



ANTHONY BABAJIDE BALOGUN
ID No: **UB73361SIN82521**

COURSE TOPIC:
PROGRAMMING PRINCIPLES AND ALGORITHMS

ATLANTIC INTERNATIONAL UNIVERSITY
APRIL 2022

TABLE OF CONTENTS

1.0	INTRODUCTION	1
2.0	PROGRAMMING LANGUAGE	3
2.1	Definitions	3
3.0	TYPES OF PROGRAMMING LANGUAGE	5
3.1	Machine Language	5
3.2	Assembly Language	6
3.3	High level language	7
4.0	CATEGORIES OF PROGRAMMING LANGUAGES	9
4.1	Interpreted programming languages	9
4.2	Functional programming languages	9
4.3	Compiled Programming languages	10
4.4	Procedural programming languages	10
4.5	Scripting languages	10
4.6	Markup languages	10
4.7	Concurrent programming language	11
4.8	Object oriented programming language	11
5.0	ALGORITHMS	12
5.1	Properties of Algorithm	13
5.2	Qualities of Algorithm	13
5.3	Advantages of Algorithm	13
5.4	Disadvantages of Algorithm	13
6.0	STEPS TO DESIGNING AN ALGORITHM	14
6.1	C Programming Language	16
6.2	C++ Programming Language	18

6.3	Python3 Programming Language	20
7.0	NOTATIONS	22
7.1	FLOWCHARTS	22
7.2	Rules for drawing a flowchart	23
7.3	Advantages of flowchart	23
7.4	Disadvantages of flowchart	23
8.0	PSEUDO CODE	24
8.1	Features of Pseudo Code	24
8.2	Guidelines for writing Pseudo Code	25
8.3	Common keywords used in Pseudo Code	25
8.4	Program Samples	26
9.0	CONCLUSION	29
10.0	BIBLIOGRAPHY	31

LIST OF FIGURES

Figure 1: What is the Binary Number System	4
Figure 2: Binary Numbers	4
Figure 3: What is algorithm?	12
Figure 4: Common Flowchart Symbols	22

1.0 INTRODUCTION

A programming language is defined as a set of symbols and rules for commanding or instructing a computer to carry out or to perform specific tasks. These types of symbols and rules are designed by some experienced and knowledgeable personnel known as programmers who have to follow all the specified rules before coding them using what is known as a programming language. As we may be aware, computer only understand what is known as machine language otherwise called binary numbers which is in form of 0's and 1's.

An algorithm is defined as a step-by-step procedure or set of instructions for solving a problem or accomplishing a task. Every known computerized device makes use of algorithms to perform its functions. It makes life easier by reducing the time it takes to manually do things and also allow workers to be more proficient.

Moreover, a programmer must make use of the five basic parts of an algorithm to create a successful program. Firstly, he/she must describe or express the problem at hand in mathematical terms before creating the formulas and processes that creates results. Secondly, the programmer inputs the outcome parameters, and then he/she executes the program repeatedly to test its functionality and accuracy. The result is produced after the parameters has gone through the set of instructions in the program.

Furthermore, in this essay, I shall be writing on the Types of Programming Language; Categories of Programming Language; Steps to Designing Algorithms; Notations, Flowcharts and Pseudo Code including a program sample in one of the favourite programming languages called JAVA compared to a Pseudo Code.

