C

ARGUMENTS TO MAIN

- So far we've run all the programs as `./executable_name` without any arguments
- Can also accept command line arguments in C
- Currently, main accepts `void`

```
int main(void) {
    return 0;
}
```

ARGUMENTS TO MAIN

- Need to replace void with arguments
  - `argc` = number of arguments passed
  - `argv` = array of arguments (each arg is `char *`)

```
int main(int argc, char *argv[]) {
    return  0;
}
```

ARGUMENTS

- First arg is always name of program
- Other args may need to be converted to a different type
  - Example `./average 100 110`
    - `argv[1]` is `char *` storing `100`
    - need to convert to int
    - `int atoi(char *str)` in `stdlib.h`
    - other conversion functions also in `stdlib.h`

MISC C

- Variables can be preceded by `const` to be read-only
  - We used #define for the same purpose
  - Using `const` qualifier will have normal variable scope
- `static` in front of variable
  - variable maintains value even after block exits
  - Ex: `static int n` in function could be used to identify if it is first call to function or not

MISC C - LIBRARIES

- Compiling our own mutli-file programs
  - Compile all together with gcc
  - Compile into separate object files, then compile those together

MISC C - LIBRARIES

- Standard libraries - don't need to compile them
  - already exist in compiled form (a library)
  - are "linked to" - in default spot, no need to specify
  - have header files that are included - in default spot
- Can do this for custom libraries
  - Need to specify where and link when compiling (`-L` and `-l` options)
  - Likely need to specify where headers live (`-I` option)