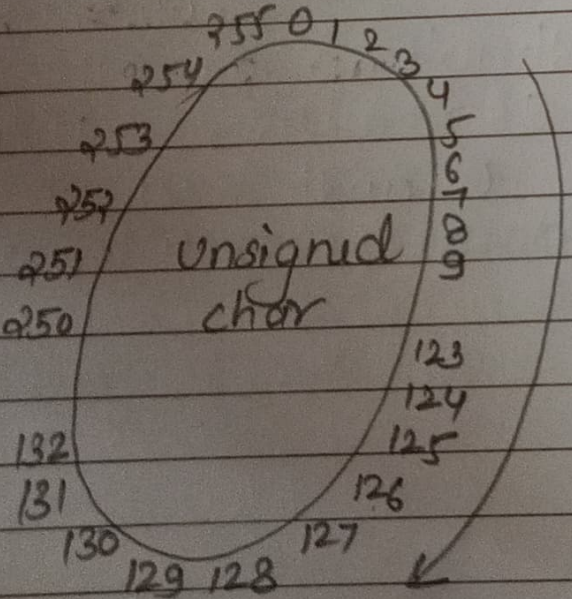
Date
Page

Cyclic Nature of datatype in C.

In C, some datatypes shows a special property called as cyclic nature of datatype. In this property, if we assign a value out of the range of that datatype then without showing any compiler error, it will assign a no. acc. to cyclic order. Data Type which shows cylic nature: (a) Char
(b) int
(c) long int

Signed Char

Range ⇒ 0 to 255.
If we assign a value below 0 or above 255 then its value will be changed.
If the value ~~to~~ will be assigned is > 255 then move in clockwise direc". If value is less than 0 then move in anticlockwise direc



for positive numbers.

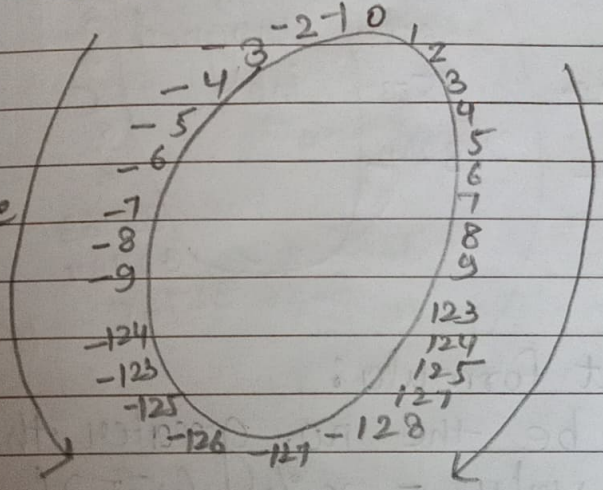
Short cut formula to find Cyclic value.

Let x be no. where x > 255
New value = x % 256

Let y be no. where y < 0
New value = 256 - (y % 256)

2.) Cyclic nature of signed char -

Range = -128 to 127
If we assign a value greater than 127 then move in clockwise direcⁿ and if we assign a value less than -128 then move in anti-clockwise direcⁿ.

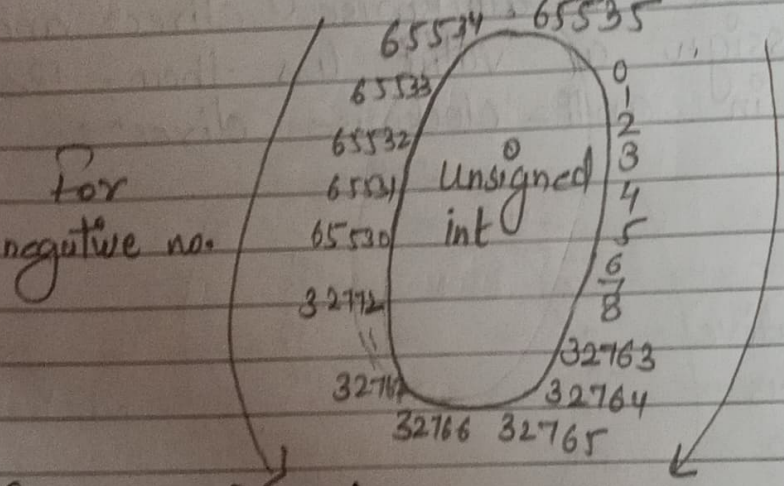


Short Cut formula -
Let x be the no. where x > 127 then,
p = x % 256
if p <= 127
New value = p
else
New value = p - 256
Let no. be y where y is less than -128 th

If p1 = 127
New value = -p
else
New value = 256 - p.

