CS 161, Lecture 14: Different Ways to Pass Parameters

How We Have Been Passing -> By Value

- Also referred to as Call by Value
- Copies the value into the formal parameter

```
void swap (int a, int b) {
        int temp = a;
        a = b;
        b = temp;
}
int main () {
        int a = 1, b = 2;
        swap(a, b);
        cout << “a: “ << a << “b: “ << b << endl;
}
```

Alternate: Pass By Reference

- Takes both the value and the address of the passed in variable
- Does not exist in C
- References can’t be null

```
void swap (int & a, int & b) {
        int temp = a;
        a = b;
        b = temp;
}
int main () {
        int a = 1, b = 2;
        swap(a, b);
        cout << “a: “ << a << “b: “ << b << endl;
}
```

Alternate: Pass By Pointer

- Pointer is a memory address
- Can be changed to hold different memory addresses
- Pointers need to be dereferenced to get to the value stored at that address

```
void swap (int* a, int* b) {
        int temp = *a;
        *a = *b;
        *b = temp;
}
int main () {
        int a = 1, b = 2;
        swap(&a, &b);
        cout << “a: “ << a << “b: “ << b << endl;
}
```

Pointer Cheat Sheet

- *
  - If used in declaration (which includes function parameters), it creates the pointer
    - Ex: int *p; // p will hold an address to where an int is stored
  - If used outside a declaration, it dereferences the pointer
    - Ex: *p = 3; //goes to the address stored in p and stores a value
    - Ex: cout << *p; //goes to the address stored in p and fetches the value
- &
  - If used in a declaration (which includes function parameters), it creates and initializes the reference
    - Ex: void fun(int &p); //p will refer to an argument that is an int by implicitly using *p (dereference) for p
    - Ex: int &p = a; //p will refer to an int, a, by implicitly, using *p for p
  - If used outside a declaration, it means “address of”
    - Ex: p=&a; //fetches the address of a (only used as rvalue) and store the address in p

Demo