CS354: Lec 005

- In C, we have to initialize ALL variables to zero; it doesn't do it by default, like Java
- We can define a *compiler variable*, or a constant, with the keyword `#define`

```c
#define MAX 10 // a constant MAX is defined with the value 10

int main() {
    int a[MAX]; // an int array with a size of MAX, or 10

    int i;
    // initialize the array to zeroes
    for (i = 0; i < MAX; ++i) {
        a[i] = 0;
    }
    // iterate through the array and show the values of each index
    for (i = 0; i < MAX; ++i) {
        printf("a[%d] = %d\n", i, a[i]);
    }
    /* If you do not initialize the array indices to zero and try to iterate through the
array,
    there will be no error. Instead, the program will display the memory addresses of each
index       rather than its actual value. The same thing occurs when you change the for loop
condition to     2 * MAX rather than MAX; it will display the correct values for indices 0-9,
but will be       incorrect for the rest. */

    // This will prove that the address of the array is identical to the address of the first
element in the array.
    printf("a = %p\n", a);

    for (i = 0; i < MAX; ++i) {
        printf("addr: %p | a[%d] = %d\n", &a[i], i, a[i]);
    }
    // remember that the size of an integer is 4 BYTES.

    char str[10] = "abcdefghi"; // terminated by '\0'; this is why there is 1 less character
than     the specified size.

    for (i = 0; i < MAX; ++i) {
        printf("addr: %p | str[%d] = %d\n", &str[i], i, str[i]);
    }

    // remember that the size of a char is 1 BYTE.
}
```

- You can also use pointers to display values.

```c
#include <stdio.h>

int main() {
    int a = 1000; // values
    int *pa = &a; // pointer of value


    // print the value of a
    printf("a = %d\t *pa = %d\n", a, *pa); // print the values of a by directly referencing a
AND     dereferencing the pointer value.

    // print the address of a
    printf("&a = %p\t pa = %p\n", &a, pa); // print the address of a by directly referencing
the     address AND using the pointer

}
```

- If you want to declare something as null, you have to explicitly do it, unlike Java:

```
#define NULL 0

int main() {
    int *p = NULL; // you can't actually define something with the keyword NULL, it causes a segmentation fault.
    *p = 10;
    return 0;
}
```

- If you want to make a function, you have to define the function at the top of the file:

```
#include <stdio .h>
#define MAX 10

void print_array(int a[]);

int main() {
    ...
}

void print_array(int a[]) {
    ...
}
// you usually put the functions at the bottom of the file. Some assembly shit or whatever
```

- You can modify the contents of an array using its indices and its addresses, like so:

```
int main() {
    int array[MAX];
    int i;
    int *parray = array;

    for(i = 0; i < MAX; i++) {
        array[i] = i;
    }

    // modify the array normally, e.g. using array index
    for (i = 0; i < MAX; ++i) {
        array[i] = i * 2;
    }

    // modify the array using array pointers
    for (i = 0; i < MAX; i++) {
        *(array+i) = i * 2;

    }

    // modify the array using pointers with array access
    for (i = 0; i < MAX; i++) {
        parray[i] = i * 3;
    }
}
```