KHATWA TECH

أساسيات لغة C

C PROGRAMMING FUNDAMENTALS

مذكرة مراجعة عملية من أول برنامج لحد Pointers وStructures وFile Handling، مع أكواد مجرّبة وتدريبات وحلول.

النسخة المستهدفة: C17 — ISO/IEC 9899:2018

إعداد: محمود منصور (Mahmoud Mansour)

المنصة: khatwa tech

التاريخ: 23 نوفمبر 2026

youtube.com/@khatwatech

35 PAGES / REVIEW NOTES / C17

خريطة المذكرة

كل صفحة مركّزة على فكرة واحدة، ومع كل مثال الناتج اللي ظهر من التشغيل.

طريقة المذاكرة في 3 خطوات: (1) راجع الفكرة والجدول. (2) توقّع الناتج قبل ما تبص على الحل. (3) حل التمرين، وبعدها اختبره بقيم مختلفة.

الصفحات	القسم	الموضوع
3	01	Introduction / Compile & Run
5–4	02	Variables, Types & Casting
6	03	Input / Output
8–7	04	Operators
10–9	05	Conditions
13–11	06	Loops
16–14	07	Arrays
18–17	08	Strings
21–19	09	Functions & Recursion
23–22	10	Pointers & Addresses
24	11	Structures
26–25	12	File Handling
28–27	13	المشروع التطبيقي
29	14	الأخطاء الشائعة
30	15	ورقة المراجعة
32–31	16	اختبر نفسك
34–33	17	الإجابات والحالات الحدّية
35	18	المراجع والخطوة التالية

معيار التنفيذ

كل الأمثلة هنا C17 (ISO/IEC 9899:2018)، وتُجمّع بـ GCC عبر `-std=c17`. لم نستخدم C++ أو تخصيص ذاكرة ديناميكي.

Introduction | 01

الـCompiler بيحوّل ملف C إلى برنامج تنفيذي، والبدء دايمًا من main.

الدالة `main()` نقطة البداية، و`#include <stdio.h>` بيوفّر تعريف `printf`. التعليقات لا تُنفّذ.

أول برنامج C — C17 / GCC

```c
#include <stdio.h>
int main(void) {
    printf("Hello, khatwa tech\n");
    return 0;
}
```

الناتج الفعلي

```
Hello, khatwa tech
```

Compile & Run

Terminal

```
gcc -std=c17 -Wall -Wextra -Wpedantic main.c -o app
./app
```

جزء الكود	معناه
#include <stdio.h>	تعريف دوال الطباعة والإدخال
int main(void)	بداية التنفيذ بدون Parameters
printf(...)	طباعة نص أو قيمة
return 0;	البرنامج انتهى بنجاح

انتبه

لغة C حساسة لحالة الحروف. واكتب الفاصلة المنقوطة ; بعد أغلب الـStatements.

Variables & Data Types | 02

كل Variable له Type بيحدد نوع القيمة اللي هيشيلها.

استخدم `int` للأعداد الصحيحة، `double` للعشرية، `char` لحرف واحد، و`bool` بعد تضمين `<stdbool.h>`.

بيانات كورس C17 / GCC

```c
#include <stdio.h>
#include <stdbool.h>
int main(void) {
    int students = 25;
    double price = 149.50;
    char level = 'A';
    bool active = true;
    printf("%d %.2f %c %d\n", students, price, level, active);
}
```

الناتج الفعلي

```
25 149.50 A 1
```

النوع	مثال	استخدامه
int	18	عدد صحيح
double	12.75	عشري أدق عادةً من float
float	12.5f	عشري بدقة أقل
char	'A'	حرف مفرد
bool	true / false	منطقي مع stdbool.h

الفرق اللي يهمك

C مش فيها `std::string`. النص بيتخزن غالبًا كمصفوفة `char` منتهية بـ `'\0'`.

02 | Const, Casting & sizeof

التحويل الصريح بيغير طريقة الحساب؛ sizeof بيقيس الحجم بالبايت.

القسمة الصحيحة والعشرية

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int total = 9, count = 2;
    printf("%d\n", total / count);
    printf("%.2f\n", (double)total / count);
}
```

الناتج الفعلي

```
4
4.50
```

const و sizeof

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    const int max_students = 30;
    int scores[3] = {80, 90, 70};
    size_t count = sizeof scores / sizeof scores[0];
    printf("%d\n", max_students);
    printf("%zu\n", count);
}
```

الناتج الفعلي

```
30
3
```

الصيغة	وظيفتها
const int n = 5;	متغير لا تعدّل قيمته
(double)total / count	تحويل قبل القسمة
sizeof scores	حجم الـArray بالبايت
%zu	طباعة قيمة size_t

خطأ مشهور

`double avg = total / count;` مش كفاية: القسمة حصلت كـInteger قبل التخزين.

Input & Output | 03

اطبع بـ printf واقرأ بـ scanf أو fgets حسب نوع البيانات.

