

*Dr. Sachin Kadam*  
*Prof. Gauri (Ranadive) Madan*

**‘C’**  
**Programming**  
**- A Practical Perspective**

**Himalaya Publishing House**

# 'C' PROGRAMMING – A PRACTICAL PERSPECTIVE

**Dr. Sachin Kadam**

*Director – MCA*

Sinhgad Institute of Management and  
Computer Application (SIMCA), Pune.

**Prof. Gauri (Ranadive) Madan**

*Lecturer*

MCA (Master of Computer Application)  
MAEER'S Arts, Commerce & Science College,  
Pune.

First Edition : 2011



**Himalaya Publishing House**

MUMBAI • NEW DELHI • NAGPUR • BENGALURU • HYDERABAD • CHENNAI • PUNE • LUCKNOW • AHMEDABAD  
• ERNAKULAM • BHUBANESWAR • INDORE • KOLKATA • GUWAHATI

© **Authors**

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording and/or otherwise without the prior written permission of the publishers.

**First Edition : 2011**

- 
- 
- Published by** : Mrs. Meena Pandey for **Himalaya Publishing House Pvt. Ltd.**,  
"Ramdoot", Dr. Bhalerao Marg, Girgaon, **Mumbai - 400 004.**  
Phone: 022-23860170/23863863, Fax: 022-23877178  
**E-mail: himpub@vsnl.com; Website: www.himpub.com**
- Branch Offices** :
- New Delhi** : "Pooja Apartments", 4-B, Murari Lal Street, Ansari Road, Darya Ganj,  
New Delhi - 110 002. Phone: 011-23270392, 23278631; Fax: 011-23256286
- Nagpur** : Kundanlal Chandak Industrial Estate, Ghat Road, Nagpur - 440 018.  
Phone: 0712-2738731, 3296733; Telefax: 0712-2721215
- Bengaluru** : No. 16/1 (Old 12/1), 1st Floor, Next to Hotel Highlands, Madhava Nagar,  
Race Course Road, Bengaluru - 560 001. Phone: 080-32919385;  
Telefax: 080-22286611
- Hyderabad** : No. 3-4-184, Lingampally, Besides Raghavendra Swamy Matham,  
Kachiguda, Hyderabad - 500 027. Phone: 040-27560041, 27550139;  
Mobile: 09848130433
- Chennai** : No. 85/50, Bazullah Road, T. Nagar, Chennai - 600 017.  
Phone: 044-28144004/28144005
- Pune** : First Floor, "Laksha" Apartment, No. 527, Mehunpura, Shaniwarpeth  
(Near Prabhat Theatre), Pune - 411 030. Phone: 020-24496323/24496333
- Lucknow** : Jai Baba Bhavan, Church Road, Near Manas Complex and Dr. Awasthi Clinic,  
Aliganj, Lucknow - 226 024. Phone: 0522-2339329, 4068914;  
Mobile: 09305302158, 09415349385, 09389593752
- Ahmedabad** : 114, "SHAIL", 1st Floor, Opp. Madhu Sudan House, C.G. Road,  
Navrang Pura, Ahmedabad - 380 009. Phone: 079-26560126;  
Mobile: 09327324149, 09314679413
- Ernakulam** : 39/104 A, Lakshmi Apartment, Karikkamuri Cross Rd., Ernakulam,  
Cochin - 622011, Kerala. Phone: 0484-2378012, 2378016;  
Mobile: 09344199799
- Bhubaneswar** : 5 Station Square, Bhubaneswar (Odisha) - 751 001. Mobile: 09861046007
- Indore** : Kesardeep Avenue Extension, 73, Narayan Bagh, Flat No. 302, IIIrd Floor,  
Near Humpty Dumpty School, Narayan Bagh, Indore - 452 007 (M.P.).  
Mobile: 09301386468
- Kolkata** : 108/4, Beliaghata Main Road, Near ID Hospital, Opp. SBI Bank,  
Kolkata - 700 010, Phone: 033-32449649, Mobile: 09910440956
- Guwahati** : House No. 15, Behind Pragjyotish College, Near Sharma Printing Press,  
P.O. Bharalumukh, Guwahati - 781009. (Assam).  
Mobile: 09883055590, 09883055536
- DTP by** : Prerana Enterprises, Mumbai
- Printed by** : A to Z Printer, New Delhi.

