

الفصل 1 : أوامر الطباعة والمتغيرات بلغة (C#)

شريحة 1: شريحة العنوان

أوامر الطباعة والمتغيرات

أساسيات علوم الحاسوب

شريحة 2: ماذَا سنتعلم اليوم؟

- **الطباعة على الشاشة:** كيف نجعل البرنامج يعرض معلومات.
- **المتغيرات:** ما هي وكيف نستخدمها لتخزين المعلومات.
- **استقبال المدخلات:** كيف نجعل البرنامج يتفاعل مع المستخدم.
- **ذاكرة الحاسوب:** أين يتم تخزين هذه المعلومات مؤقتاً.

سؤال للنقاش:

"كيف تتعرف عليك الألعاب وتتاديك باسمك؟"

شكراً على رفع الملف. كما طلبت، سأقوم بإضافة شريحة جديدة بين شريحة 2 وشريحة 3، لشرح:

"ما هو البرنامج في الحاسوب؟"

الشريحة الجديدة التي سأضيفها بعنوان:

شريحة 3: ما هو البرنامج في الحاسوب؟

النص المقترن للشريحة (بلغة تعليمية سهلة وواضحة):

شريحة 3: ما هو البرنامج في الحاسوب؟

البرنامج هو مجموعة من التعليمات التي نكتبها لكي ينفذها الحاسوب خطوة بخطوة.

فكر فيه كالوصفة!

مثل وصفة طبخ فيها خطوات مرتبة: اجلب، اخلط، اطبخ.
في البرمجة: اطبع، اقرأ، احسب، احفظ...

 البرنامج يخبر الحاسوب بما يجب أن يفعله.
بدونه، الحاسوب لا "يعرف" شيئاً.

 البرنامج يُخزن كملف) مثل .exe أو (.cs
ويتم تشغيله بواسطة المعالج.(CPU)

أمثلة على البرامج:

- الآلة الحاسبة: برنامج يقوم بالحساب.
- تطبيق: WhatsApp برنامج لإرسال رسائل.
- اللعبة التي تلعبها: هي أيضاً برنامج!

شريحة 4: الهيكل الأساسي للبرنامج

كل برنامج C# يبدأ بهذا الهيكل. فكر فيه كـ "الحاوية" التي نضع بداخلها أوامرنا.

```
class Program
{
    static void Main(string[] args)
    {
        // أوامرنا تكتب هنا
    }
}
```

• **Main**: هي نقطة البداية. الحاسوب يبدأ بقراءة وتنفيذ الأوامر من هنا.

شريحة 5: أوامر الطباعة

هناك طريقتان رئيسيتان لطباعة النصوص على الشاشة:

الأمر	الشرح	مثال الكود والناتج
Console.WriteLine()	يطبع النص ويبقى المؤشر في نفس السطر.	Console.WriteLine("مرحباً"); Console.WriteLine("يا عالم"); الناتج: مرحباً يا عالم!
Console.WriteLine()	يطبع النص ثم ينزل إلى سطر جديد.	Console.WriteLine("مرحباً"); Console.WriteLine("يا عالم"); مرحباً يا عالم!

شريحة 5: ما هو المتغير؟

ببساطة، تخيل المتغير كصندوق تخزين .

• له اسم (ملحق): لتعرف ماذا بداخله (مثال: `studentName`).

• له نوع: يحدد ما الذي يمكن وضعه بداخله (نصوص، أرقام، ... إلخ).

• يحتوي على قيمة: المعلومة الفعلية التي نخزنها (مثال: "أحمد").

نحن نستخدم المتغيرات لحفظ المعلومات التي قد تتغير أو التي تحتاجها لاحقاً في البرنامج.

شريحة 6: أنواع المتغيرات (أنواع الصناديق)

في البداية، سنركز على نوعين أساسيين:

• **string** (تص): لتخزين سلسلة من الحروف والرموز (الاسم، جملة، عنوان).

◦ مثال: `string message = "أهلاً بالجميع";`

• **int** (عدد صحيح): لتخزين أعداد صحيحة (بدون كسورة عشرية) مثل العمر أو عدد النقاط.

◦ مثال: `int age = 15;`

شريحة 7: التفاعل مع المستخدم

لجعل البرنامج تفاعلياً، نجمع بين تعريف المتغير وأمر القراءة.