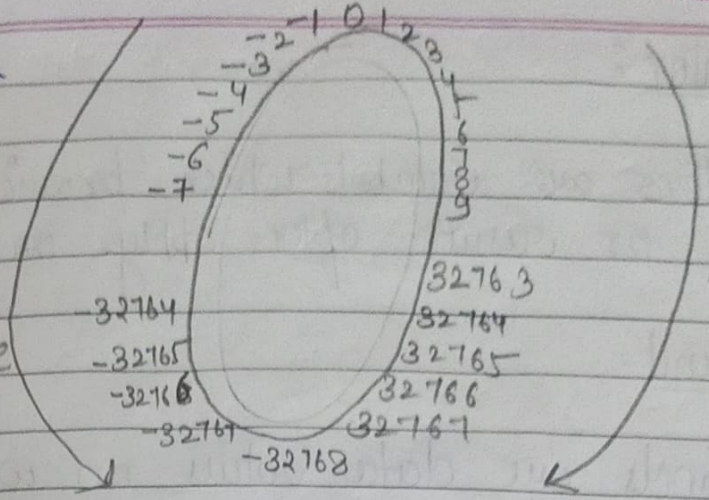
3) Cyclic nature of unsigned int.

Range = 0 to 65535
If we assign a value greater than 65535 then value of variable will be in clockwise direc". If no. is less than 0 then move in anticlockwise direc".



Short cut formula:
Let x be the no. greater than 65535 then
New value = x % 65536.
If no. y lus than 0.
New value = 65536 - (y % 65536)

4) Cyclic nature of signed int.
Range = -32768 to 32767
If we assign a value greater than 32767 than move in clockwise direc". If no is less than -32768 then move in anti clockwise



Shortcut formula :

Let x be the no. greater than 32767 then
p = x % 65536
if p <= 32767
New value = p
else New value = p - 65536
Let y be the no. less than -32768 then
p = y % 65536
if p <= 32767
New value = -p
else
New value = 65536 - p.

note → Same value is applied for signed & unsigned long int.

Date
Page

Operator:

Operators are symbols which provide a specific value or result after apply on data value.

Operand-

Operands are data values on which operator works and to get a result.

Expression -

Collection of operators and operands.

operator

Eg, C=A*B+7 → Expression.

operand

→ Category of operators:

1.) Arithmetic Operation
2.) Relational operator
3.) Equality operator
4.) Logical operator
5.) Assignment and short hand assignment operator
6.) Bitwise operator
7.) Conditional / Ternary operator
8.) Unary plus and unary minus
9.) Increament and decrement operator (p++, ++p, p--, --p)
10.) Comma Operator
11.) Sizeof operator
12.) Miscell operator

→ Types of Operator.

1) Unary operator
2) Binary operator
3) Ternary operator

1 Arithmetic Operator-
(Binary) (*, +, −, /, %).

S.no	Operation	Operator	Expression	Result
1	Addition	+	C=5+8	C=13
2	Subtraction	−	C=5-8	C=-3
3	Multiplication	*	C=5*8	C=40
4	Division	/	C=13/4	C=3
5	Modulus	%	C=13%4	C=1

(+, −, *, / : Char, int, float; % : Char, int)

Eg ① int x= 9*8+7/5-2%6;

Precedence- If in an expression there are two or more operators of diff. type than the operator having highest priority will evaluate first than lowest priority.

Associativity- Order of evaluation of same precedence/priority operators. It could be either left to right or right to left depending on the operator.

*, /, % → precedence of arithmetic operator.
+, − → left to right is associativity

Date
Page

① int x = 9 * 8 + 7 / 5 - 2 % 6

72 + 1 - 2

x = 71

② int x = 8 * 3 % 2 - 10 + 9 / 5

24 % 2 - 10 + 1

x = -9 ∴ 0 - 10 + 1

Q. Write a program to find the value of A % B without modulus operator.

R = A - B * (A / B);

C = 13 % 4 ⇒ 1
C = -13 % 4 ⇒ -1
C = 13 % -4 ⇒ 1
C = -13 % -4 ⇒ -1

B → 4 | A → 13 | 3
-12
1

Eg.
∴ 13 - 4 * (-13/4)
-13 - 4 * (-3)
-13 + 12
-1

Q. int x = 5.0 % 3;
printf("%d", x);

Error (5.0 is float can't apply on %).

