



Humans communicate through language, animals communicate through gestures, noise and actions. If a person wants to explain something to another person, how will he do it? The answer is simple he/she will explain it verbally through a language understood by both.

मराठी मलयाळം हिन्दी
ગુજરાતી తెలుగు లిపి മലയാളം
भाषा বাংলা
ଓଡ଼ିଆ ବ୍ଲା ଚନ୍ନଡ଼ संस्कृतम्
விக்சிப்பீடியா اُردُو অসমীয়া

Programming languages generally consist of instructions for a computer. Programming languages can be used to create programs that implement specific algorithms.

In other words, A programming language is a special language that programmers are use to develop software programs, scripts, or other sets of instructions for computers to execute.



The most common and basic languages are C, C++, Java, Python, Php, C#, VB, SQL and Swift etc. These are programming languages. Instructions or programs are written in these programming languages.





1.1.1 Types of computer programming languages

Programming languages are used to communicate with the computer through a set of code and instruction which are collectively known as Programs.

The person who writes the code using a programming language is known as a Programmer or a Coder. A programmer is a person who writes programs using programming languages. A programmer writes programs using characters such as numbers, symbols, letters etc. each computer language has its own set of characters.



Mainly there are three major types of programming languages:



1.1.1.1 Machine Language

The only language that is understood directly by a computer is the Machine Language. Machine Language is also known as Binary language.



Machine Language is the lowest level of programming language and was the first type of programming language to be developed. Machine Language only consists of 0's and 1's.

Machine language uses strings of 1's (ones) and 0's (zeros) codes as a character sets for writing programs. Therefore, when the computer is given the sequence of binary code, it identifies the code and executes the program.

Advantages of Machine Language

- As machine language is the only language understood by the computer, it does not require any translator or mediator.
- Machine language is very fast and utilizes the computer efficiently.



Disadvantages of Machine Language

- All the code has to be remembered and as the code consists of only 1's and 0's it is very difficult.
- It is very inefficient for programmers.
- It is also very difficult to find errors or mistakes in a program written in Machine Language.





1.1.1.2 Assembly Language

Intermediate-level programming language which is higher than the Machine language. Code written in assembly language are converted into machine language by translators.

These translators are specialized programs called assemblers. The Assembly Language is considered to be a second-generation language.

ADD A, byte	add A to byte, put result in A
ADDC A, byte	add with carry
SUBB A, byte	subtract with borrow
INC A	increment A
INC byte	increment byte in memory
INC DPTR	increment data pointer

Advantages of Assembly Language

- The Assembly Language is easier to use and understand as compared to the machine language.
- It saves a lot of time and effort of the programmer, because it uses character sets. This provides some flexibility compare to machine language.
- It is easy to correct errors and hence, it is easy to modify the code.



Disadvantages of Assembly Language

- Like machine language, assembly language is also machine dependent.
- As it is machine dependent, the programmer also needs to understand the hardware.



1.1.1.3 High level languages

High-Level languages were introduced in the 1950's. High level Languages also abbreviated as HLL are computer language which are developed independent of a computer i.e. that they are not limited by the computer and are designed for a specific job.

It is much easier to understand than Machine Language. But, for a computer to interpret and execute a program created in a High-Level language, it must be compiled into Machine Language. The translator that is used for converting High-Level language to Machine Language is known as Compiler.



Advantages of Hight level languages

- High-Level languages are more user friendly.





Dr. Shyam N. Chawda, C Language Tutorial , 78 74 39 11 91

- The code and instructions are very similar and use well known English vocabulary and symbols.
- High-Level languages are easier to learn and maintain.
- High-Level languages are problem oriented rather than machine based and are designed to solve specific problems.
- High-Level languages can be translated into many Machine languages and can run on any computer, if the computer has an appropriate translator.



Disadvantages of High level languages

- High-Level language utilizes more time as it must be translated into machine language by a translator.
- Difficult to manage the libraries.

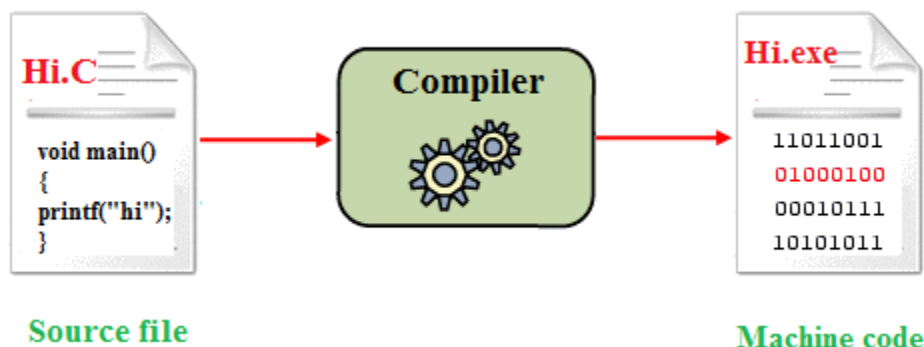


Compiler

Compiler is a software.

Compiler is a program that converts a High-Level language into Machine language that a computer can understand and execute. This interpretation is must as computer only understands Machine language.

While a compiling a code, compiler will report any error it finds in the code. If it does not find any error it translates the code into Machine language.



1.1.1.4 Procedure Languages

A procedural programming language is simply a programming language where you write statements for the computer to execute. You simply tell the computer what to do in basic steps. If any instruction in a program is complicated to execute it would be divided into fragments of small subroutines. Thus, procedural programming involves decomposition of a program into small procedures to make it simple.

High level languages, such as COBOL, FORTRAN, Pascal, and C, are known as procedural languages because they use a sequence of instructions to specify on how to solve a problem.





Advantages of Procedural Programming

- It is easier to write and debug the programs.
- It can divide the program into different modules and subroutines.
- In procedural programming, one can reuse the program.
- An easier way to keep track of program flow.
- Requires less memory.



Disadvantages of Procedural Programming

- When the program is fragmented into modules and subroutines, the data is exposed to the whole program. Hence there is no security for data.
- It is difficult to relate with real world objects.
- Significance is given to the operation on data rather than data.





1.2 Flowchart

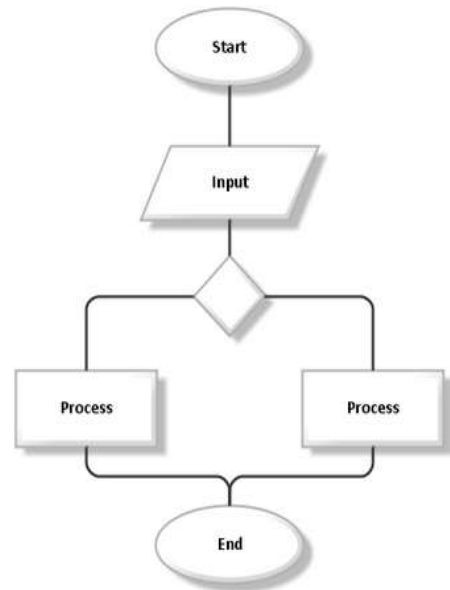
A flowchart is a graphical representation of the plan.

A flowchart is' a diagrammatic or pictorial representation of the algorithm.

Flowchart is a type of program design technique. Before writing a program in a computer language it should be designed using flowchart to make the program easier to convert into computer languages.

Flowchart is a visual representation of sequence of steps and decisions needed to perform a process to get the solution.

Flowchart is like a blueprint that is created before starting the construction on a building. Likewise, a programmer chooses to illustrate a flowchart before writing the program in computer languages. The symbols used in the flowchart are easy to learn.



Advantages of the flowchart

- Flowcharts are a better way to communicate the logic of a program.
- Flowchart helps in analyzing the program in a more effective way ultimately reducing the cost and wastage of time.
- A Problem can be examined in an efficient way.
- Program flowcharts serve as a good programming documentation, which is required for several tasks.
- The flowchart also aids in debugging process.
- The maintenance of operating program becomes easy with the help of flowchart. Hence a programmer would initially put more efforts in designing a flowchart.




Disadvantages of the flowchart

- Sometimes, the program logic is quite complicated. In that case, flowchart becomes complex and clumsy.
- Flowchart are quite costly to produce and difficult to use and manage.
- Time consuming






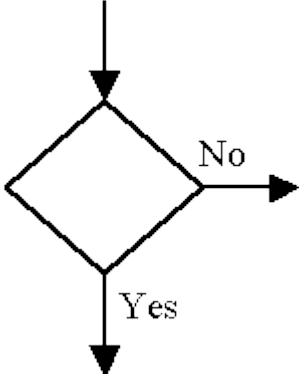
1.2.1 Symbols of flowchart

Listed below are the standard symbols used in a flowchart:

	Start/Stop of Program flow (rounded rectangle)
-------------------------------------------------------------------------------------	------------------------------------------------





	Input / Output Operation (parallelogram)
	Connector
	Process to be performed (rectangle)
	Decision / Comparison / Operation <i>Note that one arrow goes in, two go out</i>

1.2.2 Guidelines for preparing flowchart

1. Every requirement should be noted down out in a logical order.
2. Use consistent design. The shapes, line and text in a flowchart should be uniform.
3. The whole flowchart should be on one page.
4. A flowchart should always have the top to bottom approach.
5. If the flowchart is horizontal, then the data flow should be form left to right.
6. Every flowchart should be started with start symbol and ended with end symbol.
7. Always use the standard flowchart symbols.
8. Only one flow line should come out from a process symbol.
9. Instead of the traditional decision symbol use a split path.
10. Ensure that the flowchart has a logical start and finish
11. It is always good practice to test the flowchart by passing a trial data before finalizing it.



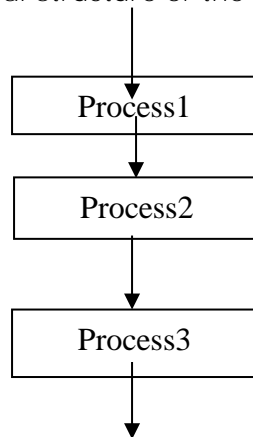


1.2.3 Flowchart structure

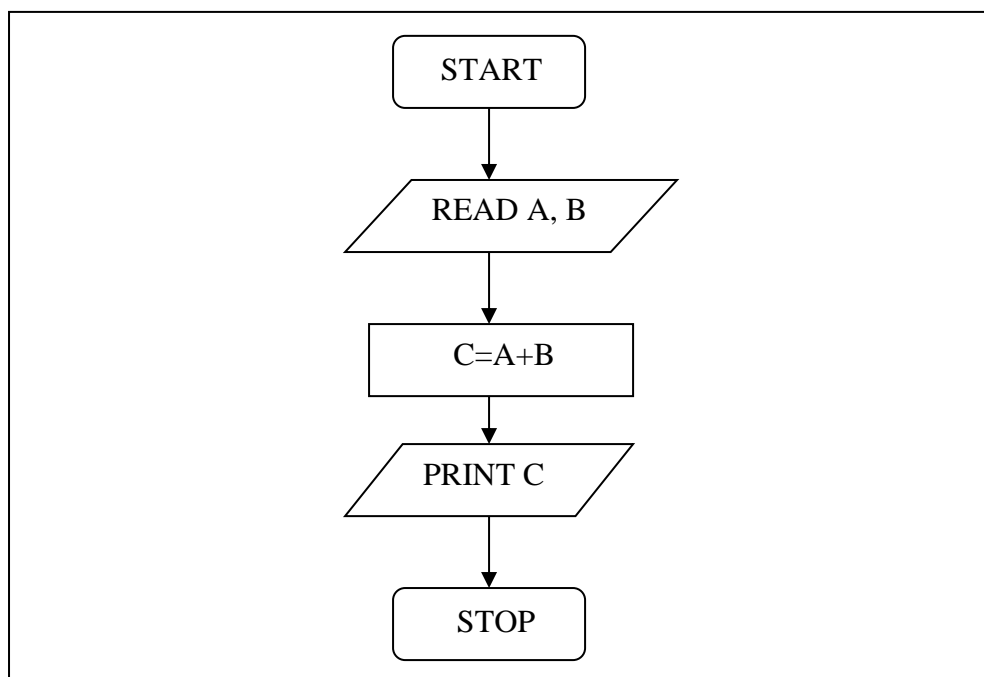
Mainly there are three aspects of a flowchart: Sequential, selection (branching) and repetition.

1.2.3.1 Sequential

The sequential process is just a series of processes carried out one by one. Following is the sequential structure of the flowchart.



Example# To find out sum of two numbers.

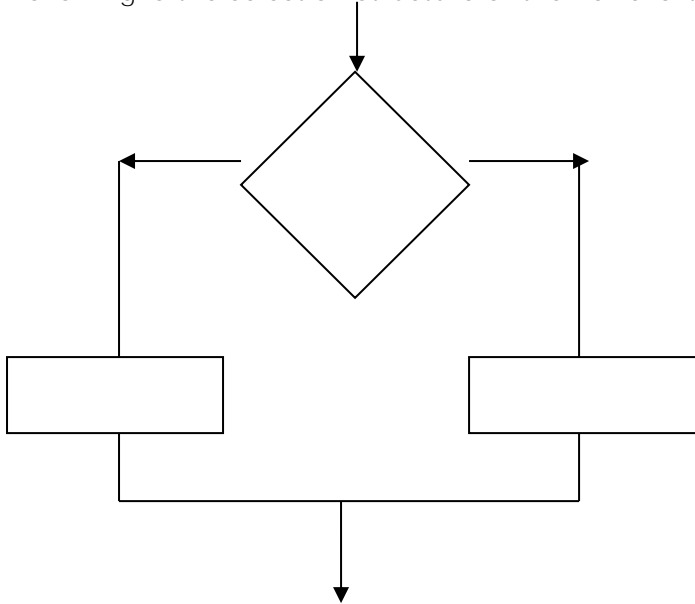




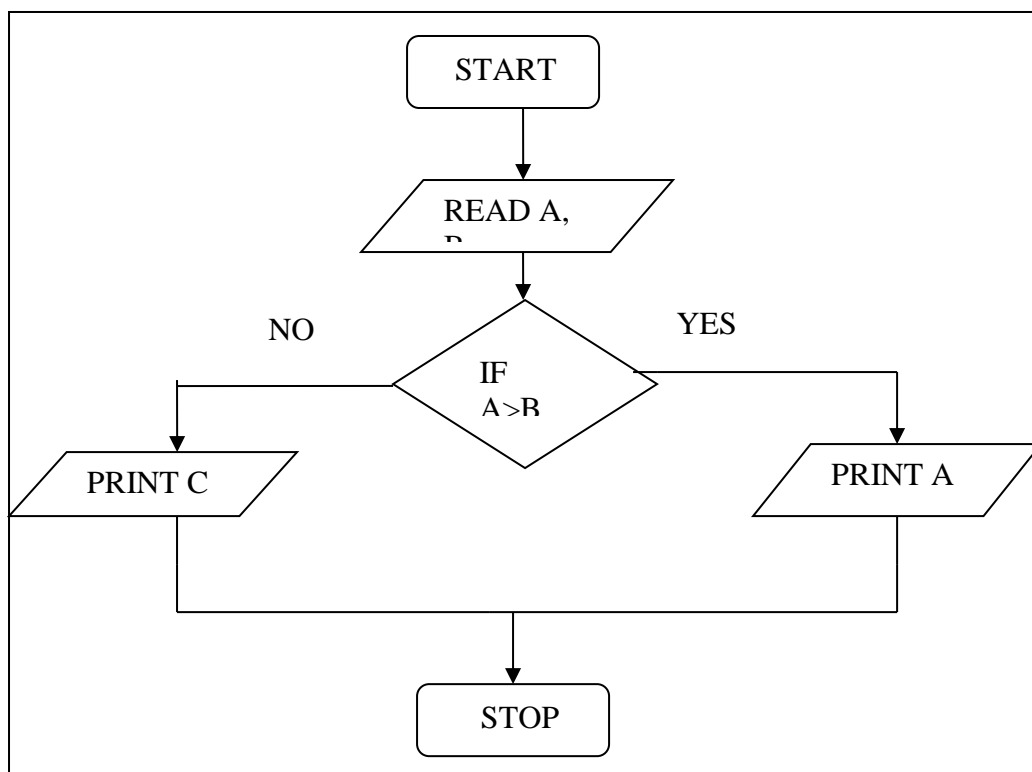
1.2.3.2 Selection

In this structure, decision is to be made and one of the processes is to be carried out in each path from the decision.

Following is the selection structure of the flowchart.



Example# To find maximum value between two values.