## Preface

---

We have had a love/hate relationship with C for years. We love the ease with which one can write C programs. We love the development environments that come with many of today's C compilers. But we hate the ease with which one can make mistakes. We hate the attention to picky details that C programming often requires. And above all, we hate the way many C programmers disparage other languages. Let's face it: C is not the ultimate programming language (no other language either). It is, however, a language with which every software developer should become familiar. It has become, for better or for worse, the *lingua franca* of the computer world. The present book is an attempt to bring this *lingua franca* closer to would be computer professionals.

This book is designed as a primary text for a C course. Previous programming experience in a high-level language is helpful, but not necessary for a computer-literate reader. Since the book is self-contained and usable for reference as well as learning, it makes an excellent companion text for a course in data structures, compiler design, operating systems, computer graphics, or other courses that use C as programming language. Due to its ample collection of programs, exercises and emphasis on practical problems, the book will also appeal to readers who are enrolled in a training class or who are learning C by self-study.

This book has been designed with the following goals in mind;

- Be clear, readable and entertaining
- Be accessible to a broad range of readers
- Be authoritative without being pedantic
- Be organized for easy teaching and learning
- Emphasize style
- Avoid dependence on a particular machine, compiler or operating system
- Use illustrations to clarify key concepts

While making sure to achieve the above mentioned goals, we have also incorporated certain features in the book which the readers will certainly appreciate;

- Illustrated programs: Choosing illustrative programs is not an easy job. If programs are too brief and artificial, readers will not get any sense of how the features are used in the real world. On the other hand, if a program is too realistic, its point can easily be lost in a forest of details. We have chosen a middle course, using small, simple examples to make concepts clear when they are first introduced, then gradually building up the complexity
- Exercises: Each chapter ends with a series of questions related to material covered in the chapter. Topics addressed in these sections include Frequently Asked Questions (FAQs).

The book is divided into four parts:

**Part - I:**

Chapter 1: Algorithms and Flowcharts

Chapter 2: Introduction to C

Chapter 3: Tokens and Data Types

Chapter 4: Operators and Expressions

Chapter 5: Control Statements

**Part-II:**

- Chapter 6: Preprocessor
- Chapter 7: Input and Output
- Chapter 8: Functions
- Chapter 9: Arrays

**Part-III:**

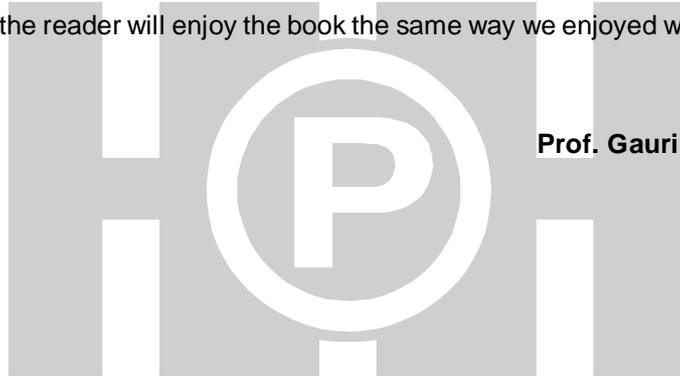
- Chapter 10: Pointers
- Chapter 11: Structures and Unions
- Chapter 12: File Management

**Part-IV:**

- Appendix-I: A complete list of ASCII symbols
- Appendix-II: A complete list of C keywords and operators
- Appendix-III: A header-file wise listing of all functions in the C library

We hope that the reader will enjoy the book the same way we enjoyed writing it.

**Dr. Sachin Kadam**  
**Prof. Gauri (Ranadive) Madan**



# CONTENTS

---

## Preface

### PART – I

<b>1. Algorithms and Flowcharts</b>	<b>3-16</b>
1.1 Problem Solving with Computers	3
1.2 Algorithms	4
1.3 Flowchart	4
1.4 Characteristics of Algorithms	4
1.4.1 Input	
1.4.2 Definite	
1.4.3 Effective	
1.4.4 Terminate	
1.4.5 Output	
1.5 Basic Control Structures in Algorithms	6
1.5.1 Sequencing	
1.5.2 Branching (Selections)	
1.5.3 Repetitions (Looping)	
1.6 Flowchart Symbols	10
1.7 Pseudocode	11
1.7.1 Rules for Pseudocode	
1.7.2 Advantaghes and Disadvantages	
1.8 System Flowcharts	12
1.9 Some examples on Algorithms and Flowcharts	13
<b>2. Introduction to C</b>	<b>17-23</b>
2.1 A Brief History of C	17
2.2 C Is Usually First	18
2.3 ANSI-C Standard	18
2.4 Why use C?	19
2.5 Uses of C	19
2.6 C for Personal Computers	19
2.7 C's Character Set	21
2.8 The form of a C Program	21
2.9 Pre-processor Directives	22
<b>3. Tokens and Data Types</b>	<b>24-33</b>
3.1 C Tokens	24
3.2 Keywords	25