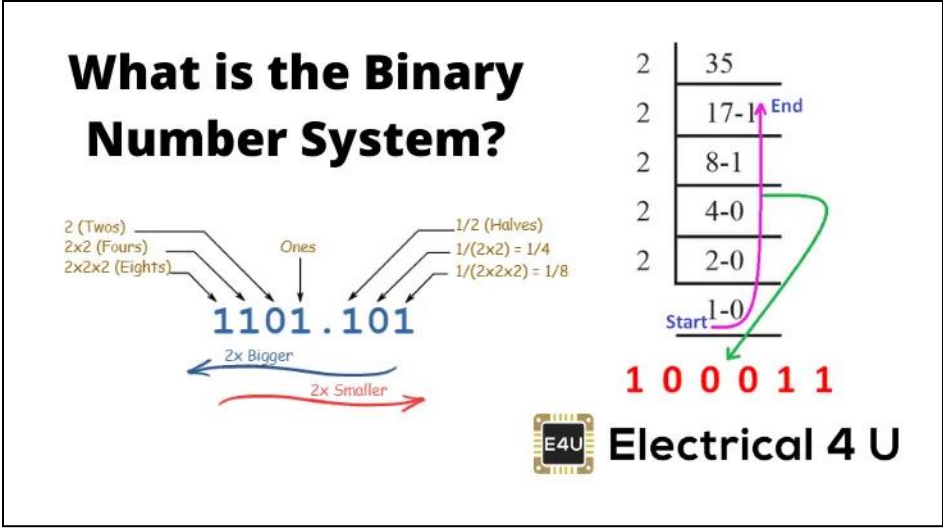
2.0 PROGRAMMING LANGUAGE

2.1 Definitions

A programming language is defined as a set of symbols and rules for commanding or instructing a computer to carry out or to perform specific tasks. These types of symbols and rules are designed by some experienced and knowledgeable personnel known as programmers who have to follow all the specified rules before coding them using what is known as a programming language. As we may be aware, computer only understand what is known as machine language otherwise called binary numbers which is in form of 0's and 1's.

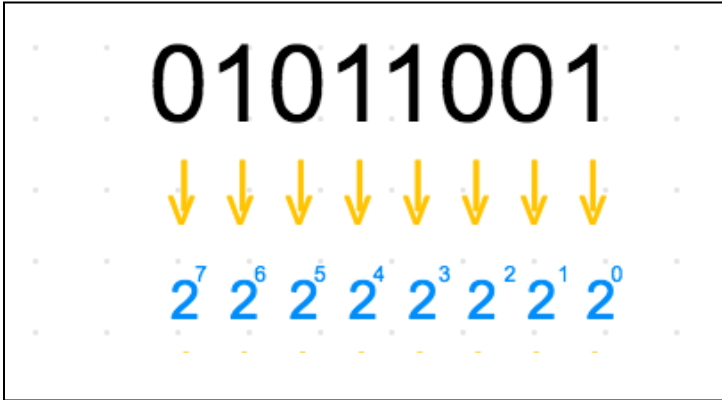
The English-like instructions or codes must first be converted into binary numbers before a computer can understand it, in other words, the user must communicate with the computer using the language (binary numbers) which it can understand.

As humans, we normally represent numbers in the decimal system. Therefore, counting the number from 1 to 10 is as simple as 1,2,3,4,5,6,7,8,9,10. On the other hand, Computers represent all information in bits. Computers use the binary number system in order to represent numbers with 0s and 1s. So, when a computer counts from 1 to 10, it looks like this: 0001, 0010, 0011, 0100, 0101, 0110, 1000, 1001, 1010, 0111.



(Electrical4U, 2020)

(Figure 1: What is the Binary Number System)



(Vivah, 2018)

(Figure 2: Binary Numbers)

3.0 TYPES OF PROGRAMMING LANGUAGE

Programming language is classified into the following types;

- Machine language
- Assembly language
- High level language

3.1 Machine Language

Computer only understand what is known as machine language otherwise called binary numbers which is in form of 0's and 1's. In order to execute any program written in any programming language, the conversion to machine language is very essential. Conversion is not required for any program written in machine language as this can be executed directly on computer. The machine language program is absolutely translation free and it saves time because it's execution is pretty fast.

Its disadvantages include;

- Hard finding errors in the machine language
- Time consuming
- Machine dependent: program developed or written for a particular computer may not run on another type of computer

3.2 Assembly Language

Assembly language was developed in order to make the programming process easier. It is logically equivalent to machine language but the only added advantage is that, it is easier and more convenient for people to read, write and better to understand.

It uses symbolic notation to represent machine language instructions which are called low level language because they are relatively closer to the machines. An assembler translates assembly language instructions into a machine language which makes it easier to understand and use. Unlike machine language, errors are easily detected and located.

