Procedural Programming – I

UNIT – I

How is the World Powered by C?

In today's world, almost everything is powered by computers. Computers are integral to our lives, from the smallest electronic devices to the largest supercomputers. And while there are many types of computers, they all share one thing in common: they're powered by the C programming language.

C is a versatile language that can create all sorts of applications. It's used to write the operating system for many of the world's most popular computers and the software that runs on them. It's also used to create the websites and apps we use daily.

But C isn't just used for computers, it's also used to control the devices that we use in our everyday lives, from cell phones to microwaves. It's estimated that over 90% of the world's electronic devices are powered by C.

So next time you're using your computer, or even just flipping a switch, remember that you're using the power of C.

The C language is a high-level, general-purpose programming language. It provides a straightforward, consistent, powerful interface for programming systems. That's why the C language is widely used for developing system software, application software, and embedded systems.

The C programming language has been highly influential, and many other languages have been derived from it. For example, C++ and Java are two popular modern dialect of C.

And C is an excellent choice for system programming, for example, developing operating systems, compilers, and network drivers. Despite its popularity, C is not without its criticisms. Some have argued that its syntax could be more complex and easier to learn, while others have noted its lack of standardization as a significant issue. Nevertheless, C remains a widely used and influential language and will probably continue for many years.

Applications of C

Real Life Applications of C Programming with Examples

In this Tutorial you will learn about the applications of C Programming in various different fields.

1. **Operating systems:** C is often used to develop operating systems, such as Windows, Linux, and MacOS.
2. **Embedded systems:** C is often used to program microcontrollers and other types of embedded systems.
3. **Games:** Many game engines and video games are written in C or C++ (a closely related language).
4. **Compilers:** C is often used to develop compilers for other programming languages.
5. **Networking:** C is commonly used to write network drivers and other networking software.
6. **Databases:** C is used in the development of many database systems, such as MySQL and PostgreSQL.
7. **Graphics:** C is used in the development of many graphics libraries, such as OpenGL and DirectX.
8. **Word processors:** C is used in the development of many word processors, such as Microsoft Word and LibreOffice.
9. **Spreadsheets:** C is used in the development of many spreadsheet programs, such as Microsoft Excel and Google Sheets.
10. **Mobile apps:** Many mobile apps are developed using C or C++, especially those that need to perform resource-intensive tasks or work with low-level hardware features.
11. **Financial software:** C is often used in the development of financial software, such as stock trading platforms and banking systems.
12. **Medical devices:** C is used to program many types of medical devices, such as X-ray machines and glucose monitors.
13. **Industrial control systems:** C is used to develop software for industrial control systems, such as those used to control factory machinery or assembly lines.
14. **Aerospace and defense:** C is used in the development of many types of aerospace and defense systems, such as aircraft and spacecraft control systems.

15. **Cryptocurrency:** C is used in the development of some cryptocurrency systems, such as the Bitcoin core reference implementation.

Introduction to C :-

C language was developed in the **1972s** by **Dennis Ritchie**, who was an American computer scientist at **AT & T Bell Labs in USA**.

C language was created as a replacement for the 'B' language, which was used for writing operating systems.

C language was designed to be a general-purpose, high-level programming language that is both powerful and efficient.

C language is often used for system programming and embedded systems development, as it provides low-level access to memory and allows for efficient manipulation of hardware resources.

C language has become one of the most widely used programming languages, and it is used for a variety of applications, including operating systems, device drivers, and embedded systems.

C is mid-level programming language (C is considered as a middle-level language because it supports the feature of both low-level and high-level languages)
C is procedural language (A procedure is known as a function, method, routine, subroutine, etc. A procedural language specifies a series of steps for the program to solve the problem.

A procedural language breaks the program into functions, data structures, etc.)
C as a structured programming language (Structure means to break a program into parts or blocks so that it may be easy to understand.)

C programming language is a machine-independent programming language that is mainly used to create many types of applications and operating systems such as Windows
The UNIX OS was totally written in C.