3.3	Variables	25
3.3.1	Declaration of variables	
3.4	Constant Variables	25
3.4.1	Types of Constants	
3.4.1.1	Integer Constant	
3.4.1.2	Real Constant	
3.4.1.3	Character Constant	
3.4.2	Defining Symbolic Constants	
3.4.3	Declaring Variable as Constant	
3.4.4	Naming Constants and Variables	
3.5	Storage Class	27
3.5.1	Auto	
3.5.2	Static	
3.5.3	Extern	
3.5.4	Register	
3.6	Data Types	28
3.6.1	Primary data type	
3.6.1.1	Integer Type	
3.6.1.2	Floating Point Types	
3.6.1.3	Void Type	
3.6.1.4	Character Type	
3.7	Declaration of Variables	29
3.8	User Defined Type Declaration	30
3.9	Special Data Type Enum	30
3.10	Precedence	31
<b>4.</b>	<b>Operators and Expressions</b>	<b>34-45</b>
4.1	Operators Introduction	34
4.1.1	Arithmetic operators	
4.1.2	Relational Operators	
4.1.3	Logical Operators	
4.1.4	Assignment Operators	
4.1.5	Increments and Decrement Operators	
4.1.6	Conditional Operators	
4.1.7	Bitwise Operators	
4.1.8	Special Operators	
4.1.9	The Comma Operator	
4.1.10	The sizeof Operator	
4.2	Expressions	39
4.2.1	Arithmetic Expressions	
4.2.2	Evaluation of Expressions	
4.2.3	Precedence in Arithmetic Operators	
4.2.4	Rules for evaluation of expression	
4.3	Implicit Type Conversion	40

4.4	Explicit Conversion	41
4.5	Operator Precedence and Associativity	41
<b>5.</b>	<b>Control Statements</b>	<b>46-59</b>
5.1	Introduction	46
5.2	Simple and Compound Statements	47
	5.2.1 Assignment	
	5.2.2 Control	
5.3	The if else Statement	47
5.4	The Switch Statement	49
5.5	Loops	49
	5.5.1 While Loop	
	5.5.2 Do While Loop	
	5.5.3 For Loop	
5.6	The Break Statement	52
5.7	The Continue Statement	52
5.8	The Goto Statement	52
<b>PART – II</b>		
<b>6.</b>	<b>Preprocessor</b>	<b>63-66</b>
6.1	Introduction	63
6.2	Preprocessor Directives	64
6.3	Macros	64
<b>7.</b>	<b>Input and Output</b>	<b>67-72</b>
7.1	Introduction	67
7.2	Single Character Input Output	68
7.3	String Input and Output	68
7.4	Formatted Input for scanf	69
	7.4.1 Scanf()	
	7.4.2 Printf()	
7.5	Printing Numbers and Escape Sequences	70
7.6	Specifier Meaning	70
<b>8.</b>	<b>Functions</b>	<b>73-108</b>
8.1	What is a Function?	73
8.2	Structure of a Function	74
	8.2.1 Function Header	
	8.2.2 Function Body	
	8.2.3 Function Prototypes	
8.3	Why we use Functions?	75
8.4	Function Declaration	75
8.5	Calling the Function	75
8.6	Function Definition	76
8.7	The General Guideline for the Function	76

8.8	Types of Functions	79
8.8.1	Functions with no arguments and no return values	
8.8.2	Functions with arguments but no return values	
8.8.3	Functions with arguments and return values	
8.8.4	Return value data type of function	
8.9	Void Functions	84
8.10	Call By Value	84
8.11	Call By Reference (Read chapter pointer)	85
8.12	Recursion	85
8.13	The Scope and Lifetime of Variables in Functions	87
8.13.1	Automatic variables	
8.13.2	External variables	
8.13.3	Static variables	
8.13.4	Register variables	
8.14	String Operations	91
<b>9.</b>	<b>Arrays</b>	<b>109-147</b>
9.1	Introduction	109
9.2	Declaration of Arrays	110
9.2.1	Some Variations in Declaring Arrays	
9.3	Initialization of Arrays	112
9.4	Multidimensional Arrays	114
9.5	Subscripted Variable Rules	114
9.6	Elements of Multidimensional Arrays	115
9.6.1	Initialization of Multidimensional Arrays	
9.7	Static and Dynamic Allocation of Multidimensional Arrays	117
<b>PART – III</b>		
<b>10.</b>	<b>Pointers</b>	<b>151-164</b>
10.1	Introduction	151
10.2	What is a pointer?	152
10.2.1	Address operator	
10.3	Pointer Expressions and Pointer Arithmetic	154
10.4	Pointers and Function	155
10.5	Pointer to Arrays	156
10.6	Pointers and Structures	157
10.7	Pointers on Pointer	158
10.8	Dynamic Memory Allocation	158
10.9	Memory Allocations Process	159
10.9.1	Allocating a block of memory	
10.9.2	Allocating multiple blocks of memory	
10.10	Releasing the Used Space	160
10.11	To Alter the Size of Allocated Memory	160

