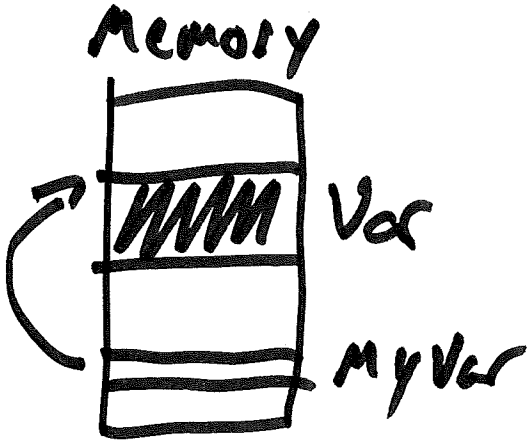
Pointers! Lecture 3

Pointer: indirection to data (object)

Like java references:

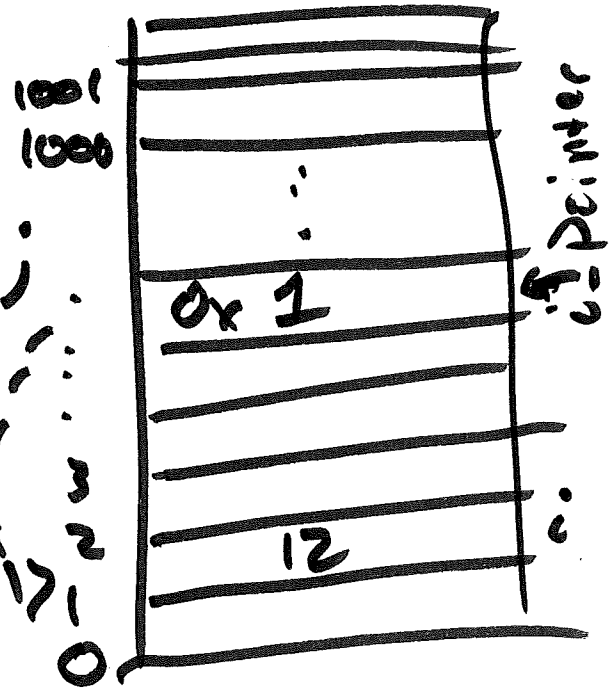
myVar = new Var();



---

int i = 12;

int *i_pointer = &i;



int *ip;