إدخال درجة مع التحقق

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int score;
    if (scanf("%d", &score) == 1)
        printf("Score: %d\n", score);
    else
        puts("Invalid input");
}
```

Input المستخدم

```
85
```

الناتج الفعلي

```
Score: 85
```

قراءة اسم فيه مسافات

C17 / GCC

```c
#include <stdio.h>
#include <string.h>
int main(void) {
    char name[40];
    if (fgets(name, sizeof name, stdin) != NULL) {
        name[strcspn(name, "\n")] = '\0';
        printf("Hello, %s\n", name);
    }
}
```

Input المستخدم

```
Mona Ali
```

الناتج الفعلي

```
Hello, Mona Ali
```

Format	معناه
%d	int
%f	double مع float / *printf مع scanf
%lf	*double مع scanf
%c	حرف
%s	نص من []char
%zu	size_t

انتبه

لا تستخدم `gets` لأنها غير آمنة ومحذوفة. لازم تتحقق من ناتج `scanf` ولا تتجاوز حجم الـBuffer.

Arithmetic Operators | 04

العمليات الحسابية بتعتمد على نوع الـOperands، و% مع الأعداد الصحيحة.

حساب سعر وخصم

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int price = 200, discount = 25;
    int final_price = price - discount;
    printf("Final: %d\n", final_price);
    printf("Half: %.2f\n", final_price / 2.0);
}
```

الناتج الفعلي

```
Final: 175
Half: 87.50
```

Division / Modulo

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int a = 11, b = 4;
    printf("%d %d\n", a / b, a % b);
    printf("%.2f\n", (double)a / b);
}
```

الناتج الفعلي

```
2 3
2.75
```

العامل	المعنى
+ - *	جمع وطرح وضرب
/	قسمة صحيحة إذا الطرفان صحيحان
%	باقي قسمة أعداد صحيحة
++ / --	زيادة أو نقصان واحد

جرّب بنفسك

غيّر b إلى 3، واحسب النتيجة بنفسك قبل التشغيل.

Comparison & Logical Operators | 04

الشرط في C نتيجته صفر لو False وغير صفر لو True.

مقارنة وشروط

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int score = 78, age = 19;
    printf("%d %d\n", score >= 50, score == 100);
    printf("%d\n", score >= 50 && age >= 18);
}
```

الناتج الفعلي

```
1 0
1
```

تحديث قيمة والتقييم

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int seats = 3;
    seats += 2;
    printf("%d\n", seats);
    printf("%d\n", !(seats < 5));
}
```

الناتج الفعلي

```
5
1
```

العامل	المعنى
== / !=	يساوي / لا يساوي
> < >= <=	مقارنات
&& / || / !	And / Or / Not
= / +=	إسناد / تحديث

انتبه

المقارنة بـ ==، والإسناد بـ =. والأولوية قد تغيّر النتيجة؛ استخدم أقواسًا لتوضيح القصد.

khatwa tech / الشروط 1/2 / 09 — 35

05 | if / else

if تختار تنفيذ جزء من البرنامج حسب شرط محدد.

تقدير طالب

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int score = 82;
    if (score >= 90) puts("Excellent");
    else if (score >= 50) puts("Passed");
    else puts("Failed");
}
```

الناتج الفعلي

```
Passed
```

التحقق من نطاق الدرجة

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int score = 110;
    if (score < 0 || score > 100)
        puts("Invalid score");
    else
        puts("Valid score");
}
```

الناتج الفعلي

```
Invalid score
```

الفرع	متى ينفّذ؟
if	أول شرط صحيح
else if	الشرط السابق فشل والحالي صحيح
else	كل اللي قبل كده فشل

> **الفرق اللي يهمك**
> الترتيب مهم: اختبر الـInvalid input قبل ما تحسب Passed أو Failed.

05 | switch & Ternary

استخدم switch لقيم ثابتة و?: للاختيار البسيط بين قيمتين.

اختيار كورس

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int choice = 2;
    switch (choice) {
        case 1: puts("C course"); break;
        case 2: puts("Algorithms"); break;
        default: puts("Unknown");
    }
}
```

الناتج الفعلي

```
Algorithms
```

قرار مختصر

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int score = 49;
    puts(score >= 50 ? "Passed" : "Failed");
}
```

الناتج الفعلي

```
Failed
```

الأداة	أفضل استخدام
if / else	نطاقات وشروط مركّبة
switch	أرقام أو char أو قيم ثابتة
?:	اختيار ناتج مختصر

خطأ مشهور

لو نسيت `break` بعد case، التنفيذ ممكن ينتقل للحالة اللي بعدها (fallthrough).

for Loop | 06

مناسبة لما عدد التكرارات واضح أو عندك عداد بيتغير.

اطبع الأعداد من 1 إلى 5

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    for (int i = 1; i <= 5; ++i)
        printf("%d%s", i, i == 5 ? "\n" : " ");
}
```

الناتج الفعلي

```
1 2 3 4 5
```

مجموع 4 أعداد

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int total = 0;
    for (int i = 1; i <= 4; ++i) total += i;
    printf("Sum: %d\n", total);
}
```

الناتج الفعلي