الخطوة 1: تعريف متغير (صندوق) لتخزين اسم المستخدم //

```
string userName;
```

الخطوة 2: طباعة سؤال للمستخدم //

```
Console.WriteLine("ما هو اسمك؟");
```

الخطوة 3: قراءة إجابة المستخدم وتخزينها في المتغير //

```
userName = Console.ReadLine();
```

الخطوة 4: استخدام القيمة المخزنة //

```
Console.WriteLine("أهلاً بك يا " + userName);
```

• **Console.ReadLine()**: يجعل البرنامج يتوقف وينتظر المستخدم ليكتب شيئاً ويضغط .Enter

شريحة 8: أين تحفظ المتغيرات؟

عندما تشغّل برنامجك، يقوم نظام التشغيل بتخصيص مساحة مؤقتة له في ذاكرة الوصول العشوائي .(RAM)

- فكر فيها كـ "رف" خاص ببرنامجك.
 - يتم وضع كل "صناديقك" (متغيراتك) على هذا الرف لاستخدامها.
 - عندما يغلق البرنامج، يتم تنظيف الرف بالكامل وتختفي كل البيانات.
-

شريحة 9: تحدي التطبيق

المهمة: اكتب برنامجاً كاملاً يقوم بالخطوات التالية:

1. يسأل المستخدم عن اسمه ويخزنه في متغير.
2. يسأل المستخدم عن عمره ويخزنه في متغير.
3. يطبع رسالة ترحيبية مخصصة تجمع الاسمين، مثل: "مرحباً يا خالد! عمرك 16 عاماً. سعيد بلقائك!"

تلميح: يرجع نصاً ()Console.ReadLine(string). ستحتاج إلى طريقة لتحويل العمر من نص إلى رقم ()Convert.ToInt32(int)! ابحث عن

شريحة 10: تخزين المتغيرات - مثال عملي

لنرى كيف تبدو عملية تخزين المتغيرات في ذاكرة الحاسوب.

```
class Program
{
    static void Main(string[] args)
    {
        // 1. تعريف متغير (صندوق) اسمه x
        string x;

        // 2. قراءة القيمة من المستخدم ووضعها في الصندوق
        x = Console.ReadLine();
    }
}
```

عندما يقوم المستخدم بكتابة كلمة "hey" والضغط على :Enter

- في البرنامج: يتم أخذ القيمة "hey" وتخزينها في المتغير x.
- في الذاكرة: يتم حجز مساحة للمتغير x وتوضع القيمة "hey" بداخلها.

شريحة 11 أنواع المتغيرات:

- نصوص **string**:
- عدد صحيح **int**:
- عدد عشري **double**:
- حرف واحد **char**:
- صح/خطأ **bool**:

أمثلة	الحجم في الذاكرة	نوع البيانات	اسم النوع
"hi"	غير محدد	نص	string
2, 7	4بايت	عدد صحيح	int
6.8	8بايت	عدد عشري	double
'A'	2بايت	حرف	char
true	1بايت	منطقي	bool

شريحة 12: طباعة قيمة المتغير

الآن بعد أن خرّنا القيمة، يمكننا استخدامها وطباعتها على الشاشة.

```
class Program
{
    static void Main(string[] args)
    {
        string x;
        Console.WriteLine("Enter a word:");
        x = Console.ReadLine();
    }
}
```

نستخدم علامة + لدمج النصوص مع قيمة المتغير //

```
Console.WriteLine("The word is " + x);  
}  
}
```

النتيجة عند التشغيل:

Enter a word:

hey

The word is hey

- لاحظ كيف أن البرنامج استخدم القيمة "hey" التي كانت مخزنة في الصندوق x لعرضها في الجملة الأخيرة.

شريحة 13: مصطلحات أساسية: الأمر والدالة

لنتعلم الفرق بين مصطلحين مهمين:

- الأمر (Instruction)

- هو كلمة محفوظة يفهمها الحاسوب لتنفيذ مهمة واحدة ومحددة.

- أمثلة نعرفها:

- أمر الكتابة على الشاشة : Console.WriteLine.
- أمر القراءة من المستخدم : Console.ReadLine.

- الدالة (Function/Method)

- هي مجموعة من الأوامر معًا لتنفيذ مهمة أكبر وأكثر تعقيداً.

