C PROGRAMMING FOR PROBLEM SOLVING (18CPS23)

Module 1 : Overview of C

Prof. Prabhakara B. K.
Associate Professor
Department of Computer Science and Engineering
Vivekananda College of Engineering and Technology, Puttur.

HISTORY OF C PROGRAMMING

Year	Language	Description
1960's	ALGOL	ALGOrithmic Language -By International Group
1967	BCPL	Basic Combined Programming Language -By Martin Richards
1970	B	is created for UNIX operating system at Bell Laboratories -By Ken Thompson
1972	C	was evolved from ALGOL, BCPL and B Developed by Dennis Ritchie at the Bell Laboratories in USA. It was developed along with UNIX operating system.
1978	K&R C	Developed by Karnighan and Dennis Ritchie
1989	ANSI C	American National Standards Institute (ANSI) formed a team in 1983 for defining standards for C.
1990	C89	International Standards Organization (ISO) has approved C by enforcing some additional standards.
1995	C95	Added some more features
1999	C99	Added some more features
2011	C11	Added some more features

STEPS TO LEARN A LANGUAGE

1. Alphabets of that language
2. Words of that language
3. Sentences of that language
4. Paragraphs of that language

Steps to learn any Communication Language

1. Character set of C language
2. Tokens of C language
3. instructions/ statements of C language
4. Program of C language

Steps to learn C Programming Language

CHARACTER SET OF C LANGUAGE

1. **Alphabets (52) – a to z A to Z**
2. **Digits (10) – 0 to 9**
3. **Special Characters (30)**

Character	Description	Character	Description	Character	Description
.	Period	<	left angular bracket	*	Asterisk
,	Comma	>	Right angular bracket	-	Minus
;	Semicolon	^	Caret	+	Plus
:	Colon	&	Ampersand	/	Slash
#	Hash	(	Left Parenthesis	\	Back Slash
'	Apostrophe	)	Right Parenthesis	=	Equal
$	Dollar	%	Percentage	[	Left bracket
_	Underscore	?	Question Mark	]	Right bracket
|	Vertical tab	"	Quotation Mark	{	Left Brace
~	Tilde symbol	!	Exclamation Mark	}	Right Brace

CHARACTER SET OF C LANGUAGE

4. White space characters (13)

They are Control Characters to perform actions.

These won't print in the screen.

These character follows \

Character	Description
	Space
\b	Back space
\a	Bell
\n	New line
\r	Enter Key press
\f	Form feed
\v	Vertical tab
\t	Horizontal tab
\0	NULL
\\	Printing \
\'	Printing '
\"	Printing "
\?	Printing ?

TOKENS

It is collection of characters.

Or

It is smallest unit of a program.

Types of tokens

1. Keywords
2. Identifiers
3. Constants
4. Strings
5. Operators
6. Special Characters

1. KEYWORDS

These are tokens having predefined meaning in C Compilers

Always coded in lowercase.

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. IDENTIFIERS

It is named memory location to store data for processing.

Example: variables, functions, arrays, pointers etc.

Rules to form an Identifier

1. First character must be an alphabet or underscore.
2. Must consist of only alphabets, digits or underscores.
3. First 31 characters are significant.
4. Cannot duplicate keywords.
5. Must not contain white space.

VALID AND INVALID IDENTIFIERS

Identifies	Valid / Invalid	Reason for Invalid
sum	valid	-
1sum	Invalid	First character is a digit
avg_value	valid	-
avg$sum	Invalid	Special character $ is not allowed
add123	valid	-
_abc	valid	-
int	Invalid	It is a keyword
1234	Invalid	First character is a digit
vivekanandacollegeofengineeringputtur	Invalid	No. of characters more than 31

3. CONSTANTS

Constant is a named memory location to store a value which will remains same throughout the execution of a program.

Types of Constants

1. Boolean constants 0 - False Non-zero - TRUE
2. Character constants - A character enclosed within apostrophes

   Valid 'a' '5' ' ' '+' Invalid 'ab' b' '4 "3"

   - Backslash constants (Escape sequences) \a \b \n \t \v \0 \? etc
3. **String** constants - Zero or more characters enclosed within quotation marks

   Valid "vcet“ "" "23.45“ " " Invalid "std sum avr" 'abc'
4. Integer constants - whole numbers with or without prefix + or -

- Decimal formed using 0 to 9 and their combination with or with out + or - sign

  Valid 25 +3434 -3294 Invalid 1,234 ±24 012
- Octal formed using 0 to 7 digits and their combinations with prefix 0 (zero)

  Valid 0123 0457 Invalid 01,234 0568 ±0456
- Hexadecimal formed using digits 0 to 9 or a to f (uppercase/lowercase) & their combination with prefix 0x or 0X Valid 0x12 0xabba 0X45ABF Invalid 23 0345 ±0x12 0xabg

