UNIT 1

Programming Languages and Problem solving

Programming Language

A programming language is a standardized communication technique for describing instructions for a computer. Each programming language has a set of syntactic and semantic rules used to define computer programs.

A programming language enables a programmer to precisely specify what data computer is act upon, how these data are to be stored/transmitted and what actions are to be taken under circumstances.

Programming language are classified mainly in two categories:

i) Low level programming language
ii) High level programming language

i) Low level programming language

Low level language are specific to hardware. Before creating a program, it is required to have through knowledge of that hardware. low level programming language are divided into two types.

- Machine language
- Assembly language

Machine language

Machine language are lowest- level programming language. A computer understands program written only in the machine language. It is directly executable by computer without the need for translation by a compiler or an assembler.

Machine code consist entirely of the 0’s and 1’s of the binary system. Early computers were programmed using machine language. Programs written in machine language are faster and efficient.

Writing program in machine language is very tedious, time consuming, difficult to find bugs in longer programs.

Assembly language

In assembly language, instead of using numeric opcodes (i.e. pattern of 0 &1),mnemonics are used eg. ADD,SUB etc. Program written in assembly language must be converted into machine language which could be done by assembler. Assembly language program written for one type of CPU won’t run on another. So that assembly language is time consuming and machine dependent.

ii) High Level programming language

The syntax of high level is closer to human language. High level language was developed to make programming easier. Most of the high-level language are English like languages. They use familiar English words, Special symbols (_ , & etc.) in their syntax. Therefore, high level language is easier to read, write, understand and maintain.

Each high-level language has their own set of grammar and rules to represent set of instructions. Eg. C, C++,Java, FORTAN etc.

Program written in high level language also need to translated into machine language. This can be done either by compiler or interpreter.

Language Translator

A programmer writes a program in high level language that is to be translated into machine language equivalent code. This task is achieved by using language translator.

The common language translators are:

- Compiler
- Interpreter
- Assembler

Difference between compiler and interpreter

Compiler	Interpreter
1. A compiler translates the entire source code to object code and then only object code is executed.	1. An interpreter translates one statement at a time, executes it and continues for another statement.
2. Compiler is faster than interpreter.	2. Interpreter is slower than compiler.
3. It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.	3. It continuously translates the program until the first error is met, in which case it stops. Hence debugging is easy.
4. A compiler is a complex program.	4. Interpreter is less complex program than compiler.
5. As compared to an interpreter developing a compiler is difficult.	5. As compared to a compiler developing interpreter is easier.
6. Programming language like C,C++,FORTAN use compiler.	6. Python use interpreter.

Assembler

An assembler is a program (software) which translates the program written in assembly language to machine language. It takes the basic commands and operations from assembly code and converts them into binary code that can be recognized by a specific type of processor.

Assemblers are similar to compilers in that they produce executable code. However, assemblers are more simplistic since they only convert low-level code (assembly language) to machine code. Since each assembly language is designed for a specific processor, assembling a program is performed using a simple one-to-one mapping from assembly code to machine code in that they produce executable code.

Differences between high level and low-level programming language

High level programming language	Low level programming language
1. It is programmer friendly language.	1. It is machine friendly language.
2. Compiler or interpreter is required for translation.	2. Assembler is required for translation.
3. They execute slower.	3. They execute faster.
4. For writing program hardware knowledge is not required.	4. For writing program hardware knowledge is required.
5. They are easier to learn and understand by human.	5. It is difficult to learn and understand by human.
6. It can run on any platform.	6. It is machine dependent.
7. It is simple to debug and maintain.	7. It is difficult to debug and maintain as compared to high level language.
8. Example: C, C++, Java etc.	8. Example: Assembly language

Advantages and Disadvantages of High level and low-level programming language

Advantages of High-level language

- ✓ High level languages are programmer friendly.
- ✓ They are easy to write, debug and maintain.
- ✓ They are portable, which means same code can run on different hardware.
- ✓ It is machine independent language.
- ✓ Easy to learn.

