CSCI 125

Instructor Azhar Bhatt

Machine

A mechanically, electrically, or electronically operated device for performing a task

What is a computer?

A computer is an electronic device that manipulates information, or data. It has the ability to store, retrieve, and process data. You may already know that you can use a computer to type documents, send email, play games, and browse the Web. You can also use it to edit or create spreadsheets, presentations, and even videos.

Hardware

Hardware is any part of your computer that has a physical structure, such as the keyboard or mouse. It also includes all of the computer's internal parts.

Software

Software is any **set of instructions** that tells the hardware **what to do** and **how to do it**. Examples of software include web browsers, games, and word processors.

Everything you do on your computer will rely on both hardware and software.

CPU – or Central Processing Unit is considered the most important component in a computer and for good reason. It handles most operations that make it function, by processing instructions and giving signals out to other components. The CPU is the main bridge between all the computer's major parts.

RAM – Random Access Memory, or RAM for short, is a computer component where data used by the operating system and software applications store data so that the CPU can process them quickly. Everything stored on RAM is lost if the computer is shut off. Depending on the applications you use, there is typically a maximum limit of RAM you will need for the computer to function properly.

HDD – Also known as Hard Disk Drive, it is the component where photos, apps, documents and such are kept. Although they are still being used, we have much faster types of storage devices such as solid state drives (SSD) that are also more reliable.

Motherboard – There is no acronym for this component but without it, there can't be a computer. The Motherboard acts as the home for all other components, allows them to communicate with each other and gives them power in order to function. There are components that don't require a physical connection to the Motherboard in order to work, such as Bluetooth or Wi-Fi but, if there is no connection or signal what so ever, the computer won't know it's there.

How does it Work?

CPU Components

The Control Unit

The control unit of the CPU contains circuitry that uses electrical signals to direct the entire computer system to carry out, or execute, stored program instructions. Like an orchestra leader, the control unit does not execute program instructions; rather, it directs other parts of the system to do so. The control unit must communicate with both the arithmetic/logic unit and memory.

The Arithmetic/Logic Unit

The arithmetic/logic unit (ALU) contains the electronic circuitry that executes all arithmetic and logical operations.The arithmetic/logic unit can perform four kinds of arithmetic operations, or mathematical calculations: addition, subtraction, multiplication, and division. As its name implies, the arithmetic/logic unit also performs logical operations. A logical operation is usually a comparison. The unit can compare numbers, letters, or special characters. The computer can then take action based on the result of the comparison. This is a very important capability. It is by comparing that a computer is able to tell, for instance, whether there are unfilled seats on airplanes, whether charge- card customers have exceeded their credit limits, and whether one candidate for Congress has more votes than another.

Logical operations can test for three conditions:

Equal-to condition. In a test for this condition, the arithmetic/logic unit compares two values to determine if they are equal. For example: If the number of tickets sold equals the number of seats in the auditorium, then the concert is declared sold out.

Less-than condition. To test for this condition, the computer compares values to determine if one is less than another. For example: If the number of speeding tickets on a driver's record is less than three, then insurance rates are $425; otherwise, the rates are $500.

Greater-than condition. In this type of comparison, the computer determines if one value is greater than another. For example: If the hours a person worked this week are greater than 40, then multiply every extra hour by 1.5 times the usual hourly wage to compute overtime pay.

A computer can simultaneously test for more than one condition. In fact, a logic unit can usually discern six logical relationships: equal to, less than, greater than, less than or equal to, greater than or equal to, and not equal.

- **Registers: Temporary Storage Areas**
  Registers are temporary storage areas for instructions or data. They are not a part of memory; rather they are special additional storage locations that offer the advantage of speed. Registers work under the direction of the control unit to accept, hold, and transfer instructions or data and perform arithmetic or logical comparisons at high speed. The control unit uses a data storage register the way a store owner uses a cash register-as a temporary, convenient place to store what is used in transactions.

  Computers usually assign special roles to certain registers, including these registers:

  - **An accumulator**, which collects the result of computations.
  - **An address register**, which keeps track of where a given instruction or piece of data is stored in memory. Each storage location in memory is identified by an address, just as each house on a street has an address.
  - **A storage register**, which temporarily holds data taken from or about to be sent to memory.
  - A general-purpose **register**, which is used for several functions.

