Pointers

* ⟶ &

09/28/2016

EECS 338

Making a pointer

```
int * x;
```

- Name the variable 'x'
- Create a variable of type 'int *'

x is now a "pointer"

x is an address that points to somewhere in memory

currently, x has not been given a value, so it points to ???. It may be a "null pointer", or it may point to garbage

```
int y;
```

- name the variable 'y'
- Create a variable of type 'int'

y is now an integer.

Like all variables, including 'x', it has a location in memory that it exists in.

To give x a value:

```
x = &y;
```

- variable to get address of
- "get the address of"
- set x to the address of y

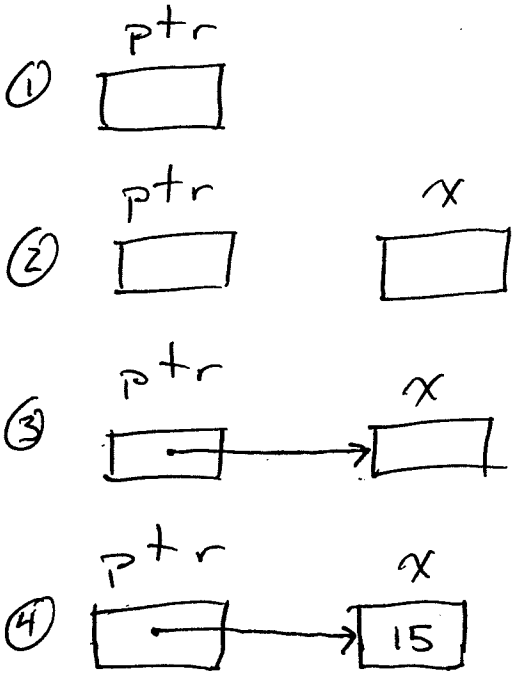
① `int *ptr;` // declare pointer

② `int x;` // declare variable

③ `ptr = &x;` // set pointer to be the address of x

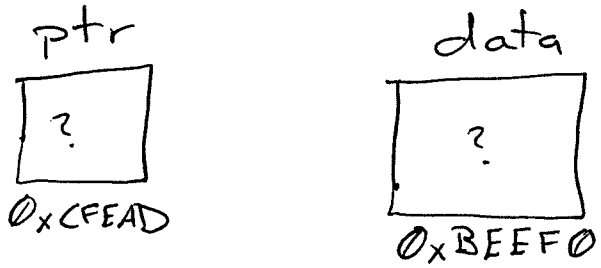
④ `x = 15;` // give the variable a value

ptr is still the same. The address of the data of x

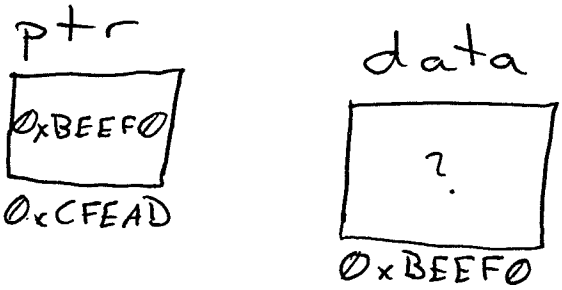


Dereferencing a pointer

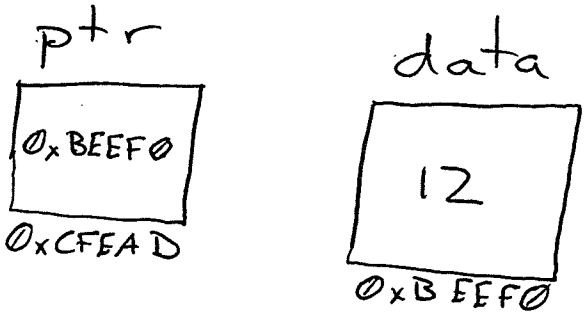
```
int *ptr;
int data;
```



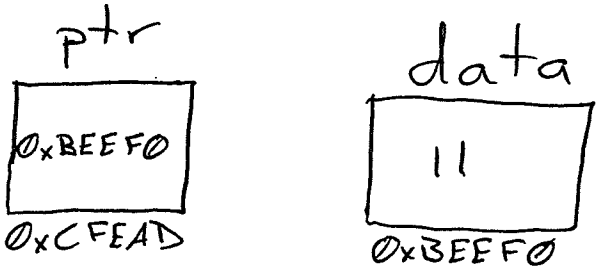

```
ptr = &data;
```




```
data = 12;
```

