COMP1511 PROGRAMMING FUNDAMENTALS

# LECTURE 15

Lecture Program: World Cup Time!

## LAST WEEK…

- Practiced lots with linked lists…
- Forgot to talk about leakcheck commang

## TODAY...

- Multi File Projects - at slower speed with an actual example
- Lecture Program - FIFA World Cup Time

## WHERE IS THE CODE?

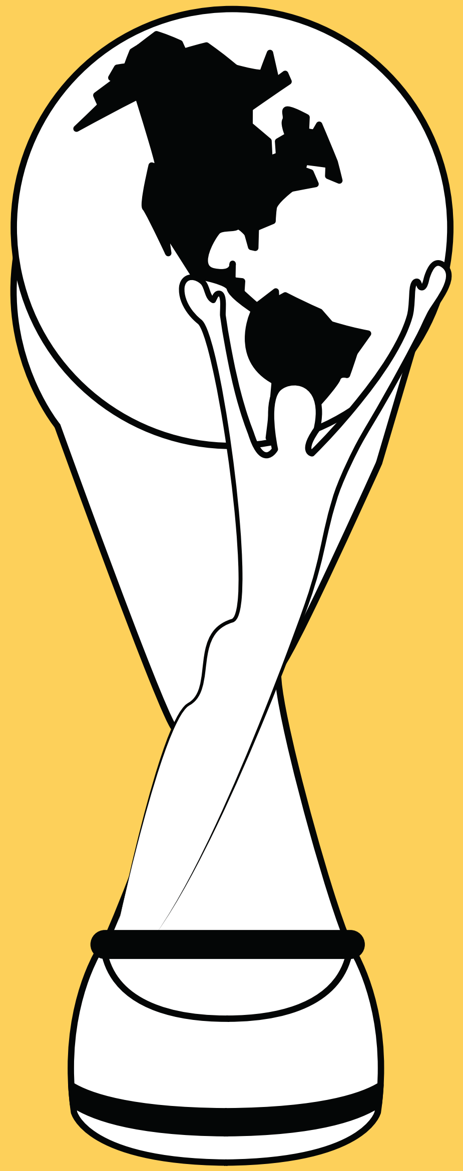
**Live lecture code can be found here:**

HTTPS://CGI.CSE.UNSW.EDU.AU/~CS1511/22T3/LIVE/WEEK09/

## LINKED LIST - ALWAYS THINK ABOUT!

- Some special boundary conditions that you need to consider when you manipulate lists:
    - Empty list
    - List with 1 element
    - Something happening at the beginning of the list
    - Something happening at the end of the list
    - Something will not occur, the item is not in the list (inserting after a number that doesn’t exist etc)

## LECTURE PROGRAM



It’s a FIFA World Cup in just a few weeks (Huzzah!)

To celebrate, I want to be able to keep track of who is on the Australian team by adding players to the list as they get added to the lineup. I will then use the list to keep track of who has been knocked out due to injuries, who has been carded, and who is still playing.

## LECTURE PROGRAM

So:

- I need to create a list, to which I can add all the participating players (initialise)
- I want then to have all the players for each country and separately a list of injured players (a player name/number/etc)
- I want to be able to print out this list (p)

- I want to be able to ”knock” players out of the competition (d player name)

# LECTURE PROGRAM

This is the structure I am provided:

```
enum card_type {NONE, GREEN, YELLOW, RED};

struct fifa {
    char country[MAX_COUNTRY_LENGTH];
    struct player *players;
    struct injured_player *injured_players;
};

struct player {
    char name[];
    int number;
    enum card_type type;
    struct player *next;
};

struct injured_player {
    struct player *player;
    char name[];
    int number;
    int length_of_time_out;
    struct injured_player *next;
};
```

## LECTURE PROGRAM

There is also some starter code (just like in your assignment!). Your job is to:

Add player with 'a' command by typing:

a first_name last_name card_type

Print out the list with 'p' command

Update card status with 'c' command by typing:

c first_name last_name

Delete a player with 'd' command by typing:

d first_name last_name

## MULTI FILE PROJECT

### WHAT ARE THEY?

- Big programs are often spread out over multiple files. There are a number of benefits to this:
    - Improves readability (reduces length of program)
    - You can separate code by subject (modularity)
    - Modules can be written and tested separately
- So far we have already been using the multi-file capability. Every time we #include, we are actually borrowing code from other files
- We have been only including C standard libraries

# MULTI FILE PROJECT

## WHAT ARE THEY?

- You can also #include your own! (FUN!)
- This allows us to join projects together
- It also allows multiple people to work together on projects out in the real world
- We will also often produce code that we can then use again in other projects (that is all that the C standard libraries are - functions that are useful in multiple instances)

