**内蒙古农业大学**

**计算机与信息工程学院**

实 验 报 告

实验三

课程名称： 嵌入式基础实践

班 级： 21计科四班

姓 名： 李振浩

学 号： 2021122143170

授课教师： 李慧旻

**实验三 C语言进阶-指针实验**

1. **实验目的**
2. 掌握指针的定义与应用编程；
3. 掌握指针作为函数参数的编程方法；
4. 熟悉C语言中多文件编程的基本方法。
5. **实验任务与要求**
6. 根据实验内容编写C语言程序；
7. 写出各个运程序的行结果并分析；
8. 总结C语言中多文件编程和指针作为函数参数的程序设计方法和参数传递过程。
9. **实验内容与结果**
10. 编程实现如下几种指针变量的定义，并给每个指针变量赋值。
11. 定义整型、字符型和double型指针变量pa、pc、pd；

1 #include<stdio.h>

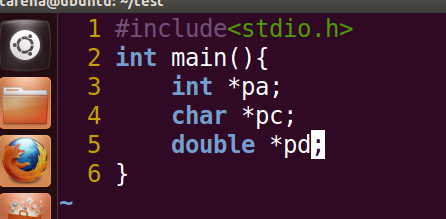
2 int main(){

3 int \*pa;

4 char \*pc;

5 double \*pd;

6 }



1. 定义一个整型数组a[5]，并进行赋值，定义一个指针变量pt，使pt指向数组a，利用指针pt输出数组a中的每一个元素的值。

1 #include<stdio.h>

2 int main(){

3 int \*pt;

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

5 int i;

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

7 pt=arr+i;

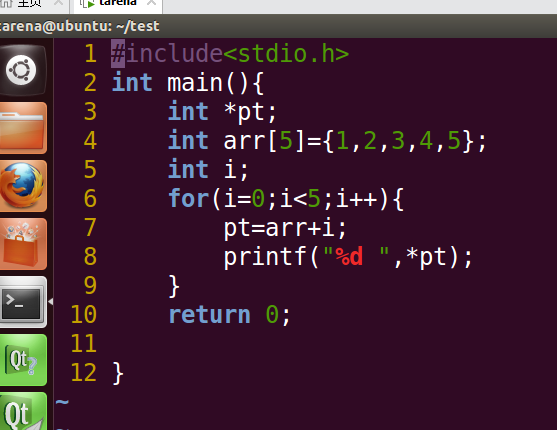
8 printf("%d ",\*pt);

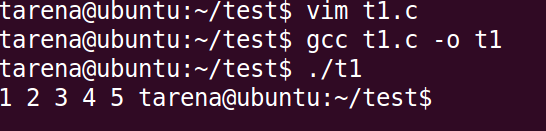
9 }

10 return 0;

11

12 }





1. 定义一个结构体，其成员变量为“学号”、“姓名”、“成绩”，再定义一个该结构体的变量s1及一个该结构体变量的数组s[2]，定义两个结构体指针p1和p2，分别指向s1和数组s，利用p1和p2输出结构体变量s1和结构体数组s中的成员变量的值。

#include<stdio.h>

typedef struct{

int sno;

char name[50];

int grade;

} stu;

int main(){

stu s1={2021101,"zhangsan",60};

stu s[2]={{2021102,"lisi",100},{2021103,"wangwu",59}};

stu \*p1=&s1;

stu \*p2;

printf("%d ,%s ,%d \n",p1->sno,p1->name,p1->grade);

int i=0;

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

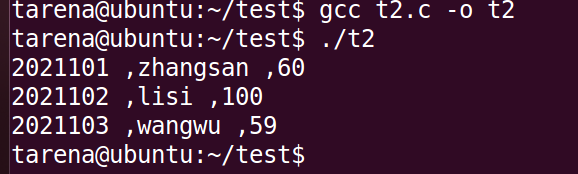
p2=s+i;

