

Data Structures and Object Oriented Programming using C++

Ahsan Ijaz

Topics Covered

- Classes
- Objects
- Data Hiding
- Constructors
- Destructors
- Namespaces

Topics Covered

- Classes
- Objects
- Data Hiding
- Constructors
- Destructors
- Namespaces

Topics Covered

- Classes
- Objects
- Data Hiding
- Constructors
- Destructors
- Namespaces

Topics Covered

- Classes
- Objects
- Data Hiding
- Constructors
- Destructors
- Namespaces

Topics Covered

- Classes
- Objects
- Data Hiding
- Constructors
- Destructors
- Namespaces

Topics Covered

- Classes
- Objects
- Data Hiding
- Constructors
- Destructors
- Namespaces

Hall of Fame

- Uzair Arshad
- Afeef Ahmed
- Ayesha Sarwar
- Bashir-ud-Din
- Hassan Hashmi
- Zohaib Shabir
- Shahzaib
- Waleed Ahmed

Sample Quiz 1

Hassan Hashmi | 3A-MIS-A
10/10

int ages, a, b, c, d;
float height, skin_temperatures;
String name, hobbies, socialprofile, schooling,
sports, mischiefs;

age = 20;
name = "Zurain Malik Awan";
height = 1.82;
skin_temperature = 98.0;
hobbies = "biking, swimming, books,
travelling";
social_profile = "Tough, sometimes boring,
teasing, friendly, all G
friend can be";
mischiefs = "brakes, masters, quick responses,
rule breaking, etc";
schooling = "from APP School, now
in FAST Lahore";

sport() = "To lazy for sports"

a =	Dinking()	Void Dinking()
b =	Speaking()	Void Speaking()
c =	Bloating()	Void bloating()
d =	Fiddling()	Void fiddling()

Figure: Hassan Hashmi

Sample Quiz 2

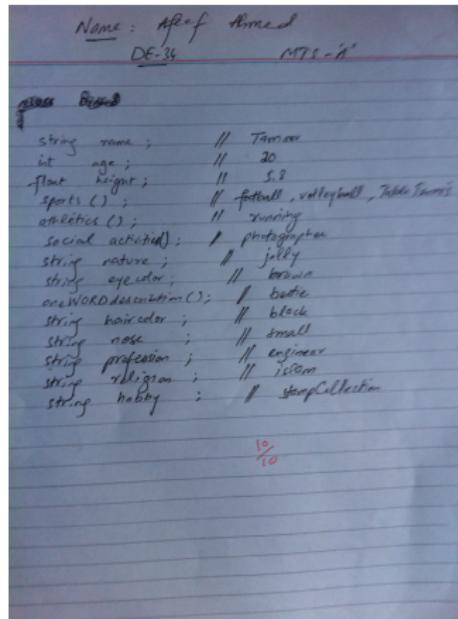


Figure: Afeef Ahmed

Operator Overloading

- Using Operators to perform different functions
 - Built-in operators can be used with classes to perform Class specific tasks
 - Similar concept to function overloading

Operator Overloading

- Using Operators to perform different functions
- Built-in operators can be used with classes to perform Class specific tasks
- Similar concept to function overloading

Operator Overloading

- Using Operators to perform different functions
- Built-in operators can be used with classes to perform Class specific tasks
- Similar concept to function overloading

Why Operator Overloading?

Synaptic Sugar: Designed to make things easier to express and understand.

```
1 Complex a(1.2,1.3);
2 //this class is used to represent complex numbers
3 Complex b(2.1,3);
4 Complex c = a+b;
5 //for this to work the addition operator must
6 // be overloaded
```

Operator Overloading Basic Example

```
1 // vectors: overloading operators example
2 #include <iostream>
3 using namespace std;
4
5 class CVector {
6 public:
7     int x,y;
8     CVector () {};
9     CVector (int ,int );
10    CVector operator + (CVector);
11 };
12
13 CVector::CVector (int a, int b) {
14     x = a;
15     y = b;
16 }
17
18 CVector CVector::operator+ (CVector param) {
19     CVector temp;
20     temp.x = x + param.x;
21     temp.y = y + param.y;
22     return (temp);
23 }
24
25 int main () {
26     CVector a (3,1);
27     CVector b (1,2);
28     CVector c;
29     c = a + b;
30     cout << c.x << "," << c.y;
31     return 0;
32 }
```

Operator Overloading Basic Example

Calling the overloaded operator.

```
1 c = a + b; // Implicit Calling  
2 c = a.operator+ (b); // Explicit Calling
```

This Pointer

- this pointer is passed as a hidden argument to all nonstatic member function calls
- Every object has access to its own address through this pointer.
- this -> member-identifier

This Pointer Example 1

```
1 class Something
2 {
3     private:
4         int nData;
5
6     public:
7         Something(int nData)
8         {
9             this->nData = nData;
10        }
11    };
```

This Pointer Example 2

```
1 #include <iostream>
2 using namespace std;
3 class Box
4 {
5     public:
6         Box(double l, double b, double h)
7         {
8             cout << "Constructor called." << endl;
9             length = l;
10            breadth = b;
11            height = h;
12        }
13        double Volume()
14        {
15            return length * breadth * height;
16        }
17        int compare(Box box)
18        {
19            return (this->Volume() > box.Volume());
20        }
21     private:
22         double length;      // Length of a box
23         double breadth;     // Breadth of a box
24         double height;      // Height of a box
25    };
26
27 int main(void)
28 {
29     Box Box1(3.3, 1.2, 1.5);    // Declare box1
30     Box Box2(8.5, 6.0, 2.0);    // Declare box2
31     big = Box1.compare(Box2);
32     return 0;
33 }
```