Disadvantages of High-level language

- ✓ It takes additional translation times to translate the source to machine code.
- ✓ Execution of high-level programs are comparatively slower than low level programs.
- ✓ Compared to low level programs, they are generally less memory efficient.
- ✓ Cannot communicate directly with the hardware.

Advantages of low-level languages

- ✓ Programs developed using low level languages are fast and memory efficient.
- ✓ There is no need of any compiler or interpreters to translate the source to machine code. Thus, cuts the compilation and interpretation time.
- ✓ It can directly communicate with hardware devices.

Disadvantages of low-level languages

- ✓ Programs developed using low level languages are machine dependent and are not portable.
- ✓ It is difficult to develop, debug and maintain.
- ✓ Programmer must have additional knowledge of the computer architecture of particular machine, for programming in low level language.

Software and its types

Software is a computer program which is a sequence of instructions designed to direct a computer to perform certain task. The software enables a computer to receive input, store information, make decisions, manipulate and output data in the correct format. A program consists of instruction that tell the computer what to do and how to behave.

1. **System software**

The purpose of system software is to help run the computer system by controlling, Integrating and managing the individual hardware components of a computer system.eg. Operating system, device drivers etc. Operating systems like linux , windows and DOS (Disk operating system) control all parts of the computer system by handling I/O devices, coordinating and managing resources like memory, disk ,CPU etc. and provide a environment over which other programs(software) can run.

2. **Application software**

Software which is used for user's specific application known as application software. Application software are designed to process data and support users in an organization such as solving equations or producing bills, result processing of campuses, data processing of accounts in banks. E.g. Word, Excel, PowerPoint, Photoshop, etc.

Application software can be classified into following categories.

- **Tailored software**: These kinds of software are developed for solving particular problem. eg. A software for payroll of an organization, attendance system for student, software for ticket reservation etc.
- **Packaged software:** This software that is often used together, performs similar functions, or includes similar features, and is bundled together as a set of software programs. For example, Microsoft Office is packaged software, including multiple software programs used in a home or office, such as Microsoft Excel, Microsoft Word, and Microsoft PowerPoint. Video and audio editing software may be available as packaged software as well, as they may be used together for editing music and video files used in a movie.

- **Utility software:** These are special types of application software which help us to fine tune the performance of a computer, prevent unwanted actions or perform system related tasks such as checking for virus and removing virus, system utilities which provide information about current state of the use of files, memory, users and peripherals eg disk info, check disk, debuggers for removing "bugs" from program.

Generations of programming Language

The generation of programming languages refers to the evolution and categorization of programming languages based on their features, paradigms, and design principles. Programming languages are often classified into different generations based on when they were developed and the features they introduced.

There are five generations of programming language. They are:

1. **First Generation Languages:**

Machine languages are considered as first-generation language and are directly executable by the computer's hardware. They consist of binary code (0s and 1s) and are specific to the architecture of the computer hardware.

Characteristics:

- ✓ No translator is required.
- ✓ Fast execution due to directly interact with hardware.
- ✓ Difficult to understand, write and debug for humans.

2. **Second Generation Languages:**

Assembly languages are considered the second generation of programming languages. They provide a more human-readable format compared to machine language by using mnemonics (eg. MOV AX, BX) to represent machine instructions. Each mnemonic corresponds to a specific machine operation.

Characteristics:

- ✓ Requires an assembler to convert assembly code into machine code.
- ✓ Machine-dependent but more human-readable than 1GL.

3. **Third Generation Languages:**

High level programming languages are considered as third generation language. They consist of English like words which are easier to understand, write and debug by humans. For execution, a program in this language needs to be translated into machine language using a Compiler/Interpreter.

Examples of this type of language are C, Java, Python, PASCAL, FORTRAN, COBOL, etc.