Assembly language comes with a few disadvantages though and they include;

- Machine dependent - the program which can be executed on a particular machine depends solely on the architecture of that particular computer.
- Assembly language is considered to be hard to learn
- Programmer needs to have the hardware knowledge to create applications
- It is less efficient
- Execution time is more than machine language program
- Assembler is needed for conversion to machine language

3.3 High level language

It consists of 'English-like words' and 'rules symbols' which are to be conformed with while coding a program. Furthermore, interpreter or compilers which translate high level language into machine language are required in order to convert these programs into machine readable form.

A compiler reads the whole program written in high level language at a go and translates it to machine language in lumpsum and if any error is encountered, such error will be displayed on the computer screen. Interpreter on the other hand reads and translates high level language program in line-by-line manner. It translates statement from a source code into a machine code at a time, and runs it before translating the next statement. The execution of the program is stopped when an error is encountered and such error message is displayed on the computer screen.

Its advantages include;

- Readability
- Easier to learn and understand
- Portability between machines.
- Easy debugging
- Easy to find and correct errors

Disadvantages

- Less efficient
- More execution times

4.0 CATEGORIES OF PROGRAMMING LANGUAGES

Programming language is classified into the following Categories:

- Interpreted programming languages
- Functional programming languages
- Compiled programming languages
- Procedural programming languages
- Scripting programming language
- Markup programming languages
- Concurrent programming languages
- Object oriented programming languages

4.1 Interpreted programming languages

Under this programming language, most of its instructions executes directly without previously compiling a program into machine language instructions. Examples include; Pascal and Python programming languages

4.2 Functional programming languages

This defines every computation bound to mathematical calculations as a mathematical evaluation. Examples include; Clean and Haskel programming languages

4.3 Compiled Programming languages

This is a programming language whose execution are typically run by compilers and not interpreters. It generates a machine code from the source code. Examples include; "C", "C++", "C#" and "JAVA"

4.4 Procedural programming languages

It specifies the steps that the programs should take in order to get to an intended state. Furthermore, it can be referred through a procedure call which makes the programs to be structured. Examples include; Hyper talk and MATLAB

4.5 Scripting languages

These are programming languages which controls an application and can execute independent of any other application. Mostly embedded in the application that they control and are used to automate the communication with external program. Examples include; Apple script and VB script

4.6 Markup languages

This is known as an artificial language that uses annotations to text that shows hoe the text is to be displayed. Examples include; HTML and XML

4.7 Concurrent programming language

It provides for the implementation of operation concurrently. This could be either within a single computer or across multiple systems. Examples include; Joule and Limbo

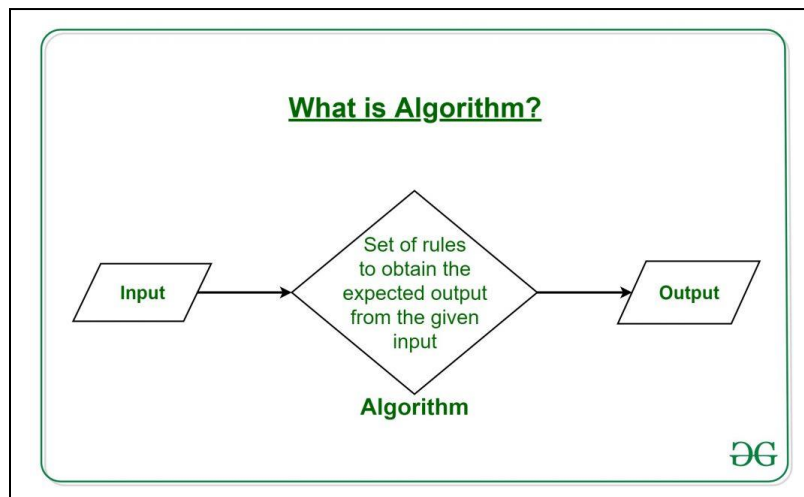
4.8 Object oriented programming language

This is paradigm based on the concept of objects which may contain data in the form of procedures commonly known as methods. Examples include; Lava and Moto

5.0 ALGORITHMS

An algorithm is defined as a step-by-step procedure or set of instructions for solving a problem or accomplishing a task. Every known computerized device makes use of algorithms to perform its functions. It makes life easier by reducing the time it takes to manually do things and also allow workers to be more proficient.

