**C语言 第六次作业**

**刘泓尊 自81 2018011446**

1. **数组元素循环移位**

#include<stdio.h>

int main()

{

void move(int\*p,int x,int y);

int m,n,a[100];

int \*pointer;

pointer=a;

printf("please input the value of n:");

scanf\_s("%d",&n);

printf("please inpuit the elements of a[]:\n");

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

scanf\_s("%d",pointer+i);

while(1)

{

printf("please input the value of m:");

scanf\_s("%d",&m);

if(m>=0&&m<=n)break;//判断输入是否合理

else printf("Error!please input m again!\n");

}

move(pointer,n,m);

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

{

printf("%d ",\*(pointer+i));

}

putchar(10);

return 0;

}

void move(int\*p,int x,int y)

{

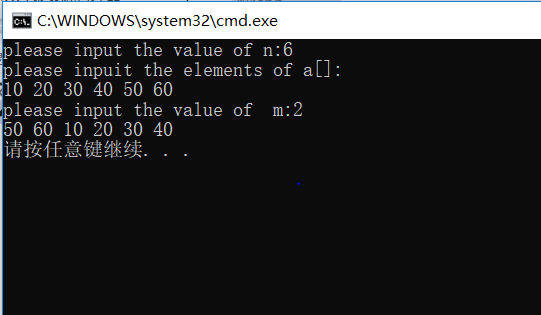
for(int i=x;i>0;i--)

\*(p+i+y)=\*(p+i);//\*(p+i+x)=a[i+x]

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

\*(p+i-x)=\*(p+i);

}



1. **矩阵与排序**

#include<stdio.h>

#include<malloc.h>

void sort1(int \*a[],int m,int n)//指针数组

{

int\*\*p;

p=a;

int i,j,k,min;

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

{

for(j=0;j<n;j++)//选择排序

{

min=j;

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

if(\*(\*(p+i)+min)>\*(\*(p+i)+k))min=k;

if(min!=j)

{

int tmp=\*(\*(p+i)+min);//指向数组元素的指针

\*(\*(p+i)+min)=\*(\*(p+i)+j);

\*(\*(p+i)+j)=tmp;

}

}

}

}

void sort2(int \*a[],int m,int n)

{

int\*\*p;

p=a;

int i,j,k,min;

sort1(a,m,n);//此时每行最后一个元素最大

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

{

min=i;

for(k=i+1;k<m;k++)

if(\*(\*(p+min)+n-1)>\*(\*(p+k)+n-1))min=k;

if(min!=i)

{

for(j=0;j<n;j++)//交换行

{

int tmp=\*(\*(p+min)+j);

\*(\*(p+min)+j)=\*(\*(p+i)+j);

\*(\*(p+i)+j)=tmp;

}

}

}

}

void sort3(int \*a[],int m,int n)

{

int\*p,\*min,\*k,t=0,i=0,count1=0;

//选择排序

for(p=\*a;count1<=m\*n;p++)

//为了使用指向一维数组的指针，下面的过程比较麻烦，请问助教有没有更好一点的方法可以使用指向一维数组的指针呢？

{

if(p==\*(a+i)+n&&i<m-1)//换行

{

i++;

p=\*(a+i);

}

if(p==(\*(a+m-1))+n)break;

min=p;

count1++;

for(k=p+1,t=i;k<=\*(a+t)+n&&t<m;k++)

{

if(k==(\*(a+t))+n&&t<m-1)//换行

{

t++;

k=\*(a+t);

}

if(k==(\*(a+m-1))+n)break;

if((\*min)>(\*k))min=k;

}

if(min!=p)

{

int tmp=(\*min);

(\*min)=(\*p);

(\*p)=tmp;

}

}

}

int main()

{

int m,n,\*\*a;

printf("please input m n:");

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

a=(int\*\*)malloc(m\*sizeof(int\*));//动态分配内存

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

\*(a+i)=(int\*)malloc(n\*sizeof(int));

printf("please input the elements of the matrix:\n");

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

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

scanf\_s("%d",(\*(a+i)+j));

sort1(a,m,n);

printf("\nThe result of sort1:\n");

for(int i=0;i<m;i++,putchar(10))

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

printf("%4d",\*(\*(a+i)+j));

sort2(a,m,n);

printf("\nThe result of sort2:\n");

for(int i=0;i<m;i++,putchar(10))

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

printf("%4d",\*(\*(a+i)+j));

sort3(a,m,n);

printf("\nThe result of sort3:\n");

for(int i=0;i<m;i++,putchar(10))