Characteristics:

- ✓ User-friendly and independent of the underlying hardware architecture.
- ✓ Includes structured and procedural programming paradigms.

4. **Fourth Generation Languages:**

Fourth-generation languages are designed to be even more abstract and domain-specific, focusing on specific problem domains rather than general-purpose computing. They are often used in database querying, report generation, and data manipulation.

Example:

- ✓ SQL (Structured Query Language) for databases.
- ✓ MATLAB for numerical computing.

Characteristics:

- ✓ Focus on specifying what tasks need to be done, without needing to worry about the specific implementation details.
- ✓ Increases productivity with minimal programming effort.

5. **Fifth Generation Languages:**

Fifth-generation languages focus on problem-solving using constraints rather than algorithms. Instead of writing step-by-step instructions, the programmer defines goals, and the system generates the code to achieve them. They are used in developing artificial intelligence, expert systems, and natural language understanding. Examples of fifth-generation languages are Prolog, OPS5, and Mercury.

Characteristics:

- ✓ Allows systems to learn and reason.
- ✓ Use of logic and declarative programming paradigms.

Structured programming

Structured programming is a programming paradigm aimed at improving code clarity, quality, and development efficiency by organizing the program into small, manageable blocks or modules. This is programming approach in the program is made as a single structure. It doesn't support the possibility of jumping from one instruction to some other with the help of any statement like GOTO, etc. Therefore, the instructions in this approach will be executed in a serial and structured manner. The languages that support Structured programming approach are: C, C++, Java, C# etc.

Key Principles of Structured Programming

1) **Control Structures**: Structured programming relies on three main control structures:

- ✓ **Sequence:** Executing statements in a linear order.
- ✓ **Selection:** Making decisions with conditional statements (e.g., if, switch).
- ✓ I**teration**: Repeating a block of code using loops (e.g., for, while).

2) **Modularity:** The code is divided into smaller, manageable sections or modules (functions or procedures). Each module should perform a specific task, making it easier to understand and test.
3) **Top-Down Design:** This approach involves breaking down a complex problem into smaller, more manageable parts. Start with a high-level overview and gradually refine it into detailed steps.
4) **Avoiding Goto Statements:** Structured programming discourages the use of goto statements, which can lead to "spaghetti code" (complex and tangled control flow). Instead, structured programming promotes clear paths of execution.
5) **Clear Variable Scope:** Variables should be defined with a clear scope, limiting their accessibility to the smallest necessary context. This reduces side effects and makes the program easier to understand.

Benefits of Structured Programming

- ✓ **Readability and Maintainability**

One of the primary advantages of structured programming is improved code readability. Readable code is easier to understand, maintain, and debug. This, in turn, contributes to long-term project sustainability.

- ✓ **Error Reduction**

By breaking down a problem into smaller components, structured programming minimizes the chance of errors and makes it easier to identify and rectify issues during the development process.

- ✓ **Code Reusability**

Modularization promotes code reusability, allowing developers to use specific modules in different parts of the program or even in other projects. This not only saves time but also enhances the overall efficiency of software development.

Why C is called structured Programming Language?

C is called a structured programming language because to solve a large problem, C programming language divides the problem into smaller structural blocks each of which handles a particular responsibility. Structured programming is a programming approach that focuses on enhancing the clarity, quality, and efficiency of computer programs. It achieves this goal by heavily relying on structured control flow constructs, such as selection (if/then/else) and repetition (while and for) statements, along with block structures and subroutines.

Here are the key points:

1. **Emphasis on Control Flow:** C prioritizes clear and concise control flow with constructs like if-else, switch, for, and while loops. This structured approach enhances readability and maintainability by avoiding complex logic.
2. **Modular Design:** C encourages breaking down programs into smaller, self-contained functions. This modularity improves code reusability, reduces complexity, and makes debugging easier.