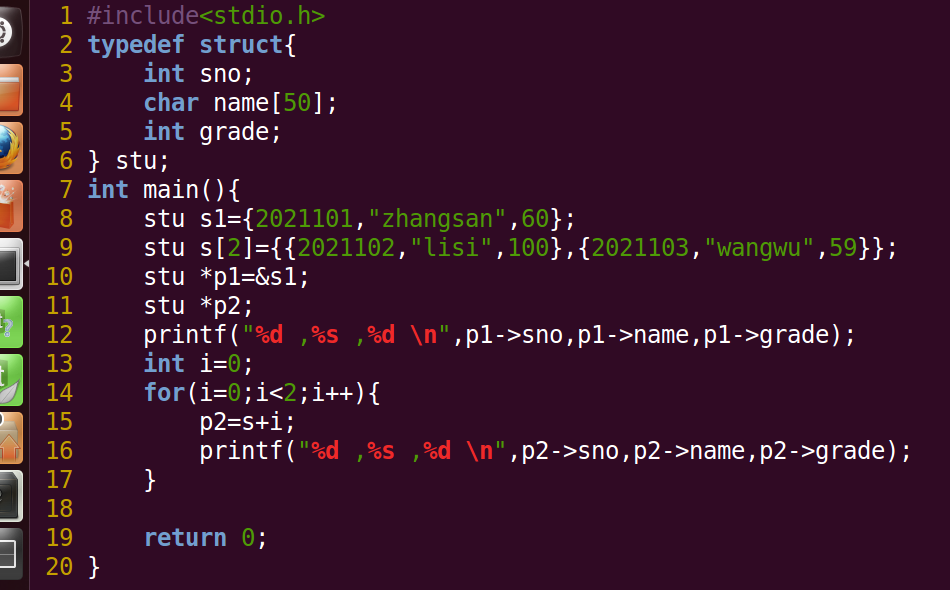
printf("%d ,%s ,%d \n",p2->sno,p2->name,p2->grade);

}

return 0;

}





1. 定义一个字符数组a[5]，并将字符串“hello world”存入数组a中，再定义一个指针变量p，p指向数组a，利用指针p输出数组a中的字符串“world”。

1 #include<stdio.h>

2 int main(){

3 char a[12]={"hello world"};

4 char \*p;

5 p=a;

6 int i;

7 for(i=6;i<11;i++){

8 p=a+i;

9 printf("%c",\*p);

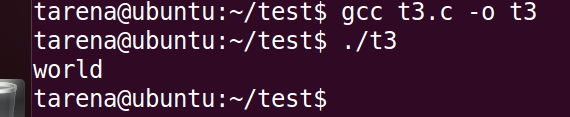
10 }

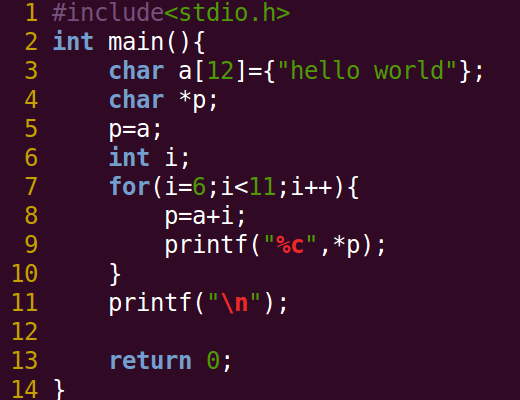
11 printf("\n");

12

13 return 0;

14 }





1. 编写一个程序实现两个整数的交换，要求按下列要求完成任务：
2. 定义一个函数swap(int x, int y)，在函数swap中交换x，y的值，利用传值的参数传递方式调用swap( )函数，观察主调函数main( )中的实参值是否交换，并分别输出main( )函数和swap( )函数中的两个整数值。

1 #include<stdio.h>

2 void swap(int x,int y){

3 int temp;

4 temp=x;

5 x=y;

6 y=temp;

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

8 }

9 int main(){

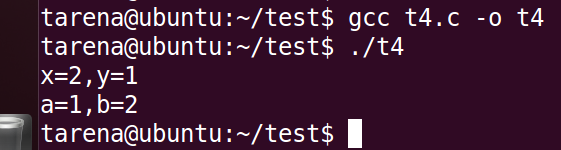
10 int a=1,b=2;