History of C language

The C programming language was created at Bell Laboratories in the early 1970s, mainly by Ken Thompson and Dennis Ritchie. For the UNIX operating system, which at the time required applications to be written in assembly language, programmers needed a more user-friendly set of instructions. Assembly programmes, which communicate directly with a computer's hardware, are lengthy and complex to debug, and adding new functionality requires a lot of time and effort. Thompson's first high-level language was named B after the BCPL system programming language on which it was built. Thompson rewrote B to better match the demands of the modern time. As a result C, the B's successor, was created. By 1972, C had matured to the point that it could be used to rewrite the UNIX operating system.
Other programmers needed documentation that detailed how to use C before it could be utilized effectively outside of Bell Labs. In 1978, Brian Kernighan and Dennis Ritchie's book "The C Programming Language," sometimes known as K & R or the "White Book" by C enthusiasts, became the canonical source for C programming. The second edition of K & R, first published in 1988, is still commonly accessible as of this writing. Based on the book, the first, pre-standard version of C is known as K & R C.
Throughout the 1980s, C developers sought to build standards for the language in order to prevent others from developing their own dialects. The American National Standards Institute (ANSI) standard X3.159-1989 became the official U.S. standard for C in 1989. In 1990, the International Organization for Standardization (ISO) issued the ISO/IEC 9899:1990 standard. These standards, as well as their later updates, are referenced in C versions after K&R. (C89, C90 and C99).
The 1980s saw a surge in operating system development, with C and its use in UNIX being only such instances. Despite its advancements over its predecessors, C was still difficult to use for creating larger software programmes. As computers got more powerful, there was a growing demand for a more user-friendly programming environment. This desire pushed programmers to

use C to create their own compilers and, as a result, new programming languages. These new languages may make it easier to code complex operations with many moving elements. For example, object-oriented programming, a programming method that maximizes a programmer's ability to reuse code, was eased by languages like C++ and Java, both of which were derived from C.

Features of C:-

C is a general-purpose computer programming language for system administration, network programming, and embedded software.

It has several features that make it desirable for these applications:

1. It has an easy-to-learn and simple syntax.
2. C is a structured and procedural programming language.
3. There are many libraries and header files.
4. C **provides a lot of inbuilt functions** that make the development fast.
5. It is a compiler type of language translator.
6. It is case-sensitive.
7. All types of expressions can be written by using operators.
8. Better programming controls and No programming limitations.
9. C programs are relatively short compared to other languages, which reduces the time needed to complete them.
10. C is a powerful programming language that enables developers to create sophisticated software systems.
11. C language is extensible because it **can easily adopt new features**.
12. Recursion-In C, we **can call the function within the function**. It provides code reusability for every function.

Structure of C program

Include header file section

Global declaration section

main()

{

Declaration part;

Executablae part;

}

User defined functions

{

Statements;

}

Include header file section

C program depends upon some headers files for function definition that are used in program. Each header file should be included using #include directive as given here extended with .h.

Examples-

1) stdio.h : All the standard input/output functions are defined

Eg-- printf(), scanf()

2) conio.h. : consol input/output function

Eg.—clrscr(). getch()

To include the header file in the program #include statement as-

Syntax - #include<headerfiliname>

Eg- #include<stdio.h>

#include<conio.h>

Global declaration-

This section declares some variables that are used in more than one function. These variables are known as global variables. This section must be declared outside of all the functions.

Main() function-

Every program written in c language must contain main() is a starting point of every c program.

The execution of the program always begins with function main()

After linking and declaration satements can start your program with main function.

Declaration part-(local variable)

The declaration part declares the entire variables that are used in executable part. The initialization of variables are also done in this section.

Initialization means providing initial value to the variables.

Local variables are the one that are declared inside a function. They are accessible to the function in which they are defined.

Executable part-(input statement)

This part contains the statement following the declaration of variable .

This part contains a set of statements or a single statement.

These statements are enclosed between the braces.

Input statement can be given to input or accept the data from user. For this c provides standard input function: scanf()

User defined function-

The function defined by the user are called user defined function.

They can be included as per the program requirement .

Output statement-

Finally result will be displayed on screen using the output statement (function): printf().

- **Print()-** is a input function used to print the output on screen.
  Eg- printf("Welcome to c programming");

  It will be print the output-
  Welcome to c programming

- **Scan()-** is used to read the values of a variables.(reading data from keyboard)

  Syntax- scanf("format specifier", &variable name);

  Printf() & scanf() functions are defined in header files stdio.h. and whenever we are using them program, we must include stdio.h preprocessor.

  Example-

```
/* first program in c*/

#include<stdio.h>

#include<conio.h>

Void main()

{

clrscr();

printf("hello world");

getch();

}
```

In c we can write a comment in program using /*comment */ , compiler will ignore this step as it is a comment not a programming instruction.

