

المحاضرة 10 · قوائم منظمة

10

المصفوفة الأحادية: بيانات كثيرة باسم واحد

كيف نخزن علامات 30 طالبًا من دون ثلاثين متغيرًا؟

تطبيق عملي

أمثلة C++17

رسم توضيحي

فهم قبل الحفظ

مستوحاة من برمجة 1 · صفحات المصدر 109-116 · إعداد تربوي: م.براءة علي رحيمو

++C

01 · المشكلة أولاً

قبل التعريف... لماذا نحتاج هذه الفكرة؟

كيف نخزن علامات 30 طالبًا من دون ثلاثين متغيرًا؟

مشهد من الواقع

إذا كان لديك أسماء 30 طالبًا وعلاماتهم، إنشاء grade1 و grade2... يجعل المعالجة مرهقة.

أين المشكلة في الطريقة الساذجة؟

البيانات المتشابهة تتكرر؛ والحلقة وحدها لا تكفي دون بنية يمكن الوصول إلى عناصرها عبر فهرس.

كيف تساعدنا الفكرة؟

المصفوفة تخزن عناصر من النوع نفسه؛ يصل البرنامج لكل عنصر عبر index يبدأ من 0 ويستخدم حلقة واحدة.

شبهها بما تعرفه: كأن لديك صف خزائن مرقمة بدءًا من الصفر؛ تذهب مباشرة للخزانة التي تحتاجها.

لا تحفظ كلمة جديدة قبل أن تعرف المشكلة التي وُجدت لتساعد في حلها.

02 · النموذج العقلي

اجعل المفهوم مرئيًا

ثلاث أفكار رئيسية ثم مسار واضح يربط بينها.

int grades[5]

خمس خانات صحيحة متجاورة في المصفوفة.

Index 0..4

أول عنصر رقم 0 وآخر عنصر رقم 4.

Bounds

الوصول إلى خانة خارج الحدود في مصفوفة ++C يسبب سلوكًا غير معرّف.

الرسم التوضيحي: كيف تعمل الفكرة؟

```
int grades[5] = {80, 70, 90, 60, 100}
```

0	1	2	3	4
80	70	90	60	100

Index: يبدأ 0 · آخر فهرس 4

السؤال الأهم: كيف نخزن علامات 30 طالبًا من دون ثلاثين متغيرًا؟

03 · مثال عملي أول

شغل وافهم كل سطر

متوسط علامات خمسة طلاب

C++17 / source.cpp

```
#include <iostream>
using namespace std;
int main() {
    int grades[5] = {80, 70, 90, 60, 100};
    int sum = 0;
    for (int i=0; i<5; ++i)
        sum += grades[i];
    cout << "Average = " << sum / 5.0;
}
```

الخرج المتوقع Average = 80

لماذا كتبنا البرنامج بهذا الشكل؟

المصفوفة تحفظ القيم في مكان منظم، وتسمح بالمرور عليها بحلقة واحدة.

💡 قبل تجربة البرنامج: حاول توقع النتيجة على الورق، ثم قارن توقّعتك بالخرج الفعلي.

04 · مثال إضافي

نفس المبدأ، حالة استخدام مختلفة

اكتشاف أكبر عنصر

C++17 / source.cpp

```
#include <iostream>
using namespace std;
int main() {
    int data[4] = {12, 19, 7, 15};
    int best = data[0];
    for (int i=1; i<4; ++i)
        if (data[i] > best) best = data[i];
    cout << best;
}
```

الخرج المتوقع 19

ما الذي نتعلمه من المثال الثاني؟

نبدأ بأول عنصر لأنه موجود فعلاً؛ ثم نقارن باقي العناصر.

جرب تغيير قيمة واحدة فقط، ثم توقع أثرها: هل تتغير مسار البرنامج أم الناتج فقط؟

05 · مختبر الأخطاء

الخطأ ليس للحفظ... اكتشف سببه

قارن الكود الخاطئ بالصحيح ولا تكتفِ بمعرفة الإجابة.

الكود الذي يحتاج انتباهًا

```
int a[3]={1,2,3}; cout << a[3];
```

تصحيح وتفسير

```
int a[3]={1,2,3}; cout << a[2];
```

لماذا يُعد هذا خطأ؟

عندما يكون الحجم 3 فإن الفهارس 0 و 1 و 2 فقط؛ $a[3]$ خارج الحدود.

جرب بنفسك

تحدي 01

احسب عدد العلامات الناجحة في مصفوفة.

تحدي 02

بدّل البرنامج ليحسب أقل قيمة.

تحدي 03

اسأل: ماذا يحصل لو بدأت $i=1$ بدل 0؟

06 · تثبيت الفهم

ماذا يجب أن يبقى في ذاكرتك؟

لا تقرأ القواعد فقط؛ اربط كل قاعدة بخطأ تمنعه أو مشكلة تحلها.

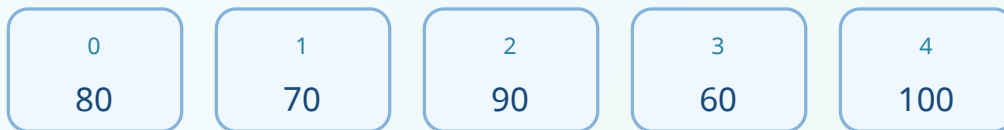
✓ حجم 5 لا يعني وجود عنصر `index=5`.

✓ المصفوفة تحل تنظيم البيانات المتجانسة.

✓ اربط الحلقة دومًا بحد صحيح للحجم.

الرسم التوضيحي: كيف تعمل الفكرة؟

```
int grades[5] = {80, 70, 90, 60, 100}
```



Index: يبدأ 0 · آخر فهرس 4

تحضير الكويز التفاعلي

قبل فتح اختبار هذه المحاضرة، حاول شرح المثال الأول لشخص آخر دون النظر إلى الشيفرة. إذا فهم لماذا تعمل الخطوات، فأنت جاهز.

الجسر إلى المحاضرة التالية

الخطوة التالية: ترتيب البيانات على شكل صفوف وأعمدة، ثم تمثيل النصوص.

المرجع: «برمجة 1» لم. براءة سراقبي، الصفحات 109–116 من الملف المرفق. الأمثلة والشروح الإضافية صيغت للأغراض التعليمية وتستخدم C++17.