CS2100: Computer Organisation
Lab #2: Debugging using GDB II

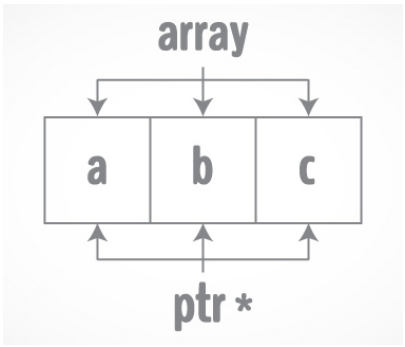
Name: _____ Student No.: _____

Lab Group: _____

Remember to bring and prepare this report before attending the lab!

C Arrays

Arrays are data structures that store fixed-size sequential collections of elements of the same type. While an array simply stores a collection of data, it is often more useful to think of the collection as a collection of variables of the same type.

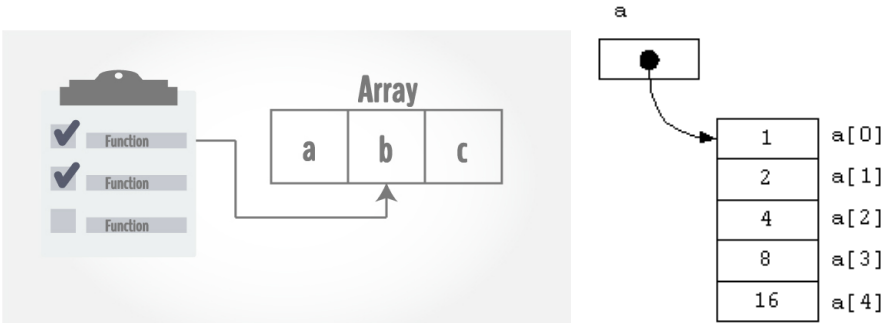


Instead of declaring individual variables, eg. `number0`, `number1`... `number99`, we can declare a single array variable `numbers` and use `numbers[0]`, `numbers[1]`,…`numbers[99]` to represent individual variables. A specific element in an array is accessed by an index which starts from 0.

All arrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.

C Functions and Arrays

In C programming, both a single array element or an entire array can be passed to a function. A single value will be passed by value, whereas a whole array is always passed as a reference (think pointer) to the first element of the array. In other words, the array itself is represented by a pointer to the first element of the array.



Objective: You will learn how to use arrays and functions in C.

Procedure:

1. Download the files **`lab2a.c`**, **`lab2b.c`** and **`lab2c.c`** from Canvas.

2. Compile **`lab2a.c`** with **`gcc`** using the following command: **`gcc –o lab2a lab2a.c`**

3. What is the output of the program?

4. Which line in the code should be changed to get output **`"2"`** instead? Show the changed line.
   Note: The output should be related to the **`ageArray`**. Do not hardcode "2" in your code!

5. What is the purpose of the unary operator **`sizeof`**?
   What datatype will **`sizeof`** give the value **`"1"`** for all architectures?

6. Can you get the number of elements in **`ageArray`**? Write a modified main function below to produce the following output. Show your lab TA the output of the code.

   ```
   2
   Size of the array is 4
   ```

   Note: The output **`"2"`** and size of array (i.e., **`4`**) are related to **`ageArray`**. Do not hardcode the value "2" and "4" in your code! (instead change the initialisation of the array)

7. Compile **`lab2b.c`** with **`gcc`** using the following command: **`gcc –o lab2b lab2b.c`**

8. Give 2 ways of displaying the stored value of the first element of an array.
   Give 2 ways of displaying the address value of the first element of an array.

9. Can you define the function **`hexToDecimal(char hex[], size_t size)`** in **`lab2b.c`**, using pointers to traverse the array?
   Write your function below and show your labTA the output.

   Note: You are not allowed to use **`strtoul`**, **`strtol`**, or other functions from **`stdlib.h`** or **`math.h`**. *Hint: Reading from the back of array is easier. Furthermore, you are already given the function* **`hexVal(char hex)`** *to simplify your work.*

10. Why do we pass the size of the array to the **`hexToDecimal`** function in lab2b.c? Can we calculate the size of the array inside the function?

11. What is the format specifier to print a variable of datatype **`size_t`**?

12. Compile **`lab2c.c`** with: **`gcc –o lab2c –DTEST0 lab2c.c.`**
    What does the option **`–DTEST0`** do?
    Hint: read the man page of gcc, i.e. issue the command: **`man gcc`**.

13. Execute **`lab2c`** and report what happened. Explain how the output was obtained.

14. Now recompile **`lab2c.c`** with: **`gcc –o lab2c –DTEST1 lab2c.c`**.
    Execute **`lab2c`** and report what happened. Explain how the output was obtained.

15. Now recompile **`lab2c.c`** with: **`gcc –o lab2c –DTEST2 lab2c.c`**.
    Execute **`lab2c`** and report what happened. Explain how the output was obtained.

16. Now recompile **`lab2c.c`** with: **`gcc –o lab2c –DTEST3 lab2c.c`**.
    Report what happened. Explain why.

Marking Scheme: Report – 16 marks; correct output – 4 marks; Total: 20 marks.