OOPS, Quiz 3

14/9/2023

Q1 . How many arguments are required in the definition of an overloaded unary operator?

Ans:
For an overloaded unary operator that is a member function of the same class, no additional arguments are required. The operator function should have no parameters because it operates on the current object using the `this` pointer implicitly.

However, when overloading a unary operator using a friend function (which is not a member of the class), you still need one argument, which is the object on which the operator is applied. This argument is explicitly passed to the friend function.

So, in summary:

1. For member function-based overloading: No additional arguments are needed; the object itself is implicitly passed as `this`.
2. For friend function-based overloading: You need one additional argument, which is the object you are operating on.

Q2.When used in prefix form, what does the overloaded ++ operator do differently from what it does in postfix form?

Ans :
The overloaded `++` operator behaves differently in prefix and postfix forms, and the key distinction lies in the order of operations and the value returned:

- Prefix increment increments the object before using its value and returns the modified object.
- Postfix increment uses the current value, returns a copy of the original value, and then increments the object afterward.

Q3.When you overload an arithmetic assignment operator, the result

a. goes in the object to the right of the operator.
b. goes in the object to the left of the operator.
c. goes in the object of which the operator is a member.
d. must be returned.

Ans: (b.) the result goes in the object to the left of the operator.
clarifying option d =>
it is not necessary to return the modified object but it is a preferred way and not returning it will deprive us of some functionalities like chain assignment (like Point p1, p2, p3; p1 = p2 = p3;).In case of p1 = p2;We do modify the p1 so returning or not returning does not affect us, but in chaining first p2 = p3 will happen and the return value for to be assigned to p1 would come out to be void which will throw compile error.

Q3. Define a class **STRPLUS.** Include one data member which is a **string**. Write the complete definition of an overloaded ++ operator that works with the String class from the STRPLUS example and has the effect of changing its operand to uppercase. You can use the library function toupper() (header file CCTYPE), which takes as its only argument the character to be changed and returns the changed character (or the same character if no change is necessary).

Ans :

```
#include <iostream>
#include <string>
#include <cctype>
namespace std;
class STRPLUS {
   string _str;
public:
    STRPLUS() : _str(“”) {}
   STRPLUS(const string& s) : _str(s) {}
   STRPLUS& operator++() {
      for (char& c : _str) {
         c = toupper(c);
      }
      return *this;
   }
};
```

Q4. To convert from a user-defined class to a basic type, you would most likely use

a. built-in conversion operator.

b. one-argument constructor.

c. overloaded = operator.

d. conversion operator that’s a member of the class.

Ans: (d) conversion operator that’s a member of the class.

Q5. True or false: The statement objA=objB; will cause a compiler error if the objects are of different classes.

Ans : True;

Q6. To convert from a basic type to a user-defined class, you would most likely use

A. a built-in conversion operator.
B. a one-argument constructor.
C. an overloaded = operator.
D. a conversion operator that’s a member of the class.

Ans : (B.) a one-argument constructor.

Q7. True or false: If you've defined a constructor to handle definitions like **aclass obj = intvar;** you can also make statements like obj = intvar;.

Ans :
True
Like in code =>

```
MyClass obj;
obj = 5;
The first line calls the default constructor and the second line calls
the copy constructor to make an object with integer(5) and this does
not throw any compilation error.
```

Q8.True or false: The compiler won't object if you overload the * operator to perform division

Ans : True;

Q9.Operator overloading is

A. making C++ operators work with objects.
B. giving C++ operators more than they can handle.
C. giving new meanings to existing C++ operators.
D. making new C++ operators.

Ans: C (giving new meanings to existing C++ operators.)