<b>11. Structures and Unions</b>	<b>165-187</b>
11.1 What is a Structure?	165
11.2 Giving Values to Members	167
11.3 Initializing Structure	169
11.4 Functions and Structures	169
11.4.1 Passing structure to elements to functions	
11.4.2 Passing entire structure to functions	
11.5 Arrays of Structure	171
11.6 Structure within a Structure	172
11.7 Union	172
<b>12. File Management</b>	<b>188-210</b>
12.1 Introduction	188
12.2 File Operation Functions in C	189
12.2.1 Defining and opening a file	
12.2.2 Closing a file	
12.3 File I/O	190
12.3.1 The getc and putc functions	
12.3.2 The getw and putw functions	
12.3.3 The fprintf & fscanf functions	
12.3.3.1 The general format of fprintf is	
12.3.3.2 The general format of fscanf is	
12.4 Command Line Parameters: Arguments to main()	197
<b>PART – IV</b>	
<b>Appendix</b>	
Appendix – I: A Complete List of ASCII Symbols	<b>213-214</b>
Appendix – II: A Complete List of C Keywords and Operators	<b>215-217</b>
Appendix – III: A Header-filewise Listing of all Functions in the C Library	<b>218-233</b>

# PART - I

1. **Algorithms and Flowcharts**
2. **Introduction to C Programming**
3. **Tokens and Data Types**
4. **Operators and Expressions**
5. **Control Statement**

# 1

# ALGORITHMS AND FLOWCHARTS

## OBJECTIVE

*In this chapter you will learn Algorithm, Flowchart, and Characteristics of Algorithm, Control structure in Algorithm, Flow chart Symbols*

## 1.1 Problem Solving with Computers

The computer always demands unambiguous definition of the problem, which is simple to carry out and straight forward to implement in order to solve the problem. At this level the programmer is assumed to have certain amount of background knowledge, reasoning skills and hints for problem solving.

Steps involved in problem solving using computers are :

- ☞ Make a closer study of given problem and analyze it
- ☞ Redefine the problem if necessary
- ☞ Specify what is expected as an outcome of processing
- ☞ Find a suitable solution method to solve the given problem
- ☞ Based on above analysis, formulate a set of steps or procedural steps (known as Algorithm) of the chosen method of solution

- ☞ Express the algorithm in precise notation, feed it to computer, compute the results and reevaluate it
- ☞ Repeat the above steps till you get correct solution of given problem.

## 1.2 Algorithms

The finite set of procedural steps, where each step is precise, unambiguous and capable of being carried out by a machine in a finite time is known as **Algorithm**. As computer expects unambiguous, precise and definite instructions or command to solve a problem so the algorithm must be expressed in a 'precise notation'. An algorithm when it is expressed in one of the precise notation is called as a computer program. Before writing the computer program from algorithm an intermediate step, i.e., flowcharting is needed.

## 1.3 Flowchart

A flowchart is the pictorial representation of algorithmic steps to be performed sequentially to arrive at the result. Actually the computer program is written only after writing the flowchart.

**Statement:** Find sum and average of three numbers and print the result

Redefined problem: Write a program to find the sum and average of 3 float number. Accept 3 numbers from standard input device and display the result on monitor, Outcome of process is sum and average of three numbers taken from input device.

### (1) Algorithm to add 3 numbers and prints sum and average.

- Step 1. [First input the 3 numbers]  
Input a, b, c
- Step 2. [Finding the sum of given numbers]  
 $Sum \leftarrow (a+b+c)$
- Step 3. [Finding the average]  
 $Average \leftarrow sum/3.0$
- Step 4. [Print the results]  
Write sum, average
- Step 5. [End of the algorithm]  
End

## 1.4 Characteristics of Algorithms

### 1.4.1. Input

Algorithm starts with a procedural step to accept input data. The subsequent steps in the algorithm process these data elements. But this input step is optional and is used only when the algorithm needs data at the time of execution.

