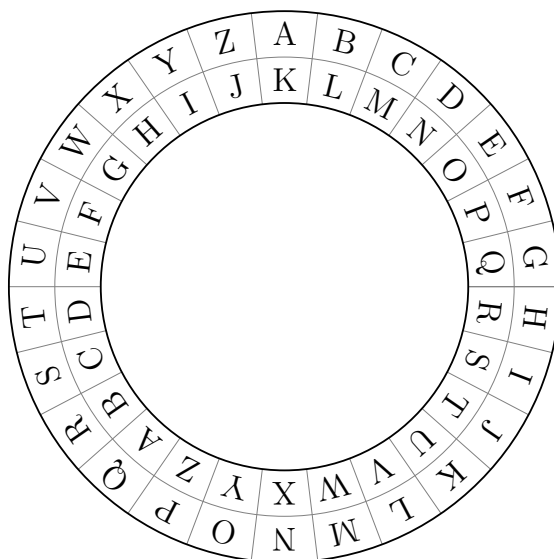


C4

Pointers, Files & Strings

One of the challenges of writing larger programs in C is to make the code easier to read and understand by another programmer. To help with this, C has a number of features to aid in breaking up the code into smaller modules with clearly defined interfaces. To use them effectively you will learn about pointers which enable you to pass variables by reference to functions, enabling the function to update the values and return these updates to the point of the code that called the function. Alongside this you will learn more about strings and how to read from text files to enable you to develop a simple code breaking tool for cryptography.



Schedule

Preparation Time : 3 hours

Lab Time : 3 hours

Items required

Tools :

Components :

Equipment :

Software : gcc, text editor (e.g. notepad++)

Version: October 25, 2020

©2012


Steve R. Gunn

Electronics and Computer Science

University of Southampton

Before the laboratory you should read through this document and complete the preparatory tasks detailed in section [2](#).

Academic Integrity – *If you wish you may undertake the preparation jointly with other students. If you do so it is important that you acknowledge this fact in your logbook. Similarly, you will probably want to use sources from the internet to help answer some of the questions. Again, record any sources in your logbook.*

You will be working on your own in this laboratory. During the exercise you should use your logbook to record your observations, which you can refer to in the future – perhaps to write a formal report on the exercise, or to remind you about the procedures. As such it should be legible, and observations should be clearly referenced to the appropriate part of the exercise. As a guide the  symbol has been used to indicate a mandatory entry in your logbook. However, you should always record additional observations whenever something unexpected occurs, or when you discover something of interest.

For each Task you should create a new directory so that you have a working version of every program at the end of the lab. Remember to place comments in your code.

You will be marked using the **standard laboratory marking scheme**.

Notation

This document uses the following conventions:



An entry should be made in your logbook



Care should be exercised

Remarkable text

A point of note

`command input`

Command to be entered at the command line

`output response`

Response produced at the command line

1 Introduction









This lab introduces you to programming with pointers which is a powerful and fundamental concept in C programming. In addition you will develop some string manipulation techniques and learn how to read from text files.

1.1 Outcomes

At the end of the exercise you should be able to:

- ▶ Design and implement simple C programs where you declare, define and call functions which pass arguments both by value and reference.
- ▶ Design and implement simple C programs where you are able to perform basic string manipulation.
- ▶ Design and implement simple C programs where you are able to read in and process a text file.

2 Preparation

1. Describe the behaviour of the * and & operators in C with regard to pointers. 
2. Detail how an integer array can be passed as an argument to a function by reference so that the function is able to update the contents of the array and return these changes to the calling function. 
3. Look up the definitions for FILE, fopen, fclose, fgetc and EOF and describe their purpose. 
4. Write code that opens a text file for reading and displays the contents of the file on stdout. 
5. Which header file contains the interfaces for isalpha and toupper? Describe the operation of these two library functions. 
6. Characters in C are represented in ASCII format. What value is stored in memory to represent the character 'A'? 
7. Look up the operation of a Caesar Cipher and describe how it works. 
8. What operator is used to implement the modulo operation in C? 

3 Laboratory Work

Create a new directory ELEC1201/Labs/C4 to store your files for the laboratory. For each task you should create a new sub-directory so that you have a working version of every program at the end of the lab.

3.1 Reading from files

1. Take your code from the preparation and produce a program which reads in a text file and prints the contents to `stdout`.
2. Modify your program so that it only prints out alphabetic characters in upper case.
3. Add a function called `calculateHistogram` which takes a filename and a 26 element array as arguments and calculates the number of occurrences of each letter in the text file. Hint: You will need to move the file opening and reading operations into this new function from your main program.
4. Add a function which takes an integer array as an argument and prints out the value stored in each element on a new line. Use the `const` keyword to declare that the function will not modify the contents of the array.
5. Add a second function that prints the contents of the array in a different format using a number of `*` on each line proportional to the element of the array. You will need to add a normalisation term so that the largest element corresponds to the line width to make this legible. To do this search for the maximum term in the array and use this to scale the number of `*`.



3.2 Manipulating Strings

1. Implement a function

```
void encipher(const char *p, char *c, const unsigned int offset)
```

that takes a plain text string `p` and enciphers it using a caesar cipher with `offset` to produce a cipher text `c`.
2. Implement a complementary function

```
void decipher(const char *c, char *p, const unsigned int offset)
```

that takes a cipher text string `c` and deciphers it using a caesar cipher with `offset` to produce a plain text `p`.
3. Write a program that declares a plain text string, enciphers it, and then deciphers it, displaying the results at each stage.

3.3 Code Breaking

1. By combining your work above write a program which builds up statistics on the occurrence of alphabetic characters, and then uses this to automatically determine the correct offset to use in the deciphering stage of the caesar cipher. To do this you can try all 26 possible offsets and score the resulting string by summing the likelihood of each resultant character. The string with the largest score is the most likely result.
2. Investigate the effectiveness of the approach by enciphering a number of plain texts of varying length and comment on the results.



4 Optional Additional Work

1. Modify the `encipher` and `decipher` functions to implement a more advanced cryptographic approach.