```
Sum: 10
```

الجزء	المقصود
int i = 1	بداية العداد
i <= 5	شرط الاستمرار
++i	تحديث بعد كل دورة

> **جرّب بنفسك**
> بدّل `i <= 5` بـ `i < 5` وشوف الفرق في عدد التكرارات.

while & do while | 06

while تختبر قبل التنفيذ؛ do while تختبر بعد أول تنفيذ.

While Counter — C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int left = 3;
    while (left > 0) {
        printf("%d\n", left);
        --left;
    }
}
```

الناتج الفعلي

```
3
2
1
```

do while تنفّذ مرة — C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int attempts = 0;
    do {
        ++attempts;
        printf("Attempt %d\n", attempts);
    } while (attempts < 2);
}
```

الناتج الفعلي

```
Attempt 1
Attempt 2
```

النوع	امتى تختاره؟
for	عدد الدورات معروف
while	شرط توقف يتغيّر
do while	الجسم لازم ينفذ مرة على الأقل

خطأ مشهور

لو نسيت تحديث العداد ممكن تعمل Infinite Loop. اتأكد إن الشرط هيبقى False في وقت ما.

06 | Nested Loops, break, continue

الحلقة الداخلية بتتكرر بالكامل لكل دورة خارجية.

جدول ضرب صغير

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    for (int r = 1; r <= 2; ++r) {
        for (int c = 1; c <= 3; ++c)
            printf("%d%s", r * c, c == 3 ? "\n" : " ");
    }
}
```

الناتج الفعلي

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

تخطّي رقم ووقف الحلقة

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    for (int n = 1; n <= 6; ++n) {
        if (n == 3) continue;
        if (n == 6) break;
        printf("%d ", n);
    }
    puts("");
}
```

الناتج الفعلي

```
1 2 4 5
```

الكلمة	تعمل إيه
break	تخرج من أقرب Loop أو Switch
continue	تتخطى باقي الدورة الحالية
Nested Loops	تنظيم صفوف وأعمدة أو جداول

الفرق اللي يهمك

الـbreak لا يخرج من كل الحلقات المتداخلة؛ بيوقف أقرب حلقة فقط.

One-Dimensional Arrays | 07

الـArray بتخزن عناصر من نفس النوع في مواقع متتالية.

إنشاء ومراجعة درجات

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int scores[3] = {70, 85, 95};
    printf("%d %d\n", scores[0], scores[2]);
    scores[1] = 90;
    printf("%d\n", scores[1]);
}
```

الناتج الفعلي

```
70 95
90
```

احسب عدد العناصر

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int values[] = {10, 20, 30, 40};
    size_t n = sizeof values / sizeof values[0];
    printf("Count: %zu\n", n);
}
```

الناتج الفعلي

```
Count: 4
```

المفهوم	القاعدة
Index	يبدأ من صفر
scores[2]	العنصر الثالث
int a[3] = {...};	مصفوفة ثابتة الحجم
sizeof a / sizeof a[0]	عدد العناصر داخل نطاق Array نفسها

انتبه

القراءة أو الكتابة خارج الحدود Undefined Behavior؛ مش شرط يظهر Error.

07 | Traversal & Search

مرّ على العناصر بحلقة واحسب مجموعًا أو ابحث عن قيمة.

متوسط درجات

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int scores[] = {70, 80, 90};
    int total = 0;
    for (size_t i = 0; i < 3; ++i) total += scores[i];
    printf("Average: %.2f\n", (double)total / 3);
}
```

الناتج الفعلي

```
Average: 80.00
```

Linear Search

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int ids[] = {10, 12, 15, 18};
    int target = 15, found = 0;
    for (size_t i = 0; i < 4; ++i)
        if (ids[i] == target) found = 1;
    puts(found ? "Found" : "Not found");
}
```

الناتج الفعلي

```
Found
```

المهمة	الفكرة
Total	ابدأ من صفر واجمع كل عنصر
Average	تحقق من count > 0 قبل القسمة
Search	امشِ على العناصر وقارن بالهدف

جرّب بنفسك

غيّر target إلى 99، وجرب عدد عناصر يساوي صفر في تمرين الحلول.

07 | 2D Arrays

المصفوفة الثنائية مفيدة للدرجات على صفوف وأعمدة.

درجات طالبين في مادتين

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int grades[2][2] = {{80, 90}, {70, 75}};
    for (int row = 0; row < 2; ++row) {
        int total = 0;
        for (int col = 0; col < 2; ++col)
            total += grades[row][col];
        printf("Student %d: %d\n", row + 1, total);
    }
}
```

الناتج الفعلي

```
Student 1: 170
Student 2: 145
```

الكود	المقصود
int grades[2][2]	صفّان وعمودان
grades[0][1]	الصف الأول، العمود الثاني
for row / for col	مرور على كل الخانات

انتبه

أبعاد الـArray لازم تبقى معروفة عند تمرير مصفوفة 2D إلى Function؛ هنشوف الصيغة في قسم Functions.

08 | C Strings & Terminator

النص في C عبارة عن char[] آخره ، مش String Object.

