第一天 博弈机

第二天

#include <stdio.h>

//数组定义: int score[100];

//数组下标: 从0开始,score[0],score[1]...score[99];

//a[n], a[0].....a[n-1]

// 数据类型 标示符[常量表达式]

// 1. 数组中的每个元素的类型相同

// 2. 数组的长度必须为常量(长度指的是数组元素个数)

// 3. 数组名必须符合标示符定义

// 4. 数组下标从0开始

// 5. 数组名是一个地址常量,不是变量

// 6. 数组中的每个元素是变量

//数组在内存中占用的大小 = 数据类型长度 \* 数组元素个数

//数组访问不能越界操作,

//数组遍历

/\*int main(int argc, const char \* argv[])

{

int a[10];

int i;

for (i=0; i<10; i++) {

a[i]=i;

}

for (i=0; i<10; i++) {

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

}

return 0;

}\*/

//int main(int argc, const char \*argv[])

//{

// int n;

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

// int a[n];//不能这样定义,错误

//}

/\*int main(int argc, const char \*argv[])

{

int a[20];

printf("a==%ld\n", sizeof(a));

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int a[5];

for (int i=0; i<10; i++) {

a[i]=i;

}

for (int i=0; i<10; i++) {

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

}

return 0;

}\*/

// 倒序操作数组

/\*int main(int argc, const char \*argv[])

{

int a[5];

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

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

}

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

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

}

return 0;

}\*/

//数组初始化

//完全初始化

//

/\*int main(int argc, const char \*argv[])

{

int a[5]={1,2,3,4,5};

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

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

}

return 0;

}\*/

//不完全初始化

/\*int main(int argc, const char \*argv[])

{

int a[5]={3,6};

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

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

}

return 0;

}\*/

//初始化为0

/\*int main(int argc, const char \*argv[])

{

//int a[5]={};

int a[5]={0};

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

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

}

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int a[]={1,2,3,4,5,6};

printf("%ld\n",sizeof(a)/sizeof(int));

return 0;

}\*/

//求学生的平均成绩

/\*int main(int argc, const char \*argv[])

{

int total=0;

float average;

int a[10];

for (int i=0; i<10; i++) {

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

total+=a[i];

}

average = (float)(total/10);

printf("average == %.2f\n", average);

return 0;

}\*/

//斐波那契数列

//输出斐波那契数列的前20项

// 1 1 2 3 5 8 13 21 34 55 89 ....

// a[n]=a[n-1]+a[n-2]

/\*int main(int argc, const char \*argv[])

{

int a[20]={1,1};

for (int i=2; i<20; i++) {

a[i]=a[i-1]+a[i-2];

}

for (int i=0; i<20; i++) {

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

}

return 0;

}\*/

//输入10个数据, 查找该数据中的最大数,并将其输出到屏幕上

/\*int main(int argc, const char \*argv[])

{

int a[10];

int max;

for (int i=0; i<10; i++) {

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

}

max = a[0];

for (int i=1; i<10; i++) {

if (a[i]>max) {

max = a[i];

}

}

printf("The max value is : %d \n", max);

return 0;

}\*/

//查找某个数值

/\*int main(int argc, const char \*argv[])

{

int a[10]={12,34,56, 67, 78, 65, 43, 32,90,88};

int n;

int i=0;

scanf("%d", &n);

for (i=0; i<10; i++) {

if (n==a[i]) {

break;

}

}

if (i==10) {

printf("没有该数据\n");

}

else

{

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

}

return 0;

}\*/

/\*#define LEN 100

int main(int argc, const char \*argv[])

{

int a[LEN];

for (int i=0; i<LEN; i++) {

a[i]=i;

}

for (int i=0; i<LEN; i++) {

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

}

return 0;

}\*/

//字符数组

// char str[100];

// 每个元素占用1个字节的空间

//

/\*int main(int argc, const char \*argv[])

{

char str[100];

scanf("%s",str);

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

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

char str[5]={'a','b','c','d','e'};

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

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

}

putchar('\n');

return 0;

}\*/

//字符串初始化

/\*int main(int argc, const char \*argv[])

{

char str[100]="qianfengjiaoyu";

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

}\*/

/\*int main(int argc, const char \*argv[])