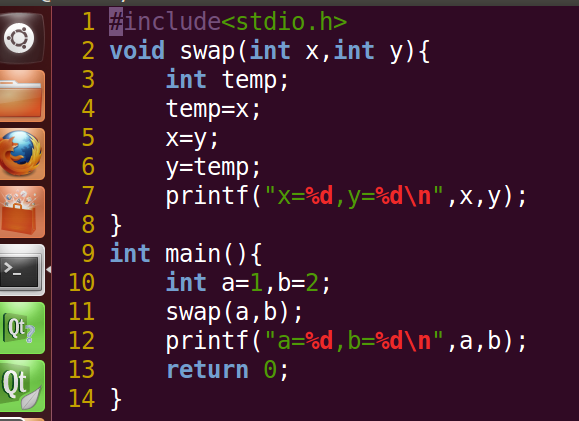
11 swap(a,b);

12 printf("a=%d,b=%d\n",a,b);

13 return 0;

14 }





1. 定义一个函数swap1(int \*px, int \*py)，在swap1中实现交换两个指针px和py，观察主调函数main( )中的实参值是否交换，并分别输出main( )函数和swap( )函数中的两个整数值。

1 #include<stdio.h>

2 void swap1(int \*px,int\*py){

3 int \*temp;

4 temp=px;

5 px=py;

6 py=temp;

7 printf("swap function\n");

8 printf("x=%d,y=%d\n",\*px,\*py);

9 }

10

11 int main(){

12 int x=1,y=2;

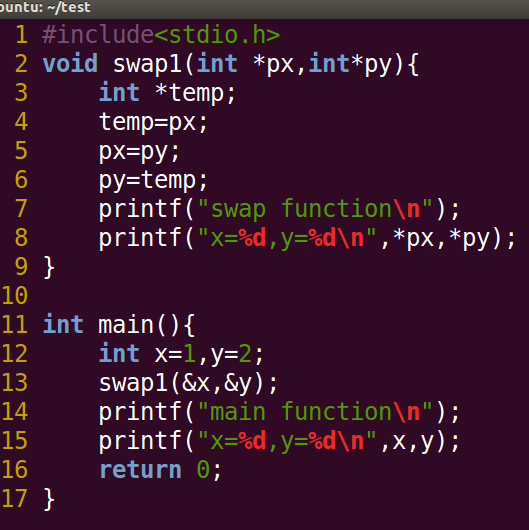
13 swap1(&x,&y);

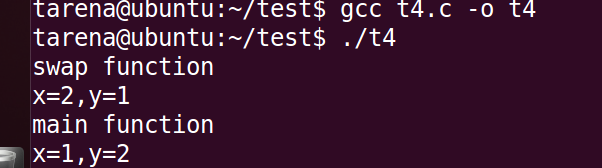
14 printf("main function\n");

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

16 return 0;

17 }





1. 定义一个函数swap2(int \*px, int \*py)，在swap1中实现交换两个指针指向的变量的值\*px和\*py，观察主调函数main( )中的实参值是否交换，并分别输出main( )函数和swap2( )函数中的两个整数值。

1 #include<stdio.h>

2 void swap2(int \*px,int\*py){

3 int temp;

4 temp=\*px;

5 \*px=\*py;

6 \*py=temp;

7 printf("x=%d,y=%d\n",\*px,\*py);

8 }

9 int main(){

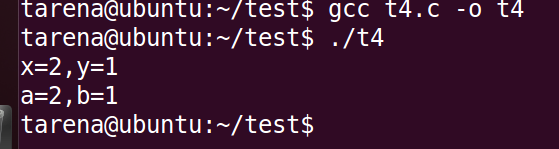
10 int a=1,b=2;

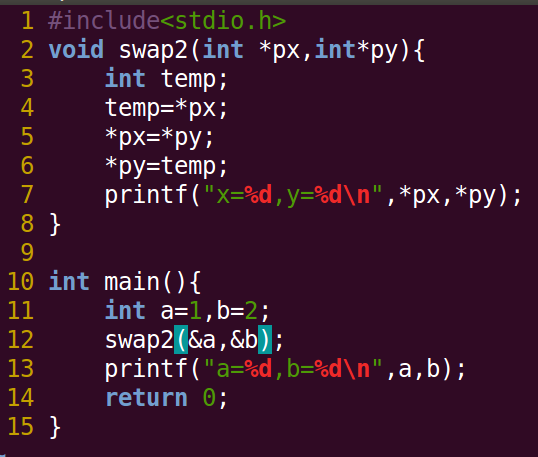
11 swap2(&a,&b);

12 printf("a=%d,b=%d\n",a,b);

13 return 0;