- **Memory and Storage**
  Memory is also known as primary storage, primary memory, main storage, internal storage, main memory, and RAM (Random Access Memory); all these terms are used interchangeably by people in computer circles. Memory is the part of the computer that holds data and instructions for processing. Although closely associated with the central processing unit, memory is separate from it. Memory stores program instructions or data for only as long as the program they pertain to is in operation. Keeping these items in memory when the program is not running is not feasible for three reasons:

  - Most types of memory only store items while the computer is turned on; data is destroyed when the machine is turned off.
  - If more than one program is running at once (often the case on large computers and sometimes on small computers), a single program can not lay exclusive claim to memory.
  - There may not be room in memory to hold the processed data.

Before an instruction can be executed, program instructions and data must be placed into memory from an input device or a secondary storage device (the process is further complicated by the fact that, as we noted earlier, the data will probably make a temporary stop in a register). As Figure 2 shows, once the necessary data and instruction are in memory, the central processing unit performs the following four steps for each instruction:

The control unit fetches (gets) the instruction from memory.

The control unit decodes the instruction (decides what it means) and directs that the necessary data be moved from memory to the arithmetic/logic unit. These first two steps together are called instruction time, or I-time.

The arithmetic/logic unit executes the arithmetic or logical instruction. That is, the ALU is given control and performs the actual operation on the data.

Thc arithmetic/logic unit stores the result of this operation in memory or in a register. Steps 3 and 4 together are called execution time, or E-time.

Program

As a verb, to program a computer is the writing of statements or commands that instruct the computer how to process data. There are several programming languages used to program a computer.

As a noun, a program, also called an application or software, is a set of instructions that process input, manipulate data, and output a result. For example, Microsoft Word is a word processing application that allows users to create and write documents.

Operating system

An operating system or OS is a software installed on a computer's hard drive that enables the computer hardware to communicate and operate with the computer software. Without a computer operating system, a computer and software programs would be useless. The picture shows Microsoft Windows XP in its original packaging.

When computers were first introduced, the user interacted with them using a command line interface, which required commands. Today, almost every computer is using a GUI (Graphical User Interface) operating system that's easy to use and operate.

The first program

The first software program that was held in electronic memory was written by Tom Kilburn. The program calculated the highest factor of the integer 2^{18} = 262,144, and was successfully executed on June 21, 1948, at the University of Manchester, England. The computer was called the SSEM (Small Scale Experimental Machine), otherwise known as the "Manchester Baby.". This event is widely celebrated as the birth of software.

What are Computer Programming Languages?

Computer programming languages allow us to give instructions to a computer in a language the computer understands. Just as many human-based languages exist, there are an array of computer programming languages that programmers can use to communicate with a computer. The portion of the language that a computer can understand is called a “binary.” Translating programming language into binary is known as “compiling.” Each language, from C Language to Python, has its own distinct features, though many times there are commonalities between programming languages.

These languages allow computers to quickly and efficiently process large and complex swaths of information. For example, if a person is given a list of randomized numbers ranging from one to ten thousand and is asked to place them in ascending order, chances are that it will take a sizable amount of time and include some errors

Binary

Binary is a base-2 number system invented by Gottfried Leibniz that is made up of only two numbers: 0 and 1. This number system is the basis for all binary code, which is used to write data such as the computer processor instructions used every day

The 0s and 1s in binary represent OFF or ON, respectively. In a transistor, an "0" represents no flow of electricity, and "1" represents electricity being allowed to flow. In this way, numbers are represented physically inside the computing device, permitting calculation

Binary is the primary language for computers for the following reasons.

It is a simple and elegant design.

Binary's 0 and 1 method is quick to detect an electrical signal's off or on state.

The positive and negative poles of magnetic media are quickly translated into binary.

Binary is the most efficient way to control logic circuits.

Machine language

Sometimes referred to as machine code or object code, machine language is a collection of binary digits or bits that the computer reads and interprets. Machine language is the only language a computer is capable of understanding.

The exact machine language for a program or action can differ by operating system. The specific operating system dictates how a compiler writes a program or action into machine language.

Below is an example of machine language (binary) for the text "Hello World.

```
01001000 01100101 01101100 01101100 01101111 00100000 01010111 01101111 01110010
01101100 01100100
```

Assembly language

Machine language is a series of numbers, which is not easy for humans to read. Using ASM, programmers can write human-readable programs that correspond almost exactly to machine language.

