**Al- Mustaqbal University**

**College of Sciences**

**Department of Cybersecurity**

**Object-Oriented Programming**

**(OOP)**

**( *CONSTRUTORS & DESTRUCTOR* )**

Second stage

Lecture 8/ Practical Part

2025-2024

**Lectuer : Muntather AL-Mousawi**

**CONSTRUTORS**

A constructor is a special member function that is automatically called when an object is created. Its primary purpose is to initialize objects of a class. The constructor has the same name as the class and does not have a return type.

(class) من فئة معينة (object) كائن المنشئ هو دالة خاصة في البرمجة الكائنية تُستدعى تلقائيًا عند إنشاء الهدف الأساسي للمنشئ هو تهيئة بيانات الكائن بقيم أولية، ويكون له نفس اسم الفئة ولا يمتلك نوع إرجاع **Types of Constructors**

1. Default Constructo
2. Parameterized Constructor
3. Copy Constructor

المنشئ الافتراضي -1

هو منشئ لا يأخذ أي معطيات ويقوم بتهيئة الكائنات بقيم افتراضية

2- المنشئ ذو المعاملات

هو منشئ يأخذ معاملًا واحدًا أو أكثرو يسمح بتهيئة الكائن بقيم محددة عند إنشائه

3- منشئ النسخ

هو منشئ يُستخدم لإنشاء كائن جديد كنسخة من كائن آخر موجود من نفس الفئة

يُستخدم لتهيئة كائن عبر نسخ بيانات كائن آخر من نفس النوع

**(1)**

**Example of a Default Constructor**

**#include <iostream>**

**using namespace std;**

**class Student {**

**public:**

**string name;**

**int age; // Default Constructor**

**Student() {**

**name = “Unknown”;**

**age = 0;**

**}**

**void displayInfo() {**

**cout << “Name: “ << name << “, Age: “ << age << endl;**

**}**

**};**

**int main() {**

**Student student1; // Default constructor is called**

**student1.displayInfo();**

**return 0;**

**}**

**(2)**

**Example of a Parameterized Constructor**

**#include <iostream>**

**using namespace std;**

**class Student {**

**public:**

**string name;**

**int age; // Parameterized Constructor**

**Student(string n, int a) {**

**name = n;**

**age = a;**

**}**

**void displayInfo() {**

**cout << "Name: " << name << ", Age: " << age << endl;**

**}**

**};**

**int main() {**

**Student student1("Alice", 20); // Parameterized constructor is called**

**student1.displayInfo();**

**return 0;**

**}**

**(3)**

**Example of a Copy Constructor**

**#include <iostream>**

**using namespace std;**

**class Book {**

**public:**

**string title;**

**int pages; // Parameterized Constructor**

**Book(string t, int p) {**

**title = t;**

**pages = p;**

**}**

**Book(const Book &b) { // Copy Constructor**

**title = b.title;**

**pages = b.pages;**

**}**

**void displayInfo() {**

**cout << "Title: " << title << ", Pages: " << pages << endl;**

**}**

**};**

**int main() {**

**Book book1("C++ Programming", 500); // Parameterized constructor is called**

**Book book2 = book1; // Copy constructor is called**

**book2.displayInfo();**

**return 0; (4)**

**Destructor**

A destructor is a special member function that is automatically called when an object is destroyed. Its primary purpose is to release any resources that the object may have acquired during its lifetime. The destructor has the same name as the class but is preceded by a tilde (~).

الهدف الأساسي (object) المهدم هو دالة خاصة في البرمجة الكائنية تُستدعى تلقائيًا عند تدمير كائن للمهدم هو تحرير أي موارد قد يكون الكائن قد استحوذ عليها أثناء فترة حياته، مثل الذاكرة أو الملفات (~)لضمان إدارة فعّالة للموارد يكون للمهدم نفس اسم الفئة ولكنه يسبق برمز

**Example of a Destructor**

**#include <iostream>**

**using namespace std;**

**class Student {**

**public:**

**string name;**

**int age; // Constructor**

**Student(string n, int a) {**

**name = n;**

**age = a;**

**cout << "Constructor is called for " << name << endl;**

**}**

**~Student() { // Destructor**

**cout << "Destructor is called for " << name << endl;**

**(5)**

**}**

**void displayInfo() {**

**cout << "Name: " << name << ", Age: " << age << endl;**

**}**

**};**

**int main() {**

**Student student1("Alice", 20);**

**student1.displayInfo();**

**return 0;**

**}**

**(6)**

**Constructor Overloading**

زيادة التحميل على المنشئات تعني أن الفئة يمكن أن تحتوي على عدة منشئات ++C لغة في

بقوائم معاملات مختلفة. أي أن الفئة تستطيع أن تمتلك أكثر من منشئ، بحيث يقبل كل واحد عددًا أو نوعًا مختلفًا من المعاملات. عند إنشاء كائن، يتم اختيار المنشئ المناسب بناءً على المعاملات المقدمة. وتعتبر زيادة التحميل على المنشئات مفيدة لإنشاء كائنات بتهيئات مختلفة حسب المعاملات المعطاة

مثال:

class Example {

public:

Example() { // المنشئ الافتراضي

}

Example(int x) { // منشئ ذو معامل واحد

}

Example(int x, int y) { // منشئ ذو معاملين

}

};

على ثلاثة منشئات، ولكل منها قائمة معاملات مختلفة في هذا المثال، تحتوي الفئة

**Example of Constructor Overloading**

**#include <iostream>**

**using namespace std;**

**(7)**

**class Rectangle {**

**public:**

**int length, width; // Default Constructor**

**Rectangle() {**

**length = 0;**

**width = 0;**

**}**

**Rectangle(int l, int w) { // Parameterized Constructor**

**length = l;**

**width = w;**

**}**

**void displayArea() {**

**cout << "Area: " << length \* width << endl;**

**}**

**};**

**int main() {**

**Rectangle rect1; // Default constructor is called**

**Rectangle rect2(5, 10); // Parameterized constructor is called**

**rect1.displayArea(); //Displays area of default rectangle (0)**

**rect2.displayArea(); // Displays area of rectangle (50)**

**return 0;**

**}**

**(8)**