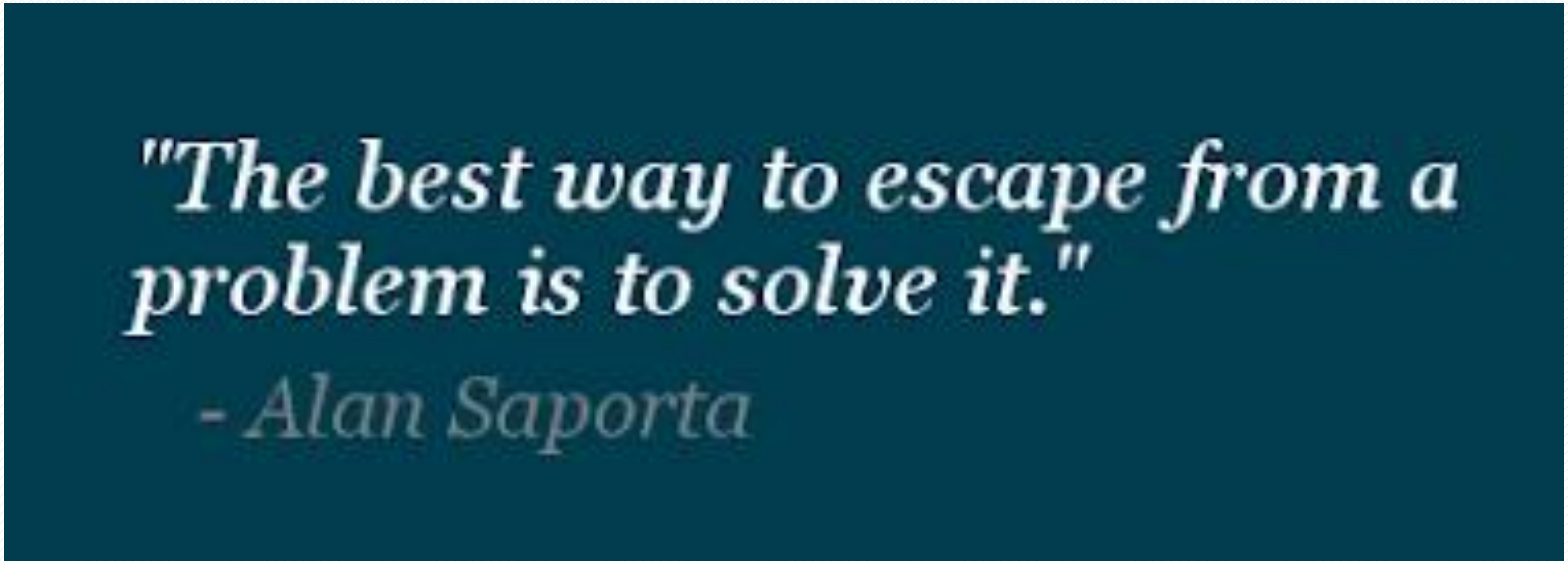
MCS-011 Problem Solving and Programming

An Introduction to C (Block : 1 Unit 1 to 4)

Prof. V. V. Subrahmanyam,
Director
School of Computer and Information Sciences
IGNOU, New Delhi

Date: 10-04-2020 Time: 11-00



"The best way to escape from a problem is to solve it."

- Alan Saporta

Introduction

- Knowledge in a programming language is prerequisite to the study of most of the computer science courses.
- A programming language is the principal interface with the computer.
- Understanding the variety of programming languages and the design trade offs between the different programming paradigms makes it much easier to master new languages quickly.

Algorithm

- An **algorithm** is a finite set of steps defining the solution of a particular problem. An algorithm is expressed in pseudocode - something resembling C language or Pascal, but with some statements in English rather than within the programming language.

