|  |  |  |  |
| --- | --- | --- | --- |
| **Object Oriented Programming Lab 05** | | | |
| **Course Code:** | COMP-112L | **Class** | CS (B) |
| **Lab Engineer** | Laiba Khalid | **Semester** | 2nd |
| **Lab Title** | C++ Classes | **Section** | - |
| **Name** |  | **Reg no.** |  |
| **Content Covered** | Copy Constructor & it’s types | | |
| **Instructions:**  • Submit the file with your names following your registration numbers like AI001\_Name.  • Submit soft copy of the report before deadline. Marks will be deducted for late submissions. | | | |

**Copy Constructor & It’s Types**

**Introduction**

A **copy constructor** is a special type of constructor in C++ that initializes a new object as a copy of an existing object.

**Syntax:**

|  |
| --- |
| ClassName (const ClassName &obj) {  // Copy the values from obj to the new object  } |

**Types of Copy Constructors:**

1. **Default Copy Constructor (Shallow Copy)** 
   * Automatically provided by C++ if no custom copy constructor is defined.
   * Copies memory addresses instead of actual values.

**Syntax:**

|  |
| --- |
| #include<iostream>  using namespace std;  class Example {  public:  int a;  Example(int x) { a = x; }  };  int main() {  Example obj1(10);  Example obj2 = obj1; // Default Copy Constructor  cout << "Value of obj2.a: " << obj2.a << endl;  return 0;  } |

1. **User-Defined Copy Constructor (Shallow Copy)**

* Explicitly defined by the user.
* Performs member-wise copying.

**Syntax:**

|  |
| --- |
| #include<iostream>  using namespace std;  class Example {  public:  int a;  Example(int x) { a = x; }  Example(const Example &obj) { a = obj.a; }  };  int main() {  Example obj1(10);  Example obj2(obj1); // User-Defined Copy Constructor  cout << "Value of obj2.a: " << obj2.a << endl;  return 0;  } |

1. **Deep Copy Constructor**

* Used when the class has dynamically allocated memory.
* Prevents shared memory issues between objects.

**Syntax:**

|  |
| --- |
| #include<iostream>  using namespace std;  class Example {  private:  int \*a;  public:  Example(int x) {  a = new int(x);  }    Example(const Example &obj) {  a = new int(\*obj.a); // Deep copy  }    void show() { cout << "Value: " << \*a << endl; }    ~Example() { delete a; }  };  int main() {  Example obj1(10);  Example obj2 = obj1; // Deep Copy Constructor  obj2.show();  return 0;  } |

**Conclusion**

* **Shallow Copy** only copies memory addresses.
* **Deep Copy** creates a separate memory allocation.
* **User-Defined Copy Constructor** allows control over the copying process.