The disadvantage is that everything the computer does must be described explicitly, in precise detail. The advantage is that the programmer has maximum control over what the computer is doing.

Assembly is called a low-level programming language because there's (nearly) a one-to-one relationship between what it tells the computer to do, and what the computer does. In general, one line of an assembly program contains a maximum of one instruction for the computer. Assembly languages are tied to one specific computer architecture, they are not portable.

Here is "Hello, World" written for a 32-bit Intel processor.

```
        global  _main
        extern  _printf
          section .text
    _main:
        push    message
        call    _printf
        add     esp, 4
        Ret
Message:
    db  'Hello, World!', 10, 0
```

How does it all work

A computer doesn’t actually understand the phrase ‘Hello, world!’, and it doesn’t know how to display it on screen. It only understands on and off. So to actually run a command like print 'Hello, world!', it has to translate all the code in a program into a series of ons and offs that it can understand.

To do that, a number of things happen:

The source code is translated into assembly language.

The assembly code is translated into machine language.

The machine language is directly executed as binary code.

Confused? Let’s go into a bit more detail. The coding language first has to translate its source code into assembly language, a super low-level language that uses words and numbers to represent binary patterns. Depending on the language, this may be done with an interpreter (where the program is translated line-by-line), or with a compiler (where the program is translated as a whole).

The coding language then sends off the assembly code to the computer’s assembler, which converts it into the machine language that the computer can understand and execute directly as binary code.

High Level Language

High-level languages are very important, as they help in developing complex software and they have the following advantages −

- Unlike assembly language or machine language, users do not need to learn the high-level language in order to work with it.
- High-level languages are similar to natural languages, therefore, easy to learn and understand.
- High-level language is designed in such a way that it detects the errors immediately.
- High-level language is easy to maintain and it can be easily modified.
- High-level language makes development faster.
- High-level language is comparatively cheaper to develop.
- High-level language is easier to document.

What Is a Programming Compiler?

A compiler is a software program that converts computer programming code written by a human programmer into binary code (machine code) that can be understood and executed by a specific CPU. The act of transforming source code into machine code is called "compilation." When all the code is transformed at one time before it reaches the platforms that run it, the process is called ahead-of-time (AOT) compilation.

WHAT IS AN IDE?

n IDE, or Integrated Development Environment, enables programmers to consolidate the different aspects of writing a computer program.

IDEs increase programmer productivity by combining common activities of writing software into a single application: editing source code, building executables, and debugging.

EDITING SOURCE CODE

Writing code is an important part of programming. We start with a blank file, write a few lines of code, and a program is born! IDEs facilitate this process with features like syntax highlighting and autocomplete.

SYNTAX HIGHLIGHTING

An IDE that knows the syntax of your language can provide visual cues. Keywords, words that have special meaning like `class` in Java, are highlighted with different colors.

virtual machine (VM)

A virtual machine (VM) is an operating system (OS) or application environment that is installed on software, which imitates dedicated hardware. The end user has the same experience on a virtual machine as they would have on dedicated hardware.

Specialized software, called a hypervisor, emulates the PC client or server's CPU, memory, hard disk, network and other hardware resources completely, enabling virtual machines to share the resources. The hypervisor can emulate multiple virtual hardware platforms that are isolated from each other, allowing virtual machines to run Linux and Windows Server operating systems on the same underlying physical host. Virtualization limits costs by reducing the need for physical hardware systems. Virtual machines more efficiently use hardware, which lowers the quantities of hardware and associated maintenance costs, and reduces power and cooling demand. They also ease management because virtual hardware does not fail. Administrators can take advantage of virtual environments to simplify backups, disaster recovery, new deployments and basic system administration tasks.

Java

Java programming language was originally developed by Sun Microsystems which was initiated by James Gosling and released in 1995 as core component of Sun Microsystems' Java platform (Java 1.0 [J2SE]).

The latest release of the Java Standard Edition is Java SE 8. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

The new J2 versions were renamed as Java SE, Java EE, and Java ME respectively. Java is guaranteed to be Write Once, Run Anywhere.