3. CONSTANTS

These constants are represented in C like below

- Decimal Notation eg 65.25 0.00123
- Scientific Notation eg $6.525X10^{1}$ $1.23X10^{-3}$
- Exponential Notation eg 6.525E+01 1.23E-03

5. Real constants

- Single Precision (6 digits after decimal point)
  Examples 1.000000 -34.123456 +34563.012000
- Double Precision (more than 6 digits after decimal point (15 digits))
  Examples 4.12300045 -23.4561230 +3.1415927

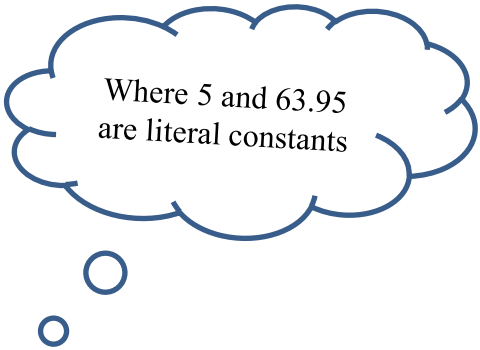
6. Complex constants Examples 2 + i5 3 + i5 2 - i5 3.12 – i5.53

7. Symbolic constants (Defined constants)

Syntax `#define symbolicname value`

Examples #define PI 3.1415927

#define printf display



8. Literal constant (Unnamed constants) - Examples c = a + 5 m = n - 63.95

9. Memory constants

float PI = 3.141593; is a declaration of variable in C

const float PI = 3.141593; is a declaration of constant in C

BASIC DATA TYPES for 16 bits machine

(Fundamental/Primary/Primitive)

	Data Type	Keyword	Size (in Byte)	Range	Control Strings
Basic Data Types	Void	void	0	Nil	
	Character	char	1	-128 to 127	%c (%s for string)
	Integer	int	2	-32768 to 32767	%d %o %x %i
	Real	float	4	3.4E-38 to 3.4E38	%f %e %g
	Double	double	8	1.7E-308 to 1.7E308	%lf
Basic Data Type's Modifiers	Signed character	signed char	1	-128 to 127	%c
	Unsigned character	unsigned char	1	0 to 255	%c
	Short integer	short int	2	-32768 to 32767	%hi
	Signed short integer	signed short int	2	-32768 to 32767	%hi
	Unsigned short integer	unsigned short int	2	0 to 65535	%hu
	Signed integer	signed int	2	-32768 to 32767	%d %o %x %i
	Unsigned integer	unsigned int	2	0 to 65535	%u
	Long integer	long int	4	-2147483648 to 2147483647	%li %ld
	Signed long integer	signed long int	4	-2147483648 to 2147483647	%li
	Unsigned long integer	unsigned long int	4	0 to 4294967295	%lu
	Long double	long double	10	3.4E-4932 to 1.1E4932	%Lf %Le %Lg

VARIABLES

It is named memory location to store data which may change during the execution of program.

Declaration

Is used to specify a) Name of a variable b) Type of data for a variable

Syntax

```
datatype variable1, variable2,…;
```

Examples

```
int a;
int a, b, c;
float avg_value;
char m_staus;
char name[30];
```

INITIALIZING VARIABLES DURING DECLARATION

It is used to give first value for a variable during declaration.

When a variable appears either side of an equal sign in an expression, this approach is must.

Declaration

Syntax datatype variable1=value, variable2=value,…;

Examples

```
int a=5, b=6, c;
float avg_value=5.64;
char m_staus='M';
 char name[30]="Ritchie“;
```

5. OPERATORS

What is an operator?

Operator is symbol that tells the computer to perform certain logical or mathematical manipulations.

What is an operand?

Operands are data provided for operators.

. or

Operands are those upon which operators are acts on.

What is an expression?

An expression is sequence of operators and operands to reduce into a single value (result).

TYPE OF OPERATORS

1. Arithmetic Operators + - * / %
2. Relational Operators > < >= <= == !=
3. Logical Operators && || !
4. Bitwise Operators & | ^ >> << ~
5. Assignment Operators = += -= *= /= %= &= |= ^= >>= <<=
6. Increment and Decrement Operators ++ --
7. Special Operators ( ) { } [ ] , * & -> sizeof() etc
8. Conditional Operator ? :

1. ARITHMETIC OPERATORS

Operator	Description	Precedence
+	Addition	2
-	Subtraction	2
*	Multiplication	1
/	Division	1
%	Modulus	1

Precedence (Hierarchy / Priority)

The order in which different operators of an expression are evaluated.

Example 1

C = 5 + 6 / 9

C = 5 + (6 / 9)

C = 5 + 0

C = 5

1. ARITHMETIC OPERATORS

Associativity

The order in which different operators with same precedence are evaluated in an expression.