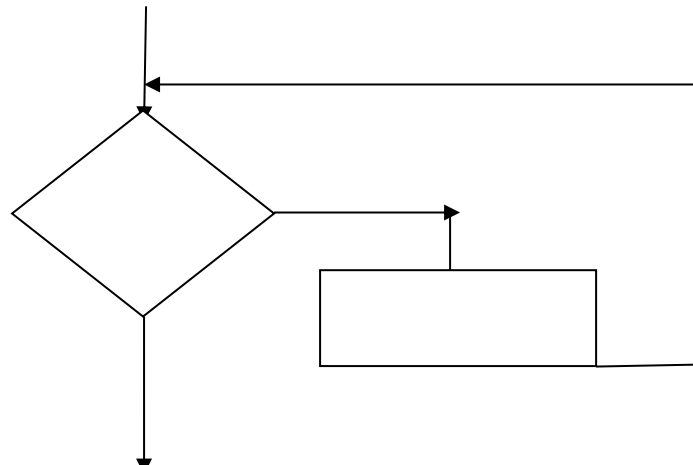




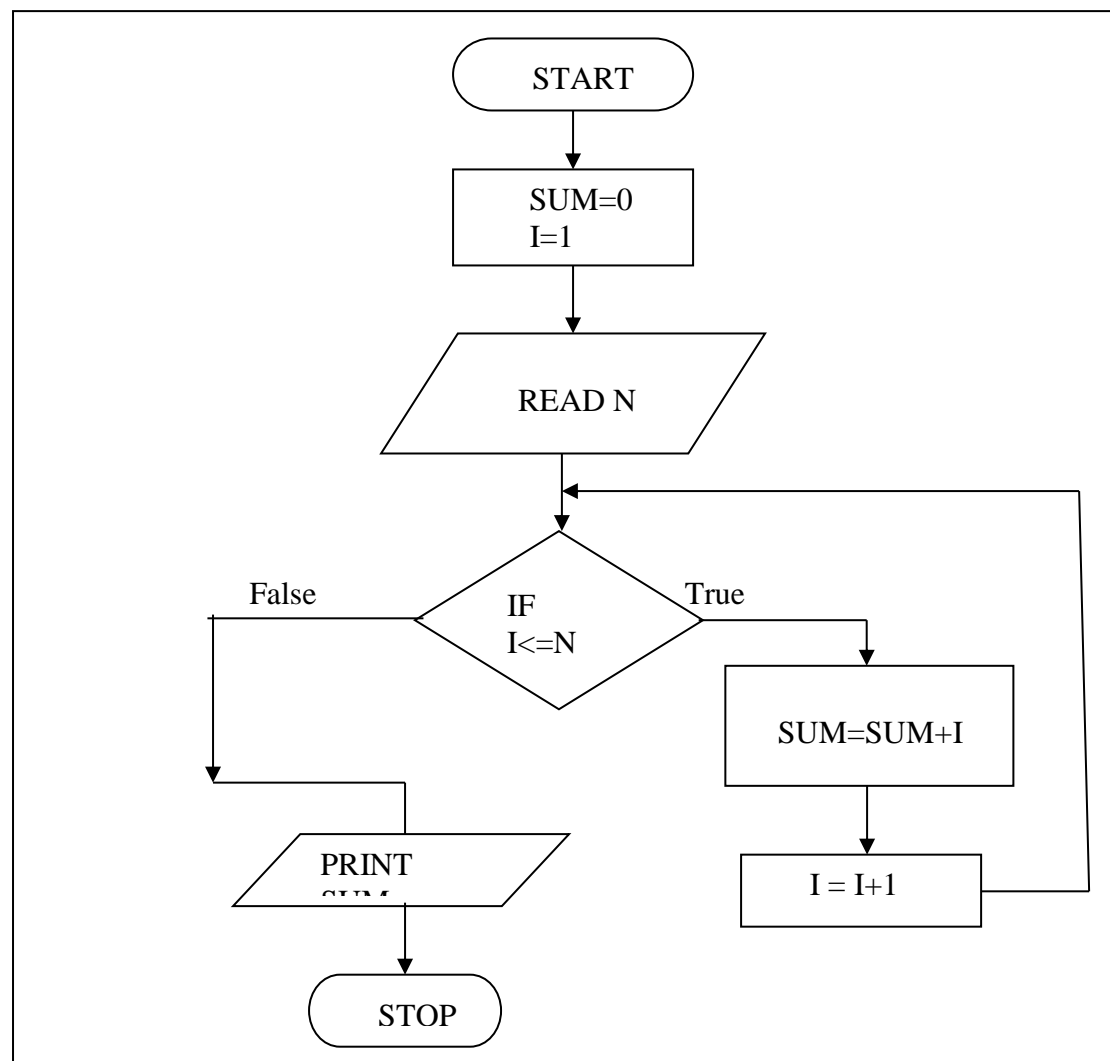
1.2.3.3 Repetition

This structure allows representation of a conditional loop. The decision to execute the process in the loop is made prior to the first execution of the process.

Following is the repetition structure of the flowchart.



Example# To find out the sum of series $1+2+3+\dots+N$ numbers.





1.3 Algorithm

Algorithm is a problem-solving technique.

An algorithm is a sequence of steps for solving a problem. Here the instructions are written in English words and that are similar to the high-level languages instructions.

It is also called pseudo-code and is commonly used to express algorithms.

```
Step 1: Start
Step 2: Read N
Step 3: Is (N%2=0) then
        Print "Even"
        else
        Print "Odd"
Step 4: Stop
```

Properties of an Algorithm

- Every instruction must be clear and have single clearly defined meaning.
- Every algorithm has input/output.
- It must have sequence of finite instructions; it means instructions must be finished after a fixed time duration.
- Algorithm should be simple.
- Algorithm should be clear with no ambiguity.
- Algorithm should lead to a unique solution of the problem.

Advantages of the algorithm

- It is simple to write and understand.
- It is not a programming language and hence any user can write it.
- Errors can be easily highlighted.
- From the algorithm, programmer can write the programs using high level languages.



Disadvantages of the algorithm

- It is not familiar for big projects.
- Writing algorithm takes a long time.



1.3.1 Algorithm Structure

An algorithm has following parts and these parts are always in below shown order:

Header	It specifies the name of the algorithm.
Declaration	Describes the algorithm.
Body	Sequence of steps for the solution.
Terminator	End statement to terminate algorithm.





1.3.2 How to write algorithm

An algorithm can be written in pseudo code for the different types of operations:

- 1 To read a data.
 Use Read or Get instructions
 For example, Read name, no1
- 2 To print the data
 Use Print or Write Instructions
 For example, print "The average is", average
3. To perform arithmetic operation
 Add number to sum or
 $Sum = sum + Number$
4. To assign some value to a variable
 Variable=value or set variable to value
 For example, Total =0
 sum=0
5. To compare the value
 Use selection type instructions if...else ...endif
 For example,
 If number $a < b$ then
 Print "maximum is ",a
 else
 Print "maximum is ",b
 end-if





6. To perform repetitions operation

Use looping type instructions repeat...until or while..do

For example,

```
Repeat until I = 10
    read number
    write number
    I = I + 1
end-repeat
```

OR

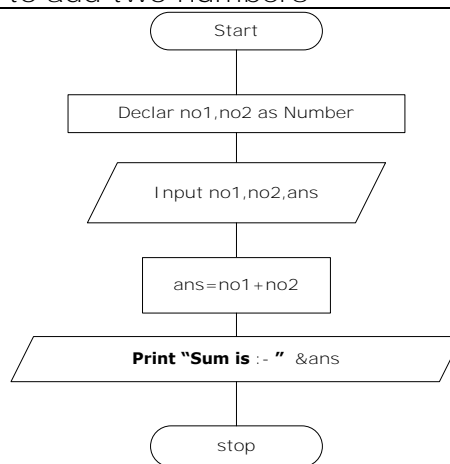
```
while I < 10 do:
    read number
    write number
    I = I + 1
end-while
```

Algorithm and Flowchart

Write an algorithm and flowchart to add two numbers

Step

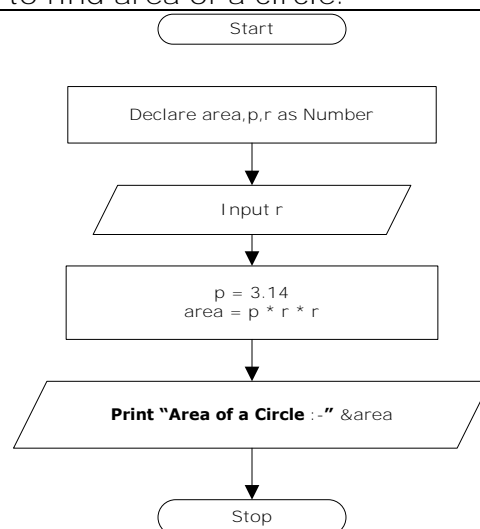
1. Start
2. Declare two variables a,b
3. **Print "Enter value of a & b ->"**
4. Input a and b
5. **Print "Ans = " & (a+b)**
6. End



Write an algorithm and flowchart to find area of a circle.

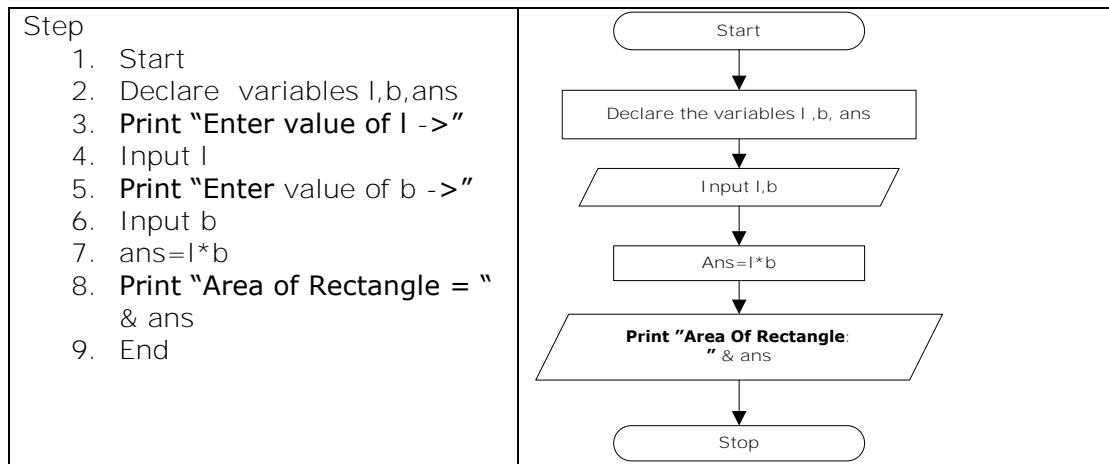
Step

1. Start
2. Declare two variables r,ans
3. **Print "Enter value of r ->"**
4. Input r
5. $ans = 3.14 * r * r$
6. **Print "Area of circle = " & ans**
7. End

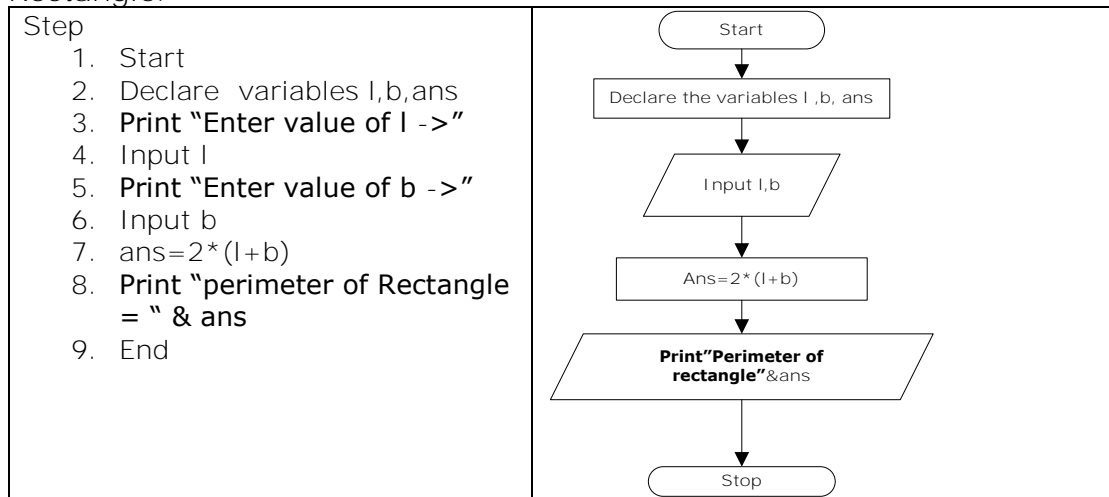


Write an algorithm and flowchart to find area of a rectangle.



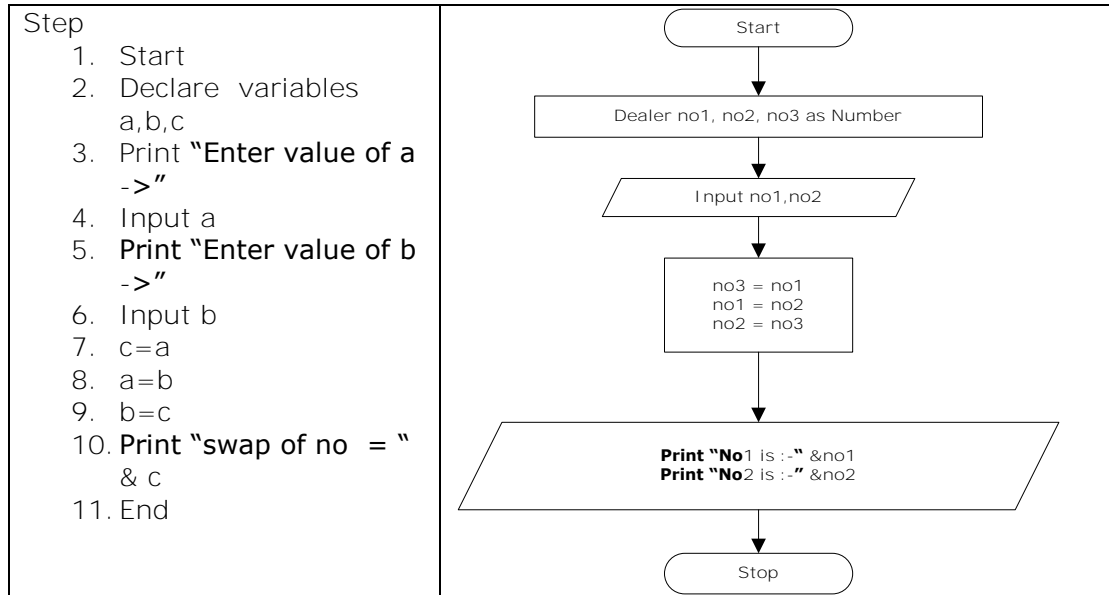


Write an algorithm and flowchart to find area of a perimeter of Rectangle.

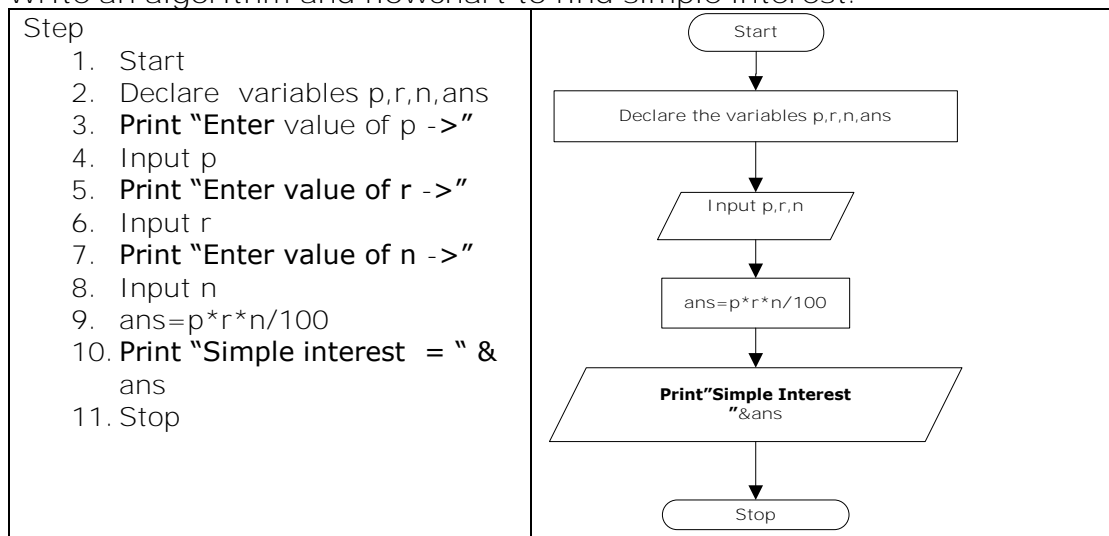


Write an algorithm and flowchart to swap the value of two variables.



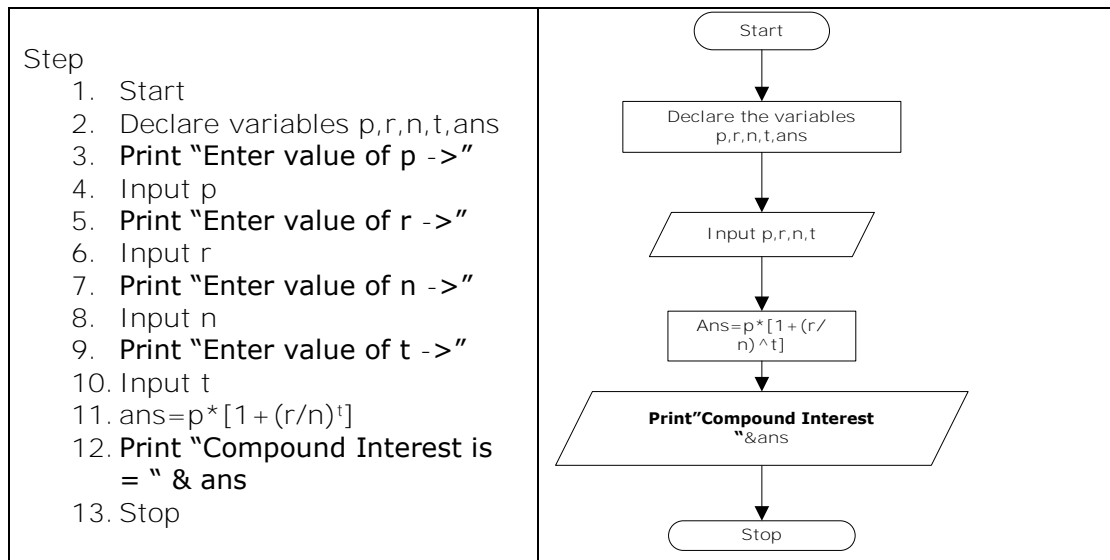


Write an algorithm and flowchart to find simple interest.