---

### 1.4.2. Definite

Each operational step must be definite, i.e., it should specify what should be done. In other words there should not be any confusing instruction.

- For example: (1) "Compute 5/0"  
(2) "Add 5 or 10 to 5"

Above statements are not allowed in algorithms because in first statement result of 5/0 is infinity which is not defined in Computers and also in all branches of Mathematics. The second statement, which one of the two numbers 5 or 10 should be added to 5, is not clear.

### 1.4.3. Effective

Each operation must be effective to carry out. Effectiveness property of an algorithm says that each operational step can be at least in principle, being carried in a minimum amount of time.

For example, analyze the mathematical formula given below

$$Y = PX^3 + QX^2 + RX + S$$

The above statement can be written as C - statement as follows

$$Y = (P * X * X * X) + (Q * X * X) + (R * X) + S;$$

But to execute the above statement, the computer has to do six multiplications and three additions. We can modify the above formulas as below i.e.

$$\implies Y = X (PX^2 + QX + R) + S$$

$$\implies Y = X (X(PX + Q) + R) + S$$

In C-statement, this can be written as  $Y = X * (X * ((P * X) + Q) + R) + S$ .

Now there are only 3 multiplications and 3 additions. Therefore the above statement is more efficient and takes less time for execution. The computing speed decreases as we go from operations additions, subtractions, multiplication, divisions, power( ) functions, logarithmic calculations, trigonometric functions and other mathematical functions. Therefore one should try to replace slow operations by faster operations as far as possible.

For examples:

Replace pow(x, 3) by  $x * x * x$

Replace  $3 * y$  by  $y + y + y$

Replace  $n / 2.0$  by  $n * 0.5$

Also try to avoid the repetitions of operations.

For example:

Repeated operations (slow)	more efficient (fast)
Sum = a + b + c + d	sum = a + b + c + d
avg = (a + b + c + d) / 4.0	avg = sum / 4.0

---

#### 1.4.4. Terminate

After some minimum number of operations, algorithm must come to an end, i.e., the total time to carry out all the steps in algorithm must be limited (finite).

#### 1.4.5. Output

An algorithm is written to solve the problem therefore it must produce one or more computed results. In order to incorporate the definiteness property of the algorithm, algorithms will be expressed using a precise notation called programming languages. Such computer programming languages are designed to assign clarity, definiteness and effectiveness to each of the instructions of the algorithm. The languages actually consist of all necessary features to support the characteristics of an algorithm.

### 1.5 Basic Control Structures in Algorithms

Usually the algorithm consists of finite set of sequential procedural steps. The algorithm may have conditions, which are used to take decision, and depending upon decisions, a particular group of statements are selected for execution. There are also situations in which some of the statements have to be executed repeatedly. So to meet above situations of problem solving, algorithm uses 3 basic forms of control structures:

1.5.1 Sequencing

1.5.2 Branching (selections)

1.5.3 Repetitions (looping)

These are also called as program structures used in developing structured programs.

#### 1.5.1 Sequencing

The sequence structure indicates the sequential flow of program logic, i.e., step by step execution of program statements in the algorithm.

For example:

**(1) In algorithm:**

Step 1: [Input the three numbers]

Input  $n_1, n_2, n_3$

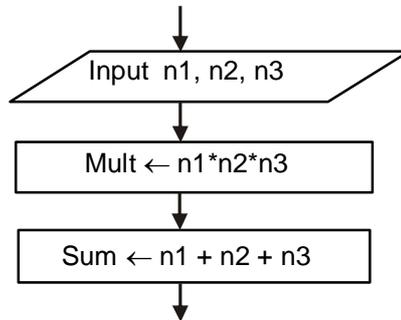
Step 2: [Multiply given numbers and store the result in another variable]

Mult  $\leftarrow n_1 * n_2 * n_3$

Step 3: [Add all numbers and store the sum in another variable]

Sum  $\leftarrow n_1 + n_2 + n_3$

**(2) In Flowchart:**



**(3) In C program:**

```
.  
. .  
scanf("%d%d%d",&n1,&n2,&n3);  
mult=n1*n2*n3;  
sum=n1+n2+n3;  
. .
```

The step-by-step execution of statements in the algorithm (or in flowchart, programs) is known as sequencing or sequential execution.