3. **Data Encapsulation**: The use of curly braces {} to define blocks of code helps to group related statements and control the scope of variables, enhancing code organization.
4. **Reduced Reliance on goto**: C discourages the use of goto statements, which can lead to unstructured and difficult-to-understand code.
5. **Top-Down Design**: C supports a top-down approach to problem-solving, where complex problems are broken down into smaller, more manageable subproblems.

Problem solving using Computer

- Problem is defined as the difference between an existing situation and desired situation, that is, in accordance with calculation; problem is numerical situation and has complex form. Solution is a desired situation and has simplest form.
- If a problem is solved by computing using machine called computer, then such process is called problem solving using computer.

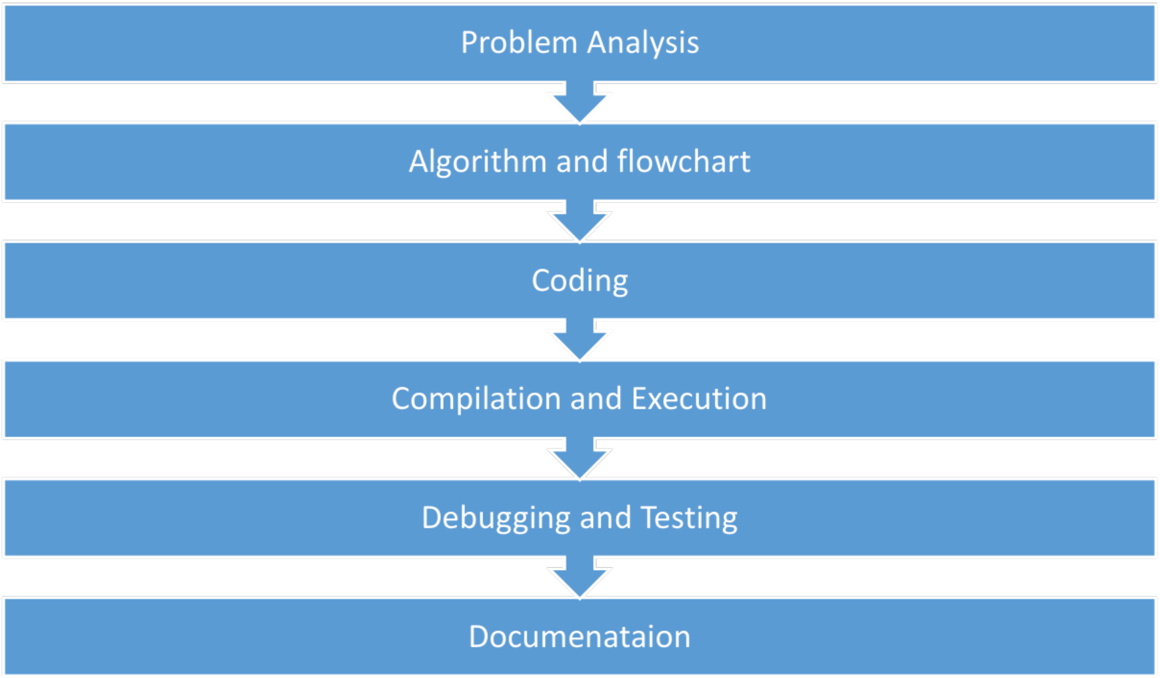


Fig. Steps in problem solving by a computer

1) **Problem Analysis**: It is important to give a clear and precise problem Statement. It is also called Problem definition. In this step, we should specify the following things.
   i. **Objectives:** The problem should be stated clearly.
   ii. **Input requirements:** To get desired output it is required to define the input data and source of input data.eg. In student information system, input data may be students records and source may be college administration.
   iii. **Output Requirements:** We must know what exactly we are expecting out of the system.

iv. **Processing Requirements:** It is required to clearly defined processing requirements to convert the given input data to the required output. In processing requirements, there may be hardware platform, software platform, manpower etc.

v. **Evaluating feasibility:** It is one of the most important phases, where we mainly decide whether the purposed software development task is technically and economically feasible.