Example -1

Let us try to develop an algorithm to compute and display the sum of two numbers:

1. Start
2. Read two numbers *a* and *b*
3. Calculate the sum of *a* and *b* and store it in *sum*
4. Display the value of *sum*
5. Stop

Example -2

Algorithm to calculate the factorial of a given number.

```
1. Start
2. Read the number n
3. [Initialization] $i \leftarrow 1$ , $fact \leftarrow 1$
4. Repeat steps 4 through 6 until $i = n$
5. $fact \leftarrow fact * i$
6. $i \leftarrow i + 1$
7. Print fact
8. Stop
```

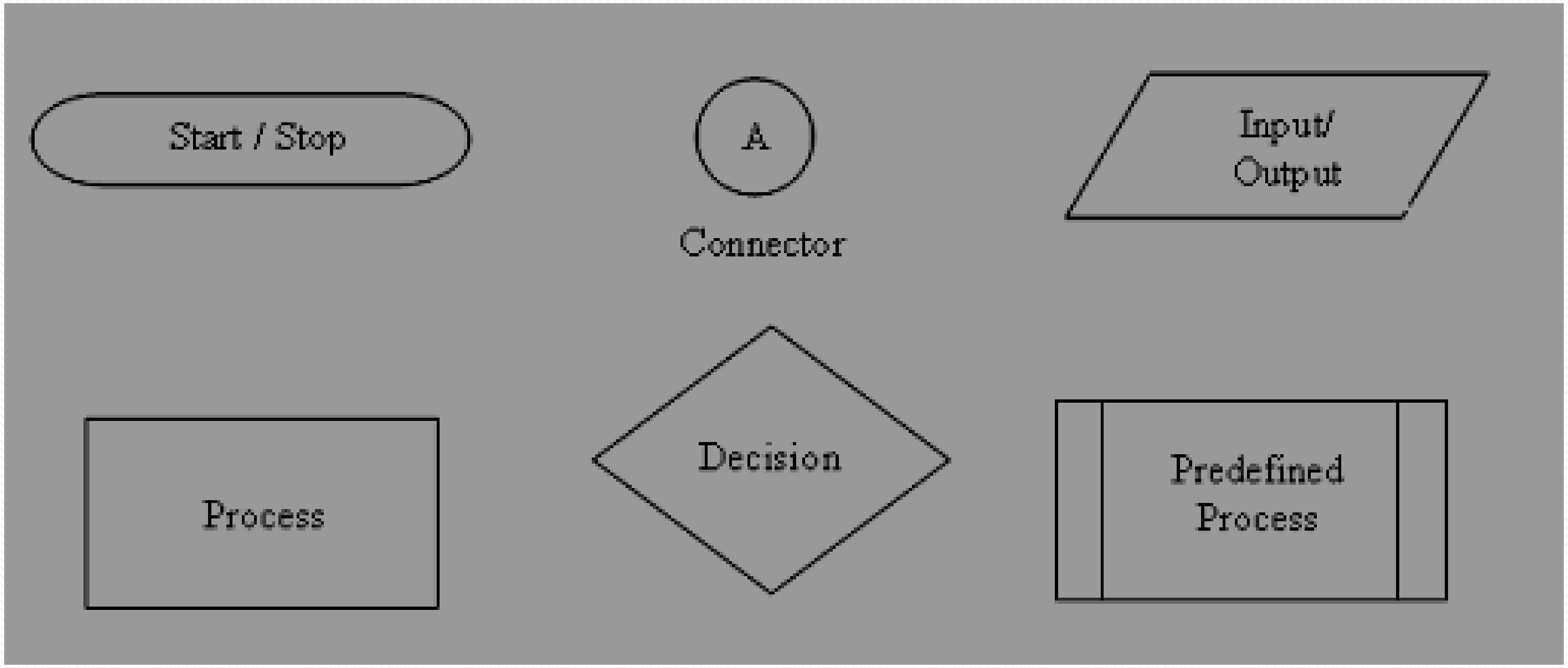
Check your progress - 1

- Write algorithms for the following simple problems:
  - To find the largest among the 3 numbers given.
  - To find the sum and average of given 10 integers.
  - To check whether the given number is prime or not.
  - To check whether the given number is odd or even.

