UNIT -1 : Introduction To 'C' Language

Brief history of C language

C was developed at Bell Laboratories in 1972 by Dennis Ritchie. Many of its principles and ideas were taken from the earlier language B and B's earlier ancestors BCPL and CPL. CPL (Combined Programming Language) was developed with the purpose of creating a language that was capable of both high level, machine independent programming and would still allow the programmer to control the behaviour of individual bits of information. The one major drawback of CPL was that it was too large for use in many applications. In 1967, BCPL (Basic CPL) was created as a scaled down version of CPL while still retaining its basic features. In 1970, Ken Thompson, while working at Bell Labs, took this process further by developing the B language. Finally in 1972, a co-worker of Ken Thompson, Dennis Ritchie, returned some of the generality found in BCPL to the B language in the process of developing the language we now know as C.

In 1983, the American National Standards Institute (ANSI) formed a committee to establish a standard definition of C which became known as ANSI Standard C. Today C is in widespread use with a rich standard library of functions.

C programming language was developed by **Dennis Ritche** in 1972.

C language is improved version of B language where B language is improved version of BCPL(Basic Combined programming Language)

The letter „C" is the second letter of BCPL. Hence C stands for "combined".

C is a high level language.

C can be used for developing application programs and OS and also for processing scientific and engineering data.

Features of C language

1. C supports variety of data types.
2. It supports powerful operators.
3. It has rich set of built-in functions.
4. It has 32 keywords/reserved words.
5. C is highly portable.
6. Allows user to add functions to C library.
7. C is faster than BASIC
8. C is case sensitive.
9. Highly flexible in developing application programs and OS.
10. It supports pointers.

C character set.

Character set: The character set is the fundamental raw material of any language and they are used to represent information. Like natural languages, computer language will also have well defined character set, which is useful to build the programs.

Character Set Consists Of:

Types	Character Set
Lowercase Letters	a-z
Uppercase Letters	A to Z
Digits	0-9
Special Characters	!@#$%^&*
White Spaces	Tab Or New line Or Space

Valid C Characters : Special Characters are listed below:

Symbol	Meaning
~	Tilde
!	Exclamation-mark
#	Number-sign
$	Dollar-sign
%	Percent-sign
^	Caret
&	Ampersand
*	Asterisk
(	Left-parenthesis

)	Right-parenthesis
_	Underscore
+	Plus-sign
|	Vertical bar
\	Backslash
`	Apostrophe
–	Minus sign
=	Equal to sign
{	Left brace
}	Right brace
[	Left bracket
]	Right bracket
:	Colon
”	Quotation mark
;	Semicolon
<	Opening angle bracket
>	Closing angle bracket
?	Question mark
’	Comma

.	Period
/	Slash

Variables and Identifiers:

Variables:

Variable in C Programming is also called as container to store the data. Variable name may have different data types to identify the type of value stored. Suppose we declare variable of type integer then it can store only integer values. Variable is considered as one of the building block of C Programming which is also called as identifier.

A Variable is a name given to the memory location where the actual data is stored.

Variables are those whose value may vary during program execution.

Variables are represented by symbolic names.

Example: num1, sum, total.

Examples for valid variable names:

Rama
Gptmbl_
_1947
key_12
pralaya12

Examples for invalid variable names

12_diode: invalid because first letter cannot be a digit.
nav@gmail: invalid because it contains a special character @.
continue: invalid because continue is a keyword.
Naveen kumar: invalid because it contains a special character(blank space).

Identifiers:

Identifiers are the names of variables denoting values, labels, functions, arrays etc.

