UNIT - I

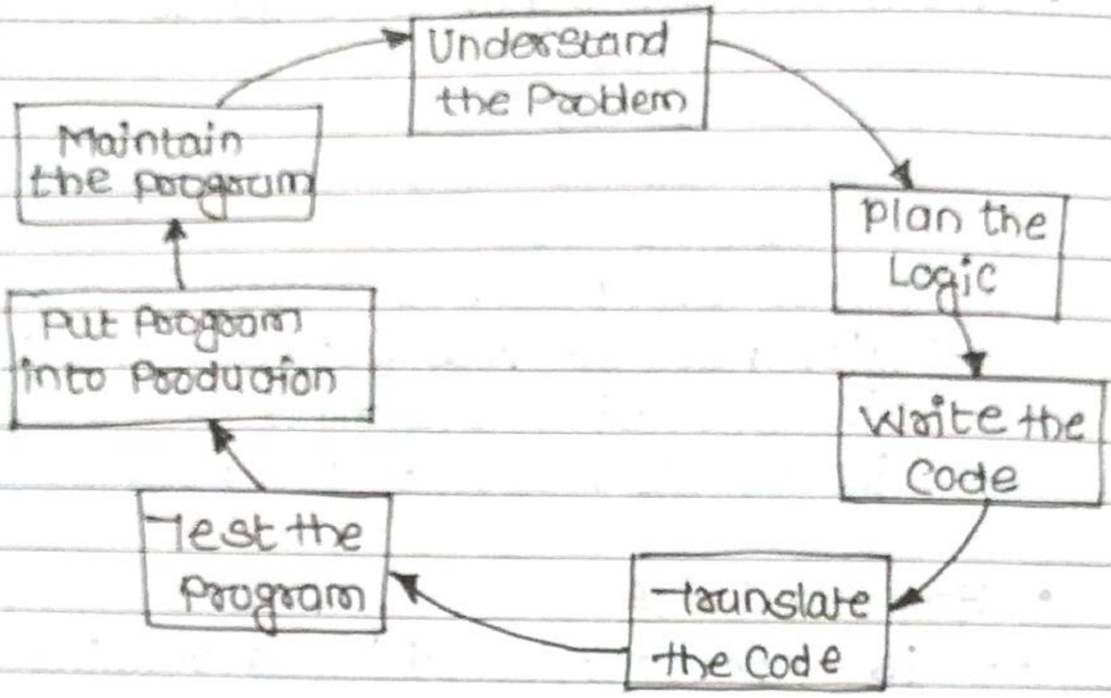
* Why C Language is So Important?

- Worth to Know about C Language.
  - Oracle is written in C.
  - Core libraries of Android are written in C.
  - My SQL is written in C.
  - Almost every device driver is written in C.
  - Major part of Web browser is written in C.
  - Unix operating system is developed in C.
  - C is world's most popular programming language.
- For students
  - C is important to build programming skills.
  - C covers basic features of all programming language.
  - C is most popular language for hardware dependent programming.

* History of C language

- In 1966 Martin Richard developed BPCL (Basic Combined programming Language.)
- In 1969 Ken thomson developed B Language. He is also developer of UNIX operating system. He is also developed first Master Level chess called Belle in 1980.
- Dennis Ritchie developed C language in 1972 at AT&T's Bell Labs, USA. He is also Co-developer of UNIX operating system.

* Process of Programming *



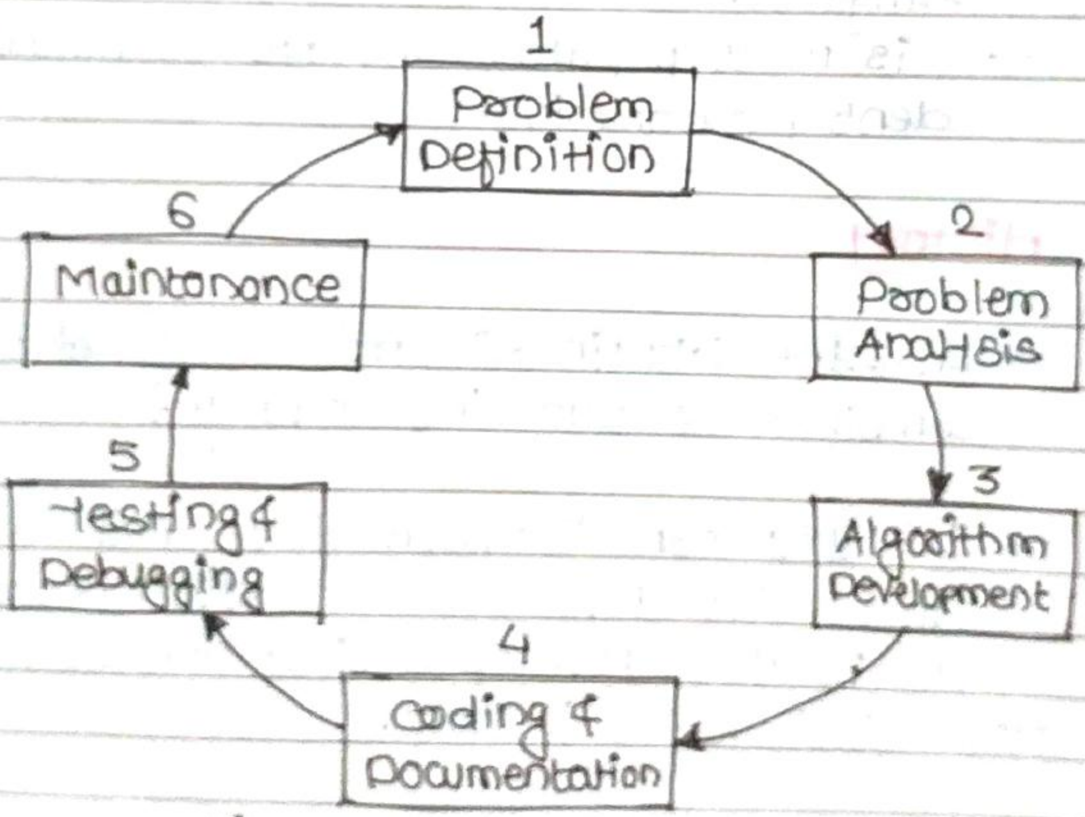


figure:- Process of programming

Computer programming (often shortened to programming) is a process that leads from an original formulation of a computing problem to executable computer programs. Programming involves activities such as analysis, developing understanding, generating algorithms, verification of requirements of algorithms including their correctness and resources consumption, and implementation (commonly refered to as coding) of algorithms in a target programming language.

☆ Steps involved in programming

→ Analyzing the problem.
→ Algorithm design/Psuedocode
→ Flowchart
→ Coding
→ Debugging
→ Testing
→ Final Output
→ Documentation

① Analyzing the problem:-

This is the first step of proggramming and involves getting the following information. These things are very importent for the programmer because it provides him with the basis for planning about the programming and to control the potential difficulties that may arise.

② Algorithm Design:-

All instructions to be performed at differeut stage are listed. This is done in simple English languge. We may call it the strategic step.

③ Flowchart:-

It is graphical tool that shows the steps/stages which are to be executed in a program.

All the steps which are written in the second stage are now presented in diagrammatic manner so to make it easily understandable. Making of Flow chart helps us in increasing our process of program development because it facilitates ~~its facilities~~ our ability to define the logic, detecting and removing errors in program design.

Types of Flow charts

- System Flowchart shows the processing of the entire system. It describes the input/output devices, the media being used and the flow of data in the system.
- programs Flowchart shows the complete steps involved in the execution of a program including I/O, processing, loops and branching. It is more detailed than a system flowchart.

④ Coding:-

In this step, the programmer writes the instructions in a computer language to solve the problem. All coding processes depends upon the information obtained from previous steps. Choice of language depends upon the requirements and facilities available with a language.

⑤ Debugging:-

In this stage, We removes all the errors in the program because when we are coding, there are chances that some mistakes may occurs at that time. Here the program is executed manually. Called DRY RUN. This is done several times untill all the errors are removed from the program and the system becomes error-free.

⑥ Testing:-

The program is tested by entering a dummy data (which includes usual, unusual and invalid data) to check the behaviour and result of the program towards the given data.

⑦ Final Output:-

After going through all the above stages, the program is given the TRUE DATA. Here the programmer expects positive results of the program and expects full efficiency of the program.

⑧ Documentation:-

Most programmers neglect this stage by giving many reasons, but this is very important because this will help the programmer to correct the problems that may occurs in the program.

Types of documentation

There are two types of documentations

- User Manual provides user with complete information about how to operate the program & what needs to be done when the user faces a problem.
- Technical manual contains information about the program. This is used to get technical details of the program when the system is not working properly or requires modifications.

