

المحاضرة 02 · الخطة قبل اللغة

02

الخوارزميات وشبه الكود: فكّر قبل أن تكتب

كيف أضمن أن خطواتي تؤدي إلى الإجابة الصحيحة كل مرة؟

تطبيق عملي

أمثلة C++17

رسم توضيحي

فهم قبل الحفظ

مستوحاة من برمجة 1 · صفحات المصدر 9-14 · إعداد تربوي: م.براءة علي رحيمو

++C

01 · المشكلة أولاً

قبل التعريف... لماذا نحتاج هذه الفكرة؟

كيف أضمن أن خطواتي تؤدي إلى الإجابة الصحيحة كل مرة؟

مشهد من الواقع

إذا سُئلت عن أكبر عدد من ثلاثة أعداد، فقول «أعرف الأكبر» ليس خوارزمية؛ نحتاج خطوات تفصيلية غير ملتبسة.

أين المشكلة في الطريقة الساذجة؟

البرنامج قد يبدو واضحًا للإنسان لكنه غامض للحاسوب: متى نقارن؟ ماذا نفعل عند التساوي؟ ومتى نتوقف؟

كيف تساعدنا الفكرة؟

نكتب Pseudocode يستخدم READ و IF و WHILE و WRITE، ونجربه على مدخلات صغيرة قبل تحويله إلى لغة حقيقية.

شبهها بما تعرفه: الخوارزمية مثل تعليمات تركيب قطعة أثاث: خطوة محددة بعد أخرى، مع نهاية معروفة.

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.

Algorithm

خطوات منتهية واضحة تحوّل مدخلات إلى مخرجات.

Pseudocode

وصف شبه برمجي مستقل عن قواعد لغة محدّدة.

Dry run

تتبع القيم يدويًا لكشف أخطاء المنطق قبل تشغيل البرنامج.

خريطة المفهوم: من الفكرة إلى التنفيذ



السؤال الأهم: كيف أضمن أن خطواتي تؤدي إلى الإجابة الصحيحة كل مرة؟

03 · مثال عملي أول

شغل وافهم كل سطر

تتبع أقصى قيمة بثلاثة أعداد

C++17 / source.cpp

```
#include <iostream>
using namespace std;
int main() {
    int a = 9, b = 12, c = 7;
    int largest = a;
    if (b > largest) largest = b;
    if (c > largest) largest = c;
    cout << largest;
}
```

الخرج المتوقع 12

لماذا كتبنا البرنامج بهذا الشكل؟

نبدأ بفرضية ثم نحدّثها؛ عند انتهاء المقارنات تصبح largest أكبر قيمة.

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

04 · مثال إضافي

نفس المبدأ، حالة استخدام مختلفة

خوارزمية العدّ بالمكرّر

C++17 / source.cpp

```
#include <iostream>
using namespace std;
int main() {
    int negative = 0;
    int values[5] = {-3, 4, 0, -1, 8};
    for (int i = 0; i < 5; ++i)
        if (values[i] < 0) ++negative;
    cout << "Negative = " << negative;
}
```

الخرج المتوقع Negative = 2

ما الذي نتعلمه من المثال الثاني؟

كل قيمة تفحص مرة واحدة، والعدّاد يزيد فقط عندما يتحقق الشرط.

جرب تغيير قيمة واحدة فقط، ثم توقّع أثرها: هل تغيّر 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

```
while (x < 10) { cout << x; }
```

تصحيح وتفسير

```
while (x < 10) { cout << x; ++x; }
```

لماذا يُعد هذا خطأ؟

إن لم تتغير قيمة الشرط، فقد لا تنتهي الحلقة. إنه شرط أساسي لصحة الخوارزمية.

جرب بنفسك

تحدي 01

اكتب Pseudocode يحسب مساحة مستطيل.

تحدي 02

تتبع يدويًا قيم largest للأعداد 3 ثم 9 ثم 5.

تحدي 03

صمّم خوارزمية للقاسم المشترك الأكبر ووضح شرط توقفها.

06 · تثبيت الفهم

ماذا يجب أن يبقى في ذاكرتك؟

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

✓ الخوارزمية ليست شيفرة لغة بعينها.

✓ لكل تكرار شرط استمرار وطريقة اقتراب من التوقف.

✓ التتبع اليدوي يكشف القيم الخاطئة مبكرًا.

خريطة المفهوم: من الفكرة إلى التنفيذ



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

سنحوّل خطوات الخوارزمية إلى أشكال مرئية باستخدام Flowchart.

المرجع: «برمجة 1» لم. براءة سراقبي، الصفحات 9–14 من الملف المرفق. الأمثلة والشروح الإضافية صيغت للأغراض التعليمية وتستخدم C++17.