Example 2

C = 4 + 9 * 6 – 13 / 5

C = 4 + (9 * 6) – 13 / 5

C = 4 + 54 – (13 / 5)

C = (4 + 54) – 2

C = 58 – 2

C = 56

Associativity Left to Right

1. ARITHMETIC OPERATORS

Example 3

C = 25 % 4 + 9 * 30 / 4 – 2 +100 / 2 % 4

C = (25 % 4) + 9 * 30 / 4 – 2 +100 / 2 % 4

C = 1 + (9 * 30) / 4 – 2 +100 / 2 % 4

C = 1 + (270 / 4) – 2 +100 / 2 % 4

C = 1 + 67 – 2 +(100 / 2) % 4

C = 1 + 67 – 2 + (50 % 4)

C = (1 + 67) – 2 + 2

C = (68 – 2) + 2

C = 66 + 2

C = 68

2. RELATIONAL OPERATORS

Operator	Description	Precedence
>	Greater than	1
<	Less than	1
>=	Greater than equal to	1
<=	Less than equal to	1
==	Equal to	2
!=	Not equal	2

- These operators are used in comparison process.
- If relational operators are used in an expression, the result of that expression is always either ‘0’ or ‘1’

2. RELATIONAL OPERATORS

Example 1

5 == 8 > 2

5 == (8 > 2)

5 == 1

0

Associativity
Left to Right

Example 2

9 > 3 < 5 == 1 != 0

(9 > 3) < 5 == 1 != 0

(1 < 5) == 1 != 0

(1 == 1) != 0

1 != 0

1

2. RELATIONAL OPERATORS

Example 3

If a =100, b =20, c =10, d = 5, e = 1 then Solve

a / b <= c – d + a % c – d == b >= e != b

100 / 20 <= 10 – 5 + 100 % 10 - 5 == 20 >= 1 != 20

(100 / 20) <= 10 – 5 + (100 % 10) - 5 == 20 >= 1 != 20

5 <= (10 – 5) + 0 - 5 == 20 >= 1 != 20

5 <=(5 + 0) – 5 == 20 >= 1 != 20

5 <= (5 – 5) == 20 >= 1 != 20

(5 <= 0) == (20 >= 1) != 20

(0 == 1) != 20

0 != 20

1

Operator	Precedence
/ % *	1
+ -	2
> < >= <=	3
== !=	4

3. LOGICAL OPERATORS

Operator	Description	Precedence
!	Negate/Complement	1
&&	Logical AND	2
||	Logical OR	3

A	B	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Example 1

C=2 && 5 || !6

C=2 && 5 || (!6)

C= (2 && 5) || 0

C=1 || 0

C= 1

Example 2

C= 12 && 5 && 4 || 3 || !0

C= 12 && 5 && 4 || 3 || (!0)

C=(12 && 5) && 4 || 3 || 1

C= (1 && 4) || 3 || 1

C= (1 || 3) || 1

C=1 || 1

C=1

3. LOGICAL OPERATORS

Example 3

11 + 2 > 6 && !0 || 11!=7 && 11 – 2 <= 5

11 + 2 > 6 && (!0) || 11!= 7 && 11 – 2 <= 5

(11 + 2) > 6 && 1 || 11 != 7 && (11 - 2) <= 5

(13 > 6) && 1 || 11 != 7 && (9 <= 5)

1 && 1 || (11 != 7) && 0

1 && 1 || 1 && 0

(1 && 1) || (1 && 0)

1 || 0

1

Operator	Precedence
!	1
/ % *	2
+ -	3
> < >= <=	4
== !=	5
&&	6
||	7

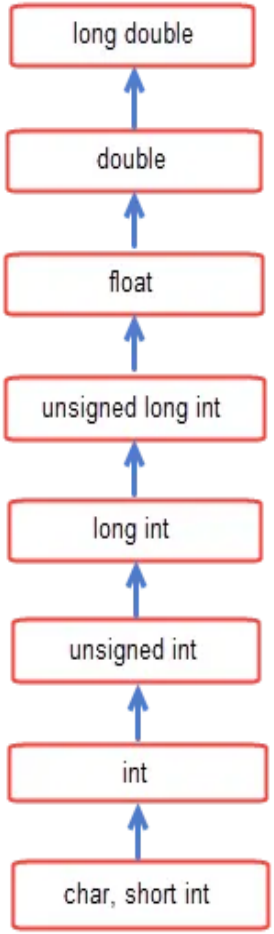
Associativity
Left to Right

TYPE CONVERSION

The process of converting data from one type to another type.

Techniques

1. Implicit - this is process of converting lower data type into higher data type automatically by the compiler.



Example 1

```
c = 7 / 2.0
c = 7 / 2.000000
c = 7.000000 / 2.000000
c = 3.500000
```

Example 2

What is x?

```
int a, b;
float x;
a=4;
b = 5;
x = b / a;
```

Solution

```
x = b / a
x = 5 / 4
x = 1
x = 1.000000
```

TYPE CONVERSION

2. Explicit (Type Casting)- this is process of converting lower data type into higher data type or higher data type into lower data type manually by user.

