Data Structures & Algorithms

By

Dr. Emmy MUGISHA

May 2023

LAB 1 ***(/5 Marks*****)**

- ***Using C***, write a program to compute the addition of ***5 entered*** numbers (***integers***) with a ***For loop***.

Introduction: Algorithms

- The term **algorithm** originally referred to any computation performed via a set of rules applied to

- numbers written in decimal form. It is a representation of a solution to a problem.

- An algorithm is a ***step by step method*** of ***solving*** a ***problem***.

- It is commonly ***used*** for ***data processing***, ***calculations***, and other related computer and mathematical ***operations***.

Introduction: Algorithms

- We solve many such trivial problems every day without even thinking about it,

- for example **making breakfast**, **travelling to the workplace**, **crossing a road** etc.

- It is used to ***manipulate*** data in various ways, such as;
- ***inserting*** a new data item
  - ***searching*** for a particular item and
  - ***sorting*** items

Introduction: Algorithms

- In a ***non-technical*** approach, they are used in everyday tasks,

- i.e. ***recipe to bake a cake or a do-it-yourself handbook***.

- Technically, computers use algorithms ***to list the detailed instructions for carrying out an operation***.

- For instance, to compute an ***employee's paycheck, the computer uses an algorithm.***

Introduction: Algorithms

Structure of an Algorithm

An algorithm has the following ***structures***;

- Input Step
- Assignment Step
- Decision Step
- Repetitive Step
- Output Step

Introduction: Algorithms

- An ***algorithm*** has the following ***properties***:
  - ***Finiteness***: An algorithm must ***terminate*** after a ***finite*** number of ***steps***.
  - ***Definiteness***: The steps of the algorithm must be precisely ***defined***.
  - ***Generality***: An algorithm must be ***generic*** enough to **solve** all ***problems*** of a particular ***class***.

Introduction: Algorithms

- ***Effectiveness***: the operations of the algorithm must be ***basic*** enough to be ***put*** down on ***pencil*** and ***paper***.

  They should ***not*** be too ***complex*** to warrant writing ***another*** algorithm for the ***operation***.

- ***Input-Output***: The algorithm must ***have*** certain ***initial*** and precise ***inputs***, and ***outputs*** that may be generated both at its ***intermediate*** and ***final*** steps.

Introduction: Algorithms

- **Different Approaches to Design an Algorithm**
- An algorithm does ***not enforce*** a ***language*** or mode for its ***expression*** but only demands ***adherence*** to its ***properties***.

- ***Practical Algorithm Design Issues***
  - ***To save time*** (Time Complexity): A ***program*** that ***runs*** faster is a ***better*** program.
  - ***To save space*** (Space Complexity): A ***program*** that **saves *space*** over a competing program is considerable ***desirable***.

Introduction: Algorithms

- To design an algorithm that is easy to ***understand, code, and debug***.
- To design an algorithm that makes efficient use of the ***computer's resources***.

- ***Efficiency of Algorithms***
- The ***performances*** of algorithms can be ***measured*** on the ***scales*** of ***time*** and ***space***.

- The ***performance*** of a program is the ***amount*** of computer ***memory*** and ***time*** needed to run a program.

Introduction: Algorithms

- ***Time Complexity***
- The ***time*** complexity of an algorithm or a program is a ***function*** of the running ***time*** of the algorithm or a ***program***.

- In other words, it is the ***amount*** of computer ***time*** it needs to ***run*** to ***completion***.

Introduction: Algorithms

- ***Space Complexity***
- The ***space*** complexity of an algorithm or program is a ***function*** of the ***space*** needed by the algorithm or program to ***run*** to ***completion***.

Introduction: Algorithms

- **Recursion and Iteration Algorithms**
- ***Recursion Algorithms***
- This is a ***method*** of simplification that ***divides*** the ***problem*** into ***sub-problems*** of the same ***nature***.

- The ***result*** of one ***recursion*** is the ***input*** of the ***next recursion***.

Introduction: Algorithms

- ***Iteration Algorithms***
- This is an algorithm that ***executes iterative steps***.

- It is ***aimed*** at ***finding successive*** approximation in ***sequence*** to reach a ***solution***.

- They are commonly ***used*** in linear ***programs*** where ***large numbers*** of variables are ***involved***.

Introduction: Algorithms

- ***Difference between Recursion and Iteration:***
  - A function is said to be ***recursive*** if it ***calls itself*** again and ***again within its body*** whereas ***iterative*** functions are ***loop based*** imperative ***functions***.
  - ***Recursion*** uses ***stack*** whereas ***iteration*** does ***not*** use stack.

Introduction: Algorithms

- Recursion uses ***more memory than iteration*** as its concept is based on ***stacks***.

