数据类型：空间大小

动态、静态：生命周期（存储时间）和作用范围（变量名何时可见）

作用范围不在了，但是生命周期在，可以用指针访问

跨界访问！

身份证、钱包的关系

二维数组、字符数组（指针、指针变量）、指针数组、函数入口地址、返回指针的函数

一维数组int a[4]，数组首地址a=&a[0]，第i（0开始）个元素a+i=&a[i]

第i个元素值a[i]=\*(a+i)

二维数组a：

a代表二维数组首地址（等效首行）

行指针a+i表述第i行首地址a+i=&a[i]

元素指针：\*a[0][0]第一元素地址，&a[i][j]=a[i]+j

\*&为逆运算！

行指针：把a看成一维数组，a[i]=\*(a+i)（数组），a+i=&a[i]（指针）

区分元素指针和行指针！类型不一样

区分指针和内容！

a=a[0]=&a[0][0]但是类型不同

元素指针：a[i]看为一维数组，则代表首地址和首元素地址a[i]=&a[i][0]

a[0]+i=a[0][i]

表示方法：（三种）

&a[i][j]=a[i]+j（混合法）=\*(a+i)+j（指针法）（数组名就是指针！地址类型转化！）

a[i][j]=\*(a[i]+j)=\*(\*(a+i)+j)（下标法）

在总数组中的地址，或行数组中的地址，或行号列号对应的地址

**+i要确定i指的是多长，自动转换类型！**

a[i]=&a[i]，但是意义不同

int\*p;p+1里1转化为int\*的类型，即4

指向元素的指针变量：和一般普通指针变量相同

行指针变量：类型标识符 (\*变量名)[列长]

如：int a[4][3], (\*p)[3];p=a;

数组除了函数传递参数，都是按元素使用

行指针的基类型：int\*[3]

\*(\*(p+i)+j)为第i行第j列的值

%p：输出指针值

行指针储存的是该行的地址

指针变量保存字符串，存储方式完全不同

char \*string=”I love China”;保存的是首地址，在代码区里保存了字符串常量

print (“%s”,string);输出字符串（接地址）

%c接值，逐个输出

内存分配：静态（电脑自动）、动态（malloc）

malloc.h：memory and location内存分配

动态内存分配malloc(长度);内存申请——分配连续空间，返回首地址，保存到指针里

ptr1=(char\*)malloc(20);（20指20位，如果int就是5个）

malloc申请的空间生命期到free(地址);结束，需要主动释放！

不释放，填满了就死机了

gets(指针);和字符串名使用相同

定义指针变量一定要有指向！

字符数组和字符指针不一样！

字符数组是每个字符保存到一个区域，不能str[]=””，但是字符指针可以

字符指针是在代码区存储字符串，数据区存一个地址值

char str自动分配了空间，但是用字符串指针只有指针没有位置，需要malloc

指针数组：

每个元素都是指针变量

用指针数组存储多行，相较于二维数组，可以节省空间（不必等长了）

数组名：二级指针！

指向指针变量的指针变量——n级指针

char \*\*p;p=name+2;name也是二级指针

行指针不是二级指针！是一级指针！只是表达式长得像

main可以带形参！

如：int argc（参数个数）, char \*argv[]（字符型指针数组矢量），写法无所谓

main是C程序的入口，操作系统调用，在控制台执行！

在命令提示符窗口（提示符：路径>程序名称，输入实参，回车）

命令行输入实参！！！

命令名（程序名） 参数1 参数2 ……

int argc自动统计参数量，表征argv长度

argv指向每一个参数

argv[0]指向程序名（默认.exe不用写）

函数放在代码区，函数名代表入口地址（第一条指令）

不知道函数名但是知道函数入口地址（函数指针），利用指针调用

函数返回值类型 (\*指针变量名) (形参类型);

int可以忽略，但是不建议

赋值：指针变量名=函数名;

调用指针：（旧版本）(\*指针变量名) (实参列表); （新版本）指针变量名 (实参列表);

空指针：不指向任何变量的指针，不是野指针！

指针是难点！易考！

看指针变量使用总结

main不写参数？