Programming with C and C++

CSC-101 (Lecture 35)

Dr. R. Balasubramanian
Professor
Department of Computer Science and Engineering
Mehta Family School of Data Science and Artificial Intelligence
Indian Institute of Technology Roorkee
Roorkee 247 667

bala@cs.iitr.ac.in
https://faculty.iitr.ac.in/cs/bala/

Copy Constructor

```
1  #include <iostream>
2  using namespace std;
3  class Example
4  {
5     public:
6      int a;
7      Example(int x) // parameterized constructor
8      {
9        a=x;
10     }
11     Example(Example &ob)  // copy constructor
12     {
13       a = ob.a;
14     }
15 };
16
```

https://ideone.com/e8SUAv

```
17  int main()
18  {
19    Example e1(36); // Calling the parameterized constructor
20    Example e2(e1); //  Calling the copy constructor
21    cout<<e2.a;
22    return 0;
23  }
24
```

stdout

36

Destructors in C++

https://ideone.com/U6qGkR

```
1   #include <iostream>
2   using namespace std;
3   class Employee
4    {
5      public:
6           Employee()
7           {
8               cout<<"Constructor Invoked"<<endl;
9           }
10          ~Employee()
11          {
12              cout<<"Destructor Invoked"<<endl;
13          }
14  };
15
```

```
16  int main(void)
17  {
18      Employee e1; //creating an object of Employee
19      cout<<&e1<<endl;
20      Employee e2; //creating an object of Employee
21      cout<<&e2<<endl;
22      return 0;
23  }
```

```
Constructor Invoked
0x7ffddbec0c06
Constructor Invoked
0x7ffddbec0c07
Destructor Invoked
Destructor Invoked
```

- Multiplication of two complex numbers by overloading * operator in c++

https://ideone.com/1Evevf

```
#include <iostream>

class Complex {
private:
    double real;
    double imaginary;

public:
    Complex(double r = 0, double i = 0) : real(r), imaginary(i) {}

    Complex operator*(const Complex& other) {
        // (a + bi) * (c + di) = (ac - bd) + (ad + bc)i
        double resultReal = (real * other.real) - (imaginary * other.imaginary);
        double resultImaginary = (real * other.imaginary) + (imaginary * other.real);
        return Complex(resultReal, resultImaginary);
    }

```

```
18      void display() {
19          std::cout << real << " + " << imaginary << "i" << std::endl;
20      }
21  };
22
23  int main() {
24      Complex num1(2, 3);
25      Complex num2(1, 2);
26
27      Complex result = num1 * num2;
28
29      std::cout << "Result of multiplication: ";
30      result.display();
31
32      return 0;
33  }
```

stdout

```
Result of multiplication: -4 + 7i
```

- Addition of two matrices using operator overloading in c++

https://ideone.com/KzWSxh

```
1  #include <iostream>
2
3  using namespace std;
4
5  class Matrix {
6  public:
7    Matrix(int rows, int cols) : rows(rows), cols(cols) {
8      data = new int*[rows];
9      for (int i = 0; i < rows; i++) {
10       data[i] = new int[cols];
11       for (int j = 0; j < cols; j++) {
12         data[i][j] = 0;
13       }
14     }
15   }
16
```

```
17    ~Matrix() {
18      for (int i = 0; i < rows; i++) {
19        delete[] data[i];
20      }
21      delete[] data;
22    }
23
24    Matrix operator+(const Matrix& other) {
25      Matrix result(rows, cols);
26      for (int i = 0; i < rows; i++) {
27        for (int j = 0; j < cols; j++) {
28          result.data[i][j] = data[i][j] + other.data[i][j];
29        }
30      }
31      return result;
32    }
33
```

```
34    void print() {
35      for (int i = 0; i < rows; i++) {
36        for (int j = 0; j < cols; j++) {
37          cout << data[i][j] << " ";
38        }
39        cout << endl;
40      }
41    }
42
43  public:
44    int rows;
45    int cols;
46    int** data;
47  };
```

```
48
49  int main() {
50      Matrix m1(2, 2);
51      m1.data[0][0] = 1;
52      m1.data[0][1] = 2;
53      m1.data[1][0] = 3;
54      m1.data[1][1] = 4;
55
56      Matrix m2(2, 2);
57      m2.data[0][0] = 5;
58      m2.data[0][1] = 6;
```

```
59      m2.data[1][0] = 7;
60      m2.data[1][1] = 8;
61
62      Matrix result = m1 + m2;
63
64      result.print();
65
66      return 0;
67  }
68
```