- Object Oriented − In Java, everything is an Object. Java can be easily extended since it is based on the Object model.
- Platform Independent − Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into platform specific machine, rather into platform independent byte code. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.
- Simple − Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.
- Secure − With Java's secure feature it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.
- Architecture-neutral − Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of Java runtime system.
- Portable − Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. Compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.
- Robust − Java makes an effort to eliminate error prone situations by emphasizing mainly on compile time error checking and runtime checking.
- Multithreaded − With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.
- Interpreted − Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since the linking is an incremental and light-weight process.
- High Performance − With the use of Just-In-Time compilers, Java enables high performance.
- Distributed − Java is designed for the distributed environment of the internet.
- Dynamic − Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry extensive amount of run-time information that can be used to verify and resolve accesses to objects on run-time.

JDK, JRE, and JVM

JVM

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be executed. It can also run those programs which are written in other languages and compiled to Java bytecode.

JVMs are available for many hardware and software platforms. JVM, JRE, and JDK are platform dependent because the configuration of each OS is different from each other. However, Java is platform independent. There are three notions of the JVM: *specification*, *implementation*, and *instance*.

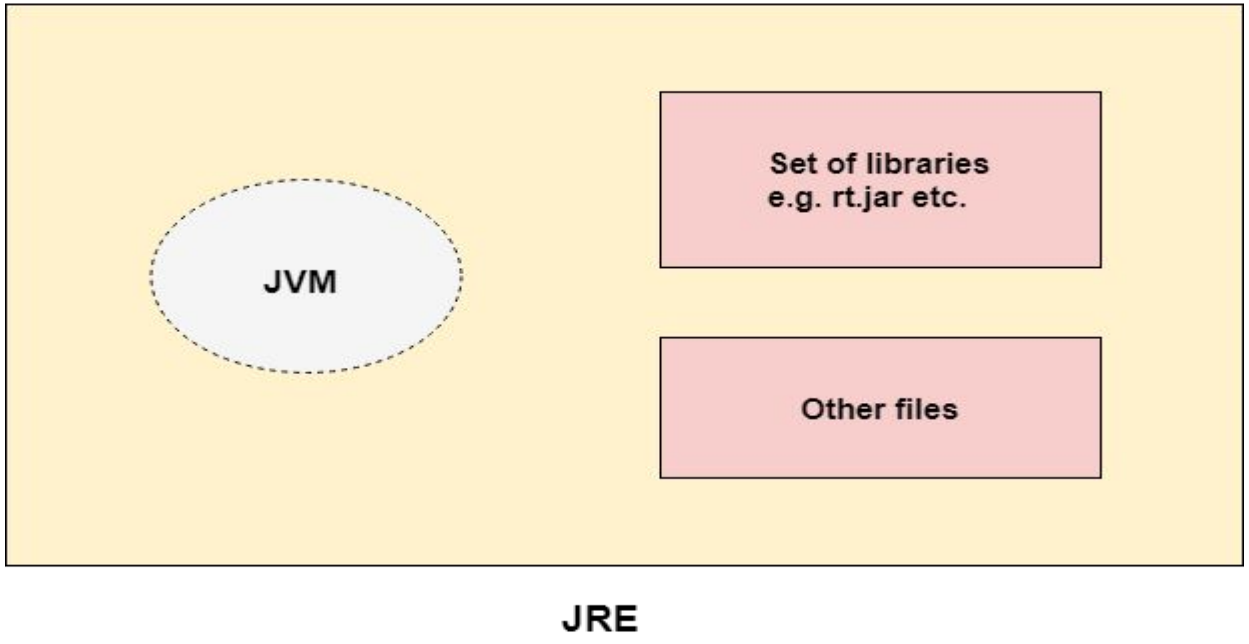
The JVM performs the following main tasks:

- Loads code
- Verifies code
- Executes code
- Provides runtime environment

JRE

JRE is an acronym for Java Runtime Environment. It is also written as Java RTE. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

The implementation of JVM is also actively released by other companies besides Sun Micro Systems.