★ EDITING ★

— A source-code editors is a text editors program designed specifically for editing source code of computer programs. It may be a stand alone application or it may be built into an integrated development environment (IDE) or web browser.

— Editors or text editors are software programs that enables the user to create and edit text files. In the field of programming, the term editors usually refers to source code editors that include many special features for writting and editing code. Notepad, Wordpad are some ~~special features~~ of the common editors used on windows OS & vi, emacs, Jed, pico are the editors on UNIX OS. Features normally associated with text editors are moving the cursor, deleting, replacing, pasting, finding, and replacing, saving etc.

Types of Editors

There are five types of editors as described below:

1. Line editor:

In this, you can only edit one line at a time or a integral number of lines. You can not have a free-flowing sequence of characters. It will take care of only one line.

Examples:- teleprinter, edlin, teco.

Date

2. Stream editors:-

In this tipe of editors, the file is treated as continious flow or sequence of characters instead of line numbers, which means here you can type paragraphs.

Example:- Sed editor in UNIX.

3. Screen editors:-

In this tipe of editors, the user is able to see the cursor on the screen & can make a copy, cut, paste operations easily. It is very easy to use mouse pointer.

Example:- Vi, emacs, Notepad.

4. Word Processors:-

Overcomming the limitations of screen editors, it allows one of use some format to insert images, files, videos, use font, size, stile features. It majorraly focuses on Natural Language.

5. Structure Editor:-

Structure editor focuses on programming languages. It provides features to write and edit source code.

Examples:- Netbean IDE, gEdit.

* Compiler

→ A compiler is a software that typically takes a high-level language (Like C, C++, & Java) code as input and converts the input to a lower level language at once. It lists all the errors if the input code does not follow the rules of its language. This process is much faster than interpreter but it becomes difficult to debug all the errors together in a program.

→ A compiler is a translating program that translates the instructions of high-level language to machine level languge. A program which is input to the compiler is called a source program. This program is now converted to a machine level language by compiler is known object code.

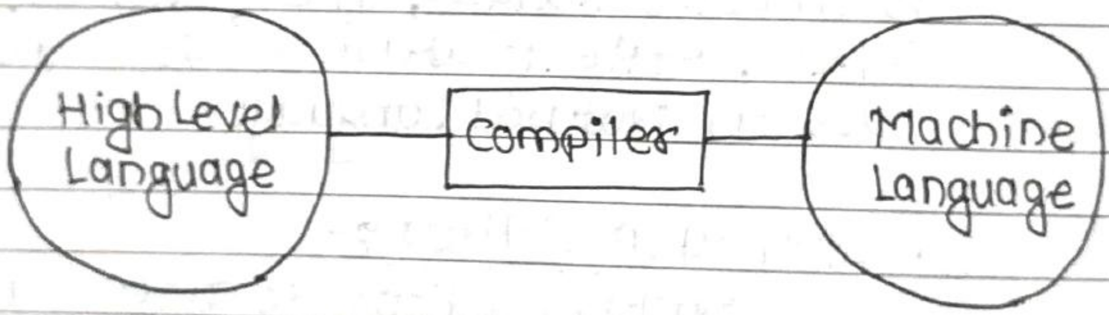


figure: compiler

There are different compilers:

- Cross-Compiler – the compiled program can run on a computer whose CPU or operating System is different from the one on which the compiler runs.
- Bootstrap Compiler – the compiler written in the language that it intends to compile.
- Decompiler – The compilers that translates from a low-level language to a higher level one.

- transcompiler – The compiler that translates high level languages.

— A compiler can translate only those source programs which have been written in the language for which the compiler is meant. Each high-level programming language requires a seperate compiler for the conversion.

— For example, a FORTRAN compiler is capable of translating into a FORTRAN program. A computer system may have more than one compiler to work for more than one high level languages.

— Top most Compilers used according to the computer Languages –
- C – Turbo C, Tiny C Compiler, GCC, Clang, Portable C Compiler.
- C++ – GCC, clang, Dev C++, Intel C++, Code Block
- Java – IntelliJ, IDEA, Eclipse IDE, NetBeans, BlueJ, JDeveloper.
- Kotlin – IntelliJ IDEA, Eclipse IDE.
- Python – CPython, JPython, Wing, Spyder.
- JavaScript – WebStorm, AtomIDE, Visual Studio Code, Komodo Edit.

☆ Compiling and Executing C program

— C is compiled language.
— Once the program is written, you must run it through C compiler that can create an executable file to be run by the computer. While the C program is human-readable on other side executable file is machine readable.
— The compiler translates source code into an object code. The object code contains machine instructions for the CPU.

- However object file is not executable file. Therefore in next step, the object file is ~~not~~ processed with another special program called as Linker. The output of linker is an executable or runnable file. The process is shown in following figure.

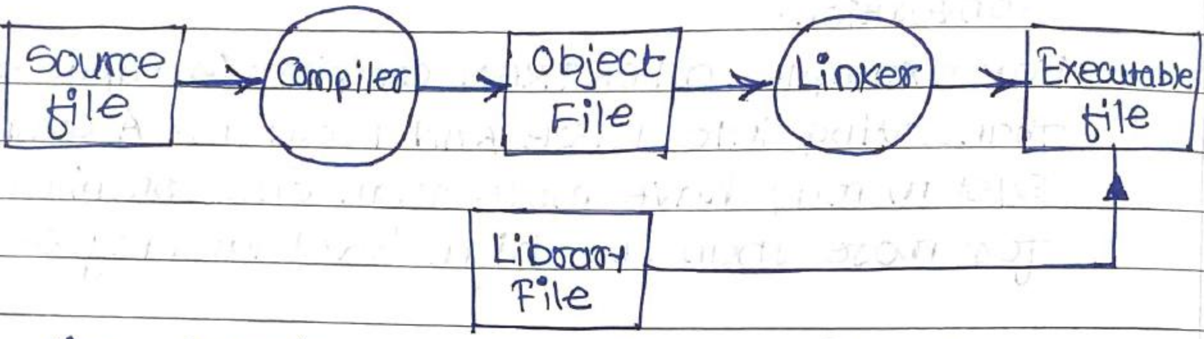


fig:- Overview of compilation and Execution Process.

★ Error Checking ★

— Error handling refers to the anticipation, detection, and resolution of programming, application, and communications errors.

— Specialized programs, called error handlers, are available for some applications. The best programs of this type forestall errors if possible, recover from them when they occurs without terminating the application, or (if all else fails) gracefully terminate an affected application and save the error information to a log file.

— In programming, a development error is one that can be prevented. Such an error can occur in syntax or logic. Syntax errors, which are typographical mistakes or improper use of special characters, are handled by rigorous proofreading. Logic errors, also called bugs, occurs when executed code does not produce the expected or desired result. Logic errors are best handled by meticulosus program debugging. This can be an ongoing process that involves, in addition to the traditional debugging routine, beta testing prior to official release and customer feedback after official release.

A run-time error takes place during the execution of a program, and usually happens because of adverse system parameters or invalid input data. An example is the lack of sufficient memory to run an application or a memory conflict with another program.

As such, C programming does not provide direct support for error handling but being a system programming ~~does not provide direct~~ language, it provides you to access at lower level in the form of return values. Most of the C or even Unix function call return -1 or NULL in case of any error and set an error code errno. It is set as global variable and indicates an error ~~codes~~ occurred during any function call. You can find various error codes defined in <error.h> header file.

So a C programmer can check the returned values and can take appropriate action depending on the return value. It is a good practice, to set errno to 0 at the time of intializing a program. A value of 0 indicates that there is no error in the program.

errno, perror(), strerror()

The C programming language provides perror() and strerror() functions which can be used to

display the text message associated with errno.
The perror() function displays the string you pass to it, followed by a colon, a space, & then the texual representation of the current ~~error~~ errno value.
The strerror() function, which returns a pointer to the texual representation of the current e errno value.

Let's try to simulate an error condition and try to open a file which does not exist. Here we are using both the functions to show the usage, but you can use one or more ways of printing your errors. Second important point to note is that you should use stderr file stream to output all the errors.

```
#include <stdio.h>
#include <errno.h>
# include <string.h>

externinterrno;

int main() {
        FILE * pf;
interrnum;
pf = fopen ("unexist.txt", "rb");

if (pf == NULL) {

 errnum= errno;
fprintf(stderr, " Value of errno: %d \n", errno);
perror (" Error printed by perror");
fprintf (stderr, " Error opening file: %s \n",
          strerror(errnum));
}
else {
  fclose (pf);
  }
 return ~~EOF~~; 0;
}
```

When the above code is compiled and executed, it produces the following result -

```
value of errno: 2
Error printed by perror: No such file or directory.
Error opening file: No such file or directory.
```

① Divide by Zero Errors :-

It is common problem that at the time of dividing any number, programmers do not check if a divisor is zero and finally it creates a runtime error.