- Recursion is comparatively ***slower than iteration*** due to ***overhead*** condition of ***maintaining stacks***.

- While using recursion ***multiple activation*** records are ***created*** on ***stack*** for each ***call,*** where as in ***iteration everything*** is ***done*** in ***one*** activation ***record***.

Introduction: Algorithms

- ***Infinite*** recursion can crash the system whereas ***infinite looping*** uses ***CPU*** cycles ***repeatedly***.

- Recursion uses ***selection*** structure whereas iteration uses ***repetition*** structure.

- Recursion makes code ***smaller*** and iteration makes code ***longer***.
- Iteration ***terminates*** when the ***loop-continuation*** condition ***fails,*** whereas recursion ***terminates*** when a base case is ***recognized***.

Introduction: Algorithms

- ***Types of Recursion***
- Recursion is of ***two*** types ***depending*** on whether a ***function*** calls ***itself*** from within ***itself*** or whether ***two functions*** call ***one*** another ***mutually***.

- The former is called ***direct recursion*** and the later is called case ***indirect recursion***.

Introduction: Algorithms

- ***Thus there are two types of recursion;***
  - Direct Recursion
  - Indirect Recursion
- **Recursion may be further categorized as;**
  - Linear Recursion
  - Binary Recursion
  - Multiple Recursion

Introduction: Algorithms

- ***Linear Recursion***
- It is the most common type of Recursion in which a ***function calls itself repeatedly*** until base condition (termination case) is ***reached***.

Introduction: Algorithms

- ***Binary Recursion***
- These are recursive functions which ***don't just have one call to themselves***; they have two (or ***more***).

- Functions with ***two*** recursive ***calls*** are ***referred*** to as ***binary recursive*** functions.

Introduction: Algorithms

- ***Tail recursion/ Multiple***
- Tail recursion is a form of linear recursion. In tail recursion, the recursive ***call*** is the ***last thing*** the function ***does***.

- Often, the ***value*** of the recursive call is ***returned***.

Introduction: Algorithms

- **Searching Techniques**
- ***Searching*** is a ***process*** of ***finding*** a particular ***data item*** from a **collection** of data items based on specific ***criteria***.

Introduction: Algorithms

- ***Linear Search***
- Linear search algorithm is the most ***simplest*** algorithm to do ***sequential*** search and

- this technique ***iterates*** over the ***sequence*** and ***checks*** one ***item*** at a time, until the ***desired item*** is found or all items have been ***examined***.

Introduction: Algorithms

- **Sorting Techniques**
- Sorting in general ***refers*** to various methods of arranging or ordering things based on criteria's (numerical, chronological, alphabetical, hierarchical etc.).

Introduction: Algorithms

- ***Bubble Sort***
- This sorting technique is also known as exchange sort, which arranges values by iterating over the list several times and

- in each iteration the larger value gets bubble up to the end of the list.

- This algorithm uses multiple passes and in each pass the first and second data items are compared.

Introduction: Algorithms

- ***Selection Sort***
- Selection sort algorithm is one of the simplest sorting algorithm, which sorts the elements in an array by finding the minimum element in each pass from unsorted part and keeps it in the beginning.

Introduction: Algorithms

- ***Insertion Sort***
- An algorithm consider the elements one at a time, inserting each in its suitable place among those already considered (keeping them sorted).

- Insertion sort is an example of an incremental algorithm.

Introduction: Algorithms

- ***Quick Sort***
- Quick sort is a divide and conquer algorithm. Quick sort first divides a large list into two smaller sub-lists:

- the ***low*** elements and the ***high*** elements.

- Quick sort can then recursively sort the sub-lists.

Introduction: Algorithms

- ***Merge Sort***
- Merge sort is based on Divide and conquer method.

- It takes the list to be sorted and divide it in half to create two unsorted lists.

- The two unsorted lists are then sorted and merged to get a sorted list.

Introduction: Algorithms

- ***Merge Sort Procedure***
  - Divides the input which we have to sort into two parts in the middle. Call it the left part and right part.
  - Sort each of them separately. Note that here sort does not mean to sort it using some other method. We use the same function recursively.
  - Then merge the two sorted parts.

1st Group Assignment

1. What is an *algorithm*?
2. List & explain *5 structures* of an algorithm.
3. Give & explain *5 properties of* an algorithm.
4. Outline & explain different approaches in designing an algorithm.
5. Define the following terms; *Recursion Algorithms & Iteration Algorithms*
6. Give *3* differences between *Recursion* and *Iteration* algorithms.
7. List & explain *2 types* of *recursion.*
8. Supply & explain 3 categories *of* recursion algorithms.
9. Define Searching in Data Structures.
10. List & define 2 search techniques in Data Structures.
11. What is Sorting in Data Structures?
12. Give & explain 5 *Sorting techniques* in Data Structures.