



PRIFYSGOL
BANGOR
UNIVERSITY

Imperative Programming in C

Laboratory 3: Conditional Statements

Dr. Llyr ap Cenydd

Overview

In this lab we are going to be exercising the use of the following:

- If and If-Else statements
- Data precision
- Integer / Float divide
- The use of temporary variables
- Switch case statements
- Enumerated types
- The ternary operator (?)
- The modulus operator (%)
- Generating random numbers

For each lab, there will be an appendix that contains example code snippets to help you get started, and to act as a reference. Alternatively all the commands required to complete these exercises can be found in the lecture notes. The full C language specification, and detailed reference on all the main library headers, can also be found [here](#).

Exercise 1 - Temperature Conversion

We can convert a temperature from Celsius to Fahrenheit and vice-versa using the following formulae:

$$^{\circ}\text{F} = ^{\circ}\text{C} \times 9/5 + 32$$

$$^{\circ}\text{C} = (^{\circ}\text{F} - 32) \times 5/9$$

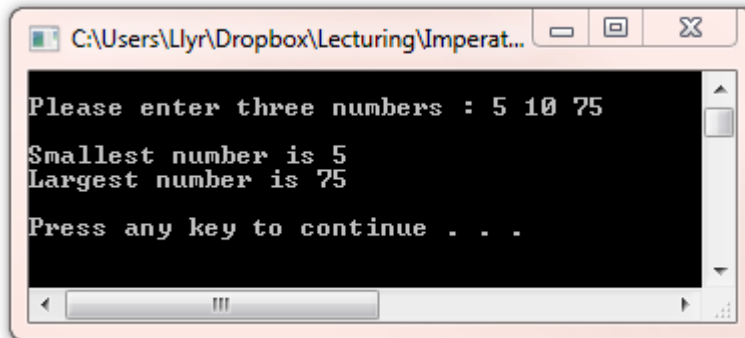
Write a program that converts between Celsius and Fahrenheit. It should ask the user for a temperature and whether they would like to convert from C->F or F->C, before performing the calculation. Print the result to two decimal places.

Hints

- You will need to use an if-else statement here
- Be careful that information isn't being lost in the formula due to int vs float precision
- You can “cast” an integer to a float by adding (float) in front of it. For example (float)(9/5) would treat the calculation as 9.0/5.0, and return 1.8.
- You can test your program knowing that 30.5°C is 86.9°F

Exercise 2 – Min Max

Write a program that asks the user to enter three integer values and finds the largest and smallest of the three values. Try to do this only using if-statements rather than if-else. Your output should resemble the following:



Hints

- It's worth taking a minute to think about how you would sort these values on paper
- Break the task down
- You can do it in three if-statements

Exercise 3 - Switch Month is it?

Write a program asking the user to enter a number representing a month of the year. Using Switch case statements, print out the name of that month.

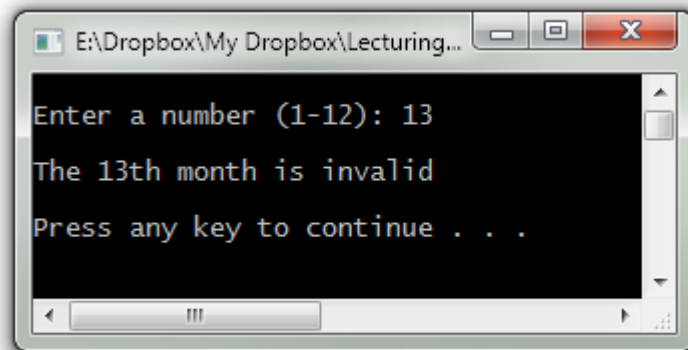
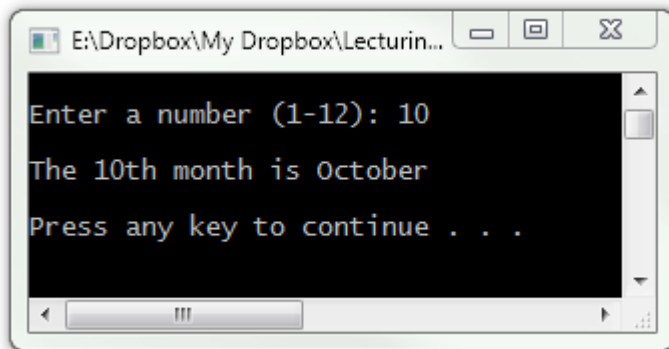