Syntax (Required data type) expression

Example 1

```
c = 1 / 2
c = (float)1 / 2
c = 1.000000 / 2
c = 1.000000 / 2.000000
c = 0.500000
```

Example 2

```
x = 1.000000
x = (int) 1.000000
x = 1
```

Example 3

What is x?

```
int a, b;
float x;
a = 4;
b = 5;
x = (float) b / a;
```

Solution

```
x = (float) b / a
x = (float) 5 / 4
x = 5.000000 / 4
x = 5.000000 / 4.000000
x = 1.250000
```

CONVERTING MATHEMATICAL EXPRESSION INTO C EXPRESSION

x^z	pow(x,z)
e^x	exp(x)
sin(ɵ)	sin(theta*3.1415927/180)
sin(x)	sin(x)
| x |	abs(x)
$e^{\|x\|}$	exp(abs(x))
$\sqrt{x}$	sqrt(x)
$\frac{x^2 - 2x}{\sqrt{5 + x^{y-2}}} - y$	(x*x – 2*x)/sqrt(5+pow(x,y-2))-y
$\frac{\sqrt{\|x + y\|}}{e^x} + \frac{2}{5}$	sqrt(abs(x+y))/exp(x)+2/5

CONVERTING MATHEMATICAL EXPRESSION INTO C EXPRESSION

area= $\sqrt{s(s-a)(s-b)(s-c)}$	area=sqrt(s*(s-a)*(s-b)*(s-c))
Sin $\left(\frac{b}{\sqrt{a^2+b^2}}\right)$	sin(b/sqrt(a*a+b*b))
$\tau_1=\sqrt{\left\{\frac{\sigma_x-\sigma_y}{2}\right\}+\tau xy^2}$	tow1=sqrt((rowx-rowy)/2+tow*x*y*y)
$\tau_1=\sqrt{\left\{\frac{\sigma_x-\sigma_y}{2}\right\}^2+\tau xy^2}$	**tow1=sqrt(pow((rowx-rowy)/2,2)+tow*x*y*y)**
$y=\frac{\alpha+\beta}{\sin\theta}+\lvert x\rvert$	**y=(alpha+beta)/sin(theta*3.1416/180)+abs(x)**

4. BITWISE OPERATORS

These operators are used to process data in binary form.

There are six bitwise operators

- Bitwise AND **&**
- Bitwise OR |
- Bitwise XOR ^
- Bitwise Right Shift >>
- Bitwise Left Shift <<
- Bitwise Negate ~

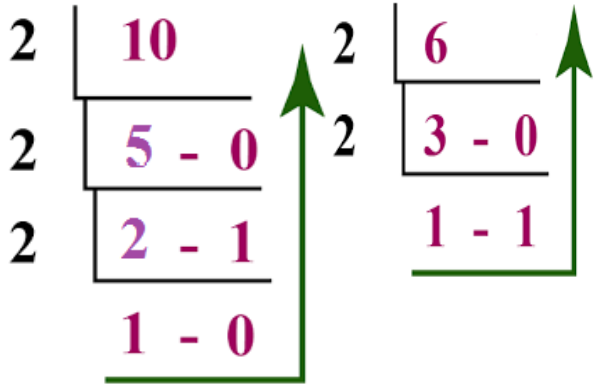
4. BITWISE OPERATORS

In computer memory data is stored in binary form (1's and 0's) is processed using bitwise operators.

Bitwise AND &

int a=10, b=6, c;

c = a & b;



A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

After evaluating above expression **c = 2**

```
a       0000 1010
b       0000 0110
--------------------
c       0000 0010
--------------------
```

$c = 0x2^7+0x2^6+0x2^5+0x2^4+0x2^3+0x2^2+1x2^1+0x2^0$

c = 0 + 0 + 0 + 0 + 0 + 0 + 2 + 0

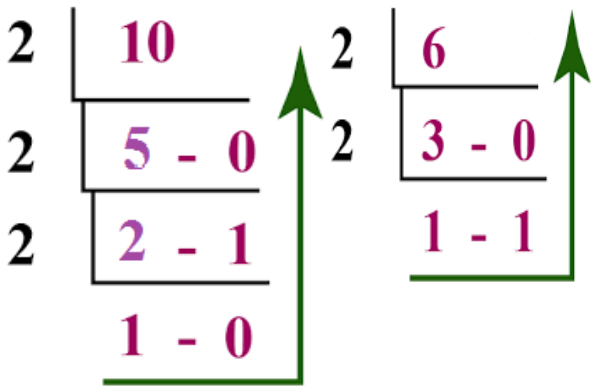
c = 2

4. BITWISE OPERATORS

Bitwise OR |

int a=10, b=6, c;

c = a | b;