14 }





1. 多文件编程：定义一个结构体，其成员变量为“学号”、“姓名”、“成绩”，再定义一个该结构体变量的数组s[5]，并给数组s中的每个成员变量赋值，利用多文件编程方法编写源文件和头文件完成下列任务：
2. 编写一个文件要求文件中的函数实现将s[5]中的成绩按照升序排序，并显示排序后的5名学生信息。

#include<stdio.h>

2 #include"stu.h"

3 void sort\_grade(stu s[]){

4 int i,j;

5 stu temp;

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

7 for(j=0;j<4-i;j++){

8 if(s[j].grade>s[j+1].grade){

9 temp=s[j];

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

11 s[j+1]=temp;

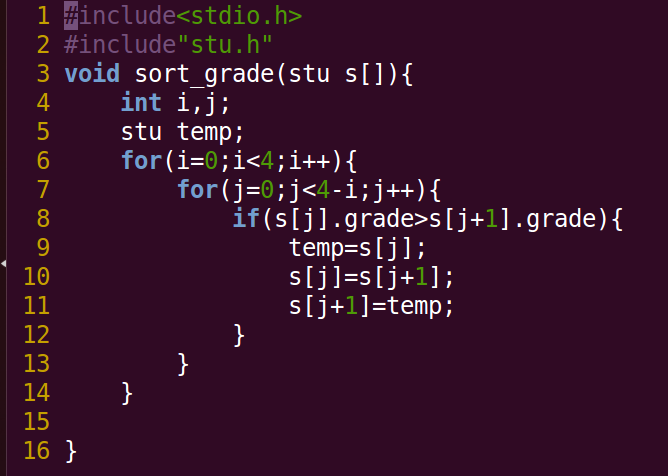
12 }

13 }

14 }

15

16 }



1. 编写一个文件要求文件中的函数实现将s[5]中的成绩不及格的学生信息显示出来。

1 #include<stdio.h>

2 #include"stu.h"

3 void display(stu s[]){

4 int i;

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

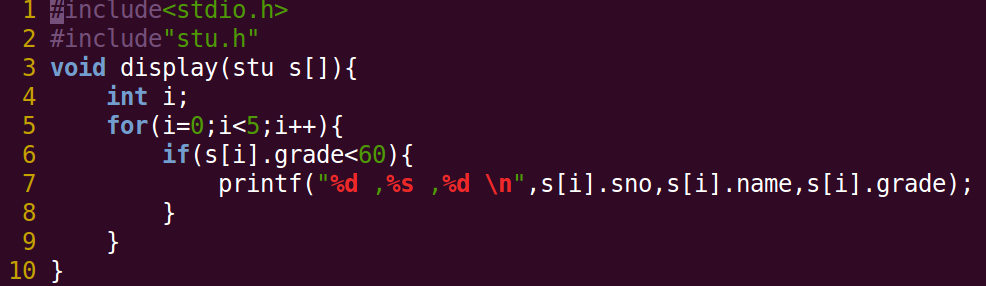
6 if(s[i].grade<60){

7 printf("%d ,%s ,%d \n",s[i].sno,s[i].name,s[i].grade);

8 }

9 }

10 }



1. 编写一个文件，要求文件中是main函数用来定义结构体数组变量s[5]，并给5名学生的成员变量赋值，分别调用上面两个文件中的函数实现成绩排序和不及格学生信息。

1 #ifndef \_STU\_H\_

2 #define \_STU\_H\_

3 typedef struct {

4 int sno;

5 char name[50];

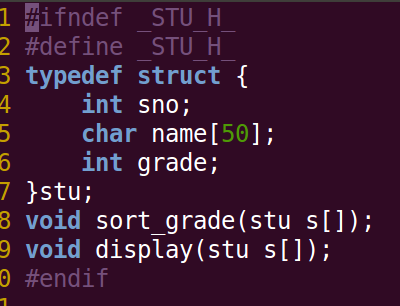
6 int grade;

7 }stu;

8 void sort\_grade(stu s[]);

9 void display(stu s[]);

10 #endif



1 #include<stdio.h>

2 #include"stu.h"

3 int main(){

4 stu s[5]={{101,"zs",30},{102,"ls",100},{103,"ww",59},{104,"zl",80},{105,"qq",90}};

5 sort\_grade(s);

