KHATWA TECH

أساسيات C++ قبل الـOOP

C++ FUNDAMENTALS

مذكرة مراجعة عملية تختصر المفاهيم الضرورية قبل الدخول في Object-Oriented Programming، مع أمثلة مجرّبة وتدريبات ومشروع.

النسخة المستهدفة: C++20

إعداد: محمود منصور (Mahmoud Mansour)

المنصة: khatwa tech

التاريخ: 23 نوفمبر 2026

youtube.com/@khatwatech

34 PAGES / REVIEW NOTES / C++20

خريطة المذكرة

اتبع الصفحات بالترتيب، واستخدم التمارين للتأكد إن الفكرة ثبتت.

طريقة المذاكرة: ١) اقرأ الفكرة والجدول بسرعة. ٢) توقّع ناتج الكود قبل ما تشوف البوكس. ٣) جرّب التمارين وقارن بالحلول.

الصفحة	القسم	الموضوع
3	01	Introduction / Compile & Run
5–4	02	Variables, Types & Casting
7–6	03	Input / Output & Operators
9–8	04	Conditions
12–10	05	Loops
16–13	06	Arrays & Vectors
18–17	07	Strings
21–19	08	Functions & Recursion
23–22	09	References & Pointers
24	10	Structs
26–25	11	المشروع العملي
27	12	الأخطاء الشائعة
28	13	ورقة المراجعة
30–29	14	اختبر نفسك
33–31	15	الإجابات
34	16	Edge Cases & References

01 | Introduction

برنامج C++ بيتحوّل من Source Code لملف تنفيذي بواسطة Compiler.

نقطة البداية هي `int main()`. نستخدم `#include` لإضافة تعريفات الأدوات، و`std::cout` للطباعة.

أول برنامج قابل للتشغيل C++20

```cpp
#include <iostream>
int main() {
    std::cout << "Hello, khatwa tech\n";
    return 0;
}
```

الناتج الفعلي

```
Hello, khatwa tech
```

Compile & Run

احفظ الكود باسم `main.cpp`، ثم شغّله:

Terminal

```
g++ -std=c++20 -Wall -Wextra -pedantic main.cpp -o app
./app
```

الجزء	وظيفته
#include <iostream>	أدوات الإدخال والإخراج
int main()	نقطة بداية البرنامج
std::cout	الطباعة على الشاشة
return 0;	انتهاء البرنامج بنجاح

انتبه

اكتب كل Statement منتهية بـ ; عند اللزوم. C++ حسّاسة لحالة الحروف: Main مختلفة عن main.

Variables & Data Types | 02

المتغير اسم لمكان بيحفظ قيمة بنوع محدد.

اختيار النوع بيحدد العمليات المتاحة وطبيعة القيمة؛ `std::string` نص، و `bool` قيمة منطقية.

بيانات طالب وكورس C++20

```cpp
#include <iostream>
#include <string>
int main() {
    int age = 19;
    double price = 149.5;
    char level = 'A';
    bool enrolled = true;
    std::string course = "C++";
    std::cout << course << " " << age << "\n";
    std::cout << price << " " << level << " " << std::boolalpha << enrolled << "\n";
}
```

الناتج الفعلي

```
C++ 19
149.5 A true
```

النوع	مثال	المقصود
int	25	عدد صحيح
double	12.5	عدد عشري
char	'A'	حرف واحد
bool	true / false	منطقي
std::string	"++C"	نص

الفرق اللي يهمك

الـ char بين single quotes، والـ string بين double quotes. النوع نفسه مش بيتغير بعد إنشاء المتغير.

02 | Const, Auto & Casting

ثبّت اللي مش هيتغير، واستخدم تحويلًا صريحًا لما يكون المقصود واضحًا.

ثوابت ونوع تلقائي

C++20

```cpp
#include <iostream>
int main() {
    const double tax = 0.10;
    auto count = 3;
    double price = 200.0;
    std::cout << count << "\n";
    std::cout << price * (1 + tax) << "\n";
}
```

الناتج الفعلي

```
3
220
```

القسمة الصحيحة والتحويل

C++20

```cpp
#include <iostream>
int main() {
    int total = 9, students = 2;
    std::cout << total / students << "\n";
    std::cout << static_cast<double>(total) / students << "\n";
}
```

الناتج الفعلي

```
4
4.5
```

الصيغة	استخدمها لما
const int limit = 10;	القيمة ثابتة
auto n = 5;	النوع واضح من القيمة
static_cast<double>(n)	محتاج تحويلًا صريحًا

خطأ مشهور

القسمة بين int وint تطلع عددًا صحيحًا حتى لو خزّنت الناتج في double. حوّل قبل القسمة.

Input / Output | 03

cin تقرأ المدخلات، وgetline تقرأ سطرًا كاملًا بما فيه المسافات.