Identifier must be unique. They are created to give unique name to identify it during the execution of the program. For example:

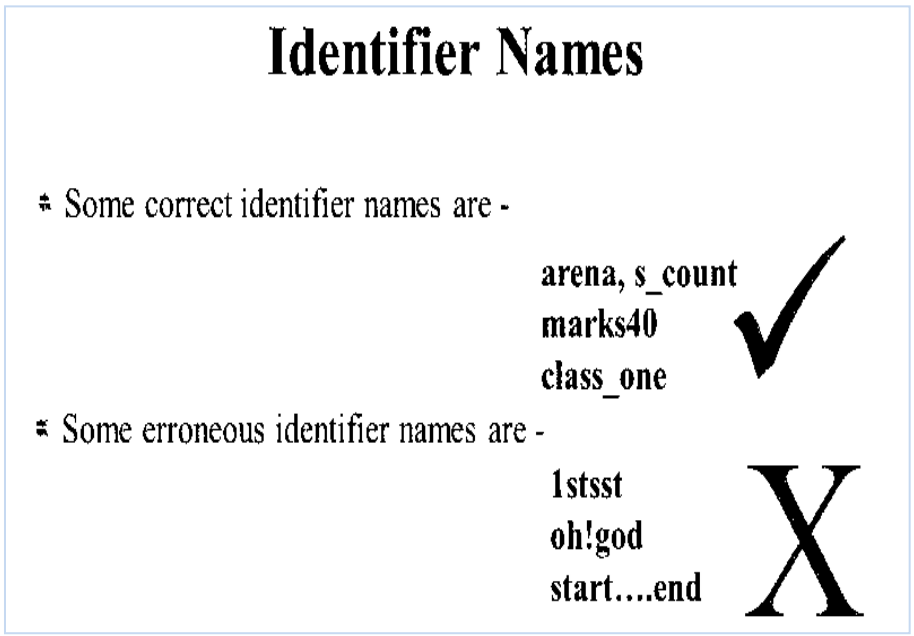
```
int money;

double accountBalance;
```

Here, *money* and *accountBalance* are identifiers.

Also remember, identifier names must be different from keywords. You cannot use *int* as an identifier because *int* is a keyword.

Rules for writing an identifier:

1. A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores.
2. The first letter of an identifier should be either a letter or an underscore. However, it is discouraged to start an identifier name with an underscore.
3. There is no rule on length of an identifier. However, the first 31 characters of identifiers are discriminated by the compiler.

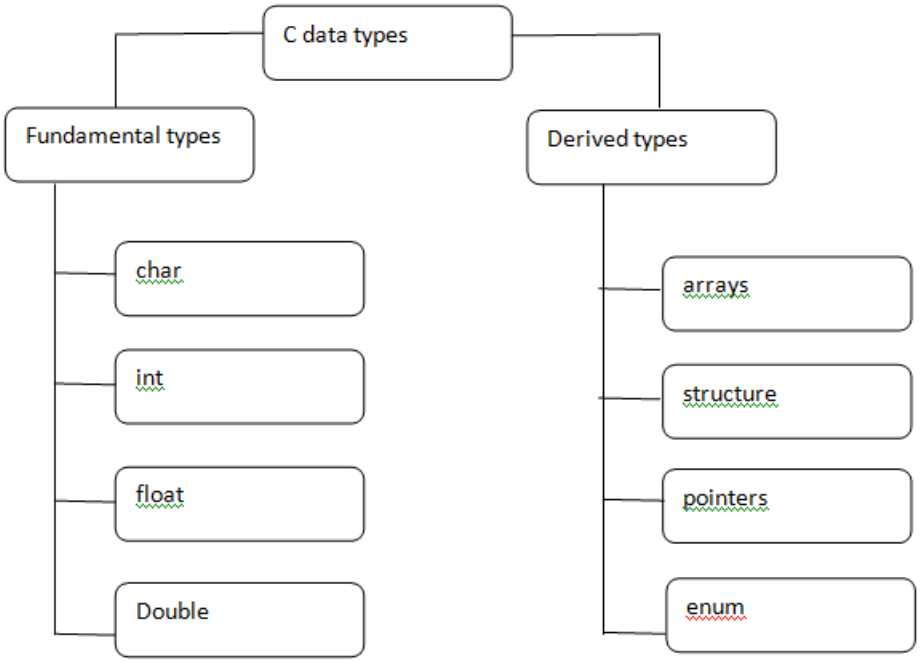


Built-in Data Types

Data types refers to the type and size of data associated with variables.

Built in data types are also known as fundamental or basic data types.

C data types can be divided into:



The four built in data types are,

1. char
2. int
3. float
4. double

char

- The variable of char data type holds a character constant. A character constant is a c character enclosed within a pair of single quotes.
- Char data type reserves 1 byte of memory for the variable storage.
- Example: char s1,s2;

  Here s1 and s2 are variables of char data type.