نص وحروف

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    char course[] = "C language";
    printf("%s\n", course);
    printf("%c %c\n", course[0], course[2]);
}
```

الناتج الفعلي

```
C language
C l
```

احذف Enter من fgets

C17 / GCC

```c
#include <stdio.h>
#include <string.h>
int main(void) {
    char name[20];
    if (fgets(name, sizeof name, stdin)) {
        name[strcspn(name, "\n")] = '\0';
        printf("Name: %s\n", name);
    }
}
```

Input المستخدم

```
Salma
```

الناتج الفعلي

```
Name: Salma
```

الصيغة	معناها
'A'	char واحد
"ABC"	أربع بايتات مع في الترميز العادي
'\0'	نهاية C String
char text[20]	مساحة محدودة للنص

خطأ مشهور

مقارنة النصوص بـ == بتقارن عناوين Arrays مش محتوى النص؛ استخدم strcmp.

08 | string.h Functions

استخدم strlen للطول وstrcmp للمقارنة، وراعِ حجم الوجهة عند النسخ.

Length & Compare

C17 / GCC

```c
#include <stdio.h>
#include <string.h>
int main(void) {
    char a[] = "C17", b[] = "C17";
    printf("%zu\n", strlen(a));
    printf("%d\n", strcmp(a, b));
}
```

الناتج الفعلي

```
3
0
```

نسخ آمن بمساحة واضحة

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    char title[32];
    int n = snprintf(title, sizeof title, "%s", "C Basics");
    if (n >= 0 && (size_t)n < sizeof title)
        printf("%s\n", title);
}
```

الناتج الفعلي

```
C Basics
```

الدالة	وظيفتها
strlen(s)	عدد حروف النص بدون
strcmp(a,b)	صفر لو النصين متساويين
strchr(s,c)	مكان أول حرف مطابق أو NULL
snprintf(buf,size,...)	تكوين نص مع حد لحجم الـBuffer

انتبه

strcpy وstrcat موجودتان في C17 لكن ممكن تسببان Overflow لو الوجهة صغيرة. افحص الأحجام، وفضّل الصيغ محدودة الحجم عند اللزوم.

09 | Functions: Declaration & Return

قسّم البرنامج إلى وظائف صغيرة كل واحدة لها Input وOutput واضحين.

Function لحساب مربع

C17 / GCC

```c
#include <stdio.h>
int square(int n) { return n * n; }
int main(void) {
    printf("%d\n", square(6));
}
```

الناتج الفعلي

```
36
```

Prototype قبل main

C17 / GCC

```c
#include <stdio.h>
int add(int a, int b);
int main(void) {
    printf("%d\n", add(3, 4));
}
int add(int a, int b) { return a + b; }
```

الناتج الفعلي

```
7
```

الجزء	المقصود
Return Type	نوع الناتج، أو void
Parameter	مدخل الدالة وقت تعريفها
Argument	القيمة المُمرّرة عند الاستدعاء
Prototype	إعلان يسبق الاستخدام

> **الفرق اللي يهمك**
> الدالة ممكن تتنفذ كذا مرة، و`return` ينهي تنفيذ الدالة ويرجع قيمة إن كان نوعها غير void.

09 | Parameters & Scope

C بتمرر القيم للـFunction بنسخها؛ تغيير الأصل يحتاج Pointer.

Pass by Value

C17 / GCC

```c
#include <stdio.h>
void change(int x) { x = 100; printf("Inside: %d\n", x); }
int main(void) {
    int score = 50;
    change(score);
    printf("Outside: %d\n", score);
}
```

الناتج الفعلي

```
Inside: 100
Outside: 50
```

Local Scope

C17 / GCC

```c
#include <stdio.h>
int bonus(void) { int points = 5; return points; }
int main(void) {
    int points = 10;
    printf("%d %d\n", points, bonus());
}
```

الناتج الفعلي

```
10 5
```

المفهوم	الأثر
Local Variable	مُعرّف داخل Block
Global Variable	متاح بنطاق الملف من نقطة التعريف
Pass by Value	النسخة تتغير، الأصل لا
Pointer Parameter	تقدر تغيّر الأصل بشكل صريح

انتبه

لا ترجّع عنوان متغير Local انتهى عمره؛ الـPointer الناتج هيبقى غير صالح.

09 | Recursion & Array Parameters

Recursion دالة بتنادي نفسها بشرط توقف، والـArray بتحتاج طولها كـParameter.

Factorial مع حالة أساس

C17 / GCC

```c
#include <stdio.h>
int factorial(int n) {
    if (n <= 1) return 1;
    return n * factorial(n - 1);
}
int main(void) { printf("%d\n", factorial(5)); }
```

الناتج الفعلي

```
120
```

مجموع عناصر Array

C17 / GCC

```c
#include <stdio.h>
int sum(const int values[], size_t n) {
    int total = 0;
    for (size_t i = 0; i < n; ++i) total += values[i];
    return total;
}
int main(void) {
    int a[] = {10, 20, 30};
    printf("%d\n", sum(a, 3));
}
```

الناتج الفعلي

```
60
```

انتبه

في تعريف Function، `int values[]` تتحول عمليًا إلى Pointer، فـ `sizeof values` مش بيدي حجم الـArray الأصلية.

10 | Pointers: Address & Dereference

Pointer بيحفظ عنوان قيمة، و* بتقرأ أو تعدّل القيمة عند العنوان.