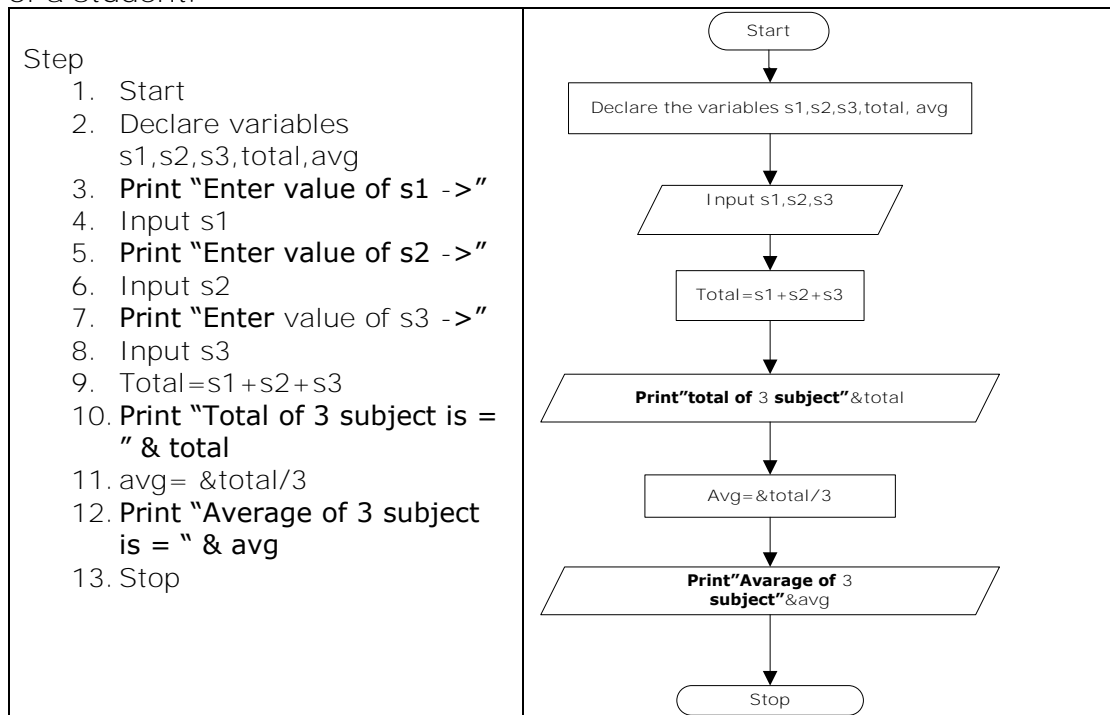


Write an algorithm and flowchart to find Compound interest.





Write an algorithm and flowchart to find the average marks of 3 subjects of a student.



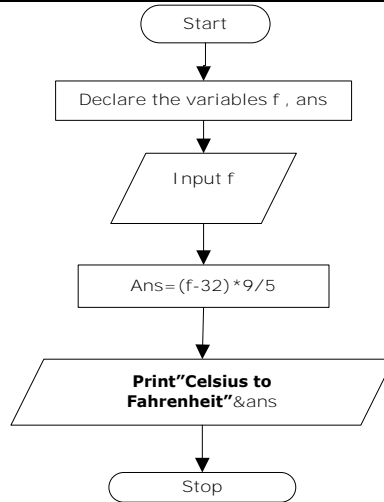
Write an algorithm and flowchart to convert temperature from Celsius to Fahrenheit.





Step

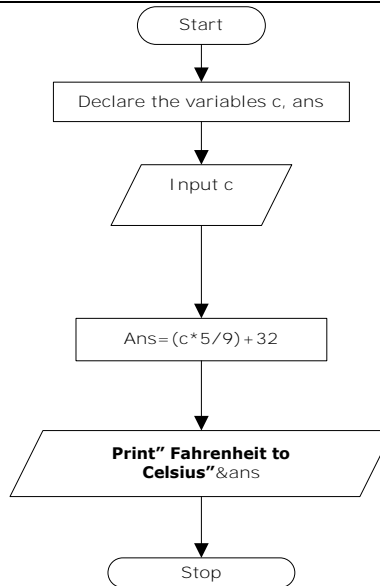
1. Start
2. Declare variables f,ans
3. **Print "Enter value of Fahrenheit: ->"**
4. Input f
5. $\text{ans} = (f - 32) * 9 / 5$
6. **Print "Celsius to Fahrenheit = " & ans**
7. Stop



Write an algorithm and flowchart to convert temperature from Fahrenheit to Celsius.

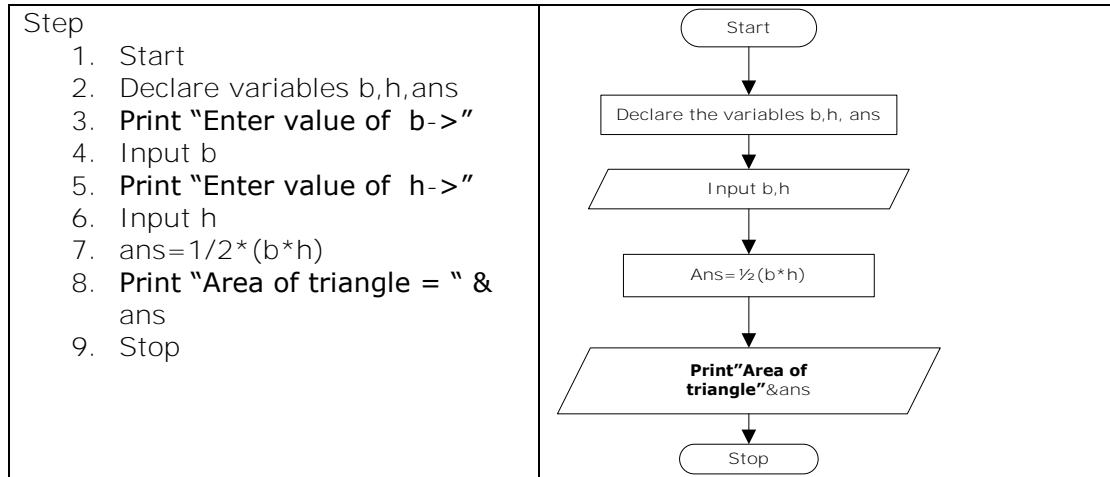
Step

1. Start
2. Declare variables c,ans
3. **Print "Enter value of celsius: ->"**
4. Input c
5. $\text{ans} = (c * 5 / 9) + 32$
6. **Print "Fahrenheit to Celsius = " & ans**
7. Stop

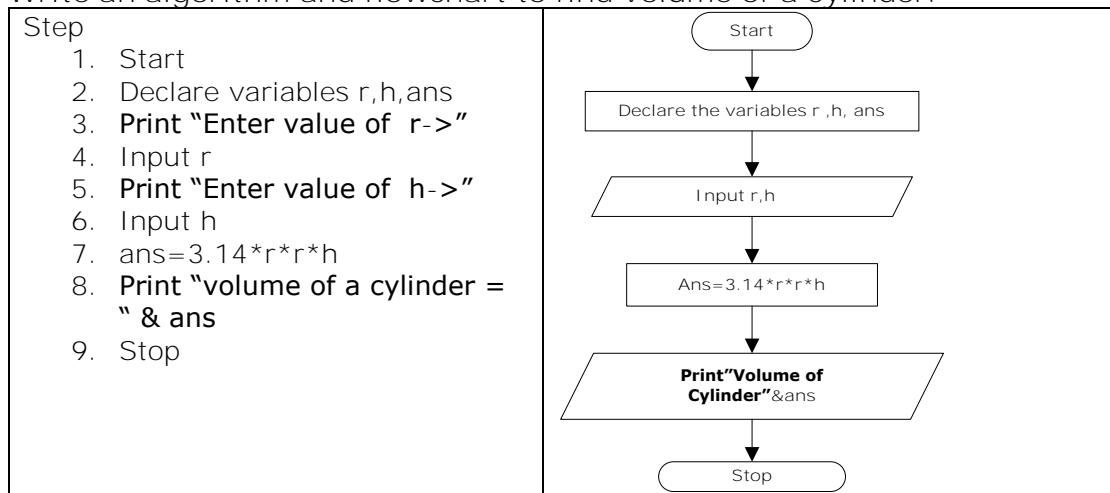


Write an algorithm and flowchart to find area of triangle.

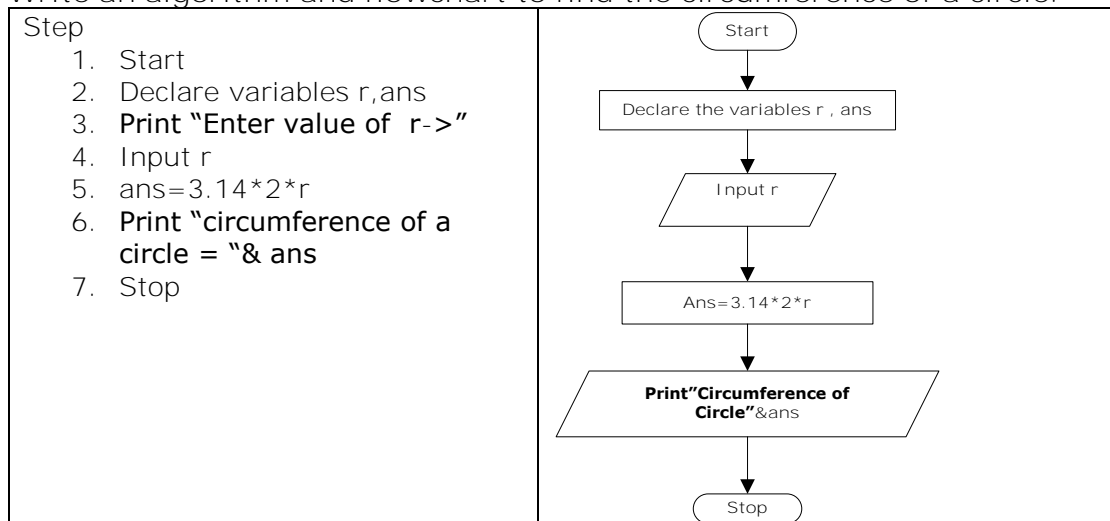




Write an algorithm and flowchart to find volume of a cylinder.



Write an algorithm and flowchart to find the circumference of a circle.



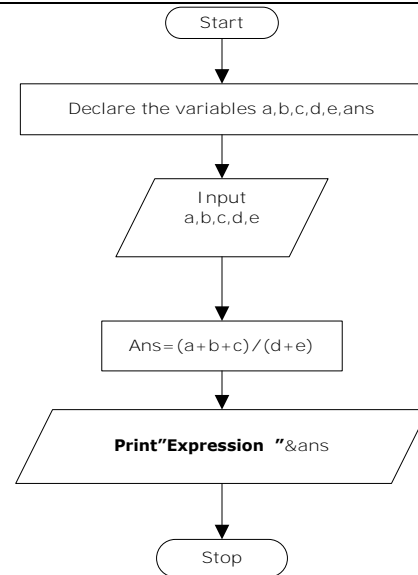
Write an algorithm and flowchart to solve the expression $x = (a + b + c) / (d + e)$.





Step

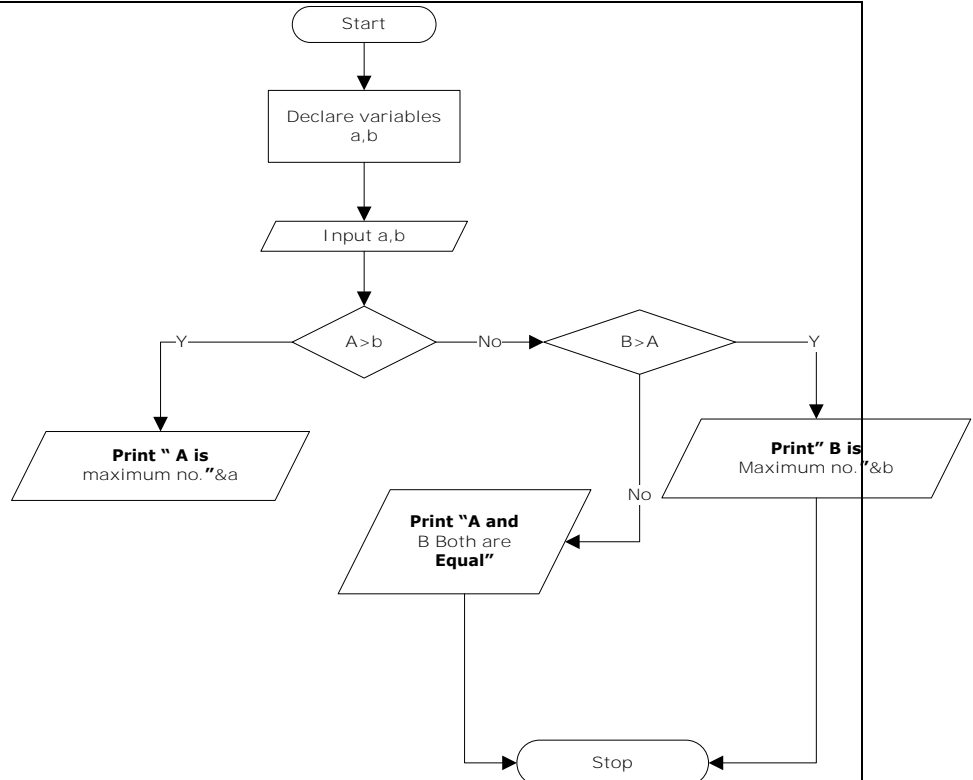
1. Start
2. Declare variables
a,b,c,d,e,ans
3. **Print "Enter value of a->"**
4. Input a
5. **Print "Enter value of b->"**
6. Input b
7. **Print "Enter value of c->"**
8. Input c
9. **Print "Enter value of d->"**
10. Input d
11. **Print "Enter value of e->"**
12. Input e
13. $\text{ans} = (a+b+c)/(d+e)$
14. **Print "Ans is = "& ans**
15. Stop



Write an algorithm and flowchart to find the maximum of two numbers

Step

1. Start
2. Declare two variable
s a,b
3. **Print "Enter value a and b ->"**
4. Input a & b
5. If $a > b$ then
6. **Print "a is maximum"**
7. Else if $b > a$ then
8. **Print "b is maximum"**
9. Else
10. **Print "Both are same"**
11. Stop

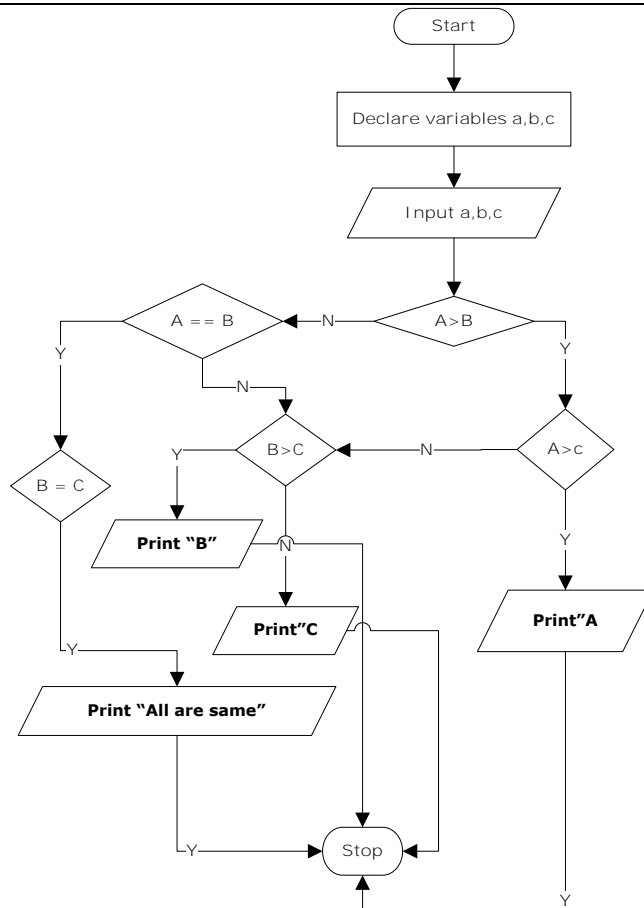




Write an algorithm and flowchart to find the maximum of three numbers

Step

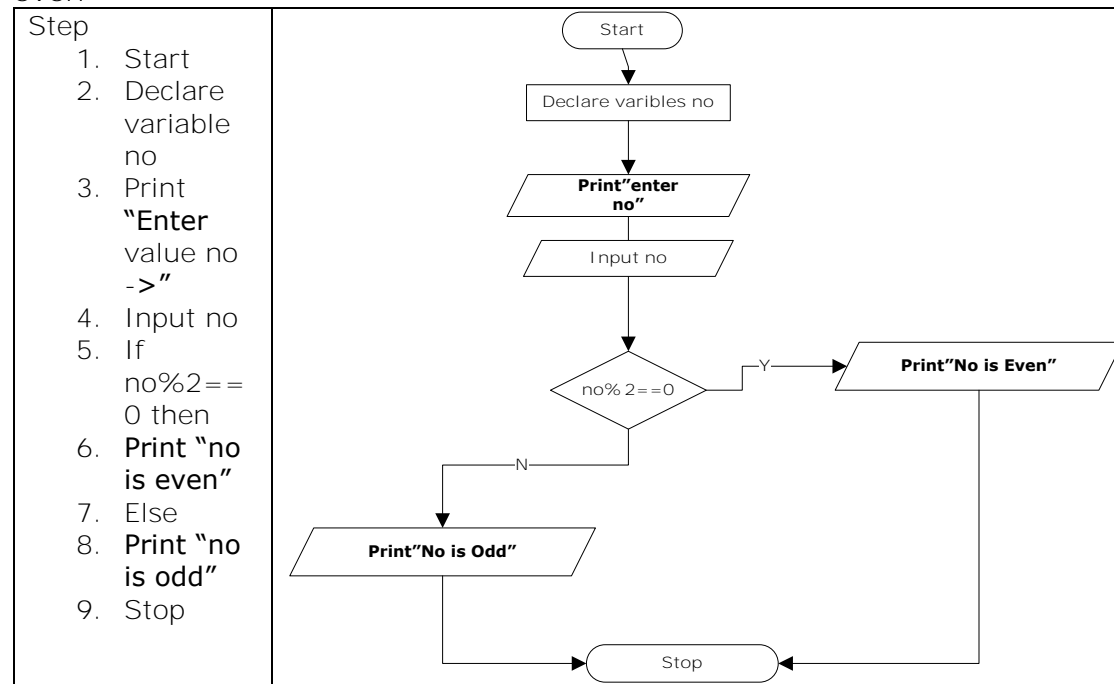
1. Start
2. Declare two variables a,b,c
3. **Print "Enter value a , b , c ->"**
4. Input a, b,c
5. If $a > b$ && $a > c$ then
6. **Print "a is maximum"**
7. Else if $b > a$ && $b > c$ then
8. **Print "b is maximum"**
9. Else if $c > a$ && $c > b$ then
10. **Print "c is maximum"**
11. Else
12. **Print "All are same"**
13. Stop