A character denotes any alphabet, digit or special symbol used to represent information. Below figure shows the valid alphabets, numbers and special symbols allowed in C.

Alphabets	A, B, ….., Y, Z a, b, ……, y, z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Special symbols	~ ‘ ! @ # % ^ & * ( ) _ - + = | \ { } [ ] : ; " ' < > , . ? /

Character Data Types

Type	Size (in bytes)	Range
char	1	- 128 to 127
Signed char	1	- 128 to 127
unsigned char	1	0 to 255

int

Integers are whole numbers that can have both positive and negative values but no decimal values. Example: 0, -5, 10

- A variable of int data type holds integer numbers.
- int data type reserves 2 bytes of memory for the variable storage.
- Example: int n1, n2;

Here n1 and n2 are variables of int data type.

float

Floating type variables can hold real numbers such as: 2.34, -9.382, 5.0 etc.

- A variable of float data type holds real (fractional) numbers.
- float data types reserves 4 bytes of memory for the variable storage.
- Variable of float data type holds a number with six decimal digits of precision.
- Example: float length, height;

Here length and height are variables of float data type.

Double

Double type variables can hold real numbers such as: 475.25, -78.85, 345.0 etc.

- A variable of double data type holds real or fractional numbers with high precision than float data type .
- In double data type the fractional numbers are stored in exponent form.
- Variable of double data type holds a number with ten decimal digits of precision.
- double data type reserves 8 bytes of memory for the variable storage.
- Example: double voltage, percentage;

Here voltage and percentage are variables of double data type.

Note: By default a real number is treated as a double.

Data types in C

Data Type	Memory Requirement	Range
char	1 byte	-128 to +127
short or int	2 bytes	-32768 to +32767
long	4 bytes	-2147483648 to +2147483647
float	4 bytes	-3.4e38 to +3.4e38
double	8 bytes	-1.7e308 to +1.7e308

Variable Definition, Declaration.

Variables are simply names used to refer to some location in memory – a location that holds a value with which we are working.

- Variables are those whose value may vary during program execution.

A variable declaration indicates the data type of the variables and number of memory locations required for variable storage.

Any variable to be used in the program must be declared using declaration statement.

General syntax: data_type variable_list;

The data types can be int, float, double, char etc.

Examples for variable declarations.

1. int marks1, marks2;
2. float average;

Here int and float are data types.

marks1, marks2 and average are variables.

C Key Words-Rules & Guidelines for Naming Variables.

C Keywords or reserved words.

- Keywords are those whose meaning is already defined in C compiler.
- Keywords are also called as reserved words.
- Keywords are written in lower case.
- Keywords cannot be used as identifiers. Compilers show error message when keywords are declared as variables/ constants.
- They cannot store values
- C has 32 keywords.

Example : int, float, char ,if etc.

List of 32 keywords :

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

Rules and guidelines for naming variables:

1) A variable name can contain digits, alphabets and underscores.
2) The first character must be an alphabet or underscore.
3) Maximum number of characters in variable name is 8.
4) Variable names are case sensitive.
5) Variable name must not contain any special character except underscore.
6) Variable name must not be a reserved word.

Arithmetic operators and Expressions.

An operator is a symbol which operates on a value or a variable. For example: + is an operator to perform addition.

C programming has wide range of operators to perform various operations.

Arithmetic operators

Arithmetic operators perform arithmetic operations like addition, subtraction, multiplication and division .Arithmetic operators are as shown in below.

Symbol used	Operations
+ (plus)	Adds two numbers
- (minus)	Subtract two number
* (asterisk)	Multiplies two number
/ (slash)	Divides two numbers
% (percent)	gives remainder of integer division

- These operators are called binary operators because they operate on two operands.
- +,-,* ,& ,/ operators operate on int, float and double data type variables.

