كليـــة العلـــــوم
قسم الأمن السيبراني

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

**Second Stage**

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

**Lecture (9)**

**Object Pointers and this Pointer in C++**

**Topic Overview**

1. **Pointers to Objects**: How pointers can point to instances of classes and access member functions.
2. **this Pointer**: How to use the this pointer to refer to the current instance of a class.
3. **Reference Members**: Explanation of how to use references within classes.

**1. Pointers to Objects**

In C++, a pointer is a variable that holds the memory address of another variable, including objects. You can declare a pointer to an object, allowing you to dynamically allocate, access, and manage the object’s members through the pointer.

**Syntax**:

|  |
| --- |
| ClassName \*pointerName = &objectName; |

**Example 1**: Writing an OOP program to read and display student information using an object pointer.

|  |
| --- |
| #include <iostream>using namespace std;class Student {private: int stageNumber; int age; char gender; float height; float weight;public: void getInfo() { cout << "Enter stage number: "; cin >> stageNumber; cout << "Enter age: "; cin >> age; cout << "Enter gender (M/F): "; cin >> gender; cout << "Enter height (in cm): "; cin >> height; cout << "Enter weight (in kg): "; cin >> weight; } void displayInfo() const { cout << "Stage number: " << stageNumber << endl; cout << "Age: " << age << endl; cout << "Gender: " << gender << endl; cout << "Height: " << height << endl; cout << "Weight: " << weight << endl; }};int main() { **Student \*ptr = new Student;** // الطريقة الاولى لتعريف البوينتر للاوبجكت **Student std;** // الطريقة الثانية لتعريف البوينتر للاوبجكت **Student \*ptr =std;** cout << "Enter the following information:\n"; **ptr->getInfo();** // Using pointer to access getInfo() cout << "\nStudent Information:\n"; **(\*ptr).displayInfo();** // Using pointer to access displayInfo() delete ptr; // Free allocated memory return 0;} |

**2. Using the this Pointer**

The this pointer in C++ is an implicit pointer that points to the current object of a class. It is especially useful for differentiating between class members and function parameters that have the same name.

**Example 1**: Displaying the **memory address** of an object using this.

|  |
| --- |
| #include <iostream>using namespace std;class Sample {public: void showAddress() const { cout << "Object address: " << **this** << endl; }};int main() { Sample obj1, obj2, obj3; obj1.showAddress(); obj2.showAddress(); obj3.showAddress(); return 0;} |

**Explanation**:

* Each object (obj1, obj2, obj3) will output a unique memory address when showAddress is called.
* The this pointer is implicitly (ضمنياً) used to refer to the object that called the function.

**Example 2**: Using this to access members.

|  |
| --- |
| #include <iostream>using namespace std;class Sample {private: int value;public: Sample(int value) { **this->**value = value; // Assigning parameter 'value' to member 'value' } void display() const { cout << "Value: " << **this->**value << endl; }};int main() { Sample obj(42); obj.display(); // Displays: Value: 42 return 0;} |

**Explanation**:

* The this pointer is used to resolve ambiguity between the member variable value and the constructor parameter value.

**3. Reference Members**

In C++, reference members allow you to initialize class members that are references to other variables or objects. However, references must be initialized when declared and cannot be changed to refer to different objects.

**Example**: Using a reference member in a class.

|  |
| --- |
| #include <iostream>using namespace std;class Wrapper {private: int &ref; // Reference memberpublic: Wrapper(int &val) : ref(val) {} // Initialize reference member in constructor void showValue() const { cout << "Referenced value: " << ref << endl; } void updateValue(int newVal) { ref = newVal; // Update the original variable referenced by 'ref' }};int main() { int num = 10; Wrapper wrapper(num); wrapper.showValue(); // Displays: Referenced value: 10 wrapper.updateValue(20); cout << "Updated original value: " << num << endl; // Displays: Updated original value: 20 return 0;} |

**Explanation**:

* The reference ref in the Wrapper class directly refers to num in main. Changes to ref will directly impact num, demonstrating how reference members work in classes.

### 4. ****Class Object Members****

In C++, a data member of a class can be an object of another class. This is known as **class composition**. It allows a class to contain objects of other classes as members, facilitating a "has-a" relationship between classes.

**Example 2: Summing the Coordinates of Points Using a Class Object Member**

Here, we have two classes: Point and PointSummation. The PointSummation class contains two Point objects as members and calculates the sum of their coordinates.

|  |
| --- |
| #include <iostream>using namespace std;// Class representing a Point with x and y coordinatesclass Point {private: int x, y;public: Point(int xCoord, int yCoord) : x(xCoord), y(yCoord) {} int getX() const { return x; } int getY() const { return y; }};// Class that uses Point objects to sum their coordinatesclass PointSummation {private: Point point1, point2; // Two Point objects as memberspublic: PointSummation(int x1, int y1, int x2, int y2) : point1(x1, y1), point2(x2, y2) {} // Initializing Point objects in constructor void displaySummation() const { int sumX = point1.getX() + point2.getX(); int sumY = point1.getY() + point2.getY(); cout << "Sum of X coordinates: " << sumX << endl; cout << "Sum of Y coordinates: " << sumY << endl; }};int main() { int x1, y1, x2, y2; cout << "Enter coordinates for point 1 (x y): "; cin >> x1 >> y1; cout << "Enter coordinates for point 2 (x y): "; cin >> x2 >> y2; PointSummation summation(x1, y1, x2, y2); summation.displaySummation(); return 0;} |

**Explanation**:

* The Point class holds x and y coordinates.
* The PointSummation class contains two Point objects as members.
* The displaySummation function in PointSummation calculates and displays the sum of the x and y coordinates of the two Point objects.