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

printf("%4d",\*(\*(a+i)+j));

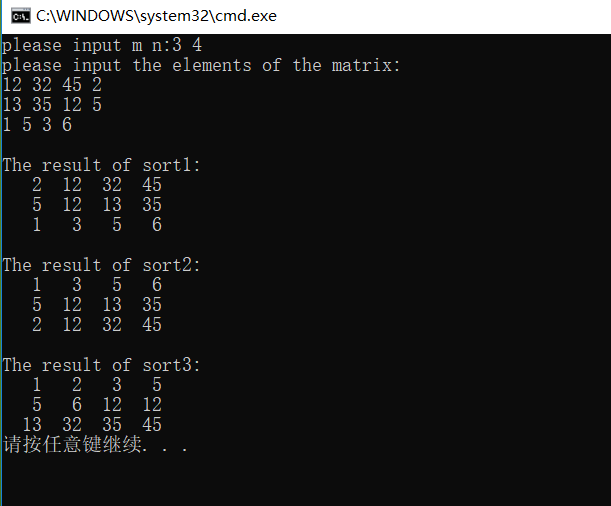
for(int i=0;i<m;i++)

free(\*(a+i));

free(a);

return 0;

}



1. **字符串压缩**

#include<stdio.h>

#include<string.h>

char\*StringCompression(char\*p,int n)

{

int i,j,k=0;

int count;

static char b[200];

char\*pointer;

pointer=b;

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

{

if(\*(p+i)==' ')

\*(pointer+k++)=\*(p+i);

else{

for(count=0,j=0;j<i;j++)

{

if(\*(p+j)==\*(p+i))count++;

}

if(count==0||count==2||count==5)

\*(pointer+k++)=\*(p+i);

//把字符串放入一个新的数组，使得次数的检查得以正常进行

}

}

\*(pointer+k)='\0';

return (pointer);

}

int main()

{

char s[200];

char\*pointer,\*newstr;

int len;

pointer=s;

printf("please input a string:\n");

gets\_s(s);

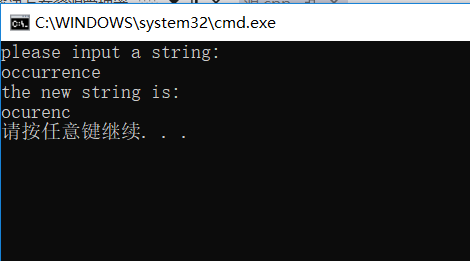
len=strlen(s);

newstr=StringCompression(pointer,len);

printf("the new string is:\n%s\n",newstr);

return 0;

}



1. **字符串库函数的实现**

#include<stdio.h>

#include<string.h>

char\*strncpy(char\*s,char\*t,int n)

{

int i,len1,len2;

len1=strlen(s);

len2=strlen(t);

if(len2<n)n=len2;//对于n超过t的长度的处理

for(i=0;i<n&&\*(t+i)!='\0';i++)

\*(s+i)=\*(t+i);

if(len1<n)\*(s+n)='\0';

return(s);

}

char\*strncat(char\*s,char\*t,int n)

{

int i,len1,len2;

len1=strlen(s);

len2=strlen(t);

if(len2<n)n=len2;//对于n超过t的长度的处理

for(i=0;i<n&&\*(t+i)!='\0';i++)

\*(s+len1+i)=\*(t+i);

\*(s+len1+n)='\0';

return(s);

}

int strncmp(char\*s,char\*t,int n)

{

int m,i=1;

while(\*s==\*t)

{

if(i==n||\*s=='\0'||\*t=='\0')return 0;

s++;

t++;

i++;

}

m=\*s-\*t;

if(m>0)m=1;

else m=-1;

return m;

}

void input(char\*s,char\*t,int\*n)

{

printf("please input str1:\n");

gets\_s(s,100);

int len=strlen(s);

printf("please iput str2:\n");

gets\_s(t,100);

printf("please input n:");

scanf\_s("%d",n);

}

int main()

{

char a[100],b[100];

char\*p1,\*p2;

int cmp,n,\*p,len;

p1=a;p2=b;p=&n;

input(p1,p2,p);

len=strlen(p1);

cmp=strncmp(p1,p2,\*p);

printf("\nstrncmp函数：%d\n",cmp);

strncat(p1,p2,\*p);

printf("strncat函数：\n%s\n",p1);