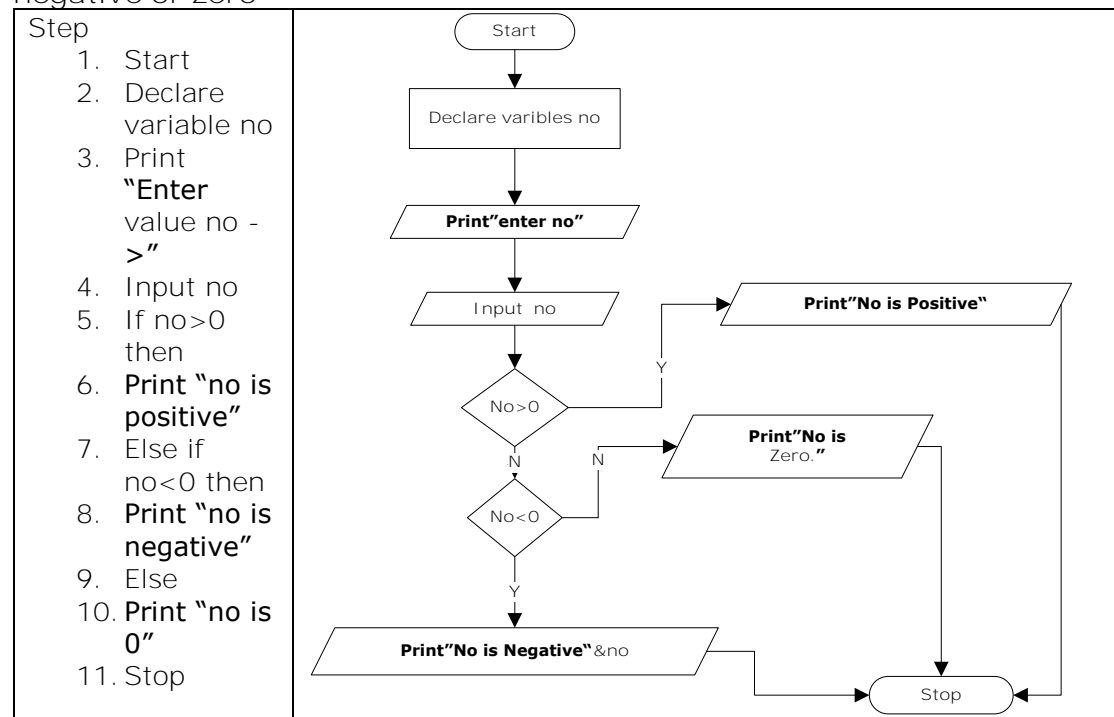


Dr. Shyam N. Chawda, C Language Tutorial , 78 74 39 11 91

Write an algorithm and flowchart to find if the given number is odd or even



Write an algorithm and flowchart to find if the given number is positive, negative or zero



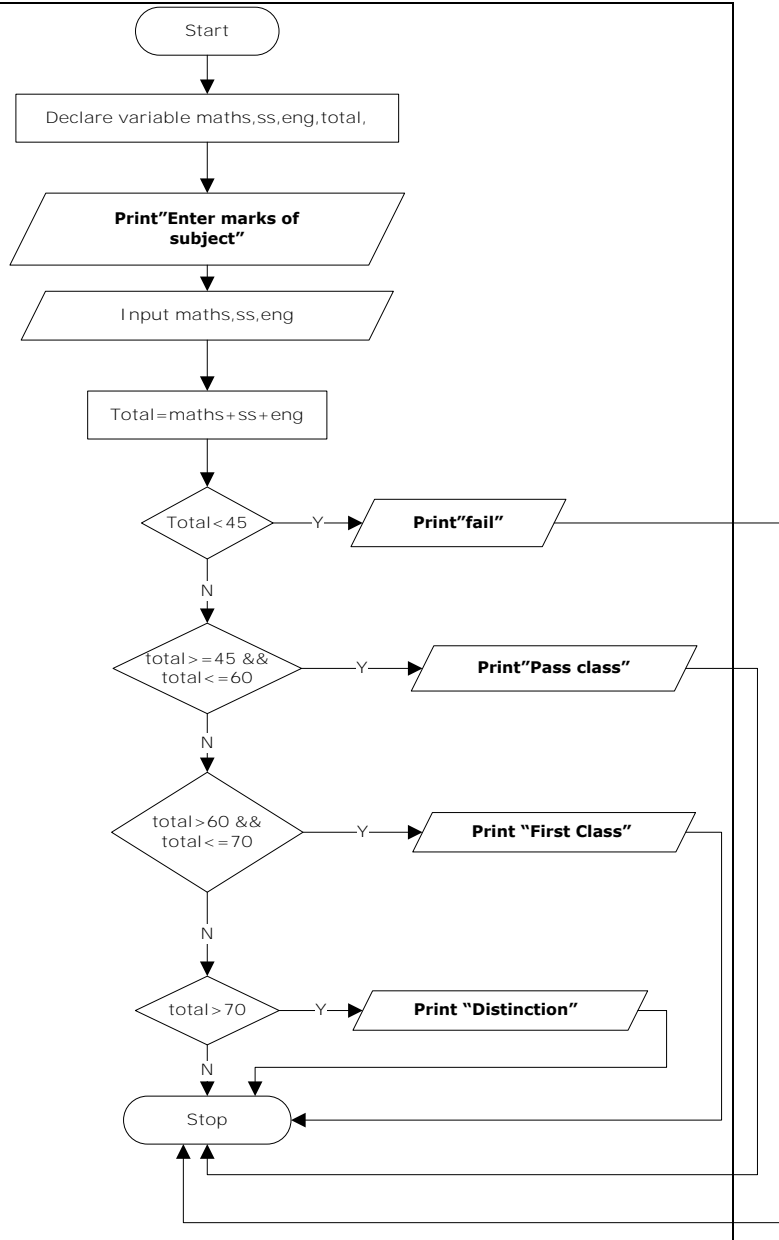
Write an algorithm and flowchart to give grade to the marks of a student (if marks between 45 to 60 pass class, if marks between 60 to 70, First class, if the marks >70: Distinction and if marks <45: fail)





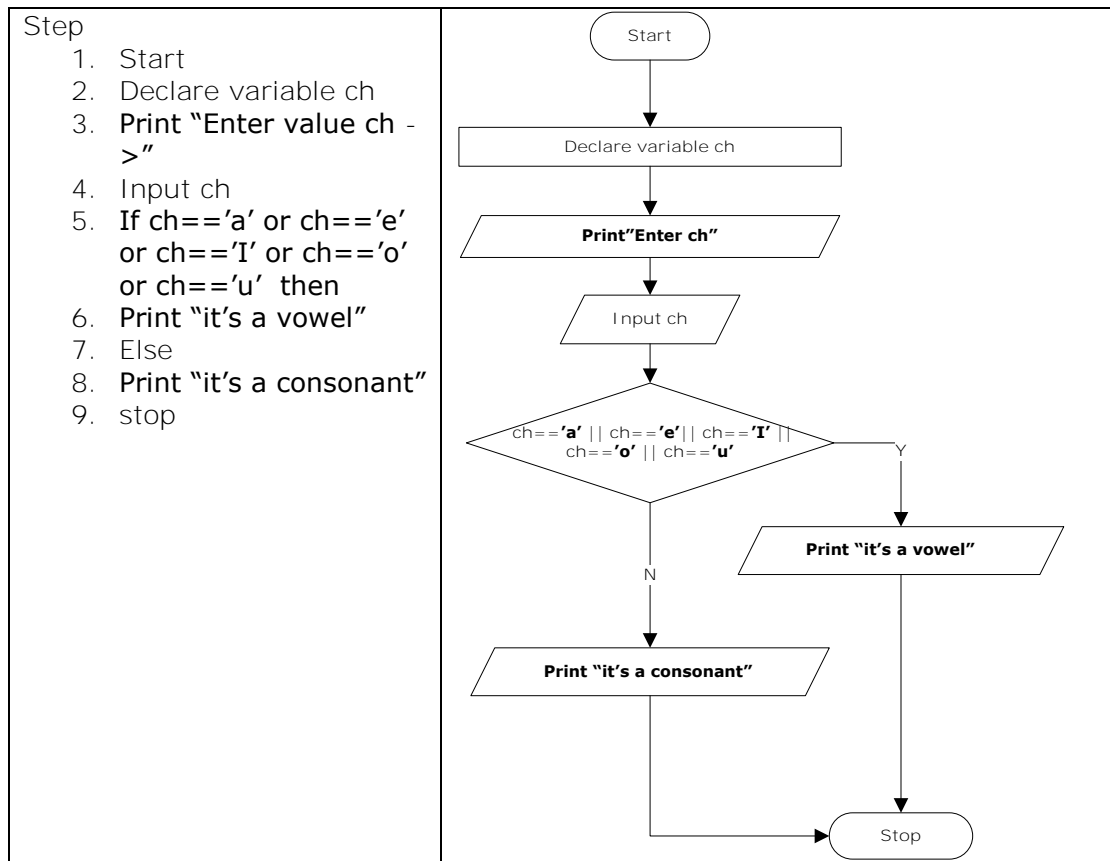
Step

1. Start
2. Declare variables
maths,ss,eng,tot
al
3. **Print "Enter value
maths,ss,eng ->"**
4. Input
maths,ss,eng
5. Total=maths+ss
+eng
6. If total<45 then
7. **Print "Fail"**
8. Else if total>=45
and total<=60
then
9. **Print "Pass class"**
10. Else if total>60
and total<=70
then
11. **Print "First class"**
12. Else if total>70
then
13. **Print "Distinction"**
14. Stop

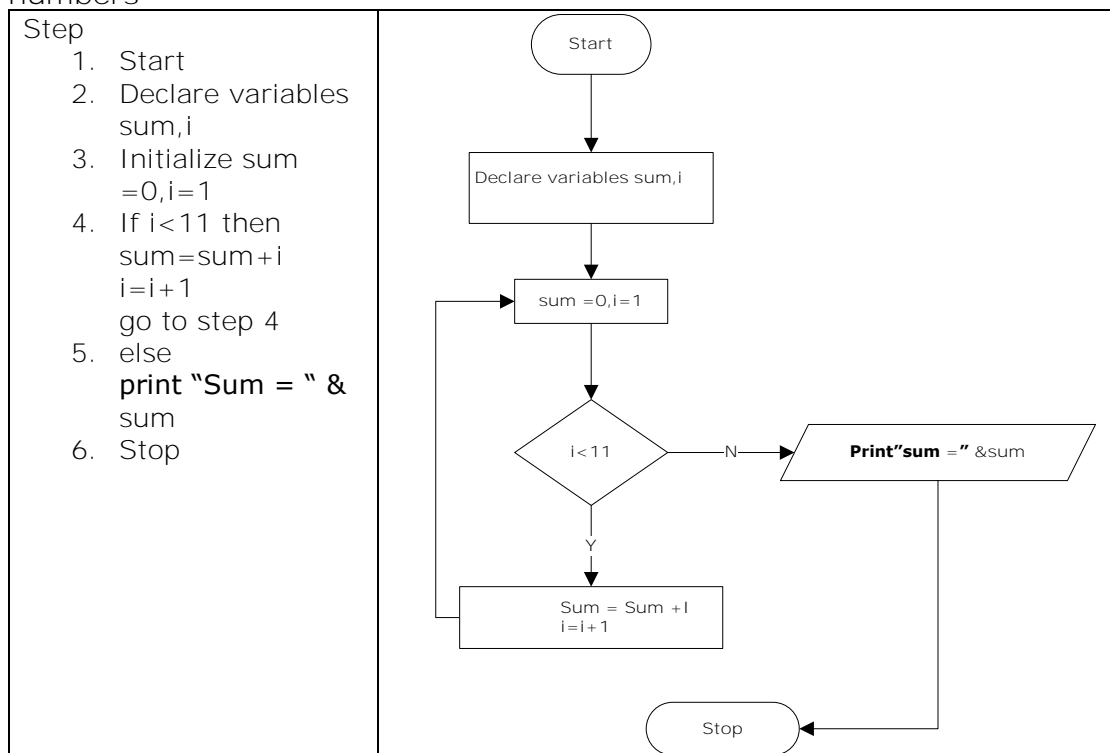


Write an algorithm and flowchart to check if the entered value is vowel or a consonant





Write an algorithm and flowchart to find the sum of first 10 successive numbers



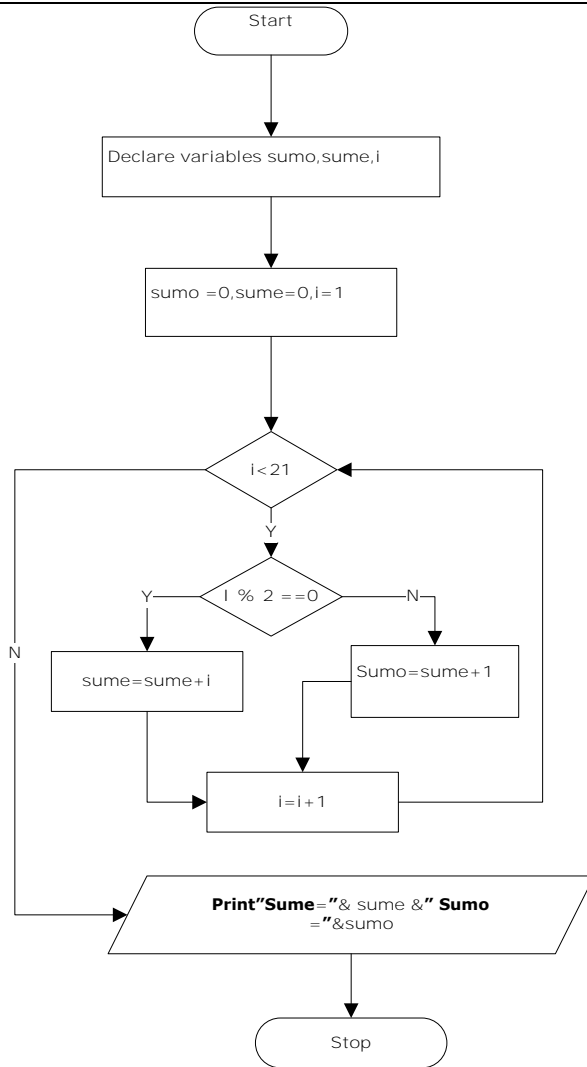
Write an algorithm and flowchart to find the sum of first 10 odd and even numbers





Step

1. Start
2. Declare variables
sumo,sume,i
3. Initialize sumo
=0,sume=0,i=1
4. If $i < 21$ then
 If $i \% 2 == 0$
 then
 sume=sume+i
 Else
 sumo=sumo+i
 i=i+1
 go to step 4
5. else
 print "Sume = " &
 sume &" Sumo = " &
 sumo
6. stop



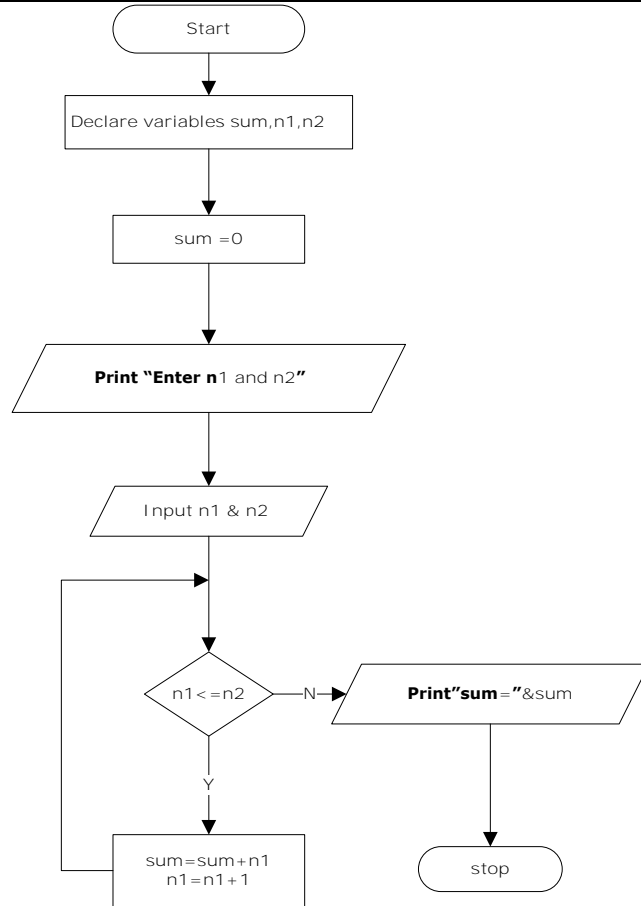
Write an algorithm and flowchart to find the sum of n1 to n2