Every statement in c has to be end by **;** called as terminator operator.

Getchar()-
Getchar is a standard library function that takes a single input character from standard input.

Basic Elements
Character set of C:-

The character set in c used to represent information.
It is used to specify the acceptable characters that can be inserted into the source code or interpreted while the programme is executing.
In C, the character set is applied in the following ways:

- Character representation: The character set represents characters in the source programme. The ASCII code 65, for instance, is used to represent the letter A.

- To represent special symbols: In the source programme, special symbols are represented by the character set. For instance, addition in C is denoted by the symbol +.

- To create words and phrases: In the source programme, words and expressions are created using the character set. For instance, the letters h, e, l, and o combine to make the word hello.

There are four type of character set in C programming.

- Alphabets
- Digit
- Special Character
- White space

Alphabets
Alphabets has a two case.

1. **Upper case** A-Z
2. **Lower case** a-z.

In C programming, small letters and capital letters are distinct.

Alphabets	***ASCII Code***	***Alphabets***	***ASCII Code***
A	65	a	97
B	66	b	98
C	67	c	99
D	68	d	100
E	69	e	101
F	70	f	102
G	71	g	103
H	72	h	104
I	73	i	105
J	74	j	106
K	75	k	107
L	76	l	108
M	77	m	109
N	78	n	110
O	79	o	111
P	80	p	112
Q	81	q	113
R	82	r	114
S	83	s	115
T	84	t	116
U	85	u	117
V	86	v	118
W	87	w	119
X	88	x	120
Y	89	y	121
Z	90	z	122

Digits
C programming support 10 digit which are used to construct numerical value.

- **Digit** 0-9.

Alphabets	***ASCII Code***
0	48
1	49
2	50
3	51
4	52
5	53
6	54
7	55
8	56
9	57

Special Character
C programming supports set of special symbols that include symbols to perform mathematical operation, check condition and other special symbol.

- **Special character** %,&,*,etc.

Special Character	***ASCII Code***	***Special Character***	***ASCII Code***
~	126	–	45
!	33	_	95
@	64	+	43
#	35	=	61
$	36	{	123
%	37	}	125
&	38	|	124
*	42	\	92
(	40	;	58
)	41	:	59
‘	39	/	47
“	34	?	63
<	60	>	62

White space
In computer programming, white space is a character that represent vertical space in typography.

- **White space** \b,\v,\? and so on.

White Space	***Meaning***	***White Space***	***Meaning***
\b	blank space	\\	back slash
\t	horizontal tab	\’	single quote
\v	vertical tab	\”	double quote
\r	carriage return	\?	Question mark
\f	form feed	\0	null
\n	new line	\a	alarm (bell)

Importance of Character Set in C

The importance of the Character set in C are:

- Character set in C allows programmers to compose the program statements.
- The Character set in C supports different languages and scripts, thus allowing programmers to work with multilingual text data.
- Character set conversion is very useful for reading and writing different text files.
- Communication between different networks involves dealing with different character sets. Thus charset conversion plays an important part here for encoding and decoding purposes.

C Program to print Characters from 0-9 using ASCII value

Code:

```
#include <stdio.h>
int main()
{
   int j;
   printf("ASCII ==> character\n");
   // loop to print all ASCII values from 48 to 57
   for(int j = 48; j <= 57; j++)
      // display the ASCII value to the corresponding j value
      printf("%d ==> %c\n", j, j);
   return 0;
}
```

Output:

ASCII ==> character

48 ==> 0

49 ==> 1

50 ==> 2

51 ==> 3

52 ==> 4

53 ==> 5

54 ==> 6

55 ==> 7

56 ==> 8

57 ==> 9

C Program to print Characters from A-Z using ASCII value

Code:

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

```
int main()
{
   int j;
   printf("ASCII ==> character\n");
   // loop to print all ASCII values from 65 to 90
   for(int j = 65; j <= 90; j++)
      // display the ASCII value to the corresponding j value
      printf("%d ==> %c\n", j, j);
   return 0;
}
```

Output:

ASCII ==> character

65 ==> A

66 ==> B

67 ==> C

68 ==> D

69 ==> E

70 ==> F

71 ==> G

72 ==> H

73 ==> I

74 ==> J

75 ==> K

76 ==> L

77 ==> M

78 ==> N

79 ==> O

80 ==> P

81 ==> Q

82 ==> R

83 ==> S

84 ==> T

85 ==> U

86 ==> V

87 ==> W

88 ==> X

89 ==> Y

90 ==> Z

C Program to print Characters from a-z using ASCII value

Code:

```
#include <stdio.h>

int main()

{

   int j;

   printf("ASCII ==> character\n");

   // loop to print all ASCII values from 97 to 122

   for(int j = 97; j <= 122; j++)

      // display the ASCII value to the corresponding j value

      printf("%d ==> %c\n", j, j);

   return 0;

}
```

Output:

ASCII ==> character
97 ==> a
98 ==> b
99 ==> c
100 ==> d
101 ==> e
102 ==> f
103 ==> g
104 ==> h
105 ==> i
106 ==> j
107 ==> k

108 ==> l
109 ==> m
110 ==> n
111 ==> o
112 ==> p
113 ==> q
114 ==> r
115 ==> s
116 ==> t
117 ==> u
118 ==> v
119 ==> w
120 ==> x
121 ==> y
122 ==> z

Special Symbols

The following special symbols are used in C having some special meaning and thus, cannot be used for some other purpose. Some of these are listed below:

Brackets[]: Opening and closing brackets are used as array element references. These indicate single and multidimensional subscripts.

Parentheses(): These special symbols are used to indicate function calls and function parameters.

Braces{}: These opening and ending curly braces mark the start and end of a block of code containing more than one executable statement.

Colon(:): It is an operator that essentially invokes something called an initialization list.

Semicolon(;): It is known as a statement terminator. It indicates the end of one logical entity. That's why each individual statement must be ended with a semicolon.

Asterisk (*): It is used to create a pointer variable and for the multiplication of variables.

Assignment operator(=): It is used to assign values and for logical operation validation.

Pre-processor (#): The preprocessor is a macro processor that is used automatically by the compiler to transform your program before actual compilation.

Period (.): Used to access members of a structure or union.

Tilde(~): Bitwise One's Complement Operator.

Square brackets [ ]: The opening and closing brackets represent the single and multidimensional subscripts.

Comma (,): It is used for separating for more than one statement and for example, separating function parameters in a function call, separating the variable when printing the value of more than one variable using a single printf statement.

Tokens in C

A token in C can be defined as the smallest individual element of the C programming language that is meaningful to the compiler. It is the basic component of a C program.

Tokens in C is the most important element to be used in creating a program in C. We can define the token as the smallest individual element in C. For `example, we cannot create a sentence without using words; similarly, we cannot create a program in C without using tokens in C. Therefore, we can say that tokens in C is the building block or the basic component for creating a program in C language.

1. C Token – Keywords

The keywords are pre-defined or reserved words in a programming language. Each keyword is meant to perform a specific function in a program. Since keywords are referred names for a compiler, they can't be used as variable names because by doing so, we are trying to assign a new meaning to the keyword which is not allowed. You cannot redefine keywords. However, you can specify the text to be substituted for keywords before compilation by using C preprocessor directives. C language supports 32 keywords which are given below:

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**	**volatile**
do	**if**	**static**	**while**

2. C Token – Identifiers

Identifiers are used as the general terminology for the naming of variables, functions, and arrays. These are user-defined names consisting of an arbitrarily long sequence of letters and digits with either a letter or the underscore(_) as a first character. Identifier names must differ in spelling and case from any keywords. You cannot use keywords as identifiers; they are reserved for special use. Once declared, you can use the identifier in later program statements to refer to the associated value. A special identifier called a statement label can be used in goto statements.

Identifiers are used for naming variables, functions, arrays, structures, etc. Identifiers in C are the user-defined words. It can be composed of uppercase letters, lowercase letters, underscore, or digits, but the starting letter should be either an underscore or an alphabet. Identifiers cannot be used as keywords. Rules for constructing identifiers in C are given below:

Rules for Naming Identifiers

Certain rules should be followed while naming c identifiers which are as follows:

- They must begin with a letter or underscore(_).
- They must consist of only letters, digits, or underscore. No other special character is allowed.
- It should not be a keyword.
- It must not contain white space.
- It should be up to 31 characters long as only the first 31 characters are significant.
- The first character of an identifier should be either an alphabet or an underscore, and then it can be followed by any of the character, digit, or underscore.
- It should not begin with any numerical digit.
- In identifiers, both uppercase and lowercase letters are distinct. Therefore, we can say that identifiers are case sensitive.
- Commas or blank spaces cannot be specified within an identifier.
- The length of the identifiers should not be more than 31 characters.
- Identifiers should be written in such a way that it is meaningful, short, and easy to read.

