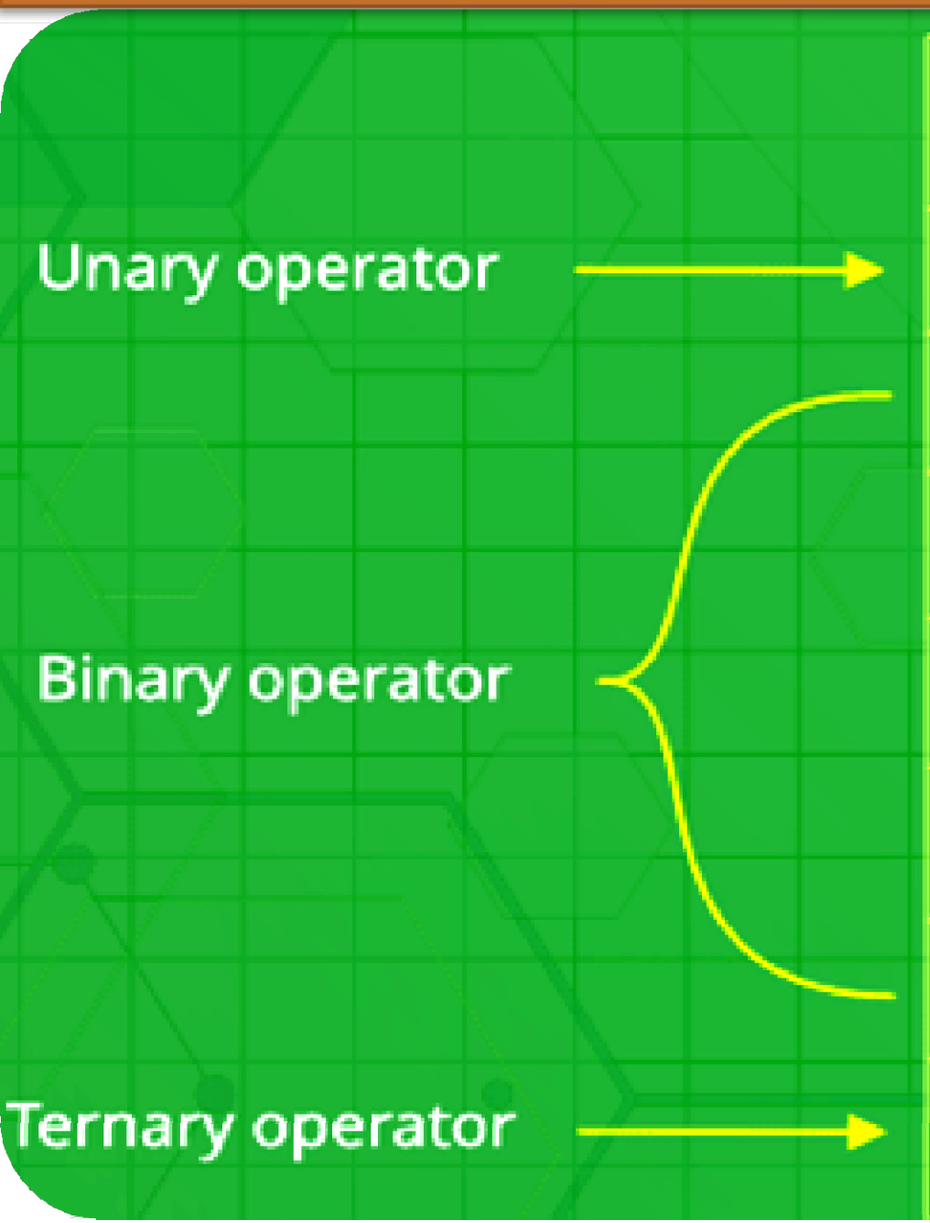
C++ OPERATORS

C++ OPERATORS



Operator	Type
+ +, - -	Unary operator
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
&&, ||, !	Logical operator
&, |, <<, >>, ~, ^	Bitwise operator
=, +=, -=, *=, /=, %=	Assignment operator
?:	Ternary or conditional operator

Increment and Decrement Operators

- C++ also provides increment and decrement operators: ++ and -- respectively.
- ++ increases the value of the operand by **1**,
- while -- decreases it by **1**.