The following table shows all the arithmetic operators supported by the C language. Assume variable **A** holds 10 and variable **B** holds 20, then :

Operator	Description	Example
+	Adds two operands.	A + B = 30
−	Subtracts second operand from the first.	A − B = 10
*	Multiplies both operands.	A * B = 200
/	Divides numerator by de-numerator.	B / A = 2
%	Modulus Operator and remainder of after an integer division.	B % A = 0

++	Increment operator increases the integer value by one.	A++ = 11
--	Decrement operator decreases the integer value by one.	A-- = 9

Arithmetic expressions

Arithmetic expression contains numeric variables and constants associated with arithmetic operators.

Ex; 2*x-3*y+z

Modes of Arithmetic Expressions.

There are 3 types

1. Integer mode arithmetic expressions.

2. Real (floating – point) mode arithmetic expression.

3. Mixed mode arithmetic expression.

1. Integer mode arithmetic statement /expression.

This expression contains integer operands (constants or variable). An arithmetic operation between

integer operands results in integer value only .

Ex; int A,B,Sum;

Sum=A+B;

2. Real mode arithmetic statement/expression.

These expressions contain real type (floating –point or double data type) operands. An arithmetic operation between real operands results in real value only.

Ex; float a, b, sum;

sum =a+b;

3. Mixed mode arithmetic statement/expression

These expressions contain mixture of int and real data type operands. An arithmetic operation between integer and real operands results in real value only.

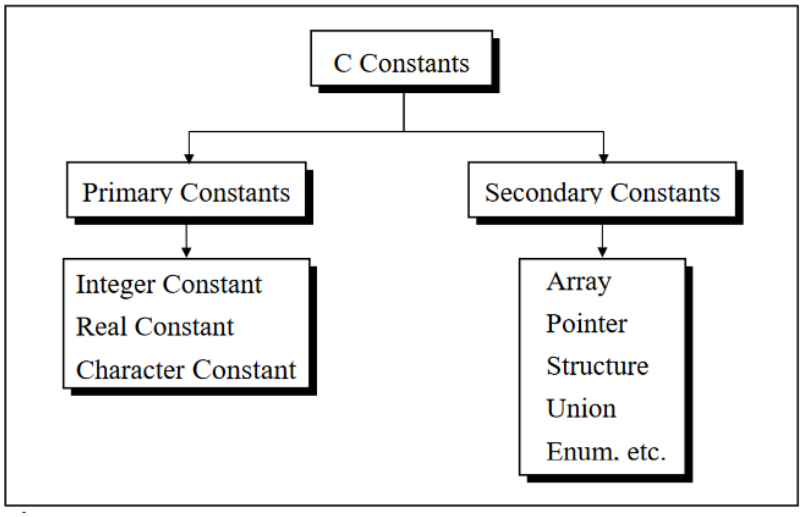
```
int total =10

float avg ,sum =55.6

avg = sum /total ;/*Avg gets real/fractional value*/
```

Constants and Literals.

Constants refer to fixed values that the program may not alter during its execution. These fixed values are also called *literals*. Constants can be of any of the basic data types like an integer constant, a floating constant, a character constant, or a string *literal*.

Types of C Constants
C constants can be divided into two major categories:
(a) Primary Constants
(b) Secondary Constants

These constants are further categorized as shown in Figure



Precedence and Order of Evaluation.

The pre-defined order in which the arithmetic operations are carried out in an arithmetic expression is called the *precedence of arithmetic operators*.

Hierarchy rules are given below.

0. Zero =>contents within parenthesis are evaluated first.
1. First => unary minus operator(-)
2. Second => multiplication, division and modulus operators(*,/,%)
3. Third => addition, subtraction operators (+,-)
4. Last => the assignment operator(=)

Priority	Operators	Description
1st	* / %	multiplication, division, modular division
2nd	+ -	addition, subtraction
3rd	=	assignment

If the expression contains the operators of same hierarchy, then the expression is evaluated from left to right.

If the expression contains many parentheses, then contents of inner most parenthesis is evaluated first, the next innermost as second step and so on.

