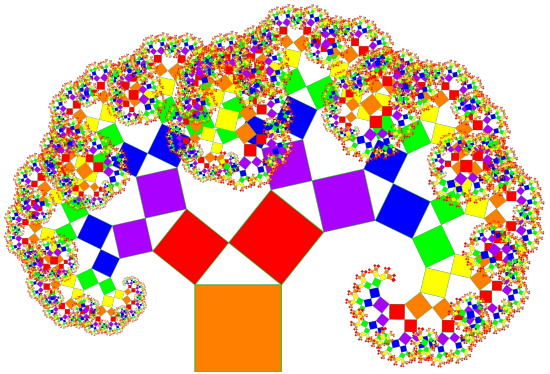
Pointers

Function Arguments (Review)

- The contents of an actual parameter are *copied* into the function parameters
  - Two variables = two different memory locations
  - Have NO RELATION to each other at this point
  - Remember: One variable refers to one (and only one) memory address

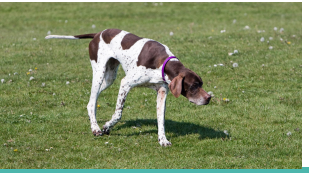
Function Arguments (Review)

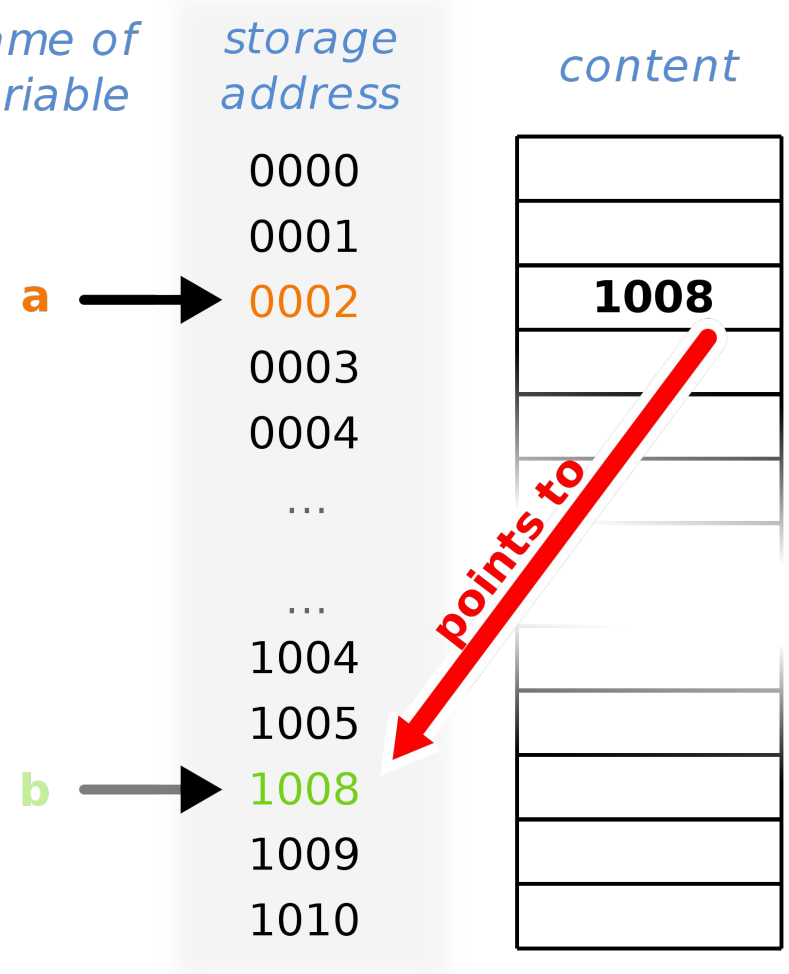
- The contents of an actual parameter are *copied* into the function parameters
    - Remember: One variable refers to one (and only one) memory address



Pointers

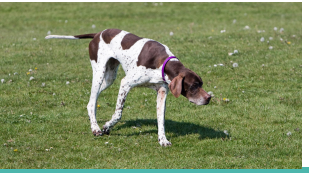
- Pointer = a variable that stores the *memory location* of another variable