درجة من المستخدم

C++20

```cpp
#include <iostream>
int main() {
    int score{};
    std::cin >> score;
    std::cout << "Score: " << score << "\n";
}
```

الإدخال للتجربة

```
85
```

الناتج الفعلي

```
Score: 85
```

اسم فيه مسافات

C++20

```cpp
#include <iostream>
#include <string>
int main() {
    std::string name;
    std::getline(std::cin, name);
    std::cout << "Hello, " << name << "\n";
}
```

الإدخال للتجربة

```
Mona Ali
```

الناتج الفعلي

```
Hello, Mona Ali
```

الأداة	الفرق
std::cin >> value	تقرأ القيمة التالية وتترك نهاية السطر
std::getline(std::cin, text)	تقرأ السطر لحد Enter
std::cout << value	تطبع
"n\"	سطر جديد

انتبه

بعد cin >> number، لو هتستخدم getline، تعامل مع سطر الإدخال المتبقي. ممكن تستخدم std::getline(std::cin >> std::ws, name) لو تجاهل المسافات الأولية مناسب.

Operators | 03

العمليات تحسب وتقارن وتربط شروطًا وتحدّث قيم المتغيرات.

Arithmetic / Comparison C++20

```cpp
#include <iostream>
int main() {
    int a = 10, b = 3;
    std::cout << a + b << " " << a / b << " " << a % b << "\n";
    std::cout << std::boolalpha << (a > b) << " " << (a == b) << "\n";
}
```

الناتج الفعلي

```
13 3 1
true false
```

Logical / Assignment C++20

```cpp
#include <iostream>
int main() {
    int score = 78;
    bool passed = score >= 50 && score <= 100;
    score += 2;
    std::cout << std::boolalpha << passed << "\n";
    std::cout << score << "\n";
}
```

الناتج الفعلي

```
true
80
```

العامل	معناه
%	باقي قسمة أعداد صحيحة
== / !=	يساوي / لا يساوي
&& / || / !	و / أو / ليس
+= / -=	تحديث القيمة
++ / --	زيادة / نقصان واحد

خطأ مشهور

علامة = للإسناد، و== للمقارنة؛ استخدم أقواسًا للتعبيرات المركبة عشان تبقى واضحة.

If / Else | 04

اختيار مسار تنفيذ حسب شرط نتيجته true أو false.

تقدير الطالب C++20

```cpp
#include <iostream>
int main() {
    int score = 82;
    if (score >= 90) {
        std::cout << "Excellent\n";
    } else if (score >= 50) {
        std::cout << "Passed\n";
    } else {
        std::cout << "Failed\n";
    }
}
```

الناتج الفعلي

```
Passed
```

شرطين معًا C++20

```cpp
#include <iostream>
int main() {
    int age = 20;
    bool paid = true;
    if (age >= 18 && paid) std::cout << "Access granted\n";
    else std::cout << "Access denied\n";
}
```

الناتج الفعلي

```
Access granted
```

الجزء	بيحصل إمتى
if	لو الشرط الأول صحيح
else if	لو اللي قبله فشل وده صحيح
else	لو كل الشروط فشلت

جرّب بنفسك

بدّل score إلى 49 و90، وتوقّع الرسالة قبل التشغيل.

Switch & Ternary | 04

switch مناسب للاختيارات الثابتة، و?: مناسب لقرار بسيط من فرعين.

Switch في قائمة كورسات

C++20

```cpp
#include <iostream>
int main() {
    int choice = 2;
    switch (choice) {
        case 1: std::cout << "C++\n"; break;
        case 2: std::cout << "Python\n"; break;
        default: std::cout << "Unknown\n";
    }
}
```

الناتج الفعلي

```
Python
```

Ternary Operator

C++20

```cpp
#include <iostream>
int main() {
    int score = 45;
    std::cout << (score >= 50 ? "Passed" : "Failed") << "\n";
}
```

الناتج الفعلي

```
Failed
```

الأداة	أنسب استخدام
if / else	شروط معقدة أو حدود
switch	مطابقة قيم ثابتة من نوع مناسب
condition ? A : B	اختيار قيمة من احتمالين

انتبه

وجود break في switch يمنع الانتقال للحالة التالية. لا تحوّل كل if إلى ternary لو ده هيقلل الوضوح.

For Loop | 05

لما تعرف عدد التكرارات، for بتجمع البداية والشرط والتحديث.

طباعة 1 إلى 5

C++20

```cpp
#include <iostream>
int main() {
    for (int i = 1; i <= 5; ++i) {
        std::cout << i << " ";
    }
    std::cout << "\n";
}
```

الناتج الفعلي

```
1 2 3 4 5
```

مجموع أول أربع درجات

C++20

```cpp
#include <iostream>
int main() {
    int scores[] = {10, 20, 30, 40};
    int total = 0;
    for (int score : scores) total += score;
    std::cout << "Total: " << total << "\n";
}
```

الناتج الفعلي

```
Total: 100
```

الجزء	دوره
int i = 1	بداية العد
i <= 5	الاستمرار
i++	التحديث بعد كل دورة
for (int x : values)	مرور على العناصر

انتبه

الفرق بين < و<= ممكن يغيّر عدد المرات. تأكد من حدود الحلقة.

