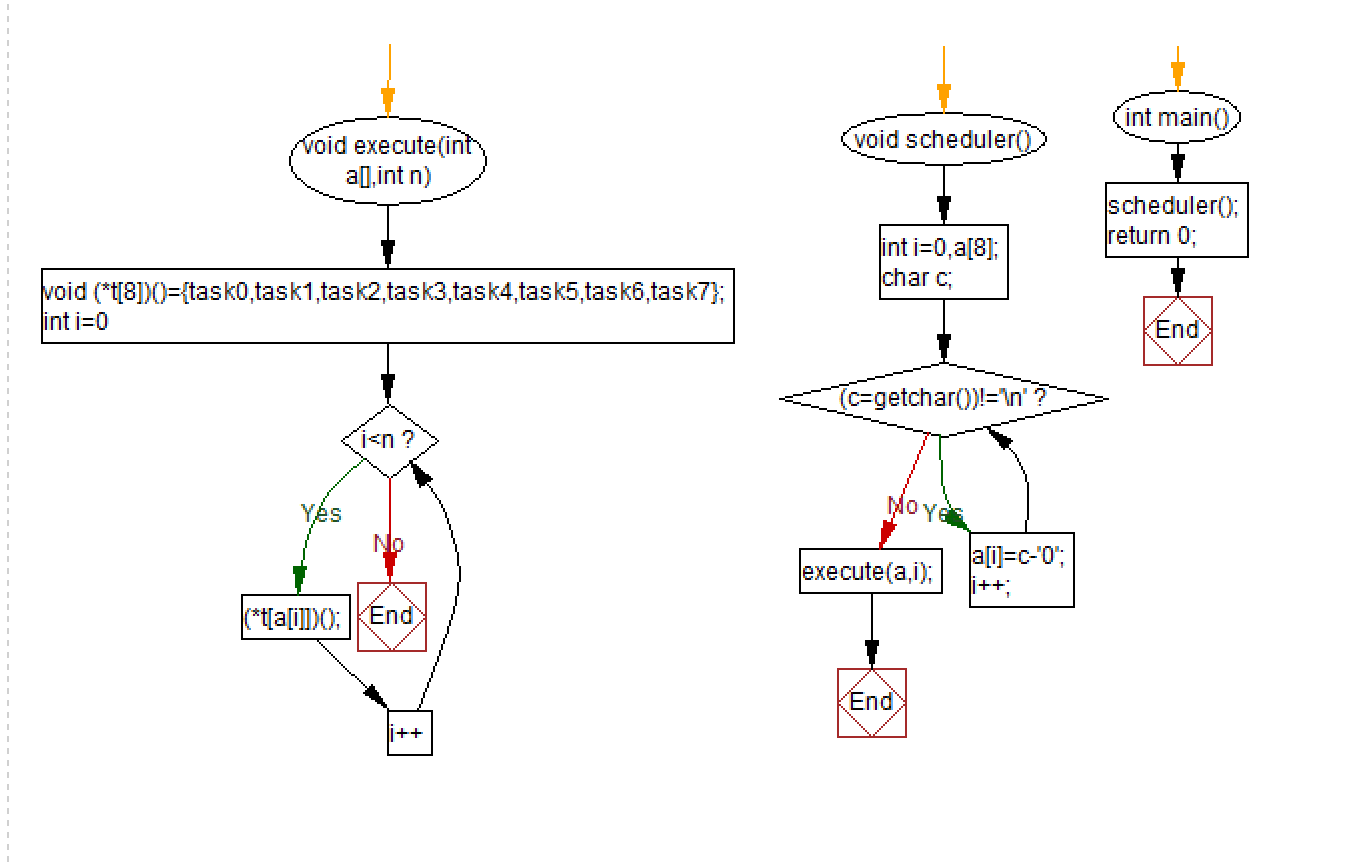
return 0;

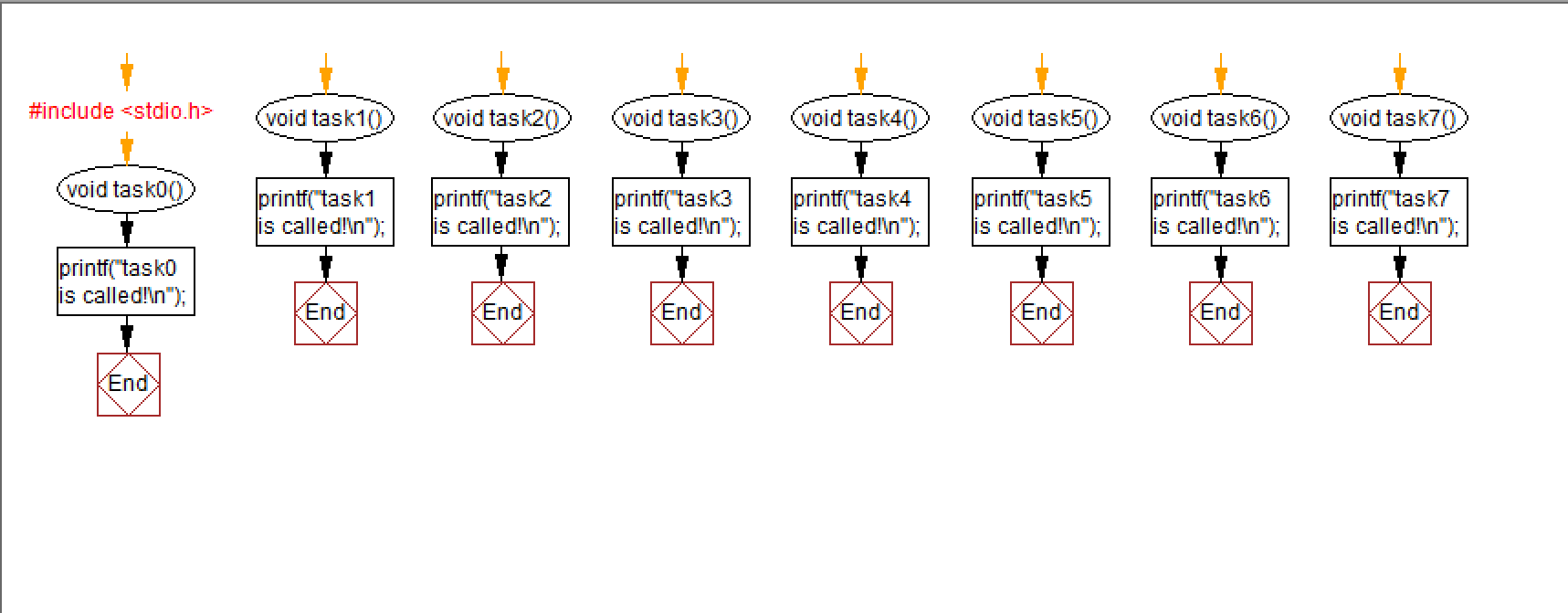
}

程序运行截图：



程序流程图：





**6.3实验小结**

指针是c语言的灵魂，提供了一种间访成员的方法，比数组的运行效率更高。

动态内存分配malloc函数要及时的申请然后及时的释放内存，以达到无冗余的目的。

区分清楚函数指针和指针函数，指针数组和数组指针的区别。

逆时针顺时针旋转一个数组的算法思想要体会好，首先是构造一个一维数组，然后进行旋转赋值。

对于堆和栈有了初步的了解，动态内存分配申请的是堆里的空间，而栈是程序运行开始生成，结束的时候消失，对于函数的调用有着重要的作用，常量 区里的东西不可改变，数组存于堆区所以可以改变。