جامـــــعة المـــــستـقـبـل
AL MUSTAQBAL UNIVERSITY

قـســـــم الامـــــــــــــــن الـــــــــــــــــسيبرانـــــــــــــــــــي

Department of Cyber Security

Subject:

Object Oriented Programming (OOP)

Class:

Second

Lecturer:

Dr. Abdulkadhem A. Abdulkadhem

Lecture: (7)

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 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);   // 20 إنشاء كائن وتهيئة المتغير الخاص به إلى
    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 (ثبات) 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.

Questions about the lecture

1. Which operator is used in C++ to access a global variable when a local variable with the same name exists?

2. What is the output of the following code snippet?

```
int var = 100;
class Test {
public:
    int var;
    Test(int v) { var = v; }
    void show() {
        int var = 10;
        cout << ::var;
    }
};
int main() {
    Test obj(50);
    obj.show();
    return 0;
}
```

3. In C++, a member initialization list is mainly required when initializing which of the following?

4. Which of the following correctly defines a constructor using a member initialization list?

A. `Student(int id, string name) { id = id; name = name; }`
B. `Student(int id, string name): id(id), name(name) {}`
C. `Student(int id, string name); id = id; name = name;`
D. `Student::Student(int id, string name) { id=id; name=name; }`
E. `Student(id, name): (id, name) {}`

5. Which statement is true regarding constant member functions in C++?

A. They can modify all class members.
B. They cannot be called on constant objects.
C. They cannot modify any non-static data members.
D. They cannot be declared inside a class.
E. They can only return `void`.

6. What will happen if a constant member variable is initialized inside the constructor body instead of using a member initialization list?

7. In the following code, what is the output?

```
class Counter {
public:
    static int count;
    Counter() { count++; }
};
int Counter::count = 0;
int main() {
    Counter c1, c2;
    cout << Counter::count;
}
```

8. Which of the following statements about static member functions is TRUE?

A. They can access both static and non-static members directly.
B. They cannot be called using the class name.
C. They belong to a specific object.
D. They can be called without creating an object.
E. They must always return `int`.

9. Why should static members be defined outside the class?