ISM

We inspire you to learn

C

Premier Embedded Training Centre in the country

CLASSIFICATION OF OPERATORS

ARITHMETIC OPERATORS

+	ADDITION OR UNARY PLUS
-	SUBTRACTION OR UNARY MINUS
*	MULTIPLICATION
/	DIVISION
%	MODULO DIVISION

RELATIOINAL OPERATORS

ISM
We inspire you to learn

<	IS LESS THAN
<=	IS LESS THAN OR EQUAL TO
>	IS GREATER THAN
>=	IS GREATER THAN OR EQUAL TO
==	IS EQUAL TO
!=	IS NOT EQUAL TO

LOGICAL OPERATORS

&&	LOGICAL AND
||	LOGICAL OR
!	LOGICAL NOT

ASSIGNMENT OPERATORS

```
x+=y+1;
        x=x+(y+1);
a+=1;
        a=a+1;
a%=b;
        a=a%b;
```

INCREMENT AND DECREMENT OPERATORS

++a	pre increment operator
a++	post increment operator
--a	pre decrement operator
a--	post decrement operator

CONDITIONAL OPERATORS

```
x=(a>b)?a:b;
if(a>b)
x=a;
else
x=b;
```

BITWISE OPERATOR

ISM
We inspire you to learn

&	BITWISE AND
|	BITWISE OR
^	BITWISE EX-OR
<<	SHIFT LEFT
>>	SHIFT RIGHT
~	ONE’S COMPLEMENT

SPECIAL OPERATORS

,	COMMA OPERATOR
sizeof	SIZE OF OPERATOR
& AND *	POINTER OPERATOR
. AND ->	MEMBER SELECTION OPERATOR

PRECEDENCE OF ARITHMETIC OPERATOR

An arithmetic expression without parentheses will be evaluated from left to right using the rules of precedence of operators

* / %	highest priority
+ -	lowest priority

ISM
We inspire you to learn

P	OPERATOR CATEGORY	OPERATORS	ASSO CI
1	Parentheses, braces	() , []	L to R
2	Unary operators	++, --, !, ~	R to L
3	Multiplicative, divison, modul	*, /, %	Lto R
4	Additive operators	+, -	L to R
5	Shift operators	<<, >>	L to R
6	Relational operators	<, <=, >, >=	L to R
7	Equality operators	==, !=	L to R
8	Bitwise operators	&, ^, |	L to R
9	Logical operators	&&, ||	L to R
10	Conditional operators	?:	R to L
11	Assignment operators	+=,-=, /=, *=, %=, &=,|=, <<=, >>=	R to L
12	Comma operators	,	L to R

READING A CHARACTER

getchar() **FUNCTION**

--------------<defined in stdio.h>

Reads a single character from keyboard and returns it

variable_name=getchar();

variable name is a valid C name that has been declared as **char** type.

WRITING A CHARACTER

ISM
We inspire you to learn

putchar(variable_name);

Eg:

```
#include<stdio.h>
main(){

        char answer;

        printf(“\n Enter any character”);

        answer=getchar();    /*  READING A CHARACTER*/

        printf(“\n Entered Character is “);

        putchar(answer);

        return 0;

}
```

FORMATTED INPUT

ISM
We inspire you to learn

scanf(“control string”,&variable1,&variable2…);

& -> address of the variable

control string -> Format specifier

Contains format of data being received

scanf returns the number of values read.

Eg: **scanf(“%d”,&**a);

%d-> Value read should be stored as integer

a-> Variable name

&a->address of variable a

Format specifies in C

Code	Meaning
%c	Reads a single character
%d	Reads a decimal integer
%i	Reads integer in decimal/octal/hex.
%e	Reads signed scientific notation
%f	Reads a floating pt number
%g	Reads signed floating point or signed scientific notation, whichever is shorter
%o	Reads an octal number
%s	Reads a string
%x	Reads a hexadecimal number
%p	Reads a pointer
%u	Reads an unsigned integer
%[]	Scans for a set of characters
%h	Reads a short integer

FORMATTED OUTPUT

ISM
We inspire you to learn

- ✓ printf(“control string”,arg1,arg2…argn);
- ✓ Control string consists of three types of items.
  1. Characters that will be printed on the screen as they appear.
  2. Format specifications that define the output format for display of each item.
  3. Escape sequences characters such as \n, \t and \b.
- ✓ Arguments are the variables whose values are formatted and printed according to the specifications of the control string.
- ✓ Eg: printf(“the sum of two digits is %d \n”,sum);
- ✓ printf returns the total number of characters displayed

TYPE CASTING

x=(int) 7.5   7.5 is converted into Integer

Type casting is useful when performing integer division

```
float f;
 f = 3/2; /*f=1.00*/
 f=(float)3/2;
```

Back slash constants

\a	System alaram
\b	Back space
\f	Form feed
\n	New line
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab
\”	Double quote
\’	Single quote
\0	Null character
\\	Back slash character