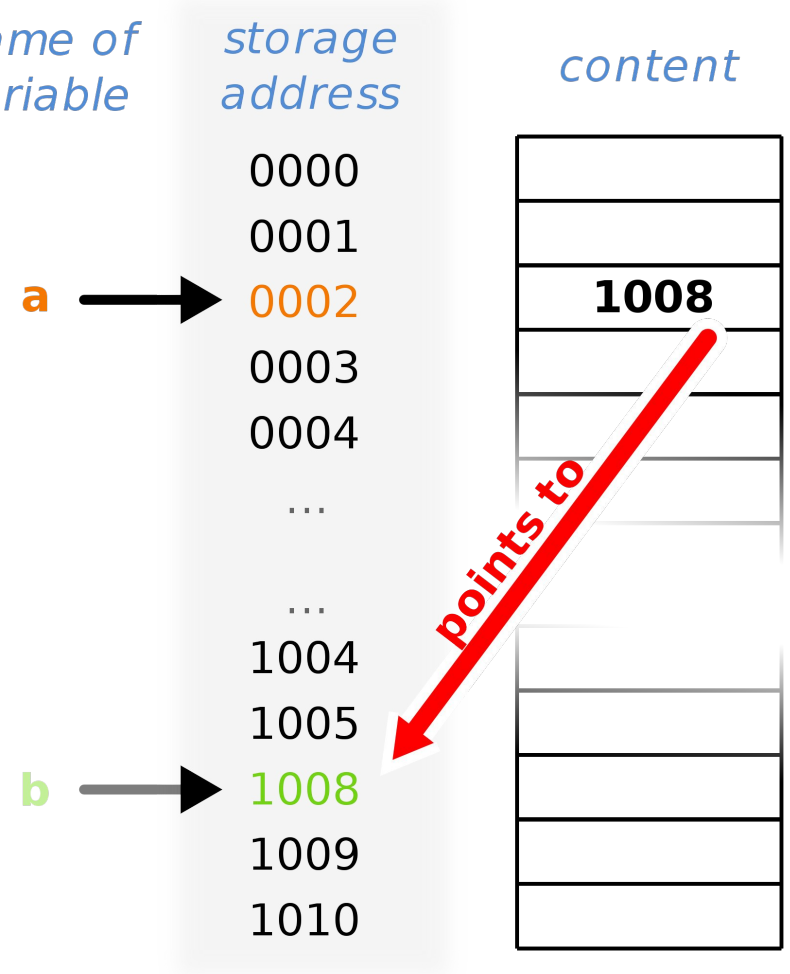




Pointers

- Allows variables to modify each other (without breaking the one-variable-refers-to-1-memory-address rule)

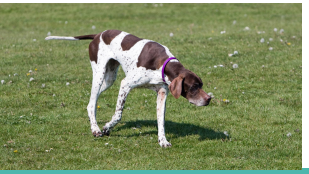




Pointers

- Memory locations are accessed through **&**

- Scanf(“%d”, &num1);
  - The number scanned in is placed into the memory location given by **&num1**



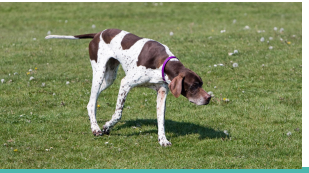
Pointers

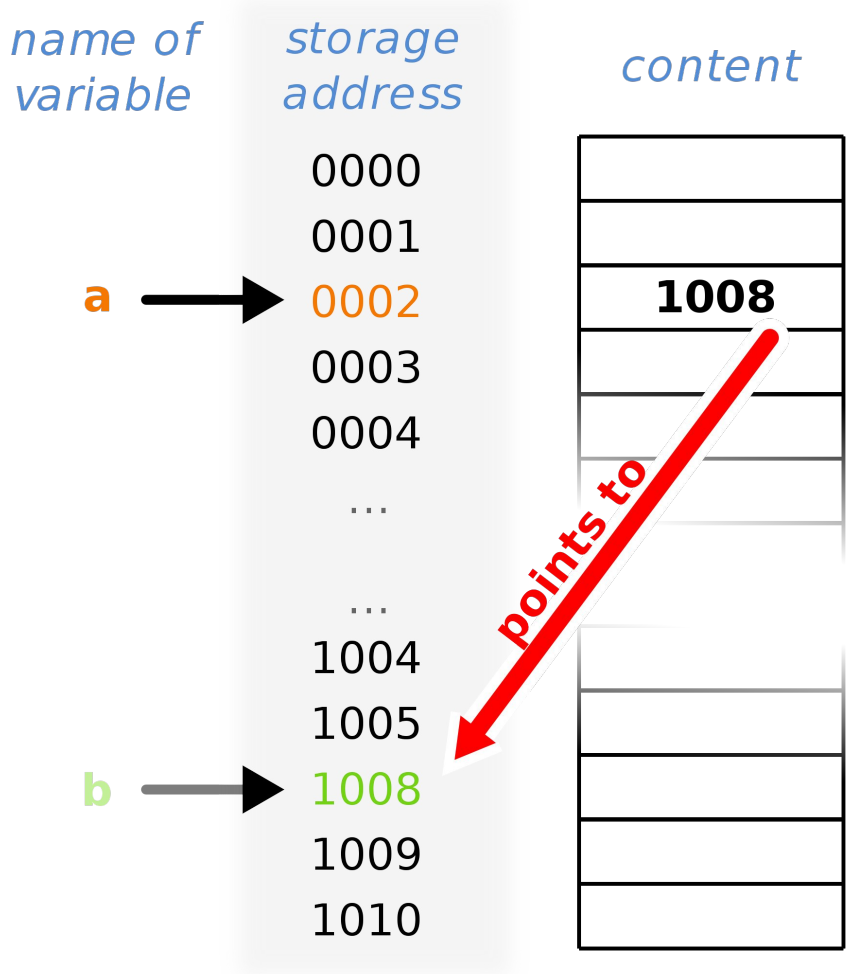
- Code:

```
int b;

int a = &b;

/*Now, a holds the memory
address of b*/
```

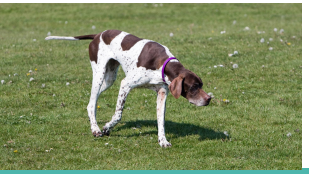




Pointers

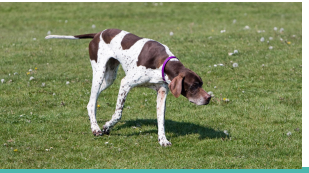
- Memory locations are accessed through **&**
- Memory locations are printed using **%p** along with **&**

```
int num1 = 13;
printf("The memory location of num1 is: %p",
&num1);
```



Pointers

- Values at specific memory addresses can be accessed using *****
  - NOT multiplication!
  - Is placed directly before a variable (no spaces in between)
  - *Examples: *b, *memory, *football*



Pointers

- More specifically, the object that a variable points to can be *referenced* by *

```
int a = 5;
int b = &a; /*Holds the memory address of a*/

*b += 1;      /*Finds the value at the memory address
referenced by b (5), then adds one to that value*/

printf(“%d”, a); /*Should print the value 6*/
```