{

char str[12]="hello world!";

//str[12]='\0';

//printf("%s",str);

for (int i=0; i<12; i++) {

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

}

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int p[8]={11,12,13,14,15,16,17,18};

int i=0,j=0;

while(i<7)

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

if(p[i]%2)

j+=p[i];

i++;

}

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

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int p[7]={11,13,14,15,16,17,18},i=1,k=1;

while(i<7&&p[i]%2)

{

k=k+p[i];

i++;

}

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

}\*/

下午

#include <stdio.h>

//冒泡排序: 相邻两数比较,大数往后移动,小数往前移动;

//4 5 7 8 9

// 9 7 8 5 4

//第一次排序,交换4次

// 7 9 8 5 4

// 7 8 9 5 4

// 7 8 5 9 4

// 7 8 5 4 9

//第二次排序,交换3次

// 7 8 5 4

// 7 5 8 4

// 7 5 4 8 9

//第三次排序,交换2次

// 5 7 4 8 9

// 5 4 7 8 9

//第四次排序

// 4 5 7 8 9

#define LEN 10

/\*int main(int argc, const char \* argv[])

{

int a[LEN];

int i, j;

int temp;

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

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

}

for (i = 0; i<LEN-1; i++) {

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

if (a[j]>a[j+1]) {

temp = a[j];

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

a[j+1]=temp;

}

}

}

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

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

}

putchar('\n');

return 0;

}\*/

//选择排序

//9 7 8 5 4

//

//第一次排序

//4 7 8 5 9

//第二次排序

//4 5 8 7 9

//第三次排序

//4 5 7 8 9

//第四次排序

//4 5 7 8 9

/\*int main(int argc, const char \*argv[])

{

//int a[5]={9, 7, 8, 5, 4};

int a[LEN];

int i,j;

int k,temp;

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

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

}

for (i=0; i<LEN-1; i++) {

k=i;

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

if (a[k]>a[j]) {

k=j;

}

}

if (k!=i) {

temp = a[k];

a[k] = a[i];

a[i] = temp;

}

}

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

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

}

putchar('\n');

return 0;

}\*/

//插入法排序

//

/\*int main(int argc, const char \*argv[])

{

int i,j,temp;

//int a[5]={9, 7, 8, 5, 4};

int a[LEN];

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

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

}

for (i=0; i< LEN-1; i++) {

temp = a[i+1];

j = i;

while (j>=0&&a[j]>temp) {

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

j--;

}

a[j+1]=temp;

}

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

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

}

printf("\n");

return 0;

}\*/

//二维数组

// int a[3][3]

//

//二维数组初始化

//直接初始化

/\*int main(int argc, const char \*argv[])

{

int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};

int i, j;

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

for (j=0; j<4; j++) {

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

}

}

return 0;

}\*/

/\*int main(int argc,const char \*argv[])

{

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

{5,6,7,8},

{9,10,11,12}};

for (int i=0; i<3; i++) {

for (int j=0; j<4; j++) {

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

}

printf("\n");

}

printf("\n");

return 0;

}\*/

//1

//1 1

//1 2 1

//1 3 3 1

//1 4 6 4 1

//

//

#include "stdio.h"

#define N 20

int main(int argc,const char \*argv[])

{

int a[N][N];

int i, j;

int m;

scanf("%d",&m);

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

{

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

{

if (j==0 || i==j)

{

a[i][j]=1;

}

else

{

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

}

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

}

printf("\n");

}

return 0;

}

//int main(int argc, const char \*argv[])

//{

// //int a[5]={9, 7, 8, 5, 4};

// int a[LEN];

// int i,j;

// int k,temp;

// for (i=0; i<LEN; i++) {

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

// }

// for (i=0; i<LEN-1; i++) {

// k=i;

// for (j=i+1; j<LEN; j++) {

// if (a[k]>a[j]) {

// k=j;

// }

// }

// if (k!=i)

// {

// temp = a[k];

// a[k] = a[i];

// a[i] = temp;

// }

// }

// for (i = 0; i<LEN; i++) {

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

// }

// putchar('\n');

// return 0;

//}

第三天

#include <stdio.h>

//什么是地址: 我们说变量都有自己的地址, 计算机要找到某个变量,必须知道该变量的地址.我们可以通过 & 获得变量地址

//取地址符 : &

//通过取地址符就能把变量的地址给取出来

/\*int main(int argc, const char \* argv[])

{

int a;

printf("%p\n", &a);

return 0;

}\*/

//什么是指针?