The code below fixes this by checking if the ~~divisor~~ divisor is zero before dividing -

```
# include <stdio.h>
# include <stdlib.h>
 main ()
 {
  int dividend = 20;
  int divisor = 0;
  int quotient;

  if (divisor = = 0)
   {
    fprintf (stderr, "Division by zero! \n");
    exit (-1);
   }
  quotient = dividend / divisor;
  fprintf (Stderr, "Value of quotient: %d \n", quotient);

  exit (0);
 }
```

— When the code is compiled and executed, it produces the following result-

Division by zero!

② Program Exit Status:-

It is common practice to exit with a value of EXIT_SUCCESS in case of program coming out after a successful operation. Here, EXIT_SUCCESS is a macro and ~~it~~ is defined as 0.

If you have an error condition in your program and you are coming out then you should exit with a status EXIT_FAILURE which is defined as -1.

★ Executing C Programs

① Write, Compile and Run C program

→ Open a new file from File → New in the Turbo C++ IDE. Write a small program in the IDE.

→ Now go to file > Save As and save the program with the filename of your choice (make sure extension of the filename is .c).

→ Click on Options and goto Directories. Click on Directo~~ries~~ry as you want and Source Directory as where you have saved the C program file.

→ Now goto compile and click on compile. And then click on Run.

→ You will see the output of your C program.

② Write, compile and Run C program using wxDev-C++ in Windows.

wxDev-C++ is easy to use IDE which you may opt for to write C program. You may download the installer from wxdsgn.sourceforge.net. We found it working perfectly on Windows 7 & Windows XP.

It also install MinGW along with & you don't need to set any environment variables. The default compiler for this IDE is gcc.

After downloading the installer, run the exe file by double clicking on it and an installation wizard will guide you to install. Once you finish the installation, you start it from programs & the first time when you run it, it takes a while for parsing header files. ~~The IDE windows tools~~

- You may start programming by clicking on File → New → source file in the window. Note that while saving file, you must select file type as C as this IDE supports C++ also.
- You may use F9 or ~~as~~ compile and Run program.
- When compilation is done, it opens a new window to show you output.

- Though sligtly dated, we find WxDev-C++ an excellent IDE for programming C. You may try it if you are using Windows.

③ Install, Compile and Execute C program in Linux

- Most of the time, when you are installing Linux, GNU Gcc compiler is already installed. If not, run the following command (our system is Ubuntu Linux):

```
sudo apt-get install build-essential ↵
```

If C compiler is already installed, it will show you a message. ~~like~~ If not, it will install all the necessary packages.

Now open a text editor and write a small C program like following and save it as demo.c:

```
#include <stdio.h>

main()
{
 printf("Welcome to C programming");
}
```

Now run the command as shown below to compile and execute the file:

```
Linux:~/Desktop$ gcc -o demo demo.c ↵
Linux:~/Desktop$ ./demo ↵
Welcome to C programming Linux:~/Desktop$ █
```

This is how you can install GNU Gcc compiler, write a C program and run it under Linux.

* Testing and Debugging C program

→ Testing means verifying correct behaviour. Testing can be done at all stages of module development: requirement analysis, interface design, Algorithm design, implementation, and integration with other modules. In the following, attention will be directed at implementation testing. Implementation testing is not restricted to execution testing. An implementation can also be tested using correctness proofs, code tracing, and peer reviews, as described below.

→ Debugging is a cyclic activity involving execution testing and code correction. The testing that is done during debugging has a different aim than final module testing. Final module testing aims to demonstrate correctness, whereas testing during debugging is primarily aimed at locating errors. This difference has a significant effect on the choice of testing strategies.

★ Preconditions for effective Debugging:

In order to avoid excessive time spent on debugging, the programmer should be mentally prepared for effort. The following steps are useful to prepare for debugging.

- Understand the design and Algorithm:

If you are working on a module and you do not understand its design or its algorithms, then debugging will be very difficult. If you don't understand the design then you can't test module because you do not know what it is suppose to do.

- Check correctness:

There are several methods for checking correctness of an implementation prior to execution.

- Correctness Proofs:

One useful code check is to examine code using the logical methods of correctness proofs.

- Code Tracing:

Often, errors can be detected by tracing through the execution of various calls to module services, starting with a variety of initial conditions for the module. For poorly understood psychological reasons, tracing works best if you are describing your tracing to someone else.

- Peer Reviews:

A peer review involves having a peer examine your code for errors. To be effective, the peer should beither already be familior with the algorithm, or should be given the algorithm and code in advance. For commercial programming, however, quality of code is much more important. Thus peer reviews are a significant part of a software quality assurance program.

- Anticipate Errors: Unfortunately, human make errors with correctness arguments and sometimes miss cases in code tracing, and peers don't always catch errors either. So a programmer should be prepared for some errors remaining in the code after the step listed above. Hopefully, there won't be too many.

① **Requirements for Debugging**

To effectively debug code you need two capabilities. First, you need to be able to efficiently call on the services provided by the module. Then you need to be able to get information back about results of the calls, changes in the internal state of the module, error conditions, and what the module was doing when an error occurred.

Driving the Module

To effectively debug a module, it is necessary to have some methods for calling upon the services provided by the module. There are two common methods for doing this.

- Hardwired Drivers – A Hardwired driver is a main program module that contains a fixed sequence of calls to the services provided by the module that is being tested.
- Command Interpreters – A command interpreter drives the modules under test by reading input and interpreting it as commands to execute calls to module services. Command interpreters can be designed so that the commands can either be entered ~~interpretation~~ interactively or read from a file.

Obtaining Information about the Module

Being able to control the sequence of calls to module services has little value unless you can also obtain information about the effects of those calls. If the services generate outputs then some information is available without any further effort. However, for many modules, including data structure modules, the primary effect of calls to services is change in the internal state of the module. This leads to needs for three kinds of information for debugging.

- Module State: Data structure modules generally have services for inserting and deleting data. These services almost never generate output on their own, and often do not return any information through parameters. Therefore in order to test or debug the module, the programmer must add code that provides information about changes in the internal module state.
- Module Errors: When a module has a complex internal state, with incorrect code it is usually possible for invalid state to arise. Also it is possible that provide subroutines are called incorrectly. Both of these situations are module errors. When practical, code can be added to the module to detect these errors.
- Execution State: In order to locate the cause of module errors, it is necessary to know what services and private subroutines have been called when the errors occurs. This is the execution state of the module. One common method for determining the execution state is the addition of debugging print statements that indicate entry and exit from segments of code.

② Principles of Debugging

- Report Error Conditions Immediately:
  Much debugging time is spent zeroing in on the cause of errors. The earlier an error is detected, the easier it is to find the cause. If an incorrect module state is detected as soon as it arises then the cause can often be determined with minimal effort. If it is not detected until the symptoms appear in the client interface then may be difficult to narrow down the list of possible causes.
- Maximize Useful Information & Ease of Interpretation:
  It is obivious that maximizing useful information is desirable, and that it should be easy to interpret. Ease of Interpretation is important in data structures. Some module errors can not easily be detected by adding code checks because they depend on the entire structure. Thus it is important to be able to display the structure in a form that can be easily scanned for correctness.
- Minimize Useless and Distracting Information:
  Too much information can be as much of a handicap as too little. If you have to work with a printout that shows entry and exit from every procedure in a module then you will find it very difficult to find the first place where something went wrong. Ideally, module execution state reports should be issued only when an error has occurred. As a general rule, debugging information that says "the problem is here" should be preferred in favor of reports that say "the problem is not here."

- Avoid Complex one: Use testing code - One reason why it is counterproductive to add module correctness checks for errors that involves the entire structure is that the code to do so can be quite complex. It is very discouraging to spend several hours debugging a problem, only to find that the errors was in the debugging code, not the module under test. Complex testing code is only practical if the difficult part of the code are reusable.

③ Debugging Aids

Aids Built into Programming Language

- Assert Statements: Some Pascal compilers and all C compilers that meet the ANSI standard have assert procedures. The assert procedure has a single parameter, which is a Boolean expression. When a call to assert is executed the expression is evaluated. If it evaluates to true then nothing happens. If it evaluates to false the the program terminates with an error message. The assert procedure can be used for detecting and reporting error conditions.
- Tracebacks: Many pascal compilers generate code that results in tracebacks whenever a runtime error occurs. A traceback is a report of the sequence of subroutines that are currently active. Sometimes a traceback will also indicate line numbers in the active subroutines. If available, a traceback revebals where the runtime errors occurred, but it is upto the programmer to determine where the cause lies.
- General Purpose Debuggers: Many computer systems or compilers come with debugging programs. For example, most UNIX operating systems have

general purpose debuggers such as sdb & dbx. Debugging programs provides capabilities for stepping through a program line-by-line and running a program with breakpoints set by the user. When a line with breakpoint is about to be executed the program is interrupted so that the user can examine or modify program data. Debugging programs also can provide tracebacks in case of run-time errors.