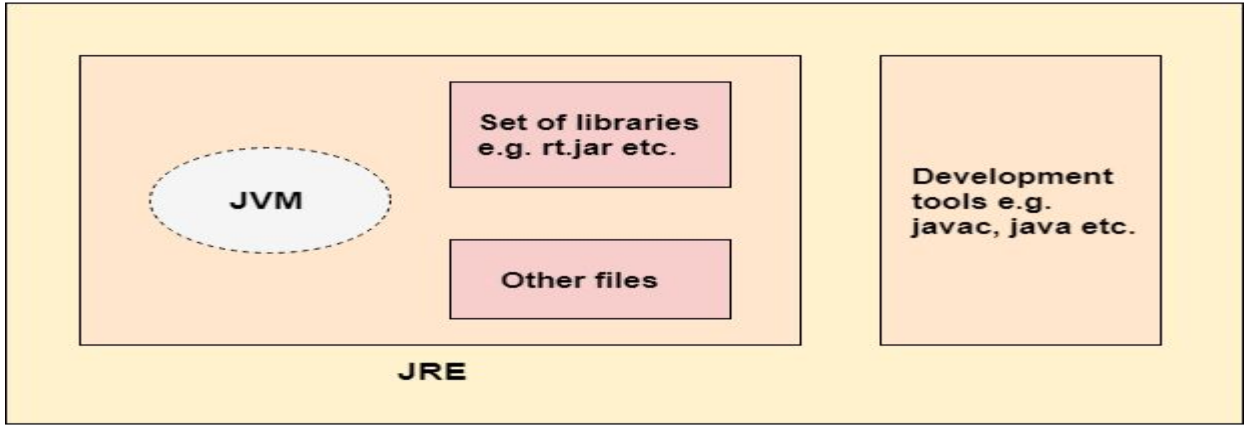
JDK

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools.

JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

- Standard Edition Java Platform
- Enterprise Edition Java Platform
- Micro Edition Java Platform

The JDK contains a private Java Virtual Machine (JVM) and a few other resources such as an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (Javadoc), etc. to complete the development of a Java Application.


JVM
Set of libraries e.g. rt.jar etc.
Other files
JRE
Development tools e.g. javac, java etc.

JDK

Algorithm

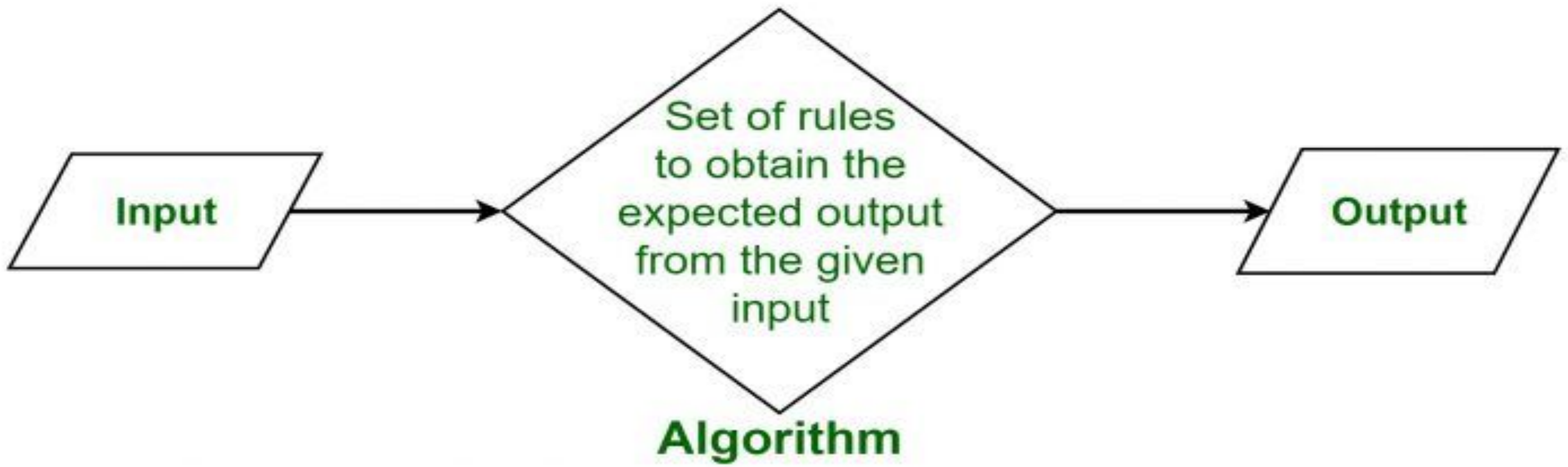
An **algorithm** is a finite list of instructions, most often used in solving problems or performing tasks. You may have heard the term used in some fancy context about a genius using an algorithm to do something highly complex, usually in programming. Indeed, you've most likely heard the term used to explain most things related to computer processes. However, what would you say if I was to tell you that there is a very good chance that you, yourself, have followed an algorithm? You may have followed some algorithms hundreds or thousands of times!

