C Operators

An operator is simply a symbol that is used to perform operations. There can be many types of operations like arithmetic, logical, bitwise, etc.

There are following types of operators to perform different types of operations in C language.

- Arithmetic Operators
- Relational Operators
- Shift Operators
- Logical Operators
- Bitwise Operators

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right

NOTES BY: MRUNAL SALUNKE
mrunalsalunke@aiscpune.org
ABEDA INAMDAR SENIOR COLLEGE, BCA (SCIENCE) DEPARTMENT

Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

- Ternary or Conditional Operators
- Assignment Operator
- Misc Operator

Precedence of Operators in C

The precedence of operator species that which operator will be evaluated first and next. The associativity specifies the operator direction to be evaluated; it may be left to right or right to left.

Let's understand the precedence by the example given below:

1. `int value=10+20*10;`

The value variable will contain **210** because * (multiplicative operator) is evaluated before + (additive operator).

The precedence and associativity of C operators is given below:

C Format Specifier

The Format specifier is a string used in the formatted input and output functions. The format string determines the format of the input and output. The format string always starts with a '%' character.

The commonly used format specifiers in printf() function are:

Format specifier	Description

%d or %i	It is used to print the signed integer value where signed integer means that the variable can hold both positive and negative values.
%u	It is used to print the unsigned integer value where the unsigned integer means that the variable can hold only positive value.
%o	It is used to print the octal unsigned integer where octal integer value always starts with a 0 value.
%x	It is used to print the hexadecimal unsigned integer where the hexadecimal integer value always starts with a 0x value. In this, alphabetical characters are printed in small letters such as a, b, c, etc.
%X	It is used to print the hexadecimal unsigned integer, but %X prints the alphabetical characters in uppercase such as A, B, C, etc.
%f	It is used for printing the decimal floating-point values. By default, it prints the 6 values after '.'.
%e/%E	It is used for scientific notation. It is also known as Mantissa or Exponent.
%g	It is used to print the decimal floating-point values, and it uses the fixed precision, i.e., the value after the decimal in input would be exactly the same as the value in the output.
%p	It is used to print the address in a hexadecimal form.
%c	It is used to print the unsigned character.
%s	It is used to print the strings.
%ld	It is used to print the long-signed integer value.

Let's understand the format specifiers in detail through an example.

- **%d**

```
1. int main()
2. {
3.   int b=6;
4.   int c=8;
```

```
5.   printf("Value of b is:%d", b);
6.   printf("\nValue of c is:%d",c);
7.
8.     return 0;
9.  }
```

In the above code, we are printing the integer value of b and c by using the %d specifier.

Increment and Decrement Operators in C

Operators are the predefined symbols of the C/C++ library, and it is used to perform logical as well as mathematical operations to the operands. There are various types of operators in the C programming language, such as arithmetic, logical, bitwise, increment or decrement operators, etc.

Increment Operator

Increment Operators are the unary operators used to increment or add 1 to the operand value. The Increment operand is denoted by the double plus symbol (++). It has two types, Pre Increment and Post Increment Operators.

Pre-increment Operator

The pre-increment operator is used to increase the original value of the operand by 1 before assigning it to the expression.

Syntax

```
1.  X = ++A;
```

In the above syntax, the value of operand 'A' is increased by 1, and then a new value is assigned to the variable 'B'.

Example 1: Program to use the pre-increment operator in C

```
1.  #include <stdio.h>
2.  #include <conio.h>
```

```
3.  int main ()
4.  {
5.   // declare integer variables
6.  int x, y, z;
7.  printf (" Input the value of X: ");
8.  scanf (" %d", &x);
9.  printf ("  Input the value of Y: ");
10. scanf (" %d", &y);
11. printf (" Input the value of Z: ");
12. scanf (" %d", &z);
13. // use pre increment operator to update the value by 1
14. ++x;
15. ++y;
16. ++z;
17.
18. printf (" \n The updated value of the X: %d ", x);
19. printf (" \n The updated value of the Y: %d ", y);
20. printf (" \n The updated value of the Z: %d ", z);
21. return 0;
22. }
```

Output

```
Input the value of X: 10
Input the value of Y: 15
Input the value of Z: 20

The updated value of the X: 11
The updated value of the Y: 16
The updated value of the Z: 21
```

Post increment Operator

The post-increment operator is used to increment the original value of the operand by 1 after assigning it to the expression.

Syntax

```
1.  X = A++;
```

NOTES BY: MRUNAL SALUNKE
mrunalsalunke@aiscpune.org
ABEDA INAMDAR SENIOR COLLEGE, BCA (SCIENCE) DEPARTMENT

In the above syntax, the value of operand 'A' is assigned to the variable 'X'. After that, the value of variable 'A' is incremented by 1.

