Pointers

Queens College, CUNY
CS211 Lab
Instructor: Kent Chin
In Real Life

Business Listing

Cheese Shop
Address: 12-34 Whatever St.

The Place Itself

visit
Pointers

Pointers are data types which store memory addresses. We visit the location the memory address refers to by dereferencing the pointer.

```c
int i = 10;

int *ip; // declaring a pointer to an integer
ip = &i; // storing address of i (an int variable) into pointer ip
*ip = 5; // dereference ip (get/visit variable i)
```

**Notes**

- When the * operator is part of a variable declaration, it means the variable is a pointer; elsewhere, when it is on the left-side of a pointer variable, it serves as the dereference operator
- The & operator is the “address of” operator
### Pointer Examples

```c
int x = 5;
int *xptr = &x; //xptr is a pointer to an int variable
int *yptr, y; //yptr is a int pointer, y is a “regular” int
yptr = &y; //yptr stores the address of y
*yptr = 3; //yptr dereferenced gives you variable y itself

cout << x << endl;       //5
cout << xptr << endl;    //prints address stored in xptr
cout << &x << endl;      //prints address of x (same output as above)
cout << *xptr << endl;   //5 (dereferencing xptr gives you variable x)

cout << y << endl;       //3
cout << yptr << endl;    //prints address stored in yptr
cout << &y << endl;      //prints address of y (same output as above)
cout << *yptr << endl;   //3 (dereferencing yptr gives you variable y)

//xptr and yptr get dereferenced first, then multiplied
int product = *xptr * *yptr; //product = 15
```
Pointers, like business listings, *store addresses*. To *visit* the location which the (memory) address refers to, we *dereference* the pointer.
Swap Function...with Pointers!

Your Memory

```c
int main() {
    int j = 1, k = 2;
    swap(&j, &k);
}
```

```c
void swap(int* a, int* b) {
    int temp = *a; //get variable which `a` points to and store value
    *a = *b; //get variables which `a` and `b` point to, reassign value
    *b = temp; //reassign value of variable which `b` points to
}
```

Variable *scope* is *irrelevant* when you have the *memory address* of a variable.