المصفوفات والعمليات (Methods)

Arrays as Parameters & Return Values

إعداد: الأستاذ محمود اغبارية

💡 لماذا نستخدم المصفوفات مع الدوال؟

بدلاً من كتابة كود معالجة المصفوفة (مثل الطباعة، البحث، الجمع) داخل `Main` وتكراره، نقوم بوضعه في عملية خارجية (Method).

الفائدة:

- تنظيم الكود وجعله أسهل للقراءة.
- إعادة الاستخدام (Reusability).
- تقليل الأخطاء.

</> أولاً: تمرير مصفوفة للدالة

عند تعريف الدالة، نحدد أن البارامتر هو مصفوفة بإضافة [] بعد النوع.

```
public static void PrintArray(int[] arr)
{
  //  نتعامل مع arr هنا كمصفوفة عادية
   for (int i = 0; i < arr.Length; i++)
   {
     Console.WriteLine(arr[i]);
   }
}
```

لاحظ أننا استخدمنا `arr.Length` داخل الدالة، لأن المصفوفة تعرف طولها.

📞 كيفية الاستدعاء من Main

عند الاستدعاء، نرسل **اسم المصفوفة فقط** بدون أقواس مربعة.

```
static void Main(string[] args)
{
  int[] grades = { 90, 85, 95 };

  // استدعاء الدالة
  PrintArray(grades);
}
```

⚠ خطأ شائع:
لا تكتب `PrintArray(grades[])` أو `PrintArray(grades[3])`.
أرسل الاسم فقط: `PrintArray(grades)`.

مفهوم المرجع (Reference)

مهم جداً: المصفوفات هي Reference Types.
عندما تمرر مصفوفة لدالة، أنت تمرر **عنوانها في الذاكرة**، لا نسخة منها.

هذا يعني: إذا قامت الدالة بتغيير قيمة داخل المصفوفة، **ستتغير المصفوفة الأصلية** في Main أيضاً!

توضيح بالرسم

المتغير في Main والبارامتر في الدالة يشيران لنفس المكان.

أي تعديل على arr سيراه grades فوراً.

مثال: دالة تعدل المصفوفة

دالة تقوم بزيادة كل عنصر بمقدار 5.

```
public static void AddBonus(int[] arr)
{
  for (int i = 0; i < arr.Length; i++)
  {
    arr[i] += 5;//  التغيير يحدث في الأصل
  }
}
```

```
//  في Main
int[] marks = { 50, 60 };
AddBonus(marks);
Console.WriteLine(marks[0]);//  55 سيطبع
```

↩ ثانياً: إرجاع مصفوفة من الدالة

يمكن للدالة أن تُنشئ مصفوفة جديدة وتعيدها.
نوع الإرجاع يجب أن يكون `type[]`.

```
// لاحظ int[] بدل int
 public static int[] CreateArray(int size)
 {
   int[] newArr = new int[size];
  // كود لتعبئة المصفوفة.
   return newArr;// إرجاع المصفوفة كاملة
 }
```

مثال: دالة تعيد الأعداد الزوجية

```
public static int[] GetEvens(int size)
{
  int[] result = new int[size];
  for (int i = 0; i < size; i++)
  {
    result[i] = i * 2;
  }
  return result;
}
```

الاستدعاء في Main:

```
int[] myEvens = GetEvens(5);
// myEvens {8 ,6 ,4 ,2 ,0} الآن تحتوي
```

🔁 دالة تستقبل وتعيد مصفوفة

أحياناً نريد مصفوفة جديدة مبنية على القديمة دون تغيير القديمة.

```
public static int[] CopyAndDouble(int[] original)
{
  int[] copy = new int[original.Length];
  for (int i = 0; i < original.Length; i++)
    copy[i] = original[i] * 2;

  return copy;//  نعيد النسخة الجديدة
}
```

أخطاء شائعة (1)

الخطأ: تعريف نوع الإرجاع خطأ.

```
// خطأ! لا يمكن إرجاع مصفوفة ونوع الدالة int
public static int BadMethod()
{
  int[] arr = {1, 2};
  return arr; // Compile Error
}
```

التصحيح: يجب أن يكون النوع `int[]`.

أخطاء شائعة (2)

الخطأ: محاولة طباعة المصفوفة مباشرة عند الاستدعاء.

```
int[] arr = GetEvens(5);

// هذا سيطبع "System.Int32[]" وليس القيم
Console.WriteLine(arr);
```

الحل: يجب استخدام حلقة (Loop) لطباعة العناصر المسترجعة، أو دالة طباعة مساعدة.

❓ اختبار سريع (1)

ما هو ناتج الكود التالي؟

```
void Change(int[] a) { a[0] = 99; }

void Main() {
  int[] x = { 1, 2 };
  Change(x);
  Console.WriteLine(x[0]);
}
```

◄ اضغط للإجابة

99

(لأن المصفوفة مُررت بالمرجع، فالتعديل دائم)

❓ اختبار سريع (2)

أين الخطأ في الكود التالي؟

```
public static int[] Test()
{
  int[] nums = { 10, 20 };
  return nums[0];
}
```

<details>
<summary>◄ اضغط للإجابة</summary>

الخطأ في `return nums[0];`

الصحيح: `return nums;` لكنك أرجعت رقماً واحداً، `int[]` الدالة تتوقع إرجاع مصفوفة كاملة
</details>

ملخص

- لتعريف بارامتر مصفوفة: استخدم `type[] name`.
- للاستدعاء: مرر اسم المصفوفة فقط `Func(myArr)`.
- المصفوفات تُمرر بالمرجع (Reference)، أي تعديل داخل الدالة يؤثر على الأصل.
- لإرجاع مصفوفة: اجعل نوع الدالة `type[]` واستخدم `return arrName;`.

الأستاذ محمود اغبارية