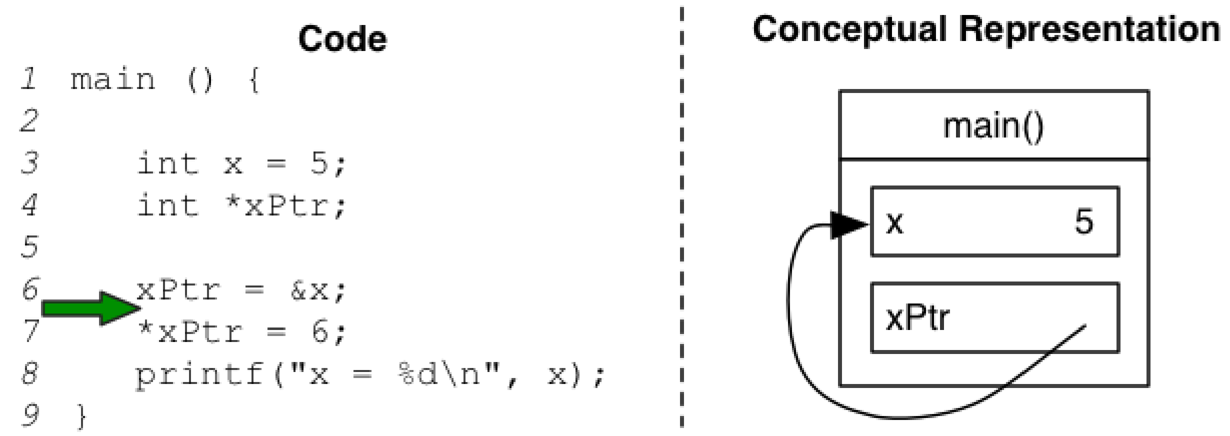
Pictures

/run/media/lijunjie/资料/Learning Video/pointers-arrays-recursion/01_pointers/02_pointers-conceptually/02_pictures_instructions.html

When working with pointers, *always draw pictures* **.** Although there are only three basic pointer actions—declaring, assigning (including initializing), and dereferencing—trying to read code that does all of these actions—often multiple times within a single line—can be very confusing. Until one can successfully read code that uses pointers, it will be almost impossible to write code that does so. Furthermore, when you are writing code with pointers, carefully planning—by drawing pictures—is even more important in order to write correct code.

Pointers are variables and should be drawn like any other variable at the time of declaration. Draw a box with the name of the variable on the left and the value of the variable on the right. When a pointer has been declared but not yet initialized, it is not known where the pointer points. The uninitialized value can be represented with a question mark just as it is for uninitialized variables of other types.

Assignment statements in C (*x = y;*) can be broken down into two parts. The left-hand side (here, *x*) is called the *lvalue*. This is the box that will have a new value placed inside it. Before this chapter, lvalues were simply variables (*e.g.* , *x* or *y*). With the introduction of pointers, we can also use the a derferenced pointer as an lvalue—if *p* is a pointer, then **p* is an lvalue, as it names the box at the end of p's arrow. For example, line 6 in the figure below assigns a value to the variable *xPtr* and line 7 assigns a value to the location that xPtr points to.



```
1 main () {
2
3     int x = 5;
4     int *xPtr;
5
6     xPtr = &x;
7     *xPtr = 6;
8     printf("x = %d\n", x);
9 }
```

The right-hand side (the *y* in the statement *x* = *y* ;) is called the *rvalue* . This is the value that will be placed inside the box/lvalue. With the introduction of pointers, we add two new types of expressions that can appear in rvalues: the address of an lvalue (*&x*), and dereferencing a pointer (**p*).