Computer Science 5C
Chapter 8-- Pointers

Dr. Scott A. Brandt
Professor, Computer Science Department
Director, UCSC/LANL Institute for Scalable Scientific Data Management

CMPS 5C
Recall: parameters to functions

• Suppose we have a function foo() defined like this:
  • int add (int a, int b);
  • foo() has two parameters of type int and returns an int
  • When foo() is called, a and b are copies of the parameters passed to foo()
This doesn’t work!

• void swap(int a, int b) {
    int temp;
    temp = a;
    a = b;
    b = temp;
}
• int x = 5; y = 7;
• swap(x, y);
• x is still 5 and y is still 7
Storing data in memory

• Analogy: “My office”
  • When I use that name, I am referring to a location on campus
• Similarly, declaring a variable = naming a memory location
• Every variable is stored somewhere in memory
• The computer interprets the name of a variable to refer to the data stored in the variable
Memory addresses

• My office has an address that uniquely identifies it
  • E2 347B

• Every memory location has an address
  • They are numbered from 0 to however many words of memory the computer contains

• We can access data by name or by address

• “Put that in my office” vs “Put that in E2 347B”
  • Two ways of saying the same thing
Pointers

• * means “pointer to”
  • A pointer is a variable that stores the address of another variable

• Example: int i, *p;
  • i is an int
  • i can contain an integer
  • p is a pointer to an int
  • p can contain the address of an integer
Addresses

• & means “address of”
  • It can be used to get the address of any variable

• Example: int i, *p;
  • p = &i;
  • p contains the address of the variable i
Using pointers

• * also means “the thing pointed to by”
  • It can be used to access the thing pointed to by a pointer

• Example: int i, *p;
  • p = &i;
  • i = 7; /* sets i to 7 */
  • *p = 9; /* sets i to 9 */
Let’s draw some pictures

• Variables
• Addresses
• Pointers
• &
• *
This works!

• void swap(int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}

• int x = 5; y = 7;

• swap(&x, &y);

• x is now 7 and y is now 5
Why does that work?

- Remember: the parameters to a function are copies of the parameters passed in the function call
- A copy of a pointer is still a pointer
- A copy of a pointer points to the same thing as the original pointer
  - int i, *p, *q;
  - p = &i;
  - q = p;
  - *q = 7; /* sets i to 7 */