

المحاضرة 05 · ذاكرة مفهومة

05

# المتغيرات وأنواع البيانات: ماذا نخزن ولماذا؟

لماذا يحتاج الكمبيوتر إلى معرفة نوع القيمة قبل تخزينها؟

تطبيق عملي

أمثلة C++17

رسم توضيحي

فهم قبل الحفظ

مستوحاة من برمجة 1 · صفحات المصدر 40-47 · إعداد تربوي: م.براءة علي رحيمو

++C

## 01 · المشكلة أولاً

# قبل التعريف... لماذا نحتاج هذه الفكرة؟

لماذا يحتاج الكمبيوتر إلى معرفة نوع القيمة قبل تخزينها؟

## مشهد من الواقع

نظام طلاب يحتفظ بالعمر والاسم ومتوسط العلامات وحالة النجاح. التعامل معها كلها كنصوص يجعل العمليات مربكة.

## أين المشكلة في الطريقة الساذجة؟

النوع الخاطئ قد يجعل القسمة تفقد الكسور أو يمنع تخزين حرف أو يتسبب بتجاوز المدى.

## كيف تساعدنا الفكرة؟

نختار int للأعداد الصحيحة وdouble للكسر وchar للحرف وstring للنص وbool لصح/خطأ، ونستخدم const للقيم الثابتة.

**شبهها بما تعرفه:** كل نوع يشبه درجًا مصممًا لغرض معين؛ لا تحفظ رقم الهاتف على أنه قيمة تحتاج عمليات حسابية.

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 / double**

الأول للعدد الصحيح والثاني لتمثيل أعداد عشرية تقريبية.

**char / string**

الأول حرف واحد والثاني سلسلة من الحروف.

**bool / const**

قيمة منطقية، وثابت لا يجوز تعديله بعد تعريفه.

## خريطة المفهوم: من الفكرة إلى التنفيذ



**السؤال الأهم:** لماذا يحتاج الكمبيوتر إلى معرفة نوع القيمة قبل تخزينها؟

## 03 · مثال عملي أول

## شغل وافهم كل سطر

ملف طالب بأنواع مختلفة

C++17 / source.cpp

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string name = "Rana";
    int age = 19;
    double average = 84.5;
    bool passed = average >= 60;
    cout << name << " / " << age << " / "
         << average << " / " << passed;
}
```

الخرج المتوقع Rana / 19 / 84.5 / 1

## لماذا كتبنا البرنامج بهذا الشكل؟

تحديد النوع يجعل معنى كل قيمة واضحاً ويسمح للمترجم بفحص العمليات.

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

## 04 · مثال إضافي

# نفس المبدأ، حالة استخدام مختلفة

الثابت يمنع تغيير قاعدة ثابتة

• • • C++17 / source.cpp

```
#include <iostream>
using namespace std;
int main() {
    const double pi = 3.14159;
    double radius = 2;
    cout << pi * radius * radius;
}
```

الخرج المتوقع تقريباً 12.5664

## ما الذي نتعلمه من المثال الثاني؟

إذا كتبت `pi=4` بعد التصريح فسيفرض المترجم تغيير قيمة `const`.

جرب تغيير قيمة واحدة فقط، ثم توقع 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 انتباهاً

```
int count; cout << count;
```

تصحيح وتفسير

```
int count = 0; cout << count;
```

## لماذا يُعد هذا خطأ؟

المتغير المحلي غير المهيأ قد يحمل قيمة غير محددة؛ تهيئة المعطيات خطوة أساسية.

## جرب بنفسك

## تحدي 01

اقترح أنواع مناسبة للاسم والسن والسعر وناجح/راسب.

## تحدي 02

لماذا قد تعطي 5/2 النتيجة 2؟ جرب 5.0/2.

## تحدي 03

عرّف enum لأيام العمل واطرح فائدة القيم المحددة.

## 06 · تثبيت الفهم

# ماذا يجب أن يبقى في ذاكرتك؟

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

✓ اسم المتغير الجيد يخدم القارئ.

✓ اختيار النوع يحدد العمليات والحدود والدقة.

✓ لا تفترض أن `int` أو `long double` له حجم ثابت على كل بيئة؛ افحص `sizeof`.

## 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

الدرس التالي يشرح ماذا تفعل العمليات الحسابية والمنطقية بهذه القيم.

المرجع: «برمجة 1» لم. براءة سراقبي، الصفحات 40–47 من الملف المرفق، الأمثلة والشروح الإضافية صيغت للأغراض التعليمية وتستخدم C++17.