- مثال نعرفه:

- Main: هي الدالة الرئيسية التي تحتوي على كل أوامر برنامجنا.

شريحة 14: قواعد تسمية المتغيرات

عندما تختار اسمًا "لصندوقك" (المتغير)، هناك بعض القواعد المهمة:

- يمكن أن تحتوي الأسماء على حروف إنجليزية (كبيرة وصغيرة)، أرقام، والرمز _ .
- لا يمكن أن يبدأ اسم المتغير برقم. (✓ student1 ✗ student1 لكن *).
- لا يمكن أن يحتوي الاسم على مسافات أو رموز خاصة (مثل *، -، +).
- لا يمكن استخدام الكلمات المحجوزة في اللغة كأسماء للمتغيرات (مثل int, string, class).

: نصيحة احترافية (Best Practice) ★

لجعل الكود سهل القراءة، استخدم طريقة CamelCase. تبدأ الكلمة الأولى بحرف كبير ، وكل كلمة تالية تبدأ بحرف صغير.

مثال: StudentFirstName , NumOfStudents .

شريحة 15: التمهيد (Assignment) والإسناد (Initialization)

- التعريف (Declaration): هو إنشاء صندوق فارغ. int x;
- الإسناد (Assignment): هو وضع قيمة داخل الصندوق. x = 5;
- التمهيد (Initialization): هو إنشاء الصندوق ووضع قيمة بداخله في نفس الخطوة.

int y = 15

⚠ خطأ شائع:

لا يمكنك استخدام متغير (القراءة منه) قبل وضع قيمة بداخله. سيؤدي هذا إلى خطأ برمجي لأنك تحاول استخدام "صندوق فارغ"!

تعريف صندوق فارغ //

// هذا السطر سيعطي خطأ لأن x لا تحتوي على قيمة بعد!

Console.WriteLine(x);

شريحة 16: خطأ في تحويل الأنواع

ماذا يحدث لو طلبنا من المستخدم إدخال رقم، لكنه أدخل نصاً بدلاً من ذلك؟

```
int num;  
Console.WriteLine("Enter a number: ");  
num = int.Parse(Console.ReadLine());  
Console.WriteLine("The number is " + num);
```

إذا أدخل المستخدم "82": يعمل البرنامج بشكل سليم.

إذا أدخل المستخدم "7ع": سيتوقف البرنامج ويعطي خطأ!

السبب: الحاسوب لا يعرف كيف يحول الحرف `u` إلى جزء من رقم. هذا الخطأ يسمى `.FormatException`.

شريحة 17: كيف يفكر الحاسوب؟ (النظام الثنائي)

نحن البشر نستخدم النظام العشري (10 أرقام: 0-9)، ربما لأن لدينا 10 أصابع.

الحاسوب لا يملك أصابع! لكنه يملك مكونات إلكترونية لها حالتان فقط:

- **ON (تشغيل):** يمثلها بالرقم 1.
- **OFF (إيقاف):** يمثلها بالرقم 0.

هذا هو **النظام الثنائي (Binary)**. كل المعلومات في الحاسوب، من النصوص والصور إلى الألعاب، يتم تخزينها في النهاية على شكل سلسلة طويلة جداً من الأصفار والاحاد.

بالتأكيد، لستكمel معالجة الشرائح الجديدة بنفس المنهجية البسيطة والتفاعلية.

شريحة 17: أهمية تمييد المتغيرات (Initialization)

في البرمجة، لا يمكنك استخدام متغير قبل أن تضع قيمة بداخله. فكر في الأمر كـ"محاولة الشرب من كوب فارغ"، فهذا لا يعقل!

• **الحالة الصحيحة (باستخدام `int`):**

1. `int x, y, z;` ; (تعريف 3 صناديق فارغة)

2. $z = 9$; (وضع قيمة في الصندوق z)
3. $x = 5$; (وضع قيمة في الصندوق x)
4. $y = x * 3$; (استخدام قيمة x لحساب قيمة y)

• **الحالة الصحيحة (باستخدام string):**

1. `string str;` (تعريف صندوق نصي فارغ)
2. `str = "hello";` (وضع قيمة في الصندوق)
3. `Console.WriteLine(str);` (استخدام الصندوق للطباعة)

القاعدة الذهبية: دائمًا ضع قيمة أولية في متغيراتك قبل محاولة استخدامها لتجنب الأخطاء. 