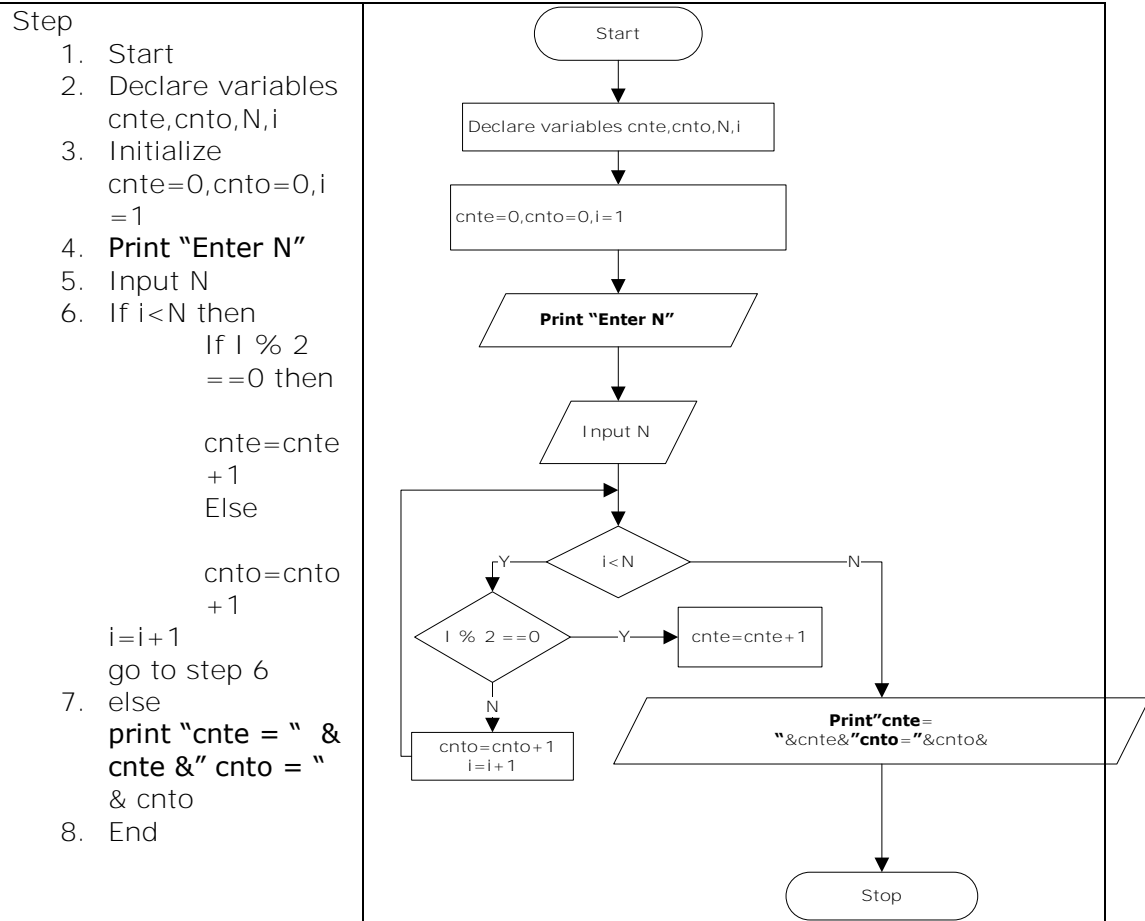
Step

1. Start
2. Declare variables
sum,n1,n2
3. Initialize sum =0
4. **Print "Enter n1 and
n2"**
5. Input n1 & n2
6. If $n1 \leq n2$ then
sum=sum+n1
n1=n1+1
go to step 6
7. else
**print "Sum = " &
sum**
8. stop



Write an algorithm and flowchart to count total number of odd and even in Range





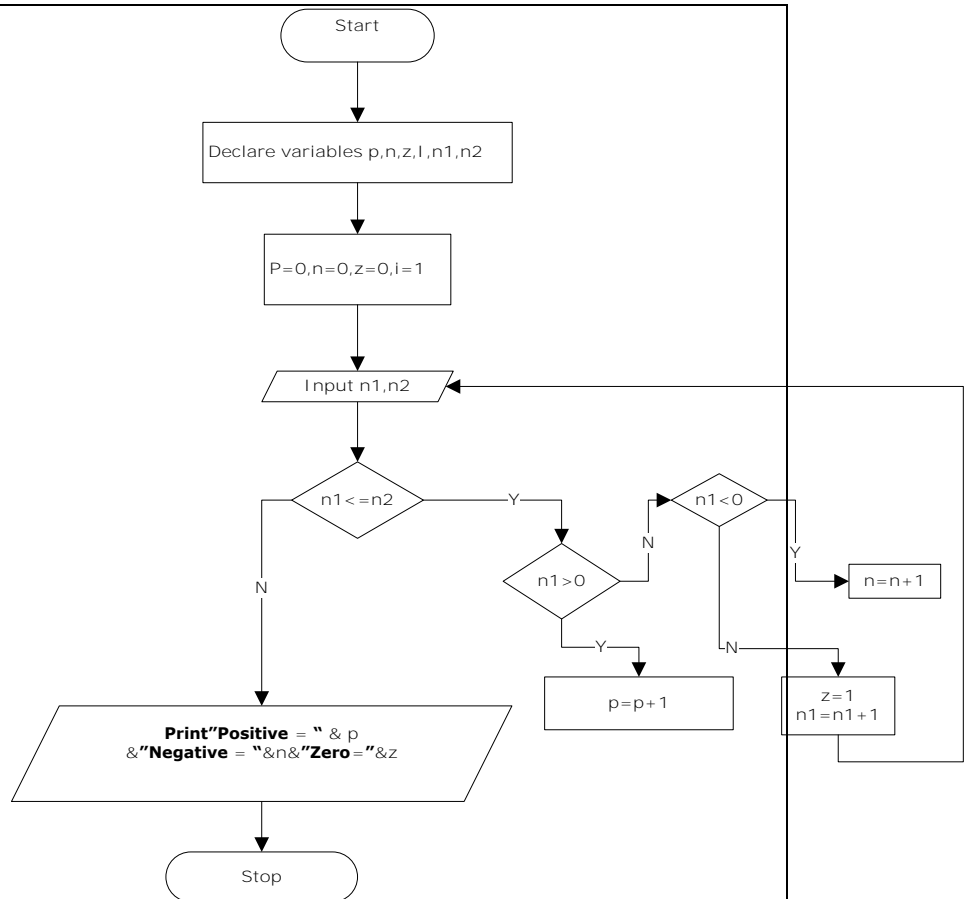
Write an algorithm and flowchart to count total number +ve,-ve and zeros from range of value





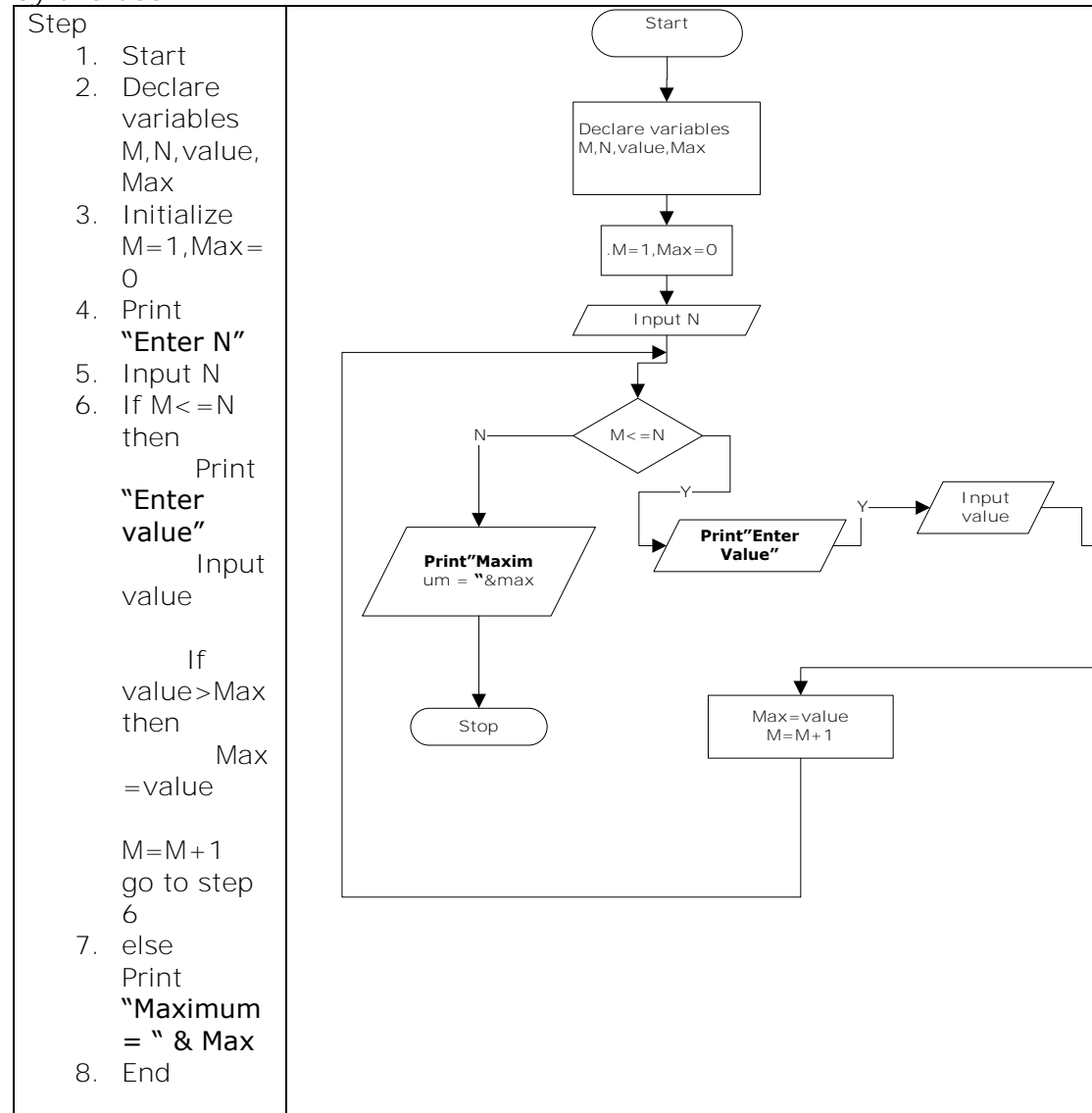
Step

1. Start
2. Declare variables
p,n,z,l,n1,n2
3. Initialize
p=0,n=0,z=0
,i=1
4. Print "Enter
n1 and n2"
5. Input n1 & n2
6. If n1 <= n2
then
 If
 n1 > 0
 then
 p=p+
 1
 else if
 n1 < 0
 then
 n=n+
 1
 else
 z=1
 n1
 =n1+
 1
 go to step 6
7. else
 print "Positive
 = " & p & "
 Negative = "
 & n & " Zero =
 " & z
8. End



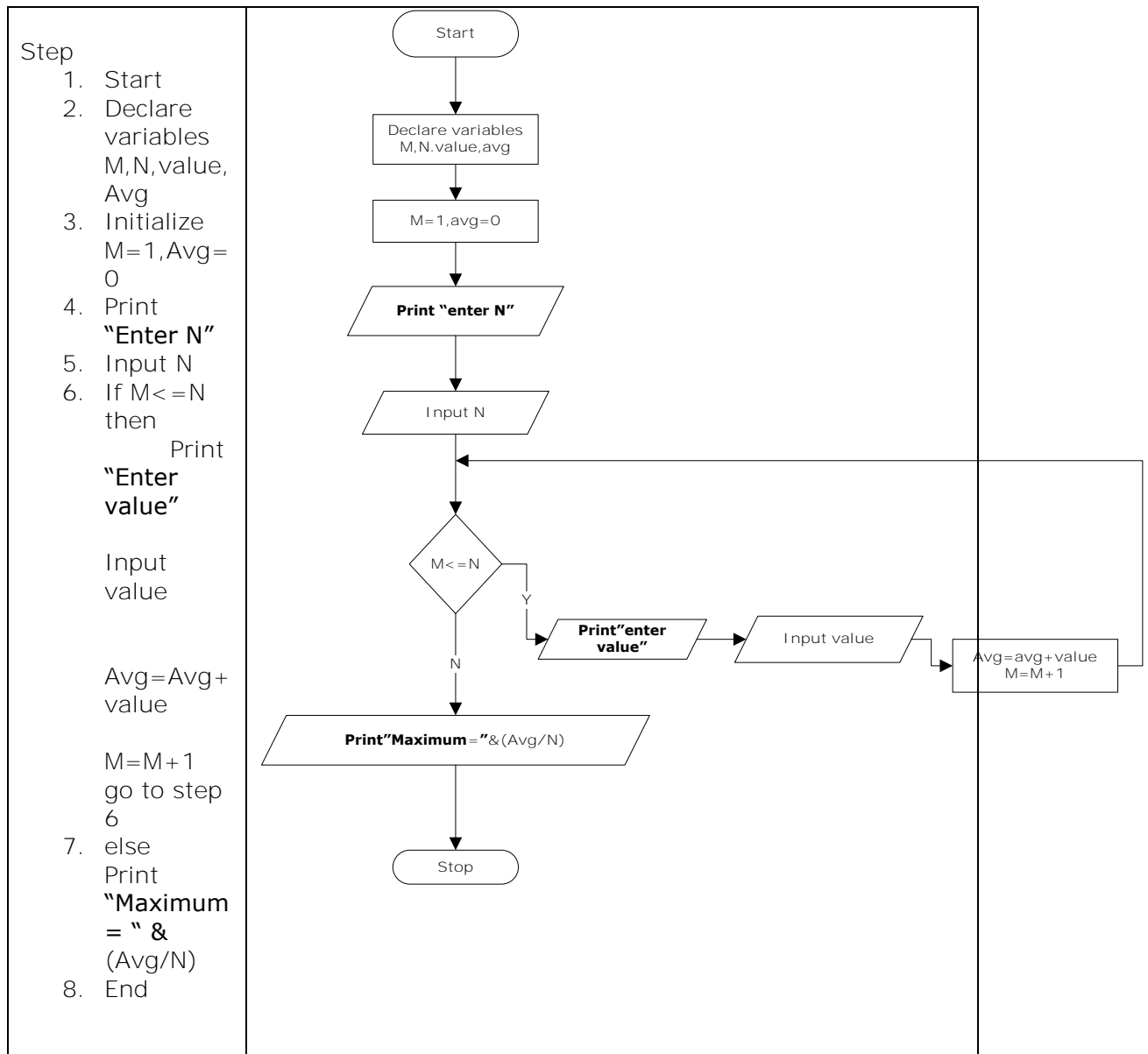


Write an algorithm and flowchart to find maximum of N numbers entered by the user



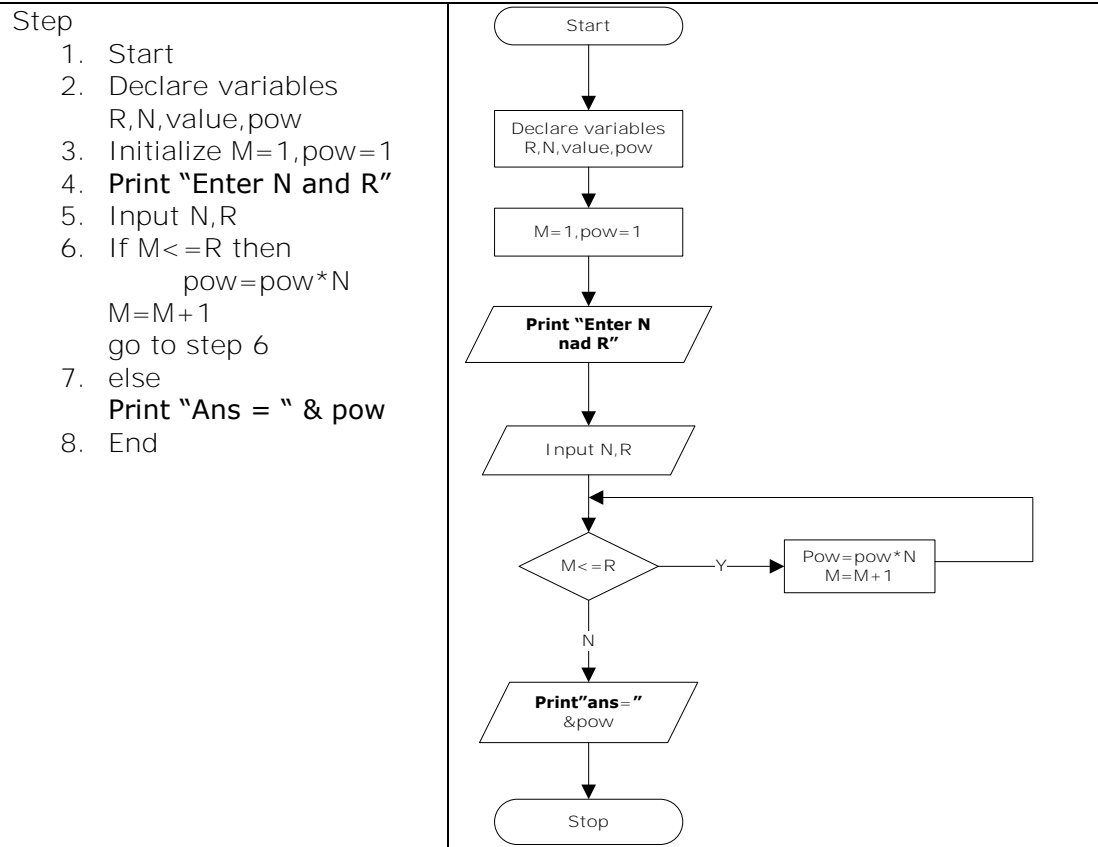
Write an algorithm and flowchart to find average of N numbers entered by the user