Example:

⇨ 2 + 1 * 3 - 4 % 3 * 1 + 16 / 2 % 5
⇨ 2 + **3** - 4 % 3 * 1 + 16 / 2 % 5
⇨ 2 + 3 - **1** *1 + 16 / 2 % 5
⇨ 2 + 3 - **1** + 16 / 2 % 5
⇨ 2 + 3 - 1 + **8** % 5
⇨ 2 + 3 - 1 + **3**
⇨ **5** – 1 + 3
⇨ **4** + 3
⇨ **7**

Simple assignment statement

An **assignment statement** gives a value to a variable. For example,

```
x = 5;
```

gives x the value 5.
The value of a variable may be changed. For example, if x has the value 5, then the assignment statement

```
x = x + 1;
```

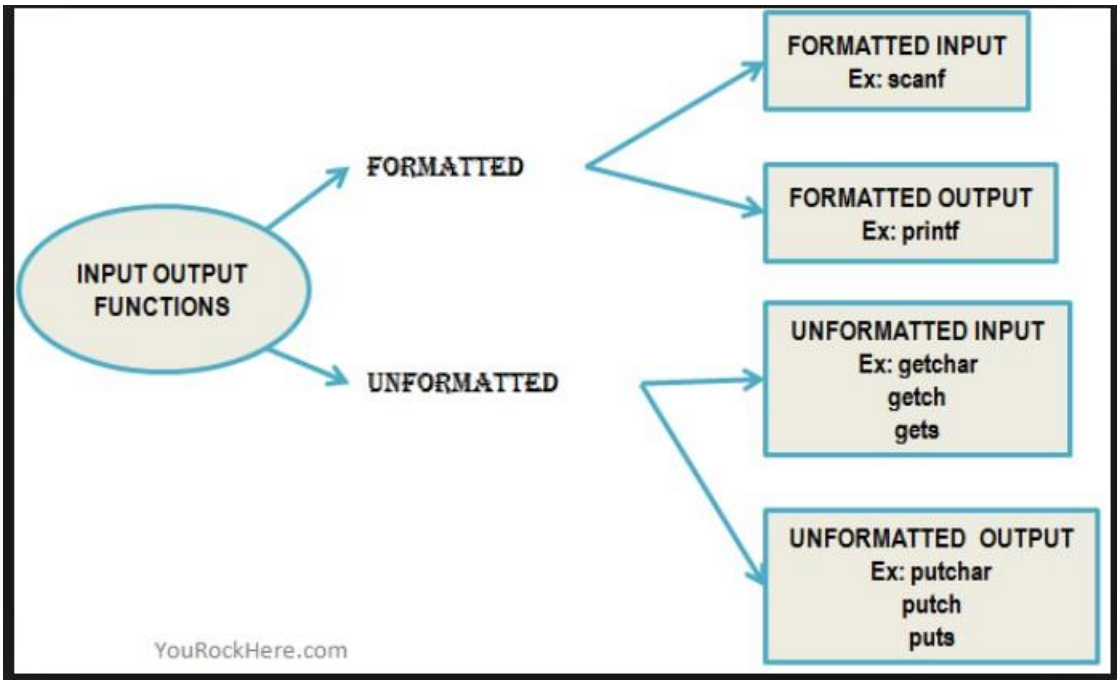
will give x the value 6.
The general syntax of an assignment statement is :

variable = expression;

Basic input/output statement

C supports standard built-in functions to read the data from input devices (like keyboard) and to display the data on output devices (like monitor).
These functions are classified as :

1. Formatted I/O functions
   scanf() and printf()
2. Unformatted I/O functions
   getch(), getche(),getchar(),gets(), putch(),putchar(),puts().



Format specifier/ Conversion specifiers:

Format specifiers indicate the data type of the value to be read or displayed when used in formatted I/O functions (scanf() and printf()).

The list of format specifiers and associated data types are listed below.

Format specifier	Data type
%d	decimal integer(signed)
%f	floating-point
%c	character constant
%s	string constant

%u	unsigned decimal integer
%o	unsigned octal number
%x	unsigned hexadecimal number
%ld	long decimal number
%e	Floating point number in exponent form

Formatted I/O function

These use format specifiers like (%d,%f,%c) to specify the data type of the variable to be read or displayed. scanf() and printf() are formatted I/O functions.

scanf() function:

- It is an input function
- It is used to read the value from keyboard.
- scanf stands for scan formatted or scan function.

