1.二维数组

1》定义：数据类型 数组名［数组的数量］［子数组的长度］

2》如：int arr[3][4] ={ {1,2,3,4}, {1,2,3,4}, {1,2,3,4} }

3》赋值：同数组

4》特点：

与数组一样，但传递子数组时传递的是地址

子数组数量控制行数，单个子数组长度控制列数

打印数组：

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

```
for (int i=0; i<3; i++) {
    for (int j=0; j<4; j++) {
        printf("%d\t",arr1[i][j])
    }

}
```

每个子数组类型长度一致

2》练习：打印正三角

```
#include <stdio.h>
int main()
{
    int arr[4][7] = {{0,0,0,1,0,0,0},{0,0,1,1,1,0,0},{0,1,1,1,1,1
    ,0},{1,1,1,1,1,1,1,}};
    for (int i = 0; i< 4; i++) {
        for (int j = 0; j < 7; j++) {
            if (arr[i][j] == 0) {
                printf(" ");
            }else{
                printf("*");
            }

        }
        printf("\n");
    }
    return 0;
}
```

2.字符串

1》概念：以\0结尾的字符数组

2》定义：char arr[]={'a','b','c','d','\0'};或者char arr[]="abcd";

3》输出：输出字符串变量：printf("%s\n,"s);

输出字符串常亮：printf("%s\n,"&arr[0]);

4》赋值：strcoy（字符变量名，“字符串”）；（#include <string.h>）

5》特点：

.后面必须有'\0'结尾.否则只算普通的字符数组. 但'\0'不会输出,只表示字符串结束.

.字符串输出占位用%s必须遇到\0.才能结束,否则会继续输出更高位地址值的字符.

.strlen函数用于计算一个字符串的长度(字符数量),使用必须引入<string.h>.

. strlen不会计算\0. 且碰到\0结束,但是sizeof不受\0影响,且长度会包含\0.

.字符串一定是字符数组.但字符数组不一定是字符串.

3.指针

1》概念：存地址的数据类型

2》定义：类型 *指针名

3》赋值：指针变量名＝指针（地址）

4》特点：

.指针本身也有地址.指针定义之后,自身地址不变,但里面的存的地址可以变化.
.&p表示指针自身地址,而p表示指针存的地址 *p表示指针所存地址的内存空间.
.指针不初始化赋值不要使用.未赋值前地址是随机的.未赋值不可以用*p.
*修饰什么,就表示是什么的指针.判断原则:先确定变量的类型.(根据符号优先级)
指针是什么类型的,就应该存什么类型的地址.否则*p取对应内存空间的值会出错.

4.指针与数组

1》定义

```
//定义:固定写法.一定要加括号,不然变量p先与[]结合.成数组了.
int (*p)[3];
```

2》特点

.数组名表示第一个元素的地址,&数组名表示整个数组的地址.二者值相同,但意义不同.
.指针可以运算.但仅限于加减运算.运算的实质就是地址运算.即地址的移动.
.地址(+1)向高位移动,而(-1)向低位移动.不同指针类型移动一位(+1或-1)的字节数不同.
.实际移动的字节数看指针类型,类型占多少个字节,一位就移动多少个字节.

3》指针数组：

```
int a=2,b=1,c=0;

int *pa=&a;
int *pb=&b;
int *pc=&c;

int *arr[3];

arr[0]=pc;
arr[1]=pb;
arr[2]=pa;
```

4》数组指针

```
int intArr[3]={1,2,3};
int (*p)[3]=&intArr;
```

5》数组元素的指针

```
//注意这种是错误的.这个int d会重新在内存中开辟
int d=intArr[0];//这只是表示把数组第一个元素
int *pf=&d;//取的是新的变量d地址.这时pf表示d

int *pArr=&intArr[0];//或 int *pArr=intA

int *pArr1=&intArr[1];

int *pArr2=&intArr[2];
```