Session 4

Mostafa Akram

int 4byte =

printf("%d", var);

char letter = 'A'; → %d or %i → رقم

%c → رجع الحرف '7' → 55

→ %p → address

→ %f , %.lf

→ %Zu

int → 7 + 3 = 10

char → '7' + 3 = 58

%

letter

256

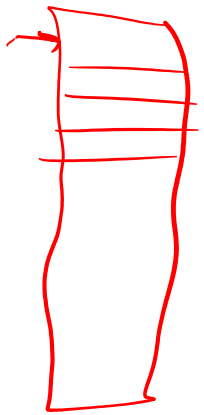
0, 255

char → حرف

int → عنوان %p

53 417

123 000



signed → $-\infty : \infty$

unsigned → متغير أرقام بالبت $0 \rightarrow \infty$

byte = 8bit = 1111 ... = 255 + (01) عنوان = 256

0000 0000

1111 1111

0000 0000 → 256

signed char →

-128 0 127

128 | 128 |

{ 128 → [-128, 0[

128 [0, 127]

One more thing from last session →

- **Using Range in switch Case in C**

Syntax

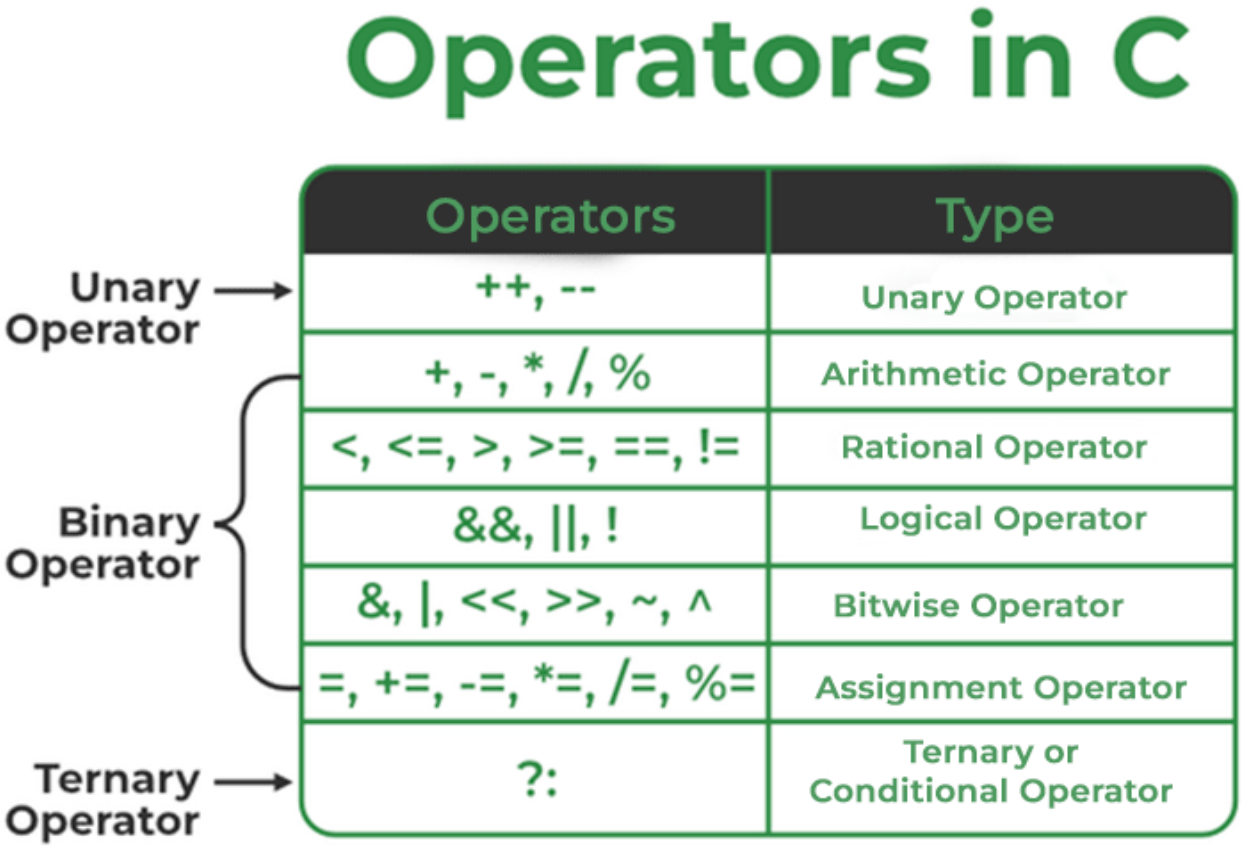
The syntax for using range case is:

```
case low ... high:
```