C Token--Constants and Variables

- The constants in C are the read-only variables whose values cannot be modified once they are declared in the C program. The type of constant can be an integer constant, a floating pointer constant, a string constant, or a character constant. In C language, the const keyword is used to define the constants.
- A constant in C is a user-assigned name to a location in the memory, whose value cannot be modified once declared. This is in contrast to a variable in C, which is also a named memory location, however whose value may be changed during the course of the code.
- The constants refer to the variables with fixed values.
- For example, the value of mathematical constant PI is a high-precision floating point number 3.14159265359, and if it is likely to appear frequently, it is declared as a constant and used by its name.

For example- int a;

Int a,b,c; etc

Initialize and declared by-

Type variable_name=value;

Advantages of C Constants:

1. There are several advantages of C Constants. Some main advantages of C Constants are as follows:
2. Programmers may use constants to provide names that have meaning to fixed numbers, which makes the code simpler to comprehend and update.
3. Constants assist in avoiding the usage of magic numbers, which are hard-coded values, in the code. Instead, constants offer named representations of such values, enhancing the code's readability.
4. Constants are reusable throughout the program, allowing for constant values in various locations and lowering the possibility of errors brought on by typos or inconsistent values.
5. Calculations or processes inside the program can be optimized by using certain constants, such as mathematical or physical constants.
6. A constant is a value or variable that can't be changed in the program, for example: 10, 20, 'a', 3.4, "c programming", etc.

Constant refers to a value of a variable which does not change throughout the program.

List of the types of constants used in C language:

Type of Constant	**Example**
Floating-point constant	3.14, 23.33, 52.23, etc.
Integer constant	10, 53, 26, etc.
Hexadecimal constant	0x9x, 0x3, 0X8, etc.
Octal constant	022, 019, 07, 0121, etc.
String constant	“Python”, “Program”, “C”,etc.
Character constant	‘a’, ‘b’, ‘c’, etc.

Strings in C

Strings in C are always represented as an array of characters having null character '\0' at the end of the string. This null character denotes the end of the string. Strings in C are enclosed within double quotes, while characters are enclosed within single characters. The size of a string is a number of characters that the string contains.

Now, we describe the strings in different ways:

char a[10] = "javatpoint"; // The compiler allocates the 10 bytes to the 'a' array.

char a[] = "javatpoint"; // The compiler allocates the memory at the run time.

char a[10] = {'j','a','v','a','t','p','o','i','n','t','\0'}; // String is represented in the form of characters.

❖ **Data-Types**

While declaring the variable we must have to specify the datatype.
It determines the type of value the variable is gong to hold like numeric, character,etc.
One of the feature of c is that c has a variety of data-types.
There are two types of data-types.
1) Built in datatypes
2) User defined datatypes