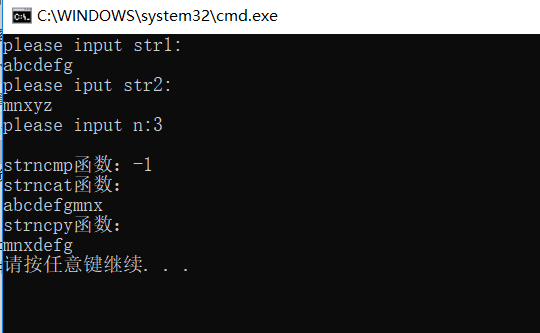
\*(p1+len)='\0';//重置字符串a为初始输入值

strncpy(p1,p2,\*p);

printf("strncpy函数：\n%s\n",p1);

return 0;

}



1. **星期的查找**

#include<stdio.h>

int main()

{

void weekday(char \*p[],int x);

int m;

char \*a[7]={"一","二","三","四","五","六","日"};//指针数组

while(1){

printf("please input a weekday number：");

scanf\_s("%d",&m);

if(m>=1&&m<=7)//判断条件是否合理

{

weekday(a,m);

break;

}

else printf("Error!Please input a number between 1 and 7 !!!\n");

}

putchar(10);

return 0;

}

void weekday(char \*p[],int x)

{

switch(x)

{

case 1:printf("星期%s",\*p);break;

case 2:printf("星期%s",\*(p+1));break;

case 3:printf("星期%s",\*(p+2));break;

case 4:printf("星期%s",\*(p+3));break;

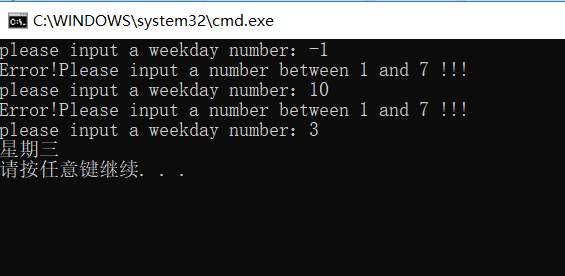
case 5:printf("星期%s",\*(p+4));break;

case 6:printf("星期%s",\*(p+5));break;

case 7:printf("星期%s",\*(p+6));break;

}

}



1. **字符串排序**

#include<stdio.h>

#include<string.h>

int main()

{

void sort(char\*s[]);

char\*s[8];//指针数组

char a[8][21];

int i;

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

{

printf("please input str%d:\n",i+1);

gets\_s(a[i]);

s[i]=a[i];

}

sort(s);

printf("\n排序结果:\n");

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

printf("%s\n",\*(s+i));

return 0;

}

void sort(char\*s[])

{

char\*\*p;//二级指针

int i,j;

for(i=7;i>=0;i--)//按字典序排列

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

{

p=s+j;

if(strcmp(\*p,\*(p+1))>0)

{

char\*tmp;

tmp=\*p;

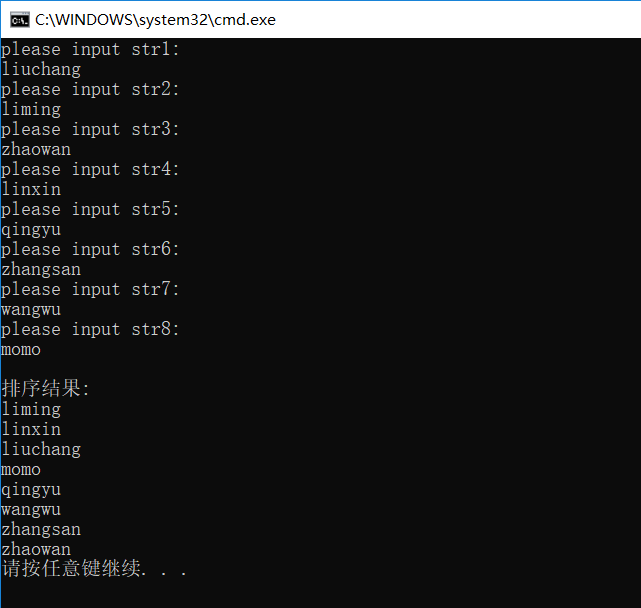
\*p=\*(p+1);

\*(p+1)=tmp;

}

}

}



1. **字符查找**

#include<stdio.h>

#include<string.h>

char\*\*search(char\*p[],char\*s,char \*\*str,int len,int\*num)

{

char\*\*result,\*re[8];//定义二级指针和指针数组

result=re;

int i,j,flag,count=0;

for(i=0;i<8;i++)\*(re+i)=\*(str+i);//指针初始化

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

{

flag=0;

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

if(p[i][j]!=\*(s+j))

{

flag=-1;

break;

