Week 9

- Preprocessor
- Splitting up projects into multiple files
- Header

Where do we struggle?

- Modularizing our code
- Splitting up projects
- Not having a huge main.c

How Compile?

- Preprocessor :Replace Stuff
- Compilation : C -> Assembler
- Assembly : Assembler -> Machine Language
- Linking : Combine all files to an executable

Preprocessor

- Macros : #define
- File Inclusion : #include
- Conditional Compilation : #ifdef #ifndef #endif

#define

- #define FIVE 5
- Replace all occurrences of the first with the second
- Literally like search and replace

#include

- Include the contents of a file
- <stdio.h> includes the system header
- „noodle.h“ includes a header in your workspace

What is a header?

- Contain for example function declarations
- Used like an interface
- Some restrictions if you want to use it right

```c
int i = 0;

int add(int a, int b){
  i++;
  return a+b;
}
```

fancy.h

```c
#include <stdio.h>
#include „fancy.h"

int main(){
  printf(„%i %i\n“, add(7,5), i);
}
```

main.c

```c
#include „fancy.h“
```

fancy.c

```
➜ gcc main.c fancy.c
duplicate symbol _i in:
    /var/folders/41/jc8sjvmd52q2cz20fp8wnqtr0000gn/T/main-18c436.o
    /var/folders/41/jc8sjvmd52q2cz20fp8wnqtr0000gn/T/fancy-7d92ef.o
duplicate symbol _aaaa in:
    /var/folders/41/jc8sjvmd52q2cz20fp8wnqtr0000gn/T/main-18c436.o
    /var/folders/41/jc8sjvmd52q2cz20fp8wnqtr0000gn/T/fancy-7d92ef.o
ld: 2 duplicate symbols for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)

~/Desktop/cmpi
```

Our code after Preprocessing

```c
int i = 0;

int add(int a, int b){
  i++;
  return a+b;
}
```

fancy.c

```c
#include <stdio.h>

int i = 0;

int add(int a, int b){
  i++;
  return a+b;
}

int main(){
  printf("%i %i\n", add(7,5), i);
}
```

main.c

Linking is impossible
Both contain I and add

How fix?

- Need to remove instantiations from header files
- Declare functions in Header
- Implement in Source files

Seperation between Header and Source Files

- Put everything your codebase needs to know into a Header file

- Put everything your file needs to know into a Source file

```c
int i = 0;

int add(int a, int b){
  i++;
  return a+b;
}
```

fancy.h

```c
#include <stdio.h>
#include „fancy.h"

int main(){
  printf(„%i %i\n“, add(7,5), i);
}
```

main.c

```c
#include „fancy.h“
```

fancy.c

```h
#ifndef FANCY_H
#define FANCY_H

extern int i;
int add(int a, int b);

#endif
```

Omg

fancy.h

```c
#include <stdio.h>
#include „fancy.h"

int main(){
  printf(„%i %i\n“, add(7,5), i);
}
```

main.c

```c
#include "fancy.h"

int i = 0;
int add(int a, int b){
  i++;
  return a+b;
}
```

fancy.c

Extern

- Say that something is in another file
- Declare a variable without allocating memory
- Not possible otherwise

Our code still suxx

- Why do we need to give access to i in the whole codebase?
- Everyone may modify it
- Write a getter for that

```h
#ifndef FANCY_H
#define FANCY_H

int add(int a, int b);
int getI();

#endif
```

fancy.h

```c
#include <stdio.h>
#include "fancy.h"


int main(){
  printf("%i %i\n", add(7,5),getI());
}
```

main.c

```c
#include "fancy.h"

int i = 0;
int add(int a, int b){
   i++;
   return a+b;
}
int getI(){
     return i;
}
```

fancy.c

Exercises!

- Extract your ArrayList or some other program you wrote into src and header file and use them in the main

- Create the bool type using #define
  Look for a solution in <stdbool.h>

- Literally search in your system for stdbool.h and look into it

- Write a Macro to print a number
  Look up macro functions