++ and --

num++ is equivalent to num=num+1;

num -- is equivalent to num=num-1;

increment and decrement operators

- Increment (++) and Decrement (- -)
- **Pre Increment is ++x**
- (The value is incremented first and then applied)
- **Post Increment is x++**
- (First Value is Applied and then value is incremented)
- **Pre Decrement - -x**
- (The value is decremented first and then applied)
- **Post Decrement x- -**
- (First value is applied and then value is decremented)

increment and decrement operators

```
main()
{
int a = 10, b = 100;
cout <<"++a = “<<++a << endl;
cout <<"++b = “<<++b << endl;
cout <<"a-- = “<<a-- << endl;
cout <<“--b = “<<--b << endl;
}
```

C++ Arithmetic Operators

• C++ Arithmetic Operators

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Operation (Remainder after division)

Arithmetic Operators

```
#include <iostream.h>
#include <conio.h>
void main()
{
int a = 7, b = 2;
cout << "a + b = " << (a + b) << endl;
cout << "a - b = " << (a - b) << endl;
cout << "a * b = " << (a * b) << endl;
cout << "a / b = " << (a / b) << endl;
cout << "a % b = " << (a % b) << endl;
}
```

Output

a + b = 9

a - b = 5

a * b = 14

a / b = 3

a % b = 1

SIMPLE INTEREST

```
void main()
{
double p,r,si,t;
cout<<"enter the value of principal,rate and time" <<endl;
cin>>p>>r>>t;
si=(p*r*t)/100;
cout<<"Amount = Rs."<<p<<endl;
cout<<"Rate = Rs."<<r<<endl;
cout<<"Time = year"<<t<<endl;
cout<<"Simple interest ="<<si<<endl;
getch();
}
```

C++ Assignment Operators

Operator	Example	Equivalent to
=	`a = b;`	`a = b;`
+=	`a += b;`	`a = a + b;`
-=	`a -= b;`	`a = a - b;`
*=	`a *= b;`	`a = a * b;`
/=	`a /= b;`	`a = a / b;`
%=	`a %= b;`	`a = a % b;`

Example of Assignment Operators

```
main()
{
int a = 10; int b = 20;
b = a;
cout<<“ = Output: "<<b;
a += b;
cout<<"+= Output: "<<a;
a -= b;
cout<<"-= Output: "<<a;
a *= b;
cout<<"*= Output: "<<a;
a /= b;
cout<<"/= Output: "<<a;
a %= b;
cout<<"%= Output: "<<a;
}
```

C++ Relational Operators

C++ Relational Operators

Operator	Meaning	Example
`==`	Is Equal To	`3 == 5` gives us **false**
`!=`	Not Equal To	`3 != 5` gives us **true**
`>`	Greater Than	`3 > 5` gives us **false**
`<`	Less Than	`3 < 5` gives us **true**
`>=`	Greater Than or Equal To	`3 >= 5` give us **false**
`<=`	Less Than or Equal To	`3 <= 5` gives us **true**

```
// C++ program demonstrating ! operator truth table
void main()
 {
    int a = 5;
    cout << (a == 0) << endl;
    cout << (a == 5) << endl;
    }
```

Example : Relational Operators

```
Void main()
{
int a=3, b=5;
cout << (a == b)<<endl;
cout << (a!= b)<<endl;
cout << (a > b)<<endl;
cout << (a < b)<<endl;
cout << (a >= b)<<endl;
cout << (a <= b)<<endl;
}
```

C++ Logical Operators

C++ Logical Operators

- Logical operators are used to check whether an expression is **true** or **false**.
- If the expression is **true**, it returns **1** whereas

Operator	Example	Meaning
&&	expression1 **&&** expression2	Logical AND. True only if all the operands are true.
||	expression1 **||** expression2	Logical OR. True if at least one of the operands is true.
!	**!**expression	Logical NOT. True only if the operand is false.

Relational Operators

```
// C++ program demonstrating || operator truth table
#include <iostream>
void main()
{
int a = 5, b = 9;
cout << ((a == 0) || (a > b)) << endl;
cout << ((a == 0) || (a < b)) << endl;
cout << ((a == 5) || (a > b)) << endl;
cout << ((a == 5) || (a < b)) << endl;
cout << ((a == 0) && (a > b)) << endl;
cout << ((a == 0) && (a < b)) << endl;
cout << ((a == 5) && (a > b)) << endl;
cout << ((a == 5) && (a < b)) << endl;
}
```

C++
Bitwise Operators

C++ Bitwise Operators

- In C++, bitwise operators are used to perform operations on individual bits.
- They can only be used char and int data types.