}

if(flag==0)

{

strcpy\_s(\*(result+count),20,\*(p+i));

count++;

}

}

\*num=count;

if(count>0)return result;

else return NULL;//返回空指针

}

int main()

{

int i,count=0,\*num;

num=&count;

char\*p[8]={"FORTRAN","PASCAL","Python","BASIC","C","C++","COBOL","SMALLTALK"};//指针数组

char s[10],\*pointer,str[8][10],\*str\_pointer[8];

for(i=0;i<8;i++)\*(str\_pointer+i)=\*(str+i);//指针数组初始化

pointer=s;

gets\_s(s);

int len=strlen(s);

if((search(p,pointer,str\_pointer,len,num))!=NULL)

{

printf("possible language:\n");

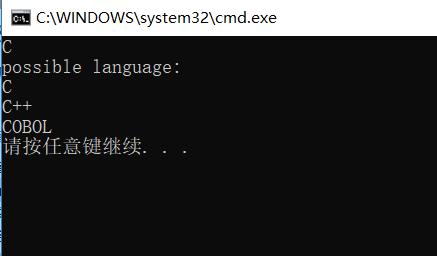
for(i=0;i<\*num;i++)printf("%s\n",\*(search(p,pointer,str\_pointer,len,num)+i));

}

else printf("NULL\n");

return 0;

}



1. **算数练习程序**

#include<stdio.h>

#include<time.h>

#include<stdlib.h>

#include<conio.h>

int main()

{

char expression[10];

char\*GetRanddomExpression(char expr[],int\*result);

int re,\*p,result;

p=&re;

char flag;

printf("100以内的加减乘除运算,按任意键开始和继续,按q/Q结束练习\n");

while(1)

{

fflush(stdin);

expression[0]='\0';

flag=\_getch();

if(flag!='q'&&flag!='Q')

{

char\*exp=GetRanddomExpression(expression,p);

printf(" %s",exp);

scanf\_s("%d",&result);

if(result==\*p)printf(" 计算正确！非常棒！O(∩\_∩)O\n");

else printf(" 计算错误！加油啊!o(≧v≦)o \n");

}

else break;

}

return 0;

}

char\*GetRanddomExpression(char expr[],int\*result)

{

int n,a,b,flag=0;

char\*s;

char op;

s=expr;

int\*pointer\_a,\*pointer\_b;

pointer\_a=&a;pointer\_b=&b;

srand((unsigned int)time(NULL));

\*pointer\_a=rand()%100+1;

\*pointer\_b=rand()%100+1;//100以内正整数

n=rand()%4;

switch(n)

{

case 0:

while(\*pointer\_a+\*pointer\_b>=100)

\*pointer\_b=rand()%100+1;

op='+';

\*result=\*pointer\_a+\*pointer\_b;

break;

case 1:

while(\*pointer\_a-\*pointer\_b<=0)

\*pointer\_b=rand()%100+1;

op='-';

\*result=\*pointer\_a-\*pointer\_b;

break;

case 2:

while((\*pointer\_a)\*(\*pointer\_b)>=100)

\*pointer\_b=rand()%100+1;

op='\*';

\*result=(\*pointer\_a)\*(\*pointer\_b);

break;

default:

while((\*pointer\_a)%(\*pointer\_b)!=0)

\*pointer\_b=rand()%100+1;

op='/';

\*result=(\*pointer\_a)/(\*pointer\_b);

break;

}

if((\*pointer\_a)/10>0)

{

\*s=(\*pointer\_a)/10+'0';

\*(s+1)=(\*pointer\_a)%10+'0';

flag+=2;

}

else {\*s=(\*pointer\_a)%10+'0';flag+=1;}

\*(s+flag)=op;

flag++;

if((\*pointer\_b)/10>0)

{

\*(s+flag)=(\*pointer\_b)/10+'0';

\*(s+flag+1)=(\*pointer\_b)%10+'0';

flag+=2;

}

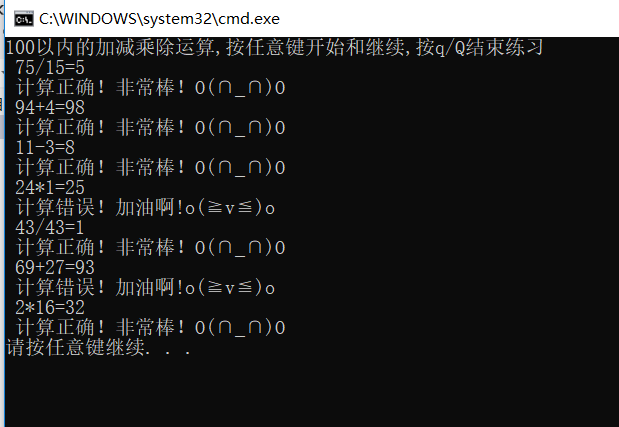
else {\*(s+flag)=(\*pointer\_b)%10+'0';flag++;}