شريحة 18: تحويل أنواع البيانات (Type Casting)

أحياناً تحتاج لتحويل قيمة من نوع إلى آخر. تخيل أنك تسكب سائلاً من وعاء لآخر.

الشرح والنتيجة	مثال الكود	التحويل
يتم "قص" الجزء العشري. النتيجة: 97 (فقدان بيانات).	<code>int n1 = (int)97.2;</code>	double من int إلى
يتم التحويل تلقائياً بدون مشاكل. النتيجة: .97.0 .	<code>double d = 97;</code>	من int إلى double
نستخدم الأداة Parse لتحويل النص إلى رقم. النتيجة: .45.6 .	<code>double d = double.Parse("45.6")</code>	من string إلى double
أسهل طريقة هي إضافة الرقم إلى نص فارغ. النتيجة: "97" .	<code>string s = 97;"" +</code>	من int إلى string

شريحة 19: لغة الحاسوب للحروف: ASCII

بما أن الحاسوب يفهم الأرقام فقط، كيف يتعامل مع الحروف والنصوص؟

عن طريق جدول ASCII: وهو قاموس يربط كل حرف ورمز برقم فريد.

- American Standard Code for Information (ASCII) هو اختصار لـ (Interchange).

• أمثلة من القاموس:

- الحرف 'A' يقابل الرقم 65.
- الحرف 'a' يقابل الرقم 97.
- الحرف الذي يمثل الرقم '0' يقابل الرقم 48.

شريحة 20: متغير الحرف الواحد: char

لدينا نوع خاص من "الصناديق" مخصص لتخزين حرف واحد فقط، وهو char.

- بما أن الحاسوب يخزن كل حرف كرقم ASCII، يمكننا تعريف متغير char بطريقتين:
//الطريقة الأولى: مباشرة باستخدام الحرف

```
char ch1 = 'a';
```

//الطريقة الثانية: باستخدام رقم ASCII الخاص بالحرف

```
char ch2 = (char)97;
```

//في الحالتين، النتيجة واحدة عند الطباعة

```
Console.WriteLine("ch1=" + ch1 + ", ch2=" + ch2);
```

الناتج: ch1=a, ch2=a

شريحة 21: الفرق بين الرقم والحرف الذي يمثل الرقم

هذه نقطة مهمة جداً لتجنب الأخطاء:

- '8': هذا هو الرمز أو الحرف '8'. عند تخزينه في متغير `char`, يبحث الكمبيوتر في جدول ASCII ويجد أن رقمه هو 56.
- 8: هذا هو الرقم ثمانيّة كقيمة حسابية.

الكمبيوتر يخزن الرقم 56

الكمبيوتر يخزن الرقم 8 (وهو رمز تحكم غير قابل للطباعة) //

للهذا السبب عند التعامل مع أرقام داخل البرنامج، نستخدم النوع `int` وليس `char`.

شريحة 22: تمرين 1 - بناء رقم من أجزائه (باستخدام `int`)

المهمة: كتابة برنامج يقرأ 3 أرقام منفصلة (آحاد، عشرات، مئات) من المستخدم، ثم يدمجها معًا لتكوين رقم واحد كامل.

```
int digit1, digit2, digit3;
```

```
int newNum = 0;
```

```
Console.Write("Enter the hundreds digit: ");
```

مثال: 3

```
Console.Write("Enter the dozens digit: ");
```

مثال: 4

```
Console.Write("Enter the unit digit: ");
```

مثال: 8

// المعادلة السحرية للدمج

```
newNum = digit1 * 100 + digit2 * 10 + digit3;
```

```
Console.WriteLine("The number is " + newNum);
```

الناتج: The number is 348

شريحة 23: تمرين 2 - فخ التعامل مع الحروف `char` كأرقام

ماذا لو حاولنا حل نفس التمرين باستخدام `char`؟

```
char digit1, digit2, digit3;  
// نقرأ الحروف '3', '4', '8' من المستخدم ...  
// وليس الأرقام ASCII انتبه! هذا الكود سيجمع قيم //  
// '3' (51) + '4' (52) + '8' (56) = 159  
int wrongNum = digit1 + digit2 + digit3;  
Console.WriteLine("Wrong result: " + wrongNum);
```

الناتج: Wrong result: 159

الدرس المستفاد: لا يمكن إجراء عمليات حسابية مباشرة على الحروف التي تمثل أرقاماً. يجب تحويلها إلى النوع `int` أولاً لتكون أرقاماً حقيقة.