Example 2: Program to use the post-increment operator in C

```
1.  #include <stdio.h>
2.  #include <conio.h>
3.  int main ()
4.  {
5.   // declare integer variables
6.  int x, y, z, a, b, c;
7.  printf (" Input the value of X: ");
8.  scanf (" %d", &x);
9.  printf (" Input the value of Y: ");
10. scanf (" %d", &y);
11. printf (" Input the value of Z: ");
12. scanf (" %d", &z);
13. // use post-increment operator to update the value by 1
14. a = x++;
15. b = y++;
16. c = z++;
17.
18. printf (" \n The original value of a: %d", a);
19. printf (" \n The original value of b: %d", b);
20. printf (" \n The original value of c: %d", c);
21.
22. printf (" \n\n The updated value of the X: %d ", x);
23. printf (" \n The updated value of the Y: %d ", y);
24. printf (" \n The updated value of the Z: %d ", z);
25. return 0;
26. }
```

Output

```
Input the value of X: 10
Input the value of Y: 15
```

```
Input the value of Z: 20

The original value of a: 10
The original value of b: 15
The original value of c: 20

The updated value of the X: 11
The updated value of the Y: 16
The updated value of the Z: 21
```

Decrement Operator

Decrement Operator is the unary operator, which is used to decrease the original value of the operand by 1. The decrement operator is represented as the double minus symbol (--). It has two types, Pre Decrement and Post Decrement operators.

Pre Decrement Operator

The Pre Decrement Operator decreases the operand value by 1 before assigning it to the mathematical expression. In other words, the original value of the operand is first decreases, and then a new value is assigned to the other variable.

Syntax

1. B = --A;

In the above syntax, the value of operand 'A' is decreased by 1, and then a new value is assigned to the variable 'B'.

Example 3: Program to demonstrate the pre decrement operator in C

```
1. #include <stdio.h>
2. #include <conio.h>
3. int main ()
4. {
5.  // declare integer variables
6. int x, y, z;
7. printf (" Input the value of X: ");  5
8. scanf (" %d", &x);
9. printf (" \n Input the value of Y: ");  6
10. scanf (" %d", &y);
```

```
11. printf ("n Input the value of Z: ");  7
12. scanf (" %d", &z);
13. // use pre decrement operator to update the value by 1
14. --x;
15. --y;
16. --z;
17.
18. printf (" \n The updated value of the X: %d ", x);
19. printf (" \n The updated value of the Y: %d ", y);
20. printf (" \n The updated value of the Z: %d ", z);
21. return 0;
22. }
```

Output

```
Input the value of X: 5
Input the value of Y: 6
Input the value of Z: 7

The updated value of the X: 6
The updated value of the Y: 7
The updated value of the Z: 8
```

Post decrement Operator:

Post decrement operator is used to decrease the original value of the operand by 1 after assigning to the expression.

Syntax

```
1. B = A--;
```

In the above syntax, the value of operand 'A' is assigned to the variable 'B', and then the value of A is decreased by 1.

Example 4: Program to use the post decrement operator in C

```
1. #include <stdio.h>
2. #include <conio.h>
3. int main ()
```

```
4.  {
5.   // declare integer variables
6.  int x, y, z, a, b, c;
7.  printf (" Input the value of X: ");  6
8.  scanf (" %d", &x);
9.  printf (" Input the value of Y: ");  12
10. scanf (" %d", &y);
11. printf (" Input the value of Z: ");  18
12. scanf (" %d", &z);
13. // use post-decrement operator to update the value by 1
14. a = x--;
15. b = y--;
16. c = z--;
17.
18. printf (" \n The original value of a: %d", a);
19. printf (" \n The original value of b: %d", b);
20. printf (" \n The original value of c: %d", c);
21.
22. printf (" \n\n The updated value of the X: %d ", x);
23. printf (" \n The updated value of the Y: %d ", y);
24. printf (" \n The updated value of the Z: %d ", z);
25. return 0;
26. }
```

Output

```
Input the value of X: 6
Input the value of Y: 12
Input the value of Z: 18

The original value of a: 6
The original value of b: 12
The original value of c: 18

The updated value of the X: 5
The updated value of the Y: 11
The updated value of the Z: 17
```