↳ ip is a pointer

↳ type ip points to

& → address of

*pointer

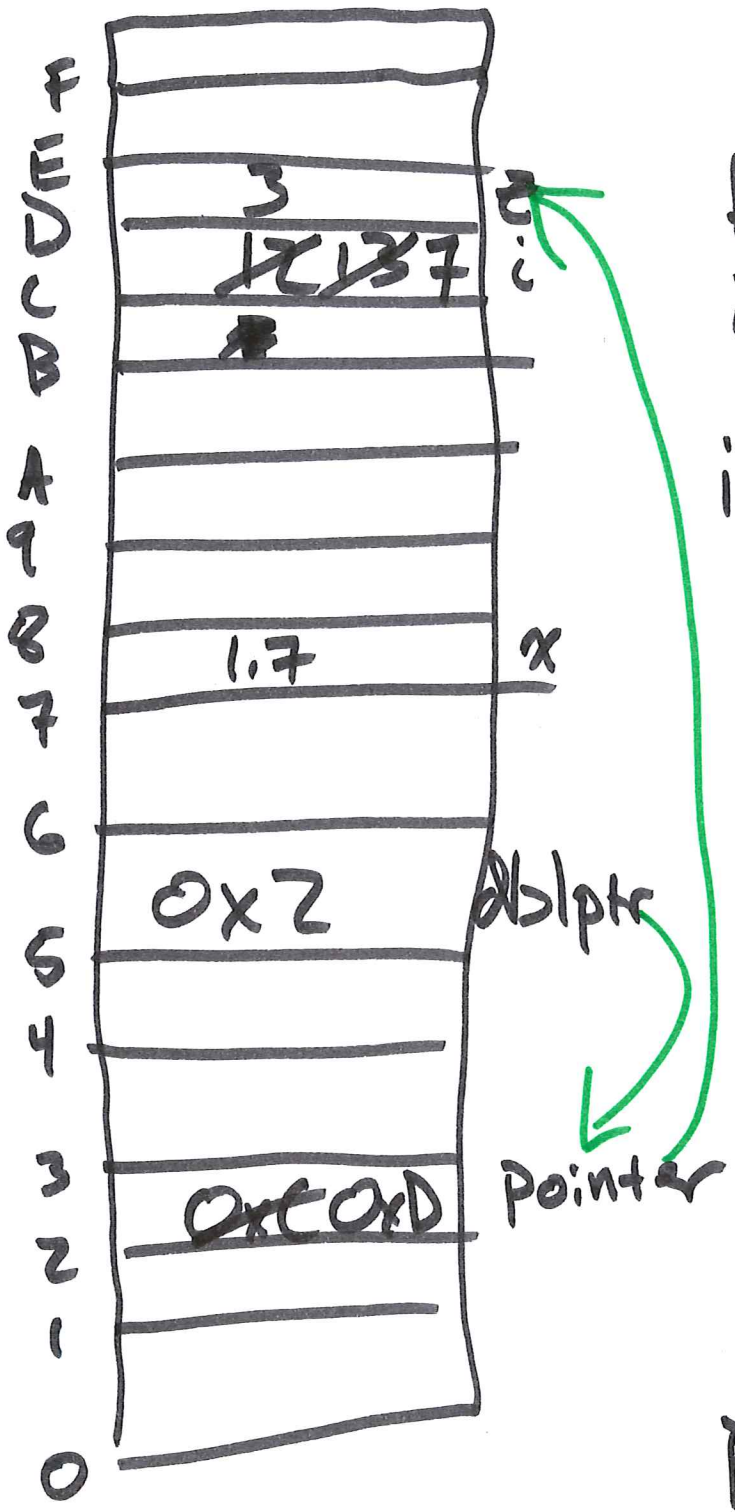
↳ dereference

return value in address pointer

```
int i = 12;
int *ip = &i;
printf("%p", ip); -> 0x1
print f("%p", &i); -> 0x1
printf("%d", i); -> 12
printf("%d", *ip); -> 12
```

dereference

Tiffany mobile 678-521-1082
Mom Home 931-[redacted]5-7286
Dad mobile 931-247-[redacted]2699




```
int i = 12;
float x = 1.7
int z = &3;

int * pointer = &i;

i = i + 1
```

12
13
3

```
* pointer ↦
```

```
i ↦ 13

*pointer = 7

i ↦ 7

* pointer ↦ 7

pointer = pointer + 1

*pointer ↦ 3

float * p2 = 0x7

*p2 ↦ 1.7
```

Allowed operations

```
int *px;
int *py;
int i;
int j;
```

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

NOT Allowed (or advised)

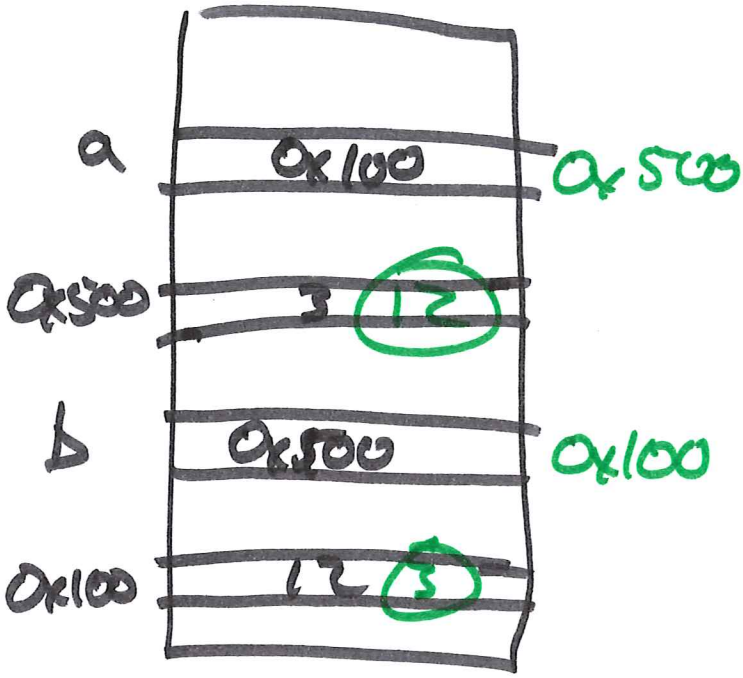
```
px = 0x123
pz = px + py
px = py * 3
```