It can be used for a range of ASCII character codes like this:

```
case 'A' ... 'Z':
```

Operators in C



Arithmetic Operations in C

S. No.	Symbol	Operator	Description	Syntax
1	+	Plus	Adds two numeric values.	a + b
2	–	Minus	Subtracts right operand from left operand.	a – b
3	*	Multiply	Multiply two numeric values.	a * b
4	/	Divide	Divide two numeric values.	a / b
5	%	Modulus	Returns the remainder after diving the left operand with the right operand.	a % b

Arithmetic Operations in C

6	+	**Unary Plus**	Used to specify the positive values.	**+a**
7	–	**Unary Minus**	Flips the sign of the value.	**-a**
8	++	**Increment**	Increases the value of the operand by 1.	a++
9	—	**Decrement**	Decreases the value of the operand by 1.	a–

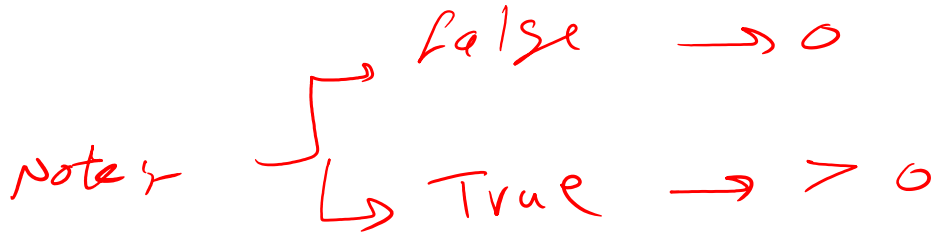
Relational Operators in C

S. No.	Symbol	Operator	Description	Syntax
1	<	**Less than**	Returns true if the left operand is less than the right operand. Else false	**a < b**
2	>	**Greater than**	Returns true if the left operand is greater than the right operand. Else false	**a > b**
3	<=	**Less than or equal to**	Returns true if the left operand is less than or equal to the right operand. Else false	**a <= b**
4	>=	**Greater than or equal to**	Returns true if the left operand is greater than or equal to right operand. Else false	**a >= b**

Relational Operators in C

5	==	**Equal to**	Returns true if both the operands are equal.	**a == b**
6	!=	**Not equal to**	Returns true if both the operands are NOT equal.	**a != b**

Logical Operator in C

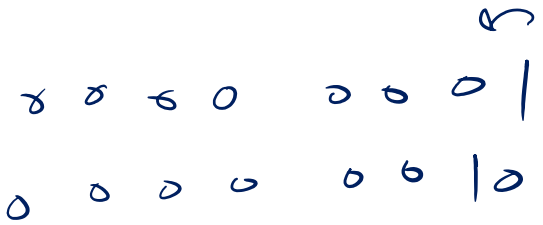


S. No.	Symbol	Operator	Description	Syntax
1	&&	Logical AND	Returns true if both the operands are true.	a && b
2	||	Logical OR	Returns true if both or any of the operand is true.	a || b
3	!	Logical NOT	Returns true if the operand is false.	!a

