# المصفوفات الأحادية في #C

## Single-Dimensional Arrays

C# ...

إعداد: الأستاذ محمود اغبارية

# مقدمة: ما هي المصفوفة؟

هي بنية معطيات تمكننا من تخزين مجموعة من القيم من **نفس النوع** تحت اسم واحد.

- تخزين مجموعة كبيرة من البيانات بطريقة منظمة.
- سهولة الوصول إلى العناصر عن طريق **الفهرس (Index)**.
- تسهيل معالجة البيانات باستخدام الحلقات (Loops).
- توفير في الكود وتجنب التكرار.

# 💡 مثال حياتي

تخيّل أنك تريد تخزين درجات 30 طالباً. بدون المصفوفات:

```
int grade1, grade2, grade3, ... , grade30;
```

أما مع المصفوفات:

```
int[] grades = new int[30];
```

# </> التصريح عن المصفوفة وإنشاؤها

الصيغة العامة:

```
type[] arrayName;
```

- type: نوع العناصر (int, double, string, ...).
- []: تدل على أن المتغير مصفوفة.

## إنشاء المصفوفة (Allocation)

استخدام الكلمة المفتاحية new لتحديد الحجم.

```
//  التصريح والإنشاء بخطوة واحدة
int[] numbers = new int[5];
```

# القيم الافتراضية

عند إنشاء المصفوفة باستخدام new، يتم تصفير القيم تلقائياً:

| النوع | القيمة الافتراضية |
|---|---|
| int, double | 0 |
| bool | false |
| string, object | null |
| char | '0\' |

# 🛎 طرق التهيئة المباشرة

يمكنك إعطاء قيم أولية للمصفوفة عند إنشائها:

```
// طريقة 1: مع تحديد الحجم
int[] n1 = new int[3] { 10, 20, 30 };

// طريقة 2: الحجم تلقائي
int[] n2 = new int[] { 10, 20, 30 };

// طريقة 3: الأقصر (مفضلة)
int[] n3 = { 10, 20, 30 };
```

# 👆 الوصول للعناصر (الفهرس)

الفهرس (Index) يبدأ من **0** وينتهي عند **Length - 1**.

```
int[] arr = { 10, 20, 30, 40, 50 };

int first = arr[0];//  10
int third = arr[2];//  30
arr[4] = 99;//  تعديل العنصر الأخير
```

# 📏 خاصية الطول (Length)

تُستخدم لمعرفة عدد عناصر المصفوفة.

```
int[] nums = { 10, 20, 30, 40, 50 };
int size = nums.Length;//  5

//  الوصول للعنصر الأخير دائماً:
int last = nums[nums.Length - 1];
```

⚠️ إذا حاولت الوصول للفهرس `nums[5]` ستحصل على خطأ `IndexOutOfRangeException` لأن الفهارس من 0 إلى 4 فقط.

# 🔁 التعامل مع المصفوفة (الحلقات)

طباعة جميع العناصر باستخدام for.

```
for (int i = 0; i < arr.Length; i++)
{
  Console.WriteLine(arr[i]);
}
```

# ⌨ قراءة القيم من المستخدم

مثال لقراءة 5 درجات:

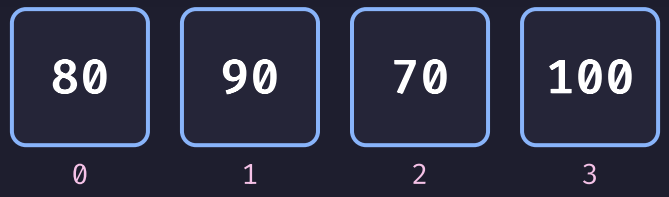
```
int[] grades = new int[5];

Console.WriteLine("أدخل 5 درجات:");
for (int i = 0; i < grades.Length; i++)
{
  Console.Write($"الدرجة {i + 1}: ");
  grades[i] = int.Parse(Console.ReadLine());
}
```

# 🧮 عمليات أساسية: المجموع

جمع عناصر المصفوفة.

```
int sum = 0;
for (int i = 0; i < grades.Length; i++)
{
  sum += grades[i];
}
```



# عمليات أساسية: العد

مثال: عد الأعداد الزوجية في المصفوفة.

```
int evenCount = 0;

for (int i = 0; i < numbers.Length; i++)
{
  if (numbers[i] % 2 == 0)
    evenCount++;
}
```

# 🏆 إيجاد العدد الأكبر

نفترض أن الأول هو الأكبر (max)، ثم نقارن مع البقية.

```
int max = nums[0];
for (int i = 1; i < nums.Length; i++)
{
  if (nums[i] > max) max = nums[i];
}
```

# إيجاد أكبر عنصر وموقعه

نحفظ القيمة والموقع (Index) معاً.

```
int max = numbers[0];
int maxIndex = 0;

for (int i = 1; i < numbers.Length; i++)
{
  if (numbers[i] > max)
  {
    max = numbers[i];
    maxIndex = i;
  }
}
```

# 🔍 البحث الخطي (Linear Search)

البحث عن عنصر والتوقف عند إيجاده.

```
int searchFor = 15;
bool found = false;

for (int i = 0; i < arr.Length; i++)
{
  if (arr[i] == searchFor)
  {
    found = true;
    Console.WriteLine("Found at: " + i);
    break;// توقف عند الإيجاد
  }
}
```