### 1.5.2. Branching (Selections)

Execution of a group of statements depending upon the given condition is known as branching or selection. In algorithm if-then-else statement is used for branching (selection). Its syntax is as follows:

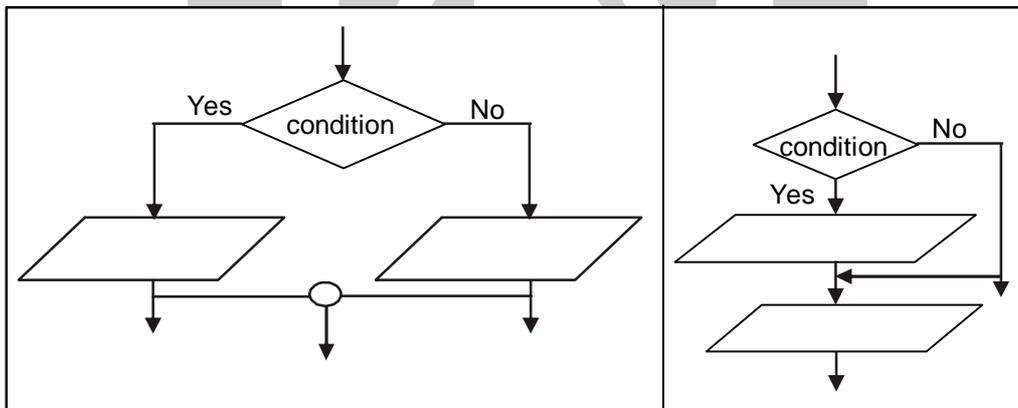
```
if (condition) then  
    statement-1  
    statement-2  
    -  
    -  
else  
    statement-a  
    statement-b  
    -
```

Here the condition is evaluated first and if the condition is true, the statements after “then” are executed and if condition is false the statements after “else” part are executed. Here the “else” part is optional and this is used only when it is required. For example:

**(1) In Algorithm:**

- (a) if (  $a > 0$  ) then  
    write “A is greater than 0”  
else  
    write “A is not greater than 0”  
     $\leftarrow x + A$
- (b) if (  $a > 0$  ) then  
    write “A is +ve number”  
     $\leftarrow x + a$

**(2) In Flowchart:**



### 1.5.3. Repetitions (looping)

The repetitive execution of a group of statements or program segment takes place as long as the loop condition is true and when the loop condition becomes false, the loop or repetition will be terminated. This process is known as repetition or looping.

**(1) In algorithms, repeat, for, do-while etc are used for looping.**

Ex: Algorithm for finding sum and numbers from 1 to N

Step 1: [Input numbers]

Input N

Step 2: [Initialize the variable sum to zero]

Sum  $\leftarrow$  0

Step 3: [Calculate the sum by looping]

Do

Sum  $\leftarrow$  Sum+N

N  $\leftarrow$  N-1

while N>0

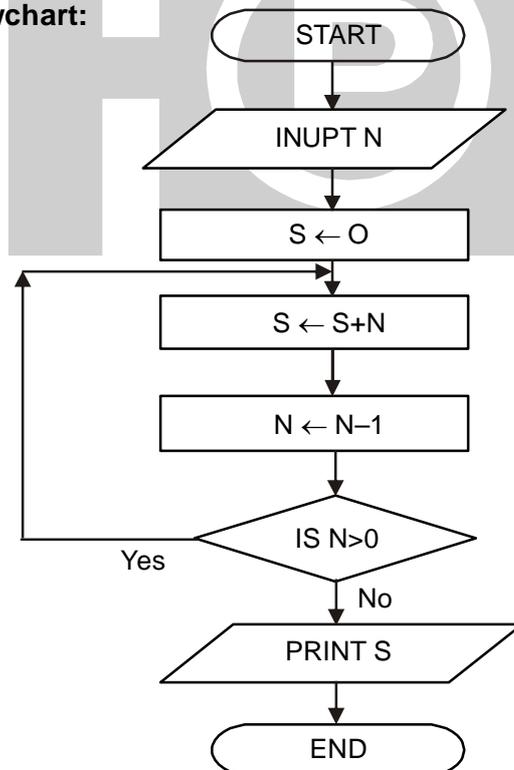
Step 4: [Print the sum of numbers]

Write Sum

Step 5: [End of the algorithm]

End

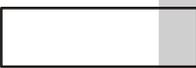
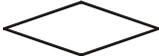
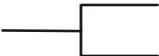
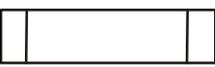
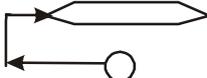
**(2) In Flowchart:**



**(3) In C programs:**

```
main()
{
    int n,sum=0;
    do
    {
        sum+=n;
        n--;
    } while(n>0);
    printf("\n The sum is %d",sum);
}
```

## 1.6 Flowchart Symbols

Symbol	Meaning
	Start and End of Flowchart.
	Represent input and output operations.
	Assignment operations and other computational operations.
	Conditions tested. Based on results path is selected. Also called as test box.
	Connectors for connecting two parts of flowcharts.
	Comment.
	Connectors for different boxes of flowchart and shows the direction of execution. Used to indicate predefined process like functions in C.
	Manual task.
	Repetitions of statements mainly for "for loop."

