كليـــة العلـــــوم  
قسم الأمن السيبراني

**Subject: Object Oriented Programming (OOP)**

**Second Stage**

**Lecturer: Dr. Abdulkadhem A. Abdulkadhem**

**Lecture (8)**

**Scope Operator Resolution, Member Initialization List, Constant Members and Static Members**

**1. Scope Operator Resolution (::)**

The scope resolution operator :: in C++ is used to define or access a global variable when there is a local variable with the same name or to access members of a namespace or class.

**Example:**

|  |
| --- |
| #include <iostream>  using namespace std;  **int var = 100;** // متغير عام عالمي  class MyClass {  public:  **int var;** // متغير خاص بالكائن  MyClass(int value) {  var = value; // تهيئة المتغير الخاص بالكائن  }  void display() {  **int var = 10;** // متغير محلي داخل الدالة  cout << "Local var: " << **var** << endl; // يطبع المتغير المحلي  cout << "Object's var: " << **MyClass::var** << endl; // يطبع متغير الكائن الحالي  cout << "Global var: " << **::var** << endl; // يطبع المتغير العام  }  };  int main() {  MyClass obj(20); // إنشاء كائن وتهيئة المتغير الخاص به إلى 20  obj.display();  return 0;  } |

**2. Member Initialization List**

In C++, a member initialization list allows initializing class members directly before the constructor's body. This is particularly useful for const members and references.

**Example:**

|  |
| --- |
| #include <iostream>  using namespace std;  class Student {  private:  **const int id; // const member**  string name;  public:  // Member Initialization List  Student(int id, string name) : id(id), name(name) {  // Constructor body can be empty for initialization  }  void display() {  cout << "ID: " << id << ", Name: " << name << endl;  }  };  int main() {  Student student1(**101**, "Alice");  student1.display();  return 0;  } |

**Explanation:**

* The id member is const and can only be initialized in a member initialization list. Without this syntax, initializing const or reference members would result in a compilation error.

**3. Constant Members**

Constant members in C++ are members that cannot be modified once they are initialized. This concept extends to function arguments and member functions.

* **Constant Function Argument**: Declares an argument as const to prevent it from being modified inside the function.
* **Constant Member Function**: Declares a member function as const, ensuring it does not modify any class members.

**Example:**

|  |
| --- |
| #include <iostream>  using namespace std;  class Rectangle {  private:  int width, height;  public:  Rectangle(int w, int h) : width(w), height(h) {}  // Constant member function  int area() **const** {  **// width++; // This line would cause a compilation error**  return width \* height;  }  // Function with const argument  void display(**const** string &prefix) **const** {  cout << prefix << " Area: " << area() << endl;  }  };  int main() {  Rectangle rect(5, 3);  rect.display("Rectangle");  return 0;  } |

**Explanation:**

* area() is a constant member function, meaning it cannot modify the class members width and height.
* display() takes a constant string reference as an argument, which prevents modification of the input string.

**4. Static Members**

Static members belong to the class rather than any particular object. They retain their values across all instances of the class, and there is only one copy of each static member.

**Example:**

|  |
| --- |
| #include <iostream>  using namespace std;  class Counter {  public:  **static int count=0; // Static member variable**  Counter() {  **count++;**  }  static void showCount() {  cout << "Count: " << count << endl;  }  };  // Initialize static member  int Counter::count = 0;  int main() {  Counter c1, c2, c3;  Counter::showCount(); // Accessing static member function  return 0;  } |

**Explanation:**

* count is a static member variable shared among all objects of Counter.
* Each time a Counter object is created, count is incremented.
* The showCount() function is a static member function that can access only static members and can be called without an object instance.

**Summary**

In this lecture, we covered advanced C++ concepts that are critical for efficient and organized code development:

1. **Scope Operator Resolution**: Used to access specific variables within scopes.
2. **Member Initialization List**: Allows efficient and necessary initialization of class members, particularly const and references.
3. **Constant Members**: Ensures data immutability within certain contexts, including const arguments and const functions.
4. **Static Members**: Allows shared variables and functions that are independent of individual objects.

These concepts are fundamental for writing clean, maintainable C++ code in both simple and complex applications.