

- In C, there is a *variable type* for storing an address: a *pointer*
- Contents of a pointer is an unsigned (0+, positive numbers) memory address
- When the Rside of a variable contains a memory address, (it evaluates to an address) the variable is called a pointer variable
- A pointer is defined by placing a *star* (or *asterisk*) (\*) before the identifier (name)
- Be careful when defining multiple pointers on the same line:  

```
int * p1, p2;                vs                int *p1, *p2;
```
- Pointers are typed! Why?  
 The compiler needs the size (sizeof()) of the data you are pointing at (number of bytes to access)
- The \* is part of the definition of p and is not part of the variable name
  - The name of the variable is simply p, not \*p
- Pointer variables all use the same amount of memory no matter what they point at

## & operator

- Unary **address operator** (&) produces the **address** of where an identifier is in memory
  - *Tip:* printf() format specifier to display an address/pointer (in hex) is "%p"
- Requirement: identifier must have a Lvalue
  - Cannot be used with constants (e.g., 12) or expressions (e.g., x + y)

### The *indirection operator* (\*)

- Also called the *dereference operator to a variable* is the inverse of the *address operator* (&)
- address operator (&) can be thought of as:  
*“get the address of this box”*
- **indirection operator (\*)** can be thought of as:  
*“follow the arrow to the next box and get its contents”*

```
int i = 0x1234babe;

int *p;

p = &i;

p = NULL;

int j = 0xbeefface;

p = &j;

i = 1 + *p;

j = *(&p);

*p = i + j;
```

