Operator Overloading

- Operator overloading is to allow the same operator to be bound to more than one implementation, depending on the types of the operands.

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Operator overloading

Consider the following class:

```
class Complex{
private:
 double real, img;
public:
 Complex add(const Complex &);
 Complex subtract(const Complex &);
 Complex multiply(const Complex &);
 …
}
```


Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Operator overloading

Function implementation:

```
Complex Complex::add(
  const Complex & c1){
 Complex t;
 t.real = real + c1.real;
 t.img  = img  + c1.img;
 return t;
}
```


Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Operator overloading

- The following statement:

```
Complex c3 = c1.add(c2);
```

Adds the contents of **`c2`** to **`c1`** and assigns it to **`c3`** (copy constructor)

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Operator overloading

- To perform operations in a single mathematical statement e.g:

```
c1 + c2 + c3 + c4
```

- We have to explicitly write:

```
c1.add(c2.add(c3.Add(c4)))
```


Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Operator overloading

- Alternative way is:

```
t1 = c3.add(c4);
t2 = c2.add(t1);
t3 = c1.add(t2);
```


Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Operator overloading

- If the mathematical expression is big:
  - Converting it to C++ code will involve complicated mixture of function calls
  - Less readable
  - Chances of human mistakes are very high
  - Code produced is very hard to maintain

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Operator overloading

►C++ provides a very elegant solution:

"***Operator overloading***"

►C++ allows you to overload common operators like **+**, **–** or ***** etc…

►Mathematical statements don't have to be explicitly converted into function calls

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Where it required

- Application for mathematical functions of addition, subtraction, multiplication and division. Complex number is one example of it.
- In case of Date class, the operators can be effectively used to get the future or past dates.
- Sometimes in case of non-mathematical manipulation. The example of String class to manipulate strings help us understand it in a better way. The operator + can be used to concatenate two strings.

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Operator Semantics

We have to write a function and make use of the operator semantics correctly while implementing the function.

This means that the function written to overload + operator should do addition or concatenation of strings in case of String objects.

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Restrictions

- The operator overloading functions for overloading (), [], -> or the assignment (=) Operators must be declared as class members.
- The arity (number of operands) cannot be changed. If you are overloading an operator that requires two operands e.g. *. It cannot be used as a unary operator that requires one operand.
- No new operators can be created.
- Overloading can’t be performed for the built-in (sometimes called primitive or native) data types. For example, we cannot change how two integers are added. That means that operators are overloaded to use with defined data types like classes.

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Restrictions Cont…

- Precedence of an operator cannot be changed. For example, the * has higher precedence than +. This precedence cannot be changed.

  c1*c2+c3

  c3+c2*c1

- Associativity of an operator cannot be changed. If some operator is right associative, it cannot be changed to be left associative.
  Following arithmetic expression always is evaluated from left to right:

  c1 + c2 + c3 + c4

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Operator overloading

►C++ automatically overloads operators for pre-defined types

►Example of predefined types:

- `int`
- `float`
- `double`
- `char`
- `long`

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Operator overloading

- List of operators that can be overloaded in C++:

```
new   delete     new[]      delete[]
+     -     *     /     %     ^     &     |     ~
!     =     <     >     +=    -=    *=    /=    %=
^=    &=    |=    <<    >>    >>=   <<=   ==    !=
<=    >=    &&    ||    ++    --    ,     ->*   ->
()    []
```


Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Operator overloading

- List of operators that can't be overloaded:

  `.` `.*` `::` `?:` `#` `##`

- Reason: They take name, rather than value in their argument except for **?:**

- **?:** is the only ternary operator in C++ and can't be overloaded

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Binary operators

- Binary operators act on two quantities
- Binary operators:

+	-	*	/	%	^	&	|	~
!	=	<	>	+=	-=	*=	/=	%=
^=	&=	|=	<<	>>	>>=	<<=	==	!=
<=	>=	&&	||	,	->*	->		

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.

Binary operators

- Drawback of void return type:
  - Assignments and cascaded expressions are not possible
  - Code is less readable
  - Debugging is tough
  - Code is very hard to maintain

Copyright © 2021, Virtual Academy of Pakistan(www.virtualacademy.com.pk). All rights reserved.