**Al- Mustaqbal University **

**College of Sciences**

**Department of Cybersecurity**

**Object Oriented Programming**

**(OOP)**

Second stage

Lecture 4/ Practical Part

FUNCTIONS

2025-2024

**Overloading:**

## :Overloading مفهوم ال

تعني تعريف أكثر من عامل, دالة أو كونستركتور لهم نفس الإسم و لكنهم يختلفون في عدد أو نوع الباراميترات.

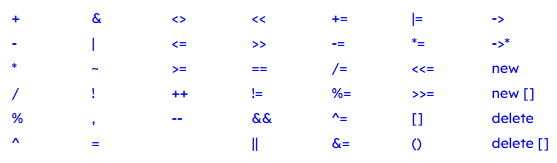
\*عند تعريف أكثر من كونستركتور لهم نفس الإسم يكون الهدف أنه عند إنشاء كائن يكون هناك أكثر من طريقة متاحة لتمرير قيم أولية للخصائص الموجودة فيه.

\*عند تعريف أكثر من دالة لهم نفس الإسم يكون الهدف منهم إمكانية تنفيذ نفس العملية مع مراعاة عدد و أنواع القيم التي يتم تمريرها لللدالة عند استدعائها.

## :Overloadingشروط ال

* يطبق فقط على العوامل, الدوال و الكونستركتورات.
* يجب أن يملكوا نفس الإسم
* يجب أن يختلفوا في نوع أو عدد الباراميترات.
* بالنسبة للدوال, نوع الإرجاع غير مهم لأن المترجم لا يستطيع التفريق بين الدوال إذا كانوا مختلفين في نوع الإرجاع فقط.

