

المحاضرة 12 · قسم لتفهم

12

الدوال: حوّل التكرار إلى أدوات قابلة لإعادة الاستخدام

لماذا نجمع عشرين مهمة داخل main إذا كان يمكن تقسيمها؟

تطبيق عملي

أمثلة C++17

رسم توضيحي

فهم قبل الحفظ

مستوحاة من برمجة 1 · صفحات المصدر 129-141 · إعداد تربوي: م.براءة علي رحيمو

++C

01 · المشكلة أولاً

قبل التعريف... لماذا نحتاج هذه الفكرة؟

لماذا نجمع عشرين مهمة داخل main إذا كان يمكن تقسيمها؟

مشهد من الواقع

تطبيق لحساب مساحات أشكال مختلفة: إذا كتبت قواعد الحساب مرارًا داخل main يصبح تعديل معادلة واحدة مشكلة متكررة.

أين المشكلة في الطريقة الساذجة؟

الدالة الطويلة تسبب تكرارًا وصعوبة الاختبار؛ والكود لا يوضح النية حين تختلط الحسابات مع العرض والإدخال.

كيف تساعدنا الفكرة؟

نمنح كل دالة مسؤولية واحدة واسمًا معبرًا ومعاملات مدخلة وقيمة معادة عند الحاجة.

شبهها بما تعرفه: المطبخ يحتوي محطات: تحضير، طبخ، تقديم. كل محطة لها مهمة محددة، ويمكن إعادة استعمالها.

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.

Function

كتلة سلوك مسمّاة نستدعيها عند الحاجة.

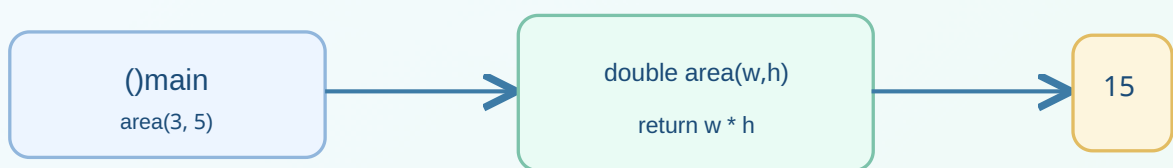
Parameters

قيم تستقبلها الدالة لتعمل على مدخلات مختلفة.

return / void

return تعيد قيمة؛ void تعني أن الدالة لا تعيد قيمة.

الرسم التوضيحي: كيف تعمل الفكرة؟



تفصل الدالة الحساب عن طريقة عرض الناتج

السؤال الأهم: لماذا نجمع عشرين مهمة داخل main إذا كان يمكن تقسيمها؟

03 · مثال عملي أول

شغل وافهم كل سطر

دالة حساب مساحة المستطيل

C++17 / source.cpp

```
#include <iostream>
using namespace std;
double area(double w, double h) {
    return w * h;
}
int main() {
    cout << area(3, 5) << '\n';
    cout << area(4, 2);
}
```

الخرج المتوقع 8 | 15

لماذا كتبنا البرنامج بهذا الشكل؟

نفس الدالة تعمل على مستطيلين دون تكرار قاعدة الضرب.

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

04 · مثال إضافي

نفس المبدأ، حالة استخدام مختلفة

دالة تجمع ودالة تعرض

C++17 / source.cpp

```
#include <iostream>
using namespace std;
int add(int a, int b) { return a + b; }
void printResult(int value) {
    cout << "Result: " << value;
}
int main() {
    int total = add(6, 4);
    printResult(total);
}
```

الخرج المتوقع Result: 10

ما الذي نتعلمه من المثال الثاني؟

فصل الحساب عن العرض يجعل اختبار add مستقلاً عن شكل الرسالة.

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

```
int add(int a,int b){ cout << a+b; }
```

تصحيح وتفسير

```
int add(int a,int b){ return a+b; }
```

لماذا يُعد هذا خطأ؟

دالة تعلن int يجب أن تعيد قيمة في المسارات المطلوبة؛ الطباعة ليست بديلاً عن return.

جرب بنفسك

تحدي 01

اكتب دالة square(int x) تعيد مربعه.

تحدي 02

قسّم حساب فاتورة إلى computeTotal وprintTotal.

تحدي 03

اجعل main تستدعي دالة مرتين بقيم مختلفة.

06 · تثبيت الفهم

ماذا يجب أن يبقى في ذاكرتك؟

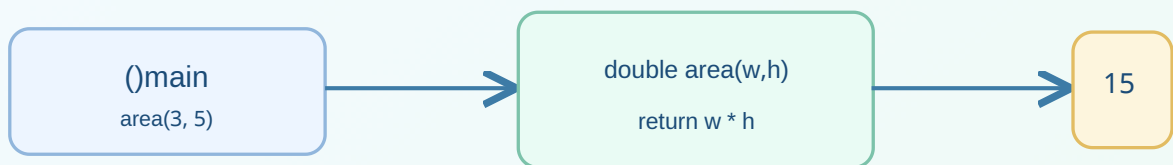
لا تقرأ القواعد فقط؛ اربط كل قاعدة بخطأ تمنعه أو مشكلة تحلها.

✓ الدالة الصغيرة تحمل فكرة واحدة.

✓ المعاملات تسمح بإعادة الاستخدام بقيم جديدة.

✓ تميّز بين «حساب قيمة» و«عرض قيمة».

الرسم التوضيحي: كيف تعمل الفكرة؟



تفصل الدالة الحساب عن طريقة عرض الناتج

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

بعدها سنرى كيف تمرر المعاملات وكيف تتحكم بنطاق المتغيرات.

المرجع: «برمجة 1» لم. براءة سراقبي، الصفحات 129-141 من الملف المرفق. الأمثلة والشروح الإضافية صيغت للأغراض التعليمية وتستخدم C++17.