Write an algorithm and flowchart to find power ($2^4=16$)





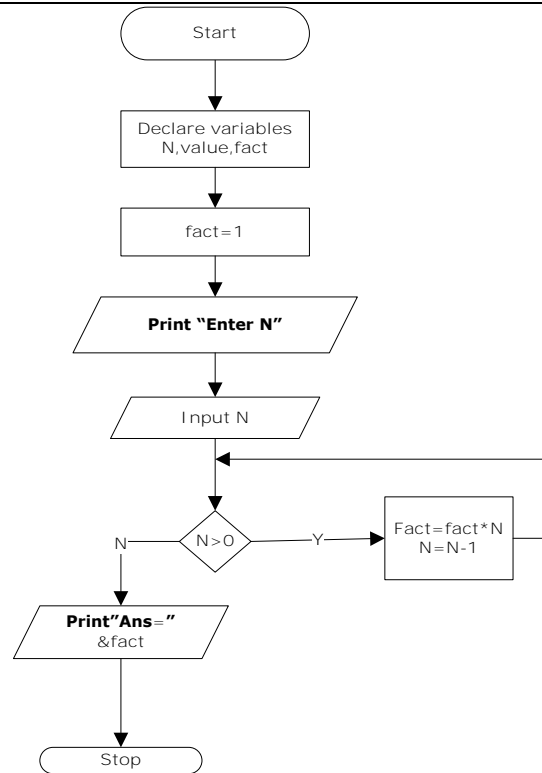
Write an algorithm and flowchart to find Factorial





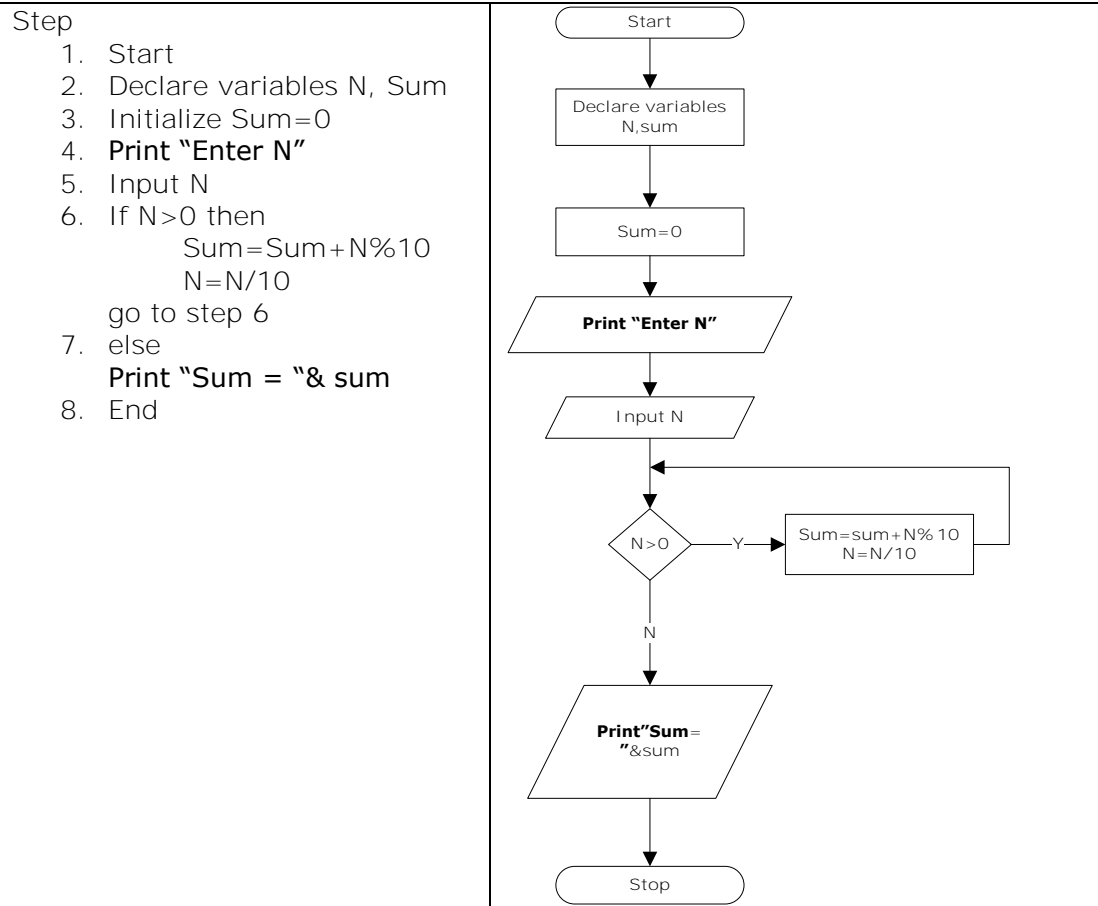
Step

1. Start
2. Declare variables N, value, fact
3. Initialize fact=1
4. **Print "Enter N"**
5. Input N
6. If $N > 0$ then
 fact=fact*N
 N=N-1
 go to step 6
7. else
 Print "Ans = " & fact
8. End



Write an algorithm and flowchart to sum of digits of an integer number. (123=1+2+3)





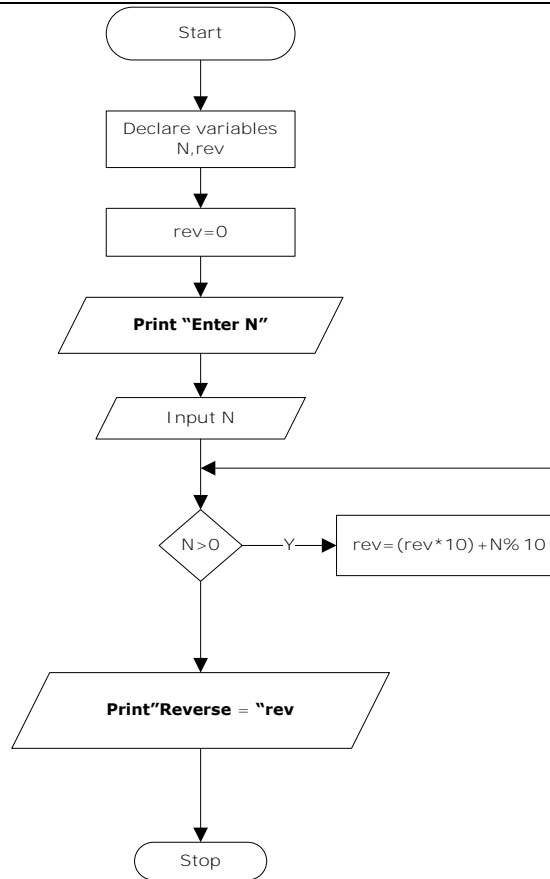
Write an algorithm and flowchart to reverse number. (123=321)





Step

1. Start
2. Declare variables N, rev
3. Initialize rev=0
4. **Print "Enter N"**
5. Input N
6. If $N > 0$ then
 $rev = (rev * 10) + N \% 10$
 $N = N / 10$
go to step 6
7. else
Print "Reverse = "& rev
8. End



Write an algorithm and flowchart to Prime Number

Step

1. Start
2. Declare variables N, flag=0, i
3. Initialize i=2
4. **Print "Enter N"**





<ol style="list-style-type: none"> 5. Input N 6. If $i < N$ then <ul style="list-style-type: none"> If $N \% i == 0$ then <ul style="list-style-type: none"> flag=1 Go to step 7 Else <ul style="list-style-type: none"> Go to step 6 7. If flag==1 then <ul style="list-style-type: none"> Print "No is Not prime " Else <ul style="list-style-type: none"> Print "No is prime" 8. End 	
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

Write an algorithm and flowchart to $1^2 + 2^2 + 3^2 + \dots + N^2$

<p>Step</p> <ol style="list-style-type: none"> 1. Start 2. Declare variables N, sum,i 3. Initialize sum=0 4. Print "Enter N" 5. Input N 6. If $i \leq N$ then <ul style="list-style-type: none"> sum=sum+ (i*i) go to step 6 7. Print sum 8. End 	<pre> graph TD Start([Start]) --> Declare[Declare variables N, sum, i] Declare --> Sum0[sum=0] Sum0 --> Input[/Input N/] Input --> Decision{i <= N} Decision -- Yes --> SumAdd[sum=sum+(i*i)] SumAdd --> Print[/Print sum/] Print --> Stop([Stop]) Decision -- No --> Decision </pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

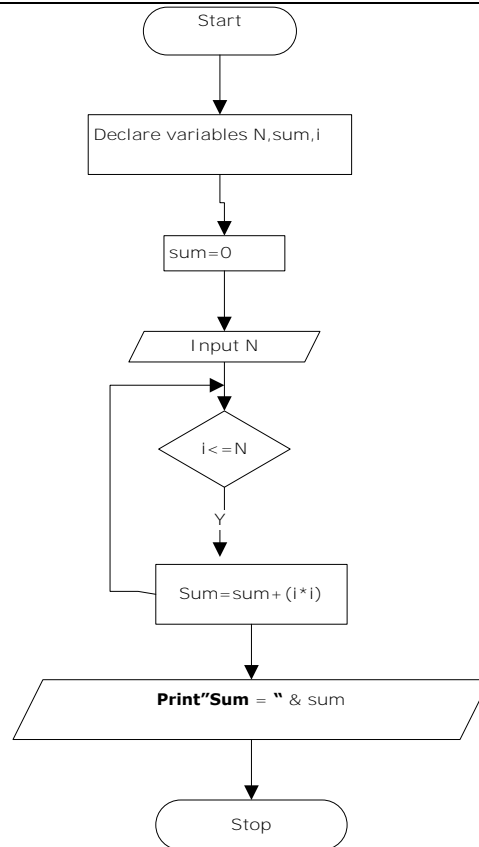
Write an **algorithm and flowchart** to $1/1 + 1/2 + 1/3 + \dots + 1/N$





Step

1. Start
2. Declare variables N, sum, i
3. Initialize sum=0
4. **Print "Enter N"**
5. Input N
6. If $i \leq N$ then
sum = sum + (i*i)
go to step 6
7. Print sum
8. End





1.4 About 'C' Programming Language

C is the most widely used computer language.

The C programming language is used to program desktop applications, compilers, tools and utilities and even hardware devices.

The C programming language follows a procedure oriented paradigm. So, In C language, a large program is divided into small programs called functions. Prime focus is on functions and procedures that operate on the data.

C Program structure follows "Top Down Approach".

Hey guys the founder of C Language is Dennis Ritchie. He developed C Language in Bell Laboratories in 1971.

Dennis Ritchie is best known as the creator of the C programming language and also a key developer of the Unix operating system.



1.4.1 Why C Language is Popular? Or Say Characteristics of C Language or Say Advantages of C Language

Keywords

- There are only 32 keywords in ANSI C and its strength lies in its built-in functions.

Libraries

- C Compiler comes with list of header files which consist of many built in functions which can be used to develop program.

Modularity

- It is one of the important characteristics of C. we can split the C program into no. of modules instead of repeating the same logic statements. It allows reusability of modules.

Middle level language

- It allows the programmer to perform operations from high level language to low level of assembly language. Therefore, it is also known as middle level language.
- As a middle level language C combines both the advantages of low level and high-level languages.





Operating Systems are written in C language

- Major operating systems (OS) like Windows, Linux and Unix are all written in C language. So, if we want to modify programs in Windows or Linux, basic understanding of C is inevitable.

Portability

- We can compile or execute C program in any operating system(unix,dos,windows). This means that C programs written for one computer can easily run on another computer without any change or by doing a little change.

Powerful programming language

- C is very efficient and powerful programming language, it is best used for data structures and designing system software.

Case Sensitive language.

- C is case sensitive language. It is essential to write a code according the case sensitively.

Procedure Oriented Language

- C Language is procedure oriented language, means in the C language user creates procedures or functions to execute their specific task. Usually C programs are divided into small functions. Procedure oriented language is follows algorithm to execute your statements.

Easy to learn

- C programming language is easy to Learn. C language syntax is very easy to understand

Speed

- It is Close to hardware make many 'C' programs run at speed close to their assembly language.

1.4.2 Applications of C Programming

Operating Systems	Network Drivers	Print Spoolers
Language Compilers	Assemblers	Text Editors
Modern Programs	Data Bases	Language Interpreters
Simulators	Utilities	Embedded System





1.4.3 What is ANSI?

The **American National Standards Institute** (ANSI) has developed a standard and formed a committee for the C language in 1983.

The ANSI established a committee to provide a modern, comprehensive definition of C. The resulting definition, the **ANSI standard**, or "ANSI C", was completed late 1988.



Today C is in widespread use with a rich standard library of functions.

So, we are going to learn ANSI C Language rather not C Language.

1.5 Creating the Source file, Compiling and Linking

C programs are compiling and run by many C compilers commercially available in the market such as Borland turbo C++, Intel C/C++ , Microsoft Visual C++, GCC/G++ for Linux, etc.

In this book, all the examples are tested using DOS based Borland turbo C++ Version 3.0 compiler.

Following are the steps that we have to follow practically.

Step 1: Open a turbo C++ editor or any editor and write a program.

```
DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
File Edit Search Run Compile Debug Project Options Window Help
FIRST.C
#include<stdio.h>
#include<conio.h>
main()
{
clrscr();
printf("Welcome to the world of C programming");
printf("\nDon't fear I am here !!!!");
getch();
}
```

9:65

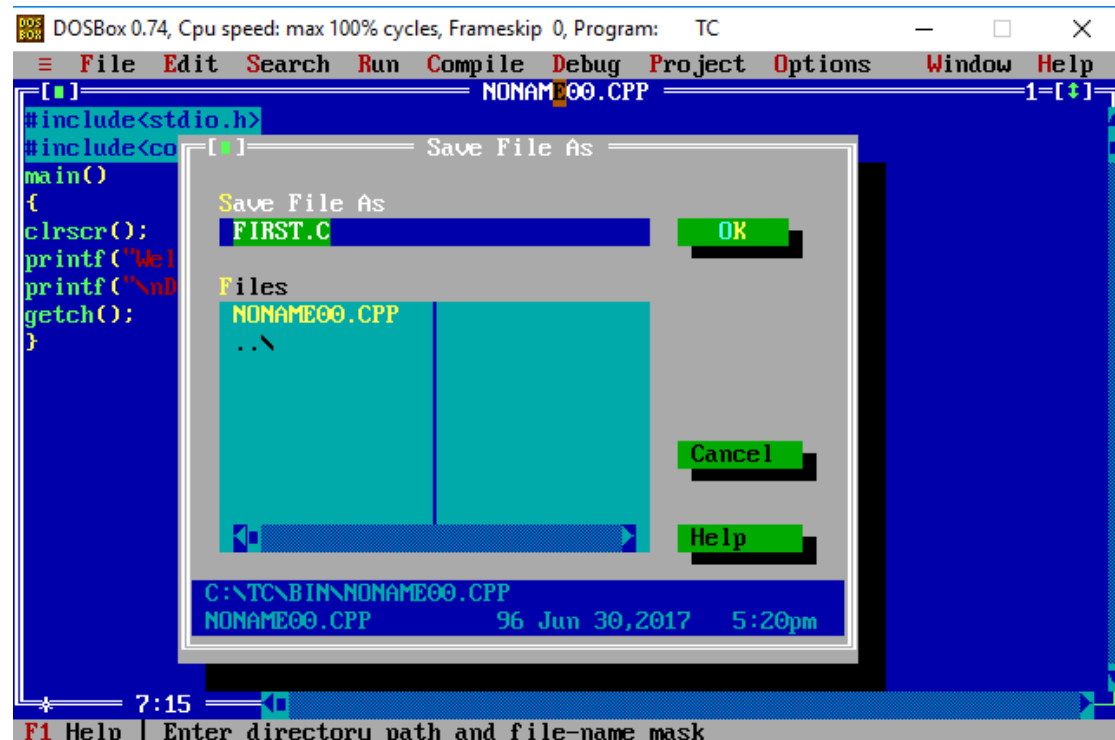
F1 Help Alt-F8 Next Msr Alt-F7 Prev Msr Alt-F9 Compile F9 Make F10 Menu





Step 2: Save the program and give the file name with extension.

For C program, the extension name must be .C. Filename can be any name but should not be more than 8 characters and does not contain space or any symbols.



Step 3: Compiling source file

Compile the program. If you are using turbo C then from the compile menu and select compile option. Compiler perform two operations, first find out errors in your program, if no errors then it perform second operation, converts program into binary file also known as object file(.obj) which is understood by the computer only.