Example 5: Program to perform the pre increment and pre decrement operator

```
1. #include <stdio.h>
2. #include <conio.h>
3. int main ()
4. {
5. // declare integer data type variable
6. int i, j, x, y;
7. printf (" Enter the value of i " );
8. scanf (" %d", &i);
9. printf (" Enter the value of j " );
10. scanf (" %d", &j);
11.
12. // use pre increment operator to update original value by 1
13. x = ++i;
14. printf (" After using the pre-incrementing, the value of i is %d \n", i);
15. printf (" The value of x is %d \n", x);
16.
17. // use pre decrement operator to decrease original value by 1
18. y = --j;
19. printf (" After using the pre-decrementing, the value of j is %d \n", j);
20. printf (" The value of y is %d \n", y);
21. return 0;
22. }
```

Output

```
Enter the value of i
5
Enter the value of j
10
After using the pre-incrementing, the value of i is 6
The value of x is 6
After using the pre-decrementing, the value of j is 9
The value of y is 9
```

Example 6: Program to print the post increment and post decrement operator

```
1. #include <stdio.h>
2. #include <conio.h>
```

```
3.  int main ()
4.  {
5.  // declare integer data type variable
6.  int i, j, x, y;
7.  printf (" Enter the value of i " );
8.  scanf (" %d", &i);
9.  printf (" Enter the value of j " );
10. scanf (" %d", &j);
11.
12. // use post increment operator to update original value by 1
13. x = i++;
14. printf (" After using the post-incrementing, the value of i is  %d \n", i);
15. printf (" The value of x is %d \n", x);
16.
17. // use post decrement operator to decrease original value by 1
18. y = j--;
19. printf (" After using the post-decrementing, the value of j is  %d \n", j);
20. printf (" The value of y is %d \n", y);
21. return 0;
22. }
```

Output

```
Enter the value of i 10
Enter the value of j 20
After using the post-incrementing, the value of i is 11
The value of x is 10
After using the post-decrementing, the value of j is 19
The value of y is 20
```

Difference between the Increment and Decrement Operator in C

Increment Operator	Decrement Operator

NOTES BY: MRUNAL SALUNKE
mrunalsalunke@aiscpune.org
ABEDA INAMDAR SENIOR COLLEGE, BCA (SCIENCE) DEPARTMENT

It is used to increment the value of a variable by 1.	It is used to decrease the operand values by 1.
The increment operator is represented as the double plus (++) symbol.	The decrement operator is represented as the double minus (--) symbol.
It has two types: pre-increment operator and post-increment operator.	Similarly, it has two types: the pre-decrement operator and the post-decrement operator.
Pre increment operator means the value of the operator is incremented first and then used in the expression. The post-increment operator means the operand is first used in the expression and then performs the increment operation to the original value by 1.	Pre decrement means the value of the operator is decremented first and then assigned in the expression. Whereas the post decrement operator means the operand is first used in the expression and then performs the decrement operation to the operand's original value by 1.
Syntax for the pre increment operator: X = ++a; Syntax for the post increment operator: X = a++;	Syntax for the pre decrement operator: X = --a; Syntax for the post decrement operator: X = a--;
Both operators' works only to the single operand, not values.	Both operators' works only to the single operand, not values.

Character I/O Functions

Input means to provide the program with some data to be used in the program

- Output means to display data on screen or write the data to a printer or a file.
- C programming language provides many built-in functions to read any given input and to display data on screen when there is a need to output the result.
- In this tutorial, we will learn about such functions, which can be used in our program to take input from user and to output the result on screen.
- All these built-in functions are present in C header file "#include<stdio.h>"
- To use the built in functions we have to specify header files at head in which a particular function is defined(written) while discussing about it.

NOTES BY: MRUNAL SALUNKE
mrunalsalunke@aiscpune.org
ABEDA INAMDAR SENIOR COLLEGE, BCA (SCIENCE) DEPARTMENT

---

getchar() & putchar()

getchar() Function:

- This function is used to Accept one character , from keyboard.
- It takes Only One Character as an input.

syntax

character_var = getchar();

Example:

```
#include<stdio.h>            //header file
void main()
{
char a;
printf("Enter a character>"); //B
a=getchar();         //getchar() funtion is  Assigned to variable a
printf("You Entered  %c ",a); //putchar(a); OUTPUT:B
}
```

Output:Enter a character> A
You Entered A

---

putchar() Function

- This function is used to print one character on the screen, from keyboard.

syntax

```
character_var = getchar();  //accept character
putchar(character_var);   // display the accepted character
```

NOTES BY: MRUNAL SALUNKE
mrunalsalunke@aiscpune.org
ABEDA INAMDAR SENIOR COLLEGE, BCA (SCIENCE) DEPARTMENT

