EASTERN WASHINGTON UNIVERSITY
start something big

Call by Reference
Call by Value

Computer Science Department
Eastern Washington University
Yun Tian (Tony) Ph.D.

Recall

- How to use pointers?
- Pointer Arithmetic

Today

- Pointer as function parameters
  - Call by reference
- Call by value

Formal parameters

- If a function is to use arguments, it must declare variables that accept the values of the arguments.
  - These variables are called the **formal parameters** of the function.
- The formal parameters behave like other **local variables** inside the function and are created upon entry into the function and destroyed upon exit.

Passing Parameters

- While calling a function, there are two ways that arguments can be passed to a function:
  - 1, Call By Value
    - This method copies the **actual value** of an argument into the formal parameter of the function.
    - In this case, changes made to the **parameter** inside the function have no effect on the argument.

Passing Parameters

- While calling a function, there are two ways that arguments can be passed to a function:
  - 1, Call By Value
    - This method **copies** the **actual value** of an argument into the formal parameter of the function.
    - In this case, changes made to the parameter inside the function have no effect on the argument.
    - By default, C programming language uses ***call by value*** method to pass arguments.
      - In general, this means that code within a function cannot alter the arguments used to call the function.

Example of Call by Value

demo callbyV.c

demo callbyV2.c

demo swap1.c

//Which shows that there is no change in the values though they had been changed inside the function.

Passing Parameters

- While calling a function, there are two ways that arguments can be passed to a function:
  - 2, Call By Reference
    - This method copies the **address** of an argument into the formal parameter.
    - Inside the function, the **address is used to access the actual argument** used in the call.
    - This means that changes made to the parameter **affect the argument.**

Passing Parameters

- While calling a function, there are two ways that arguments can be passed to a function:
  - 2, Call By Reference in **swap2.c**
  - When calling a function, make sure your function call follows the signature of the function prototype.
  - E.g void swap2(int *, int *) // returns void, both formal parameters are supposed to be int pointer variables(pointer holds an **address** of an integer).

Passing Parameters

- While calling a function, there are two ways that arguments can be passed to a function:
  - 2, Call By Reference
  - Therefore, when you call swap(), you should do swap2(&a, &b);
  - Here, we pass in the address of variable **a** and **b** as actual parameter.
  - Inside swap2(), instead of changing the formal parameter x or y, we changed (*x) and (*y). That will affect the **actual parameter** passed in.

Passing Parameters

- Demo of swap2.c
- Demo of callbyBoth.c

Summary of Call by Value and Call Reference

- If we change a variable itself inside a function we cannot see the effect outside of the function.(including pointer variables)
- If we change what a variable points to (if applicable), we could see the effect outside of the function.
  - Compare swap1.c and swap2.c

Use Pointers

- **IMPORTANT NOTE**
- int a[] = { 4, 5, 6, 7};
- int *ptr = a;
- In C, when accessing array element in **a,**
- ***(ptr + i) and ptr[i] do exactly the same thing,**
- **They can be used interchangeably.**
  - **They are the same thing.**

Use Pointers

- **Even under all other scenarios, when ptr is used for other purposes, in C,**
- ***(ptr + i) and ptr[i] are the same thing.**
- **A pointer variable acts exactly like an array name.**
  - **Where a function needs array as a parameter, you can pass in a pointer that has been initialized.**
  - **and vice versa.**
  - **They can be used interchangeably.**
  - **You can consider they are the same thing.**

Summary

- Call function by Value
- Call function by reference
- *(ptr +i) and ptr[i]

Next Class

- Pass pointer as function parameters
- Return pointer from inside a function