\*(s+flag)='=';

\*(s+flag+1)='\0';

return(s);

}



9.**数组静态分区**

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

void CountEvensAndOdds(int \*a,int len,int\*even,int\*odd)

{

int i;

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

{

if((\*(a+i))%2)(\*even)++;

else (\*odd)++;

}

}

void Partition(int \*a,int len,int\*evens,int\*len\_even,int\*odds,int\*len\_odd)

{

int i,j=0,k=0;

CountEvensAndOdds(a,len,len\_even,len\_odd);

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

{

if((\*(a+i))%2)

{

\*(evens+j++)=\*(a+i);

}

else

{

\*(odds+k++)=\*(a+i);

}

}

}

int main()

{

int a[100],evens[100],odds[100],n,len\_even=0,len\_odd=0;

int\*p1,\*p2;

p1=&len\_even;

p2=&len\_odd;

while(1)

{

printf("please input n:");

scanf\_s("%d",&n);

if(n>=0&&n<100)

{

srand((unsigned)time(NULL));

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

a[i]=rand()%100+1;

Partition(a,n,evens,p1,odds,p2);

printf("\n原始数组：len=%d\n",n);

for(int i=0;i<\*p1;i++)

printf("%3d",a[i]);

printf("\n奇数数组：len=%d\n",\*p1);

for(int i=0;i<\*p1;i++)

printf("%3d",evens[i]);

printf("\n偶数数组：len=%d\n",\*p2);

for(int i=0;i<\*p2;i++)

printf("%3d",odds[i]);

putchar(10);

break;

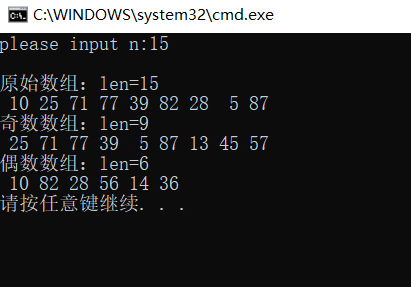
}

else printf("Error!please input 0<=n<100!\n");

}

return 0;

}



**10.数组动态分区**

#include<stdio.h>

#include<time.h>

#include<malloc.h>

#include<stdlib.h>

void CountEvensAndOdds(int a[],int n,int\*counteven,int\*countodd)

{

int i;

\*counteven=0;

\*countodd=0;

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

if((\*(a+i))%2)(\*counteven)++;

else (\*countodd)++;

}

void Partition(int a[],int n,int\*\*arr1,int\*counteven,int\*\*arr2,int\*countodd)

{

CountEvensAndOdds(a,n,counteven,countodd);

\*arr1=(int\*)malloc((\*counteven)\*sizeof(int));

\*arr2=(int\*)malloc((\*countodd)\*sizeof(int));

int i,j,k;

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

{

if((\*(a+i)%2))\*((\*arr1)+k++)=\*(a+i);

else \*((\*arr2)+j++)=\*(a+i);

}

}

int main()

{

int \*a,n,even,odd;

int \*b,\*c;//定义两个指针\*b,\*c

int \*p1,\*p2;

p1=&even;

p2=&odd;

srand((unsigned int)time(NULL));

n=rand()%50+1;//随机生成数组长度（1到50）

while((a=(int\*)malloc(n\*sizeof(int)))==NULL);

printf("随机数组:len=%d\n",n);

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

{

\*(a+i)=rand()%100;

printf("%3d",\*(a+i));

}

CountEvensAndOdds(a,n,p1,p2);

Partition(a,n,&b,p1,&c,p2);//传b,c的地址

printf("\n奇数数组:len=%d\n",\*p1);

for(int i=0;i<(\*p1);i++)

printf("%3d",\*(b+i));

printf("\n偶数数组:len=%d\n",\*p2);

for(int i=0;i<(\*p2);i++)

printf("%3d",\*(c+i));

putchar(10);

free(a);

free(b);