While / Do While | 05

while قد لا تنفّذ ولا مرة؛ do while تنفّذ مرة على الأقل.

While مع عدّاد C++20

```cpp
#include <iostream>
int main() {
    int left = 3;
    while (left > 0) {
        std::cout << left << " ";
        --left;
    }
    std::cout << "Go!\n";
}
```

الناتج الفعلي

```
3 2 1 Go!
```

Do While C++20

```cpp
#include <iostream>
int main() {
    int attempts = 0;
    do {
        ++attempts;
        std::cout << "Attempt " << attempts << "\n";
    } while (attempts < 2);
}
```

الناتج الفعلي

```
Attempt 1
Attempt 2
```

الحلقة	اختارها لما
for	عدد التكرارات معروف
while	عدد التكرارات مرتبط بشرط
do while	لازم جسم الحلقة ينفذ مرة

خطأ مشهور

نسيان تحديث عدّاد while يعمل Infinite Loop. اكتب إيه اللي هيخلي الشرط False.

05 | Nested Loops & Controls

تقدر تداخل حلقات، وتستخدم break للخروج أو continue لتخطي دورة.

جدول صغير باستخدام Nested Loops

C++20

```cpp
#include <iostream>
int main() {
    for (int row = 1; row <= 2; ++row) {
        for (int col = 1; col <= 3; ++col)
            std::cout << row * col << " ";
        std::cout << "\n";
    }
}
```

الناتج الفعلي

```
1 2 3
2 4 6
```

Break و Continue

C++20

```cpp
#include <iostream>
int main() {
    for (int n = 1; n <= 7; ++n) {
        if (n == 3) continue;
        if (n == 6) break;
        std::cout << n << " ";
    }
    std::cout << "\n";
}
```

الناتج الفعلي

```
1 2 4 5
```

الأداة	التأثير
break	خروج من أقرب Loop أو Switch
continue	تخطي بقية الدورة الحالية
Nested loop	الحلقة الداخلية تتكرر لكل دورة خارجية

الفرق اللي يهمك

break يوقف الحلقة نفسها، continue يكمّل للدورة اللي بعدها.

Fixed Arrays | 06

Array تخزّن عناصر من نفس النوع، والفهرس يبدأ من صفر.

قراءة وتعديل عنصر C++20

```cpp
#include <iostream>
int main() {
    int scores[4] = {70, 80, 90, 60};
    scores[3] = 75;
    std::cout << scores[0] << " " << scores[3] << "\n";
    int total = 0;
    for (int score : scores) total += score;
    std::cout << "Total: " << total << "\n";
}
```

الناتج الفعلي

```
70 75
Total: 315
```

قراءة حدود Array

في `int a[4]` الفهارس الصالحة 0 و1 و2 و3. محاولة قراءة `a[4]` سلوك غير معرّف (Undefined Behavior).

الخاصية	Fixed Array
الطول	ثابت بعد الإنشاء
الفهرس	يبدأ من صفر
تعديل عنصر	a[index] = value
طباعة العناصر	باستخدام loop

انتبه

ما تستخدمش فهرس برّه الحدود؛ المشكلة مش لازم تظهر برسالة خطأ واضحة.

Two-dimensional Arrays | 06

2D Array مناسبة لجدول صفوف وأعمدة ثابت.

جدول درجات مجموعتين

C++20

```cpp
#include <iostream>
int main() {
    int marks[2][3] = {{8, 7, 9}, {6, 10, 8}};
    for (int row = 0; row < 2; ++row) {
        int sum = 0;
        for (int col = 0; col < 3; ++col) sum += marks[row][col];
        std::cout << "Group " << row + 1 << ": " << sum << "\n";
    }
}
```

الناتج الفعلي

```
Group 1: 24
Group 2: 24
```

طريقة التفكير

أول فهرس للصف `marks[row][col]`، والثاني للعمود. استخدم Loop للصفوف وLoop داخلية للأعمدة.

التعبير	النتيجة
marks[0][0]	أول عنصر في الصف الأول
marks[1][2]	آخر عنصر في الصف الثاني
marks[2][0]	خارج الحدود: ممنوع

جرّب بنفسك

غيّر درجة واحدة واحسب مجموع الصف يدويًا قبل التشغيل.

06 | std::vector

Vector مصفوفة ديناميكية حجمها ممكن يكبر أو يصغر.