//假如我们是做销售的, 我们拥有很多客户的地址, 那么我们需要一个通讯录来保存这些客户的地址, 同样对于计算机来说, 也需要用一个工具来保存变量的地址, 那么这个工具,就是指针

//指针就是用来保存内存地址的变量,本质是变量,变量存放的是地址,指针变量同样具有变量名,变量类型,变量值

//我们可以通过图形来表示

//指针定义

//一般形式: 类型 \* 标示符

//标示符: 表示指针变量的名字

//类型: 表示指针指向的数据类型

/\*int main(int argc, const char \*argv[])

{

int a;

int \*p;// 定义一个指针类型的变量p

p= &a;

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

printf("%p\n", &a);

return 0;

}\*/

//间接寻址符: \*

//通过指针进行读取

//如果知道上海图书馆的地址, 那么我们可以找到上海图书馆,并在上海图书馆内阅览;

//同理, 如果指针p保存了变量a的地址, 那么就可以通过指针p来找到变量a, 并读取变量a 的值

//通过指针进行修改

//只要知道恐怖分子的地址, 那么我们就能对恐怖分子进行有效的打击

//同理, 知道变量的地址,那么我们就能修改变量的值

/\*

int main(int argc, const char \*argv[])

{

int a= 10;

int \*p= &a;

printf("a== %d\n", \*p);

return 0;

}

\*/

/\*int main(int argc, const char \*argv[])

{

int a= 10;

int \*p= &a;

\*p = 20;

printf("a == %d", a);

return 0;

}\*/

//指针类型变量大小

//

/\*

int main(int argc, const char \*argv[])

{

printf("%ld\n", sizeof(char \*));

printf("%ld\n", sizeof(short \*));

printf("%ld\n", sizeof(int \*));

printf("%ld\n", sizeof(long \*));

printf("%ld\n", sizeof(long long \*));

printf("%ld\n", sizeof(float \*));

return 0;

}

\*/

//指针与数组

/\*int main(int argc, const char \*argv[])

{

int a[5] = {5,4,3,2,1};

int \*p = a;

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

printf("%d\n", (\*p)++);

printf("%d\n", \*p++);

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

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

return 0;

}\*/

//void \* 指针

//

//#define NULL ((void \*) 0)

//空指针 NULL #define NULL ((void \*)0)

//指针就是用来保存内存地址的变量, 因此定义一个指针后一定要用它来保存一个内存地址, 假如不那么做, 那么该指针就是一个野指针, 它的默认值是随机的, 因此会造成混乱

//NULL = (void \*)0

/\*int main(int argc, const char \*argv[])

{

int a = 10;

char ch = 'a';

void \*p=&a;

void \*q = &ch;

// \*p = 10;

printf("%d\n", \*(int \*)p);

printf("%c\n",\*(char \*)q);

//printf("%d", \*p);

return 0;

}\*/

//野指针

//当一个指针指向内容非法时, 那这就是一个野指针

//定义一个指针没有赋初值, 这个指针就是野指针

//一个指针指向的内容被销毁了,这个指针就是野指针

/\*int main(int argc, const char \*argv[])

{

int \*p=NULL;//undefine

\*p = 10;

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

return 0;

}\*/

//变更指针保存的地址

//指针直接对指针赋值

/\*

int main(int argc, const char \*argv[])

{

int a= 10, b= 20;

int \*p= &a;

printf("a==%d\n", \*p);

p = &b;

printf("b==%d\n", \*p);

return 0;

}

\*/

//指针自身的地址

//指针是变量,也占用内存,因此也有自己的地址

/\*int main(int argc, const char \*argv[])

{

int a=10;

int \*p = &a;

printf("p 指针的地址为: %p\n", &p);

printf("a 变量的地址为: %p\n", p);

printf("a 的值为: %d\n", \*p);

return 0;

}\*/

//指针运算

//指针的 自增自减运算

/\*int main(int argc, const char \*argv[])