## MULTI FILE PROJECT INCLUDES

**.H FILE**
**.C FILE (MAYBE MULTIPLES)**

- In a multi file project we might have:
- (multiple) header file - this is the .h file that you have been using from standard libraries already
- (multiple) implementation file - this is a .c file, it implements what is in the header file.
- Each header file that you write, will have its own implementation file
- a main.c file - this is the entry to our program, we try and have as little code here as possible

**header file**

#include "   .h"

**impleme ntation file**

.c

## HEADER FILE
### #INCLUDE "SOMETHING.H"

Typically contains:

- function prototypes for the functions that will be implemented in the implementation file
- comments that describe how the functions will be used
- #defines
- the file basically SHOWS the programmer all they need to know to use the code
- NO RUNNING CODE
- This is like a definition file

## IMPLEMENTATION FILE

### SOMETHING.C

This is where you implement the functions that you have defined in your header file

## IMPLEMENTATION FILE

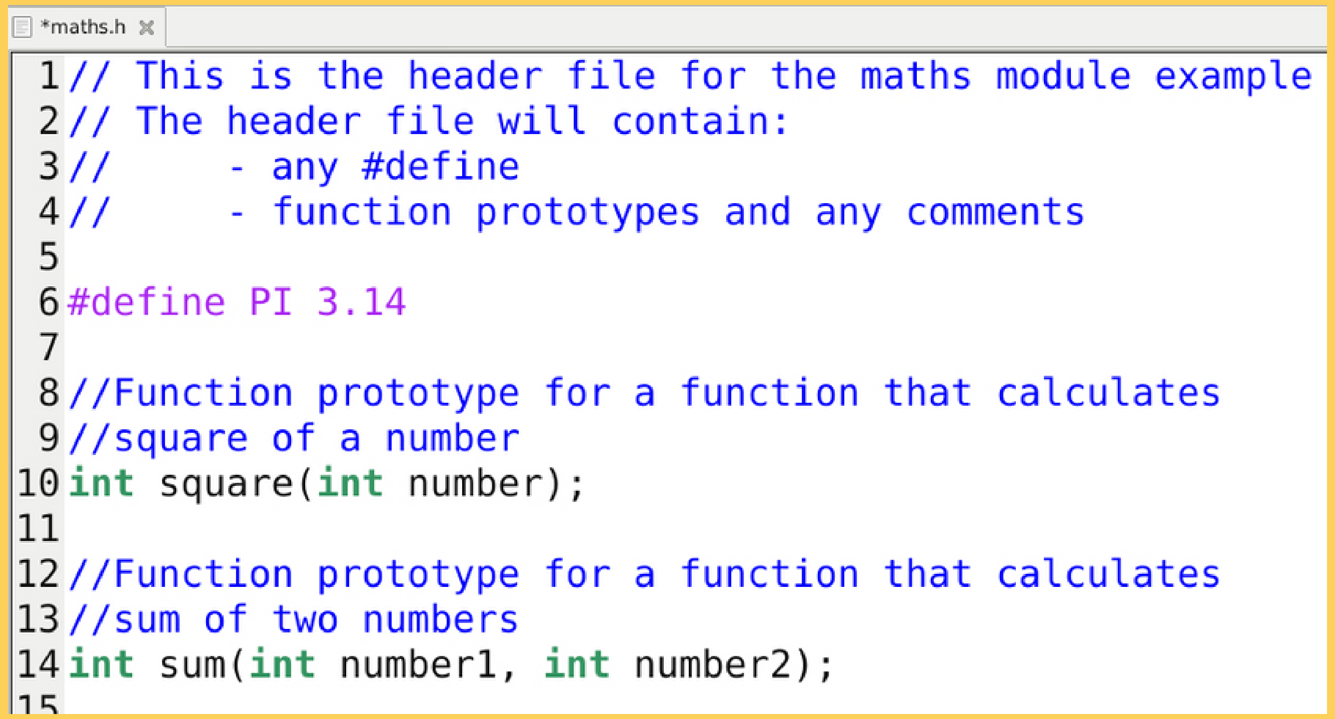
### MAIN.C

This is where you call functions from that may exist in other modules.