Q. <math.h>
float x = fmod(5.5, 3);
printf("%f", x);

= 2.5

Date
Page

Q. or. → (Typecast)

→ A - B (cint) A / B).

Q. int X = 'B' * 3.6 + 10/3 % 7 - 5.2;

66 * 3.6 + 10/3 % 7 - 5.2

237.6 + 3 % 7 - 5.2

237.6 + 3 - 5.2

240.6 - 5.2

X = 238.4.

2. Relational Operator-

S.no	Operation	Operator	Expression	Result
1.	less than	<	C = 10 < 12	C = 1
2.	less than or equal to	<=	C = 10 <= 5	C = 0
3.	greater than	>	C = 13 > 17	C = 0
4.	greater than or equal to	>=	C = 17 >= 3	C = 1

→ Precedence = <, <=, >, >= (same)

→ Associativity = L - R

Eg ① int x = 10<7>=3<=1>9;
0>=3<=1>9;
0<=1>9;
1>9;
0;

② int x = 13%5<=3+9*7>1;
3<=3+9*7>1;
3<=3+63>1;
3<=66>1;
1>1;
0

③ int x = 7*(5>9/4);
7*(5>2);
7*1;
7

3. Equality Operator-

S.no.	Operation	Operator	Expression	Result
1.	Equals to	==	c=5==5	c=1
			c=5==7	c=0
2.	Not Equals to	!=	c=5!=5	c=0
			c=5!=7	c=1

Precidence = == , !=
Associativity = L-R.

Eg -① int x = 5 == 5 != 7;
1 != 7;
1

② int x = 5 * 3 == 3 + 9 >= 1;
15 == 3 + 9 >= 1;
15 == 12 >= 1;
15 == 1;
0

③ int x = 6 + (5 * 3 < 9 == 0);
6 + (15 < 9 == 0);
6 + (0 == 0);
6 + 1;
7.

4. Logical Operators.-

S.no	Operation	Operator	Expression	Result
1.	Logical AND	&&	C = 0 && 0	C = 0
			C = 0 && 1	C = 0
			C = 1 && 0	C = 0
			C = 1 && 1	C = 1
			C = 5 && 0	C = 0
			C = 0 && -2	C = 0
			C = 5 && -7	C = 1
2.	Logical OR	||	C = 0 || 0	C = 0
			C = 0 || 1	C = 1
			C = 1 || 1	C = 1
			C = 0 || 1	C = 1
			C = 5 || 0	C = 1
			C = 7 || -5	C = 1

3.	Logical NOT	!	C=!0 C=!1 C=!5 C=!-7	C=1 C=0 C=0 C=0

Associativity = L-R.

Eg ①. int x = 7, y = 3, z = 5, A;
A = (x = x + y * z) && (y = y + x - z)
printf ("%d %d %d %d", x, y, z, A);

x = 7 + 3 * 5
= 22

y = 3 + 22 - 5
= 20

z = 5.

A = 1 && 1
A = 1.

② A = !(x = x + y * z) && (y = y + y

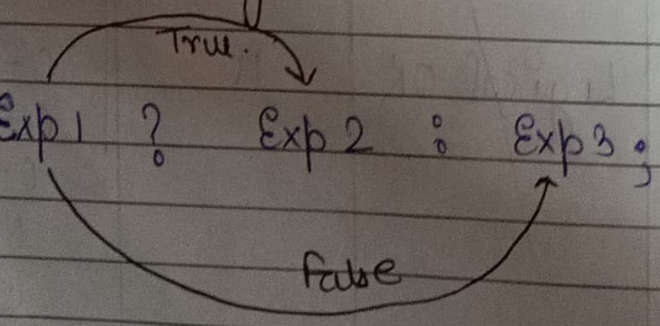
!1 && 1 — not exe

x = 7 + 3 * 5
x = 22.

y = 3.
z = 5; A = 0.

