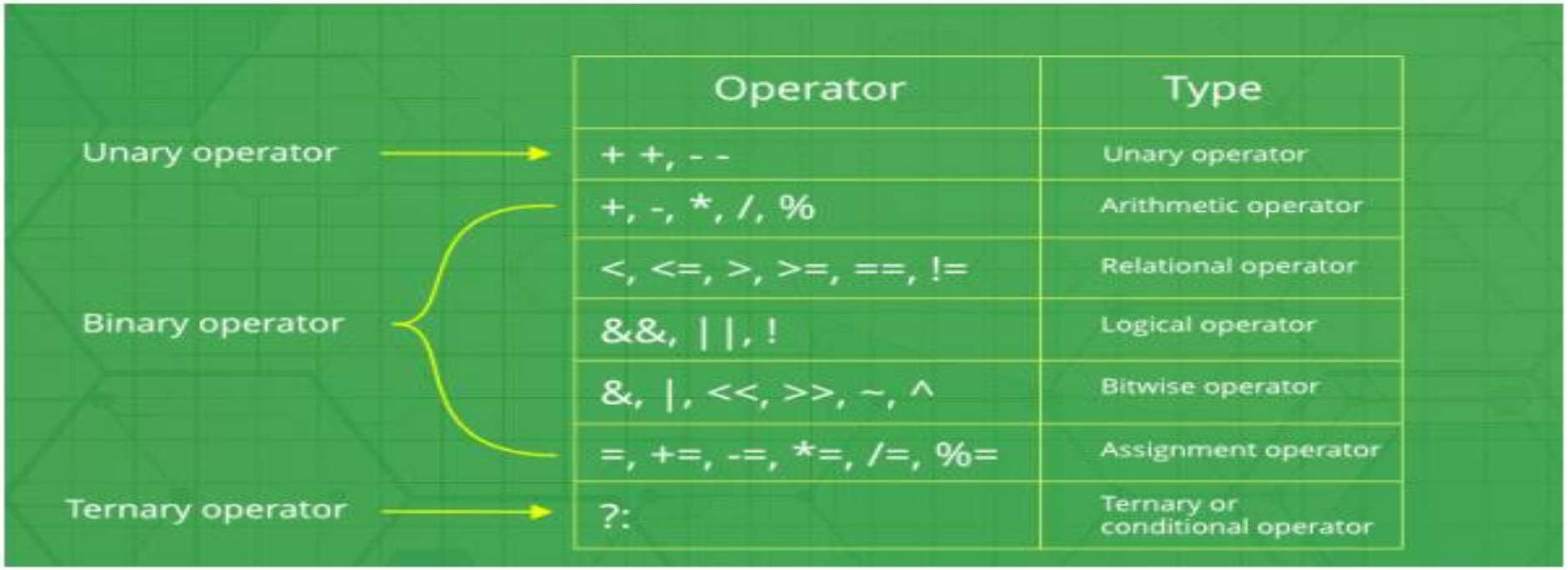
object-oriented programming

(OOP)

KIAN _ ACADEMY

Operator Overloading :-

Operators in C++



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

Example

```
#include<iostream>
using namespace std;
class Complex
{
private:
   int real;
public:
   Complex()
   {

   }
   Complex(int r)
   {
      real = r;
   }
   void print()
   {
      cout << real<<endl;
   }
};

int main()
{
   Complex c1(10);
    c1++;//error
    c1.print();
}
```

1-Unary Operator Overloading : It works for only one operand.

Syntax : Unary Function Definition

```
return_type operator_keyword operator_symbol()
{
    //body
}
```

Example

```
#include<iostream>
#include<string.h>
using namespace std;

class Overload
{
   int a,b;
public:
   void input()
   {
      cout<<"Input 1: "<<endl;
      cin>>a;
      cout<<"Input 2: "<<endl;
      cin>>b;
   }
   void operator++()
   {
      a++;
      b++;
      cout<<"Incremented values:\n";
      cout<<a<<'\t'<<b;
   }
};

int main()
{
   Overload obj;
   obj.input();
   ++obj;
   return 0;
}
```

Overloading Unary Operators

PostFix Notation :

Syntax : Unary Function Definition

```
return_type operator_keyword operator_symbol (int) //must integer
{
    //body
}
```

