

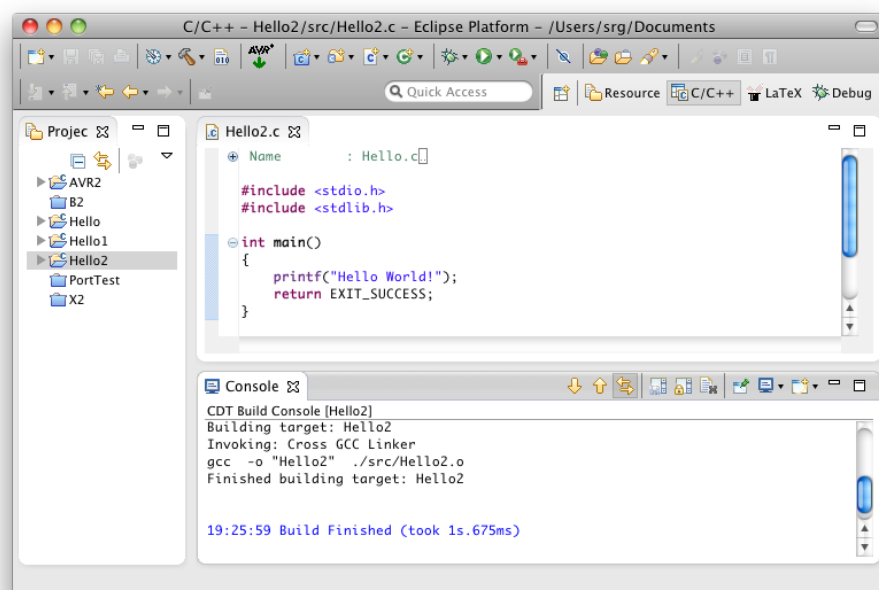
# C5

---

## Memory Allocation, Data Structures & Files

---

In this exercise you will use some of the features you have used earlier in this module, such as structures and pointers, in combination with other advanced features of the C language. You will practice dynamic memory allocation as well as reading and writing data from and to files on the computer. The combination of structures and dynamic memory allocation allows the definition of very versatile data structures, such as linked lists. The second part of this lab exercise focuses on how to use an existing linked list library and how to customize it to solve specific problems.



## Schedule

---

Preparation Time : 3 hours

Lab Time : 3 hours

## Items provided

---

Tools :

Components :

Equipment :

Software : gcc, Eclipse

## Items to bring

---

Course Textbook - *C Programming in Easy Steps*

Identity card

Laboratory logbook

Version: September 29, 2013

©2013


Enrico Costanza, Steve R. Gunn

Electronics and Computer Science

University of Southampton

Before entering 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 undertake the exercise working with your laboratory partner. 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; at the beginning of the exercise one of the laboratory demonstrators will mark your preparatory work and at the end of the exercise you will be marked on your progress, understanding and logbook.

## Notation

This document uses the following conventions:



An entry should be made in your logbook



A hint to application areas—can be ignored







## 1 Outcomes

Having successfully completed this lab session, you will be able to:

- ▶ design and implement simple C programs that read and write files;
- ▶ design and implement simple C programs that allocate dynamic memory;
- ▶ design and implement simple C programs that uses simple data structures.

## 2 Preparation

Read chapter 10 of the course textbook (McGrath, M, "C Programming in Easy Steps," In Easy Steps Ltd., 2012).

- ▶ Explain in your logbook the difference between dynamically allocating an array and statically declaring its size. How do you decide which technique to use on a case-by-case basis? 
- ▶ What are segmentation faults? 
- ▶ One problem that can occur when using dynamic memory allocation is a memory leak. What is a memory leak and how is it avoided? 
- ▶ Write a C program that creates a file "newfile.txt" in the current directory containing 25 lines of identical text "abcdefghijklmnopqrstuvwxyz". 
- ▶ Explain in your logbook: what are advantages and disadvantages of linked lists and arrays, when compared to each other? 
- ▶ Briefly describe in writing on your logbook how structures and pointers can be used to create data structures such as linked lists and trees. 

## 3 Laboratory Work

Open the Eclipse application on the computer you are using and when asked for the workspace you wish to use, make sure the checkbox for *use as default* is not ticked and type H:\ELEC1201\Labs\C5.

Remember to create a new C project (File -> New -> New project.. -> C/C++ -> C Project), for each task of the exercise and name it with the appropriate part of the exercise for easy future reference, e.g. C5\_3.1.1. You can select "Hello World ANSI C Project" as "Project type" and replace the contents with your own code, or create an "Empty Project" and add in a new source file.

Remember, if you are asked if you want to "Switch perspective" answer positively.

### 3.1 Memory Allocation

Write a program where you ask the user to input an integer number `n` and then you allocate an array of double precision floating point values of size `n`. Fill the array with random values between 0.0 and 1.0, print the content of the array and release the memory.

### 3.2 Files

The file `c_wiki.txt` contains the first section of the Wikipedia page about the C Language. The file contains 4 paragraphs. Write a program that reads the file and writes out each paragraph in a separate file.

### 3.3 Using a Linked List and Files

The files `list.h` and `list.c` provide a simple linked list. The code includes a test function, demonstrated in `test_list.c`. Create a new project in Eclipse for these 3 files (instructions for dealing with multiple files in Eclipse were given in lab C3).

Make sure you can compile and run the `test_list.c` program.

Your task for this part is to use `strtok` to parse words out `c_wiki.txt` and add them to the list so that each word appears in the list only once.

Comment on your logbook about how you could have done the same task using dynamically allocated arrays and what would have been the advantages and disadvantages compared to using the list.



*Hints:*

- ▶ This task is similar to one of the tasks you did in the previous lab, so you can start from there.
- ▶ Use the `list_insert_sorted` function so that the list is kept in alphabetical order.

## 4 Optional Additional Work

### 4.1 Counting Words

Modify the program you wrote in Section 3.3 so that it also counts how many times each word appears in the text.

*Hint:* check you get the correct results using the following information.

Words	Frequency
C	13
and	11
the	10
a, of	7
in, standard	6
as, for	5
an, is, that, to	4
The, by, generally, languages, most, was	3
In, Ritchie, approved, been, called, computer, from, known, language, later, like, on, or, programming, specification, system, there, version	2
all other words	1

## 4.2 Modifying the Linked List

Modify the program you wrote in Section 4.1 A so that the frequency of each word is stored in the list as float rather than as an int.

## 4.3 Doubly Linked List

Modify the list library from Section 3.3 so that it uses a double linked list rather than a single linked list. This means that each list element should have not only a next pointer, but also a prev pointer.

Using a double linked list uses more memory, but it makes some operations more efficient. Explain in your logbook which are these operations and how the double links help.