A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

After evaluating above expression **c = 14**

a 0000 1010
b 0000 0110
---
c 0000 1110

$c = 0x2^7+0x2^6+0x2^5+0x2^4+1x2^3+1x2^2+1x2^1+0x2^0$

c = 0 + 0 + 0 + 0 + 8 + 4 + 2 + 0

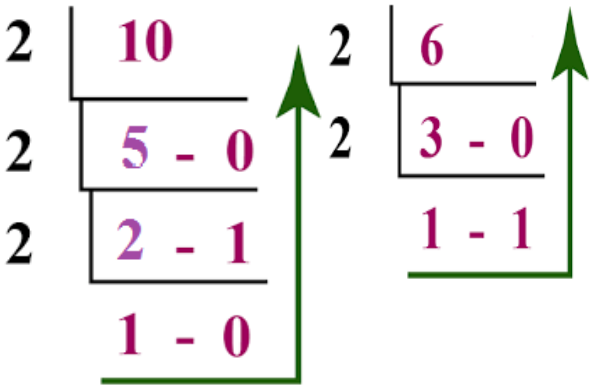
c = 14

4. BITWISE OPERATORS

Bitwise XOR ^

int a=10, b=6, c;

c = a ^ b;



A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

After evaluating above expression **c = 12**

a 0000 1010
b 0000 0110
---
c 0000 1100

$c = 0x2^7+0x2^6+0x2^5+0x2^4+1x2^3+1x2^2+0x2^1+0x2^0$

c = 0 + 0 + 0 + 0 + 8 + 4 + 0 + 0

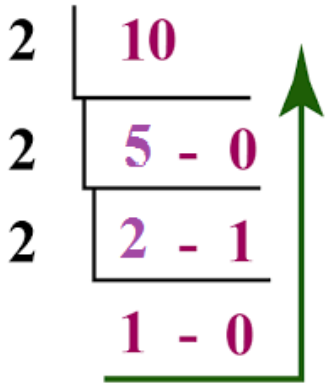
c = 12

4. BITWISE OPERATORS

Bitwise Negate ~

int a=10, c;

c = ~a;



A	Y
0	1
1	0

After evaluating above expression **c = 245**

a 0000 1010
---
c 1111 0101

$c = 1x2^7+1x2^6+1x2^5+1x2^4+0x2^3+1x2^2+0x2^1+1x2^0$

c = 128+ 64 + 32 + 16 + 0 + 4 + 0 + 1

c = 245

4. BITWISE OPERATORS

Bitwise Right Shift >>

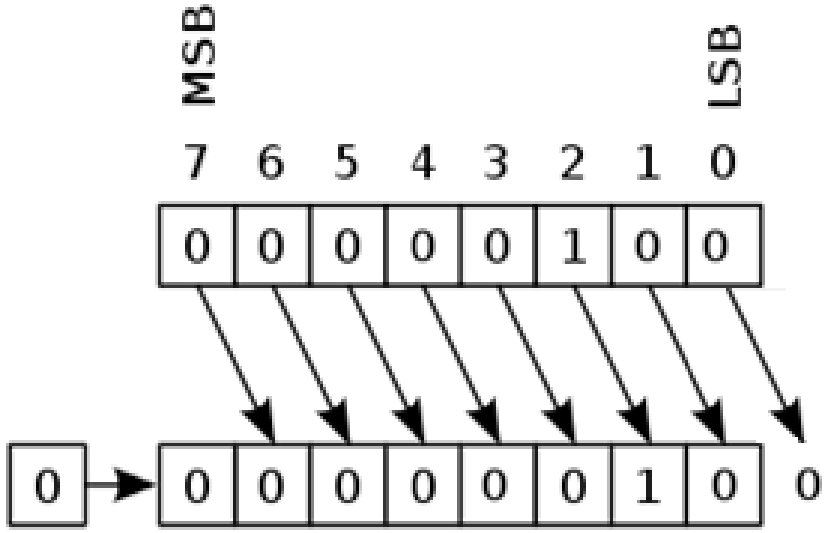
Shifting a variable or an integer right by one bit means adding a digit '0' from MSB and removing a digit from LSB.

Syntax

value>>no. of times

Example

c = 4>>1;



After evaluating above expression

c = 2

Shifting right by one bit also means dividing the given value by 2.