عنوان وقيمة

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int score = 80;
    int *ptr = &score;
    printf("%d\n", *ptr);
    *ptr = 95;
    printf("%d\n", score);
}
```

الناتج الفعلي

```
80
95
```

فحص Pointer قبل الاستخدام

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int *ptr = NULL;
    if (ptr == NULL) puts("No address");
}
```

الناتج الفعلي

```
No address
```

الصيغة	المقصود
int *p = &x;	خزّن عنوان x
*p	القيمة الموجودة عند العنوان
&x	عنوان x
NULL	لا يشير إلى كائن صالح

انتبه

لا تعمل Dereference لـ `NULL` أو Pointer غير مهيأ. طباعة العنوان بـ %p تحتاج cast إلى (void*)، والعنوان نفسه يتغير كل تشغيل.

10 | Pointer Parameters & Arrays

استخدم Pointer Parameter لو الـFunction محتاجة تعدّل الأصل.

تعديل السعر بالدالة — C17 / GCC

```c
#include <stdio.h>
void add_discount(int *price) { *price -= 10; }
int main(void) {
    int price = 100;
    add_discount(&price);
    printf("Price: %d\n", price);
}
```

الناتج الفعلي

```
Price: 90
```

Array مع Pointer — C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int scores[] = {60, 70, 80};
    int *p = scores;
    printf("%d %d\n", *p, *(p + 2));
}
```

الناتج الفعلي

```
60 80
```

المفهوم	الفرق
Pass by Value	الدالة تحصل على نسخة
Pass by Pointer	الدالة تحصل على عنوان للعنصر الأصلي
p + 1	ينتقل إلى العنصر التالي حسب نوع p

الفرق اللي يهمك

C مفيهاش Reference Parameters زي C++؛ التعديل على المتغير الأصلي بيتم بتمرير عنوانه.

11 | Structures

Struct بتجمع حقول مختلفة تحت اسم نوع واحد.

Struct لطالب

C17 / GCC

```c
#include <stdio.h>
typedef struct {
    char name[24];
    int score;
} Student;
int main(void) {
    Student s = {"Mona", 92};
    printf("%s: %d\n", s.name, s.score);
}
```

الناتج الفعلي

```
Mona: 92
```

الوصول للأعضاء عبر Pointer

C17 / GCC

```c
#include <stdio.h>
struct Course { int id; double price; };
int main(void) {
    struct Course course = {7, 250.0};
    struct Course *p = &course;
    printf("%d %.0f\n", p->id, p->price);
}
```

الناتج الفعلي

```
7 250
```

الطريقة	المعنى
item.field	عضو من متغير Struct
ptr->field	عضو عن طريق Pointer
typedef struct { ... } Name;	اسم مختصر لنوع Struct

انتبه

Struct تجميع بيانات، ومش محتاج Classes أو OOP. تقدر تعمل Array من Struct وتبعتها إلى Function.

12 | File Handling: Write

الـFILE* بيمثل ملف مفتوح؛ لازم تتأكد إن fopen نجحت وتقفله.

اكتب تقريرًا نصيًا

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    FILE *file = fopen("course.txt", "w");
    if (file == NULL) { puts("Open failed"); return 1; }
    fprintf(file, "Course: C17\nStudents: 3\n");
    if (fclose(file) != 0) { puts("Close failed"); return 1; }
    puts("Saved course.txt");
}
```

الناتج الفعلي

```
Saved course.txt
```

Mode	التأثير
"r"	فتح للقراءة، لازم الملف موجود
"w"	كتابة، مع إنشاء/تفريغ الملف
"a"	إضافة في نهاية الملف
"rb" / "wb"	فتح Binary للقراءة / الكتابة

انتبه

استخدام "w" يمسح المحتوى القديم لو الملف موجود. تحقق من fopen وfclose ولا تعتبر نجاح الطباعة على الشاشة دليلًا على نجاح الحفظ.

khatwa tech / الملفات 2/2 / 26 — 35

12 | File Handling: Read

اقرأ من الملف بـ fgets أو fscanf مع التحقق من كل نتيجة.

أنشئ ملف واقرأ سطره

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    FILE *out = fopen("lesson.txt", "w");
    if (!out) return 1;
    fputs("C Programming\n", out);
    if (fclose(out) != 0) return 1;
    FILE *in = fopen("lesson.txt", "r");
    if (!in) return 1;
    char line[40];
    if (fgets(line, sizeof line, in)) printf("%s", line);
    if (fclose(in) != 0) return 1;
}
```

الناتج الفعلي

```
C Programming
```

قراءة رقم من ملف

C17 / GCC

```c
#include <stdio.h>
int main(void) {
    FILE *file = tmpfile();
    if (!file) return 1;
    fprintf(file, "%d\n", 75);
    rewind(file);
    int score;
    if (fscanf(file, "%d", &score) == 1)
        printf("Score: %d\n", score);
    fclose(file);
}
```

الناتج الفعلي