A Real Life Algorithm

Have you ever baked or cooked something? One of the most obvious examples of an algorithm is a recipe. It's a finite list of instructions used to perform a task. For example, if you were to follow the algorithm to create brownies from a box mix, you would follow the three to five step process written on the back of the box.

One of the attributes of an algorithm is that, since it is a list of instructions, there is some step-by-step process that occurs in order. Very often, the order that the steps are given in can make a big difference. Suppose we were to reorder the steps of the recipe on the back of the brownie box and told somebody to put the brownies in the oven for 22 minutes before we told them to preheat the oven. That would be silly! That's why the ordering of the steps is very important.

What is Algorithm?



Input
Set of rules to obtain the expected output from the given input
Algorithm
Output

In order for some instructions to be an algorithm, it must have the following characteristics:

- **Clear and Unambiguous**: Algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.
- **Well-Defined Inputs**: If an algorithm says to take inputs, it should be well-defined inputs.
- **Well-Defined Outputs:** The algorithm must clearly define what output will be yielded and it should be well-defined as well.
- **Finite-ness:** The algorithm must be finite, i.e. it should not end up in an infinite loops or similar.
- **Feasible:** The algorithm must be simple, generic and practical, such that it can be executed upon will the available resources. It must not contain some future technology, or anything.
- **Language Independent:** The Algorithm designed must be language-independent, i.e. it must be just plain instructions that can be implemented in any language, and yet the output will be same, as expected.

How to Design an Algorithm?

Inorder to write an algorithm, following things are needed as a pre-requisite:

1. The **problem** that is to be solved by this algorithm.
2. The **constraints** of the problem that must be considered while solving the problem.
3. The **input** to be taken to solve the problem.
4. The **output** to be expected when the problem the is solved.
5. The **solution** to this problem, in the given constraints.

Example: Consider the example to add three numbers and print the sum.

Step 1: Fulfilling the pre-requisites

As discussed above, in order to write an algorithm, its pre-requisites must be fulfilled.

1. **The problem that is to be solved by this algorithm**: Add 3 numbers and print their sum.
2. **The constraints of the problem that must be considered while solving the problem**: The numbers must contain only digits and no other characters.
3. **The input to be taken to solve the problem:** The three numbers to be added.
4. **The output to be expected when the problem the is solved:** The sum of the three numbers taken as the input.
5. **The solution to this problem, in the given constraints:** The solution consists of adding the 3 numbers. It can be done with the help of ‘+’ operator, or bit-wise, or any other method.

Step 2: Designing the algorithm

Designing the algorithm

Now let’s design the algorithm with the help of above pre-requisites:

Algorithm to add 3 numbers and print their sum:

1. START
2. Declare 3 integer variables num1, num2 and num3.
3. Take the three numbers, to be added, as inputs in variables num1, num2, and num3 respectively.
4. Declare an integer variable sum to store the resultant sum of the 3 numbers.
5. Add the 3 numbers and store the result in the variable sum.
6. Print the value of variable sum
7. END

Step 3: Testing the algorithm by implementing it.

Follow each step

See the ouput

Home work 1

Set Up Java environment

https://www.tutorialspoint.com/java/java_environment_setup.htm

Install IDE

Eclipse − A Java IDE developed by the eclipse open-source community and can be downloaded from https://www.eclipse.org

Refrences

https://homepage.cs.uri.edu/faculty/wolfe/book/Readings/Reading04.htm

https://edu.gcfglobal.org/en/computerbasics/understanding-operating-systems/1/

https://www.computerhope.com/jargon/p/program.htm

https://tldp.org/HOWTO/Unix-and-Internet-Fundamentals-HOWTO/languages.html#:~:text=Compiled%20languages,at%20the%20source%20code%20again.

https://www.computerscience.org/resources/computer-programming-languages/

https://www.computerhope.com/jargon/b/binary.htm