Furthermore, a programmer must make use of the five basic parts of an algorithm to create a successful program. Firstly, he/she must describe or express the problem at hand in mathematical terms before creating the formulas and processes that creates results. Secondly, the programmer inputs the outcome parameters, and then he/she executes the program repeatedly to test its functionality and accuracy. The result is produced after the parameters has gone through the set of instructions in the program.



(GeeksforGeeks, 2022)

(Figure 3: What is algorithm?)

5.1 Properties of Algorithms

- Should be written in simple English
- Should be precise and unambiguous
- Instructions should not be repeated infinitely
- Should conclude after a finite number of steps
- Should have an end point
- Derived results should be obtained only after the algorithm terminates

5.2 Qualities of Algorithm

- Time
- Memory
- Accuracy

5.3 Advantages of Algorithms

- It is easy to understand
- Step-wise representation of a solution to a given problem
- Problem is broken down into smaller pieces

5.4 Disadvantages of Algorithms

- Time-consuming
- Branching and Looping statements are difficult to show in Algorithms

6.0 STEPS TO DESIGNING AN ALGORITHM

The following pre-requisite are required in order to write an algorithm:

- Identify the problem this algorithm will solve
- Identify the constraints of the problem
- Determine the input to be accepted in order to solve the problem
- Consider the output to be generated after the problem has been solved
- Determine the solution to this problem

Let us consider the steps to be taken in a simple algorithm to add three numbers and then print the sum either on a paper as a hard copy or on the screen as a soft copy.

Step 1

- Identify the problem this algorithm will solve: *Add 3 numbers and print their sum*
- Identify the constraints of the problem: *digits only*
- Determine the input to be accepted in order to solve the problem: *Add 3 digits*
- Consider the output to be generated after the problem has been solved: *addition of 3 digits accepted as the input*
- Determine the solution to this problem: *adding 3 digits by making use of the plus (+) operator*
- Consider the algorithm to be written in order to solves the problem: *algorithm to be written in 3 different programming languages*

Step 2

Designing the algorithm with the help of above pre-requisites

START

- Declare 3 integer variables ***n1, n2 and n3***
- Accept the three numbers to be added as inputs in variables ***n1, n2 and n3***
- Declare integer variable ***res*** to store the summation of the 3 numbers
- Add the 3 numbers and then store the result in the variable ***res***
- Then print the value of variable ***res***

END

Step 3

In order to test the algorithm, I will write codes to implement it in three different programming language namely; ***C, C++ and Python3***

6.1 C Programming Language

// C program to add three numbers with the help of the above designed algorithm
in step 2

```
#include <stdio.h>

int main()
{
    // Variables to take the input of the 3 numbers
    int n1, n2, n3;

    // Variable to store the resultant sum
    int res;

    // Accept the 3 numbers as input
    printf("Enter the 1st number: ");
    scanf("%d", &n1);
    printf("%d\n", n1);
    printf("Enter the 2nd number: ");
    scanf("%d", &n2);
    printf("%d\n", n2);
    printf("Enter the 3rd number: ");
    scanf("%d", &n3);
    printf("%d\n", n3);
```

```
// Calculate the sum using the plus (+) operator
// and store it in variable res
res = n1 + n2 + n3;
// Print the sum
printf("\nSum of the 3 numbers is: %d", res);
return 0;
}
```

Output

Enter the 1st number: 10

Enter the 2nd number: 30

Enter the 3rd number: 20

Sum of the 3 numbers is: 60

6.2 C++ Programming Language

// C++ program to add three numbers with the help of above designed algorithm in step 2

```
#include <bits/stdc++.h>

using namespace std;

int main()
{
    // Variables to take the input of the 3 numbers
    int n1, n2, n3;

    // Variable to store the resultant sum
    int res;

    // Accept the 3 numbers as input
    cout << "Enter the 1st number: ";
    cin >> n1;
    cout << " " << n1 << endl;
    cout << "Enter the 2nd number: ";
    cin >> n2;
    cout << " " << n2 << endl;
    cout << "Enter the 3rd number: ";
    cin >> n3;
    cout << " " << n3;
```

```
// Calculate the sum using the plus (+) operator and then store it in variable
    res

    res = n1 + n2 + n3;

// Print the sum

    cout << "\nSum of the 3 numbers is: "

    << res;

    return 0;

}
```