EXAMPLE

```
#include<stdio.h>
character_var = getchar();  //a= getchar(); accept character  is 'A'
putchar(character_var);   // putchar(a); it will  display 'A' which is the accepted character
```

Output:
A

What if we give input more than one character,lets see whats happen when we give input as ABC

```
#include<stdio.h>
character_var = getchar();  //accept character
putchar(character_var);   // it will  display 'A' which is the accepted character
```

Output:

A

Output is the same as above,Because getchar() Function neglect the other characters except of first character

---

Character I/O Functions
getch() & putch()

getch() Function:

- This function is used to Accept one character , from keyboard ,without echo to the screen
- Without echo means when we type on screen it is not visible.
- This function provides functionality of not getting echo to the screen
- This Function takes One Character as an input

- Many programmer use this Function at the end of the program to stop screen until a Character is given as an input.(but its work is not to stop the screen,it Wait's for the character)

syntax

character_var = getch();
Example:

```
#include<stdio.h>
void main{
char var1;
printf("Enter The Character:");
var1 = getch();        //Not getting echo to the screen
printf("Character  was  %c",var1");
}
```

Output:Enter the Character: A
Character was A

---

putch() Function:

- This function is used to print one character ,on the screen
- This Function also print One Character as an input

Syntax

putch(variable_name);
Example:

```
#include<stdio.h>
void main{
char var1;
printf("Enter The Character>");
var1 = getchar();        //using getchar() function
putch(var1);
}
```

NOTES BY: MRUNAL SALUNKE
mrunalsalunke@aiscpune.org
ABEDA INAMDAR SENIOR COLLEGE, BCA (SCIENCE) DEPARTMENT

Ouput:

Enter the Charater>A
A

Character Input

getche() Function:

- This function is used to Accept one character , from keyboard ,echo to the screen
- This Function also takes One Character as an input

syntax

character_var = getche();
Example:

```
#include<stdio.h>
void main{
char var1;
printf("Enter The Character:");
var1 = getche();        // echo to the screen
printf("Character  was  %c",var1");
}
```

Output:

Enter the Character: B
Character was B

---

getchar(), getch()-not show echo,getche()-show echo---------- putchar(),putch()-no echo,putce()-show echo

String input and output:

C gets() and puts() functions

The gets() and puts() are declared in the header file stdio.h. Both the functions are involved in the input/output operations of the strings.

C gets() function

The gets() function enables the user to enter some characters followed by the enter key. All the characters entered by the user get stored in a character array. The null character is added to the array to make it a string. The gets() allows the user to enter the space-separated strings. It returns the string entered by the user.

Declaration`char[] gets(char[]);`

Reading string using gets()

```
#include<stdio.h>
void main ()
{
   char s[30];
   printf("Enter the string? ");
   gets(s);
   printf("You entered %s",s);
}
```

Output

```
Enter the string?
javatpoint is the best
You entered javatpoint is the best
```

The gets() function is risky to use since it doesn't perform any array bound checking and keep reading the characters until the new line (enter) is encountered. It suffers from buffer overflow, which can be avoided by using fgets(). The fgets() makes sure that not more than the maximum limit of characters are read. Consider the following example

```
#include<stdio.h>
void main()
{
  char str[20];
  printf("Enter the string? ");
```

```
6.    fgets(str, 20, stdin);
7.    printf("%s", str);
8.  }
```

Output

```
Enter the string? javatpoint is the best website
javatpoint is the b
```

C puts() function

The puts() function is very much similar to printf() function. The puts() function is used to print the string on the console which is previously read by using gets() or scanf() function. The puts() function returns an integer value representing the number of characters being printed on the console. Since, it prints an additional newline character with the string, which moves the cursor to the new line on the console, the integer value returned by puts() will always be equal to the number of characters present in the string plus 1.

Declaration

```
int puts(char[])
```

Let's see an example to read a string using gets() and print it on the console using puts().

```
1.  #include<stdio.h>
2.  #include <string.h>
3.  int main(){
4.  char name[50];
5.  printf("Enter your name: ");
6.  gets(name); //reads string from user
7.  printf("Your name is: ");
8.  puts(name);  //displays string
9.  return 0;
10. }
```

Output:

```
Enter your name: Sonoo Jaiswal
Your name is: Sonoo Jaiswal
```

Format Input and Output

printf() and scanf() in C

The printf() and scanf() functions are used for input and output in C language. Both functions are inbuilt library functions, defined in stdio.h (header file).

printf() function

The **printf() function** is used for output. It prints the given statement to the console.

