Introduction to C

4411 OS Practicum
Friday, September 4

Enumerated Types

- Values are consecutive integers starting from 0
  - unless you say otherwise…
- Not "advanced" just really important
- ***No magic numbers!***

```
enum month_t { JANUARY,
               FEBRUARY,
               MARCH
};
```

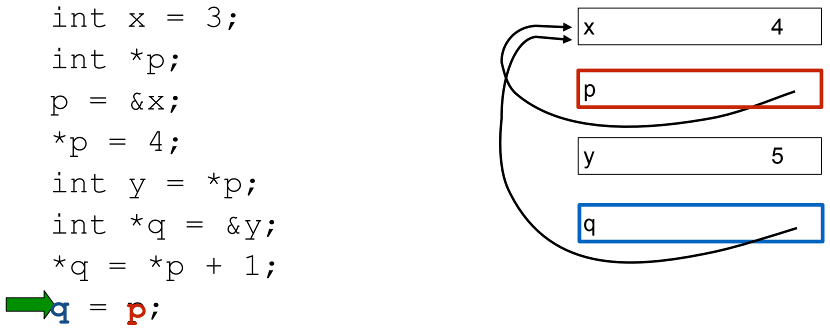
#define should also be in caps

```
#define MAX_PLAYERS   10
```

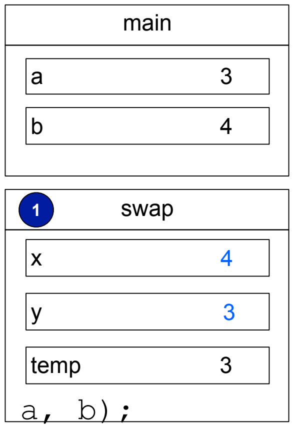
Pointers

- Simple pointer example
- Naïve swap
- Correct swap
- Deep vs. shallow copying — example on board

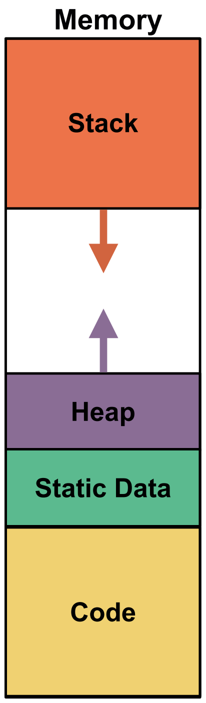
```
int x = 3;
int *p;
p = &x;
*p = 4;
int y = *p;
int *q = &y;
*q = *p + 1;
q = p;
```



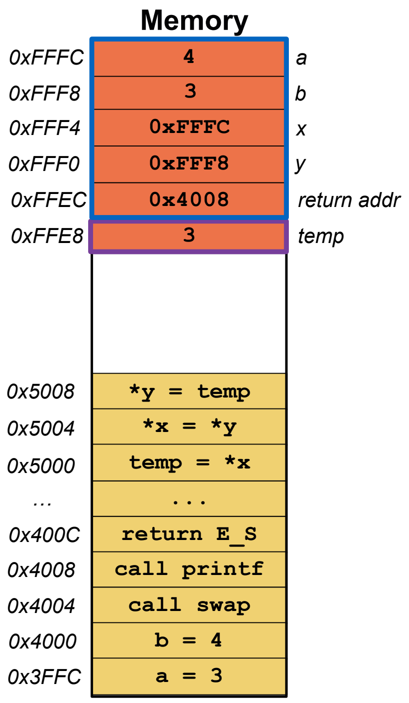
```
void swap(int x, int y) {
  int temp = x;
  x = y;
  y = temp;
}
int main(void) {
   int a = 3;
   int b = 4;
   swap (a, b);
   printf ("a = %d, b = %d\n", a, b);
   return EXIT_SUCCESS;
}
```



```
void swap(int *x, int *y){
  int temp = *x;
  *x = *y;
  *y = temp;
}
int main(void) {
   int a = 3;
   int b = 4;
   swap(&a, &b);
   printf("a=%d, b=%d\n", a, b);
   return EXIT_SUCCESS;
}
```