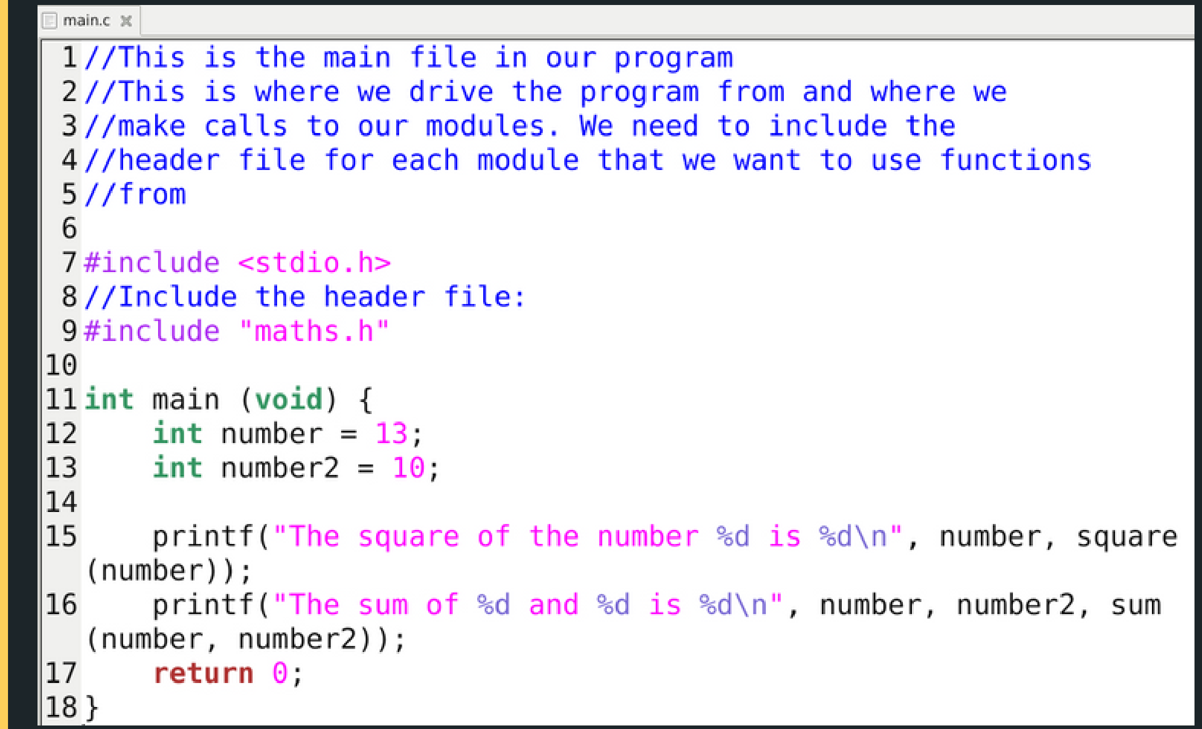
# AN EXAMPLE

## A MATHS

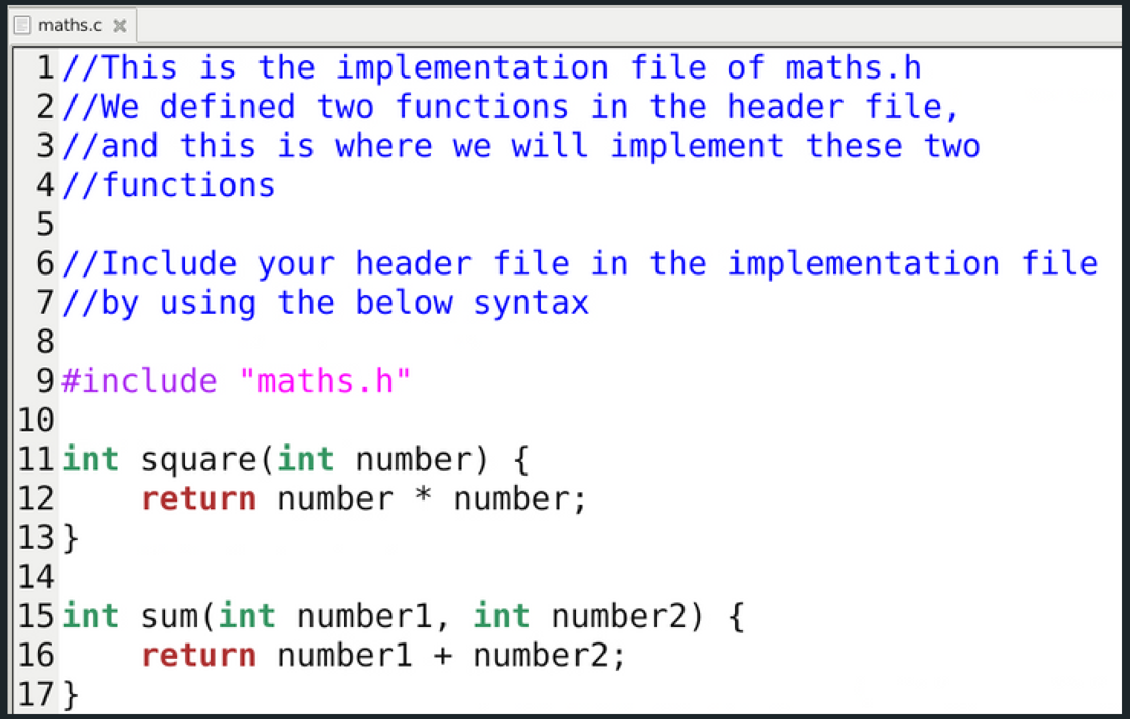
- We will have three files:
    - header file - maths.h
    - implementation file - maths.c
        - #include ”maths.h”
    - main file - main.c
        - #include ”maths.h”



```c
// This is the header file for the maths module example
// The header file will contain:
//      - any #define
//      - function prototypes and any comments

#define PI 3.14

//Function prototype for a function that calculates
//square of a number
int square(int number);

//Function prototype for a function that calculates
//sum of two numbers
int sum(int number1, int number2);