6 int i;

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

8 printf("%d ,%s ,%d \n",s[i].sno,s[i].name,s[i].grade);

9 }

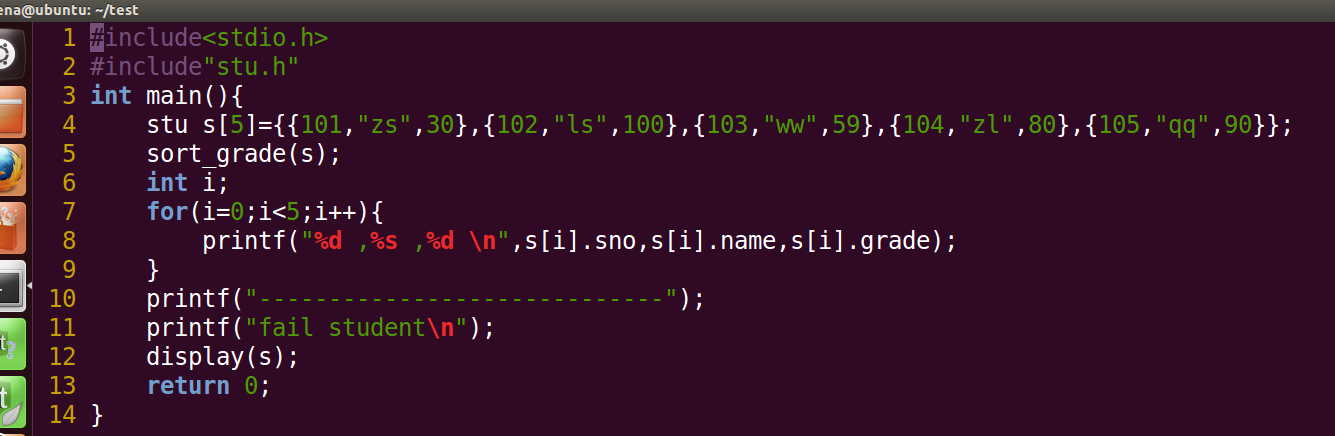
10 printf("-----------------------------");

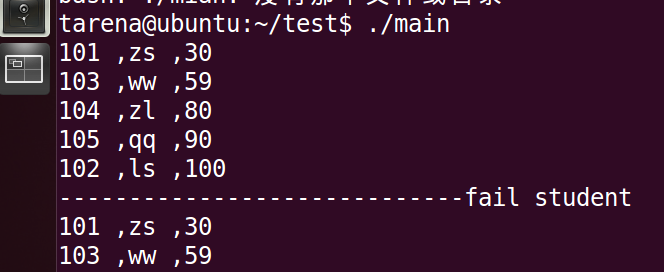
11 printf("fail student\n");

12 display(s);

13 return 0;

14 }





1. 请利用main函数的参数编写程序实现计算器的功能。

如：运行程序及结果如下：

tarena@ubuntu:~/test/d1$ ./t1 5 + 8

5 + 8=13.000000

tarena@ubuntu:~/test/d1$ ./t1 9 / 3

9/3=3.000000

#include<stdio.h>

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

3 int i;

4 int a=atoi(argv[1]);

5 char c=argv[2][0];

6 int b=atoi(argv[3]);

7 double result;

8 switch(c){

9 case '+':

10 result=a+b;

11 printf("%d+%d=%lf\n",a,b,result);

12 break;

13 case '-':

14 result=a-b;

15 printf("%d-%d=%lf\n",a,b,result);

16 break;

17 case '\*':

18 result=a\*b;

19 printf("%d\*%d=%lf\n",a,b,result);

20 break;

21 case '/':

22 result=a/b;

23 printf("%d/%d=%lf\n",a,b,result);

24 break;

25 default :

26 printf("error\n");

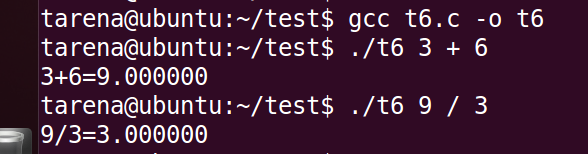
27 break;