شريحة 24: حيلة برمجية: تحويل الحرف-الرقم إلى رقم صحيح

تذكرون في التمرين السابق أن جمع الحروف '3' و '4' أعطى نتيجة خاطئة لأنه جمع قيم ASCII.
إليكم الحل!

الحيلة: الأرقام من '0' إلى '9' تأتي متسلسلة في جدول ASCII.

- قيمة ASCII للحرف '0' هي 48
- قيمة ASCII للحرف '1' هي 49
- قيمة ASCII للحرف '8' هي 56

إذا، للحصول على القيمة الرقمية الحقيقة لأي حرف، نطرح منه قيمة ASCII للحرف '0'.

مثال:

للحصول على الرقم 8 من الحرف '8':

$$8 \leftarrow 48 - 56 \leftarrow '0' - '8'$$

شريحة 25: تصحيح تمرين بناء الرقم (باستخدام `char`)

الآن، لنصلح كود التمرين السابق باستخدام هذه الحيلة الذكية.

```
char digit1, digit2, digit3;
```

```
int newNum = 0;
```

(قراءة الحروف '3', '4', '8' من المستخدم) ... //

المعادلة الصحيحة بعد تحويل كل حرف إلى رقم //

```
newNum = (digit1 - '0') * 100 + (digit2 - '0') * 10 + (digit3 - '0');
```

```
Console.WriteLine("The number is " + newNum);
```

الناتج: The number is 348

وبهذا نكون قد عالجنا المشكلة بنجاح، وتعلمنا كيف نتعامل مع المدخلات الرقمية كحروف.

شريحة 26: طباعة الحروف الخاصة (Escape Sequences)

ماذا لو أردنا طباعة رمز له معنى خاص في البرمجة، مثل علامة الاقتباس " داخل نص؟ أو الانتقال لسطر جديد؟

نستخدم الرمز \ (Backslash)، الذي يخبر البرنامج أن الحرف التالي له معنى خاص.

الرمز	معناه الخاص
\n	: الانتقال إلى سطر جديد.
\t	: إضافة مسافة بادئة كبيرة (تاب).
\"	: طباعة علامة الاقتباس المزدوجة ".
\\"	: طباعة رمز \ نفسه.

مثال:

Console.WriteLine("Hello World!\\n\\t"); // هذا سطر جديد قال المبرمج ("");

الناتج:

قال المبرمج: "Hello World"

هذا سطر جديد.

شريحة 27: الثوابت (Constants) - لماذا نحتاجها؟

لنفترض أننا نكتب برنامجًا لحساب سعر منتج بعد إضافة ضريبة القيمة المضافة (17%).

// الكود يقرأ السعر من المستخدم ...

"استخدام "رقم سحري // totalPrice = price * 1.17;"

المشكلة: هذا الرقم 1.17 غامض. ولو تغيرت الضريبة، سنحتاج للبحث عنه وتغييره في كل مكان بالكود.

الحل الأفضل؟ نستخدم متغيراً.

double tax = 17;

totalPrice = price + (price * tax / 100);

لكن تبقى مشكلة: قد يقوم مبرمج آخر بتغيير قيمة المتغير tax عن طريق الخطأ في مكان ما بالبرنامج!

شريحة 28: الحل الأمثل: الكلمة المفاتيحية const

لحل هذه المشكلة، نستخدم الثوابت (Constants). الثابت هو متغير "للقراءة فقط"، لا يمكن تغيير قيمته بعد تعريفه إطلاقاً.

• نعرف الثابت باستخدام الكلمة const.