```
Score: 75
```

الدالة	الاستعمال
fgets	قراءة سطر ضمن حجم Buffer
fprintf / fscanf	طباعة أو قراءة من File
fclose	إغلاق الملف
tmpfile	ملف مؤقت، قد لا يتوفر بكل البيئات

الفرق اللي يهمك

في File I/O برضه لازم تفحص فشل فتح الملف وفشل القراءة؛ البيانات الخارجية غير مضمونة.

13 | المشروع: Student Grades Manager

برنامج كامل يدمج Array وStruct وPointer وFunctions وFile Handling.

البرنامج يقبل من صفر إلى 5 طلاب. كل اسم كلمة واحدة، والدرجة من 0 لـ100. بيحسب المتوسط وأعلى وأقل درجة ويكتب المتوسط في grades.txt.

الكود الكامل للمشروع — C17 / GCC

```c
#include <stdio.h>
#define MAX_STUDENTS 5

typedef struct {
    char name[32];
    int score;
} Student;

const char *result(int score) {
    return score >= 50 ? "Passed" : "Failed";
}

int main(void) {
    Student students[MAX_STUDENTS];
    int count;
    if (scanf("%d", &count) != 1 ||
        count < 0 || count > MAX_STUDENTS) {
        puts("Invalid count");
        return 1;
    }
    if (count == 0) {
        puts("No students");
        return 0;
    }

    int total = 0, high = 0, low = 0;
    for (int i = 0; i < count; ++i) {
        Student *s = &students[i];
        if (scanf("%31s %d", s->name, &s->score) != 2 ||
            s->score < 0 || s->score > 100) {
            puts("Invalid student");
            return 1;
        }
        total += s->score;
        if (s->score > students[high].score) high = i;
        if (s->score < students[low].score) low = i;
    }
    for (int i = 0; i < count; ++i)
        printf("%s: %d | %s\n", students[i].name,
               students[i].score, result(students[i].score));

    double average = (double)total / count;
    printf("Average: %.2f\n", average);
    printf("Highest: %s (%d)\n",
           students[high].name, students[high].score);
    printf("Lowest: %s (%d)\n",
           students[low].name, students[low].score);

    FILE *file = fopen("grades.txt", "w");
    if (file == NULL) { puts("Save failed"); return 1; }
    fprintf(file, "Average: %.2f\n", average);
    if (fclose(file) != 0) { puts("Save failed"); return 1; }
    puts("Saved: grades.txt");
    return 0;
}
```

Input المستخدم

```
3
Mona 90
Ali 75
Karim 49
```

13 | المشروع: النتيجة والتفسير

دي نفس تجربة الإدخال اللي اتشغلت فعليًا بالكود الكامل في الصفحة اللي قبلها.

التجربة: Input

```
3
Mona 90
Ali 75
Karim 49
```

الناتج الفعلي

```
Mona: 90 | Passed
Ali: 75 | Passed
Karim: 49 | Failed
Average: 71.33
Highest: Mona (90)
Lowest: Karim (49)
Saved: grades.txt
```

محتوى الملف grades.txt بعد التشغيل: `Average: 71.33`

السطر	التفسير
Mona: 90 | Passed	الدرجة 90، أكبر أو تساوي 50
Karim: 49 | Failed	الدرجة أقل من 50
Average: 71.33	214 ÷ 3، عرض بمنزلتين
Highest / Lowest	90 هي الأكبر و49 هي الأصغر
Saved: grades.txt	حفظ ملف نصي يحتوي Average: 71.33

طوّر المشروع بنفسك

- اعمل Menu فيها عرض كل الطلاب أو البحث بالاسم.
- خلي قراءة الاسم تقبل مسافات باستخدام fgets مع معالجة نهاية السطر.
- احفظ قائمة الطلاب كاملة في ملف، واقرأها لما البرنامج يشتغل تاني.

> **جرّب بنفسك**
> اكتب Input يساوي 0 طالب: المفروض البرنامج يطبع No students فقط، من غير قسمة على صفر.

14 | الأخطاء الشائعة

اسم رسالة الخطأ ممكن يختلف حسب Compiler، فراجع السبب قبل ما تدور على حل.

رسالة / عَرَض	السبب المحتمل	تعمل إيه
expected ';' before ...	ناقص ; أو قوس	راجع السطر اللي قبله
undeclared identifier / 'x' undeclared	اسم غير معرّف أو خارج Scope	عرّف المتغير وافحص الاسم
implicit declaration of function	Header ناقص أو Prototype غير موجود	أضف include أو إعلان الدالة
undefined reference to ...	تعريف الدالة مفقود أو الربط ناقص	وفّر التعريف واربط الملفات
format ... expects ...	Format Specifier لا يطابق نوع Argument	راجع %d و%f و%lf و%zu
Segmentation fault (ممكن)	Pointer غير صالح أو Index خارج الحدود	تحقق من الحدود والعنوان
برنامج واقف عند scanf	منتظر Input أو حصل فشل قراءة	افحص Return Value
division by zero	قسمة صحيحة على صفر	تحقق من المقام
ناتج Average صحيح فقط	قَسَم int على int قبل التحويل	حوّل إلى double أولًا
File not found / fopen == NULL	مسار خطأ أو صلاحيات ناقصة	راجع المسار وMode وفشل الفتح

انتبه

الـUndefined Behavior مش لازم ينتج رسالة واضحة. Sanitizers مفيدة وقت التطوير، لكن الأمثلة هنا تتجنب كود غير معرّف.

15 | ورقة المراجعة السريعة

الصيغ اللي غالبًا هتحتاجها وقت حل مسائل C.

عايز أعمل كذا	الصيغة
أطبع عدد	printf("%d\n", n);
أطبع عشري	printf("%.2f\n", x);
أقرأ int	scanf("%d", &n) == 1
أقرأ سطر	fgets(text, sizeof text, stdin)
ثابت	const int limit = 10;
تحويل لعشري	(double)total / count
مقارنة	a == b / a != b
شرط	if (score >= 50) { ... }
اختيار سريع	ok ? "Yes" : "No"
Switch	switch (x) { case 1: ... break; }
Loop بعدد	for (int i=0; i<n; ++i) { ... }
Loop بشرط	while (condition) { ... }
Array	int a[3] = {1, 2, 3};
عدد عناصر Array	sizeof a / sizeof a[0]
2D Array	int a[2][3] = {{1,2,3},{4,5,6}};
C String	char name[20] = "Ali";
طول نص	strlen(name)
مقارنة نصين	strcmp(a, b) == 0
Function	int add(int a,int b) {return a+b;}
Pointer	int *p = &value;
تعديل عبر Pointer	*p = 42;
Struct	struct Item { int price; };
فتح ملف للكتابة	FILE *f = fopen("out.txt", "w");
طباعة في ملف	fprintf(f, "%d\n", n);
إغلاق ملف	fclose(f);

khatwa tech / الاختبار 1/2 / 31 — 35

16 | اختبر نفسك: توقّع الناتج

ما تقلبش على الإجابات قبل ما تكتب Output كل برنامج، سطر بسطر.

Q1 القسمة هنا Integer ولا Decimal؟

