Separate Compilation

Example Without using separate compilations:

    /* main.c */
    /* First include any system headers. */
    #include <stdio.h>
    /* Then include the prototypes for all functions that will be used */
    int A();
    int B();
    /* Now comes main */
    int main() {
        A();
        B();
        return 0;
    }
    /* Then finally include the definitions of the functions previously
     * declared */
    void A() {
        printf("In A\n");
    }
    void B() {
        printf("In B\n");
    }

Same example but this time Using separate compilation:

    /* main.c */
    /* (For project 2 this will be proj2.c and should only contain main) */
    #include "funcs.h" /* Notice how " is used instead of <> for including */
    /* a non-system header */
    int main() {
        A();
        B();
    }

    /* funcs.h */
    /* This is a header file which should contain the prototypes of all
     * functions that will be defined in funcs.h */
    /* (For project 2 this will be oilspill.h) */
    void A();
    void B();
/* funcs.c */
/* This is a source file which should contain the definitions for all
 * the functions declared in funcs.h */
/* (For project 2 this will be oilspill.c) */

#include <stdio.h>
#include "funcs.h"

void A() {
    printf("In A\n");
}

void B() {
    printf("In B\n");
}

To compile first:

gcc -c -ansi -Wall main.c
gcc -c -ansi -Wall funcs.c

Notice the addition on the “-c”. The -c instructs the compiler to create special files known as object files. The above
line creates two object files: main.c and funcs.c. These files are like executables except that they end in “-o” and you
can’t actually execute them because they don’t contain all the needed functions. For example main.c needs A() but
A() is provided in funcs.c. When compiling these files the header files included with #include will also be included
in your source code at the point of the include for example the compiler will see main.c as:

/* main.c */
/* the include files begins here */
void A();
void B();
/* and ends here */

int main() {
    A();
    B();
}

Thus all included files will automatically be seen by the compiler.

In order to make a program the two object files needs to be linked together with the command:

gcc main.o funcs.o

Which will then create the executable “a.out”. Here the linker will tell main.o that it should look in funcs.o to find
the definition of the functions A() and B(). Once everything is resolved it will produce an executable. If for some
reason the linker can not find a particular function it will produce a message something like:

undefined reference to ‘A’

Which means that the linker could not find the definition for A. However in this case everything is okay and the
excitable a.out should be created.
Project 2 Calculations

Example of how oil spreading effects the neighboring squares:

\[
\begin{array}{ccccccc}
0 & 0 & 5 & 10 & 0 & 0 & 5 & 10 \\
100 & 0 & \text{==West Wind=>} & 50 & 20 & + & 5 & 10 & = & 55 & 30 \\
100 & 0 & 5 & 10 & 50 & 20 & 55 & 30 \\
0 & 0 & 0 & 0 & 5 & 10 & 5 & 10 \\
\end{array}
\]

More on Random Numbers

The `rand()` function returns a random number between 0 and `RAND_MAX`. To get a number between 0 and a number n divide `rand()` by n and take the reminder, that is take “`rand() % n`”. If you need the number to start at something other than 0 simply add something to the previous expression. So to find a number between 1 and 4 inclusive use “`rand() % 4 + 1`”.

You will also need to seed the random number generator for project 2. In general this should happen in main. So near the top of main you should put the expression “`srand(time(NULL))`”.

Array Example

The Goal

Read in two 2 by 3 arrays and output the Sum.

The Input

\[
\begin{array}{ccc}
1 & 2 \\
3 & 4 \\
5 & 6 \\
6 & 5 \\
4 & 3 \\
2 & 1 \\
\end{array}
\]

The Output

\[
\begin{array}{ccc}
7 & 7 \\
7 & 7 \\
7 & 7 \\
\end{array}
\]
The Code

#include <stdio.h>

#define WIDTH 2
#define HEIGHT 3

void ReadInArray(int array[HEIGHT][WIDTH]);
void SumArray(int a[HEIGHT][WIDTH],
              int b[HEIGHT][WIDTH],
              int result[HEIGHT][WIDTH]);
void PrintArray(int array[HEIGHT][WIDTH]);

int main()
{
    int a[HEIGHT][WIDTH];
    int b[HEIGHT][WIDTH];
    int result[HEIGHT][WIDTH];

    ReadInArray(a);
    ReadInArray(b);

    SumArray(a,b,result);

    PrintArray(result);

    return 0;
}

/* This works because arrays are passed by reference */
void ReadInArray(int array[HEIGHT][WIDTH])
{
    int i, j;
    for (i = 0; i != HEIGHT; i++) {
        for (j = 0; j != WIDTH; j++) {
            scanf("%d",&array[i][j]);
        }
    }
}

void SumArray(int a[HEIGHT][WIDTH],
              int b[HEIGHT][WIDTH],
              int result[HEIGHT][WIDTH])
{
}
void PrintArray(int array[HEIGHT][WIDTH])
{
}

Compiling and Running

gcc -Wall -ansi array.c
./a.out < array.in

Final Notes

Be sure to check out the discussion section web page at http://www.cs.umbc.edu/~kevina1/cmsc201/.