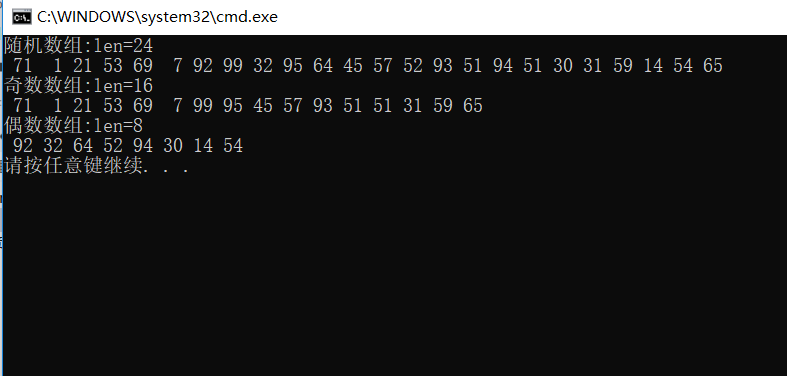
free(c);

b=NULL;

c=NULL;

return 0;

}



**11.最长公共单词**

//输出最长的多个单词

#include<stdio.h>

#include<string.h>

#include<malloc.h>

int judge(char a)

{

if(a>='a'&&a<'z'||a>'A'&&a<'Z'||a=='-')return 1;

return 0;

}

void search(char\*\*str,char word[][20],int\*num)

{

int max=0,i,j,k=0,flag=0,t;

char tmp[20];

for(i=0;\*(\*(str)+i)!='\0';i++)

{

tmp[0]='\0';t=0;

if(judge(\*(\*(str)+i)))

{

t=1;

for(j=i;judge(\*(\*(str)+j))&&(\*(\*(str)+j))!='\0';j++)

{

\*(tmp+k++)=\*(\*(str)+j);

}

\*(tmp+k)='\0';

i+=k-1;

}

for(j=1;t==1&&j<2;j++)

{

if(strstr(\*(str+j),tmp)!=NULL)

flag=1;

else flag=0;

}

if(flag==1&&max<k)

{

max=k;

for(t=0;t<=(\*num)+1;t++)\*(\*(word+t))='\0';

(\*num)=0;

strcpy\_s(\*(word+(\*num)),20,tmp);

}

else if(flag==1&&max==k)

{

(\*num)++;

strcpy\_s(\*(word+(\*num)),20,tmp);

}

flag=0;

k=0;

}

}

int main()

{

char \*\*str;

char maxword[20][20];

int i,num=-1;

int\*p;

p=&num;

printf("please input strings:（字符串长度不超过50）\n");

str=(char\*\*)malloc(2\*sizeof(char\*));//动态分配

fflush(stdin);

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

str[i]=(char\*)malloc(52\*sizeof(char));

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

gets\_s(\*(str+i),50);

search(str,maxword,p);

if(num==-1)printf("无公共单词!\n");

else{

printf("最长公共单词有%d个\n",num+1);

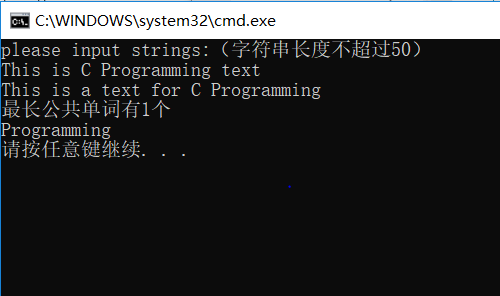
for(i=0;i<=num;i++)

printf("%s\n",\*(maxword+i));

}

return 0;

}



**12.最长公共子串**

#include<stdio.h>

#include<string.h>

#include<malloc.h>

void LCS(char\*str1,char\*str2,char\*str3,char\*LCSstr)

{

int len1,len2,len3,i,j,k,flagx=0,flagy=0,flagz=0,max=0;

len1=strlen(str1);

len2=strlen(str2);

len3=strlen(str3);

int\*\*\*flag;//定义一个三维对应矩阵，存放三个数组的匹配情况。

//动态分配内存

while((flag=(int\*\*\*)malloc((len1+1)\*sizeof(int\*\*)))==NULL);

for(i=0;i<len1+1;i++)

{

while(((\*(flag+i))=(int\*\*)malloc((len2+1)\*sizeof(int\*)))==NULL);

for(j=0;j<len2+1;j++)

while(((\*(\*(flag+i)+j))=(int\*)malloc((len3+1)\*sizeof(int)))==NULL);

}

for(i=0;i<=len1;i++)

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

for(k=0;k<=len3;k++)

\*(\*(\*(flag+i)+j)+k)=0;

for(i=1;i<=len1;i++)

for(j=1;j<=len2;j++)

for(k=1;k<=len3;k++)