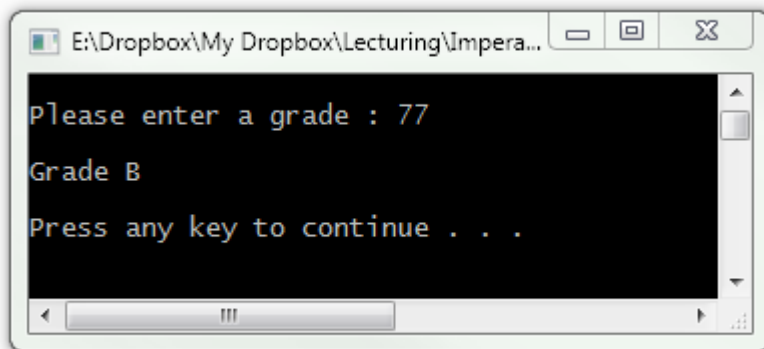
Figure out a way of doing this where the literal string "month" only appears once in the code. If the value input is outside the range 1-12, the program should say so.

Exercise 4 – Switch Grade did I get?

Write a program that uses Switch case statements to print the appropriate grade when the user types in a number. The grade brackets are as follows

- A > 80
- B > 70
- C > 60
- D > 50
- E > 40
- Fail

Your output should resemble the following:



Hints

- Remember that a switch statement can have a default value. Try to use this to simplify your code.
- Try using a simple integer division on the user value to simplify the switch cases, so that there's no need for any conditional statements

Exercise 5 – Colour Enums

Note = Windows only exercise

The following code snippet will print out a message in green. By changing the value of the variable colourPick (1-255) we can print in other colours.

```
#include <windows.h>    // WinApi header

colours()
{
    //Get a handle to the console window
    HANDLE hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

    int colourPick = 10;

    //Set the colour to be x
    SetConsoleTextAttribute(hConsole, colourPick);
    printf("\n\nThe quick brown fox jumped over the lazy hen");
}
```

Using this program as a starting point, create an enum type called colours that stores all 7 colours of the rainbow, assigning the corresponding colour code value to each (e.g. green is 10).

Once your program is finished, you should be able to change the text colour using a line of code like this:

```
SetConsoleTextAttribute(hConsole, blue);
```

Hints

- Enums can be assigned constant integer values
- Enums allow us to make code easier to read (no magic numbers)
- Multiplying a base colour by 16 will change the background colour, keeping the text black

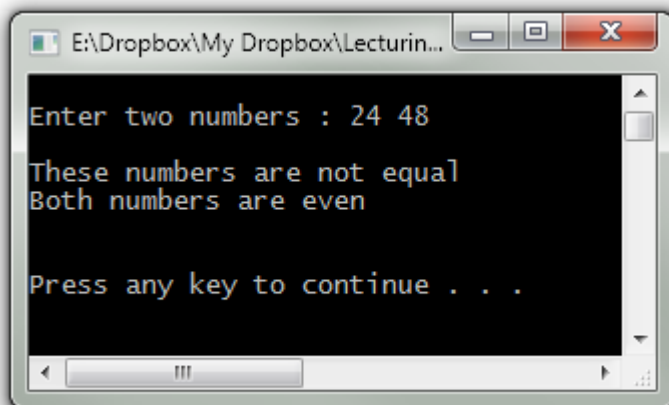
Exercise 6 - Ternary Operator

Write a program that asks for two numbers on a single line.

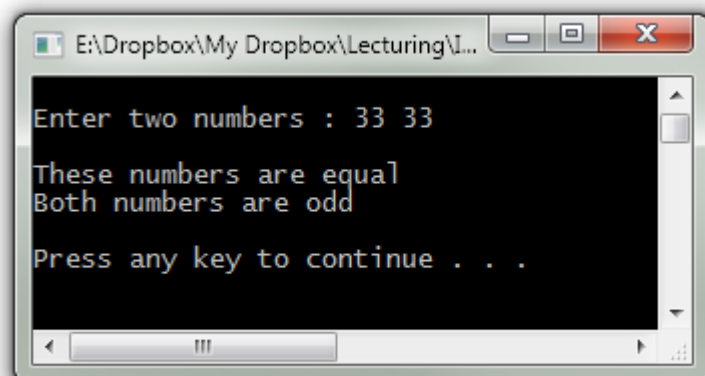
The program should then compare the two numbers and say if:

- The numbers are equal or not
- If a is greater than b
- The numbers are both odd
- The numbers are both even

You are not allowed to use if-statements for this exercise. Instead, **use only the ternary and relational operators**. Example output could be:



```
E:\Dropbox\My Dropbox\Lecturin...
Enter two numbers : 24 48
These numbers are not equal
Both numbers are even
Press any key to continue . . .
```



```
E:\Dropbox\My Dropbox\Lecturing\I...
Enter two numbers : 33 33
These numbers are equal
Both numbers are odd
Press any key to continue . . .
```

Exercise 7 - Leap Years

A leap year consists of 366 days, as opposed to a common year, which has 365 days.