5.) Conditional / Ternary Operator -

Syntax →



Date
Page

Q. Write a program to input a no. and check no. is even or odd using Conditional Operator.

```
N%2==0 ? printf("Even") : printf ("odd");
```

Q. Write a program to input two no, and check and print the largest number.

```
N > M ? printf("%d", N): printf ("%d", M);
```

OR,

```
N>M ? printf("%d", N): N==M ? printf ("Both are
    equal & value = %d", M): printf ("%d", M);
```

Q. WAP to check if a person is eligible for voting or not if he or she is not eligible then also calculate after how many years that person will be eligible for voting using conditional operator.

```
Age >= 18 ? printf ("Eligible"): printf ("You are
    not eligible & will be eligible after %d",
    18 - Age);
```

Q. WAP to input 3 no. and print largest no.

```
L = X > Y ? X : Y ;
L = Z > L ? Z : L ;
```

or.

```
X >= Y && X >= Z
```

6. Bitwise Operator -
(&, |, ^, <<, >>, ~)

1. Bitwise AND (&) -

```
Char A = 10, B = 13, C;
C = A & B;
printf("%d", C);
```

S	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0
0	0	0	0	1	1	0	1
0	0	0	0	1	0	0	0

⇒ 8.

Date
Page

2. Bitwise OR (|) -

```
Char A = 10, B = 13, C;
     C = A | B;
     printf ("%d", C);
```

S	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0
0	0	0	0	1	1	0	1
0	0	0	0	1	1	1	1

⇒ 15.

3. Bitwise EXOR (^).

$X \wedge 0 = X$
$X \wedge X = 0$.

A	B	A ^ B
0	0	0
1	0	1
0	1	1
1	1	0

Above ex → 00000111
Output ⇒ 7

4. Bitwise left shift (<<)

```
Char a=10
a = a<<1;
printf ("%d",a).
```

⇒ 20.

or a = a<<2.

⇒ 40.

S 64 32 16 8 4 2 1
0 0 0 0 1 0 1 0
0 0 0 1 0 1 0 0 ← always
16+4 ⇒ 20

0 0 1 0 1 0 0 0
32+8 ⇒ 40

Date
Page

NOTE: Shortcut is N x 2^{bits}. (when no. is in range)

a = a << 4.

S	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0
1	0	1	0	0	0	0	0

here, sign bit is 1.

case 1: either no. is -ve

case 2: or two compliment

Again two compliment for actual no.

twos compliment → ones compliment + 1.

if no. 0 then 1 or if 1 then 0.

like.

for a = 12

a = a << 4.

→ -64.

S	64	32	16	8	4	2	1
1	0	1	0	0	0	0	0
0	1	0	1	1	1	1	1
						+	1
=	1	0	0	0	0	0	0

→ -64.

5. Bitwise Right shift (>>)

```
int X = 10;
X = X >> 1;
printf("%d", X);
```

S	64	32	16	8	4	2	1
0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	1

Same as before (0 or 1)

→ 4+1 → 5.

X = x >> 2;

0	0	0	0	0	0	1	0

→ 2.

Date
Page

note Short cut, old NO./2^{bits} ⇒ New no.

5 Bitwise Not (~) → (1's compliment)

```
int x = 10;
x = ~x;
printf ("%d", x);
```

0000 1010
1111 0101
0000 1010
+ 1
1+2+8 ⇒ -11

(here sign bit 1 → 2's compli)

let x = -10. (negative no. store in 2' compliment)

0 0000 1010 → 2' compliment
1 1111 0101
+ 1
1 1111 0110 ← -10 in memory.
0000 1001 ← n=10.
⇒ 9.

note Short cut, New no. ⇒ -(N+1).
not work for float value.

7. Sizeof Operator. (also a keyword).
use to evaluate size of a datatype.

```
# include <stdio.h>
int main()
{ printf("%lu\n", sizeof(char));      ⇒ 1
  printf("%lu\n", sizeof(int));       4
  printf("%lu\n", sizeof(short int)); 2
  printf("%lu\n", sizeof(float));     4
  printf("%lu\n", sizeof(double));    8
  printf("%lu\n", sizeof(long double)); 16.
```

```
{ char x;
float y;
int z;
printf("%lu\n", sizeof(x));
printf("%lu\n", sizeof(y));
printf("%lu\n", sizeof(z));
```

⇒ 1
4
4

→
```
{ int x = 5;
printf("%d %d", sizeof(x = x+10), x);
}
```
int→x = 15 [not print 15 bcoz it will not evaluate exp]

⇒ 4, 5

~~{ int x = sizeof(printf("Hello"));
printf("%d", x);~~

→
```
{ int x = sizeof(printf("Hello"));
printf("%d", x);
}
```

⇒ 4 [as printf return int value]

→
```
{ int x = sizeof("Hello");
printf("%d", x); }
```

⇒ 6.