④ Debugging Techniques

1. Incremental testing:-

In a good design for a complex module, the code is broken up into numerous subroutines, most of which are no more than 10 to 15 lines long. For a module designed in this way, incremental testing offers significant advantages. For incremental testing, the subroutines are classified in levels, with the lowest level subroutines ~~are classified in levels~~ being those that do not call other subroutines. If subroutine A calls subroutine B then A is a higher level subroutine than B. The incremental testing strategy is to test the subroutines indivisually, working from the lowest level to higher levels.

2. Sanity Checks:-

Low level code in complex data structure is often written with the assumptions that the higher level code correctly implements the desired algorithm. For example, the low level code may be written with that assumption that a certain variable or parameter cannot be NULL. Even if that assumption is justified by the algorithm, it may still be a good idea to put in a test to see if the condition is satisfied because the higher level code may be implemented incorrectly. This kind of check is called a sanity check. If an assert procedure is available then it can be used for the checks. The advantage of sanity checks is that they give early detection of errors.

③ Boolean constant for turning Debugging code on or off
If debugging code is added to a module then it is often profitable to enclose it in an if statement that is controlled by a Boolean constant added to the module. By doing this, the debugging code can easily be turned off, yet be readily available if needed later. Different constants should be used for different stages of testing so that useless information minimized.

④ Error Variable for controlling program behaviour after Errors.

⑤ Traceback techniques.

⑥ Correcting code Errors.

* Algorithms and Flowchart :-

→ There are three ways to represent the logical steps for finding the solutions to a given problem, i.e. Algorithm, Flowchart & Psuedocode.

→ In an algorithm, description of steps for a given problem is provided. A flowchart represents solutions to a given problem graphically. Another way to express the solution to a given problem is by means of pseudo code.

1. Algorithms

- An algorithm is a finite set of precise instructions for performing a computation for solving a problem.
- A step-by-step procedure for solving a particular problem is called as an algorithm.

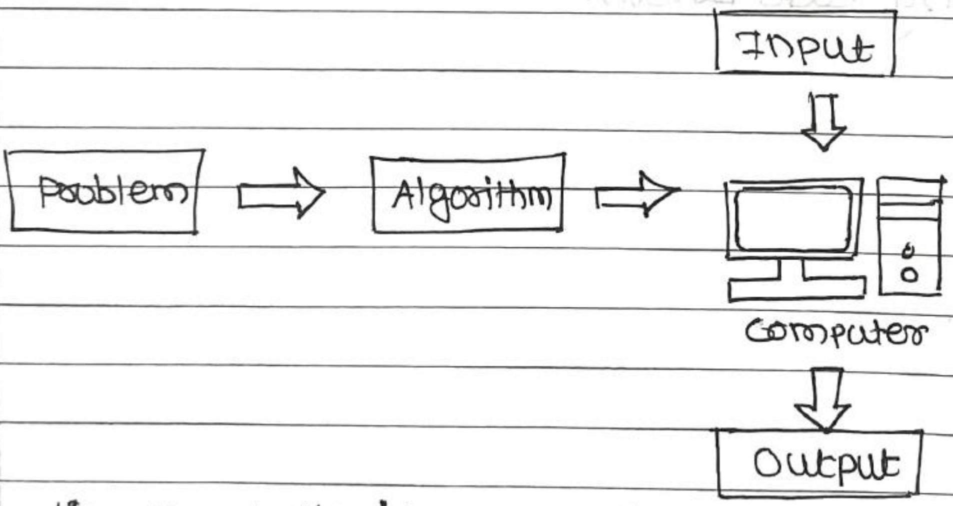


fig:- Use of Algorithms for Solving Computational Problems

- Figure shows a simple illustration of how algorithms are used for solving computational problems.
- An algorithm solves only a single problem at a time. However, the same problem can be solved using multiple algorithms.
- The advantage of using multiple algorithms to solve the same problem is purely situational.

- The choice of a particular algorithm solely depends on the type of input values i.e. single or multiple.
- Algorithmic strategies can be written using two methods as shown in below figure.

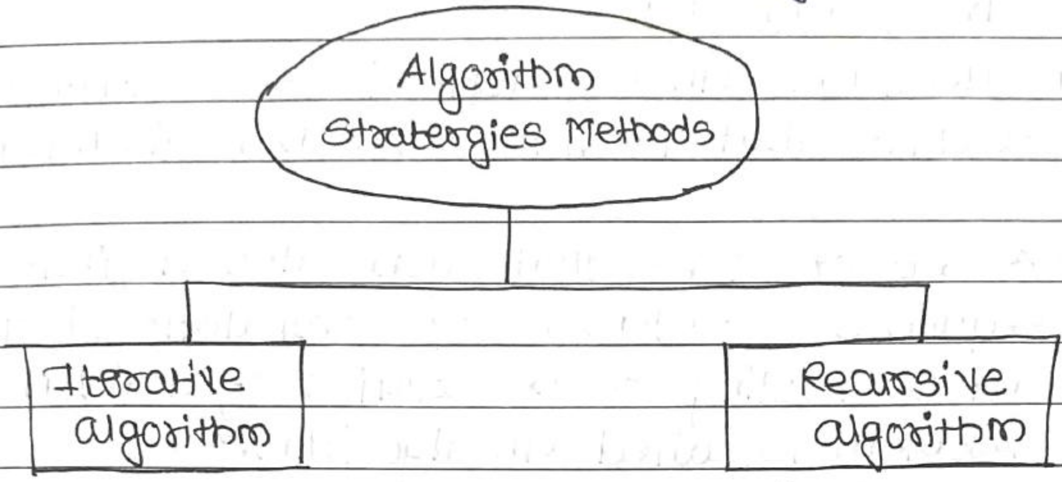


fig:- Strategies of algorithm

1. Iterative Algorithm: In iterative algorithms, the process is carried out respectively on the inputs in order to achieve the desired output.
2. Recursive Algorithm: This algorithm is an algorithm which call itself with smaller (or simple) input values, and which obtains result for the current input by applying the same operations for the smaller (or simpler) input.

Approaches for designing an Algorithm:-

- Top-down Approach: A top-down approach starts by identifying the major components of the system or program decomposing them into their lower level components and iterating until the desired level of module complexity is achieved. In this we start with the topmost module ~~complexity is achieved. In this we start with the topmost mod~~ and incrementally add modules that it calls, (For example: 'C' languge).
- Bottom-up Approach: A bottom-up design approach starts with designing the most basic or primitive components and proceeds to higher level components

starting from the very bottom, the operations that provide a layer of abstraction are implemented.

Algorithm Definition:

An algorithm can be defined as "a step-by-step procedure that provide solution to given problem."

Or

"A set of steps that generates a finite sequence of elementary computational operations leading to the solution of a given problem is called an algorithm."

Characteristics:-

Every algorithm must satisfy the following characteristics:

- Finitness: An algorithm must always terminate after a finite number of steps.
- Definiteness: Each and every step of the algorithm should be rigorously and unambiguosly defined.
- Input: The algorithm should take zero or more inputs.
- Output: The algorithm should produce one or more inputs.
- Effectiveness: A human should be able to calculate the values involved in the procedure of the algorithm using pencil & paper.

Advanteges:-

- An algorithm gives a language independent layout of the program or problem.
- It allows the programmers to use the most efficient solution of the problem.
- It eases the process of actual development of program code.

- It breaks down the solution of a problem into a series of simplified sequential steps.
- Its simplified way of representing program instructions enables other programmers to easily understand and modify it.

Disadvantages:-

- For large algorithms, it becomes difficult to understand the flow of program control.
- There are no standard conventions to be followed while developing algorithms.
- It may take considerable amount of time to write the algorithm for given problem.
- It lacks the visual representation of programming logic as is prev~~ented~~alent in flowcharts.

2. Flowcharts

- A flowchart is a visual representation of the sequence of steps for solving a problem.
- A flowchart can be referred as pictorial representation of an algorithm. The objective of using flowcharts to describe the problem solution is to ease the understanding of programming logic.
- A flowchart is a diagramatic representation of the algorithm or of the plan of solution of a problem.
- Flowchart indicates the process of solution, the relevant operations and computations, the point of decision and other information which is a part of the solution.

Principles of Flowcharts:

- Pictorial representation of flowchart makes it a convenient method of communication.
- It promotes logical accuracy and is a key to correct programming.
- It takes care that no path is left incomplete without any action being taken.

- It helps to develop program logic and serves as documentation.
- It is an important tool for planning and designing a new system.

Types of Flowcharts:

1. System Flowcharts: Used by system analyst. This flowchart shows various processes, sub-systems, outputs and operations on data in a system.
2. Program Flowchart: Used by computer programmers. This flowchart show program structure, logic flow and operations performed.