Your task in this exercise is to write a program that tells the user if the year they enter is a leap year or not. A leap year follows this criteria:

- The year is evenly divisible by 4;
- If the year can be evenly divided by 100, it is NOT a leap year, unless;
- The year is also evenly divisible by 400. Then it is a leap year.

Hints

- Example Leap years are 2008, 2012, 2020, 2032. You can use this to test your code.
- Recall that the modulus operator (%) computes the remainder that results from performing integer division.
- The key numbers are 4, 100, 400
- You will need if-else statements for this program
- Try not to Google for code

Random numbers

To generate a random number in C we can use the `rand()` function declared in `stdlib.h`.

However, we first need to **seed** the random number generator (RNG) with a value, which will determine what the random numbers are. Usually if we want different random numbers every time we run the program we seed the generator with the current time. However if we want deterministic results (level seeds in games like Minecraft for example), we can specify the seed manually.

We seed the RNG using [`srand\(seed\)`](#). To seed using the current time we need to use the `time()` function from the library `time.h`:

```
#include <time.h>
#include <stdlib.h>

srand(time(NULL)); //seed the RNG with curr time
int r = rand();
```

Unless you plan to generate billions of numbers, we only need to seed the random number generator once. In fact seeding it often will generate a less random sequence than just seeding once. Finally, we can use the modulus operator to generate a random number between 0 and 99 like this:

```
rand() % 100;
```


Exercise 8 - Time of day

Write a program that asks the user for a time of day (e.g. "Morning"), and then generates a random time that falls under that part of the day.

- Night (00:01 to 05:00)
- Early Morning (05:01 to 08:00)
- Morning (8:01-12:00)
- Afternoon (12:01 – 18:00)
- Evening (18:01 – 12:00)

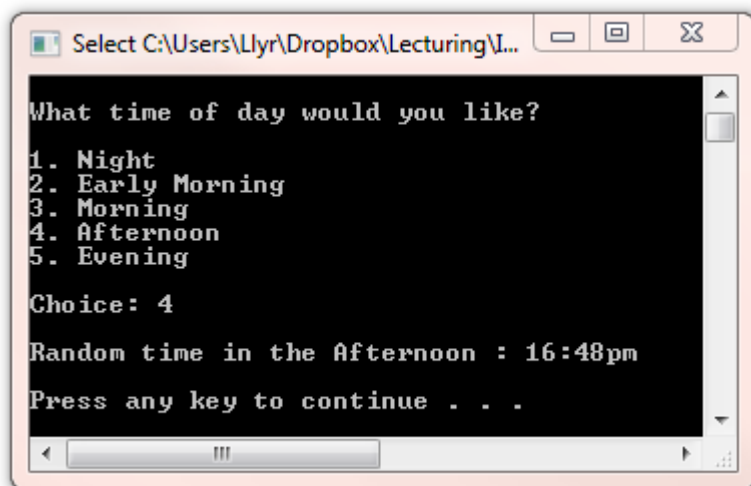
You should format your output so that it is in the format XX:XXam/pm.

It is up to you how you structure your program – try to look back to what we have covered in this lab and think about what kind of conditional statements work best here.

Hints

- Think about an efficient way of encoding and randomly generating the time
- Consider leaving the formatting to the print statement(s)
- How much can you condense this program?

Example output:



```
Select C:\Users\Llyr\Dropbox\Lecturing\L...  
What time of day would you like?  
1. Night  
2. Early Morning  
3. Morning  
4. Afternoon  
5. Evening  
Choice: 4  
Random time in the Afternoon : 16:48pm  
Press any key to continue . . .
```

Appendix

[Online C Programming Resources](#)

[Complete C Reference Library](#)

C Programming IDE's

[Dev-C++](#) (Windows)

[Code::Blocks](#) (Windows, Mac, Linux)

[Visual Studio/C++ Express](#) (Windows)

[Netbeans C/C++](#) (Windows, Mac, Linux)

[Codelite](#) (Windows, Mac, Linux)