

You are expected to know basic programming

- Datatypes (char, int, float)
- Math operations (+, -, *, /, etc)
- Loops
- Conditional statements
- Comments
- Libraries
- File structure

Readability!

- Naming variables and functions for clarity
 - oven_temperature not t
 - ConvertFahrenheitToCelsius()
 - #Define H2O_BoilingPoint 100

Your code will be marked for readability!

Number data types

- float (decimal numbers)
- long, int, char (integers)
 - unsigned (0 and positive integers only)
 - signed (all integers)

```
#include <stdio.h> // library that contains
                  // printf() code

void main(void) // must have main()!
{
int i = 15, j, k;
j = 22;
k = i+j;
printf("Sum of two numbers %i and %i is %i",
i , j, k);
}
```

Output on display:

Sum of two numbers 15 and 22 is 37

myfile.c

```
#include <stdio.h> // library that contains printf() code

int QuadraticEquation(int a, int b, int c, int x); //prototype

void MyPrintFunction(int value); //prototype

void main(void) // must have main()!
{
int i = 2, j = 4, k = -2;
int x = 3, result;
result = QuadraticEquation(i, j, k, x);
MyPrintFunction(result);
}

// Evaluate a quadratic equation //
int QuadraticEquation(int a, int b, int c, int x)
{
    int value;
    value = a*x*x + b*x + c;
    return value;
}

// Print only positive results //
void MyPrintFunction(int value)
{
    if (value >= 0 )
        printf("Result is %i", value);
    else
        printf("Result is negative");
}
}
```

Output on display:

Result is 28

myfile.c

```
#include "myfunctions.h" // for the functions called below

void main(void) // must have main()!
{
    int i = 2, j = 4, k = -2;
    int x = 3, result;
    result = QuadraticEquation(i, j, k, x);
    MyPrintFunction(result);
}
```

myfunctions.h

```
int QuadraticEquation(int a, int b, int c, int x); //prototype
void MyPrintFunction(int value); //prototype
```

myfunctions.c

```
#include "myfunction.h"
#include <stdio.h> // contains printf()
// Evaluate a quadratic equation //
int QuadraticEquation(int a, int b, int c, int x)
{
    int value;
    value = a*x*x + b*x + c;
    return value;
}

// Print only positive results //
void MyPrintFunction(int value)
{
    if (value >= 0 )
        printf("Result is %i", value);
    else
        printf("Result is negative");
}
```

[Questions](#)

Libraries

A collection of related functions in a C file and the associated header file is usually called a library. In group work, large projects are split into smaller amounts with different members of the group assigned to code different libraries. Usually, although we won't be doing it, the C file is compiled to a .o or object file which people cannot read. The header file is simply a text file so it is important that comments and instructions to users of the library be included here.

- Headers must end with a blank line. Leave it out and you can get error messages that point to the wrong place in the program usually a line with no problem.
- Header mostly contain function prototypes but may also have useful `#define` statements.
- Header functions should not contain `#include` directives to other libraries.
- The source C file and the header file must have the same name.
- Both the source C file and the header must be added to your project files.
- The source C file must `#include` its own header file before any of the code for the functions. If the functions call other functions from outside that C file, you must also `#include` the appropriate headers for those libraries. For example if your function in your library calls `printf()`, you must have `#include <stdio.h>` in your source C file.

```
/* PIC18F4525 program shell . It does nothing.    */
#include <p18F4525.h> // information specific to this chip

// standard configuration statements for this course
#pragma config WDT = OFF
#pragma config OSC = INTIO7 // puts osc/4 on pin 14 to check freq
#pragma config MCLRE = OFF
#pragma config LVP = OFF
#pragma config PBADEN = OFF

void main(void)
{
    /* put instructions that are to run only once here */
while(1) /* this while loop runs forever */
    {
        /* put instructions that are to run continuously here */
    }
// not reachable
}
```

PIC

- Designed to run continuously
- If power is lost, will automatically restart
- Why is this a good idea?

When you write a program!

- What steps need to be run only once?
(usually setup or configuration)
- What steps need to be run continuously?
(monitoring and/or response to input)

Project Development

- Think about the logic
 sketch pseudo-code or flowchart
- Modularize
 write and test small functions

Sketch a flowchart to get dressed in morning.