The syntax of printf() function is given below:

```
1. printf("format string",argument_list);
```

The **format string** can be %d (integer), %c (character), %s (string), %f (float) etc.

---

scanf() function

The **scanf() function** is used for input. It reads the input data from the console.

```
1. scanf("format string",argument_list);
```

Program to print cube of given number

Let's see a simple example of c language that gets input from the user and prints the cube of the given number.

```
1. #include<stdio.h>
2. int main(){
3. int number;
4. printf("enter a number:");
5. scanf("%d",&number);
6. printf("cube of number is:%d ",number*number*number);
7. return 0;
```

NOTES BY: MRUNAL SALUNKE
mrunalsalunke@aiscpune.org
ABEDA INAMDAR SENIOR COLLEGE, BCA (SCIENCE) DEPARTMENT

```
}
```

Output

```
enter a number:5
cube of number is:125
```

The **scanf("%d",&number)** statement reads integer number from the console and stores the given value in number variable.

The **printf("cube of number is:%d ",number*number*number)** statement prints the cube of number on the console.

Program to print sum of 2 numbers

Let's see a simple example of input and output in C language that prints addition of 2 numbers.

```
#include<stdio.h>
int main(){
int x=0,y=0,result=0;

printf("enter first number:");
scanf("%d",&x);
printf("enter second number:");
scanf("%d",&y);

result=x+y;
printf("sum of 2 numbers:%d ",result);

return 0;
}
```

Output

```
enter first number:9
enter second number:9
sum of 2 numbers:18
```

C - scanf() and sprintf() function

In our last article, we introduced you to the two most commonly used formatted console input/output functions scanf() and printf(). In this article, we are going to explain two more formatted console input/output functions, sscanf() and sprintf().

Formatted functions	Description
sscanf()	The sscanf() function reads the values from a char[] array and store each value into variables of matching data type by specifying the matching format specifier.
sprintf()	The sprintf() function reads the one or multiple values specified with their matching format specifiers and store these values in a char[] array.

Let's take a look at the prototype of sscanf() and pprintf() function and explain them with examples in the upcoming section.

- An example of sscanf() function

In the upcoming example, we are going to use scanf function to read(linearly) the multiple values present in a char[] array and store each value in a matching variable using a matching format specifier within sscanf() function.

```
#include<stdio.h>
int main()
{
char ar[20] = "User M 19 1.85";
char str[10];
char ch;
int i;
float f;

/* Calling sscanf() to read multiple values from a char[] array and store each value in mat
ching variable */
sscanf(ar, "%s %c %d", &str, &ch, &i, &f);

printf("The value in string is : %s ", str);
printf("\n");
printf("The value in char is : %c ", ch);
printf("\n");
printf("The value in int is : %d ", i);
printf("\n");
printf("The value in float is : %f ", f);
sscanf(ar, "%s %c %d", &str, &ch, &i);
return 0;
}
```

Output

```
The value in string is : User
The value in char is : M
The value in int is : 19
The value in float is : 1.850000
```

As you may see in the code and its output, we have called the sscanf() function, which linearly reads the value from a char[] array and store each value into variables of matching data type by specifying the matching format specifier in sscanf() function, such as -

- Reads the first value i.e. a string by using format specifier %s and stores it in str.
- Reads the second value i.e. a char by using format specifier %c and stores it in ch.

- Reads the third value i.e. a int by using format specifier %d and stores it in i.
- Reads the fourth value i.e. float by using format specifier %f and stores it in f.

- An example of sprintf() function

In the upcoming example, we are going to use sprintf() function, which reads one or multiple values specified with their matching format specifiers and store these values in a char[] array.

```
#include<stdio.h>

int main()
{
char target[20];
char name[10] = "Andrea";
char gender  = 'F';
int age = 25;
float height = 1.70;
printf("The name is : %s", name);
printf("\n");
printf("The gender is : %c", gender);
printf("\n");
printf("The age is : %d", age);
printf("\n");
printf("The height is : %f", height);
/* Calling sprintf() function to read multiple variables and store their values in a char[] ar
ray i.e. string.*/
sprintf(target, "%s %c %d %f", name, gender, age, height);

printf("\n");
printf("The value in the target string is : %s ", target);
return 0;
}
```

Output

```
The name is : Andrea
The gender is : F
The age is : 25
The height is : 1.700000
The value in the target string is : Andrea F 25 1.700000
```

Program Analysis

As you may see in the code and its output, we have called the sprintf() function, which linearly reads multiple values by specifying the matching format specifier with their variable names and stores each all these values in a char[] **array named** target**.**