---

## 1.7 Pseudocode

Flowcharts were the first design tool to be widely used, but unfortunately they do not very well reflect some of the concepts of structured programming. Pseudocode, on the other hand, is a newer tool and has features that make it more reflective for the structured concepts. Unfortunately, the narrative presentation is not as easy to understand and follow.

### 1.7.1 Rules for Pseudocode

- (1) Write only one statement per line
- (2) Each statement in your pseudocode should express just one action for the computer. If the task list is properly drawn, then in most cases each task will correspond to one line of pseudocode
- (3) Capitalize keywords
- (4) Indent to show hierarchy

We will use a particular indentation pattern in each of the design structures

**SEQUENCE:** keep statements that are “stacked” in sequence all starting in the same column

**SELECTION:** indent the statements that fall inside the selection structure, but not the keywords that form the selection

**LOOPING:** indent the statements that fall inside the loop, but not the keywords that form the loop

- (5) End multiline structures
- (6) Keep statements language independent

Resist the urge to write in whatever language you are most comfortable with. In the long run, you will save time! There may be special features available in the language you plan to eventually write the program in; if you are SURE it will be written in that language, then you can use the features. If not, then avoid using the special features.

**Selection:**

The pseudocode for this would be:

```
IF amount < 1000
  interestRate = 0.06
ELSE
  interestRate = 0.10
ENDIF
```

**Looping:**

The pseudocode for this would be:

```
count = 0
WHILE count < 10
```

---

ADD 1 to count  
WRITE count  
ENDWHILE  
WRITE "The end"

### 1.7.2 Advantages and Disadvantages

#### **Pseudocode Disadvantages**

- (1) It's not visual
- (2) There is no accepted standard, so it varies widely from company to company

#### **Pseudocode Advantages**

- (1) Can be done easily on a word processor
- (2) Easily modified
- (3) Implements structured concepts well

#### **Flowchart Disadvantages**

- (1) Hard to modify
- (2) Need special software (not so much now!)
- (3) Structured design elements not all implemented

#### **Flowchart Advantages**

- (1) Standardized: all pretty much agree on the symbols and their meaning
- (2) Visual (but this does bring some limitations)

## 1.8 System Flowcharts

A system flowchart is a flowchart which gives overall view of computer system in addition to the pictorial representation of the flow of operations of an algorithm.

In other words as the name itself indicates, system flowcharts are drawn to show diagrammatically, the total environment of computer system on which the given algorithm or program is to be solved. System flowcharts provide us a list of special symbols for indicating input-output devices on which data transactions are read/printed. Thereby one will get a clear picture of input/output devices connected to a computer system. Therefore system flowchart will further improve and assist the computer programming; because system flowcharts not only specify the operations to be performed pictorially but the computer system environment at which all these data processing activities are handled.

To summarize, a system flowchart is a graphical representation that gives the complete view of a data processing system. It has flowchart symbols to show the following:

The activities (either manual or computing) are carried out within such a data processing system. The I/O devices and storage media used to store/handle the data file, transactions, etc., are involved in data processing activities.

## 1.9. Some examples on Algorithms and Flowcharts

(1) Write an algorithm to accept temperature in Fahrenheit scale and convert into Celsius scale and draw a flowchart.

Formula:  $C = (F-32)/1.8$  where F is Fahrenheit value and C is Celsius value.

**Algorithm : Temperature conversion from Fahrenheit value to Celsius value.**

Step 1: [Input temperature in Fahrenheit scale]

Input F

Step 2: [Compute centigrade temperature]

$C \leftarrow (F-32)/1.8$

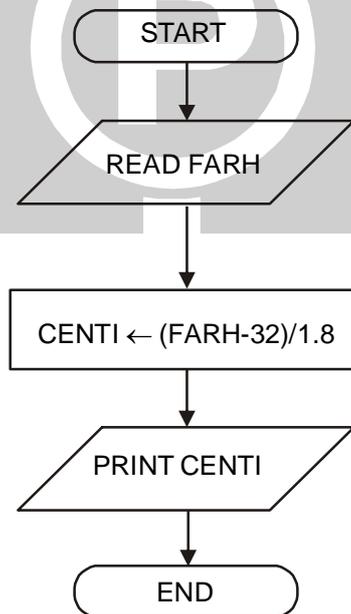
Step 3: [Print the result]

Write "Equivalent temperature in Celsius scale is", C

Step 5: [End of algorithm]

End

**Flowchart:**



(2) Write an algorithm to find biggest of three numbers and draw the flowchart.

