كراسة تجريبيه
الرقم ( )

لا يجوز تداولها خارج
القـــوات المسـلحـة

كراسة #C

دائرة العمليات
مديرية صنف الحاسبات

المحتويات

ت	الفصل	الموضوع	الصفحة	
(أ)	(ب)	(ج)	من (هـ)	الى (و)
١	مقدمة		٢	٢
	الفصل الاول	نبذة تاريخية وطريقة تثبيت بيئة العمل	٣	٥
١	الفصل الثاني	Console Application	٦	٣٧
٢	الفصل الثالث	Windows Form Applications	٣٨	٥٦

C#

المقدمة

١- لغة C# هي لغة برمجة كائنية قوية تم تطويرها بواسطة Microsoft في أوائل العقد الأول من القرن الحادي والعشرين. تم تصميمه ليكون لغة حديثة وفعالة يمكن استخدامها لإنشاء مجموعة متنوعة من التطبيقات، من تطبيقات سطح المكتب إلى تطبيقات الويب والألعاب .

٢- هي لغة موجهة نحو الكائن، مما يعني أنها تسمح للمطورين بإنشاء كائنات وفئات تمثل كيانات العالم الحقيقي. كما أنه يدعم نماذج البرمجة الأخرى مثل البرمجة الإجرائية والبرمجة الوظيفية.

٣- هي لغة برمجة متعددة الاستخدامات وقوية يتم استخدامها في مجموعة واسعة من مشاريع تطوير البرمجيات. إن بساطتها وكفاءتها ودعمها عبر الأنظمة الأساسية تجعلها خيارًا شائعًا بين المطورين. سواء كنت تطور تطبيقات سطح المكتب أو تطبيقات الويب أو تطبيقات الهاتف المحمول أو الألعاب، فإن # C هي لغة تستحق التفكير فيها.

الفصل الأول

نبذة تاريخية وتنزيل وتثبيت بيئة العمل

٤. نبذة تاريخية

بدأ تطوير منصة دوت نت بكتابة مجموعة من مكتبات الصفوف، وقد استخدم نظام تصريف مدار اسمه Simple Managed C أو اختصاراً SMC للقيام بذلك. لاحقاً وبالتحديد في كانون الثاني ١٩٩٩ شكل أندرس هيلسبرغ فريقاً من المطورين بهدف بناء لغة جديدة اسمها كول، يشكل الاسم اختصاراً لعبارة «لغة غرضية التوجه شبيهة بلغة .قررت مايكروسوفت الإبقاء على هذا الاسم إلا أنها تخلت عن ذلك لاحقاً لأسباب قانونية لها علاقة بحقوق العلامات المسجلة. على التوازي مع ذلك أُعلن مشروع دوت نت رسمياً في مؤتمر للمطورين المحترفين (PDC) في تموز عام ٢٠٠٠ وأعيد تسمية اللغة إلى سي# كما تم تصدير وقت التنفيذ الخاص بلغة إيه إس بي دوت نت بالإضافة إلى مكتبات الصفوف إلى هذه اللغة.

قال أندرس هيلسبرغ في تموز عام ٢٠٠٠ أن سي# ليست «نسخة من جافا» بل أنها «أكثر قرباً إلى لغة سي++ «من ناحية التصميم.

في تشرين الثاني ٢٠٠٥ أعلن عن الإصدار ٢.٠ من سي# ومن هنا بدأت سي# بالتطور في اتجاهات متزايدة حيث تقوم سي# بالتعامل مع الأنماط العمومية كصفوف حقيقية وتولد الكود الخاص بها وقت التنفيذ

إضافة إلى ذلك فقد أضيفت إلى سي# مجموعة من الميزات الهامة بهدف تمكين استخدام البرمجة الوظيفية فيها كُللت بإضافة لينك في الإصدار ٣.٠ والإطار البرمجي الداعم لتعابير لامبدا والطرق الملحقة والأنماط غير المسماة. تمكن هذه الميزات المطور من استخدام تقنيات البرمجة الوظيفية عندما يكون من المستحسن القيام بذلك. إن إضافات لينك وغيرها من الميزات الوظيفية تساعد المطور على كتابة أسطر أقل عند القيام بمهام روتينية كالاستعلام من قاعدة بيانات أو إعراب ملف إكس إم إل أو البحث ضمن بنية معطيات بما يمكن من التركيز على هدف البرنامج المنطقي وتحسين مقروئيته وصيانته.