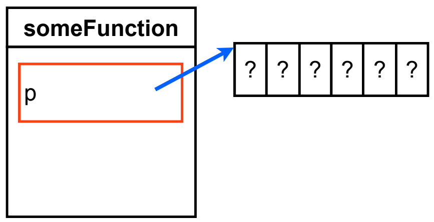
```
void swap(int *x, int *y){
  int temp = *x;
  *x = *y;
  *y = temp;
}
int main(void) {
   int a = 3;
   int b = 4;
   swap(&a, &b);
   printf("a=%d, b=%d\n", a, b);
   return EXIT_SUCCESS;
}
```



Dynamic Memory Allocation

- **Malloc**
- Free
  - Things that you can do wrong
  - For every malloc there must be an equal and opposite free!
- realloc

```
int * p;
p = malloc (6 * sizeof(*p));
```



Evaluates to pointer to that memory location

Complete assignment statement as usual

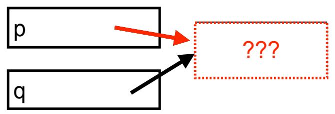
```
int main (void) {
  int x = 0;
  for (int i = 10; i < 100; i++) {
    int * p = malloc(i * sizeof(*p));
    x = doSomeComputation(x, i, p);
  }
  printf("Answer %d\n", x);
  return EXIT_SUCCESS;
}
```

Example without Free

```
int main (void) {
  int x = 0;
  for (int i = 10; i < 1000000; i++)
{
    int * p = malloc(i *
sizeof(*p));
    x = doSomeComputation(x, i, p);
    free (p);
  }
  printf("Answer %d\n", x);
  return EXIT_SUCCESS;
}
```

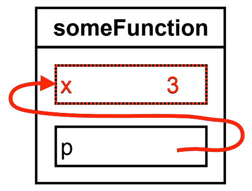
Example with Free

```
int * p = malloc(4 * sizeof(*p));
int * q = p;
…
free(q);
…
free(p);
```



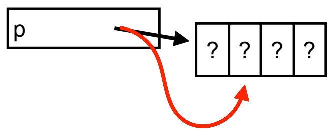
Double Free (don't do this!)

```
int x = 3;
int * p = &x;
…
free(p);
```



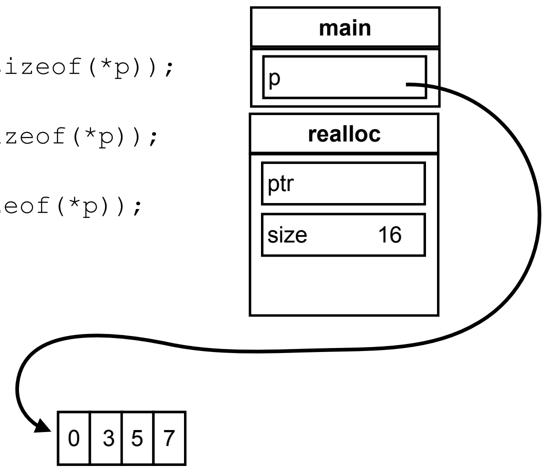
Free Memory Not In Heap (don’t do this!)

```
int * p = malloc(4 * sizeof(*p));
…
p++;
…
free(p);
```



Freeing Middle of Block (don’t do this!)

```
int main (void) {
 int *p = malloc(10 * sizeof(*p));
 readInputs(p);
 p = realloc(p, 14 * sizeof(*p));
 readMoreInputs(p);
 p = realloc(p, 4 * sizeof(*p));
 ...
 return EXIT_SUCCESS;
}
```



Somewhere in the C library
- Allocate memory
- Copy values
- Free old memory
- Return answer

Function Pointers

```
int inc(int i) {return i+1;}
int dec(int i) {return i-1;}

int apply (int (*f)(int), int i){
   return f(i);
}
int main() {
  printf(“++: %i\n”, apply(inc, 10));
  printf(“--: %i\n”, apply(dec, 10));
  return 0;
}
```