If there are any errors during compilation process, then debug them and compile the program again. During compilation process, compiler make an object file which is specific to the operating system (on which we have compiled the program) and hardware dependent, this means compiled code may not work on another operating system.





```
#include<stdio.h>
#include<conio.h>
main()
{
clrscr();
printf("Welcome to the world of C programming\n");
printf("Don't fear I am here !!!!!\n");
getch();
}
```

Compiling

Main file: FIRST.C
Compiling: EDITOR → FIRST.C

	Total	File
Lines compiled:	463	463
Warnings:	1	1
Errors:	0	0

Available memory: 1969K
Warnings : 1

Step 4: Linking and executing

Link the object file (obj) generated by the compiler with other library files(lib) to make a executable file(exe).This process is done by selecting link option from the compile menu. If any errors occur during linking process, correct them and compile the program again otherwise we can run the program by selecting run option from the run menu in turbo C.

```
Welcome to the world of C programming
Don't fear I am here !!!!!
```

1.6 Basic structure of C program

All C programs should contain three elements

The preprocessor

These are the various Helpful files which gets executed on compiling. These Preprocessor Files must be included in any program to run it smoothly, creating no any errors. These commands tell the compiler to do preprocessing before doing actual compilation. These are readymade standard library files with C provides the facilities that are not defined within C, which we will se later on... Soo Cheers!!!





Comments

Comments are used for commenting on Source code which will not get compiled but it will be very helpful in finding errors, functioning of Various Commands.

Comments are non-executable codes as already said earlier.

Comments are generally categorized in two ways:

(i) They can be written in a Single line as:

//..... called as Single-Line comments.

(ii) Comments can also be written in symbols called Multi-Line Comment

/*.....*/ between the symbols, write the comments.

Functions

Various operational areas to be performed using Functions.

A Function is a name with a pair of circular bracket after it containing information for various tasks.

These are of two types

1. User-Defined Functions
2. Derived Functions

There is only one compulsory function called - the *main* function. We can also write our own functions.

Example#

//First Program to Print HELLO WORLD on Output screen.

Source Code or Input:

```
#include<stdio.h>
#include<conio.h>

/*Here, our all main contain to be executed, starts with a main() function which is compulsory in every program*/

void main()
{
    clrscr();

    printf("HELLO WORLD...!!");

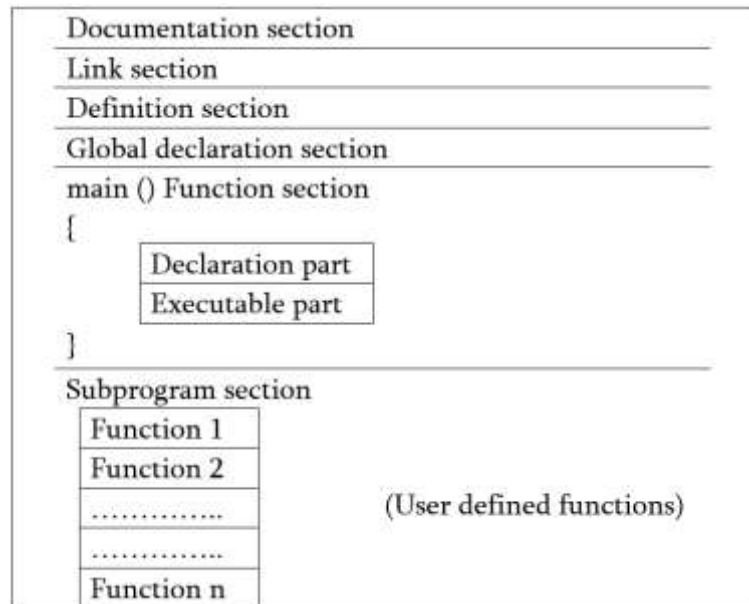
    getch();
}
```

Output:
Hello World





The Basic structure of a 'C' Program...



1. Documentation section

The documentation section consists of a set of comment lines giving the name of the program, the author and other details, which the programmer would like to use later.

2. Link section

The link section provides instructions to the compiler to link functions from the system library such as using the *#include* directive.

3. Definition section

The definition section defines all symbolic constants such using the *#define* directive.

4. Global declaration section

There are some variables that are used in more than one function. Such variables are called global variables and are declared in the global declaration section that is outside of all the functions. This section also declares all the user-defined functions.

5. main () function section

Every C program must have one main function section. This section contains two parts; declaration part and executable part

1. Declaration part

The declaration part declares all the variables used in the executable part.





2. Executable part

There is at least one statement in the executable part. These two parts must appear between the opening and closing braces. The programming execution begins at the opening brace and ends at the closing brace. The closing brace of the main function is the logical end of the program. All statements in the declaration and executable part end with a semicolon.

6. Subprogram section

If the program is a multi-function program then the subprogram section contains all the user-defined functions that are called in the main () function. User-defined functions are generally placed immediately after the main () function, although they may appear in any order.

All section, except the main () function section may be absent when they are not required.





1.7 Executing C Program

Step by Step Execution of a C program:

Step I: Edit

1. This is First Step i.e. Creating and Editing Program.
2. First Write C Program using Text Editor , such as [Intel C/C++ , Microsoft Visual C++ , GCC/G++ for Linux, Borland Turbo C/C++ 3.0 , Notepad++ , Notepad, etc.]
3. Save Program by using [.C] Extension.
4. **File Saved with [.C] extension is called "Source Program".**

Step II: Compiling

1. Compiling C Program: C Source code with [.C] Extension is given as input to compiler and compiler convert it into Equivalent Machine Instruction.
2. In Borland C/C++ 3.0 program can be compiled using key [Alt + F9].
3. Compiler Checks for errors. If source code is error-free then Code is converted into Object File [.Obj].

Step III: Checking Errors

1. During Compilation Compiler will check for error, If compiler finds any error then it will report it.
2. User have to re-edit the program.
3. After re-editing program , Compiler again check for any error.
4. If program is error-free then program is linked with appropriate libraries.

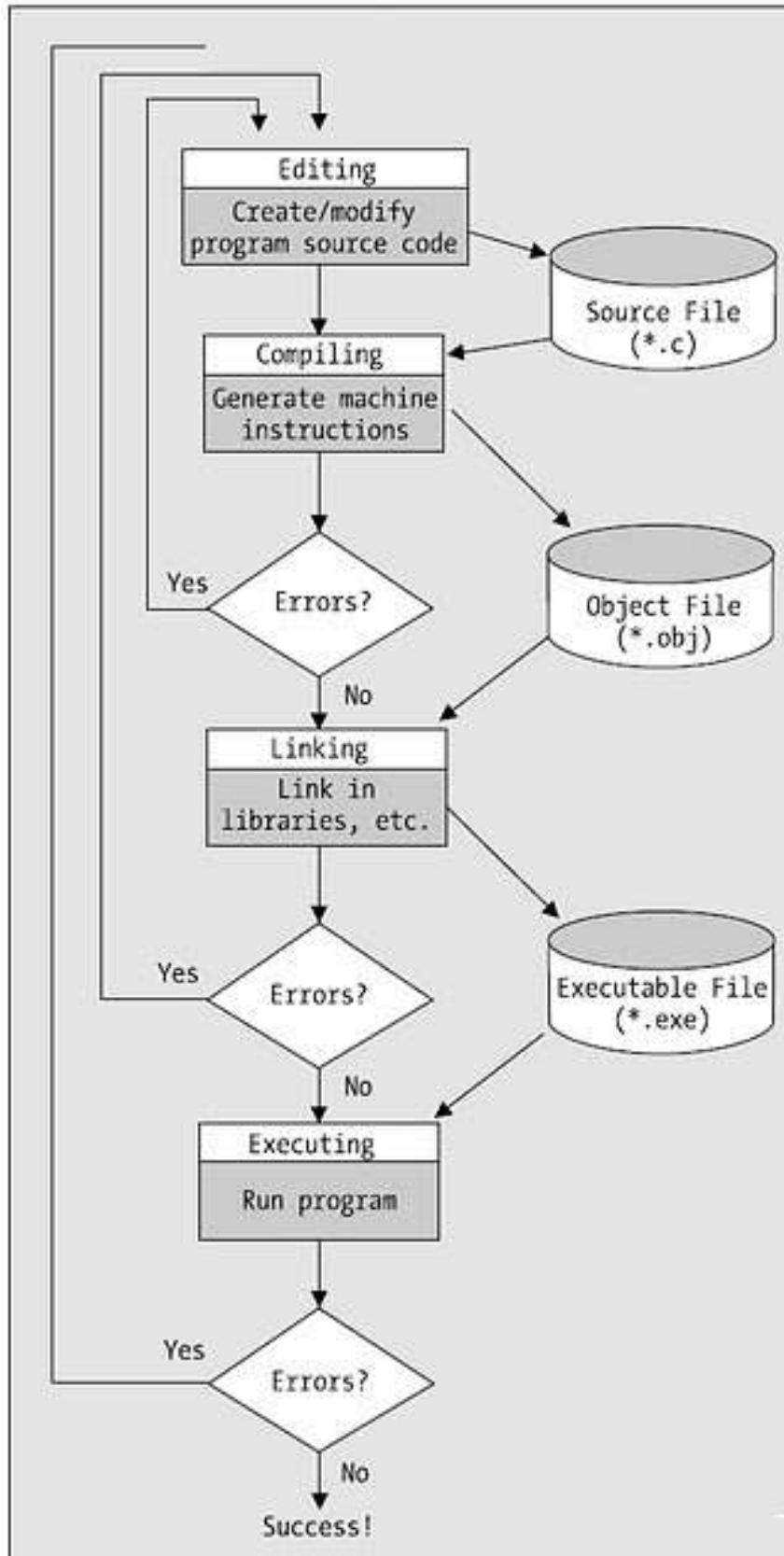
Step IV: Linking Libraries

1. Program is linked with included header files.
2. Program is linked with other libraries.
3. This process is executed by Linker.

Step V: Error Checking

1. If run time error occurs **then "Run-time" errors are reported to user.**
2. Again, programmers have to review code and check for the solution.







1.8 Character sets

As English language has characters, symbols, numbers similarly C language has set of characters which programmers used in order to write a program.

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

C Language has its own character set as given below. The only characters required by the C Programming Language are as follows:



1.8.1 Letters

LowerCase a-z

UpperCase A-Z

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

1.8.2 Digits

0-9

0 1 2 3 4
5 6 7 8 9





1.8.3 Special Characters

Symbol	Meaning
&	Ampersand
;	Semicolon
,	Comma
:	Colon
"	Quotation mark
/	Slash
.	Dot
'	Apostrophe
!	Exclamation mark
?	Question mark
%	Percentage
#	Hash
=	Equal to

Symbol	Meaning
^	Caret
-	Minus
+	Plus
/	Slash
<	Less than
>	Greater than
{ }	Braces left/right
\	Back slash
~	tide
\$	Dollar
()	Parenthesis Left/right
@	At the rate

1.8.4 White Spaces

These characters are also known as back slash or escape characters. Some white space characters are non-printable. They are new line, form feed, blank space, horizontal tab, carriage return.

These characters are used to print some character or to format the output of the program. In C, these white space characters are also identifying as character constant.

They start with back slash (\) and then following is the white space character.

White space character	Meaning
'\f'	Form feed
'\n'	End of line
'\t'	Horizontal tab
'\v'	Vertical tab
'\r'	Carriage return
'\b'	Back space
'\a'	Alert
'\"'	Double quote
'\\'	Back space
'\0'	Null character/end of string





1.9 Tokens

In a C program, smallest individual part or unit is known as token. It is recognized by the compiler. We write a program using tokens.

C tokens are the basic buildings blocks in C language which are constructed together to write a C program.

Token	Example
Keywords	int, for, if
Identifiers	main, result, add
Constants	11, 3.14
Strings	"total", "hello"
Special symbols	(), {}, "", ;
Operators	+, /, -, *

Example# Identifiers

```
void main()
{
    int a,b,sum;
    a = 10;
    b = 20;
    sum=a+b;
    printf("\nSum = %d", sum);
    getch();
}
```

main – identifier
{, }, (,) – delimiter
+ - operator
10 20 - constant
int – keyword
a,b,sum – identifier





1.10 Keywords and Identifiers

1.10.1 Keywords



A word in C is named as keyword or identifier, consisting the set of rules for Case-sensitivity, reserved word that are mainly used in writing a C program.

Keywords are basically the specific words of C language. In C, they have special meaning. Keywords are also called reserved words. These keywords are known to the compiler. We cannot change the meaning of the keywords.

32

Since keywords are referred names for compiler, they can't be used as variable name.

There are 32 keywords in C Language.

Following are some examples of C keywords:

auto	double	int	struct	const	float
short	unsigned	break	else	do	union
long	switch	if	continue	forsigned	void
case	enum	return	register	typedef	While
default	goto	Sizeof	volatile	char	static
extern					

1.10.2 Identifiers

Identifiers are used to give name to the variables, functions, arrays, structure, and union. This name is given by the programmer. Name of the identifier should be meaningful.





Following are the rules when declaring identifiers.

- ✓ Identifier contains letters (a-z), digits (0-9) and underscores characters (_) as a character sets. Reserved words are not allowed as a identifier.
- ✓ First character must be a letter (A-Z or a-z) or underscore (_).
- ✓ Blank space or white space character is not allowed.
- ✓ Then length of the identifier should not be more than 31 characters
- ✓ C language is case sensitive. Thus, names TOTAL and total are considered as different identifiers.



Example#

Valid		Invalid	
_abc		1hour	
A1, a1	✓	-sn	✗
Abc_12		if	

1.11 Constants

Constants refer to fixed values that the program may not alter during its execution. These fixed values are also called boot literals or just literals.



Example# 1264, 3, 2, π, e, etc..

- ✓ Constants can be of any of the basic data types like
- ✓ an Integer constant (value in numbers without any decimal)
- ✓ a Floating constant (value in numbers with decimal points)
- ✓ a Character constant (a single letter value)
- ✓ a String literal (containing a line or paragraph)

There are enumeration constants as well. Constants are treated just like regular variables except that their values cannot be modified after their definition.

1.11.1 Integer Literals

- ✓ An integer literal can be a decimal, octal, or hexadecimal constant. A prefix specifies the base or radix: 0x or 0X for hexadecimal, 0 for octal, and nothing for decimal. Here are some examples of integer literals –

```
212      /* Legal */
215u     /* Legal */
0xFeeL   /* Legal */
078      /* Illegal: 8 is not an octal digit */
032UU    /* Illegal: cannot repeat a suffix */
```





Dr. Shyam N. Chawda, C Language Tutorial , 78 74 39 11 91

- ✓ An integer literal can also have a suffix that is a combination of U and L, for unsigned and long, respectively. The suffix can be uppercase or lowercase and can be in any order. Following are other examples of various types of integer literals –

```
85      /* decimal */
0213    /* octal */
0x4b    /* hexadecimal */
30      /* int */
30u     /* unsigned int */
30l     /* long */
30ul    /* unsigned long */
```

1.11.2 Floating-point Literals

- ✓ A floating-point literal has an integer part, a decimal point, a fractional part, and an exponent part. You can represent floating point literals either in decimal form or exponential form.
- ✓ While representing decimal form, you must include the decimal point, the exponent, or both; and while representing exponential form, you must include the integer part, the fractional part, or both. The signed exponent is introduced by e or E.

Here are some examples of floating-point literals

```
3.14159 /* Legal */
314159E-5L /* Legal */
510E     /* Illegal: incomplete exponent */
210f     /* Illegal: no decimal or exponent */
.e55     /* Illegal: missing integer or fraction */
```

1.11.3 Character Constants

- ✓ Character literals are enclosed in single quotes, e.g., 'x' can be stored in a simple variable of char type.
- ✓ A character literal can be a plain character (e.g., 'x'), an escape sequence (e.g., '\t'), or a universal character (e.g., '\u02C0').
- ✓ There are certain characters in C that represent special meaning when preceded by a backslash for example, newline (\n) or tab (\t).

1.11.4 String Literals

- ✓ String literals or constants are enclosed in double quotes "". A string contains characters that are similar to character literals: plain characters, escape sequences, and universal characters.
- ✓ You can break a long line into multiple lines using string literals and separating them using white spaces.





- ✓ Here are some examples of string literals. All the three forms are identical strings.

1.12 Variables

In C programming language, a variable is a value that can change, depending on conditions or on information passed to the program.

Variables are used to store a data. A variable is a container (storage area) to hold data.

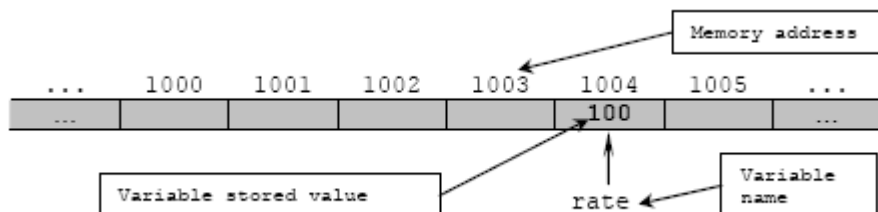
We can think of a variable like a bucket and values are like a water in bucket which can easily change.



Variable is an entity where user can store digits, characters and strings. It is similar to a box or a container.

Constant store a value that cannot be change but storing a value in a variable value can be change.

1.12.1 How variable stores the value?



Data are normally stored in a computer's memory which is organized as cells.

These cells are called the location or address that store the data. Variables are used to give a name to these locations.

1.12.2 Variable names

A name given to the memory location is known as variable name. There are rules that must be followed before giving a variable name in C. They are as under.

