Pointers! Lecture 3

Pointer: Indirection to data (object)

Like Java references:

```
myVar = new Var();
```

```
int i = 12;
int *i_pointer = &i;
```
int *ip =

- ip is a pointer
- type of ip points to address of

*pointer

- dereference
  return value in address pointer
int i = 12;
int * cp = &i;
printf("%p", cp); \rightarrow 0x1
printf("%p", &i); \rightarrow 0x1
printf("%d", i); \rightarrow 12
printf("%d", *cp); \rightarrow 12

de reference

Tiffany mobile 678-521-1082
Mom home 931-455-7286
Dad mobile 931-247-82699
int c = 12;
float x = 1.7;
int z = 3;

int *pointer = &c;

x = (int)(13 / 3);

*pointer = 13;
c += 1;

*pointer = 7;
c += 1;

pointer = pointer + 1;

*pointer = 3;

float *p2 = 0x7;

*p2 = 1.7;
Allowed operations

int * px;
px = &i;
int * py;
py = px;
int i;
px = NULL;
if (px == NULL)
    error
else
    ;

NOT Allowed (or advised)

px = 0x123
pz = px + py
px = py * 3

Swap

void swap(int *a, int *b)

Value in addr a to hold what was in addr b + vice versa
```
void swap(int* a, int* b)
{
    int* tmp = b;
    b = a;
    a = tmp;
}
```
Arrays
- really just pointers
- consecutive elements in memory

int array[6] = 31, 15, 11, 0, 3, 73;

int *pointer = array;
array -> printf("%d", array);
\[ 0 \x100 \]

Array indexing
array[0] → 1
array[5] → 7
array[1000]