$c = 0x2^7+0x2^6+0x2^5+0x2^4+0x2^3+0x2^2+1x2^1+0x2^0$

c = 0 + 0 + 0 + 0 + 0 + 0 + 2 + 0

c = 2

4. BITWISE OPERATORS

Bitwise Left Shift <<

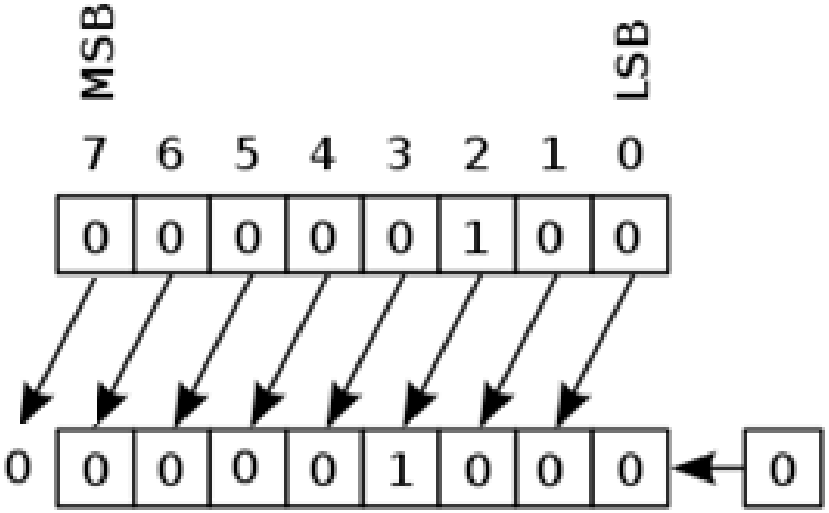
Shifting a variable or an integer left by one bit means adding a digit '0' from LSB and removing a digit from MSB.

Syntax

value<<no. of times

Example

c = 4<<1;



After evaluating above expression

c = 8

Shifting left by one bit also means multiplying the given value by 2.

$c = 0x2^7+0x2^6+0x2^5+0x2^4+1x2^3+0x2^2+1x2^1+0x2^0$

c = 0 + 0 + 0 + 0 + 8 + 0 + 0 + 0

c = 8

5. ASSIGNMENT OPERATORS

This (=) operator is used to assign result of an expression to a variable.

Syntax

variable = expression

Examples: c = 5 + 6 a = b * b + 1

d = a / 6 m=abs(x+y)/exp(x)

Shorthand Assignment operators

- They are combination of two operators.
- They save memory.
- There are 10 shorthand operators.

5. ASSIGNMENT OPERATORS

Shorthand Operator	Expression	Shorthand Expression
+=	a = a + 2	a += 2
-=	a = a – 2	a -= 2
*=	a = a * 2	a *= 2
/=	a = a / 2	a /= 2
%=	a = a % 2	a %=2
&=	a = a & 2	a &= 2
|=	a = a | 2	a |= 2
^=	a = a ^ 2	a ^= 2
>>=	a = a>>2	a>>= 2
<<=	a = a<<2	a<<= 2

5. ASSIGNMENT OPERATORS

If a = 3, b = 5, c = 8 then solve a += b *= c -= 5

Solution

a += b *= c -= 5
a += b *= c = c - 5
a += b *= c = 8 - 5
a += b *= c = 3
a += b *= 3
a += b = b * 3
a += b = 5 * 3
a += b = 15
a += 15
a = a + 15
a = 3 + 15
a = 18

Associativity
Right to Left

5. ASSIGNMENT OPERATORS

If a = 2, i = 4, j = 2 and k = 6 then solve i) i %= i / 3

ii) k *= i + j iii) j = i /= k iv) i + (j = 2 + k) v) a = i * (j /= k / 2)

i %= i / 3	k *= i + j	j = i /= k	a= i + (j = 2+k)	a = i * (j /= k / 2)
Solution	Solution	Solution	Solution	Solution
i %= 4 / 3	k *= 4 + 2	j = i = i / k	a = i + (j = 2 + 6)	a = i * (j /= 6 / 2)
i %= 1	k *= 6	j = i = 4 / 6	a = i + (j = 8)	a = i * (j /= 3)
i = i % 1	k = k * 6	j = i = 0	a = i + 8	a = i * (j = j / 3)
i = 4 % 1	k = 6 * 6	j = 0	a = 4 + 8	a = i * (j = 2 / 3)
i = 0	k = 36		a = 12	a = i * (j = 0)
				a = i * 0
				a = 0

6. INCREMENT & DECREMENT OPERATOR

These operators (++ --) are used to add a variable value by 1 or to subtract a variable value by 1

Increment Operator ++

This operator is used to add a variable value by 1.

Post Increment

Syntax | variable++ | Example a++

Pre Increment

Syntax | ++variable | Example ++a

If a=20 and b=10 then Evaluate i) b=a++ ii) b=++a

i) b=a++

Assign First Increment Next	b=a a=a+1	**b=20** a=20+1=21

ii) b=++a

Increment First Assign Next	a=a+1 b=a	a=20+1=21 **b=21**

6. INCREMENT & DECREMENT OPERATOR

Decrement Operator --

This operator is used to subtract a variable value by 1.

