Lab12
Operator Overloading

OBJECT ORIENTED PROGRAMMING

BSCS-SPRING-2022

Operator Overloading:

we can change the way operators work for user-defined types like objects and structures. This is known as operator overloading.

You can have multiple definitions for the same function name in the same scope. The definition of the function must differ from each other by the types and/or the number of arguments in the argument list. You cannot overload function declarations that differ only by return type.

Syntax:

```
class className {

   ... .. ...

   public

     returnType operator symbol (arguments) {

          ... .. ...

     }

   ... .. ...

};
```

Here,

- returnType is the return type of the function.
- operator is a keyword.
- symbol is the operator we want to overload. Like: +, <, -, ++, etc.
- arguments is the arguments passed to the function.

When you call an overloaded **function** or **operator**, the compiler determines the most appropriate definition to use, by comparing the argument types you have used to call the function or operator with the parameter types specified in the definitions. The process of selecting the most appropriate overloaded function or operator is called **overload resolution**.

Exapmle:

we can overload an operator ‘+’ in a class for Complex Numbers….

```
#include<iostream>
using namespace std;

class Complex {
private:
   int real, imag;
public:
   Complex(int r = 0, int i = 0) {real = r;   imag = i;}

   // This is automatically called when '+' is used with
   // between two Complex objects
   Complex operator + (Complex const &obj) {
        Complex res;
        res.real = real + obj.real;
        res.imag = imag + obj.imag;
        return res;
   }
   void print() { cout << real << " + i" << imag << '\n'; }
};

int main()
{
   Complex c1(10, 5), c2(2, 4);
   Complex c3 = c1 + c2;
   c3.print();
}
```

Output:

```
12+i9
```

List of operators, can be overloaded are:

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

List of operators, cannot be overloaded are:

Some of the operators cannot be overloaded. These operators are like below

- . Member access or dot operator
- ? : Ternary or conditional operator
- :: Scope resolution operator
- .* Pointer to member operator
- Sizeof The object size operator
- typeid Object type operator

These operators cannot be overloaded because if we overload them it will make serious programming issues.

For an example the sizeof operator returns the size of the object or datatype as an operand. This is evaluated by the compiler. It cannot be evaluated during runtime. So we cannot overload it.

Tasks

Problem 1:

Create a class Money that represents a money valuer (combination of dollars and cents).

Overload the following operator.

Functions:

i) **Money()** // Default Constructor Initializes dollars and cents to zero
ii) **Money(int dollar, int cents)** // Parameterized Constructor Update dollar and cents accordingly
iii) **Money& operator= (const Money& right)** Overload Assignment operator to assign Money objects to each other
iv) **bool operator== (const Money& right)** Overload Equal operator to check if Money objects are equal or not
v) **Money& operator+= (const Money& right)** Overload Addition operator to Add Money objects to each other
vi) **Money& operator-= (const Money& right)** Overload Subtraction operator to Subtract smaller Money object from larger Money object.
vii) **Money& operator*= (const Money& right)** Overload Multiplication operator to multiply Money object with an integer number
viii) **Money& operator/= (const Money& right)** Overload Division operator to divide Money object with an integer number
ix) ~Money()

Problem 2:

Design a class String having a data member as char *s, int size, and define the appropriate function members,

you task is to

1. **Overload binary operator '+' to concatenate two strings.** For example: If you pass string "hello" and "world" for s1 and s2 respectively, it will concatenate both into s3, and display the output as "helloworld".
2. **Overload character indexing or subscripting []:** For example, you can use str[n] to get the char at index n; or str[n] = c to modify the char at index n. Take note that [] operator does not perform indexbound check, i.e., you have to ensure that the index is within the bounds.

Problem 3:

Implement 3-dimensional matrix class, and overload the Sum '+' operator to add two matrices.

Your solution will have following methods:

- **i.** Default Constructor
- **ii.** Parameterized Constructor
- **iii.** Destructor
- **iv.** To insert data in matrix
- **v.** Overload + operator to add two matrix objects