{

int i= 10;

int \*p = &i;

printf("指针保存的地址为: %p\n", p);

p++;

printf("指针保存的地址为: %p\n", p);

p++;

printf("指针保存的地址为: %p\n", p);

p--;

printf("指针保存的地址为: %p\n", p);

p--;

printf("指针保存的地址为: %p\n", p);

return 0;

}\*/

//指针的相减运算

//指针不能进行加法运算

/\*int main(int argc, const char \*argv[])

{

int i= 0, j= 1;

int \*p1= &i;

int \*p2= &j;

printf("p1: %p\n", p1);

printf("p2: %p\n", p2);

long a = p1-p2;

printf("两块内存的地址差: ");

printf("%ld\n", a);

return 0;

}\*/

//指针的比较运算

//我们也可以比较两个指针的大小

/\*int main(int argc, const char \*argv[])

{

int a= 1, b= 2;

int \*p = &a;

int \*q = &b;

printf("p: %p\n" , p);

printf("q: %p\n" , q);

if (p>q) {

printf("p的内存地址大于q的内存地址\n");

}

else {

printf("p的内存地址小雨q的内存地址\n");

}

return 0;

}\*/

//指针运算符

// & 取地址符

// \* 间接寻址符

// & \* 为一对互逆运算符

/\*int main(int argc, const char \*argv[])

{

int a= 10;

int \*p = &a;

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

printf("%d\n",\*&a);

return 0;

}\*/

//指针变量作为函数参数

//封装一个函数交换两个值

/\*void swap(int a, int b)

{

int temp;

temp = a;

a = b;

b = temp;

return ;

}

int main(int argc, const char \*argv[])

{

int a = 10;

int b = 20;

printf("交换前 a== %d b== %d\n", a, b);

swap(a, b);

printf("交换后 a== %d b== %d\n", a, b);

return 0;

}\*/

//思考下为什么上面的函数值没有交换过来呢

//用什么方法能够解决这个问题呢?

/\*

void swap(int \*a , int \*b)

{

int temp;

temp = \*a;

\*a = \*b;

\*b = temp;

return ;

}

int main(int argc, const char \*argv[])

{

int a= 10, b= 20;

printf("交换前 a== %d b== %d\n", a, b);

swap(&a, &b);

printf("交换后 a== %d b== %d\n", a, b);

return 0 ;

}

\*/

//const 与指针

/\*

int main(int argc, const char \*argv[])

{

int i = 1;

int \*const p = &i;

\*p = 3;

//p++; //p 的指向不能被修改

printf("%d", i);

return 0;

}

\*/

// 指向常量的指针

/\*int main(int argc, const char \*argv[])

{

int i= 1;

const int \*p = &i;

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

p++;

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

return 0 ;

}\*/

//指向常量的常量指针

// const int \* const p;

/\*int main(int argc, const char \*argv[])

{

int i = 10;

const char \*const p = &i;

p++;

\*p = 20;

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

return 0;

}\*/

//指针与数组

//数组名就是数组的首元素地址,数组名是一个常量

/\*int main(int argc, const char \*argv[])

{

int a[5]={1, 2, 3, 4, 5};

printf("%p\n", a);

printf("%p\n", &a);

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

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

int a[5]= {1, 2, 3, 4, 5};

int \*p=a;

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

printf("a[%d]==%d\n", i,\*(p+i));

}

return 0;

}\*/

//指针作为参数

/\*void print(int \*src,int len)

{

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

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

}

printf("\n");

}

//数组名作为地址传入的时候会丢失数组长度

int main(int argc, const char \*argv[])

{

int a[5]={1, 2, 3, 4, 5};

print(a, 5);

return 0;

}\*/

//编写一个函数, 查找数组中的最大值

/\*int findmax(int \*src, int len)

{

int max;

max = \*src;

for (int i = 1; i< len; i++) {

if (max < \*(src+i)) {

max = \*(src+i);

}

}

return max;

}

int main(int argc, const char \*argv[])

{

int a[5];

printf("请输入5个整型数据: \n");

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

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

}

printf("The max value is: %d\n", findmax(a, 5));

return 0;

}\*/

//编写一个对整型数组排序的函数

//选择排序法

/\*void sort(int \*src, int len)