• من المتعارف عليه كتابة أسماء الثوابت بالحروف الكبيرة (UPPER_CASE) لتسهيل تمييزها.

const double VAT_PERCENT = 1.17;

// ...

totalPrice = price * VAT_PERCENT;

الفوائد:

1. الكود أوضح وأسهل للقراءة.
 2. أكثر أماناً: يمنع تغيير القيمة عن طريق الخطأ.
 3. سهولة التعديل: إذا تغيرت الضريبة، نغيرها في مكان واحد فقط.
-

شريحة 29: مشروع الوحدة - حاسبة أجرة التاكسي (الخطة)

المهمة: بناء برنامج لحساب أجرة رحلة في تاكسي بناءً على المعطيات التالية:

- سعر بدء الرحلة (ثابت): 10.20 شيكل.
- السعر لكل كيلومتر (ثابت): 1.30 شيكل.
- السعر لكل حقيبة أمتعة (ثابت): 2.00 شيكل.

خطة العمل (الخوارزمية):

1. **تعريف الثوابت:** .PRICE_PER_LUGGAGE ,PRICE_PER_KM ,START_FEE
 2. **استقبال المدخلات:** سؤال المستخدم عن عدد الكيلومترات وعدد الحقائب.
 3. **العملية الحسابية:** حساب السعر الإجمالي باستخدام المعادلة.
 4. **عرض المخرجات:** طباعة المبلغ النهائي المطلوب للدفع.
-

شريحة 30: مشروع الوحدة - الكود النهائي

إليكم تطبيق خطة العمل في كود C# كامل.

```
Public static void Main()
```

```
{
```

```
    //تعريف الثوابت.
```

```
    const double START_FEE = 10.20;
```

```
    const double PRICE_PER_KM = 1.30;
```

```
    const double PRICE_PER_LUGGAGE = 2.00;
```

```
    //تعريف متغيرات للمدخلات والنتائج
```

```

double numOfKm;
int numOfLuggages;
double total;

// استقبال المدخلات من المستخدم . 2
Console.WriteLine("How many kilometers? ");
numOfKm = double.Parse(Console.ReadLine());

Console.WriteLine("How many luggages? ");
numOfLuggages = int.Parse(Console.ReadLine());

// العملية الحسابية . 3
total = START_FEE + (numOfKm * PRICE_PER_KM) + (numOfLuggages *
PRICE_PER_LUGGAGE);

// عرض المخرجات . 4
Console.WriteLine("You need to pay " + total + " NIS");

}

```

شريحة 31: ملخص الوحدة الشامل

في هذه الوحدة، تعلمنا أساسيات البرمجة التي تمكنا من بناء برامج قوية:

- الطباعة على الشاشة (WriteLine و Write).
 - تعريف المتغيرات لتخزين المعلومات (char ,string ,double ,int).
 - استقبال معلومات من المستخدم (ReadLine).
 - فهم كيف يتم تخزين المتغيرات في الذاكرة بشكل مؤقت.
 - إسناد القيم للمتغيرات وأهمية التمهيد.
 - استخدام الثوابت (const) لجعل الكود أكثر أماناً ووضوحاً.
- لقد أنقذتم الآن الأدوات الأساسية التي يحتاجها كل مبرمج. أحسنتم!

وظيفة شاملة: أساسيات البرمجة في C#

الاسم:

التاريخ:

التعليمات: أجب عن جميع الأسئلة التالية. اقرأ كل سؤال بعناية قبل الإجابة.

الجزء الأول: أسئلة الاختيار من متعدد (8 أسئلة)

ضع دائرة حول الإجابة الصحيحة:

1. ما هو الفرق الأساسي بين `Console.WriteLine()` و `Console.Write()`؟

أ. أسرع من `WriteLine`.

ب. يطبع النص ثم ينتقل لسطر جديد، أما `Write` فيبقى بنفس السطر.

ج. يستخدم لطباعة الأرقام و `WriteLine` للنصوص.

د. لا يوجد فرق بينهما.

2. أي من أسماء المتغيرات التالية يعتبر اسمًا غير صالح في لغة C#؟

أ. `studentName`.

ب. `age_`

ج. `student_1`

د. `stStudent1`

3. ما هي وظيفة الكلمة المفتاحية `const`؟

أ. تجعل المتغير أسرع.

ب. تعرف متغيرًا نصيًّا.

ج. تجعل قيمة المتغير ثابتة ولا يمكن تغييرها بعد تعریفها.

د. تستخدم لطباعة نص على الشاشة.