Post Decrement

Syntax `variable--` Example a--

Pre Decrement

Syntax `--variable` Example --a

If a=20 and b=10 then Evaluate i) b=a-- ii) b=--a

i) b=a--

Assign First Decrement Next	b=a a=a-1	**b=20** a=20-1=19

ii) b=--a

Decrement First Assign Next	a=a-1 b=a	a=20-1=19 **b=19**

6. INCREMENT & DECREMENT OPERATOR

If a = 5 then Solve below expressions

i) a += a++ + ++a

a += a++ + ++a **a = 5**

a += 5 + ++a **a = 6**

a += 5 + 7 **a = 7**

a += 12

a = a +12

a = 7 +12

a = 19

ii) a = --a – a--

a = --a – a-- **a = 5**

a = 4 – a-- **a = 4**

a = 4 – 4

a = 0 – 1

a = -1

7. SPECIAL OPERATORS

Some special operators are ( ) [ ] { } * -> & , sizeof() etc.

sizeof()

This operator is used to find number of bytes for any C objects.

This operator looks like library function. Not required header file.

Example 1	Example 2	Example 3
c= sizeof(5);	d= sizeof(double);	e= sizeof(5.5);
c=2	d=8	e=4

, (Comma)

This operator is having lowest precedence among C operators.

Example 1	Example 2	Example 3
c= 10,25;	d= (10,25);	e= (x=10,y=5,x+y);
c=10	d=25	e=15

It is also used to join two expressions.

Example a=5; Possible to write like single statement a=5,b=6;

b=6;

8. CONDITIONAL (TERNARY) OPERATOR

Three operands (data) are required for the operator **? :**

Syntax

```
Expression1 ? Expression2 : Expression3
```

1. The Expression1 evaluated first gives either zero or non-zero
2. If non-zero Expression2 is evaluated to becomes result.
3. If zero Expression3 is evaluated to become result.

Example1

```
c= 9< 5 ? 9 : 5;
c= 5
```

Example2

```
5 ? 10-2 : 10-3;
 8
```

Example3

```
if a=5, b=6
c= a>b? a*b : a/b;
c= 0
```

CLASSIFICATION OF OPERATORS BASED ON NUMBER OF OPERANDS

1. Unary Operators

Operators those operate on only one operand.

Unary - Unary + ++ -- ! ~

2. Binary Operators

Operators those operate on two operands.

+ - * / % > >= < <= == != && || & | ^ >> << etc.

3. Ternary Operator

Operator which operate on three operands.

? :

Precedence of C Operators

Operator	Description	Precedence level	Associativity
( ) [ ] . -> ++ --	Parentheses: grouping or function call Brackets (array subscript) Dot operator (Member selection via object name) Arrow operator(Member selection via pointer) Postfix increment/decrement	1	Left to Right
+ - ++ -- ! ~ * & (datatype) sizeof	Unary plus Unary minus Prefix increment/decrement Logical NOT One's complement Indirection Address (of operand) Type cast Determine size in bytes on this implementation	2	Right to Left
* / %	Multiplication Division Modulus	3	Left to Right
+ -	Addition Subtraction	4	Left to Right
<< >>	Left shift Right shift	5	Left to Right
< <= > >=	Less than Less than or equal to Greater than Greater than or equal to	6	Left to Right
== !=	Equal to Not equal to	7	Left to Right
&	Bitwise AND	8	Left to Right
^	Bitwise XOR	9	Left to Right
|	Bitwise OR	10	Left to Right
&&	Logical AND	11	Left to Right
||	Logical OR	12	Left to Right
?:	Conditional operator	13	Right to Left
= *= /= %= += -= &= ^= |= <<= >>=	Assignment operators	14	Right to Left
,	Comma operator	15	Left to Right

STRUCTURE OF C PROGRAM

(1) Documentation Section	/* To find area of a circle */
(2) Link Section	#include<stdio.h >
(3) Definition Section	#define PI 3.1415927
(4) Global Declaration Section	float area;

(5) Function main()
 	
i) Local variables declarations
 	
ii) Executable statements
	void main()
	{
	float r;
	printf("Enter value for r\n");
	scanf("%f", &r);
	area = PI*r*r;
	printf(" Area=%f " , area);
	}

(6) Other user defined functions	----------

The function main() section is a compulsory section in all C programs.
Other sections are used in a C program only when they are required.

DOCUMENTATION SECTION

- This section is also called "Comment Section / Remark Section"
- This section can be written anywhere in a C program (But not in between statements).
- These documentation lines are ignored by the compilers (not converted into machine level language).
- It begins with /* and ends with */ for multiple line documentation.
- Single line documentation // is used at the beginning of a line.
- It may include information like program author name, date of compilation, date of execution etc.
- They are written in a C program to have user-friendliness.
- Examples: /* To find area of a circle */

  // Program to find area of circle

PREPROCESSOR DIRECTIVES

- Preprocessor directives are instructions to the C compiler.
- These modify a source program before converting it into object program.
- These lines are not program statements.
- They begin with # (hash) symbol
- No semicolon (;) is expected at the end of these lines.
- The preprocessor directive can be extended to more than one line by preceding \ (slash) character at the end of the line.
- The Link Section and Definition Sections are preprocessor directives.
- Examples: #include #define #undef #ifdef #ifndef #if #else #endif #error #pragma #line #elif

HEADER FILES

- These files contains declarations and definitions for library (predefined/built-in) functions and macro definitions.
- These file names have extension .h for identification.