{

int i, j, k,temp;

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

k = i;

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

if(\*(src+k) > \*(src + j))

{

k = j;

}

}

if (k!=i) {

temp = \*(src+k);

\*(src+k)= \*(src+i);

\*(src+i)= temp;

}

}

}

int main(int argc, const char \*argv[])

{

int a[5];

printf("请输入5个整型数据: \n");

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

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

}

sort(a, 5);

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

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

}

printf("\n");

return 0;

}\*/

int main(int argc,const char \*argv[])

{

int \*p,\*q;

int a,b;

p = &a;

q = &b;

return 0;

}

第四天

#include <stdio.h>

#include <stdlib.h>

//指针与数组

//指针作为参数的这种特性的实际意义:

//最重要的一个就是可以实现多个返回值。

//(从函数带回多个值)

//函数不能返回数组

//函数返回指针值:不能返回局部变量的地址

//下标访问数组

//数组名访问数组

/\*int main(int argc, const char \*argv[])

{

int a[5]={1,2,3,4,5};

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

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

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

return 0;

}\*/

//指针访问数组

/\*int main(int argc, const char \*argv[])

{

int a[5]={1,2,3,4,5};

int \*p = a;

printf("%d\n",\*(p+=2));

printf("%d\n",\*(++p));

return 0;

}\*/

//利用指针,输入10个数到数组中, 再输出

/\*int main(int argc, const char \*argv[])

{

int a[10];

int \*p = a;

for (int i=0; i< 10; i++) {

scanf("%d", p++);

}

p = a;

for (int i=0; i<10; i++) {

printf("%d ", \*p++);

}

return 0;

}\*/

//指针实现数组逆序

/\*int main(int argc, const char \*\*argv)

{

int a[10]={1,2,3,4,5,6,7,8,9,10};

int \*p = a;

int temp;

for (int i=0; i<10/2; i++) {

temp = \*(p+i);

\*(p+i)=\*(p+9-i);

\*(p+9-i)=temp;

}

for (int i=0; i<10; i++) {

printf("%d ", \*(p++));

}

return 0;

}\*/

//练习:编写函数 replace, 在一个字符数组里面查找指定字符,并用相应的字符替代。函数原型如下:

//void replace(char \*array, char old, char new, int length)

/\*#include <string.h>

char \* replace(char \*src,char old, char new, int len)

{

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

if (\*(src+i)==old) {

\*(src+i) = new;

}

}

return src;

}

int main(int argc, const char \*argv[])

{

char str[100]="qianfengjiaoyu";

printf("%s\n", replace(str, 'f', 'z', (int)strlen(str)));

return 0;

}\*/

//练习:编写函数 insert, 向一个字符数组指定位置插入一个字符,后 面的字符依次向后移动。函数原型如下:

//void insert(char \*array, int index, char new, int length)

#include <string.h>

/\*char \* intsert(char \*src, int index, char new, int len)

{

int i;

for (i=len; i>=index; i--) {

\*(src+i+1)= \*(src+i);

}

\*(src+i) = new;

return src;

}

int main(int argc, const char \*argv[])

{

char str[100]="zhongguohaoshengyin";

printf("%s\n", intsert(str, 5, 'A', (int)strlen(str)));

return 0;

}\*/

//数组作为函数参数

//传递一个数组,从数组中查找某个值

/\*int findvalue(int a[], int len, int n)

{

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

if (n == \*(a+i)) {

return i;//返回下标

}

}

return -1;//返回下标

}

int main(int argc, const char \*argv[])

{

int a[5]={1,2,3,4,5};

printf("%d\n", findvalue(a,5,5));

return 0;

}\*/

//指针的数组(指针数组)

//数组中的每个元素都是一个指针(若干个指针的集合)

//如: int \*p[10]

//p中的每个元素都是一个int \*型的指针

/\*int main(int argc, const char \* argv[])

{

char \*ps[4]={"hello","world","qian","feng"};

for (int i =0; i<4; i++) {

printf("%s\n", ps[i]);

}

return 0;

}\*/

/\*int main(int argc, const char \*\*argv)

{

char \*p[10];

int i;

for (i=0; i<10; i++) {

p[i] = malloc(sizeof(char)\*100);