Defination Of Flowchart:

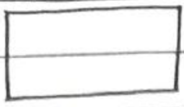
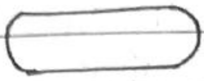
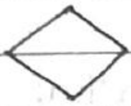
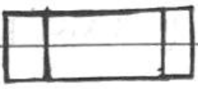
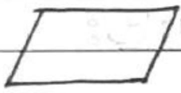
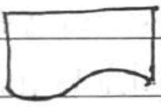
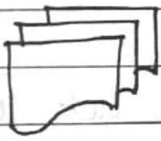
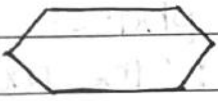
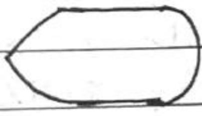
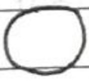
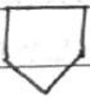
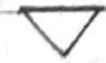
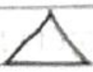
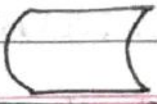
We can define flowchart as "a symbolic representation of a solution of a particular problem".

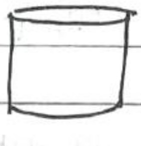
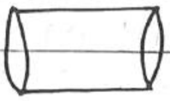
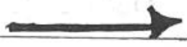
or

A flowchart is "a pictorial/graphical/symbolic representation of an algorithm".

* Flowchart Symbols

- Flowcharts are constructed or designed by using special geometrical symbols. Each symbol represents an activity. The activity could be input/output of data, computation/processing of data, taking a decision, terminating the solution, etc. The symbols are joined by arrows to obtain a complete flowchart.

Sr. No.	Symbol	Name (Alternates)	Meaning
1.		process	An operation or action step.
2.		Terminator	A start or stop point in a process or function.
3.		Decision	A question or branch in the process or function
4.		Predefined Process	A formally defined subprocess.
5.		Data (I/O)	Indicate data inputs &outputs (I/O) to and from process.
6.		Document	A document or report.
7.		Multi-document.	Same as document but well multiple document.
8.		preparation.	A preparation or set-up process step.
9.		Display	A machine display.
10.		connector	A jump from one point to another.
11.		off-page connector	Continuation onto another page.
12.		Merge (storage)	Merge multi-process into one.
13.		Extract	Exactract a measurement.
14.		Stored data	Data storage symbol in flowchart.

15.		Magnetic disk (database)	A database.
16.		Direct access storage	Storage on a hard disk.
17.		Flow Line	Indicates the direction of data flows.

☆ Advantages (Benefits) of Flowcharts:

- Proper Documentation: Flowcharts serves as a good program documentation, which is needed for various purposes.
- Efficient Coding: The flowcharts act as a guide or blueprint during the systems analysis and program developments phase.
- Efficient Program Maintenance: The maintenance of operating program becomes easy with the help of flowchart. It helps the programmer to put efforts more efficiently on that part.
- proper Debugging: The flowchart helps in debugging process.
- Communication: Flowcharts are better way of communicating the logic of a system to all concerned.
- Effective Analysis: With the help of flowchart, problem can be analysed in more effective way.

☆ Disadvantages (Limitations) of Flowcharts:
- Loss of Technical Details: The essentials of what is done can easily be lost in the technical details of how it is done.

- Complex Logic: Sometimes, the program logic is quite complicated. In that case, flowchart becomes complex and clumsy.
- Alterations and Modifications: If alterations are required, the flowchart may require redrawing completely.
- Reproduction: As the flowchart symbols cannot be typed, reproduction of flowchart becomes a problem
- Time Consuming: Developing and constructing a flowchart is time consuming. Flowchart requires more time to construct.

☆ Comparision (Difference) between Algorithm and Flowchart

Algorithm	Flowchart
An algorithm is a finite set of precise instructions for solving a problem.	A flowchart is a graphical/pictorial representation of an algorithm.
It refers to the logic for solving problem.	It is a tool which shows flow of problem solving.
An algorithm is a precise rule (or set of rules) specifying how to solve some problem. Algorithm is step-wise analysis of the work to be done.	A flowchart is a diagram of the sequence of operation in problem solving.
Algorithm gives language independent layout of the problem.	Flowchart gives logical flow of problem.
Easy to update.	Difficult to update.
Less time required for write an algorithm.	Time consuming to write and draw a flowchart.

★ Simple Example on Algorithms & Flowcharts:

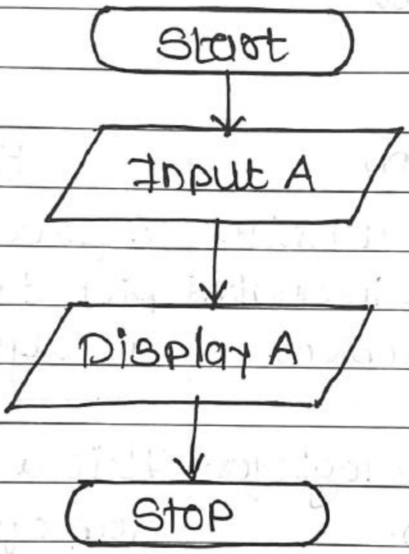
① Algorithm and Flowchart to Input a Number to the computer and Display the same number to the screen.

⇨

Algorithm:

```
Step 1 :  Start
Step 2 :  Enter number A
Step 3 :  Display A
Step 4 :  Stop.
```

Flowchart:



② Algorithm and Flowchart to convert temperature in Celsius to ~~Faren~~ Farhrenheit.

⇨ Degree Fahrenheit ($^{\circ}F$) & Degree Celsius ($^{\circ}C$) are the main two units to measure temperature. It is possible to convert a temperature from Celsius degree to Fahrenheit using following formula:

$$^{\circ}F = \left(^{\circ}C \times \frac{9}{5}\right) + 32$$

Algorithm:

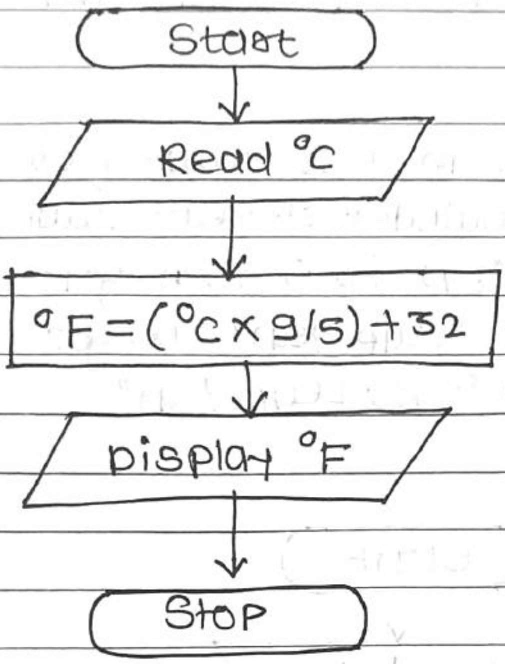
```
Step 1:  Start
Step 2:  Read °C
```

Step 3: Calculate $°F = (°C \times 9/5) + 32$

Step 4: Display °F

Step 5: Stop.

Flowchart:-

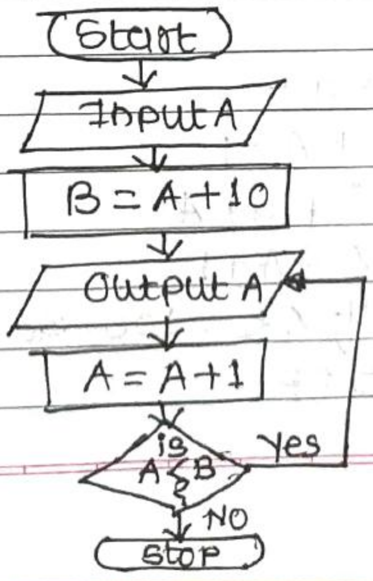


③ Algorithm and Flowchart to input one number and display Next 10 numbers in increasing order.