The general form or general syntax

scanf("control string",arg1,arg2,arg3,……argn);

As shown above scanf() function contains two parts :control string and argument list

Control string:

It contains format specifiers to indicate the data types of the values to be read. These format specifiers are %d,%f,%c etc.

Format specifiers are always included within double quotes.

Argument list

This contains variables names into which the read values are stored.

The data type of format specifiers and variables must match.

Example:

1)scanf("%d", &num1);

Here one integer is read from keyboard and stored in the variable num1.

2)scanf("%d%d",&a,&b);

Here two integer values are read and stored in the variables a and b.

printf() function

- It is an output function
- It is used to display the values or text message on monitor.
- printf stands for print formatted or print function.

The general form or general syntax

printff("control string",arg1,arg2,arg3,……argn);

As shown above printf() function contains two parts :control string and argument list

Control string:

It may contain format specifiers or any text message or both. These format specifiers are %d,%f,%c etc.

Format specifiers and text messages are always included within double quotes.
Argument list
This contains variables names whose values are to be displayed.
The data type of format specifiers and variables must match.

Example:
1)printf("Hardwork is the key to success");

output
Hardwork is the key to success
2) float a=56.565
Printf(" The area of the circle is=%f",a);
output
The area of the circle is=56.565000

Unformatted I/O function

These do not use format specifiers .The default data type is char. These are used for reading or displaying the value of character constant or string constant. Examples: getch(), getche(), getchar()/fgetchar(), putch() and puts().

getch()

- It is an unformatted input function.
- It is used to read a character constant from keyboard.
- It does not echo typed character on the screen.

```
Ex:  char p;
     p=getch();
```

getche()

- It is an unformatted input function.
- It is used to read a character constant from keyboard.
- It echoes typed character on the screen.

```
Ex:  char p;
     p=getche();
```

getchar/fgetchar()

- It is an unformatted input function.
- It is used to read a character constant from keyboard.
- It echoes typed character on the screen.

```
Ex:  char p;
     p=getche();
```

gets()

- It is an unformatted input function.
- It is used to read a character string from keyboard.
- End of the string is indicated by pressing ENTER key

putch()

- It is an unformatted output function.
- It is used to display a character constant on monitor.

Example: putch(„g");/* it will display ***g*** on the monitor*/

puts()

- It is an unformatted output function
- It is used to display a sequence of characters(string) on monitor.

Example: puts(" Hi! Good morning");
It will display ***Hi! Good morning*** on the monitor

Simple 'C' programs.

1. /*Program to find sum of two variables*/

```
#include<stdio.h>
#include<conio.h>
 void main()
{
     int a,b,c;                                          //declaration of variables
     printf("enter 1st integer value\n");
     scanf("%d",&a);
     printf("enter 2nd integer value\n");
     scanf("%d",&b);
     c=a+b;
     printf("sum of given values is %d\n" , c);
     getch();
}
```

Output:
enter 1st integer value
10
enter 2nd integer value
20
sum of given values is 30

2. /*Program to swap the values of 2 variables without using third variable*/

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

```
void main()
{
clrscr();
int A, B;
printf("Enter value of A and B \n");
scanf("%d%d", &A,&B);
printf(“Before swapping A=%d and B=%d \n”,A,B);
A=A+B;     /*formula*/
B=A-B;
A=A-B;
printf("After swapping A=%d and B=%d \n", A, B);
getch();
}
```

Output

```
Enter value of A and B
-34
47
Before swapping A=-34 and B=47
After swapping A=47 and B=-34
```

3.write a C program to read 3 integers and to print sum and average of three integer numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
int a,b,c,sum;
float average;
printf("Enter any 3 integer numbers\n");
scanf("%d%d%d", &a,&b,&c);
sum=a+b+c;
average=sum/3;
printf(“The sum of 3 numbers =%d”,sum);
printf(“The average of three numbers=%f”,average);
getch();
}
```

Output