{

if(\*(str1+i-1)==\*(str2+j-1)&&\*(str2+j-1)==\*(str3+k-1))

(\*(\*(\*(flag+i)+j)+k))=(\*(\*(\*(flag+i-1)+j-1)+k-1))+1;

if((\*(\*(\*(flag+i)+j)+k))>max)

{

max=(\*(\*(\*(flag+i)+j)+k));

flagx=i-1;

flagy=j-1;

flagz=k-1;

}

}

\*(LCSstr+max)='\0';

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

if(max>0)

{

max--;

\*(LCSstr+max)=\*(str1+i);

}

else break;

for(i=0;i<len1+1;i++)

{

for(j=0;j<len2+1;j++)

free(\*(\*(flag+i)+j));

free(\*(flag+i));

}

free(flag);

}

int main()

{

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

printf("please input string1:\n");

gets\_s(a);

printf("please input string2:\n");

gets\_s(b);

printf("please input string3:\n");

gets\_s(c);

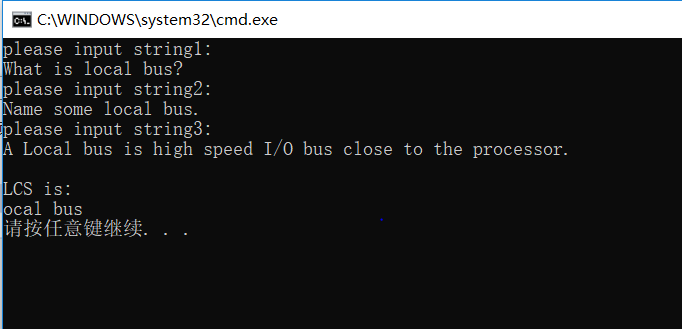
LCS(a,b,c,d);

printf("\nLCS is:\n");

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

return 0;

}



**13.动态三维数组**

#include<stdio.h>

#include<malloc.h>

int\*\*\*CreateGrid(int m,int n,int t)

{

int \*\*\*arr;

int i,j;

if((arr=(int\*\*\*)malloc(m\*sizeof(int\*\*)))==NULL)return NULL;

else{

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

if((\*(arr+i)=(int\*\*)malloc(n\*sizeof(int\*)))==NULL)return NULL;

else{

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

if((\*(\*(arr+i)+j)=(int\*)malloc(t\*sizeof(int)))==NULL)return NULL;

}

}

return arr;

}

void FreeGrid(int\*\*\*arr,int n,int m,int t)

{

int i,j;

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

{

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

{

free(\*(\*(arr+i)+j));

}

free(\*(arr+i));

}

free(arr);

arr=NULL;

}

int main()

{

int m,n,t,ele=1,i,j,k;

int\*p,\*\*\*a;

p=&ele;

printf("please input the value of m n t:");

scanf\_s("%d%d%d",&m,&n,&t);

a=CreateGrid(m,n,t);

if(a!=NULL)

{

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

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

for(k=0;k<t;k++)

{

\*(\*(\*(a+i)+j)+k)=\*p;//用整数指针赋值

(\*p)++;

}

printf("the array is:\n");

for(i=0;i<m;i++,printf("\n"))

for(j=0;j<n;j++,printf("\n"))

for(k=0;k<t;k++)

{

printf("%4d",(\*(a+i))[j][k]);//用指针数组输出每个平面

}

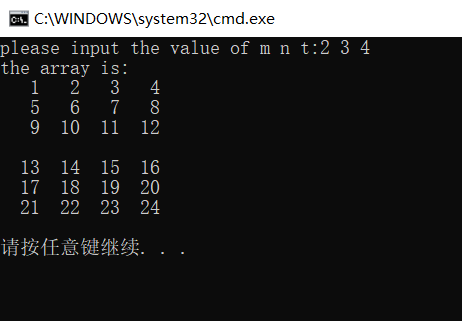
FreeGrid(a,m,n,t);

}

a=NULL;

return 0;

}



**14.程序运行分析**

**第一个程序：**

#include<stdio.h>

int main()

{

int a[4]={1,2,3,4};

int \*ptr1=(int\*)(&a+1);

int \*ptr2=(int\*)((int)a+1);

printf("%x\n%x\n",ptr1[-1],\*ptr2);//输出结果：4 2000000

return 0;