⇒ Algorithm:

```
Step 1: Start
Step 2: Input A
Step 3: B = A + 10
Step 4: Output A
Step 5: A = A + 1
Step 6: If A < B, go to step 4
Step 7: Stop.
```

Flowchart:



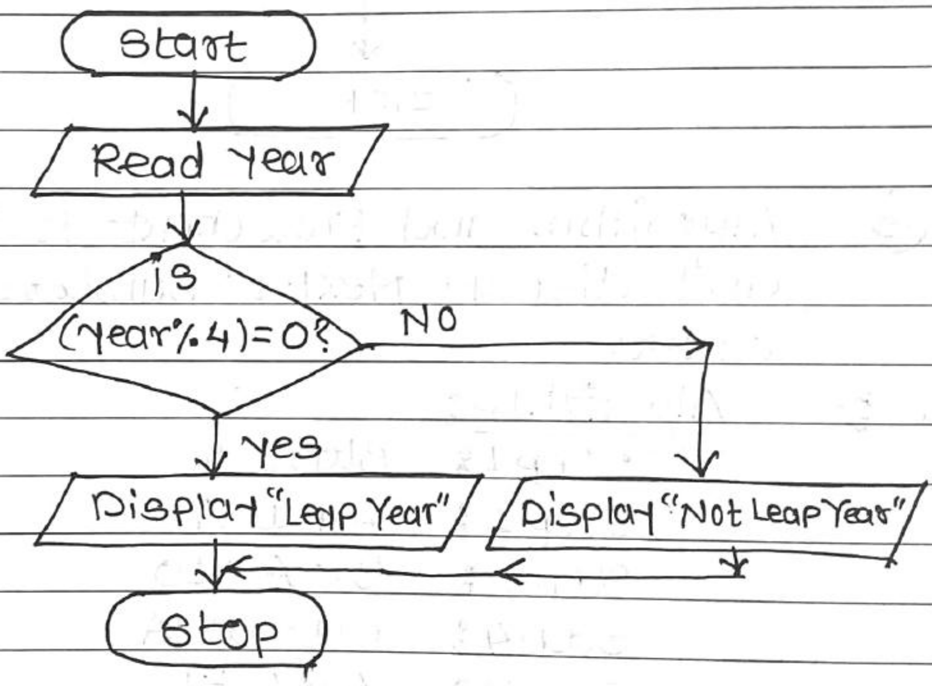
Page No.
Date

④ Algorithm and Flowchart for given Year is a Leap Year or Not.

A year is called Leap year if it is divisible by ~~400~~ 4. For example: 1600, 2000 etc leap years while ~~1500~~ 1509, 1706 are not leap years.

Algorithm:

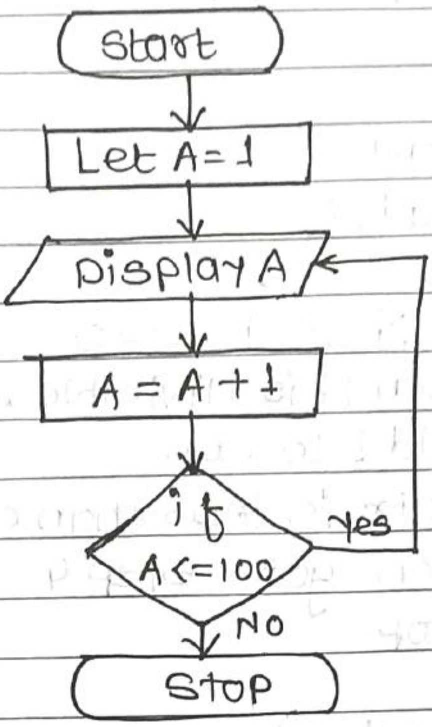
```
Step 1: Start
Step 2: Accept an year value from the user (year)
Step 3: If remainder of year value divided by 4
        (year%4) is 0 then goto step 4 else step 5
Step 4: Display "Leap Year" and goto step 6.
Step 5: Display "Not Leap Year"
Step 6: Stop.
```



⑤ Algorithm and Flowchart to display 1 through 100 Numbers.

⇨ Algorithm:

```
Step 1: Start
Step 2: Let A = 1
Step 3: Display A
Step 4: A = A + 1
Step 5: If A <= 100 Go to step 3
Step ~~5~~ 6: Stop.
```



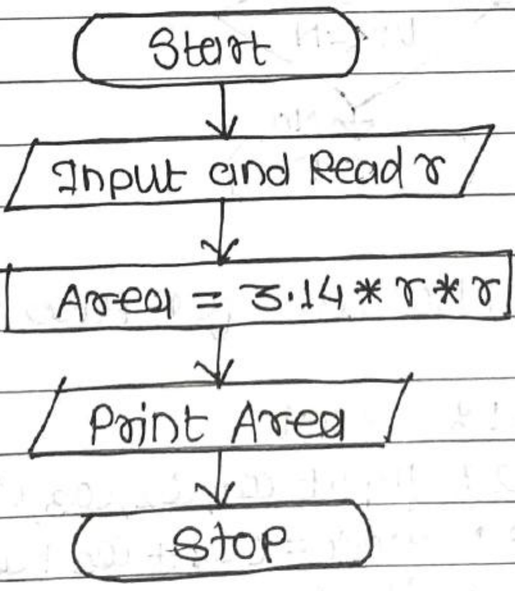
⑥ Algorithm and Flowchart to calculate the area of circle.

⟹ Area of circle measures the area enclosed by a circle. It formula is $A = \pi * r * r$.

Algorithm:

Step 1: Start
Step 2: Input and Read r.
Step 3: Area = 3.14 * r * r.
Step 4: Print Area.
Step 5: Stop

Flowchart:



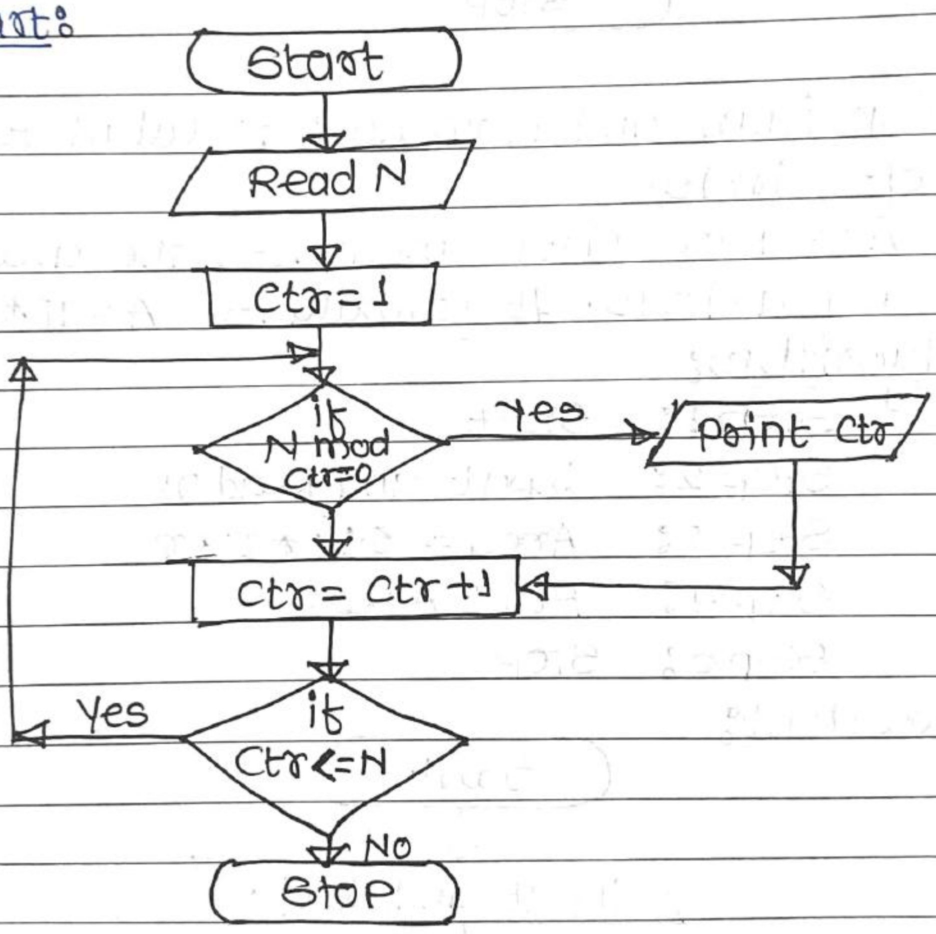
Date

⑦ Algorithm and Flowchart to print all divisions of integer N.