```c
#include <stdio.h>
int main(void) {
    int a = 7, b = 2;
    printf("%d %d\n", a / b, a % b);
}
```

اكتب الناتج هنا: .....

Q2 هل 50 داخلة في شرط النجاح؟

```c
#include <stdio.h>
int main(void) {
    int score = 50;
    puts(score >= 50 ? "Passed" : "Failed");
}
```

اكتب الناتج هنا: .....

Q3 إيه الرقم اللي اتخطّى؟

```c
#include <stdio.h>
int main(void) {
    for (int i = 1; i <= 5; ++i) {
        if (i == 3) continue;
        printf("%d ", i);
    }
    puts("");
}
```

اكتب الناتج هنا: .....

Q4 Index يبدأ من كام؟

```c
#include <stdio.h>
int main(void) {
    int a[] = {4, 8, 12};
    printf("%d %d\n", a[0], a[2]);
}
```

اكتب الناتج هنا: .....

Q5 هل الدالة غيّرت المتغير الأصلي؟

```c
#include <stdio.h>
void add(int x) { x += 10; }
int main(void) {
    int price = 20;
    add(price);
    printf("%d\n", price);
}
```

اكتب الناتج هنا: .....

Q6 ماذا حدث للقيمة عبر Pointer؟

```c
#include <stdio.h>
int main(void) {
    int x = 5;
    int *p = &x;
    *p += 3;
    printf("%d\n", x);
}
```

اكتب الناتج هنا: .....

16 | اختبر نفسك: اكتب الحل

لكل تمرين اكتب برنامج كامل من Input إلى Output، وجرب حالة حدّية.

التمرين	Input	Output الحرفي
E1 | صنّف الدرجة: 0..100 وإلا Invalid	49	Failed
E2 | اجمع الأعداد من 1 إلى n، وn غير سالب	5	15
E3 | عدد الناجحين من n درجات، وn بين 0 و20	4 ثم 50 40 90 30	2
E4 | Function تزود السعر 10 عبر Pointer	100	110
E5 | Struct باسم كورس وسعره	C17 250	C17: 250
E6 | احفظ رقمًا في File واقرأه	95	Saved: 95

الشروط

- E1: اطبع Invalid لو الدرجة خارج 0 إلى 100، وبعدها Passed أو Failed.
- E2: لو n يساوي صفر، الناتج صفر. استخدم Loop واضحة.
- E3: عدد عناصر صفر لازم يطبع 0؛ اتأكد من صحة n.
- E4: المطلوب Pointer وليس C++ Reference.
- E5: الاسم كلمة واحدة، وE6 لازم يفحص نجاح File I/O.

> **جرّب بنفسك**
> بعد الحل اختبر E1 بالدرجات -1 و0 و50 و100 و101؛ واختبر E3 بقائمة فاضية.

17 | الإجابات: توقع الناتج + E1–E3

كل كود في الحلول اتشغّل بمدخلات التمرين، والنتيجة تحته.

السؤال	الناتج (⏎ سطر جديد)	السبب
Q1	1 3 ⏎	قسمة int على int تدي 3، وباقي 7 % 2 يساوي 1.
Q2	Passed ⏎	>= شامل الحد 50، فالناتج Passed.
Q3	5 4 2 1 ⏎	continue تتخطى العدد 3؛ باقي الأعداد تتطبع.
Q4	12 4 ⏎	فهرس Array يبدأ بـ0؛ الأول 4 والثالث 12.
Q5	20 ⏎	الدالة أخذت نسخة من price، فالأصل فضل 20.
Q6	8 ⏎	p بيشاور على x، و*3 =+ p عدّل x إلى 8.

حلول التمارين E1–E3

E1 | درجة طالب — C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int score;
    if (scanf("%d", &score) != 1) return 1;
    if (score < 0 || score > 100) puts("Invalid");
    else puts(score >= 50 ? "Passed" : "Failed");
}
```

