Constants and Operators

Types of Constants in C

Type of Constants	Data type	Example of Data type
Octal constant	int	Example: 013 /*starts with 0 */
Hexadecimal constant	int	Example: 0x90 /*starts with 0x*/
character constants	char	Example: 'X', 'Y', 'Z'
string constants	char	Example: “PQRS”, “ABCD”

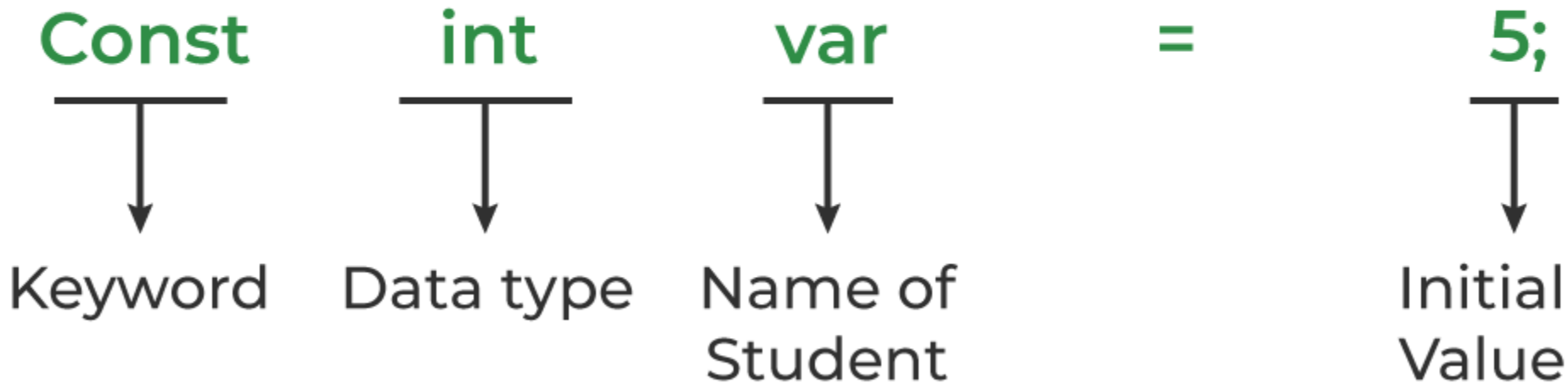
What is a constant in C?

As the name suggests, a constant in C is a variable that cannot be modified once it is declared in the program. We can not make any change in the value of the constant variables after they are defined.

How to Define Constant in C?

We define a constant in C language using the const keyword. Also known as a const type qualifier, the **const keyword** is placed at the start of the variable declaration to declare that variable as a constant.

Constants


Const
int
var
=
5;
Keyword
Data type
Name of Student
Initial Value

How to Declare Constants

```
const int var;
```

```
const int var;
var=5
```

```
Const int var = 5;
```

Defining Constant using #define Preprocessor

We can also define a constant in C using #define preprocessor. The constants defined using #define are macros that behave like a constant. These constants are not handled by the compiler, they are handled by the preprocessor and are replaced by their value before complication.

```
// C Program to define a constant using #define
#include <stdio.h>
#define pi 3.14

int main()
{

    printf("The value of pi: %.2f", pi);
    return 0;
}
```

Output

The value of pi: 3.14

What are Operators in C?

Operators can be defined as the symbols that help us to perform specific mathematical, relational, bitwise, conditional, or logical computations on operands. In other words, we can say that an operator operates the operands. For example, '+' is an operator used for addition, as shown below:

```
c = a + b;
```

Here, '+' is the operator known as the addition operator, and 'a' and 'b' are operands. The addition operator tells the compiler to add both of the operands 'a' and 'b'. The functionality of the C programming language is incomplete without the use of operators.

Operators in C

	Operators	Type
Unary Operator →	++, --	Unery Operator
Binary Operator	+, -, *, /, %	Arithmetic Operator
	<, <=, >, >=, ==, !=	Relational Operator
	&&, ||, !	Logical Operator
	&, |, <<, >>, ~, ^	Bitwise Operator
	=, +=, -=, *=, /=, %=	Assignment Operator
Ternary Operator →	?:	Ternary or Conditional Operator

The above operators have been discussed in detail:

1. Arithmetic Operations in C
These operators are used to perform arithmetic/mathematical operations on operands. Examples: (+, -, *, /, %,++,–). Arithmetic operators are of two types:

a) Unary Operators:
Operators that operate or work with a single operand are unary operators. For example: Increment(++) and Decrement(–) Operators

```
int val = 5;
cout<<++val;  // 6
```

b) Binary Operators:

Operators that operate or work with two operands are binary operators.
For example: Addition(+), Subtraction(-), multiplication(*), Division(/) operators

```
int a = 7;
int b = 2;
cout<<a+b; // 9
```

2. Relational Operators in C

These are used for the comparison of the values of two operands. For example, checking if one operand is equal to the other operand or not, whether an operand is greater than the other operand or not, etc. Some of the relational operators are (==, >= , <=)

```
int a = 3;
int b = 5;
printf(a < b);
// operator to check if a is smaller than b
```

3. Logical Operator in C

Logical Operators are used to combining two or more conditions/constraints or to complement the evaluation of the original condition in consideration. The result of the operation of a logical operator is a Boolean value either true or false.