stdout

```
6 8
10 12
```

Try this

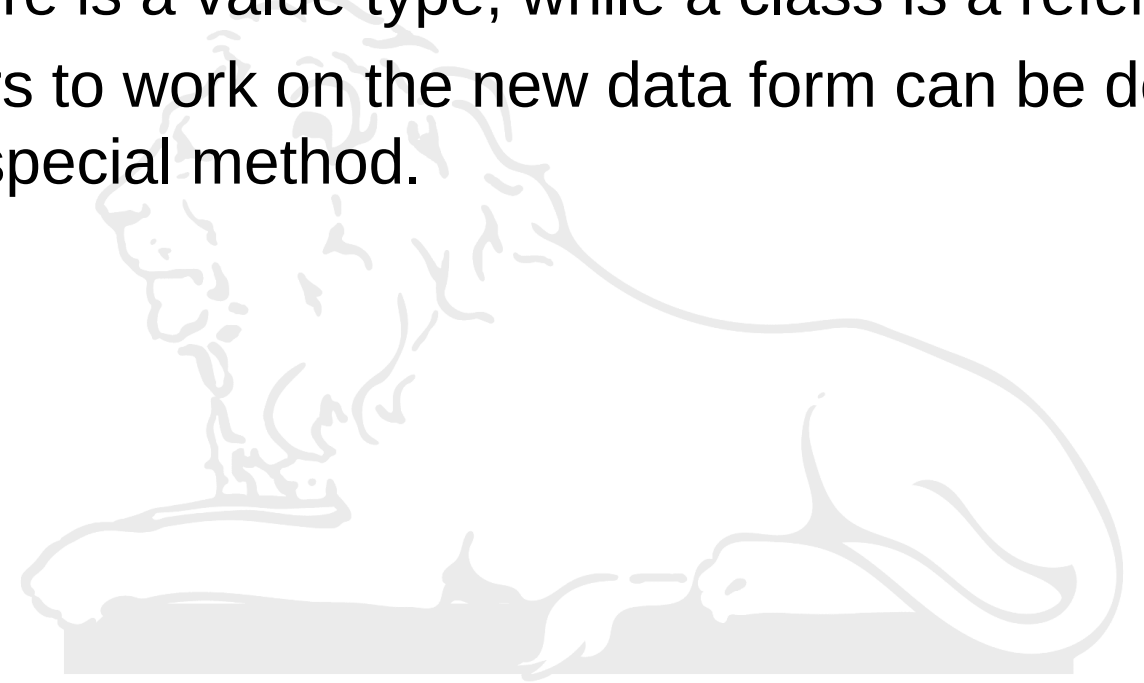
- Multiplication of two matrices using operator overloading in c++

Difference between Structures and Classes

- By default, all the members of the structure are public. In contrast, all members of the class are private.
- The structure will automatically initialize its members. In contrast, constructors and destructors are used to initialize the class members.
- When a structure is implemented, memory allocates on a stack. In contrast, memory is allocated on the heap in class.
- Variables in a structure cannot be initialized during the declaration, but they can be done in a class.

Difference between Structures and Classes

- There can be no null values in any structure member. On the other hand, the class variables may have null values.
- A structure is a value type, while a class is a reference type.
- Operators to work on the new data form can be described using a special method.



Difference between Structures and Classes

Features	Structure	Class
Basic	If no access specifier is specified, all members are set to 'public'.	If no access specifier is defined, all members are set to 'private'.
Instance	Structure instance is called the 'structure variable'.	A class instance is called 'object'.
Inheritance	It does not support inheritance.	It supports inheritance.
Memory Allocated	Memory is allocated on the stack.	Memory is allocated on the heap.
Nature	Value Type	Reference Type

Difference between Structures and Classes

Purpose	Grouping of data	Data abstraction and further inheritance.
Usage	It is used for smaller amounts of data.	It is used for a huge amount of data.
Null values	Not possible	It may have null values.
Requires constructor and destructor	It may have only parameterized constructor.	It may have all the types of constructors and destructors.

Access Specifiers in C++

```
class MyClass {
public:
    int publicVar;      // can be accessed from anywhere
    void publicFunc() { // can be accessed from anywhere
        cout << "This is a public function" << endl;
    }
private:
    int privateVar;     // can only be accessed from within the class

    void privateFunc() {// can only be accessed from within the class
        cout << "This is a private function" << endl;
    }
```

```
protected:
    int protectedVar;
// can only be accessed from within the class and its derived cla
sses
    void protectedFunc(){// can only be accessed from within the
 class and its derived classes
        cout << "This is a protected function" << endl;
    }
};
```

The public Members

- A public member is accessible from anywhere outside the class but within a program.

https://ideone.com/hBo6GY

```
1   #include <iostream>
2   using namespace std;
3
4   class Line {
5      public:
6         double length;
7         void setLength( double len );
8         double getLength( void );
9   };
10
11  // Member functions definitions
12  double Line::getLength(void) {
13     return length ;
14  }
15
```

```
16  void Line::setLength( double len) {
17      length = len;
18  }
19
20  // Main function for the program
21  int main() {
22      Line line;
23
24      // set line length
25      line.setLength(6.0);
26      cout << "Length of line : " << line.getLength() <<endl;
27
28      // set line length without member function
29      line.length = 10.0; // OK: because length is public
30      cout << "Length of line : " << line.length <<endl;
31
32      return 0;
33  }
```

stdout

```
Length of line : 6
Length of line : 10
```

Thank You!