Example

```
#include<iostream>
#include<string.h>
using namespace std;
class Overload
{
   int a,b;
public:
   void input()
   {
      cout<<"Input 1: "<<endl;
      cin>>a;
      cout<<"Input 2: "<<endl;
      cin>>b;
   }
```

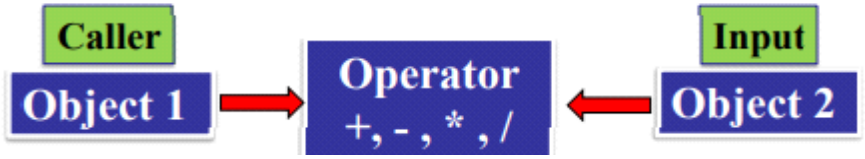
```
void operator++(int)
   {
      a++;
      b++;

cout<<"Incremented
values:\n";
      cout<<a<<'\t'<<b;
   }
};
```

```
int main()
{
   Overload obj;
   obj.input();
   obj++;
   return 0;
}
```

2 - **Binary Operator Overloading**: It works for two operands.

Systenx :



Example

```
#include<iostream>
using namespace std;
class Complex
{
private:
    int real, imag;
public:
    Complex(int r, int i)
    {
        real = r;
        imag = i;
    }
    void print()
    {
        cout << real << "\t" << imag << '\n';
    }
};
int main()
{
    Complex c1(10, 5), c2(2, 4);
    Complex c3 = c1 + c2;//error
    c3.print();
}
```

Example from the last session

```
#include<iostream>
using namespace std;
class Complex
{
private:
    int real, imag;
public:
    Complex(int r, int i)
    {
        real = r;
        imag = i;
    }
    Complex()
    {
    }
    void print()
    {
        cout << real << "\t" << imag << '\n';
    }
Complex add(Complex c)
    {
        Complex result;
        result.real = real + c.real;
        result.imag = imag + c.imag;
        return result;
    }
};
int main()
{
    Complex c1(10, 5), c2(2, 4);
    c1.print();
cout<<"**********************"<<endl;
    c2.print();
cout<<"**********************"<<endl;
    Complex c3=c2.add(c1);
    c3.print();
}
```

The output is :

```
10      5
******************************
2       4
******************************
12      9
```

Example from the last session

```
#include<iostream>
using namespace std;
class Complex
{
private:
   int real, imag;
public:
   Complex(int r, int i)
   {
      real = r;
      imag = i;
   }
   Complex()
   {
   }
 void print()
   {
      cout << real << "\t" << imag << '\n';
   }
 Complex operator +(Complex c)
   {
      Complex result;
      result.real=real+c.real;
      result.imag=imag+c.imag;
      return result;
   }
};
int main()
{
   Complex c1(10, 5), c2(2, 4);
   c1.print();
cout<<"**********************"<<endl;
   c2.print();
   cout<<"**********************"<<endl;
   Complex c3=c1+c2;
   c3.print();
}
```

****

The output is :

10 5

****

2 4

****

12 9

Exam questions:

- **(1)** Which of the following operators cannot be overloaded

- (A) . (Member Access or Dot operator)
- (B) ?: (Ternary or Conditional Operator)
- (c) :: (Scope Resolution Operator)
- (D) * (Pointer-to-member Operator)
- (E) All of the above

- **(2)** Which of the following operators are overloaded by default by the compiler in every user defined classes even if user has not written?

1) Comparison Operator (==)

2) Assignment Operator (=)

- A) - Both 1 and 2
- B) - Only 1
- C) - Only 2
- D) - None of the two

- **(3)** Which of the following operators should be preferred to overload as a global function rather than a member method?

- A) Postfix ++
- B) Comparison Operator
- C) Insertion Operator <<
- D) Prefix++

- **(4)** How does C++ compiler differs between overloaded postfix and prefix operators?

- A) C++ doesn't allow both operators to be overloaded in a class
- B) A postfix ++ has a dummy parameter
- C) A prefix ++ has a dummy parameter
- D) By making prefix ++ as a global function and postfix as a member function

- **(5)** Which of the following operator functions cannot be global, i.e., must be a member function.

- A) new
- B) delete
- C) Converstion Operator
- D) All of the above

Answer the questions :-

- 1 - e
- 2 - c
- 3 - c
- 4 - b
- 5 - c