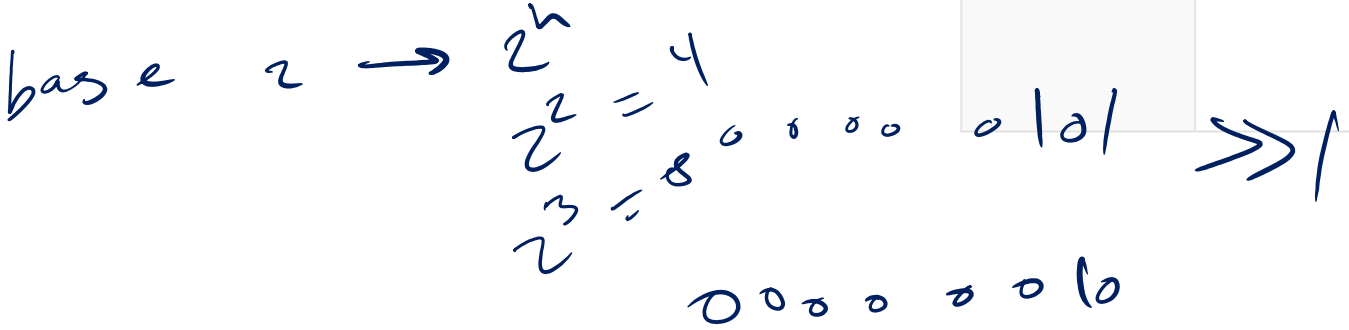
Bitwise Operators in C

S. No.	Symbol	Operator	Description	Syntax
1	&	**Bitwise AND**	Performs bit-by-bit AND operation and returns the result.	**a & b**
2	|	**Bitwise OR**	Performs bit-by-bit OR operation and returns the result.	**a | b**
3	^	**Bitwise XOR**	Performs bit-by-bit XOR operation and returns the result.	**a ^ b**

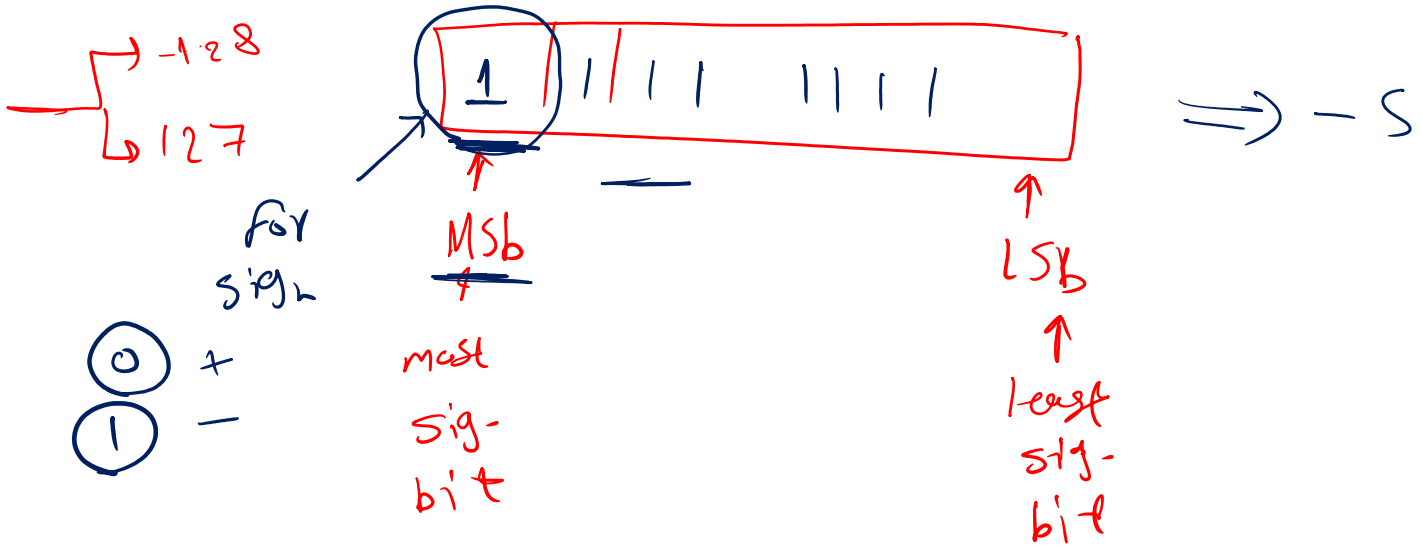
Bitwise Operators in C



4	~	**Bitwise First Complement**	Flips all the set and unset bits on the number.	**~a**
5	<<	**Bitwise Leftshift**	Shifts the number in binary form by one place in the operation and returns the result.	**a << b**
6	>>	**Bitwise Rightshilft**	Shifts the number in binary form by one place in the operation and returns the result.	**a >> b**