}

for (i=0; i<10; i++) {

scanf("%s", p[i]);

printf("%s\n",p[i]);

if (getchar()=='\n') {

break;

}

}

return 0;

}\*/

//指针数组与二维数组

//int \*a[5] int a[5][10]

/\*int main(int argc, const char \*argv[])

{

int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};

int \*b[3];

for (int i=0; i<3; i++) {

b[i]=a[i];

printf("%p\n", b[i]);

}

for (int i= 0; i<3; i++) {

for (int j=0; j<4; j++) {

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

printf("%p\n", b[i]);

}

printf("\n");

}

return 0;

}\*/

//指向数组的指针

//int (\*arr)[5];

//arr是一个指针, 是一个指向有5个整型元素的指针

// int a[10]

// int (\*pa)[10]=&a;

//注意:a是一个数组，在&a这个表达式中，数组名做左值，取整个数组的首地址赋给指针pa。

//注意，&a[0]表示数组a的首元素的首地址，而&a表示数组a的首地址，显然这两个地址的数值相同，但这两个表达式的类型是两种不同的指针类型，前者的类型是int \*，而后者的类型是int (\*)[10]。\*pa就表示pa所指向的数组a，所以取数组的a[0]元素可以用表达式(\*pa)[0]。注意到\*pa可以写成pa[0]，所以(\*pa)[0]这个表达式也可以改写成pa[0][0]，pa就像一个二维数组的名字，它表示什么含义呢？下面把pa和二维数组放在一起做个分析。

/\*int sum(int (\*arr)[5])

{

int sum=0;

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

sum+=(\*arr)[i];

}

return sum;

}

int main(int argc, const char \*argv[])

{

int a[5]={1, 2, 3, 4, 5};

printf("%d\n", sum(&a));//为什么传&a, 而不是a呢?

return 0;

}

\*/

//指向数组的指针 跟 二维数组

//指向二维数组的指针

/\*int main(int argc, const char \*argv[])

{

int a[4][2]={1,2,3,4,5,6,7,8};

int (\*p)[2];

p=&a[0];

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

p++;

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

p++;

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

p--;

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

return 0;

}\*/

/\*int sum(int (\*arr)[2])

{

int sum=0;

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

sum+=(\*arr)[i];

}

return sum;

}

int main(int argc, const char \*argv[])

{

int a[4][2]={1,2,3,4,5,6,7,8};

int (\*p[3])[2];

for (int i=0; i<4; i++) {

p[i]=&a[i];

}

for (int i=0; i<4; i++) {

printf("%d ", sum(p[i]));

}

return 0;

}\*/

//指向指针的指针(二级指针)

//int a=10;

//int \*p = &a;

//int \*\*q = &p;

/\*int main(int argc, const char \*argv[])

{

int a= 10;

int \*p = &a;

int \*\*q = &p;

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

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

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

printf("%p\n", &p);

printf("%p\n", \*q);

printf("%p\n", q);

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

return 0;

}\*/

//指向函数的指针

//int (\*pfunc)(int a, int b)//声明函数指针

//int (\*pfunc)(int a, int b)//声明函数指针数组

//声明一个指向函数的指针

//a, b: 指向的函数有两个int类型的参数

//pfunc: 指针的变量名为pfunc

//\* : 说明是一个指针类型的变量

//int : 指向的函数的返回值类型是int 类型

//int \*func(int a, int b)

//声明一个函数func

//函数的返回值是一个int \*型的指针

/\*int add(int a, int b)

{

return a+b;

}

int main(int argc, const char \*argv[])

{

int (\*pfunc)(int a, int b)=add;

printf("%d\n",(\*pfunc)(3,5));

return 0;

}\*/

//函数指针:

// 数据类型 (\*指针变量名) (参数列表)

//函数名代表一个函数的地址

//函数指针可以保存任何一个函数的地址

//

//int \*((\*q)[3])[3];

//int \*(\*q[3])[3]

//利用函数指针实现回调函数

#include <stdio.h>

/\*

void printWelcome(int len)

{

printf("Hello world -- %d\n", len);

}

void printGoodbye(int len)

{

printf("Goodbye -- %d\n", len);