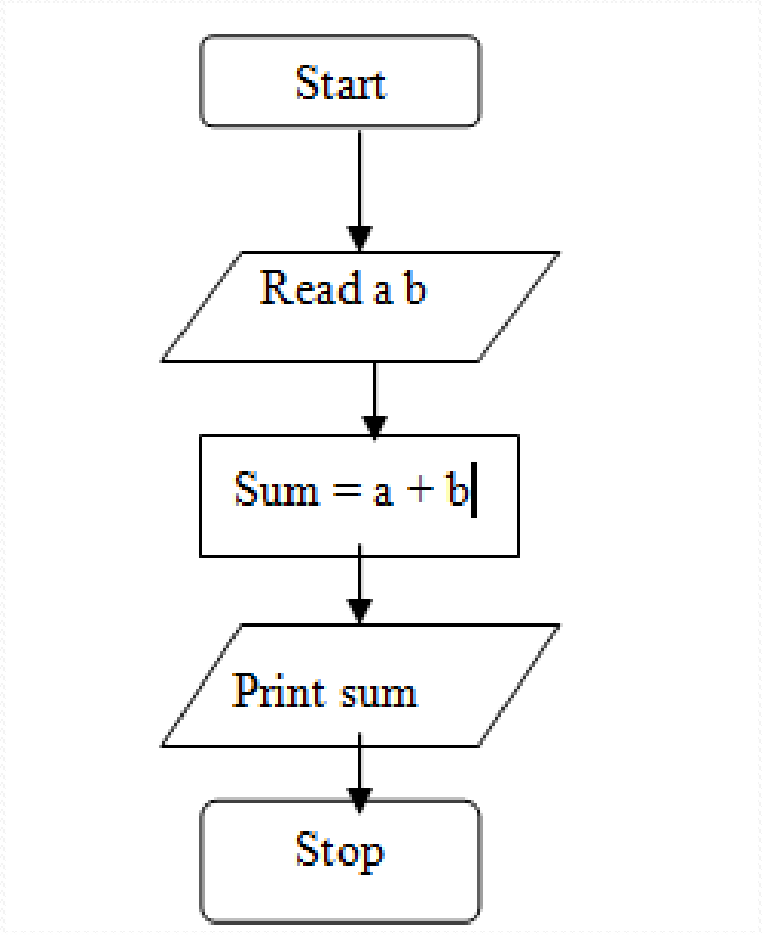
Flowchart

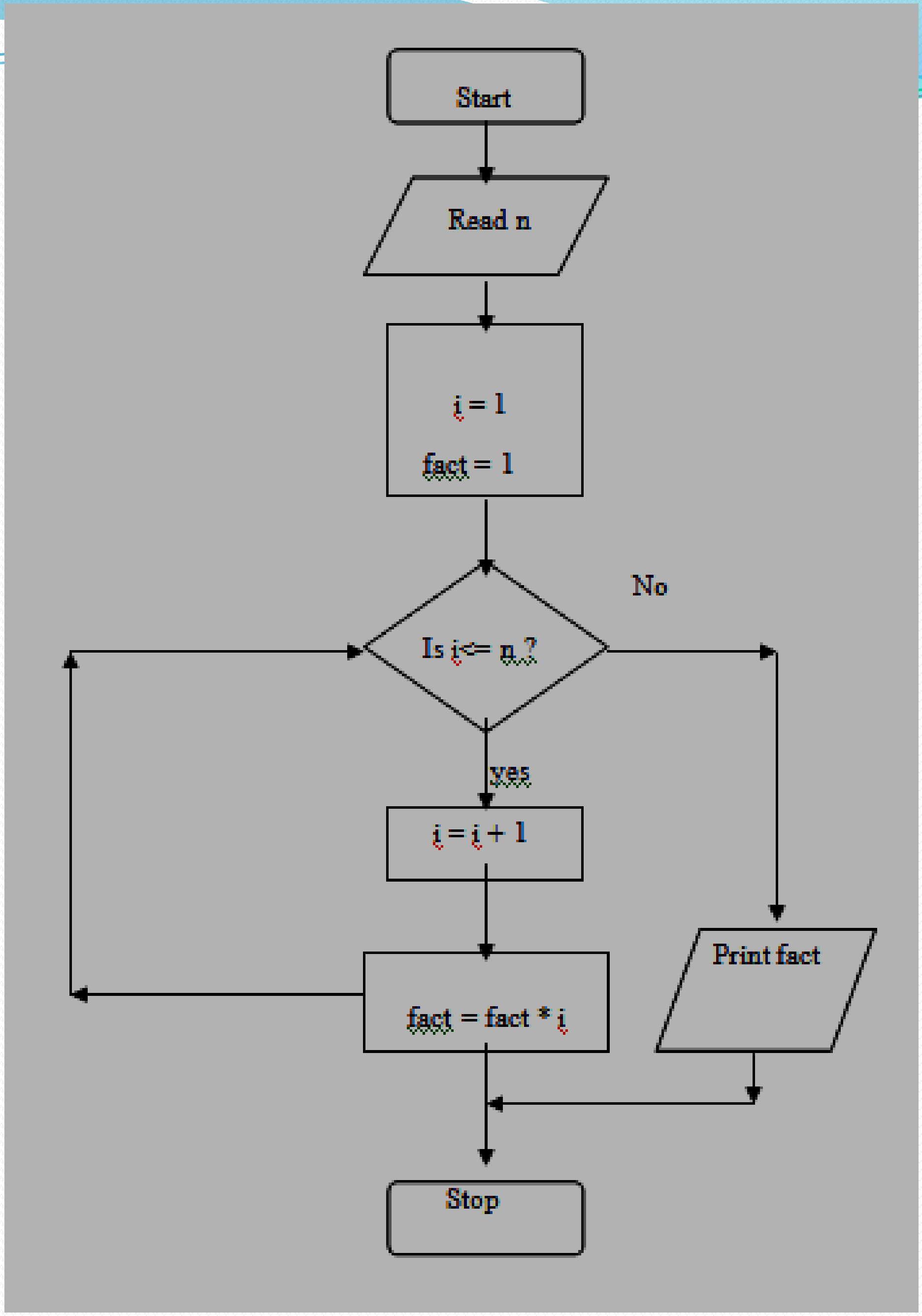
- Flowchart is a graphical representation of an algorithm.
- It makes use of symbols which are connected among them to indicate the flow of information and processing.
- It will show the general outline of how to solve a problem or perform a task.
- It is prepared for better understanding of the algorithm.

Flowchart Symbols



Flowchart for sum of 2 numbers




Start
Read n
i = 1
fact = 1
Is i<= n ?
No
yes
i = i + 1
fact = fact * i
Print fact
Stop

Check Your Progress - 2

- Draw Flowcharts for the following simple problems:
  - To find the largest among the 3 numbers given.
  - To find the sum and average of given 10 integers.
  - To check whether the given number is prime or not.
  - To check whether the given number is odd or even.

Programming Language and a Program

- **Programming Language:** In practice, it is necessary to express an algorithm using some programming language to instruct the computer to solve the problem.
- **Program:** A sequence of instructions written in any programming language to solve the problem using a computer.

Categories of Programming Languages

- **Low level languages or Machine oriented languages**
- **High Level Languages or Problem Oriented languages**

Low level languages or Machine oriented languages

- Whose design is governed by the circuitry and the structure of the machine.
- Difficult to learn
- These are designed to give a better machine efficiency i.e., faster program execution.
- Machine dependent.

Examples: Machine language, Assembly language

High level languages or Problem Oriented languages

- These are oriented towards describing the procedures for solving the problem.
- Machine Independent
- Easy to learn
- Machine directly cannot understand them.
- Examples: FORTRAN, PASCAL, C etc.

C Programming Language

- Developed at AT & T Bell Laboratory in 1970's.
- Designed by Dennis Ritchie.