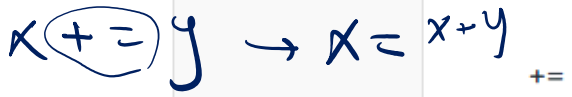


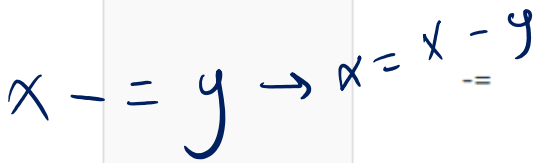
signed char x;

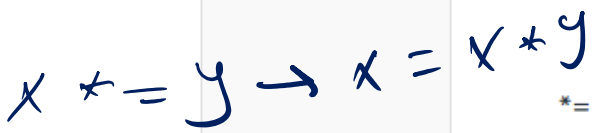


Assignment Operators in C

S. No.	Symbol	Operator	Description	Syntax
1	=	Simple Assignment	Assign the value of the right operand to the left operand.	a = b
2	+=	Plus and assign	Add the right operand and left operand and assign this value to the left operand.	a += b
3	-=	Minus and assign	Subtract the right operand and left operand and assign this value to the left operand.	a -= b
4	*=	Multiply and assign	Multiply the right operand and left operand and assign this value to the left operand.	a *= b

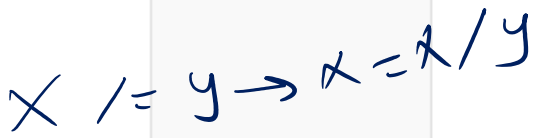


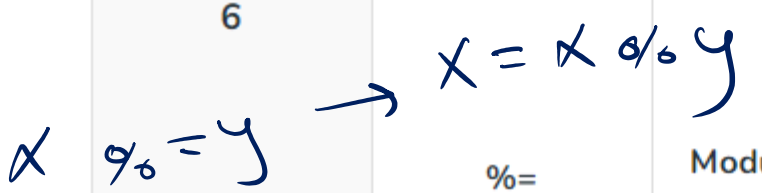


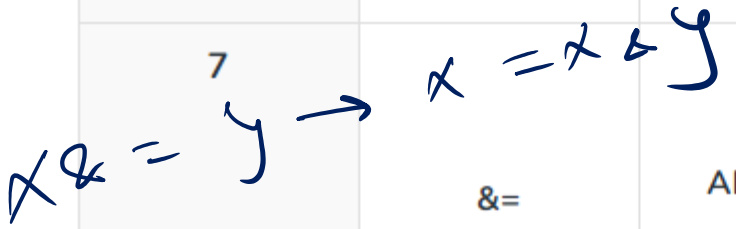


Assignment Operators in C

5	/=	**Divide and assign**	Divide the left operand with the right operand and assign this value to the left operand.	**a /= b**
6	%=	**Modulus and assign**	Assign the remainder in the division of left operand with the right operand to the left operand.	**a %= b**
7	&=	**AND and assign**	Performs bitwise AND and assigns this value to the left operand.	**a &= b**







Assignment Operators in C

8	|=	**OR and assign**	Performs bitwise OR and assigns this value to the left operand.	**a |= b**
9	^=	**XOR and assign**	Performs bitwise XOR and assigns this value to the left operand.	**a ^= b**
10	>>=	**Rightshift and assign**	Performs bitwise Rightshift and assign this value to the left operand.	**a >>= b**
11	<<=	**Leftshift and assign**	Performs bitwise Leftshift and assign this value to the left operand.	**a <<= b**

sizeof Operator

Syntax

```
sizeof (operand)
```

Condition ? Case 1 : Case 2

Conditional Operator (? :)

Syntax

```
operand1 ? operand2 : operand3;
```

Links