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)

a)
```
int *tmp = b;
b = a
a = tmp
```



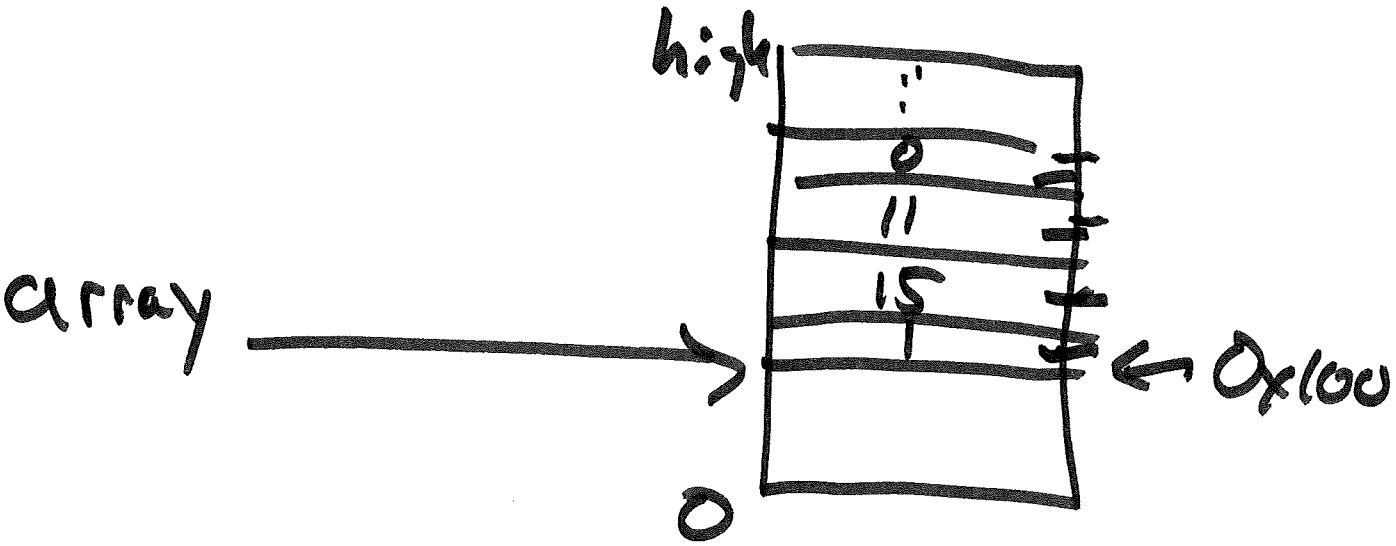
b)
```
~~int *tmp~~
int tmp = *b
b = a
*a = tmp
```

c)
```
int tmp = *b;
*b = *a;
*a = tmp;
```

Arrays

- really just pointers
- consecutive elements in memory

```
int array[6] = {1, 15, 11, 0, 3, 7};
```




```
int *pointer = array;
```

~~array~~ `printf("%p", array);`

↓

0x100

Array indexing

array[0] $\mapsto$ 1

array[5] $\mapsto$ 7

array[1000]

*array $\mapsto$ 1

*(array+3) $\mapsto$ 0