إضافة عناصر وقراءتها

C++20

```cpp
#include <iostream>
#include <vector>
int main() {
    std::vector<int> scores{70, 90};
    scores.push_back(85);
    std::cout << "Count: " << scores.size() << "\n";
    std::cout << scores.front() << " " << scores.back() << "\n";
    for (int score : scores) std::cout << score << " ";
    std::cout << "\n";
}
```

الناتج الفعلي

```
Count: 3
70 85
70 90 85
```

الأداة	وظيفتها
push_back(x)	إضافة في الآخر
pop_back()	حذف آخر عنصر (لو مش فاضي)
size()	عدد العناصر
empty()	هل فاضي؟
at(i)	قراءة بحدود مفحوصة

الفرق اللي يهمك

استخدم std::vector غالبًا لو الحجم بيتغير؛ Fixed Array مفيدة لتعلّم الذاكرة والحجم الثابت.

Vectors: Safe Operations | 06

قبل front/back/pop_back تأكد إن الـVector مش فاضي.

فحص الأمان وتحديث العناصر C++20

```cpp
#include <iostream>
#include <vector>
int main() {
    std::vector<int> prices{100, 150, 200};
    if (!prices.empty()) {
        prices.at(1) += 10;
        prices.pop_back();
    }
    std::cout << "Count: " << prices.size() << "\n";
    for (int price : prices) std::cout << price << " ";
    std::cout << "\n";
}
```

الناتج الفعلي

```
Count: 2
100 160
```

حساب المتوسط بأمان C++20

```cpp
#include <iostream>
#include <vector>
int main() {
    std::vector<int> scores{};
    if (scores.empty()) std::cout << "No scores\n";
    else {
        int total = 0;
        for (int s : scores) total += s;
        std::cout << static_cast<double>(total) / scores.size() << "\n";
    }
}
```

الناتج الفعلي

```
No scores
```

انتبه

at(i) يرمي std::out_of_range عند فهرس غير صحيح؛ operator[] مش بيعمل فحص حدود.

std::string Basics | 07

std::string بتسهّل تخزين النصوص ودمجها وقراءة حروفها.

دمج نصوص وقراءة الطول C++20

```cpp
#include <iostream>
#include <string>
int main() {
    std::string first = "Data";
    std::string second = "Science";
    std::string course = first + " " + second;
    std::cout << course << "\n";
    std::cout << "Length: " << course.size() << "\n";
    std::cout << "First: " << course.at(0) << "\n";
}
```

الناتج الفعلي

```
Data Science
Length: 12
First: D
```

الأداة	النتيجة
text.size()	عدد وحدات char في النص
text.empty()	هل النص فاضي
text.at(i)	حرف مع فحص حدود
a + b	دمج String

انتبه

size() بيحسب وحدات char (bytes مع UTF-8)، مش بالضرورة عدد الحروف العربية الظاهرة. الأمثلة النصية هنا ASCII.

String Methods | 07

استخدم find للبحث وsubstr لقص جزء، مع فحص نتيجة البحث.

البحث والقص

C++20

```cpp
#include <iostream>
#include <string>
int main() {
    std::string course = "Modern C++ Basics";
    auto pos = course.find("C++");
    if (pos != std::string::npos) {
        std::cout << "Found at: " << pos << "\n";
        std::cout << course.substr(pos, 3) << "\n";
    }
}
```

الناتج الفعلي

```
Found at: 7
C++
```

تعديل نص

C++20

```cpp
#include <iostream>
#include <string>
int main() {
    std::string track = "C++";
    track.append(" Fundamentals");
    std::cout << track << "\n";
    std::cout << std::boolalpha << (track == "C++ Fundamentals") << "\n";
}
```

الناتج الفعلي

```
C++ Fundamentals
true
```

خطأ مشهور

find يرجع std::string::npos لو ملقتش النص؛ ما تفترضش إن النتيجة رقم صالح للفهرسة.

Functions | 08

Function بتقسّم البرنامج لمهام صغيرة قابلة لإعادة الاستخدام.

إعلان وتعريف واستدعاء

C++20

```cpp
#include <iostream>
int add(int a, int b);
int main() {
    std::cout << add(10, 7) << "\n";
}
int add(int a, int b) {
    return a + b;
}
```

الناتج الفعلي

```
17
```

void Function

C++20

```cpp
#include <iostream>
#include <string>
void welcome(const std::string& name) {
    std::cout << "Welcome, " << name << "\n";
}
int main() {
    welcome("Omar");
}
```

الناتج الفعلي

```
Welcome, Omar
```

الجزء	الفكرة
return type	نوع القيمة الراجعة
parameters	قيم تستقبلها function
return	ترجع القيمة وتنهي الاستدعاء
void	لا ترجع قيمة

انتبه

إعلان function قبل main يفيد لو تعريفها بعد main. اسم الدالة والأنواع لازم يتطابقوا.

Overloading & Defaults | 08

نفس الاسم ممكن يخدم أنواعًا مختلفة، والـDefault بيسهّل الاستدعاء.