Input المستخدم

```
49
```

الناتج الفعلي

```
Failed
```

E2 | مجموع من 1 إلى n — C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int n;
    if (scanf("%d", &n) != 1 || n < 0) return 1;
    int sum = 0;
    for (int i = 1; i <= n; ++i) sum += i;
    printf("%d\n", sum);
}
```

Input المستخدم

```
5
```

الناتج الفعلي

```
15
```

E3 | عدد الناجحين — C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int n, score, passed = 0;
    if (scanf("%d", &n) != 1 || n < 0 || n > 20) return 1;
    for (int i = 0; i < n; ++i) {
        if (scanf("%d", &score) != 1) return 1;
        if (score >= 50) ++passed;
    }
    printf("%d\n", passed);
}
```

Input المستخدم

```
4
50 40 90 30
```

الناتج الفعلي

```
2
```

17 | الإجابات: E4–E6 وEdge Cases

جرّب الأقل والأعلى، والصفر، وحالة عدم وجود عناصر قبل اعتبار البرنامج صحيحًا.

E4 | تحديث عبر Pointer — C17 / GCC

```c
#include <stdio.h>
void add_ten(int *price) { *price += 10; }
int main(void) {
    int price;
    if (scanf("%d", &price) != 1) return 1;
    add_ten(&price);
    printf("%d\n", price);
}
```

Input المستخدم

```
100
```

الناتج الفعلي

```
110
```

Struct | E5 لكورس — C17 / GCC

```c
#include <stdio.h>
typedef struct { char name[24]; int price; } Course;
int main(void) {
    Course course;
    if (scanf("%23s %d", course.name, &course.price) != 2) return 1;
    printf("%s: %d\n", course.name, course.price);
}
```

Input المستخدم

```
C17 250
```

الناتج الفعلي

```
C17: 250
```

E6 | احفظ واقرأ رقمًا — C17 / GCC

```c
#include <stdio.h>
int main(void) {
    int score, loaded;
    if (scanf("%d", &score) != 1) return 1;
    FILE *f = tmpfile();
    if (!f) return 1;
    if (fprintf(f, "%d\n", score) < 0) { fclose(f); return 1; }
    rewind(f);
    if (fscanf(f, "%d", &loaded) != 1) { fclose(f); return 1; }
    if (fclose(f) != 0) return 1;
    printf("Saved: %d\n", loaded);
}
```

Input المستخدم

```
95
```

الناتج الفعلي

```
Saved: 95
```

اختبارات حالات حدّية (متوقَّع وتم اختباره)

حالة الإدخال	الناتج الحقيقي	السبب
E1: -1	Invalid	قيمة أقل من المجال
E1: 0	Failed	أقل درجة
E1: 50	Passed	حد النجاح
E1: 100	Passed	أعلى درجة
E2: 0	0	حلقة لا تنفّذ
E3: n = 0	0	قائمة درجات فاضية
E4: 0	10	Pointer عدّل الأصل
Project: count = 0	No students	تجنّب قسمة على صفر

18 | المراجع وخطوتك التالية

المرجع الأساسي هو معيار لغة C؛ الروابط التكميلية تساعدك تلاقي الشرح بسرعة.

المعيار والتوثيق الرسمي

ISO/IEC 9899:2018 | C17	https://www.iso.org/standard/74528.html
WG14 | C Working Group	https://www.open-std.org/jtc1/sc22/wg14/
GCC | Standards / C17	https://gcc.gnu.org/onlinedocs/gcc/Standards.html
GCC | Dialect Options	https://gcc.gnu.org/onlinedocs/gcc/C-Dialect-Options.html

دليل مرجعي حسب الموضوع (cppreference - مرجع مجتمعي، مش وثيقة ISO الرسمية)

Types & Variables	https://en.cppreference.com/w/c/language/type
Operators & Conditions	https://en.cppreference.com/w/c/language/operator_precedence
Arrays & Pointers	https://en.cppreference.com/w/c/language/array
Functions & Recursion	https://en.cppreference.com/w/c/language/function_definition
C Strings	https://en.cppreference.com/w/c/string/byte
Structures	https://en.cppreference.com/w/c/language/struct
Input / Output & Files	https://en.cppreference.com/w/c/io

> **الخطوة العملية بعد المذكرة**
> اكتب Student Grades Manager من الذاكرة، وبعدها جرّبه بالـInputs: صفر طلاب، طالب واحد، خمس طلاب، ودرجة غير صالحة. ثم جرّب تغيّر حفظ التقرير إلى CSV.

بيئة التحقق: GCC مع `-std=c17 -Wall -Wextra -Wpedantic -Werror`. الأمثلة التي تعتمد على الملفات تتشغّل داخل مجلد قابل للكتابة. النسخة C17، مش C23.