كان لدى سي# جالب للحظ اسمه آندي (سمي باسم أندرس هيلسبرغ (وقد أحيل إلى التقاعد في ٢٩ كانون الثاني عام ٢٠٠٤.

عُرضت سي# على لجنة آيزو الفرعية ٢٢ JTC ١/SC للمراجعة والتعيير، كان اسم المعيار ٢٣٢٧٠:٢٠٠٣ ISO/IEC وهو ملغى اليوم. تمت الموافقة فيما بعد على تعيير سي# وفق المعيارISO/IEC

٥. مميزات اللغة

أ- تحتوي #C على العديد من الميزات التي تسهّل عملية التعلم.

ب- إن بنية #C - إن صحّ القول - هي "معبّرة" للغاية، ولكنها بسيطة وسهلة التعلم.

ج- يبسط بناء #C العديد من تعقيدات.++C

د- سهلة القراءة نسبيًا.

هـ - لغة مكتوبةٌ بشكلٍ ثابتٍ، لذا يتم التحقق من الشفرة قبل أن يتم تحويلها إلى تطبيقٍ، وهذا ما يسهّل العثور على الأخطاء، وهو أمرٌ يمكن أن يكون مفيدًا بشكلٍ خاص للمبتدئين.

و- اسرع من اللغات المكتوبة ديناميكيًا لأن الأشياء محددةٌ بشكلٍ أكثر وضوحًا؛ وبالتالي، عندما يكون التطبيق قيد التشغيل، لن يتم هدار موارد الجهاز عند التحقق من تعريف شيءٍ ما في الشفرة.

٦- استخدامات اللغة

أ- العمل على تطبيقات على منصة تشغيل ويندوز .

ب- مجال تطبيقات سطح المكتب حيث أنه كانت البداية بمنصة .Net ، فلذلك فتستخدم لغة C# بشكل واسع في مجال تطبيقات سطح المكتب.

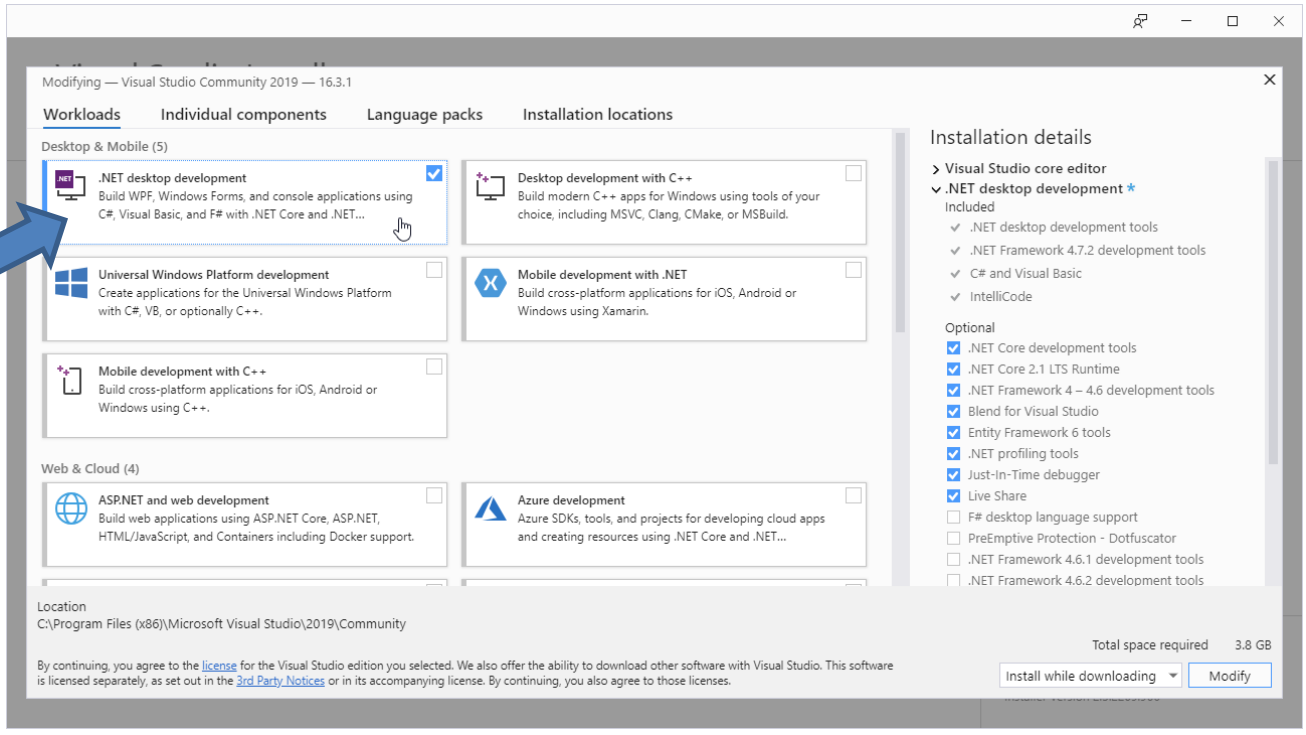
د- يمكن استخدام لغة C# في عمل تطبيقات الهاتف بنظام أندرويد و ios ولكن ذلك باستعانة ببرامج خاصة، فيستعان ببرنامج Xamarin لتستطيع برمجة تطبيقات لهواتف الأندرويد والذي هو أيضا من إنتاج مايكروسوفت، ويستعان ببرنامج Xamarin مع الفيجوال لتستطيع برمجة تطبيقات لهواتف ios