}

void callback(int times, void (\* print)(int))

{

int i;

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

{

print(i);

}

printf("\n我不知道你是迎客还是送客!\n\n");

}

int main(void)

{

callback(10, printWelcome);

callback(10, printGoodbye);

printWelcome(5);

return 0;

}\*/

第五天

#include <stdio.h>

#include <string.h>

#include <ctype.h>

//字符串

//char ch[5]={'a','b','c','d','e'};

//"qianfeng"

//双引号括起来

//每个字符占用一个字节

//以'\0'表示字符串结束

//

/\*int main(int argc, const char \* argv[])

{

char str[100]="qianfeng\0jiaoyu\0helloworld";

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

printf("%s\n",&str[9]);

printf("%s\n",&str[16]);

return 0;

}\*/

//求字符串长度

//字符串有效长度不包含'\0'

//sizeof 是一个运算符, 求变量或者常量占用内存空间大小

//strlen是一个函数, 求字符串有效字符个数

int mystrlen(const char \* src)

{

int i=0;

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

{

i++;

}

return i;

}

/\*int main(int argc, const char \*argv[])

{

char str[100]="hello world";

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

printf("%ld \n", sizeof(str));

printf("%ld \n", sizeof("hello world"));

printf("str == %ld \n", strlen(str));

return 0;

}\*/

/\*

int isdigit(int);

int islower(int);

int isspace(int);

int isupper(int);

int tolower(int);

int toupper(int);

int digittoint(int);

int isnumber(int);

\*/

/\*int main(int argc, const char \*argv[])

{

printf("digit = %d\n ",isdigit('9'));

printf("lower = %d\n ",islower('z'));

printf("space = %d\n ",isspace('h'));

printf("upper = %d\n ",isupper('A'));

printf("tolower = %c\n", tolower('A'));

printf("toupper = %c\n",toupper('a'));

printf("int = %d\n",digittoint('9'));

printf("isnumber = %d\n", isnumber('9'));

return 0;

}\*/

//字符串拷贝

//strcpy

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

//strncpy

//char \*strncpy(char \*, const char \*, size\_t);

//s1的内存空间足够容纳s2字符串

//strncpy 选择拷贝的字节数, 不一定包含'\0';

char \*mystrcpy(char \*dst, const char \*src)

{

int i =0;

while (src[i]) {

dst[i]=src[i];

i++;

}

dst[i]='\0';

return dst;

}

/\*int main(int argc, const char \*argv[])

{

char s1[100];

char s2[50]="hello world\n";

//char \*str = strcpy(s1, s2);

//printf("%s\n",strcpy(s1, s2));

//printf("%s\n",str);

//printf("%s\n",mystrcpy(s1, s2));

strncpy(s1, s2, 10);

s1[10]='\0';

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

return 0;

}\*/

//字符串拼接函数

//strcat

//char \*strcat(char \*, const char \*);

//strncat

//char \*strncat(char \*, const char \*, size\_t);

//strncat在拼接后的新字符串后会添加一个'\0';

char \*mystrcat(char \*s1, const char \*s2)

{

int i=0,j;

while (s1[i]) {

i++;

}

for (j=0; s2[j]!='\0'; j++) {

s1[i+j]=s2[j];

}

s1[i+j]='\0';

return s1;

}

/\*int main(int argc, const char \*argv[])

{

char s1[100]="hello world";

char s2[50]="qianfeng";

//printf("%s\n",strcat(s1, s2));

//printf("%s\n",mystrcat(s1, s2));

printf("%s\n",strncat(s1, s2, 5));

return 0;

}\*/

//字符串比较函数

//strcmp

//int strcmp(const char \*, const char \*);

//strncmp

//int strncmp(const char \*, const char \*, size\_t);

// s1 > s2 返回一个正数

// s1 == s2 返回0

// s1 < s2 返回一个负数

int mystrcmp(const char\* s1, const char \*s2)

{

int i;

for (i=0; (s1[i]!='\0')&&(s2[i]!='\0') ; i++) {

if (s1[i]==s2[i]) {

continue;

}

else

{

break;

//return s1[i]-s2[i];

}

}

return s1[i]-s2[i];

}

/\*int main(int argc, const char \*argv[])