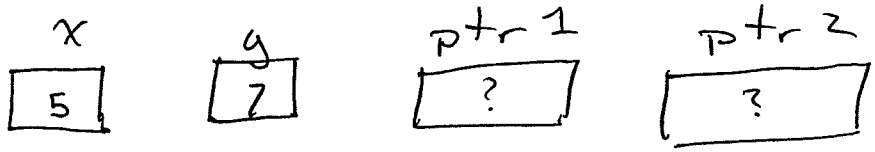



```
*ptr = 11;
```

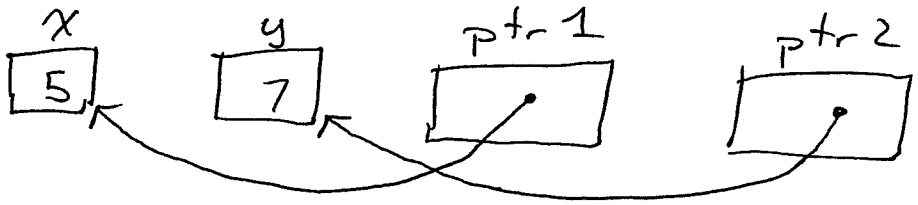


Playing with multiple pointers

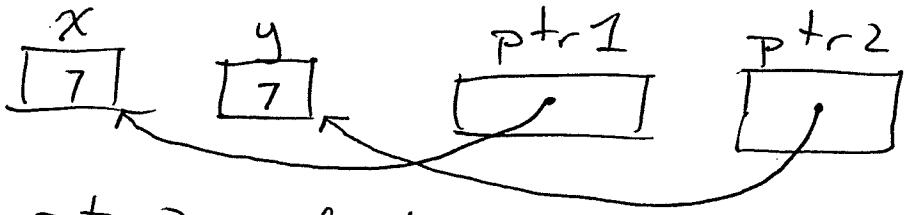
```
int x = 5, y = 7;
int *ptr1;
int *ptr2;
```



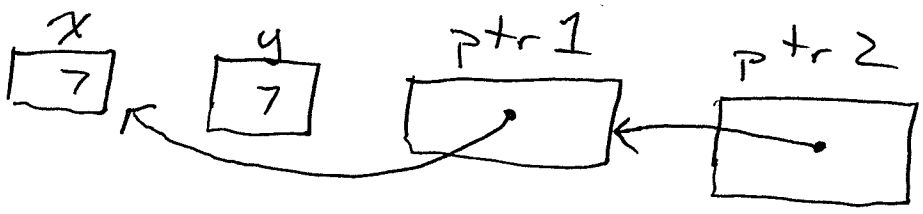

```
ptr1 = &x;
ptr2 = &y;
```




```
*ptr1 = *ptr2;
```

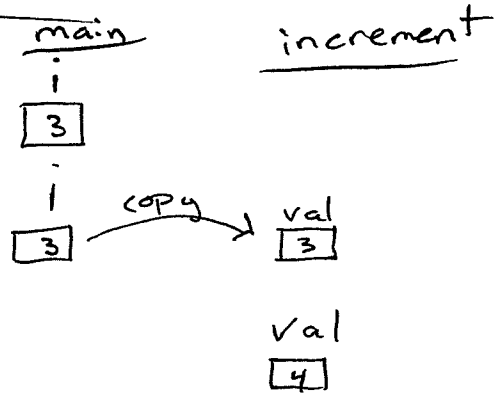



```
ptr2 = &ptr1;
```



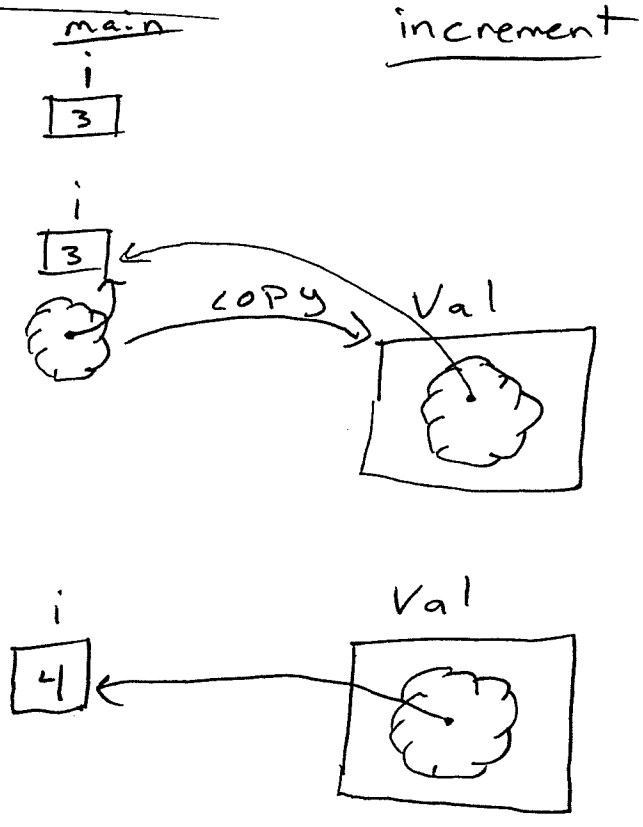
Pass By Value

```
void increment(int i);
main() {
    int i = 3;
    increment(i);
    //what is 'i'?
}
void increment (int val) {
    val++;
}
```



Pass By Reference

```
Void increment (int* i);
main() {
    int i = 3;
    increment(&i);
    //what is 'i'?
}
void increment (int* val){
    (*val)++;
}
```



Memory Scoping

```
#include <stdlib.h>
#include <stdio.h>

int * create_memory();

void main(){
    int *x = create_memory();
    sleep(1); //allow things to settle
    printf("%d\n", *x);
    fflush(stdout);
    //what is printed?
}

int * create_memory(){
    int val = 2;
    printf("set val to %d\n", val);
    int * ret_var = &val;
    printf("pointer created with value %d\n", *ret_var);
    return ret_var;
}
```

Memory Scoping Solved

```
#include <stdlib.h>
#include <stdio.h>
int * create_memory();
void main(){
    int *x = create_memory();
    sleep(1); //allow things to settle
    printf("%d/n", *x);
    fflush(stdout);
    //what is printed?
    free(x);
}

int * create_memory(){
    int * ret_var = (int*) malloc(sizeof(int));
    printf("memory created\n");
    *ret_var = 2;
    printf("memory initialized to %d\n", *ret_var);
    return ret_var;
}
```