Output

Enter the 1st number: 10

Enter the 2nd number: 30

Enter the 3rd number: 20

Sum of the 3 numbers is: 60

6.3 Python3 Programming Language

Python3 program to add three numbers with the help of above designed

Algorithm in step 2

```
if __name__ == "__main__":  
  
    # Variables to take the input of the 3 numbers  
  
    n1 = n2 = n3 = 0  
  
    # Variable to store the resultant sum  
  
    res = 0  
  
    # Accept the 3 numbers as input  
  
    n1 = int(input("Enter the 1st number: "))  
    n2 = int(input("Enter the 2nd number: "))  
    n3 = int(input("Enter the 3rd number: "))  
  
    # Calculate the sum using the plus (+) operator and store it in variable sum  
  
    res = n1 + n2 + n3  
  
    # Print the sum  
  
    print("\nSum of the 3 numbers is:", res)
```


Output

Enter the 1st number: 10

Enter the 2nd number: 30

Enter the 3rd number: 20




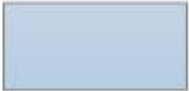

Sum of the 3 numbers is: 60

7.0 NOTATIONS

7.1 FLOWCHARTS

According to Smartdraw, “Flowcharts use special shapes to represent different types of actions or steps in a process. Lines and arrows show the sequence of the steps, and the relationships among them. These are known as flowchart symbols”.

(Smartdraw, n.d.)

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

(Smartdraw, Common Flowchart Symbols, n.d.)

(Figure 4: Common Flowchart Symbols)

7.2 Rules for drawing a flowchart

- Should be clear and easy to follow
- Must have a logical start and finish
- A process symbol should have one flow line
- A decision symbol should have one flow line
- A decision symbol may also have two or three flow lines
- A terminal symbol should have only one flow line
- Brief and precise description
- No intersection of flow lines

7.3 Advantages of flowchart

- Better Communication
- Effective Analysis
- Proper Documentation
- Efficient Coding
- Proper Debugging
- Efficient Program Maintenance

7.4 Disadvantages of flowchart

- Complex logic
- Alterations and Modifications
- Reproduction
- Too Costly

8.0 PSEUDO CODE

According to Geeksforgeeks.org, “Pseudo code is a term which is often used in programming and algorithm based fields. It is a methodology that allows the programmer to represent the implementation of an algorithm. Simply, we can say that it’s the cooked up representation of an algorithm. Often at times, algorithms are represented with the help of pseudo codes as they can be interpreted by programmers no matter what their programming background or knowledge is. Pseudo code, as the name suggests, is a false code or a representation of code which can be understood by even a layman with some school level programming knowledge”. (Geeksforgeeks.org, 2021)

8.1 Features of Pseudo Code

- Consists of short and readable styled English languages
- No variable declaration and subroutines
- Programmer or non-programmer can easily understand it
- Sketch of the program before actual coding
- Not machine readable
- Cannot be compiled and executed
- No standard syntax

8.2 Guidelines for writing Pseudo Code

- Write a statement per line
- Initial keyword must be capitalized
- Indentation to hierarchy
- Multiline structure must be terminated
- Statements language must be independent

8.3 Common keywords used in Pseudo Code

- `//`: represents a comment
- `BEGIN`, `END`: begins the first statement and ends the last statement
- `INPUT`, `GET`, `READ`: data inputs keywords
- `COMPUTE`, `CALCULATE`: used to calculate the result of a given expression
- `ADD`, `SUBTRACT`, `INITIALIZE`: used to add, subtract and initialize
- `OUTPUT`, `PRINT`, `DISPLAY`: used to display the output of the program
- `IF`, `ELSE`, `ENDIF`: decision making keywords
- `WHILE`, `ENDWHILE`: iterative statements keywords
- `FOR`, `ENDFOR`: iterative incremented/decremented keywords