For example, the logical AND represented as the '&&' operator in C returns true when both the conditions under consideration are satisfied. Otherwise, it returns false. Therefore, a && b returns true when both a and b are true (i.e. non-zero)(See this article for more reference).

```
cout<<((4 != 5) && (4 < 5));     // true
```

4. Bitwise Operators in C

The Bitwise operators are used to perform bit-level operations on the operands. The operators are first converted to bit-level and then the calculation is performed on the operands. Mathematical operations such as addition, subtraction, multiplication, etc. can be performed at the bit level for faster processing. For example, the bitwise AND operator represented as '&' in C takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1(True).

```
int a = 5, b = 9;   // a = 5(00000101), b = 9(00001001)
cout << (a ^ b);   //  00001100
cout <<(~a);       // 11111010
```

5. Assignment Operators in C

Assignment operators are used to assign value to a variable. The left side operand of the assignment operator is a variable and the right side operand of the assignment operator is a value. The value on the right side must be of the same data type as the variable on the left side otherwise the compiler will raise an error.

Different types of assignment operators are shown below:

a) “=”
This is the simplest assignment operator. This operator is used to assign the value on the right to the variable on the left.
Example:

```
a = 10;
b = 20;
ch = 'y';
```

b) “+=”

This operator is the combination of the ‘+’ and ‘=’ operators. This operator first adds the current value of the variable on left to the value on the right and then assigns the result to the variable on the left.

Example:

(a += b) can be written as (a = a + b)

If initially value stored in a is 5. Then (a += 6) = 11.

c) “-=”

This operator is a combination of ‘-‘ and ‘=’ operators. This operator first subtracts the value on the right from the current value of the variable on left and then assigns the result to the variable on the left.

Example:

(a -= b) can be written as (a = a - b)

If initially value stored in a is 8. Then (a -= 6) = 2.

d) “*=”

This operator is a combination of the ‘*’ and ‘=’ operators. This operator first multiplies the current value of the variable on left to the value on the right and then assigns the result to the variable on the left.

Example:

(a *= b) can be written as (a = a * b)

If initially, the value stored in a is 5. Then (a *= 6) = 30.

e) “/=”

This operator is a combination of the ‘/’ and ‘=’ operators. This operator first divides the current value of the variable on left by the value on the right and then assigns the result to the variable on the left.

Example:

(a /= b) can be written as (a = a / b)

If initially, the value stored in a is 6. Then (a /= 2) = 3.

6. Other Operators

Apart from the above operators, there are some other operators available in C used to perform some specific tasks. Some of them are discussed here:

i. sizeof operator

sizeof is much used in the C programming language.
It is a compile-time unary operator which can be used to compute the size of its operand.
The result of sizeof is of the unsigned integral type which is usually denoted by size_t.

```
// C Program to Demonstrate the working concept of
// Operators
#include <stdio.h>

int main()
{

    int a = 10, b = 5;
    // Arithmetic operators
    printf("Following are the Arithmetic operators in C\n");
    printf("The value of a + b is %d\n", a + b);
    printf("The value of a - b is %d\n", a - b);

    printf("The value of a * b is %d\n", a * b);
    printf("The value of a / b is %d\n", a / b);
    printf("The value of a % b is %d\n", a % b);
    // First print (a) and then increment it
    // by 1
    printf("The value of a++ is %d\n", a++);
```

```
// First print (a+1) and then decrease it
   // by 1
   printf("The value of a-- is %d\n", a--);

   // Increment (a) by (a+1) and then print
   printf("The value of ++a is %d\n", ++a);

   // Decrement (a+1) by (a) and then print
   printf("The value of --a is %d\n", --a);
  // Assignment Operators --> used to assign values to
   // variables int a =3, b=9; char d='d';

   // Comparison operators
   // Output of all these comparison operators will be (1)
   // if it is true and (0) if it is false
   printf(
      "\nFollowing are the comparison operators in C\n");
   printf("The value of a == b is %d\n", (a == b));
   printf("The value of a != b is %d\n", (a != b));
   printf("The value of a >= b is %d\n", (a >= b));
   printf("The value of a <= b is %d\n", (a <= b));
   printf("The value of a > b is %d\n", (a > b));
   printf("The value of a < b is %d\n", (a < b));
```

```
// Logical operators
    printf("\nFollowing are the logical operators in C\n");
    printf("The value of this logical and operator ((a==b) "
           "&& (a<b)) is:%d\n",
           ((a == b) && (a < b)));
    printf("The value of this logical or operator ((a==b) "
           "|| (a<b)) is:%d\n",
           ((a == b) || (a < b)));
    printf("The value of this logical not operator "
           "(!(a==b)) is:%d\n",
           (!(a == b)));

    return 0;
}
```

Output

Following are the Arithmetic operators in C

The value of a + b is 15

The value of a - b is 5

The value of a * b is 50

The value of a / b is 2

The value of a % b is 0

The value of a++ is 10

The value of a-- is 11

The value of ++a is 11

The value of --a is 10

Following are the comparison operators in C

The value of a == b is 0

The value of a != b is 1

The value of a >= b is 1

The value of a <= b is 0

The value of a > b is 1

The value of a < b is 0

Following are the logical operators in C
The value of this logical and operator ((a==b) && (a<b)) is:0
The value of this logical or operator ((a==b) || (a<b)) is:0
The value of this logical not operator (!(a==b)) is:1