Function Overloading

C++20

```cpp
#include <iostream>
int multiply(int a, int b) { return a * b; }
double multiply(double a, double b) { return a * b; }
int main() {
    std::cout << multiply(3, 4) << "\n";
    std::cout << multiply(1.5, 2.0) << "\n";
}
```

الناتج الفعلي

```
12
3
```

Default Arguments

C++20

```cpp
#include <iostream>
int finalPrice(int price, int discount = 0) {
    return price - discount;
}
int main() {
    std::cout << finalPrice(100) << "\n";
    std::cout << finalPrice(100, 15) << "\n";
}
```

الناتج الفعلي

```
100
85
```

القاعدة	ملحوظة
Overloading	عدد أو أنواع الـParameters تختلف
Default argument	استخدم قيمة افتراضية آخر قائمة Parameters
Return type فقط	مش كفاية لتمييز Overload

جرّب بنفسك

اكتب function باسم area تحسب مساحة مستطيل، وواحدة تحسب مساحة مربع.

Recursion | 08

Function تقدر تستدعي نفسها، بشرط توقف واضح.

مجموع من 1 إلى n

C++20

```cpp
#include <iostream>
int sumTo(int n) {
    if (n <= 0) return 0;
    return n + sumTo(n - 1);
}
int main() {
    std::cout << sumTo(4) << "\n";
}
```

الناتج الفعلي

```
10
```

Factorial

C++20

```cpp
#include <iostream>
int factorial(int n) {
    if (n <= 1) return 1;
    return n * factorial(n - 1);
}
int main() {
    std::cout << factorial(5) << "\n";
}
```

الناتج الفعلي

```
120
```

الجزء	التفسير
Base case	شرط التوقف
Recursive case	استدعاء بمشكلة أصغر
sumTo(4)	1 + 2 + 3 + 4

انتبه

لو مفيش Base Case أو حجم المشكلة مش بيقل، ممكن يحصل Stack Overflow. المثال factorial مفترض n غير سالب.

References & Passing | 09

Reference اسم تاني لنفس المتغير؛ Pass by Value يرسل نسخة، وبالـReference يقدر يعدّل الأصل.

Value vs Reference

C++20

```cpp
#include <iostream>
void addByValue(int n) { n += 5; }
void addByRef(int& n) { n += 5; }
int main() {
    int score = 50;
    addByValue(score);
    std::cout << score << "\n";
    addByRef(score);
    std::cout << score << "\n";
}
```

الناتج الفعلي

```
50
55
```

Const Reference لنص كبير

C++20

```cpp
#include <iostream>
#include <string>
void show(const std::string& name) {
    std::cout << "Student: " << name << "\n";
}
int main() { show("Salma"); }
```

الناتج الفعلي

```
Student: Salma
```

التمرير	التأثير
int x	نسخة مستقلة
int& x	وصول للأصل وتعديله
const std::string& x	قراءة بدون نسخ وبدون تعديل

انتبه

Reference لازم ترتبط بكائن مناسب عند تهيئتها، ومش بتغيّر الهدف اللي بتشير له بعد كده.

09 | Pointers & nullptr

Pointer يخزن عنوانًا، والـ* تصل للقيمة عند عنوان صالح.

عنوان متغير وتعديل قيمته

C++20

```cpp
#include <iostream>
int main() {
    int score = 70;
    int* ptr = &score;
    *ptr += 10;
    std::cout << score << "\n";
    std::cout << *ptr << "\n";
}
```

الناتج الفعلي

```
80
80
```

التحقق قبل Dereference

C++20

```cpp
#include <iostream>
int main() {
    int* ptr = nullptr;
    if (ptr == nullptr) std::cout << "No address\n";
    int value = 5;
    ptr = &value;
    if (ptr != nullptr) std::cout << *ptr << "\n";
}
```

الناتج الفعلي

```
No address
5
```

العلامة	المعنى
&value	عنوان المتغير
int* ptr	Pointer لـ int
*ptr	القيمة عند العنوان
nullptr	لا يشير لكائن

انتبه

ما تعملش *ptr لو ptr == nullptr أو بيشير لذاكرة غير صالحة. هنتجنب new/delete يدويًا في الأمثلة دي.

Structs Before OOP | 10

struct بتجمع بيانات مرتبطة في نوع واحد، من غير الدخول في Classes أو مبادئ OOP.

Student وFunction للعرض

C++20

```cpp
#include <iostream>
#include <string>
struct Student {
    std::string name;
    int score;
};
void printStudent(const Student& student) {
    std::cout << student.name << ": " << student.score << "\n";
}
int main() {
    Student student{"Mona", 88};
    student.score += 2;
    printStudent(student);
}
```

الناتج الفعلي

```
Mona: 90
```

التعبير	وظيفته
struct Student { ... };	تعريف نوع يجمع حقولًا
Student s{"Mona", 88};	إنشاء قيمة
s.score	الوصول لعضو
function(const Student& s)	تمرير قراءة بدون نسخ

الفرق اللي يهمك

احنا بنستخدم struct هنا كحاوية بيانات فقط. تأجيل Constructors وMethods وEncapsulation للـOOP.

