CS162: Introduction to Computer Science II

References

Pass-by-Value

```c
void swap(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}
```

- Consider the swap function above, which is intended to swap the values of x and y
- We will use it in the next slide
Pass-by-Value

```c
int main() {
    int a = 10;
    int b = 20;
    swap(a, b);
    std::cout << "a = " << a << " b = " << b << std::endl;
}
```

- What gets printed out?
  a = 10 b = 20
- What happened?

---

Pass-by-Value

```c
void swap(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}
```

- When `swap(a, b)` is called:
  - The value of `a` is copied into local variable `x`
  - The value of `b` is copied into local variable `y`
  - The swap function executes (local variables `x` and `y` are swapped)
  - Nothing happens to the original values of `a` and `b`
Pass-by-Value

• **Pass-by-value**: the parameters of a function hold copies of the values sent to them

• **Pass-by-reference**: the parameters of a function hold the actual memory locations of the values sent to them

```c
void swap(int& x, int& y) {
    int temp = x;
    x = y;
    y = temp;
}
```

• If you changed the swap function to pass by reference, you need to pass in a reference to x and a reference to y
Pass-by-Reference

- A reference variable has an \& between the data type and the variable
- Eg. x and y are reference variables below

```cpp
// Pass by reference version
void swap(int& x, int& y) {
    int temp = x;
    x = y;
    y = temp;
}
```

```cpp
int main() {
    int a = 10;
    int b = 20;
    swap(a,b);
    std::cout << "a = " << a << " b = " << b << std::endl;
}
```

- If you run the pass-by-reference version of swap, you now get: a = 20, b = 10
Pass-by-Reference

• You can also accomplish the same thing as pass-by-reference if you translate everything into pointers

    // Pointer version
    void swap(int* x, int* y) {
        int temp = *x;
        *x = *y;
        *y = temp;
    }

    int main() {
        int a = 10;
        int b = 20;
        // This is the pointer version of swap
        swap(&a,&b);
        std::cout << "a = " << a << " b = " << b << std::endl;
    }

Pass-by-Reference

• And for the pointer version, you also need to call swap with the address of a and b

    int main() {
        int a = 10;
        int b = 20;
        // This is the pointer version of swap
        swap(&a,&b);
        std::cout << "a = " << a << " b = " << b << std::endl;
    }
Const

• If a reference variable is declared const, it means the reference contents cannot be changed
• eg. the function foo doesn’t change the variable x

```c
// This is a silly function but it’s just
// an illustration
int foo(const int& x) {
    return x*x;
}
```

Const

• If we make x and y const references in the swap function, it will not compile
• This is because x and y are modified by the swap function

```c
// Won’t compile
void swap(const int& x, const int& y) {
    int temp = x;
    x = y;
    y = temp;
}
```