# إيجاد جميع مواقع العنصر

إذا كان العنصر مكرراً، لا نستخدم break.

```
int searchFor = 12;
int count = 0;

for (int i = 0; i < arr.Length; i++)
{
  if (arr[i] == searchFor)
  {
    Console.WriteLine("Index: " + i);
    count++;
  }
}
```

# نسخ المصفوفات - انتبه!

المصفوفة هي **Reference Type**. استخدام علامة = لا ينسخ القيم، بل ينسخ المرجع!

```
int[] arr1 = { 1, 2, 3 };
int[] arr2 = arr1;//  كلاهما يشير لنفس المكان!

arr2[0] = 100;
Console.WriteLine(arr1[0]);//  يطبع 100!
```

# النسخ الصحيح

يجب إنشاء مصفوفة جديدة ونسخ القيم إليها.

```
// الطريقة اليدوية
int[] arr2 = new int[arr1.Length];
for (int i = 0; i < arr1.Length; i++)
{
  arr2[i] = arr1[i];
}
```

```
// استخدام دالة جاهزة
Array.Copy(arr1, arr2, arr1.Length);
```

# 🔁 عكس المصفوفة (Reverse)

تبديل العناصر من البداية والنهاية حتى المنتصف.

```
for (int i = 0; i < arr.Length / 2; i++)
{
  int temp = arr[i];
  arr[i] = arr[arr.Length - 1 - i];
  arr[arr.Length - 1 - i] = temp;
}
```

## إزاحة العناصر لليمين (Shift Right) ←

```
int last = arr[arr.Length - 1];

for (int i = arr.Length - 1; i > 0; i--)
{
  arr[i] = arr[i - 1];
}
arr[0] = last;
```

# فلترة العناصر (Filter)

نسخ عناصر محددة (مثلاً الزوجية) إلى مصفوفة جديدة.

1. عد العناصر المطلوبة أولاً لتحديد حجم المصفوفة الجديدة.
2. إنشاء المصفوفة الجديدة.
3. نسخ القيم المناسبة.

```
int index = 0;
int[] evenNumbers = new int[evenCount];
for (int i = 0; i < numbers.Length; i++)
{
  if (numbers[i] % 2 == 0)
  {
    evenNumbers[index] = numbers[i];
    index++;
  }
}
```

## 🛠️ دوال مساعدة: طباعة المصفوفة

```
public static void PrintArray(int[] arr)
{
  Console.Write("[ ");
  for (int i = 0; i < arr.Length; i++)
  {
    Console.Write(arr[i]);
    if (i < arr.Length - 1) Console.Write(", ");
  }
  Console.WriteLine(" ]");
}
```

# 🛠️ دوال مساعدة: الجمع والبحث

```
public static int SumArray(int[] arr)
{
  int sum = 0;
  for (int i = 0; i < arr.Length; i++) sum += arr[i];
  return sum;
}

public static int FindElement(int[] arr, int val)
{
  for (int i = 0; i < arr.Length; i++)
    if (arr[i] == val) return i;
  return -1;
}
```

# أخطاء شائعة

**1. الخروج عن حدود المصفوفة:**
`arr[5] = 10;` لمصفوفة حجمها 5. (الصحيح أقصى حد هو 4).

**2. استخدام = بدلاً من == في الشرط:**
`if (arr[i] = 10)` (هذا تعيين وليس مقارنة).

**3. نسيان تهيئة المصفوفة:**
`int[] arr; arr[0]=5;` (يجب استخدام `new` أولاً).

# ❓ سؤال: تتبع الكود (1)

ما هو المخرج المتوقع للكود التالي؟

```
int[] data = { 5, 10, 15 };
data[0] = data[1] + data[2];
Console.WriteLine(data[0]);
```

<details>
<summary>◄ اضغط هنا لرؤية الإجابة</summary>

25
</details>

# ❓ سؤال: تتبع الكود (2)

ماذا سيطبع هذا البرنامج؟

```
int[] arr = { 1, 2, 3, 4 };
for(int i = 0; i < arr.Length; i++) {
  if(arr[i] % 2 == 0)
    Console.Write(arr[i] + " ");
}
```

◄ اضغط هنا لرؤية الإجابة

2 4

# ❓ سؤال: تحدي المراجع

انتبه! ما هي قيمة العنصر المطبوع؟

```
int[] x = { 10, 20 };
int[] y = x;
y[0] = 50;
Console.WriteLine(x[0]);
```

**◄ اضغط هنا لرؤية الإجابة**

**50**

(يشيران لنفس المصفوفة في الذاكرة y و x لأن)

# ملخص

- المصفوفة حجمها ثابت ونوعها موحد.
- يبدأ العد من **0** وينتهي عند **Length - 1**.
- النسخ بـ = ينسخ المرجع فقط؛ استخدم حلقة نسخ للقيم.
- الحلقة `for` هي الأداة الأساسية للتعامل مع المصفوفات.

## حظاً موفقاً!

الأستاذ محمود اغبارية