

ELEC3730 - Introduction to C Tutorial

What follows is a sequence of increasingly difficult programming problems you should try to solve by writing a 'C' program. For this purpose, I recommend you use the GNU `gcc` compiler.

On a linux system, this is pre-installed, and in order to compile your program in file (say) `q1.c` and then run it you would invoke

```
csh>gcc q1.c -o q1_executable -Wall
csh>./q1_executable
```

On a Windows platform, I recommend you use the DEVCC++ package which you can download from the ELEC3730 homepage under the “software” section, and which provides the `gcc` compiler as well as editor and integrated development environment including debugger.

You may attempt as many of the problems below as you like. The more you attempt, the better prepared you are for subsequent parts of this course. I have provided more problems than are required as a bare minimum.

1. Write a 'C' program that prints “Hello World” on the screen. We have already done this in lectures, so this question is just to make sure you can drive the compiler to generate an executable and run it.
2. Write a program that reads in a set of integers from the keyboard, adds them together, and then prints the result.
3. Write a program that reads in the radius of a sphere, in meters, and calculates and outputs the volume of that sphere in cubic meters (use the formula $V = 4/3\pi r^3$).
4. Under (old) tax laws, income between \$6000 and \$20,000 is taxed at 17%, between \$20,001 and \$50,000 at 30%, between \$50,001 and \$60,000 at 42% and beyond \$60,001 at 47%. Finally, a 1.5% Medicare levy applies. Write a program which will accept a gross income amount and return the net income.
5. Write a program which will accept a date in `dd/mm/yyyy` format and return what number day of the year it is. Eg. `10/01/2005` is day 10 of the year.
6. Write a program which, assuming a monthly savings plan of \$150/month prints out the end of year savings assuming interest rates from 2% to 7% in increments of 1%, and does the calculation for years 1 to 9.
7. Write a program that accepts an integer, and determines whether or not it is prime. If it is not prime, the program should return a factorisation that proves it is not prime.
8. Write a program that accepts an arbitrarily large number of integers, and prints out the maximum value of the numbers entered when CTRL-D is typed to signify the end of the list.
9. Repeat question 7, but this time use a separate subroutine `isprime` to test primeness. Capitalize on this modularity to calculate the smallest prime that is bigger than whatever number is entered.

10. Write a program which will accept a double precision floating point value x , and print out $\sin(x)$, $\log(x)$, $|x|$, \sqrt{x} as well as the value of π and $\sqrt{2}$. All values should be printed to 15 decimal places. Note that under linux, the compile command will now have to explicitly include the math library by use of the `-lm` switch.

```
gcc q10.c -o q10_executable -Wall -lm
```

11. Write a program that declares an automatic variable `x` and a `static` variable `w` which are not initialised, and then prints out their values and the address (in hexadecimal) they reside at. Run the resultant program several times, with alternations of running something else (anything) in-between program calls, and not.
12. Write a program that generates an 8-element array of integer values, initialises them with the first 8 prime numbers, and then prints out those values including the array element they are stored in.
13. Write a program that declares an arbitrary string, and prints out the address of the start of the string, plus the string itself. The program should then progressively move the pointer to the string forward by one place, and print out the resultant address and string until the string is exhausted. Test with different strings. Try to write a program that depends only on the string, and does not need to have any other value dependent on the string entered.
14. In order to understand what the `argc`, `argv` arguments of `main` mean (we've ignored them so far), write a program to print them out, and experiment with different input arguments in order to divine the meaning. In order to do this, you need the prior information that `argc` is an integer representing the number of arguments passed to `main` (when it is called), and `argv` is then an array of strings containing those arguments.
15. Write a subroutine that will integrate an arbitrary function between arbitrary limit points. To do this, you will need to be able to pass a pointer to a function to your subroutine. Illustrate its use by computing and printing the value

$$\int_0^{2\pi} e^{\sin x} dx.$$