⇒ Algorithm:

```
Step 1: Start
Step 2: Read N
Step 3: Let Ctr=1
Step 4: If N Mod Ctr=0
            then N is divisible by Ctr, Print Ctr
Step 5: Add 1 to Ctr.
Step 6: If Ctr is less than or equal to N
            then goto step 4
Step 7: Stop.
```

Flowchart:



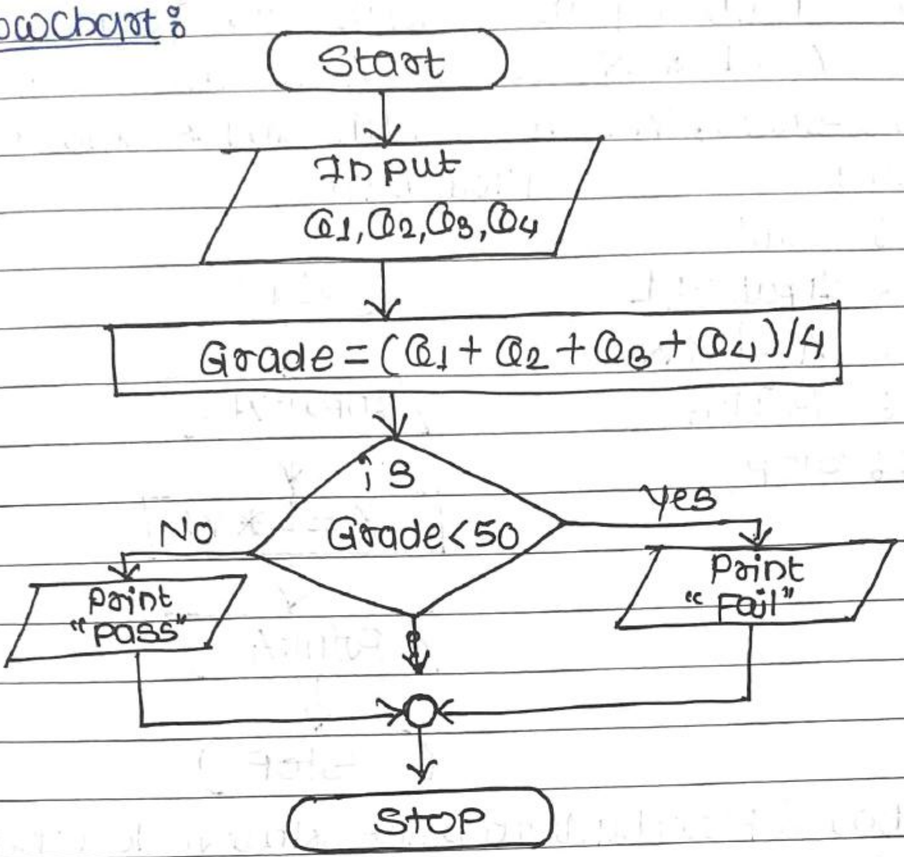
⑧ Algorithm and Flowchart for Pass or Fail.

⇒ Algorithm:

```
Step 1: Start
Step 2: Input Q1, Q2, Q3, Q4
Step 3: Grade= (Q1 + Q2 + Q3 + Q4)/4
Step 4: if (Grade < 50) then
            Print "Fail"
```

```
        else
            print "Pass"
    Step 5: Stop.
```

Flowchart:

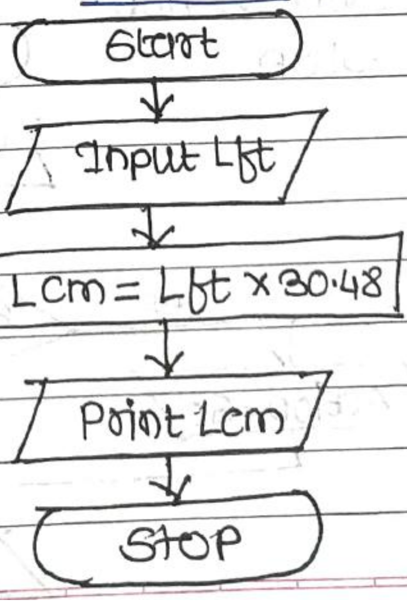


9) Algorithm & Flowchart to convert the length in Feet to Centimeter.

⇨ Algorithm:

```
Step 1: Start
Step 2: Input Lft
Step 3: Lcm = Lft × 30.48
Step 4: Print Lcm
Step 5: Stop.
```

Flowchart:



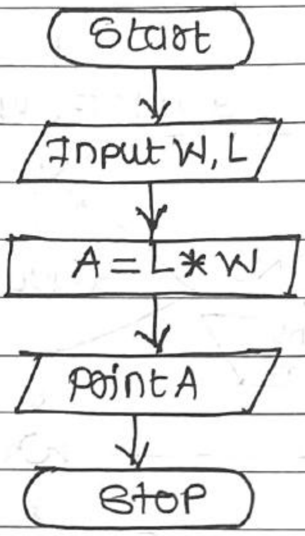
(10) Algorithm & Flowchart that will read the two sides of a Rectangle and calculates its Area.

⇒ To find the area of a rectangle, multiply the length by the width. The formula is $A = L * W$, where A is the area, L is the Length, W is the width and * means multiply.

Algorithm:

```
Step1: Start
Step2: Input W, L
Step3: A = L x W
Step4: Print A
Step5: Stop
```

Flowchart:

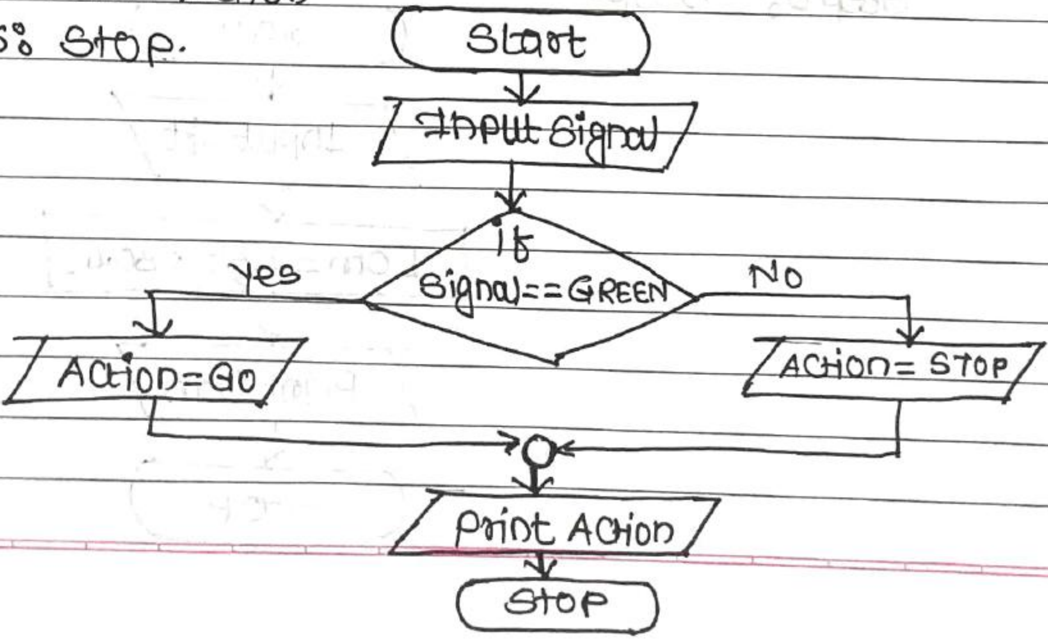


(11) Algorithm & Flowchart to print What to do when Driving to a Traffic Signal.

⇒ Algorithm:

```
Step1: Start
Step2: Read Traffic Signal
Step3: If signal is GREEN then
           set action as Go.
       Else
           set action as STOP.
Step4: Print Action
Step5: Stop.
```

Flowchart:-



(12) Algorithm and Flowchart to find the square of a number.

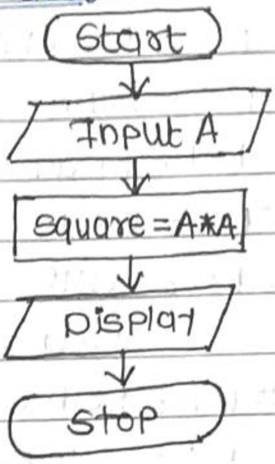
⟹ The number we get after multiplying an integer (not a fraction) by itself.

Example: 4X4 = 16, so 16 is a square number.

Algorithm:

```
Step1: Start
Step2: Input the Number A.
Step3: Let square = A*A.
Step4: Display
Step5: Stop
```

Flowchart:



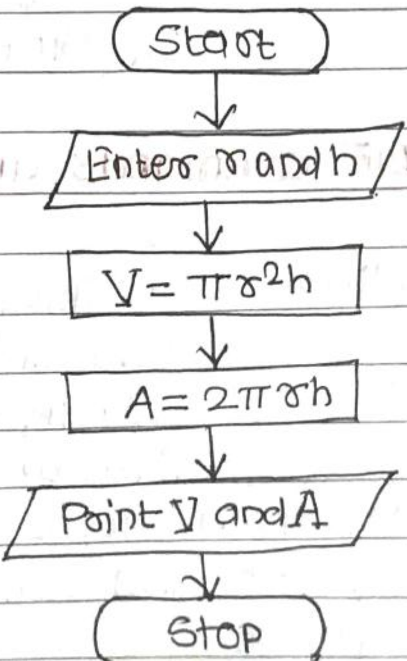
(13) Algorithm and flowchart for finding the volume and surface Area of a cylinder.

⟹ Lateral surface area of a cylinder is directly proportional to the radius and the height of the cylinder. The formula is A=2*pi*r*h. There is special formula for finding the volume of a cylinder. The volume is how much space takes up the inside of a cylinder. The formula for calculating volume (V) is π*r*r*h.