```



```c
//This is the main file in our program
//This is where we drive the program from and where we
//make calls to our modules. We need to include the
//header file for each module that we want to use functions
//from

#include <stdio.h>
//Include the header file:
#include "maths.h"

int main (void) {
    int number = 13;
    int number2 = 10;

    printf("The square of the number %d is %d\n", number, square(number));
    printf("The sum of %d and %d is %d\n", number, number2, sum(number, number2));
    return 0;
}
```

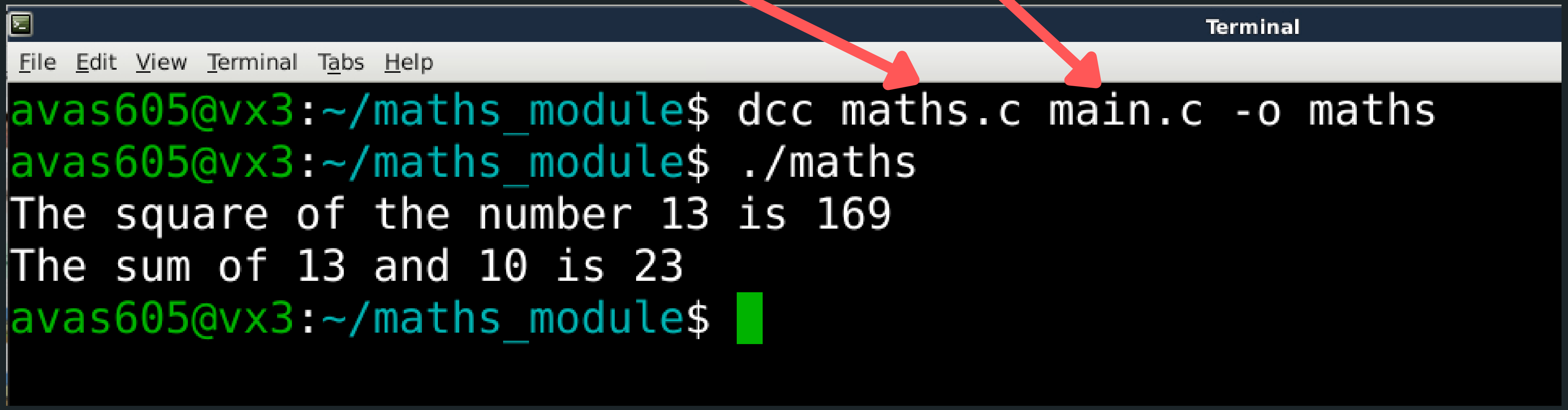


```c
//This is the implementation file of maths.h
//We defined two functions in the header file,
//and this is where we will implement these two
//functions

//Include your header file in the implementation file
//by using the below syntax

#include "maths.h"

int square(int number) {
    return number * number;
}

int sum(int number1, int number2) {
    return number1 + number2;
}
```

## COMPILING A MULTI FILE

### COMPILE ALL C FILES IN THE PROJECT

- To compile a multi file, you basically list any .c files you have in your project
    - In the case of our example, we have a maths.c and a main.c file):



```
avas605@vx3:~/maths_module$ dcc maths.c main.c -o maths
avas605@vx3:~/maths_module$ ./maths
The square of the number 13 is 169
The sum of 13 and 10 is 23
avas605@vx3:~/maths_module$ 
```

- The program will always enter in main.c, so there should only be one main.c when compiling

## Feedback please!

I value your feedback and use to pace the lectures and improve your overall learning experience. If you have any feedback from today's lecture, please follow the link below. Please remember to keep your feedback constructive, so I can action it and improve the learning experience.

https://www.menti.com/alk3kiiv4czb

# WHAT DID WE LEARN TODAY?

MULTI-FILE PROJECTS

maths.c

main.c

maths.h

LINKED LIST LECTURE PROGRAM

fifa.c

## REACH OUT

### CONTENT RELATED QUESTIONS

Check out the forum

### ADMIN QUESTIONS

cs1511@unsw.edu.au