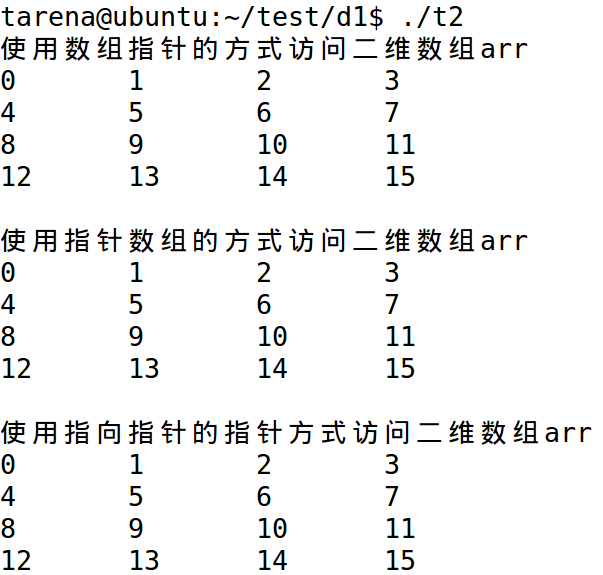
28 }

29 return 0;

30 }





1. 深入理解数组指针和指针数组的含义，利用数组指针、指针数组以及指向指针的指针分别输出二维数组的值，程序结果如下图所示。

1 #include<stdio.h>

2 int main(){

3 int arr[4][4]={{0,1,2,3},{4,5,6,7},{8,9,10,11},{12,13,14,15}};

4 int i,j;

5 printf("array ptr\n");

6 int \*p[4];

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

8 p[i]=arr[i];

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

10 printf("%d\t",\*p[i]+j);

11 }

12 printf("\n");

13 }

14 printf("ptr array\n");

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

16 int (\*p1)[4]=arr+i;

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

18 printf("%d\t",\*(\*p1+j));

19 }

20 printf("\n");

21 }

22 printf(" ptr ptr \n");

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

24 int \*p2=\*(arr+i);

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

26 int \*p3=p2+j;

27 printf("%d\t",\*p3);

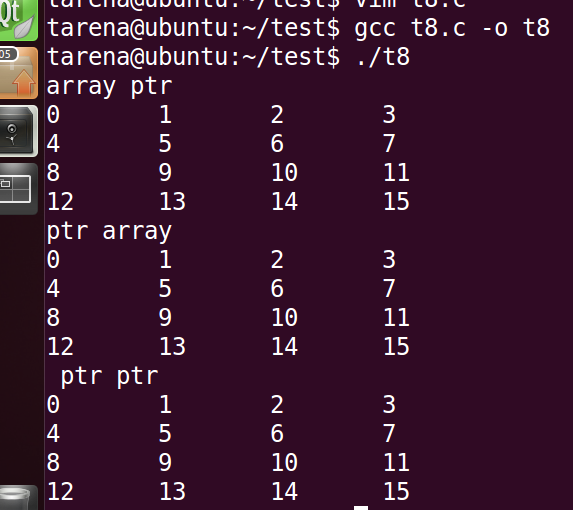
28 }

29 printf("\n");

30 }

31 return 0;

32 }





1. **实验总结**

1.使用C语言指针需要注意以下几个点：

指针的初始化：定义指针变量时，要进行初始化，可以将它初始化为NULL，表示该指针不指向任何地址。

指针变量的赋值：指针变量可以被赋值为另一个指针的值，或者是某个变量的地址。

指针运算：指针变量可以进行加减运算，但必须保证指针指向的内存空间是有效的。

解引用操作：指针变量可以通过解引用操作获取指针所指向的内存地址上的值。

空指针检查：在使用指针变量之前，需要先进行空指针检查，以防止程序崩溃或出现未知错误。

避免野指针：指针变量在使用前要确保指向有效的内存空间，避免产生野指针。

2.argc参数保存命令行参数的个数，argv为一个指向字符指针数组的指针。并且所有的命令行参数都是字符串。读取数据时通常从下标1开始。

3.数组指针与指针数组的容易混淆，数组指针是指向数组的指针变量。它指向数组的首元素地址。指针数组是指包含指针的数组。它是一个数组，其中的每一个元素都是一个指针。用于存储指针的地址值。

4.指针指向地址，难度比较大，一维指针还好理解，但是那些结构体指针，数组指针，指针数组，二维指针使用起来就绕来绕去非常容易出错，需要经常练习来增加熟练度。