指针是啥：

首先是取地址符&

使用%p可以打印输出变量的地址

使用的方法如下

printf（”%p“, &s）;

其所输出的是变量的地址的值

如果在声明变量的时候没有给变量赋值那么其就是不确定的，要严格杜绝

指针变量的定义及其初始化

类型关键字 * 指针变量名

其中变量的关键字可以看做是指针变量的基类型

为了避免爆炸，所以不知道的时候就将其设置为null进行初始化

int a = null;

a = &b;

请一定要将一个变量的值得地址赋值给它

指针变量只能指向同一基类型的变量

直接寻址和间接寻址

直接按照变量名或者变量的地址存取变量的内容的访问方式称为直接寻址

scanf("%d",&a);获取变量的地址需要使用地址运算符

printf("%d",a);

通过指针变量间接存取它所指的变量的访问方式，称为间接寻址

指针运算符，间接寻址运算符，解引用运算符，用*来进行访问

不要使用未初始化的指针

使用指针的三条准则：

指针必须指向一块有意义的内存

清楚的知道指针所指的对象的内容是什么

不要使用未初始化的指针变量

按值调用和“模拟”按引用调用

前面的学习的内容都是采用的一种按值调用的过程

那么其在传递参数的时候，传递的只**是实参的副本==形参，因而在函数内部不会改变实参的值**

要想改变实参的值，那么将参数的地址传递过去是极为有必要的。

0.C语言中所有的函数调用都是按值调用
1.指针在作为参数的时候，传递的不是变量的值，而是变量的地址
2.那么通过修改变量的地址上的值

也就是说：
传递的某个变量的地址值可以在被调用函数中改变主调函数的变量的值
相当于模拟了C++中的按引用调用，因此称为模拟按引用调用
使用的方法如下

```
fun(&a,&b,&c);
fun(int *a,int *b,char *c)
```

最最经典的题目：

```
#include <stdio.h>
//返回值是形参，该函数无法改变主函数中的实参的值
void fun(int a,int b)
{
        int temp;
        temp = a;
        a = b;
        b = temp;
}
//该函数直接修改主函数地址上的值，属于模拟按引用调用
void funn(int *a,int *b)
{
        int temp;
        temp = *a;
        *a = *b;
        *b = temp;
}
int main()
{
        int a;int b;
        a = 7;
        b = 8;
```

```
        fun(a,b);
        printf("fun函数的返回值\n");
        printf("%d,%d\n",a,b);
        funn(&a,&b);
        printf("funn函数的返回值\n");
        printf("%d,%d\n",a,b);
}
```

和函数的使用及其相似

用指针变量作为函数的参数的程序还有个一个好处：

1.使得函数的返回值变多了

2.使得函数可以修改主函数的数据

3.当使用函数的指针作为输入的参数，且接受返回值时，那么这一个、多个指针就可以被称作是**函数的出口参数**

只做输入求解输出的称为**函数的入口参数**

很神奇和冷门的一点内容：

函数指针及其应用：

第一个问题：啥叫函数指针呀？

函数指针就是指向函数的指针，指向函数的指针变量中存储的是一个函数在内存中的入口地址。

冯诺依曼体系结构强调程序和数据共同存储在内存中，函数是子程序，当然也存储在内存中，指向储存这个函数的第一条指令的地址，称为函数的入口地址。

数组名其实就是存储了数组单元的第一个元素的内存地址

同理

一个函数的的函数名就是这个函数的源代码在内存中的起始位置。

编译器将不带（）的函数名解释为该函数的入口地址。

例如一个函数指针的使用方式

比如：

```
int a(int b,int c,int (*e)(int k,int j))


int (*e)(int k,int j)
```

返回值为int类型的函数指针

int *e(int k,int j)
返回值为int指针的函数
括号是最优先处理的

再看其的使用方法上:
int a(int b,int c,int (*e)(int k,int j))
的调用的方法为
a（参数吧，参数c，符合要求的函数e）
函数e的声明长什么样子呢?
这样：

```
int e1(int a,int b)
{
}
```

下面来看例子；

```
#include <stdio.h>
#include <stdlib.h>
//返回值是形参，该函数无法改变主函数中的实参的值

//该函数直接修改主函数地址上的值，属于模拟按引用调用
int sum(int a,int b)
{
        int temp;
        temp = a+b;
    return temp;
}

int abd(int a,int b)
{
        int temp;
        temp = (a+b)/2;
    return temp;
}
int rest(int a,int b,int (*temp)(int a,int b))
{
int test;
```

```
test = (*temp)(a,b);
//在此处使用这个指针函数的时候，请务必将参数也一并带齐
//a,b两个数值都应该在大函数体下进行提供，否则讲无法实现大函数调用小函数的功能

return test;
}

int main()
{
        int a;
    int b;
        a = 7;
        b = 8;
    int test1;
    int test2;
        test1 = rest(a,b,abd);
        printf("fun函数的返回值(平均)\n");
        printf("%d\n",test1);
        test2 = rest(a,b,sum);
        printf("fun函数的返回值（求和）\n");
        printf("%d\n",test2);
        system("pause");
        return 0;
}
```

总结内容：
指针变量是一种特殊的数据类型
指针变量是具有指针类型的变量，用于存储变量的地址值的变量

指针变量与其他的类型比较
同：都要占内存；都要先定义，后使用；
特殊性：
其的内容只能是地址，而不能是数据；
必须初始化后才能使用；
只能使用同一基类型的变量；

只能参与加，减，自加减，关联，赋值运算。

使用指针变量之前一定要将其进行初始化，不允许使用未初始化的指针变量
直接寻址&间接寻址*指针
指针做函数的参数的时候是模拟引用调用可以改数字
保持指针的三条原则，才能前行指向哪里，内容是啥，不使用未初始化的变量