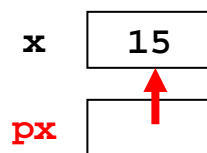


Summary:

1. In a **box-and-pointer diagram**, ordinary variables have a **box** associated with them, depicting the place in memory where the variable's value is stored. We draw the variable's value inside the box.



2. **Pointer** variables also have a box associated with them. However, pointer variables have a memory location as their value. Hence, we don't put a value inside the box for a pointer variable; instead, we draw an **arrow** from that box to the box at the location specified by the pointer. That is, we draw an arrow from the pointer to its **pointee**.



3. **Notation** for pointers:

- We declare pointer variables by appending an asterisk to their type:

```
double* px;
```

- For any variable `x`, the notation `&x` means the address (i.e., location) of variable `x`:

```
double x = 15;
double* px;
px = &x;
```

← *Note how pointer variables are declared:*

← *Establishes the pointer's pointee.*

- We refer to a pointer's **pointee** by the notation `*px`.

For example, the following statement (continuing the example above) increments variable `x`, since `x` is `px`'s pointee.

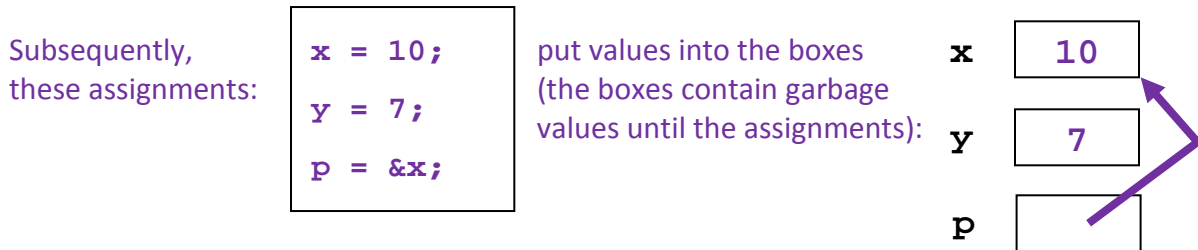
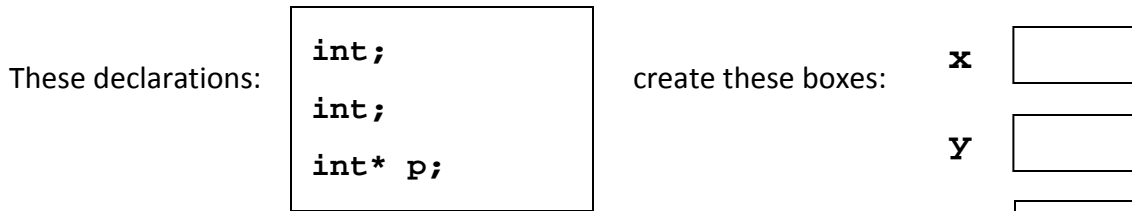
```
*px = *px + 1;
```

Don't be confused by these two uses of asterisk: one to declare that a variable is a pointer, and the other to refer to the pointer's pointee (which we call **dereferencing**).

4. **Space for variables is allocated** in several ways, including:

- **Declaring a local variable** creates a box for the variable.
- **Calling a function** creates boxes for each of the **parameters** of the function. The initial values of those boxes are copies of the boxes of the corresponding **actual arguments** in the function call.

Here is an example (on the next page):



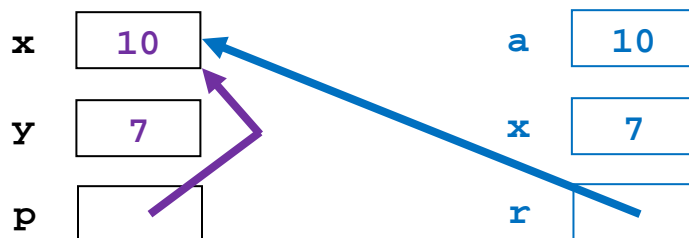
Now suppose there is a function *foo* whose prototype is as follows:

```
void foo(int a, int x, int* r) {  
    ...  
}
```

Continuing the example, this function call:

```
foo(x, y, p);
```

creates and initializes the additional boxes show below in blue.



The boxes created by the function call are new boxes; the variable called *x* in function *foo* has nothing to do with the variable called *x* in the calling code.

The new boxes are initialized by copying the values from the caller's boxes. **Copying a pointer's value means creating a new arrow that points to the same place as the old arrow.**