{

char \*str1 = "hello borld";

char \*str2 = "hello zorld";

//printf("%d\n", mystrcmp(str1, str2));

printf("%d\n", strncmp(str1, str2, 7));

//char s1[30]="hello world";

return 0;

}\*/

/\*

int main(int argc, const char \*argv[])

{

char str[100];

int i=0;

while (1) {

scanf("%c",&str[i]);

if (str[i]=='\n') {

str[i]='\0';

break;

}

i++;

}

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

return 0;

}\*/

/\*int main(int argc ,const char \*argv[])

{

char str[100];

char str1[100];

fscanf(stdin,"%s", str);

fscanf(stdin, "%s",str1);

fprintf(stdout,"%s\n", str);

fprintf(stdout, "%s\n",str1);

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

char str[100];

//gets(str);//除了'\n'

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

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

return 0;

}\*/

//字符串查找函数

//strchr

//strrchr

//char \*strchr(const char \*, int);

//char \*strrchr(const char \*, int);

char \*mystrchr(const char \*s1, char ch)

{

int i=0;

while (s1[i]) {

if (s1[i]==ch) {

break;

}

i++;

}

return (char \*)&s1[i];

}

/\*int main(int argc, const char \*argv[])

{

char str[100]="hello world!";

char \*pstr = mystrchr(str, 'l');

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

//printf("%s\n", pstr);

return 0;

}\*/

/\*int main(int argc, const char \*argv[])

{

char str[100]="hello world!";

char \*pstr = strrchr(str, 'l');

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

return 0;

}\*/

//查找字符串函数

//strstr

//char \*strstr(const char \*, const char \*);

// 0 '0' "0" NULL

int main(int argc, const char \*argv[])

{

char str1[100]="helloworld helloworld helloworld";

printf("%p\n",&str1[strlen(str1)]);

// char \*pstr = strstr(str1, "world");

// printf("%s\n", pstr);

char \*p = str1;

while (p)

{

p=strstr(p, "world");

if (p)

{

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

p+=sizeof("world")-1;

}

}

//"helloworld helloworld helloworld"

return 0;

}

//字符串分割函数

//strtok\_r

//char \*strtok(char \*, const char \*);

/\*int main(int argc, const char \*argv[])

{

char str[100]="hello world:good-bye:://qian^feng..miss()you///";

// char \*p=strtok(str, ": ");

// printf("%s\n",p);

// while ((p=strtok(NULL , ": /^.()-")))

// {

// printf("%s\n",p);

// }

char \*p = str;

while ((p=strtok(p, ": /^.()-")))

{

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

p=NULL;

}

// p = strtok(NULL, ":");

// printf("%s\n",p);

return 0;

}\*/

//sscanf

//int sscanf(const char \* \_\_restrict, const char \* \_\_restrict, ...) \_\_scanflike(2, 3);

//sprintf

//int sprintf(char \* \_\_restrict, const char \* \_\_restrict, ...) \_\_printflike(2, 3);

//int main(int argc, const char \*argv[])

//{

// char str[100]="12a1234\*678";

// int a, b, c;

// sscanf(str, "%da%d\*%d", &a,&b,&c);

// printf("%d %d %d\n", a, b, c);

// return 0;

//}

/\*int main(int argc, const char \*argv[])

{

char buf[100];

int a = 100;

char str[50]="hello world!";

float b = 3.14;

sprintf(buf, "<%d><%s><%f>",a,str,b);

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

return 0;

}\*/

//atoi

//atof

//double atof(const char \*);

//int atoi(const char \*);

#include <stdlib.h>

/\*int main(int argc, const char \*argv[])

{

char \*str="-12-3.34567";

char \*str1="-123-wesd23456789";

printf("%d\n", atoi(str1));

printf("%f\n", atof(str));

return 0;

}\*/

//itoa

//

//空格作为分割符统计输入单词个数

//

/\*int main(int argc, const char \*argv[])

{

char str[200];

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

int i=0;

int flag=1;

int cnt=0;

while (str[i]!='\0') {

if (str[i]==' ') {

if (!flag) {

flag = 1;

}

}

else

{

if (flag) {

cnt++;

flag =0;

}

}

i++;

}

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

return 0;

}\*/