Salient features of C

- General Purpose, structured programming language.
- It can considered as a High level language, however as it combines both the features, it can be treated as a Middle level language.
- Portable
- Easy to debug
- Easy to test and maintain

Structure of a C Program

```
/*Comments*/
Preprocessor directives
Global data declarations
main()
{
Declaration part;
----
Program Statements;
--
---
---
}
User defined functions
```

A Simple C Program

```
/* Program to print a message*/
#include <stdio.h>
main()
{
printf(“I am in the first semester of MCA\n”);
}
```

Program to add to numbers

```
/* Program to add to numbers*/
#include <stdio.h>
main()
{
int a, b , sum;
printf (“ Enter the values of a and b:\n”);
scanf(“%d, %d”, &a, &b);
sum = a+b;
printf(“the sum is %d”, sum);
}
```

C Character Set

Uppercase Letters: A to Z

Lowercase Letters: a to z

Digits: 0 to 9

Certain Special characters as building blocks to form basic program elements (e.g. constants, variables, operators, expressions etc..)

Special symbols: %, &, +, _ , - # etc.

Identifiers

- Identifiers are the names that are given to the various program elements, such as variables, functions and arrays.
- Identifiers consist of letters and digits, in any order, except the first character must be a letter.
- Both upper case and lower case are allowed.
- No special symbols, except the underscore(_) is allowed.
- An identifier can also begin with an underscore(_).

Examples: x, y12, sum_1, amount, _temp etc..

Keywords

- Reserved words that have standard, predefined meaning in C language.
- These are used for intended purpose only, these cannot be used as programmer-defined identifiers.

Examples: auto, break, case, switch, for, goto, struct etc..

Basic Data types

Data type	**Description**	**Typical Memory Requirements**
Int	Integer	2 bytes or one word
Char	A Character	1 byte
Float	Decimal number	4 bytes
Double	Double precision	8 bytes

Constants

- Integer Constants
- Floating-point constants
- Character Constants
- String Constants

Variables

- It is an identifier that is used to represent some specified type of information within a designated portion of a program.
- Is used to represent a single data item (a numerical quantity or a character constant).
- The data item must be assigned to the variable at some point of the program and later it can be referenced with the name.

Declarations

- A declaration associates a group of variables with a specific data type.
- In C, all the variables must be declared before they can appear in executable statements.

Examples:

```
int a;
int a, b, c;
char flag;
```

Symbolic Constants

- It is the name that substitutes for a sequence of characters.
- The characters may represent a numeric constant, a character constant and a string constant.

Examples:

```
#define RATE  0.23
#define PI   3.1415
#define TRUE   1
```

Statements

- A statement causes the computer to carry out some action.
  - Expression statement
  - Compound statement
  - Control statement

Arithmetic Operators

Operator	**Purpose**
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Remainder after the division

Relational Operators

Operator	Meaning
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Logical Operators

Operator	**Meaning**
&&	And
||	Or

Assignment Operator

Identifier = expression;

Examples: a = 3;

```
x=y;
i=j=1;
area = Length * breadth;
```

Conditional Operator

- The syntax is as follows:

  (condition)? (expression1): (expression2);

Examples

(i) x= (y<20) ? 9: 10;

This means, if (y<20), then x=9 else x=10;

(ii) printf (“%s\n”, grade>=50? “Passed”: “failed”);

The above statement will print “passed” grade>=50 else it will print “failed”

(iii) (a>b) ? printf (“a is greater than b \n”): printf (“b is greater than a \n”);

C Shorthand

- C has a special shorthand that simplifies coding of certain type of assignment statements.

For example:

```
a = a+2;
can be written as        a += 2;
```

- Syntax: ***variable operator* = *variable* / *constant* / *expression***

Precedence of Operators

Operators	**Associativity**
( )	Left to right
! ++ -- (*type*) sizeof	Right to left
/ %	Left to right
+ -	Left to right
< <= > >=	Left to right
== !=	Left to right
&&	Left to right
||	Left to right
?:	Right to left
= += -= *= /= %= &&= ||=	Right to left
,	Left to right