11 | المشروع: Student Grades Manager

بيجمع Input من المستخدم داخل Vector، وStructs، وLoops، وFunctions، وحساب متوسط بأمان.

السيناريو: هنقرأ عدد الطلاب ثم الاسم والدرجة لكل واحد (اسم من كلمة واحدة). هنطبع حالة النجاح عند حد 50 والمتوسط وأعلى درجة.

الكود الكامل للمشروع C++20

```cpp
#include <iostream>
#include <iomanip>
#include <string>
#include <vector>
struct Student {
    std::string name;
    int score;
};
std::string status(int score) {
    return score >= 50 ? "Passed" : "Failed";
}
double average(const std::vector<Student>& students) {
    if (students.empty()) return 0.0;
    int total = 0;
    for (const Student& s : students) total += s.score;
    return static_cast<double>(total) / students.size();
}
int highest(const std::vector<Student>& students) {
    if (students.empty()) return 0;
    int top = students.front().score;
    for (const Student& s : students)
        if (s.score > top) top = s.score;
    return top;
}
int main() {
    int count{};
    if (!(std::cin >> count) || count < 0) {
        std::cout << "Invalid count\n";
        return 0;
    }
    std::vector<Student> students;
    for (int i = 0; i < count; ++i) {
        Student s{};
        if (!(std::cin >> s.name >> s.score) || s.score < 0 || s.score > 100) {
            std::cout << "Invalid input\n";
            return 0;
        }
        students.push_back(s);
    }
    if (students.empty()) std::cout << "No students\n";
    for (const Student& s : students)
        std::cout << s.name << ": " << status(s.score) << "\n";
    std::cout << std::fixed << std::setprecision(2);
    std::cout << "Average: " << average(students) << "\n";
    std::cout << "Highest: " << highest(students) << "\n";
}
```

الإدخال للتجربة

```
3
Mona 90
Omar 45
Salma 75
```

الناتج الفعلي

```
Mona: Passed
Omar: Failed
Salma: Passed
Average: 70.00
Highest: 90
```

11 | تحليل النتيجة وتطوير المشروع

افهم ليه كل سطر ظهر بالشكل ده، وبعدها طوّر المشروع بنفسك.

تفسير الناتج

السطر	السبب
Mona: Passed	90 أكبر من أو يساوي 50
Omar: Failed	45 أقل من 50
Salma: Passed	75 أكبر من أو يساوي 50
Average: 70.00	(90 + 45 + 75) / 3، وتنسيق خانتين
Highest: 90	أكبر درجة موجودة

3 أفكار لتطوير المشروع

- خلّي الاسم يقبل مسافات باستخدام getline، وتعامل مع بقايا السطر بعد cin.
- أضف Function تحسب أقل درجة، مع التعامل مع Vector فاضي.
- أضف تقريرًا يوضح عدد الناجحين والراسبين ونسبة النجاح.

اختبارات مهمة قبل التطوير

الحالة	التعامل الصحيح
عدد الطلاب = 0	تظهر No students ثم Average: 0.00 وHighest: 0
درجة = 50	Passed
درجات متساوية	Highest يساوي نفس الدرجة

> **جرّب بنفسك**
> اقرأ Function average واشرح ليه التحويل إلى double تم قبل القسمة.

12 | الأخطاء الشائعة

رسالة Compiler مش دائمًا بتكون مطابقة بين الأدوات، لكن السبب غالبًا واحد.

رسالة شائعة / العرض	السبب المحتمل	تعمل إيه
expected ';' before ...	نسيان ; أو قوس قبل السطر	راجع السطر السابق والأقواس
x' was not declared'	اسم متغير خارج Scope أو مكتوب غلط	راجع التعريف وحالة الحروف
undefined reference to ...	دالة معلنة بدون تعريف مطابق أو ملف غير مربوط	طابق Signatures واربط كل الملفات
no matching function for call	عدد أو نوع Arguments مش مطابق	راجع Parameters وأنواعها
division by zero / crash	مقام = 0 في قسمة صحيحة	تحقق من المقام قبل القسمة
std::out_of_range	استخدام vector.at أو string.at بفهرس خاطئ	تحقق من index < size
Segmentation fault (ممكن)	Dereference لـPointer غير صالح	تحقق من العنوان والصلاحية
Infinite loop	شرط تكرار لا ينتهي	راجع تحديث العداد وشرط التوقف
نتيجة قسمة ناقصة	int / int ينتج قسمة صحيحة	استخدم <static_cast<double

انتبه

رسائل Runtime وCompiler بتختلف حسب النظام والأداة. Undefined Behavior ممكن يعدّي من غير رسالة واضحة.

