EXPERIMENT NO-7

Title: Write a program to implement operator overloading

Objectives:

1. Study of copy constructor.
2. Study of operator overloading.
3. Overloading assignment operator.
4. Implementation of unary, binary and array index/subscript operators.

Theory:

Operator overloading is one of the most exciting features of OOP. It can transform complex, obscure program listing into intuitively obvious ones.
For example: d3. addobjects(d1, d2);
Can be changed to much more readable d3 = d1 + d2;

The term operator loading refers to giving the normal C++ operators such as +, *, additional meanings when they are applied to user defined data types. It gives you the opportunity to redefine basic operations in the C++ language.
In order to overload operators, it is necessary to write functions (that is, the header and body). The name of the function that overloads an operator is the reserved word operator followed by the operator to be overloaded. For example, the name of the function to overload the operator >= is:
operator >=
Operator function: The function that overloads an operator.

The syntax of the heading for an operator function is:

returnType operator operatorSymbol(**const** formal parameter list) **;**

In C++, **operator** is a reserved word.

Restrictions on Overloading an Operator:

1. The precedence of an operator cannot be changed.
2. The associativity cannot be changed. (For example, the associativity of the arithmetic operator addition is from left to right, and it cannot be changed.)
3. Default parameters cannot be used with an overloaded operator.
4. The number of parameters an operator takes cannot be changed.
5. You cannot create new operators. Only existing operators can be overloaded.
6. The operators that cannot be overloaded are:
   . .* :: ?: sizeof
7. The meaning of how an operator works with built-in types, such as int, remains the same.
8. Operators can be overloaded either for objects of the user-defined types, or for a combination of objects of the user-defined type and objects of the built-in type.

Procedure:

1. Define an **EnhancedArray** class that supports additions, subtraction, increment, decrement operations on each array element using operator overloading.
2. Define no-argument and parametric constructor for the class.
3. Allocate memory for the array dynamically in the constructor.
4. Define Destructor and copy constructor for the class.
5. Deallocate the dynamically allocated memory for array in the destructor.
6. Provide overloaded assignment operation.
7. Provide overloaded operator for addition and subtraction of two enhanced array objects. This will add two arrays element by element.
8. Provide increment and decrement operation on the array that increments or decrements every element of the array.
9. Provide array subscript or index operator on an EnhancedArray object.

Keywords:

Operator overloading, copy constructor, destructor, references