Operator	Description
`&`	Binary AND
`|`	Binary OR
`^`	Binary XOR
`~`	Binary One's Complement
`<<`	Binary Shift Left
`>>`	Binary Shift Right

Example - Bitwise AND operator

```
#include <iostream>
void main()
{
int a = 5, b = 9;
cout << "a = " << a << endl;
cout << "b = " << b << endl;
cout << "a & b = " << (a & b) << endl;
getch();
}
```

Output

```
a = 5
b = 9
a & b = 1
```

Example - Bitwise OR operator

```
#include <iostream>
Void main()
{
int a = 5, b = 9;
cout << "a = " << a << endl;
cout << "b = " << b << endl;
cout << "a | b = " << (a | b) << endl;
cout << "a & b = " << (a & b) << endl;
}
```

Output

a = 5

b = 9

a | b = 13

a & b = 1

Example - Bitwise XOR operator

```
#include <iostream>
void main()
{
int a = 5, b = 9;
cout << "a = " << a << endl;
cout << "b = " << b << endl;
cout << "a ^ b = " << (a ^ b) << endl;
}
```

Output

a = 5

b = 9

a ^ b = 12

Example Bitwise Complement operator

```
#include <iostream>
main()
{
int num1 = 35;
int num2 = -150;
cout << "~(" << num1 << ") = " << (~num1) <<
    endl;
cout << "~(" << num2 << ") = " << (~num
    endl;
}
```

Output

~(35) = -36

~(-150) = 149

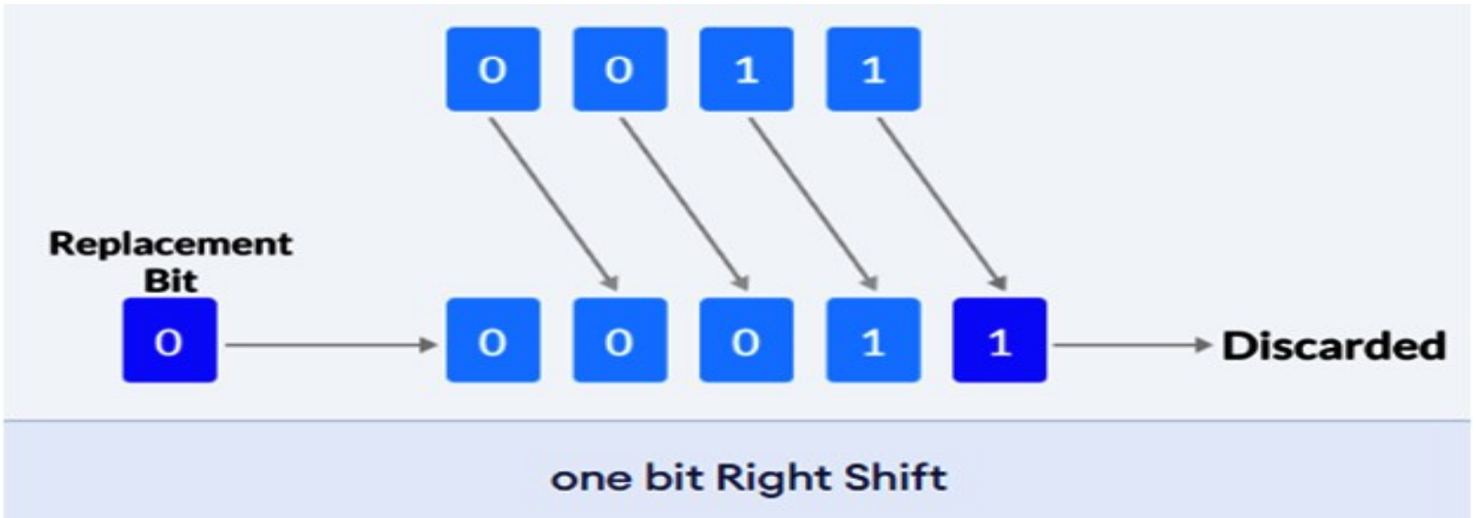
C++ Shift Operators

There are two shift operators in C++ programming:

- Right shift operator >>
- Left shift operator <<

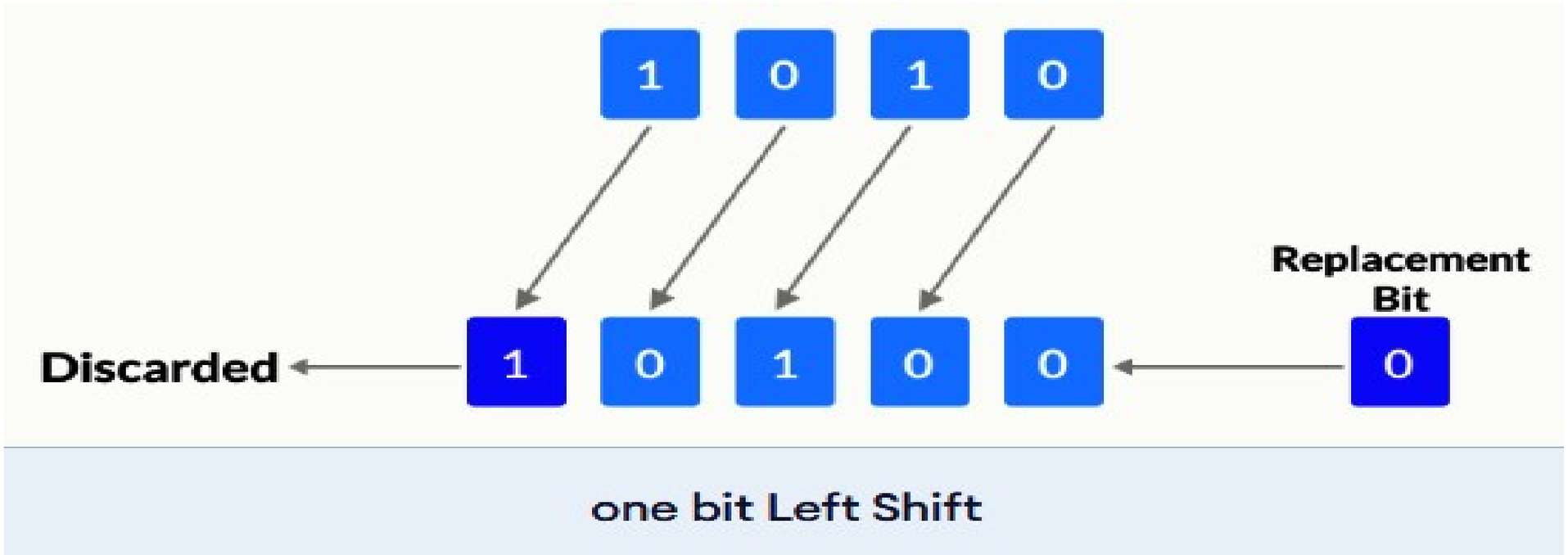
C++ Right Shift Operator

- The **right shift operator** shifts all bits towards the right by a certain number of **specified bits**. It is denoted by >>.
- When we shift any number to the right, the **least significant bits** are discarded, while the **most significant bits** are replaced by zeroes.



C++ Left Shift Operator

- The **left shift operator** shifts all bits towards the left by a certain number of **specified bits**. It is denoted by <<.



C++ Shift Operators

```
void main()
{
    int a = 5, b = 9;
    cout<<"a = " << a <<","<< " b = " << b <<endl;
    cout << "a & b = " << (a & b) << endl;
    cout << "a | b = " << (a | b) << endl;
    cout << "a ^ b = " << (a ^ b) << endl;
    cout << "~(" << a << ") = " << (~a) << endl;
    cout<<"b << 1" <<" = "<< (b << 1) <<endl;
    cout<<"b >> 1 "<<"= " << (b >> 1 )<<endl;
    }
```

C++ Shift Operators

```
void main()
{
    int a = 5, b = 9;
    cout<<"a = " << a <<","<< " b = " << b <<endl;
    cout<<"b << 1" <<" = "<< (b << 1) <<endl;
    cout<<"b >> 1 "<<"= " << (b >> 1 )<<endl;
    }
```

Ternary operator

Conditional or Ternary Operator (?:) in C/C++

Syntax

condition ? value_if_true : value_if_false

Program to Find Largest Among Two Numbers Using Ternary Operator

```
void main()
{
int a = 5, b = 10, c;
c = (a > b) ? a : b;
cout<<"Largest number between ”<<a << “ & ”<< b <<“is “ << c;
}
```

C++ COMMENTS

C++ Comments

- C++ comments are hints that a programmer can add to make their code easier to read and understand. They are completely ignored by C++ compilers.

There are two ways to add comments to code:

- // - Single Line Comments
- /* */ -Multi-line Comments

- **Why use Comments?**
- If we write comments on our code, it will be easier for us to understand the code in the future. Also, it will be easier for your fellow developers to understand the code.
- **Note:** Comments shouldn't be the substitute for a way to explain poorly written code in English. We should always write well-structured and self-explanatory code. And, then use comments.

The sizeof() operator

Sizeof ( )

sizeof() call to display the size (in bytes) of the standard C data types (char,int, long…)

sizeof (char)	**=1**
sizeof (int)	**=2**
sizeof (float)	**=4**
sizeof (double)	**=8**
sizeof (long long)	**=8**

sizeof( )

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a; double c; char d;
cout<<"Size of int ="<<sizeof(a)<<"bytes\n";
cout<<"Size of double="<<sizeof(c)<<"bytes\n";
cout<<"Size of char="<<sizeof(d)<<"bytes\n";
getch();
}
```

Conversion of Temperature from Celsius degree into Fahrenheit

```
#include<iostream.h>
#include<conio.h>
void main()
{
double c,f;
cout<<"Enter temperature in degree:";
cin>>c;
f=(1.8*c)+32;
cout<<"Temperature in degree Fahrenhieit:"<<f;
getch();
}
```

THANKS