Algorithm:

```
Step1: Start
Step2: Read radius and
       height as r and h.
Step3: Calculate V=π*r*r*h
Step4: Calculate A=2*π*r*h
Step5: Print Volume and Area.
Step6: Stop.
```

Flowchart:

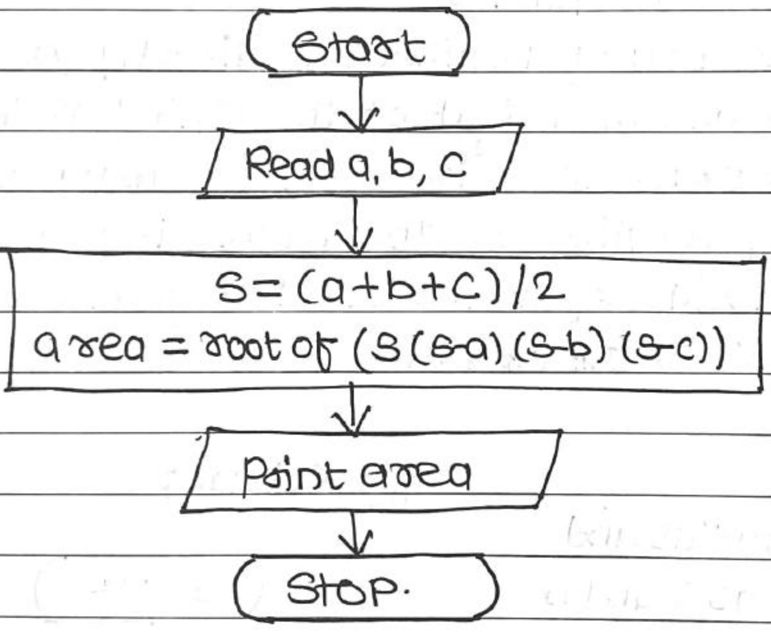


(14) Algorithm and Flowchart to find the area of a triangle for the given three sides.

⇒ Algorithm:

```
Step 1: Start
Step 2: Declare the floating variable a,b,c,s,area.
Step 3: Print the message "enter three sides"
Step 4: Read the values of side from keyboard
Step 5: Write the logic for s i.e. s=(a+b+c)/2.
Step 6: Write the logic for ~~area~~
            area = root of (s(s-a)(s-b)(s-c))
Step 7: Print the area.
Step 8: Stop.
```

Flowchart:



* IDE commands and Eclipse for C program development

- Eclipse is an integrated development environment (IDE) for Java and other programming language like C, C++, PHP, and Ruby etc. Development environment provided by Eclipse includes the Eclipse Java development tools (JDT) for Java. Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

① How to install Eclipse C/C++ Development Tool (CDT) 8.1.2 for Eclipse 4.2.2 (Juno)

Step 1: Install MinGW GCC or Cygwin GCC

To use Eclipse for C/C++ programming, you need a C/C++ compiler. On windows, you could install either MinGW GCC or Cygwin GCC. Choose MinGW if you are not sure, because MinGW is lighter and easier to install, but having less features.

→ MinGW GCC: Read "How to install MinGW".

→ Cygwin GCC: Read "How to install Cygwin". Make sure that you select "gcc", "g++", "gdb" and "make" packages under the "Devel" (Development) category - these packages are not part of the default installation.

Step 2: Install Eclipse C/C++ Development Tool (CDT)

Two ways to install CDT, depending on whether you have previously installed an Eclipse:

→ If you have already installed "Eclipse for Java Developers" or other Eclipse packages, you could install the CDT plug-in as follows:

Launch Eclipse ⇨ Help ⇨ Install New Software ⇨ In "Work with" field, pull down the drop-down menu and select "Kepler - http://download.eclipse.org/releases/Kepler" (or juno for Eclipse 4.2; or helios for Eclipse 3.7). In "Name" box, expand "programming Language" node ⇨ Check "C/C++ Development Tools" ⇨ "Next" ⇨ ... ⇨ "Finish"

→ If you have not install any Eclipse package, you could download "Eclipse IDE for C/C++ Developers" from http://www.eclipse.org/downloads, and unzip the downloaded file into a directory of your choice.

Step 3: Configuration

You ~~donot~~ do NOT need to do any configuration, as long as the Cygwin or MinGW binaries are included in the

PATH environment variable. CDT searches the PATH to discover the C/C++ compilers.

② Writing your first C/C++ program in Eclipse 2.1 C++ program.

Step 1: Launch Eclipse

- Start Eclipse by running "eclipse.exe" in the Eclipse installed directory.
- Choose an appropriate directory for your workplace (i.e. where you would like to save your works).
- If the "Welcome" screen shows up, close it by clicking the "close" button.

Step 2: Create a new C++ project

For each C++ application, you need to create a project to keep all the source codes, object files, executable files, and relevant resources.

To create a new C++ project:

→ Choose "File" menu ⇒ "New" ⇒ Project... ⇒ C/C++ ⇒ C++ project.

→ The "C++ project" dialog pops up.

↔ In "project name" field, enter "Helloco World".

↔ In "project types" box, select "Executable" ⇒ "Empty project".

↔ In "Toolchains" box, choose your compiler e.g. "Cygwin GCC" or "MinGW GCC" ⇒ Next.

→ The "select configurations" dialog appears. select both "debug" and "Release" ⇒ Finish.

Step 3: Write a Hello-World C++ program

→ In the "project Explorer" (leftmost panel) ⇒ Right-click on "HelloWorld" (or use the "File" menu) ⇒ New ⇒ Source File.

→ The "New Source File" dialog pops up.

↔ In "source file" field, enter "Hello.cpp".

↔ Click "Finish".

→ The source file "Hello.cpp" opens on the editor panel (double-click on "test.cpp" to open it necessary).

Enter the following codes:

```
#include <iostream>
 using namespace std;
  int main(){
   cout << "Hello, World!" << endl;
   return();
   }
```

If "Unresolved Inclusion Error"

If error "unresolved inclusion" appears next to #include statement, the "include path for headers" are not set properly. Select "project" menu ⇒ properties ⇒ C/C++ General ⇒ paths and symbols ⇒ In "Includes" tab:

★ For Cygwin GCC:

1. "Add the following directories to "GNU C", where $CYGWIN_HOME is your cygwin installed directory:
→ $CYGWIN-HOME\lib\gcc\i686-pc-cygwin\4.5x\include
→ $CYGWIN-HOME\lib\gcc\i686-pc-cygwin\4.5x\include-fixed
→ $CYGWIN-HOME\usr\include
→ $CYGWIN-HOME\usr\include\W32api

2. "Add" the following directories to "GNU C++", where $CYGWIN_HOME is your Cygwin installed directory:
→ $CYGWIN-HOME\lib\gcc\i686-pc-cygwin\4.5x\include\C++
→ $CYGWIN-HOME\lib\gcc\i686-pc-cygwin\4.5x\include\C++\ ~~backward~~ i686-pc-cygwin
→ $CYGWIN-HOME\lib\gcc\i686-pc-cygwin\4.5x\include\C++\backward
→ $CYGWIN_HOME\lib\gcc\i686-pc-cygwin\4.5x\include
→ $CYGWIN-HOME\lib\gcc\i686-pc-cygwin\4.5x\include-fixed
→ $CYGWIN-HOME\usr\include
→ $CYGWIN-HOME\usr\include\W32api

★ For MinGW GCC:

1. "Add" the following directories to "GNU C", where $MINGW_HOME is your MinGW installed directory:

→ $MINGW-HOME\lib\gcc\mingw32\4.6x\include
→ $MINGW-HOME\include
→ $MINGW-HOME\lib\gcc\mingw32\4.6x\include-fixed

2. "Add" the following directories to "GNU C++", where $MINGW-HOME is your Cygwin installed directory:
→ $MINGW-HOME\lib\gcc\mingw32\4.6x\include\C++
→ $MINGW-HOME\lib\gcc\mingw32\4.6x\include\C++\mingw32
→ $MINGW-HOME\lib\gcc\mingw32\4.6x\include\C++\backward
→ $MINGW-HOME\lib\gcc\mingw32\4.6x\include
→ $MINGW-HOME\include
→ $MINGW-HOME\lib\gcc\mingw32\4.6x\include-fixed

NOTE:- If you encounter "error while loading shared libraries during link. Install "libmpfr4" in Cygwin.

Step 4: Compile/Build

Right-click on the "HelloWord" (or use the "project" menu) ⇒ choose "Build Project" to compile and link the program.

Step 5: Run

To run the program, right-click on the "HelloWorld" (or anywhere on the source "test.cpp", or select the "Run" menu) ⇒ Run As ⇒ Local C/C++ Application ⇒ (If ask, choose Cygwin's gdb debugger) ⇒ the output "Hello, World!" appears on the "console" panel.