*** The standard built in datatypes**

1. Integer
2. float
3. character

*** User defined datatypes**

1. Array
2. Structure
3. Unions

Built in data types are again devided into following subtypes-

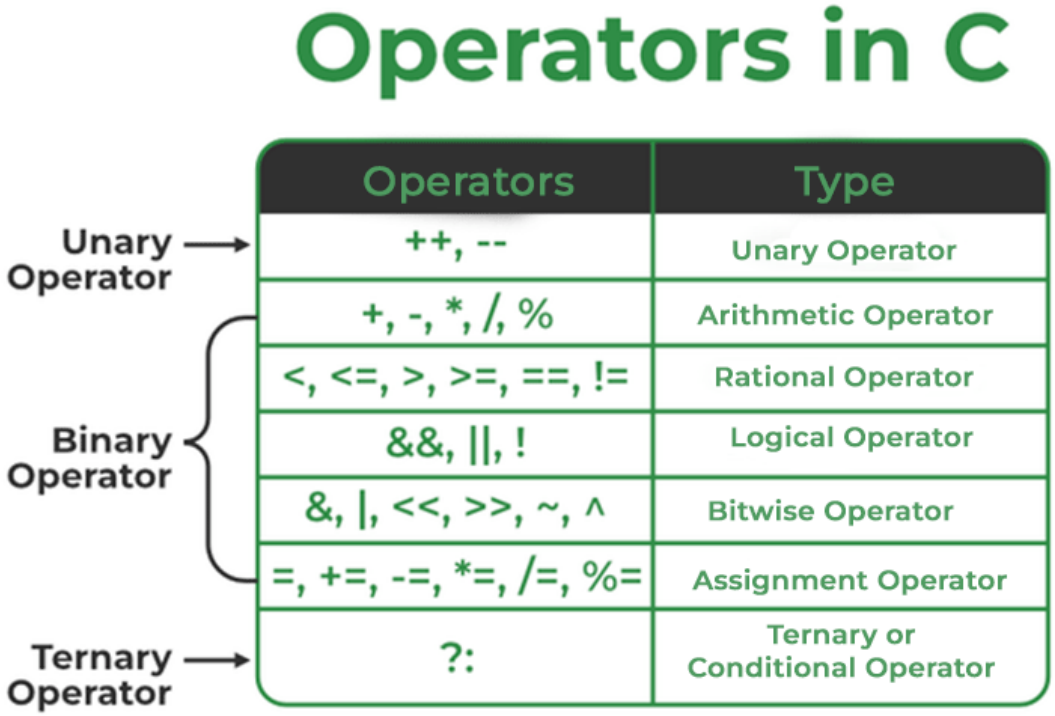
Integer	**Character**	**Float**
Integer Short integer Long integer	Single character String	Float Double foat

Operators in C

In C language, operators are symbols that represent operations to be performed on one or more operands. They are the basic components of the C programming. In this article, we will learn about all the built-in operators in C with examples.

What is a C Operator?

An operator in C can be defined as the symbol that helps us to perform some specific mathematical, relational, bitwise, conditional, or logical computations on values and variables. The values and variables used with operators are called operands. So we can say that the operators are the symbols that perform operations on operands.



For example,

```
c = a + b;
```

Types of Operators in C

C language provides a wide range of operators that can be classified into 6 types based on their functionality:

1. Arithmetic Operators

These are operators which performs the arithmetic operations like Addition, substraction, etc. hence these operators are known as arithmetic operators.

If A=10 & B=5

Operators	**Associativity**	**Example**	**Result**
Addition	Left to right	A+B	15
Substraction	Left to right	A-B	5
Multiplication	Left to right	A*B	50
Division	Left to right	A/B	2
modulus	Left to right	A%B	0

2. Relational Operators

Relational operators are also called comparision operator because relational campare 2 values & depending upon their relation, take certain decisions.
We can compare the greatest of 2-numbers, cost of 2 items, or age of 2 persons, etc.

Operators	**Meaning**	**Associativity**	**Example**
<	Less than	Left to right	A<B
<=	Less than or equal to	Left to right	A<-B
>	Greater than	Left to right	A>B
>=	Greater than or equal to	Left to right	A>=B
==	Equals to	Left to right	A==B

!=	Not equals to	Left to right	A!=B

3. Logical Operators

The logical operators are used when we want to test or check more than one conditions & make decisions.

Logical symbol	Meaning
&&	Logical AND
||	Logical OR
!	NOT

a] logical AND-

Definition- it is true if all the inputs are true otherwise it is false

Input		Output
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

b] logical OR-

It is false if and only if all the inputs are false otherwise it is true.

Input		Output
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

c] logical NOT-

I negates the input that is output will totally apposite of the input.
If input is zero output will be one and if input is one output will be zero.

Input	Output
0	1
1	0

4. Bitwise Operators

These operators work like bitwise. It first convert number into bits & then perform or work on it.

Operator	Names
&	Bitwise AND
|	Bitwise OR
^	Bitwise X'OR
<<	Left shift
>>	Right shift
~	Bitwise NOT

5. Assignment Operators

Assignment Operator is used to assign the value of constant, variable or result of an expression on its right to a variable on left.
Its also called compound assighnment operator.
You can modify the value of an existing variable using assghnment operator.

Operator	Expression	Normal expression
+=	A+=5	A=A+5
-=	A_=5	A=A-5
*=	A*=5	A=A*5
/=	A/=5	A=A/5
%=	A%=5	A=A%5

6. Conditional Operators

Question mark (?) and colon(:) are called as conditional operator
Example – int a=10; b=5;c
C=(a>b)?a:b;

Output-
C=10