```
Enter any 3 integer numbers
5
20
12
The sum of 3 numbers =37
```

The average of three numbers=12.333333

4.Write a C language to accept temperature in Fahrenheit and to convert it in to centigrade.
Formula: C=(F-32)/1.8

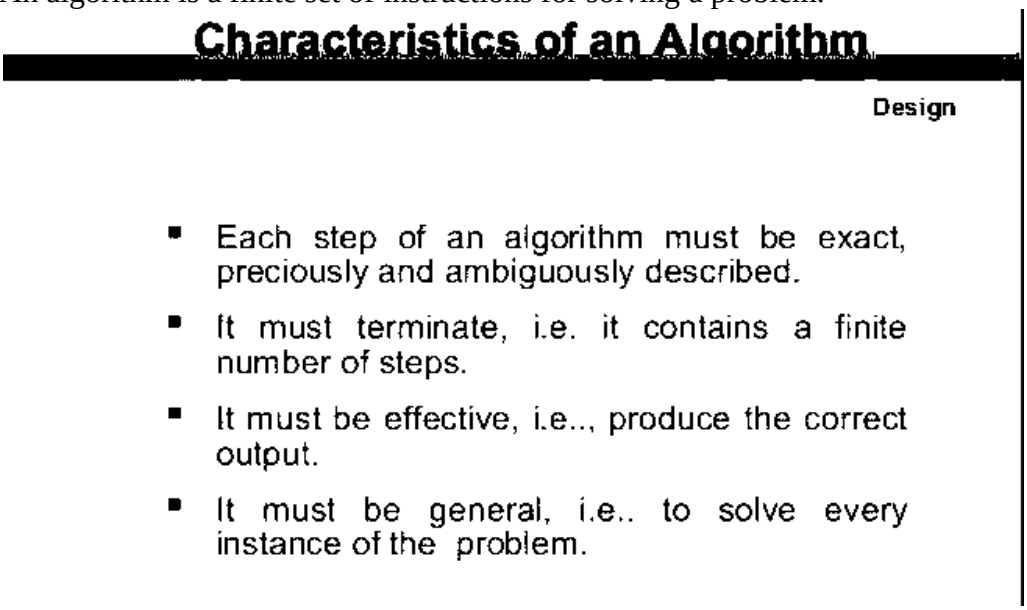
```
#include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
float fah_temp,cent_temp;
printf("Enter the temperature value in Fahrenheit \n");
scanf("%f", &fah_temp);
cent_temp=(fah_temp-32)/1.8;
printf("Equivalent temperature value in centigrade =%f",cent_temp);
getch();
}
```

Output