4. الأمر `Console.ReadLine()` يقرأ مدخلات المستخدم ويعيدها دائمًا على شكل:

أ. `int` (رقم صحيح)

ب. `double` (رقم عشري)

ج. `string` (نص)

د. `char` (حرف)

5. ما هي نتيجة تنفيذ الكود التالي؟ `int num = (int)12.9;`

أ. 13.

ب. 12.9

ج. 12

د. سيعطي البرنامج خطأ

6. أي من الرموز التالية يستخدم لبدء "تسلسل هروب" (Escape Sequence) لطباعة الحروف الخاصة؟

أ. /

ب. \

ج. *

د. #

7. ما هو جدول ASCII؟

أ. جدول لتصحيح الأخطاء البرمجية.

ب. قاموس يربط كل حرف ورمز برقم فريد لفهمه الحاسوب.

ج. قائمة بالكلمات المحفوظة في لغة C#.

د. نوع خاص من المتغيرات.

8. ما هي الدالة التي تعتبر نقطة البداية الرئيسية في أي برنامج C#؟

أ. `()Start`

ب. `()Program`

ج. `()Main`

د. `()Run`

الجزء الثاني: أكمل الفراغ (7 أسئلة)

أكمل الجمل التالية بالكلمة أو الرمز المناسب:

9. لتعريف متغير مخصص لتخزين الأعداد الصحيحة مثل 100 أو 50، نستخدم النوع

..... 10. في نهاية كل أمر برمجي في لغة C#, يجب أن نضع الرمز

..... 11. لتعريف متغير يمكنه تخزين حرف واحد فقط مثل 'A'، نستخدم النوع

..... 12. لتحويل متغير نصي (string) يحتوي على رقم إلى متغير من نوع int، نستخدم الأمر

..... 13. الحيلة البرمجية لتحويل حرف يمثل رقمًا (مثلاً '7') إلى قيمته الرقمية الحقيقية (7) هي عن طريق طرح منه.

..... 14. لطباعة مسافة Tab داخل نص، نستخدم تسلسل المروب

..... 15. الطريقة المتعارف عليها لتسمية المتغيرات في C# والتي تبدأ بحرف صغير ثم حرف كبير لكل كلمة جديدة (مثلاً myFirstName) تسمى

الجزء الثالث: أسئلة برمجية وتحديات (5 أسئلة)

اكتب كودًا كاملاً لحل كل من المسائل التالية:

16. برنامج التحية الشخصية:

اكتب برنامجًا يسأل المستخدم عن اسمه، ثم يطبع له رسالة ترحيب شخصية.

مثال للناتج:

ما هو اسمك؟

خالد

أهلًا بك يا خالد!

17. حاسبة مساحة المستطيل:

اكتب برنامجًا يسأل المستخدم عن طول وعرض مستطيل (يمكن أن تكون أرقامًا عشرية). ثم يقوم البرنامج بحساب مساحة المستطيل وطباعة الناتج. (تنذير: المساحة = الطول × العرض).

18. حساب العمر:

اكتب برنامجًا يسأل المستخدم عن سنة ميلاده، ثم يقوم بحساب عمره التقريري وطباعة الناتج. استخدم السنة الحالية (2025) كثابت `const` في برنامجك.

19. تحويل درجات الحرارة:

اكتب برنامجًا يقرأ درجة الحرارة بوحدة السيليزيوس (Celsius) من المستخدم، ثم يقوم بتحويلها إلى وحدة الفهرنهايت (Fahrenheit) ويعرض الناتج.

$$\text{Fahrenheit} = (\text{Celsius} * 9 / 5) + 32$$

20. تحدي الفاتورة:

محل ملابس يقدم خصمًا ثابتاً بقيمة 25% على جميع المشتريات. اكتب برنامجًا يقوم بالآتي:

أ. عَرَفْ الخصم (25%) كثابت `const`.

ب. اسأل المستخدم عن السعر الأصلي لقطعة.

ج. احسب السعر النهائي بعد الخصم.

د. اطبع السعر الأصلي، قيمة الخصم، والسعر النهائي بشكل واضح.

مثال للناتج:

أدخل السعر الأصلي: 100

السعر الأصلي: 100 شيكل

قيمة الخصم: 25 شيكل

السعر النهائي بعد الخصم: 75 شيكل

بالتوفيق!