Hello is string ↓

It give no. of character and one null character in end.

Date
Page

8. Comma Operator (,)

H Separates two expression.

→ { int x = 2,3;
printf ("%d", x); }.
⇒ Error.
H will show, 3 is taken a variable which is invalid.

→ { int x;
x = 2,3;
printf ("%d", x); }.
⇒ 2
}

→ { int x;
x = (2,3);
printf ("%d", x);
}
⇒ 3.

→ {int x;
x = (printf ("Hello"), printf ("Hi"));
printf ("%d", x);
}
HelloHi2

→ {int a=5, b=7, c=8, d=3, e;
e = (a=a+b*c, b=b-c+d, c=c*d+a);
printf ("%d %d %d %d %d", a, b, c, d, e);
a = 61 (24), b = 2, c = 85
e = (61, 2, 85).
Output → 61, 2, 85, 3, 85.

Date
Page

9. Assignment Operator (=, +=, -=, *=, /=, %=, &=, |= etc.)

It is used to assign r-value exp. output into l-value variable.

→
```
{int x;
 x=5;
 printf ("%d", x);
}
```
⇒ 5

→
```
{int x;
 5=x;
 printf("%d", x);
}
```
⇒ error, 5 is not variable.

→
```
{int x,y;
 x=y=5;
 printf("%d %d", x,y);
}
```
⇒ 5, 5

→
```
{int x=5;
 x+=4;
 printf ("%d", x);
}
```
⇒ 9.

/* x = x+4
= 5+4 = 9 */

→
```
{int x=5, y=3;
 x*=y+=7;
 printf("%d", x);
}
```
Output → 50

⇒ y = y+7
= 3+7 ⇒ 10
x = x*10
= 5*10 ⇒ 50

→
```
{int x=5, y=3, z;
 z= x+y=10;
 printf ("%d %d", x,y);
}
```
⇒ error. as x+y is not variable

Date
Page

10 Unary plus / Unary minus -

1.) unary minus -

- Changes sign.

Ex,

(1) int x=10, y;
y = -x;
printf("%d", y);
⇒ -10

(2) int x=10, y=7;
-x = y;
printf("%d", x);
⇒ error, L value require.

- It changes L value into R value.
- used for integer promotion.

int i;
float f;
char c;
short int s;
double d;

printf("%d %d %d %d %d \n", sizeof(-i), sizeof(-f), sizeof(-c), sizeof(-s), sizeof(-d));

⇒ 4, 4, 4, 4, 8.

2.) unary plus -

- It changes L value into R value.
- For integer promotion

(1). int x=10, y;
y = +x;
printf("%d", y);
⇒ 10.

Page

Increament and Decrement operator.

1) Post Inc. (P++)

first uses value then inc. by 1.

Eg ① int X=10, Y;
Y = X++;
printf ("%d %d", X, Y);

② 11, 10.
Y = 10 = X
X = X+1
X = 11.

2) Pre Inc. (++P)

first inc. by 1 then use value.

Eg int X=10, Y;
Y = ++X
printf ("%d %d", X, Y);
② 11, 11.
X = X+1 = 11.
Y = X = 11.

3) Post dec (P--)

fist use value then dec. by 1.

Eg → int X=10, Y;
Y = P--;
printf ("%d %d", X, Y);
② 9, 10.

Date
Page

4.) Pre. dec. (--P)

first dec. by 1 then use the value.

Eg. int X=10,y;
y=--X;
printf("%d%d", X,Y);
⇒ 9,9.

Eg→ int X=10,Y;
y=10++;
printf("%d%d", X,Y);
⇒ error.

Eg→ int X=5, Y=7, Z=6, A;
A=X++ * --Y / ++Z;
printf("%d%d%d%d", X,Y,Z,A);

⇒ 6, 6, 7, 4.

Eg→ ++X--
let X=10
11-- (error).

Eg→ int X=8, y;
y=X++ * X++
printf("%d%d", X,Y);
⇒ 10,64; 10,72; 10,81

X=8+1+1=10.
Y=8x9, 9x9, 8x8.