```
Enter the temperature value in Fahrenheit
68.0
Equivalent temperature value in centigrade =20.000000
```

Algorithms – Definition and Characteristics.

An algorithm is a finite set of instructions for solving a problem.



Simple algorithms.

1. Algorithm to find sum of 2 numbers.

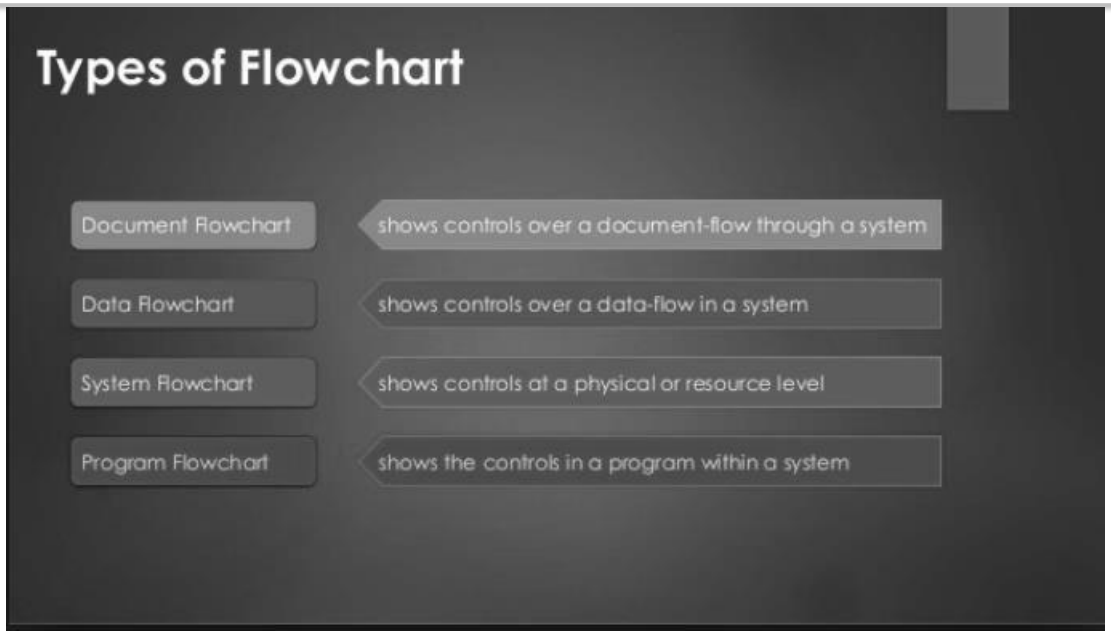
   Step 1. Start
   Step 2. Read n1.
   Step 3. Read n2.
   Step 4. Sum=n1+n2.
   Step 5. Print sum.
   Step 6. Stop.

2. Algorithm to find sum and average of 3 numbers.

   Step 1. Start
   Step 2. Read n1,n2,n3.
   Step 3. Sum=n1+n2+n3.
   Step 4. avg=(n1+n2+n3)/3.
   Step 5. Print sum.
   Step 6. Print avg.
   Step 7. Stop.

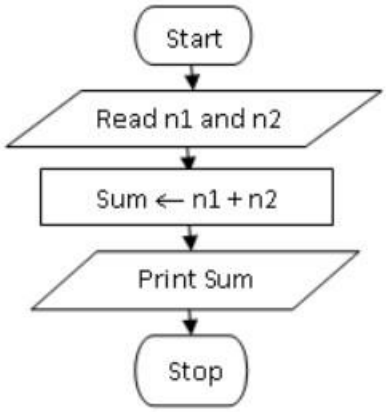
Flow chart – Type of flow chart.

A flowchart is a pictorial representation of an algorithm in which steps are drawn in the form of different shapes of boxes and the logical flow is indicated by interconnecting arrows.

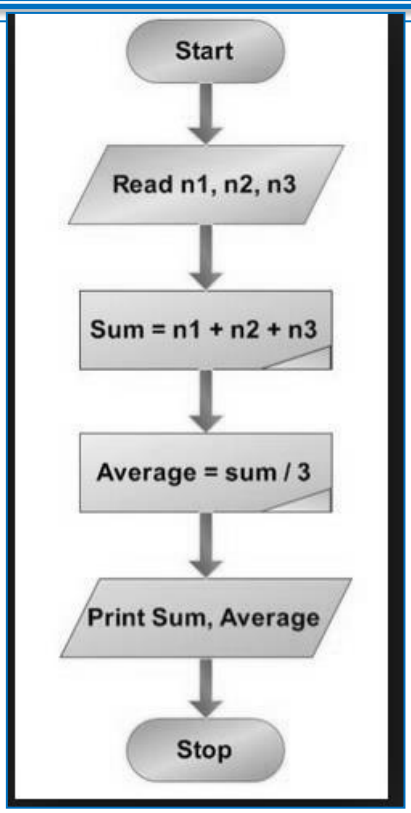


1.15. Simple flow charts

Flowchart to find sum of two numbers.



Flowchart to find sum and average of three numbers.


Start
Read n1, n2, n3
Sum = n1 + n2 + n3
Average = sum / 3
Print Sum, Average
Stop

UNIT -2 : Decision making- Branching and Looping

C Operators

1. Conditional Operators
2. Relational Operators
3. Logical Operators
4. Assignment Operators
5. Increment and Decrement Operators
6. Bitwise Operators
7. Special Operators
8. Arithmetic operators

Conditional Operators: A ternary operator pair " ? : " is used to construct conditional expressions of the form

exp1 ? exp2 : exp3

where exp1,exp2 and exp3 are expressions.

The ? : Operators works as follows

- exp1 is evaluated first.
- If it is true, then the exp2 is evaluated and becomes the value of the expression.
- If exp1 is false then exp3 is evaluated and becomes the value of the expression.

Example: a=10, b=15;

x = (a>b) ? a : b;

In this example x is assigned the value of b expression a>b is evaluated to false.