**Algorithm: Finding the biggest of three numbers**

Step 1: [Input three numbers from input device]

Input a,b,c

Step 2: [Store the first number in variable big assuming it as the biggest number]

Big  $\leftarrow$  a

Step 3: [Compare second value with big]

If  $b > \text{big}$  then  $\text{big} \leftarrow b$

Step 4: [Compare second value with big]

If  $c > \text{big}$  then  $\text{big} \leftarrow c$

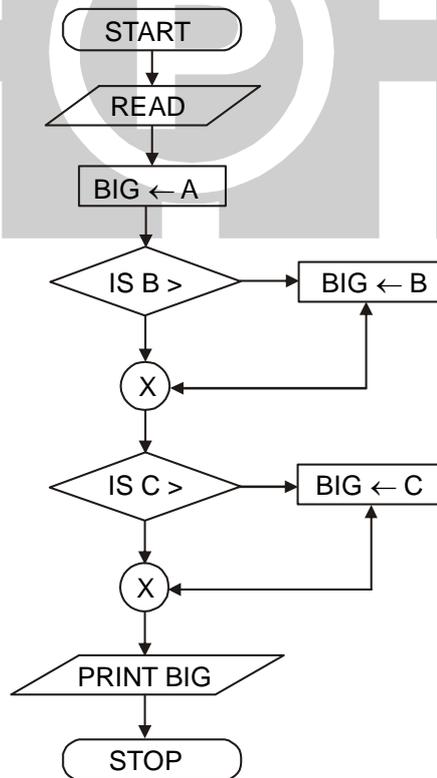
Step 5: [Print the results]

Write "Biggest number is", big

Step 6: [End of the algorithm]

End

**Flowchart:**



(3) Write an algorithm to find the factorial of a number and also draw the flowchart.

**Algorithm : To find the factorial of a given number**

Step 1: [Input the number ]

Input N

Step 2: [Initialize the factorial number to 1]

Fact  $\leftarrow$  1

Step 3: [ Calculate factorial by looping]

Repeat for I=1 to N steps 1

Fact  $\leftarrow$  Fact \* I

Loop

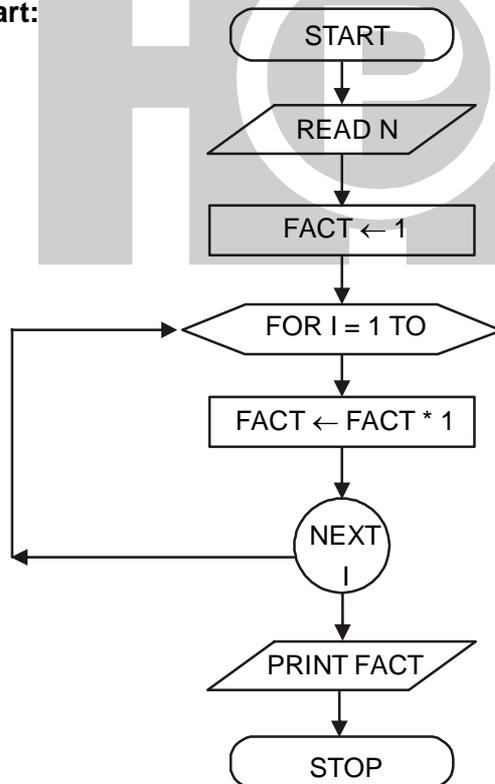
Step 4: [Print the factorial value]

Write "Factorial of",N, "is",fact

Step 5: [End of the algorithm]

End

**Flowchart:**



---

**EXERCISE**

- (1) Explain these important terms :
  - (a) Algorithm
  - (b) Flowchart
  - (c) System Flowchart
- (2) What is the difference between a flowchart and a system flowchart ?
- (3) Why are flowcharts usually used to describe algorithms instead of written descriptions ?
- (4) Flowcharts can also be used to describe activities that humans carry out. Draw a flowchart for one of these activities :
  - (a) Making a telephone call. (b) Making a cup of tea. (c) Repairing a bicycle tyre.
- (5) Choose an information system such as a spreadsheet or database that you have created. Draw either a system flowchart or a flowchart to explain how the system operates.