- In C, it must start with an alphabet or underscore (_) and following characters can be letter, underscores and digits only.





- Both uppercase and lowercase letters can be used.
- Spaces are not allowed in variable name.
- Certain keywords (these are C language reserved words) cannot be used in a variable name.
- The variable names should not be very long. (maximum 32 characters)
- The variable names should be meaningful.



Example#

Valid A, a1, a_ ,_ a
Invalid If, 1a, a#, a bc

1.12.3 Declaration of the variables

Syntax#

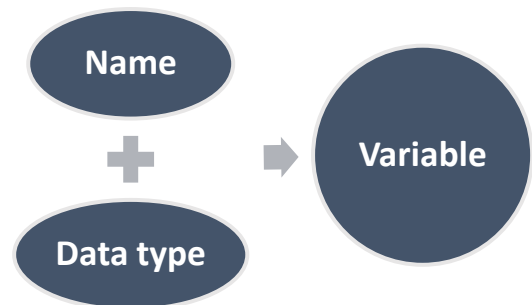
Data-type variableName1, variableName**2** ... variableNameN;

Example#

int sum, rollno;

float average;

char gender, name[20];





1.12.4 Data types in C



For example, in Our secular nation we can generally identify a person simply looking at their costumes, their identity belonging, their language, their religion, their daily eating etc. Similarly, in C language we can get idea about variable category from its data type.

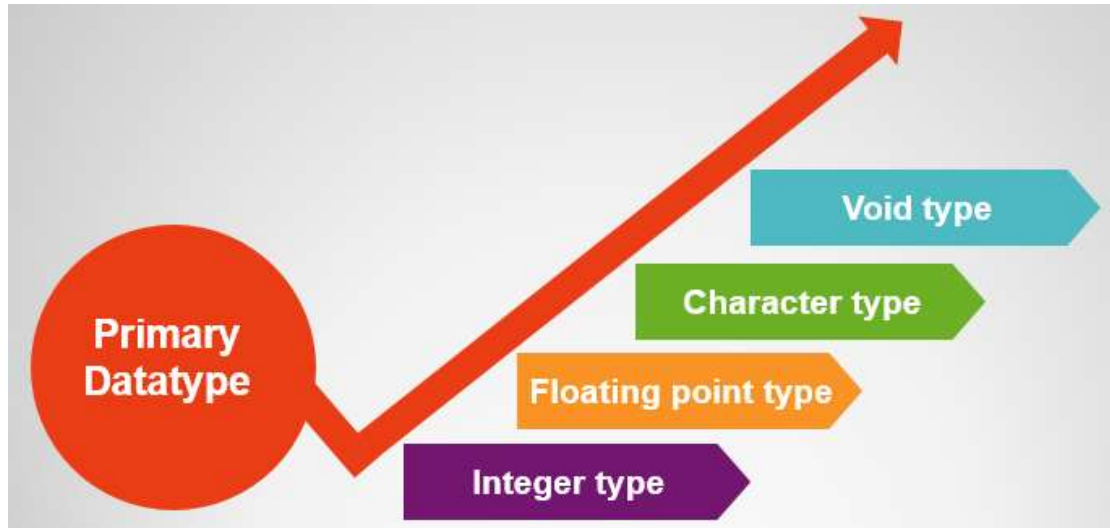
Data types represent the type of data to be stored in a variable. We store a data in computer memory through the variables. So variables type must be specified before Storing data into the computer memory.

Computer memory is organized in bytes, and byte stores the smallest amount of data. C language supports different types of data (integer, float, character etc.) and need to store the values in the program. In C, there is fixed amount of memory assigned to each data type.

Data types can be broadly classified as:



1.12.4.1 Primary data types



1.12.4.1.1 Integer types

- Integers are whole numbers with a range of values.
- Generally, an integer occupies 2 bytes memory and its value range limited to -32768 to +32767.

Example#
int num1,num2;
long int num3;

Without point, Whole Values

There are various integer data types each has different memory assignment. This gives you the flexibility of choosing the right type of variable in the program.

Data Type	No of Bytes Used in memory	Default Range (minimum to maximum value)	Control string
Int	2	-32768 to 32767	%d
long int	4	-2147483648 to 2147483647	%ld
Unsigned long int	4	0 to 4294967295	%lu
Unsigned short int	1	0 to 255	%hd
short int	1	-128 to 127	%hd
unsigned int	2	0 to 65535	%u

- Meaning of "unsigned" is that only positive number can be stored in the variable.
- Here the "signed" keyword is not compulsory if the you have declared a variable without it compiler will assume as "signed" by default another wise write "unsigned" keyword explicitly.
- Control string contains % character (known as conversion character) followed by format specifier character. Control string is used to specify what types of data you are going to read (input) or print (output) through input or output statements in C.

1.12.4.1.2 Floating Point Types



Dr. Shyam N. Chawda, C Language Tutorial , 78 74 39 11 91

The float data type is used to store fractional numbers (real numbers) with up to 6 digits of precision.

When the accuracy of the floating-point number is insufficient, we can use the double type instead of float.

The double is same as float but provide more memory space for storing large value with longer precision.

Example#

```
float percentage;  
double meterreading;
```

With point , Decimal Values

There are various float data types each has different memory assignment. this gives you the flexibility of choosing the right type of variable in the program.

Data Type	No of Bytes Used in memory	Default Range (minimum to maximum value)	Control string
Float	4	-3.4e38 to +3.4e38	%f or %e (with exponent) or %g (trailing zero or decimal point will not appears)
Double	8	-1.7e308 to +1.7e308	%lf

- Control string contains % character (known as conversion character) followed by format specifier character. This control string is used to specify what types of data you are going to read (input) or print (output).

1.12.4.1.3 Character Type

Character type variable can hold a single character. There are signed and unsigned chars; both occupy 1 byte each, but having different ranges.

Example#

```
char ch = 'a';
```

Data Type	No of Bytes Used in memory	Default Range (minimum to maximum value)	Control string
Char	1	-128 to 127 (in this range ASCII value of the character is stored.	%c
Unsigned char	1	0 to 255	%c
Char	Equal to length of character	For each character -128 to 127 (in this range ASCII value of the character is stored.	%s

- Control string contains % character (known as conversion character) followed by format specified character. This control string is used to specify what types of data you are going to read (input) or print (output).





1.12.4.1.4 void Type

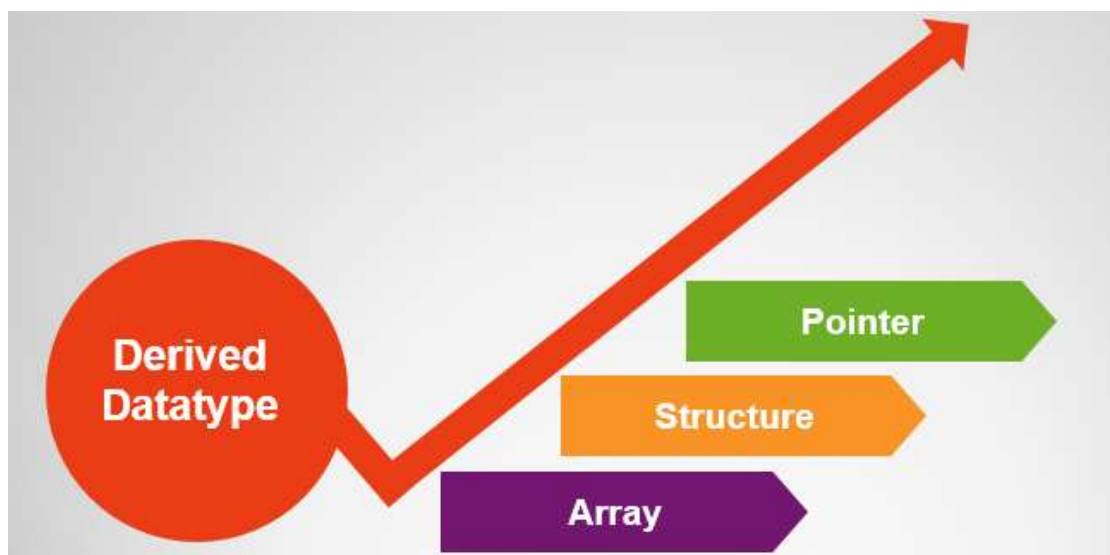
This data type is used return type of the function and it means function does not return any value.

Another use with the as an argument of the function and meaning is that function does not has any arguments.

Example#

void read(); → read() function does not return nay value.
void read(void); → read() function does not have any arguments.

1.12.4.2 Derived (structured) data types



1.12.4.2.1 Arrays

- Instead of declaring individual variables, such as a,b,c,d,... ,and f, we declare one array variable such as numbers and use ar[0], ar[1], and ..., ar[99] to represent individual variables.
 - Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type.
 - Arrays can be any of the C data-types int, float, and char. An integer array can only hold integer values and cannot hold values other than integer similarly for others data types.
 - All arrays consist of contiguous memory locations.
- 📄 Like an Excel spreadsheet, arrays have a position number for each row.





- Each object of the array can be accessed by using its positions number - index. The positions-index in an array start at 0 and go up sequentially. Each position in the array can hold a value. When we declare an array, we need to set its size.

	A	B	C
1	Position	Value	
2	0	11	
3	1	22	
4	2	33	
5	3	44	
6	4	55	
7			

- For example, we want to declare 10 variables of same int data type, instead of declaring individual variables, such as no0, no1, ..., and no9, we declare one array variable such as numbers and use no[0], no[1], and ..., no[9] to represent individual variables.

Types of Arrays in C

There are two types array.

One dimensional

Array having only one subscript variable is called.

Syntax#
type arrayName [arraySize];

Example#
int a[10];

Multi-dimensional

Array having more than one subscript variable is called Multi-Dimensional array.

Multi Dimensional Array is also called as Matrix.

Syntax#
type arrayName [arraySize] [arraySize];

Example#
int matrix[3][3];

1.12.4.2.2 Structures

Structures are used to represent a record. Suppose we want to keep track of students in a college. We might want to track the following attributes about each Student – Rollno , Name , Address ,ContactNo ,Div ,Semester

C structure is nothing but collection of different related data types.

Syntax#
struct structure_name
{
 data type member1;
 data type member2;
 ...
};

Example#
struct stu
{
 int sno;
 char sname[20];
 int sem;
};





1.12.4.2.3 Pointers

In C language, a pointer is a variable that points to a memory location in which data is stored.

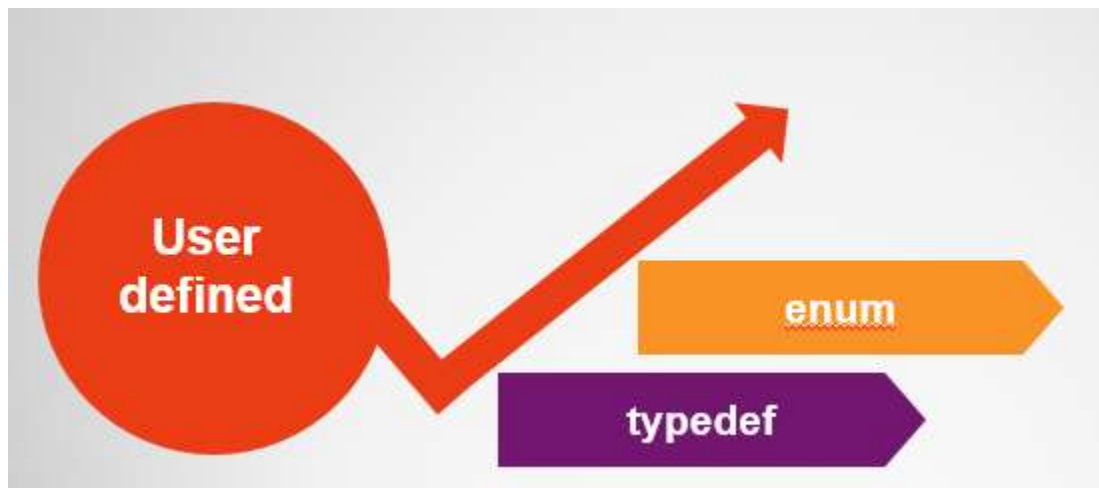
By using the pointer, we can change the content of the memory location.

Pointer allows accessing the memory location directly. This is the most important feature of the C language.

Syntax#
type * identifier;

Example#
int *ptr;

1.12.4.3 User defined data type



1.12.4.3.1 typedef

In C language, a user can define another name for the data-type also known as *alias*. This alias can later be used to declare variables.

Syntax#
typedef data-type identifier;

here data-type represent built in data type and 'identifier' denotes name given to the data type.

Example#

```
typedef unsigned short int usint;  
typedef int shyam;
```

Now we can use usint as a data-type name for unsigned short int and shyam in the program
i.e. usint a; shyam b;





1.12.4.3.2 enum

An enumeration is a user-defined data type that consists of integral constants. A enumeration type is a "list of key words"

Syntax#

```
enum identifier {value1, value2 .... Value n};
```

The "identifier" is a user defined enumerated data type which can be used to declare variables that have one of the values enclosed within the braces.

We can declare variables to be of this 'new' type as below.

```
enum identifier V1, V2, V3, ..... Vn
```

The enumerated variables V1, V2,.....Vn can have only one of the values value1, value2 value n

Example#

```
enum colour { Red,Green,Blue};  
enum colour c1;
```

```
c1 = Red;
```

C language's compiler assigns integer value 0, 1, 2.... to a member's value1,value2.....respectively. i.e Red has value 0, Green has 1 so on.

We can change default values of enum elements during declaration.

Example#

```
enum day {  
    Monday = 0,  
    Tuesday = 10,  
    Wednesday = 20,  
    Thursday = 3,  
}
```

1.12.5 Initialization of Variable

When a variable declared but not initialized, it contains garbage.

Example#

```
#include<stdio.h>  
#include<conio.h>  
  
main()  
{  
    int a;  
    clrscr();  
    printf("A = %d",a);  
    getch();
```





```
}
```

Output

A = -28762

There are two ways to initialize variable



1.12.5.1 Static initialization

Storing value at compile time is called static initialization.

Syntax#

Data-type variableName=value;

Example#

```
int rollNo=5, total=300;
```

```
float pi=3.14;
```

```
char name[]="Shyam";
```

1.12.5.2 Dynamic initialization

Storing value at runtime is called dynamic initialization.

Syntax#

VariableName3=VariableName1/VariableName2

Example#

```
int a=20,b=5,c,x;
```

```
c=a/b;
```

```
printf("\nEnter no =>");
```

```
scanf("%d",&x);
```





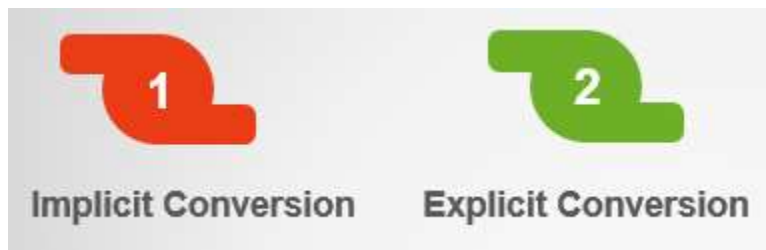
1.13 Type Conversion

Converting an expression of a given type into another type is known as type-casting. typecasting is more use in c language programming.

Here, it is best practice to convert lower data type to higher data type to avoid data loss.

Data will be truncated when higher data type is converted to lower. For example, if float is converted to int, data which is present after decimal point will be lost.

There are two types of type casting in c language:



1.13.1 Implicit Conversion

Implicit conversions do not require any operator for converted.

They are automatically performed when a value is copied to a compatible type in program.

Here, the value of a has been promoted from int to double and we have not had to specify any type-casting operator. This is known as a standard conversion.

Example#

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i=20;
    double p;
    clrscr();
    p=i; // implicit conversion
    printf("implicit value is %d",p);
    getch();
}
```

Output

implicit value is 20.





1.13.2 Explicit Conversion

In c language, many conversions, especially those that imply a different interpretation of the value, require an explicit conversion. We have already seen two notations for explicit type conversion.

They are not automatically performed when a value is copied to a compatible type in program.

Syntax#

variableName = (DataType)VariableName;

Example#

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i=20;
    short p;
    clrscr();
    p = (short) i; // Explicit conversion
    printf("Explicit value is %d",p);
    getch();
}
```

Output

Explicit value is 20.

1.14 Comments

In the C Programming Language, you can place comments in our source code that are not executed as part of the program.

Comments provide clarity to the C source code allowing others to better understand what the code was intended to accomplish and greatly helping in debugging the code.

Comments are especially important in large projects containing hundreds or thousands of lines of source code or in projects in which many contributors are working on the source code.

A comment starts with a slash asterisk /* and ends with a asterisk slash */ and can be anywhere in your program. Comments can span several lines within your C program.

Comments are typically added directly above the related C source code.





Dr. Shyam N. Chawda, C Language Tutorial , 78 74 39 11 91

Adding source code comments to your C source code is a highly recommended practice. In general, it is always better to over comment C source code than to not add enough.

There are two types of comment:



It is important that you choose a style of commenting and use it consistently throughout your source code. Doing so makes the code more readable.

1.14.1 Single Line Comment

Syntax#

```
// comment goes here
```

Example# Comment in Single Line

```
// Author: Shyam
```

1.14.2 Multi-Line Comment

Syntax#

```
/*  
    comment goes here  
*/
```

Example#

```
/*  
 * Author: Aspiration Institute- Shyam Sir  
 * Purpose: To show a comment that spans multiple lines.  
 * Language: C  
*/
```

The compiler will assume that everything after the `/*` symbol is a comment until it reaches the `*/` symbol, even if it spans multiple lines within the C program.