2) Algorithm and flowchart

Algorithm

An Algorithm is defined as a sequence of steps or procedures necessary to solve problem. To be an algorithm set of step must be unambiguous. Each step tells what task to be performed.

An Excellent Algorithm should have the following properties.

i. **Finiteness:** Each algorithm must have finite number of steps to solve a problem.

ii. **Definiteness:** The action of each step should be defined clearly without any ambiguity.

iii. **Inputs:** Inputs of the algorithm should be define precisely, which can be given initially or while the algorithm runs.

iv. **Output:** An algorithm can give one or more result. After an algorithm terminates, algorithm must give desired result.

v. **Effectiveness:** An algorithm should be more effective among different ways of solving the problem.

EXAMPLES

Algorithm to find the sum of any two numbers

```
Step 1: Start
Step 2: Declare variables a ,b and sum
Step 3: Read value of a and b
Step 4: calculate sum=a+b
Step 5: Display sum
Step 6: Stop
```

Algorithm for calculating the simple interest using the formula si=p*t*r/100

```
Step 1: Start
Step 2: Declare variables p, t, r and i
Step 3: Read value of p, t, r
Step 4: si=p*t*r/100
Step 5: Display si
Step 6: Stop
```

Algorithm to convert Fahrenheit when temp in Centigrade is given.

```
Step 1: Start
Step 2: Declare variables f and c
Step 3: Read value of c
Step 4: calculate f=1.8*c+32
Step 5: Display f
Step 6: Stop
```

Algorithm to calculate the area of rectangle

```
Step 1: Start
Step 2: Declare variables l, b and area
Step 3: Read value of l and b
Step 4: calculate area=l*b
Step 5: Display area
Step 6: Stop
```

Advantages of Algorithms:

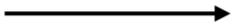
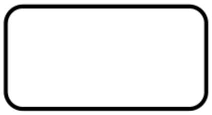
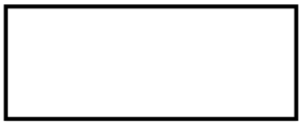
- ✓ It is a step-wise representation of a solution to a given problem, which makes it easy to understand.
- ✓ An algorithm uses a definite procedure.
- ✓ It is not dependent on any programming language, so it is easy to understand for anyone even without programming knowledge.
- ✓ Every step in an algorithm has its own logical sequence so it is easy to debug.
- ✓ By using algorithm, the problem is broken down into smaller pieces or steps hence, it is easier for programmer to convert it into an actual program.

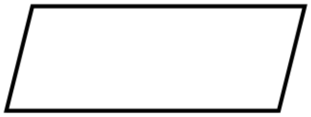
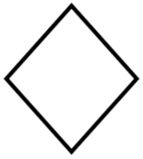
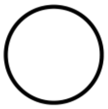
Disadvantages of Algorithms:

- ✓ Algorithm is Time consuming.
- ✓ Difficult to show Branching and Looping in Algorithms.
- ✓ Big tasks are difficult to put in Algorithms.

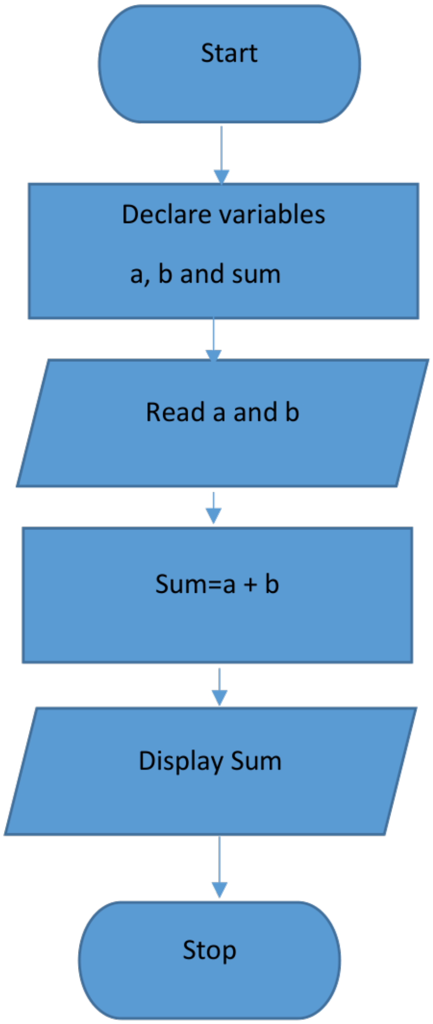
Flowchart:

A flowchart is an pictorial representation of an algorithm. It uses boxes of different shapes to denote different types of instructions. The actual instructions are written within these boxes use clear and concise statements. These boxes are connected by solid lines having arrow marks to indicate the flow of operation, that is exact sequence in which instructions are to be executed.

Symbol	Purpose	Description
	Flow Line	Used to connect symbols to indicate the flow of logic
	Terminal (Start/Stop)	Used to indicate the start and end of flowchart. One flow line exists from Start and one enter to Stop
	Processing	Uses for arithmetic and data manipulation operations

	Input/ Output	Used whenever information is entered into the flowchart or displayed from the flowchart.
	Decision	Used to represent operation in which two possible alternatives ie. Decision making and branching
	On-page Connectors	Used to connect remote flowchart portions of the same page.

Flowchart to add two numbers



Advantages of using flowcharts

i. **Communication:** Flowcharts are better way of communicating the logic of the system to all concerned.
ii. **Effective analysis:** With the help of flowchart, problem can be analyzed in more efficient way.
iii. **Proper Documentation:** Program flowcharts serve as a good program documentation, which is needed for various purposes
iv. **Efficient coding :** The flowcharts act as a guide or blueprint during the system analysis and program development phase
v. **Proper Debugging:** The flowchart helps in debugging process.
vi. **Efficient program maintenance:** Maintenance of operating program becomes easy with the help of flowchart.

Limitations of using flowchart

i. **Complex logic:** A flowchart becomes complex and clumsy when the program logic is quite complicated.
ii. **Difficulty in alternations and modifications:** If alternations are required, the flowchart may need to drawn completely.

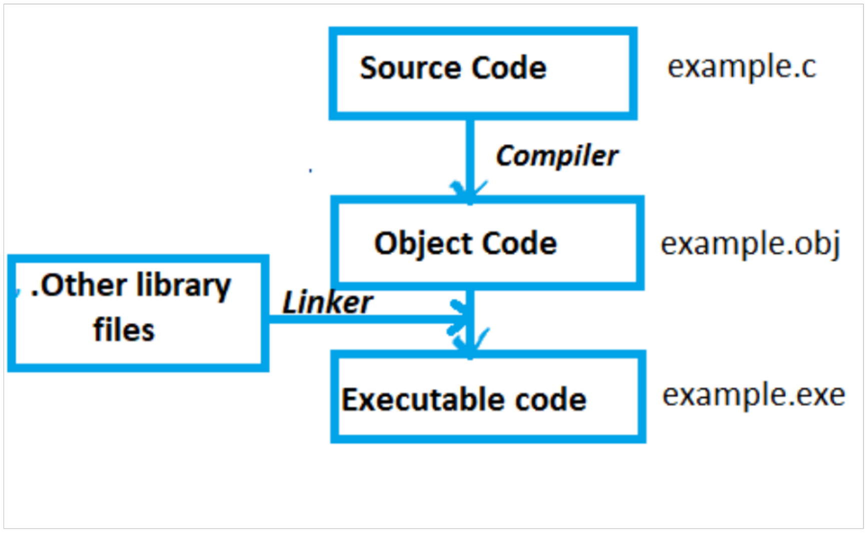
3) Coding

The coding is a process of transforming program logic or ideas into computer language format. This stage translated program design into computer instructions using some programming language, such as c, c++ etc. so, we should properly flow the rule and syntax while coding .The code writing using programming language is also known as source code.