}

**//&a取数组a的地址，之后+1表示+1\*sizeof(a),即sizeof(4\*int).ptr1指向数组a后面的相邻地址**

**//ptr[-1]等价于\*(ptr-1)，减一得到了a[3]的地址，之后取内容，得到4，输出十六进制的4**

**//a是一个地址00000000,(int)a表示把这个地址值转化为int型，+1得到0000 0001**

**//之后强制转换为(int\*)类型，故ptr2指向地址0000 0001，该地址存放数据0000 0002 0000 0000。**

**//之后取内容即0000 0002 0000 0000，输出十六进制：2000000**

**第二个程序：**

#include<stdio.h>

#include<windows.h>

int main()

{

char\*c[]={"ENTER","NEW","POINT","FIRST"};

char\*\*cp[]={c+3,c+2,c+1,c};

char\*\*\*cpp=cp;

printf("%s\n",\*\*++cpp);//POINT

printf("%s\n",\*--\*++cpp+3);//ER

printf("%s\n",\*cpp[-2]+3);//ST

printf("%s\n",cpp[-1][-1]+1);//EW

system("pause");

return 0;

}

**//\*c[]为指针数组，每个元素存放对应字符串的首地址**

**//\*c[]:c -> &'E',c+1 -> &'N',c+2 -> &'P',c+3 -> &'F'**

**//\*\*cp[]:二级指针数组，每个元素存放数组c的每个空间地址**

**//\*\*cp[]:c+3,c+2,c+1,c**

**//\*\*\*cpp:三级指针，指向数组cp的首地址**

**//\*\*++cpp等价于\*(\*(++cpp)),cpp先自增1，指向cp[1],之后取内容，变为c+2。之后再取内容，取到"POINT"的首地址**

**//因此输出"POINT"**

**//\*--\*++cpp+3等价于\*(--(\*(++cpp)))+3.**

**//即cpp自增1，后取内容取到c+1。之后c+1自减1，得到c。之后再取内容，得到"ENTER"的首地址。之后+3，得到"E"的地址。**

**//所以输出"ER"**

**//\*cpp[-2]+3等价于\*(cpp[-2])+3**

**//cpp[-2]等价于\*(cpp-2),取到c+3.之后取内容，得到"FIRST"的首地址。之后+3，得到'S'的地址。**

**//输出"ST"**

**//cpp[-1][-1]+1等价于\*(\*(cpp-1)-1)+1**

**//cpp-1再取内容，得到c+2。之后-1再取内容，得到"NEW"的首地址，再+1，得到'E'的地址**

**//输出"EW"**

**15.网络警察**

#include<stdio.h>

#include<string.h>

#include<math.h>

int Check(int key\_length,char\*keyword,int string\_length,char\*string)//检查keyword和string是否匹配

{

int k,letter1[26]={0},letter2[26]={0},flag=0;

for(k=0;k<key\_length;k++)

\*(letter1+(\*(keyword+k)-97))+=1;

for(k=0;k<string\_length;k++)

\*(letter2+(\*(string+k)-97))+=1;//对出现字母的次数进行标记

for(k=0;k<26;k++)

if(\*(letter1+k)==\*(letter2+k)||(\*(letter1+k)!=0&&\*(letter2+k)!=0&&(\*(letter2+k)-(\*(letter1+k)))<=1))flag=1;

//相似度度量标准：当单词中的字母出现的重复次数<=1时，认为两个单词相同

else {flag=0;break;}

return flag;

}

void StringCut(char (\*keyword)[50],char\*temp,int\*Size)//将一个句子中拆分成单词存入二维数组

{

char\*p;

int i=0,j;

p=temp;

while(\*p!='\0')

{

j=0;

while(\*p!=' '&&\*p!='\0')

\*(\*(keyword+i)+j++)=\*(p++);

\*(\*(keyword+i++)+j)='\0';

while(\*p==' ')p++;

}

\*Size=i;//标记单词个数

}

int main()

{

char keyword[100][50],string[100][50],temp[5000];

int key\_length,string\_length;

int i,j,flag=0;

char (\*p\_keyword)[50],(\*p\_string)[50];

p\_keyword=keyword;

p\_string=string;

printf("请输入关键词:\n");

gets\_s(temp);

StringCut(p\_keyword,temp,&key\_length);

printf("请输入待检测句子:\n");

gets\_s(temp);

StringCut(p\_string,temp,&string\_length);

printf("\n敏感单词:\n");

for(i=0;i<key\_length;i++)

for(j=0;j<string\_length;j++)

if(Check(strlen(\*(keyword+i)),\*(keyword+i),strlen(\*(string+j)),\*(string+j)))

{

printf("%s ",\*(keyword+i));

flag=1;

}

if(flag==0)printf(" 无!通过测试!");

printf("\n");

return 0;

}