13 | ورقة المراجعة السريعة

الصيغ الأكثر استخدامًا في صفحة واحدة؛ ارجع لها بعد حل التمارين.

عايز أعمل كذا	الصيغة المختصرة
أطبع نص	;"std::cout << "Hi\n
أقرأ رقم	;std::cin >> number
أقرأ سطر	;std::getline(std::cin, name)
متغير ثابت	;const int limit = 100
تحويل إلى double	static_cast<double>(total)
شرط	{ ... } if (score >= 50)
اختيار سريع	"ok ? "Yes" : "No
اختيار حالات	switch (choice) { case 1: ... break; default: ...; }
Loop بعدد	{ ... } for (int i = 0; i < n; ++i)
Loop بشرط	{ ... } while (condition)
Array ثابت	;int a[3] = {1, 2, 3}
Vector متغير	;std::vector<int> v{1, 2}
إضافة عنصر	;v.push_back(3)
مرور على العناصر	{ ... } for (int x : v)
نص	;"++std::string title = "C
البحث في نص	text.find("C++") != std::string::npos
Function	int add(int a,int b) { return a+b; }
Pass by Reference	void update(int& x) { ++x; }
Const Reference	;void show(const std::string& s)
Pointer	;int* p = &value
فحص Pointer	{ ... } if (p != nullptr)
Struct	;struct Student { std::string name; int score; }
فحص Container	{ ... } if (!v.empty())

6 / khatwa tech أسئلة توقع الناتج / 29 — 34

14 | اختبر نفسك: توقع الناتج

اقرأ كل برنامج وتوقع الناتج حرفيًا قبل ما تفتح صفحة الإجابات.

Q1 هل القسمة صحيحة ولا عشرية؟

```cpp
#include <iostream>
int main() {
    int x = 7;
    std::cout << x / 2 << " " << x % 2 << "\n";
}
```

الناتج: .....

Q2 حد النجاح هنا شامل 50؟

```cpp
#include <iostream>
int main() {
    int score = 50;
    std::cout << (score >= 50 ? "Passed" : "Failed") << "\n";
}
```

الناتج: .....

Q3 إيه الأرقام اللي اتخطّاها continue؟

```cpp
#include <iostream>
int main() {
    for (int i = 1; i <= 5; ++i) {
        if (i % 2 == 0) continue;
        std::cout << i << " ";
    }
    std::cout << "\n";
}
```

الناتج: .....

Q4 عدد العناصر والـIndex الصحيح؟

```cpp
#include <iostream>
#include <vector>
int main() {
    std::vector<int> v{2, 4};
    v.push_back(6);
    std::cout << v.size() << " " << v.at(1) << "\n";
}
```

الناتج: .....

Q5 Pass by Value بيغيّر الأصل؟

```cpp
#include <iostream>
void change(int x) { x = 99; }
int main() {
    int n = 5;
    change(n);
    std::cout << n << "\n";
}
```

الناتج: .....

Q6 إيه اللي بيتغيّر عبر Pointer؟

```cpp
#include <iostream>
struct Item { int price; };
int main() {
    Item item{80};
    int* ptr = &item.price;
    *ptr += 20;
    std::cout << item.price << "\n";
}
```

الناتج: .....

6 / khatwa tech تمارين كتابة كود / 30 — 34

14 | اختبر نفسك: اكتب الحل بإيدك

اكتب برنامج كامل لكل تمرين، وخلي الناتج مطابق للـOutput المحدد.

التمرين	Input	Output المطلوب
E1 | نتيجة طالب: >=50 Passed وإلا Failed	49	Failed
E2 | اجمع الأعداد من 1 إلى n	5	15
E3 | احسب عدد الدرجات >=50 في Vector	4 ثم 50 40 90 30	2
E4 | Function ترجّع مربع عدد	6	36
E5 | زوّد سعرًا 10 باستخدام Reference	100	110
E6 | Struct لكورس: اطبع الاسم والسعر	C++ 250	C++: 250

تعليمات الحل

- اكتب #include المناسب وint main()، وعرّف الـFunctions قبل استخدامها أو أعلن عنها.
- في E3 خليك واعي لحالة Vector فاضي؛ الناتج المفروض 0.
- في E5 اكتب Function تستخدم int&، وفي E6 استخدم struct بحقلي name وprice.

> **جرّب بنفسك**
> قبل مقارنة الحل، جرّب 0 و50 و100 وحالة العناصر الفاضية، ولاحظ الفرق بين البيانات والناتج.

15 | إجابات توقع الناتج

مش بس الإجابة: ركّز على القاعدة اللي طلّعت كل نتيجة.