4) Compilation and execution

Program written in high level language must be converted into low level language because computer only understand low level language .This process is known as compilation.so, compiler does job of translating high level language into low level language. Compilation process test the program weather it contains error or not. If syntax error are present, compiler cannot compile the code until it is corrected.

Once, compilation is completed then the program is linked with another library files needed for execution. Thereby resulting in binary program and then program is loaded into memory for the purpose of execution. During execution, program may ask user for inputs and generates output after processing inputs.



5) Debugging and testing

Debugging

Debugging is the process of isolating and correcting any type of errors. Different type of debugging techniques are as follows.

- Error Isolation
- Tracing
- Watch Values
- Break points
- Stepping

Testing

Testing ensures the program performs correctly the required task. Testing is very important phase before delivering software to customer. Testing is usually performed for the following purposes.

- To improve quality
- For verification and validation
- For reliability estimation

Testing process may include the following two stages.

i) **Human testing**

- ✓ Done before computer based testing begins
- ✓ Test is carried out statement by statement by a programmer/test group and analyzed with respect to checklist of common programming errors.

ii) **Computer based testing**

Computer based testing involves two stages namely compiler testing and runtime testing.

Compiler testing

- Debugging techniques used to find syntax error.

Runtime testing

- If there is no syntax error, we need to find logical and runtime error. After removing these errors, it is required to run program with test data to check whether program is producing correct result or not.

Types of Errors:

Generally errors are classified into following types:

Syntax Error: Any violation of rules of the language results in syntax error. Most frequent syntax error are:

- Missing parenthesis
- Missing semicolons.
- Printing the value of variable without declaring it.

Semantic Error:

This error occurs when the statements written in the program are not meaningful to the compilers.

eg. a+b=c

Runtime Error:

Error which occur during program execution after successful compilation are called runtime error.one of the most common run-time error is

- Division by zero
- Null pointer assignment
- Trying to open file that was not created

Logical Error

Logical errors are the errors in the output of the program. The presence of logical errors to undesired or incorrect output and are caused due to error in logic applied in the program to produce desired output.

Latent Error:

Latent Errors are the hidden errors that occur only when a particular set of data is used. Consider below example.

result=(a+b)/(c-d);

An error occurs only when c and d are equal because that make remainder zero (divide by zero error).Such error can be detected only by using all possible combinations of data.

6) Documentation

Program Documentation is the description of program and its logic written to support understanding of program. It keeps information of all phases described above while developing project.

- ➢ Documentation of program helps to those who use, maintain and extend the program in future.
- ➢ Properly documented program is necessary which will be useful and efficient in debugging, testing, maintenance and redesign process.

There are two types of documentation

1) Programmer documentation (Technical documentation)

Programmer's Documentation is prepared for future reference to the programmers who maintain, redesign, and upgrade the system.

It may contain

- Detail requirements of program
- Dataflow diagram, E-R diagram and flowchart
- Selection of meaningful variables name and use of comments
- Parameter and definition list to explain the meaning and purpose of each parameter and definition of each function.
- Algorithm and flowchart of each component
- Input and output data used for testing and other references

2) User documentation

- Provides the support to user who use the software.
- It provides guidelines for installation of program and use it efficiently.

Implementation

After a program has been written and tested it need to be implemented to target environment to solve problem. Various needs of physical hardware and accessories required by the program to solve the intended problem needs to be present upon implementation.

Evaluation and Maintenance

The Evaluation of the performance of program is need to be done at frequent interval once the program software is implemented. Evaluation can be done by various parameters such as quality assurance, user friendliness, flexibility and functionality.
Maintenance is the process of modifying a software for the following reason.

- ➢ To remove the a bug from the program
- ➢ To improve the efficiency of code
- ➢ Add new function to fulfill new user requirements.
- ➢ To improve user interface.