8.4 Program Samples

// Program to calculate the lowest common multiple for excessively

long input values in **JAVA**

```
import java.util.*;
```

```
public class LowestCommonMultiple {
```

```
    private static long
```

```
        lcmNaive(long numberOne, long numberTwo)
```

```
    {
```

```
        long lowestCommonMultiple;
```

```
        lowestCommonMultiple
```

```
            = (numberOne * numberTwo)
```

```
              / greatestCommonDivisor(numberOne,
```

```
                                      numberTwo);
```

```
        return lowestCommonMultiple;
```

```
    }
```

```
    private static long
```

```
        greatestCommonDivisor(long numberOne, long numberTwo)
```

```
    {
```

```
        if (numberTwo == 0)
```

```
            return numberOne;
```

```
        return greatestCommonDivisor(numberTwo,
```

```

        numberOne % numberTwo);
    }

    public static void main(String args[])
    {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the inputs");

        long numberOne = scanner.nextInt();

        long numberTwo = scanner.nextInt();

        System.out.println(lcmNaive(numberOne, numberTwo));
    }
}

```

Pseudo Code

```

// Program to calculate the lowest common multiple for excessively
long input values in Pseudo Code
function lcmNaive(Argument one, Argument two)
{
    Calculate the lowest common variable of Argument
    1 and Argument 2 by dividing their product by their
    Greatest common divisor product
    return lowest common multiple
end
}

```

```
function greatestCommonDivisor(Argument one, Argument two)
{
  if Argument two is equal to zero
  then return Argument one
  return the greatest common divisor
end
}
```


9.0 CONCLUSION

To be a successful software developer or programmer, an individual must possess the skills of logical thoughts. This however takes considerable amount of practices with a self-motivation much like a hobby. In my own opinion, today's software engineering is much greater compared to that of about twenty years ago. Presently, not only must you program an algorithm successfully, a Windows interface must also be taken into consideration and well designed.

Moreover, teaching and learning programming on a personal computer requires a Windows development environment which include; Visual Basic tools, C, C++, or even FORTRAN. Visual Basic, C#, Python presents the best and the most convenient environment to design Windows application interfaces in today's programming world.

Consequently, many students are now being taught using various commercial programs such as Microsoft's Excel spreadsheet program. Matlab is another popular commercial program which requires students to execute and implement pseudo programming steps in order to obtain an answer to a problem.

Furthermore, how can an individual be motivated to become a software developer in this highly modern computer environment?

I am particularly concerned about the fact that; the number of talented software developers is diminishing whilst the reliance on commercial and Windows-based software grows.

Our dependency on commercial software today is too heavy and I am beginning to wonder, is there is a pool of talented programmers who have developed their skills through personal learning somewhere?, but I am nowhere closer to any answer to this important question.

10.0 BIBLIOGRAPHY

Electrical4U. (2020, October 22). *Binary Number System: What is it? (Definition & Examples)*.

Retrieved from What is the Binary Number System?:

<https://www.electrical4u.com/binary-number-system-binary-to-decimal-and-decimal-to-binary-conversion/>

GeeksforGeeks. (2022, January 13). *Introduction to Algorithms*. Retrieved from What is

Algorithm? Algorithm Basics: <https://www.geeksforgeeks.org/introduction-to-algorithms/>

Geeksforgeeks.org. (2021, Sept 30). *How to write a Pseudo Code?* Retrieved from How to write

a Pseudo Code?: <https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/>

Smartdraw. (n.d.). *Common Flowchart Symbols*. Retrieved from Flowchart Symbols:

<https://www.smartdraw.com/flowchart/flowchart-symbols.htm#:~:text=Flowcharts%20use%20special%20shapes%20to,are%20known%20as%20flowchart%20symbols.>

Smartdraw. (n.d.). *Flowchart Symbols*. Retrieved from Flowchart Symbols:

<https://www.smartdraw.com/flowchart/flowchart-symbols.htm#:~:text=Flowcharts%20use%20special%20shapes%20to,are%20known%20as%20flowchart%20symbols.>

Vivah, L. (2018, June 22). *Learn How to Read Binary in 5 minutes*. Retrieved from Learn How

to Read Binary in 5 minutes: <https://medium.com/@LindaVivah/learn-how-to-read-binary-in-5-minutes-dac1feb991e>