stdio.h	Standard input output functions	printf(), scanf(), gets(), puts(), getchar(), putchar() etc
conio.h	Console input output functions	clrscr(), getch(), putch(), cputs(), cgets() etc.
math.h	Mathematical functions	sqrt(), pow(), sin(), cos(), exp(), abs(), fabs() etc.
stdlib.h	Standard library functions	exit(), abort() etc
string.h	String functions	strlen(), strupr(), strlwr(), strcmp(), strcpy(), strcat() etc
ctype.h	Character functions	isalpha(),isdigit(),isspace(),isupper(),toupper(), tolower()
dos.h	Disk Operating System functions	sleep(), sound(), delay() etc
time.h	Date and Time functions	stime(), ctime(), clock(), time() etc
complex.h	Complex number functions	abs(), pow(), exp(), tan(), cos(), cosh() etc
process.h	Threads and processes functions	exit(), abort() etc

LINK SECTION

- This section is used to link header files, to have declarations for library functions or macro definitions used in a source program before converting it into object program.
- These lines appear commonly in the beginning of a program.
- To use library functions printf(), scanf()…the header file is

  #include<stdio.h>
- To use library functions sqrt(), pow(), sin(), exp(),… the header

  #include<math.h>
- To include user defined header files it is used like below

  #include"filename"

DEFINITION SECTION

- This section gives a value for a symbolic constant (macro).
- These lines can appear anywhere in a C program before their usage.
- These lines commonly appear in the beginning of a C program.
- This section is written in a program only if the symbolic constant defined is used more than once in that C program.

```
#define  PI  3.1415927
#define M 5
#define N  M
```

- We can also use this section for giving alternate value for keywords as shown in second example.

```
#define  cent int
#define  begin main
```

GLOBAL DECLARATION SECTION

- This section is used to declare different type of variables in a C program
- These variables are declared outside user defined functions. (normally after preprocessor directives)
- These variables are possible to use anywhere in a C program.

FUNCTION main()

- This is the only section mandatory in all C programs.
- The execution of C program start from this line.
- This is a user defined function because the statements inside this function are written by user to perform some specific task
- Since it is a user defined function no header file required for this.
- The return type is void when this function is not called by any other function; also not returning any value.
- The default return type of function main() is int.

The two sub-sections

- **Local variables declarations**

  The variables declared here are used only inside this function.

- **Executable statements**

  These are used to do specific task by the user inside function main().

OTHER USER DEFINED FUNCTIONS

- This section is also known as sub-programs section
- These are user defined by the user to do some specific task.
- These definitions can appear before or after the function main().
- If function definition appears before the function main(), then declaration for that user defined function is not required.
- User defined function declaration is also called function prototype.
- If function definition appears after function main(), then that user defined function must be declared.

Write a C program to find area of a circle.

```
/* To find area of circle */
#include<stdio.h>
#define PI 3.1415927
void main()
{
    float area, r;
    printf("Enter value for radius\n");
    scanf("%f",&r);
    area=PI*r*r;
    printf("Area=%f",area);
}
```

Output

```
Enter value for radius
1
Area=3.141593
```

Write a C program to find perimeter of circle.

```
/* To find perimeter of circle */
#include<stdio.h>
#define PI 3.1415927
void main()
{
    float peri, r;
    printf("Enter value for radius\n");
    scanf("%f",&r);
    peri=2*PI*r;
    printf("Perimeter=%f\n",peri);
}
```

Output

```
Enter value for radius
5
Perimeter=31.415928
```

Write C program to swap the contents of two integer variables.

```
/* To swap values of two integer variables */
#include<stdio.h>
void main()
{
        int a, b, c;
        printf("Enter two integers\n");
        scanf("%d%d",&a, &b);
        c=a;
        a=b;
        b=c;
        printf("After swap a =%d and b=%d\n",a, b);
}
```

Output

```
Enter two integers
10
20
After swap a=20 and b=10
```

Write a C program to convert given temperature from Fahrenheit to Celsius.

```
/* To convert given temperature from Fahrenheit to Celsius */
#include<stdio.h>
void main()
{
    float c, f;
    printf("Enter the temperature in Fahrenheit \n");
    scanf("%f", &f);
    c =(f - 32) * 5 / 9;
    printf("Temperature in Celsius is %f\n", c);
}
```

Output

```
Enter the temperature in Fahrenheit
50
Temperature in Celsius is 10.000000
```

Write a C program to find greatest among three integers using conditional operator.

```
/* Greatest among three integers using conditional operator */
#include<stdio.h>
void main()
{
    int a, b, c, big;
    printf("Enter three integers\n");
    scanf("%d%d%d", &a,&b,&c);
    big = a > b ? a : b;
    big = big > c ? big : c;
    printf("Biggest is %d\n",big);
}
```

Output

```
Enter three integers
10
20
30
Biggest is 30
```