السؤال	الناتج الحرفي (⏎ = سطر جديد)	ليه؟
Q1	1 3 ⏎	7 / 2 قسمة صحيحة = 3، و7 % 2 باقي القسمة = 1.
Q2	Passed ⏎	الشرط >= يشمل 50، فبيطبع Passed.
Q3	1 3 5 ⏎	الأرقام الزوجية 2 و4 اتخطّاها continue، وباقي الأرقام اتطبعت.
Q4	3 4 ⏎	push_back زوّدت الحجم لـ3، و at(1) يقرأ العنصر الثاني = 4.
Q5	5 ⏎	change أخذت نسخة من n؛ الأصل فضل 5.
Q6	100 ⏎	ptr بيشير إلى item.price، والتعديل عليه غيّر العضو لـ100.

خلاصة الفروق

- int / int نتيجة قسمة صحيحة؛ % يرجع باقي القسمة.
- continue تتخطى الدورة، بينما push_back تزود حجم Vector.
- Pass by Value لا يغيّر المتغير الأصلي؛ Pointer صالح يقدر يغيّره.

15 | حلول التمارين E1–E3

أكواد كاملة ومعها ناتج تجربة محدد لكل تمرين.

E1 | ناجح أو راسب

C++20

```cpp
#include <iostream>
int main() {
    int score{};
    std::cin >> score;
    std::cout << (score >= 50 ? "Passed" : "Failed") << "\n";
}
```

الإدخال للتجربة

```
49
```

الناتج الفعلي

```
Failed
```

E2 | مجموع من 1 إلى n

C++20

```cpp
#include <iostream>
int main() {
    int n{};
    std::cin >> n;
    int total = 0;
    for (int i = 1; i <= n; ++i) total += i;
    std::cout << total << "\n";
}
```

الإدخال للتجربة

```
5
```

الناتج الفعلي

```
15
```

E3 | عدد الناجحين

C++20

```cpp
#include <iostream>
#include <vector>
int main() {
    int count{};
    std::cin >> count;
    std::vector<int> scores;
    for (int i = 0; i < count; ++i) {
        int score{};
        std::cin >> score;
        scores.push_back(score);
    }
    int passed = 0;
    for (int s : scores) if (s >= 50) ++passed;
    std::cout << passed << "\n";
}
```

الإدخال للتجربة

```
4
50 40 90 30
```

الناتج الفعلي

```
2
```

15 | حلول التمارين E4–E6

استخدم Function وReference وStruct لحل مهام صغيرة من الواقع.

E4 | مربع العدد

C++20

```cpp
#include <iostream>
int square(int n) { return n * n; }
int main() {
    int n{};
    std::cin >> n;
    std::cout << square(n) << "\n";
}
```

الإدخال للتجربة

```
6
```

الناتج الفعلي

```
36
```

E5 | تعديل السعر بالـReference

C++20

```cpp
#include <iostream>
void addTen(int& price) { price += 10; }
int main() {
    int price{};
    std::cin >> price;
    addTen(price);
    std::cout << price << "\n";
}
```

الإدخال للتجربة

```
100
```

الناتج الفعلي

```
110
```

E6 | Struct لكورس

C++20

```cpp
#include <iostream>
#include <string>
struct Course { std::string name; int price; };
int main() {
    Course course{};
    std::cin >> course.name >> course.price;
    std::cout << course.name << ": " << course.price << "\n";
}
```

الإدخال للتجربة

```
C++ 250
```

الناتج الفعلي

```
C++: 250
```

16 | Edge Cases & References

اختبر الحدود، وارجع للمواصفات الرسمية لو احتجت تفاصيل أعمق.

جدول اختبارات الحالات الحدّية

الاختبار	المتوقع	الملاحظة
E1: score = 0	Failed	أقل من 50
E1: score = 50	Passed	الحد الأدنى للنجاح
E1: score = 100	Passed	أعلى درجة صحيحة
E2: n = 0	0	Loop لا تنفّذ
E3: scores = {}	0	Vector فاضي
E4: n = 0	0	مربع الصفر
E5: price = 0	10	مرجع يعدّل الأصل
Project: count = 0	+ No students + Average: 0.00 Highest: 0	حماية من القسمة على صفر

المصادر المرجعية

C++20 Draft (N4860)	https://isocpp.org/files/papers/N4860.pdf
Working Draft C++20 (HTML)	https://timsong-cpp.github.io/cppwp/std20/
C++ Core Guidelines	https://isocpp.github.io/CppCoreGuidelines/CppCoreGuidelines
cppreference | Language	https://en.cppreference.com/w/cpp/language
cppreference | vector	https://en.cppreference.com/w/cpp/container/vector
cppreference | string	https://en.cppreference.com/w/cpp/string/basic_string
cppreference | Functions	https://en.cppreference.com/w/cpp/language/functions

ملحوظة: N4860 مسودة C++20 منشورة من لجنة التقييس، وcppreference مرجع مجتمعي موثوق لكنه مش التوثيق الرسمي للمعيار.

> **خطوتك التالية**
> نفّذ Student Grades Manager بمدخلات من المستخدم واكتب 5 Test Cases قبل ما تبدأ دراسة OOP.