**:Overloadالعوامل التالية يمكن أن نفعل لها**



**:Overloadالعوامل التالية لا يمكن أن نفعل لها**

**.     ::     ?:     sizeof**

**(1)**

# \*مثال حول تعريف دوال لها نفس الإسم و تختلف في نوع الباراميترات

**Example:**

**#include <iostream>**

**using namespace std;**

**void sum(int a, int b){**

**cout << "First method is called " << a << " + " << b << " = " << (a+b) << endl;**

**}**

**void sum(double a, double b){**

**cout << "Second method is called " << a << " + " << b << " = " << (a+b) << endl;**

**}**

**int main()**

**{**

**sum(4, 6);**

**// int هنا سيتم إستدعاء الدالة التي تأخذ 2 باراميترنوعهم**

**sum(2.3, 5.4);**

**// doubleهنا سيتم إستدعاء الدالة التي تأخذ2باراميتر نوعهم**

**return 0;**

**}**

**النتيجه:**

**First method is called 4 + 6 = 10**

**Second method is called 2.3 + 5.4 = 7.7**

**(2)**

# \*مثال حول تعريف دوال لها نفس الإسم و تختلف في عدد الباراميترات

**Example:**

**#include <iostream>**

**using namespace std;**

**void sum(float a, float b){**

**cout << a << " + " << b << " = " << (a + b) << endl;**

**}**

**void sum(float a, float b, float c){**

**cout <<a<< " + "<< b << " + "<< c << " = " << (a + b +c)<< endl;**

**}**

**int main(){**

**sum(1, 3);**

**sum(1, 3, 7);**

**return 0;**

**}**

**النتيجه:**

**1 + 3 = 4**

**1 + 3 + 7 = 11**

**(3)**

# مثال حول تعريف دوال تعتمد على دوال عندها نفس الإسم\*

**Example:**

**#include <iostream>**

**using namespace std;**

**double maximum(double a, double b)**

**{**

**if(a>b)**

**return a;**

**else**

**return b;**

**}**

**double maximum(double a, double b, double c)**

**{**

**if(maximum(a,b)>c)**

**return maximum(a,b);**

**else**

**return c;**

**}**

**double maximum(double a, double b, double c, double d)**

**{**

**(4)**

**if(maximum(a,b,c)>d)**

**return maximum(a,b,c);**

**else**

**return d;**

**}**

**int main()**

**{**

**cout << "The maximum number is: " << maximum(5, 20) << endl;**

**cout << "The maximum number is: " << maximum(5, 20, 15) <<endl;**

**cout << "The maximum number is: " << maximum(5, 20, 15, 30);**

**return 0;**

**}**

**النتيجه:**

**The maximum number is: 20**

**The maximum number is: 20**

**The maximum number is: 30**

**(5)**

**Inline Function:**

**(Inline Function)الدالة المضمنةC في لغة ++**

**هي دالة يطلب من المترجم استبدال استدعائها في مكان استخدامها بالكود الخاص بها بدلاً من تنفيذ عملية الاستدعاء التقليدية. الفائدة من ذلك هي تقليل زمن التنفيذ الناتج عن استدعاء الدوال** المتكرر، وخاصة في الدوال الصغيرة والبسيطة. يتم **.inline ذلك باستخدام الكلمة المفتاحية**

**:مثالً عن كيفية تعريف واستخدام الدوال المضمنة\***

**#include <iostream>**

**using namespace std;**

**// تعريف دالة مضمنة**

**inline int add(int a, int b)**

**{**

**return a + b;**

**}**

**int main() {**

**int x = 5, y = 10;**

**// استدعاء الدالة المضمنة**

**cout << "Sum: " << add(x, y) << endl;**

**return 0;**

**}**

**(6)**

**Default Arguments:**

(Default Arguments) الوسائط الافتراضية C++ في لغة

هي القيم التي يتم تمريرها إلى الدالة تلقائيًا إذا لم يقدم المستخدم قيمة لهذه الوسائط عند استدعاء الدالة. يتم تحديد هذه القيم الافتراضية عند تعريف الدالة.

\*مثال يوضح كيفية استخدام الوسائط الافتراضية:

**#include <iostream>**

**using namespace std:**

**// دالة تحتوي على وسيط افتراضي**

**void printInfo(string name, int age = 18)**

**{**

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

**}**

**int main()**

**{**

**// استدعاء الدالة مع وسيطين**

**printInfo("Alice", 25); // سيطبع: Name: Alice, Age: 25**

**// استدعاء الدالة مع وسيط واحد فقط**

**printInfo("Bob"); // سيطبع: Name: Bob, Age: 18**

**return 0;**

**}**

**(7)**

**Pass by Reference:**

(Pass by Reference) تمرير المعاملات بالإشارة C++ في لغة

يعني تمرير متغير إلى الدالة بحيث تعمل الدالة مباشرة على المتغير الأصلي وليس نسخة منه. هذا يتيح للدالة تعديل قيمة المتغير خارج نطاقها

يتم استخدام علامة & لتعريف المعامل كمرجع (Reference) في تعريف الدالة

مثال على تمرير بالمرجع:

**#include <iostream>**

**using namespace std;**

**void swap(int &a, int &b) {**

**int temp = a;**

**a = b;**

**b = temp;**

**}**

**int main() {**

**int x = 10, y = 20;**

**cout << "Before swap: x = " << x << ", y = " << y << endl;**

**// تمرير المتغيرات بالإشارة**

**swap(x, y);**

**cout << "After swap: x = " << x << ", y = " << y << endl;**

**return 0;**

**}**

**(8)**

**Return by Reference :**

) Return by Reference الإرجاع بالإشارة( C++في لغة

يعني إرجاع مرجع إلى متغير موجود خارج الدالة (بدلاً من إرجاع نسخة منه). يسمح ذلك للدالة بإرجاع متغير يمكن التعديل عليه بعد إرجاعه. عند استخدام الإرجاع بالإشارة، يتم تعديل المتغير الأصلي مباشرة.

\* مثال على الإرجاع بالإشارة**:**

**#include <iostream>**

**using namespace std;**

**// Function that returns a reference to a variable**

**int& getLargest(int &a, int &b) {**

**return (a > b) ? a : b;**

**}**

**int main() {**

**int x = 5, y = 10;**

**cout << "Before modification: x = " << x << ", y = " << y << endl; getLargest(x, y) = 100; // Modifying the largest number by reference cout << "After modification: x = " << x << ", y = " << y << endl;**

**return 0;**

**}**

**(9)**