ذ- بالاستعانة مع Asp. Net تستطيع برمجة تطبيقات الويب باستخدام سي شارب .

٧- لتنزيل بيئة العمل نتبع الخطوات التالية

أ- في البداية يتم الذهاب الى الموقع الرسمي لشركة مايكروسوفت لتنزيل بيئة العمل فيجول استوديو

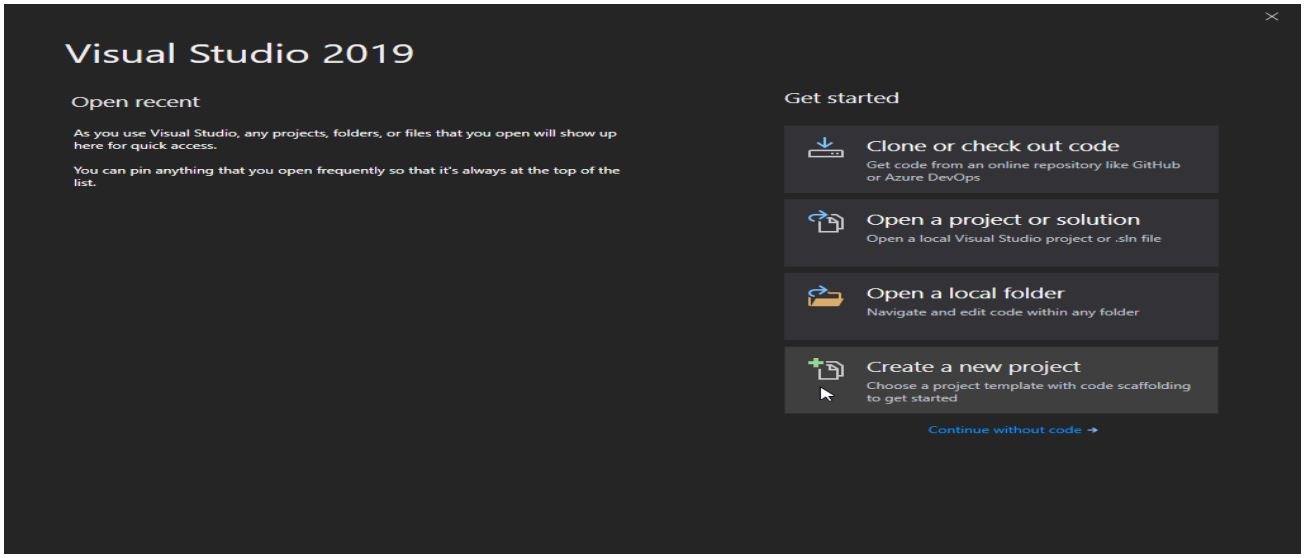
ب- بمجرد تنزيل Visual Studio Installer وتثبيته ، اختر حمل NET وانقر فوق الزر Modify: لاحظ الشكل رقم (١)



الشكل رقم (١)

ج- بعد اكتمال التثبيت يتم النقر فوق الزر "تشغيل" لبدء استخدام Visual Studio في نافذة البدء، **Create a new project**:

لأنشاء مشروع جديد لاحظ الشكل رقم (٢)

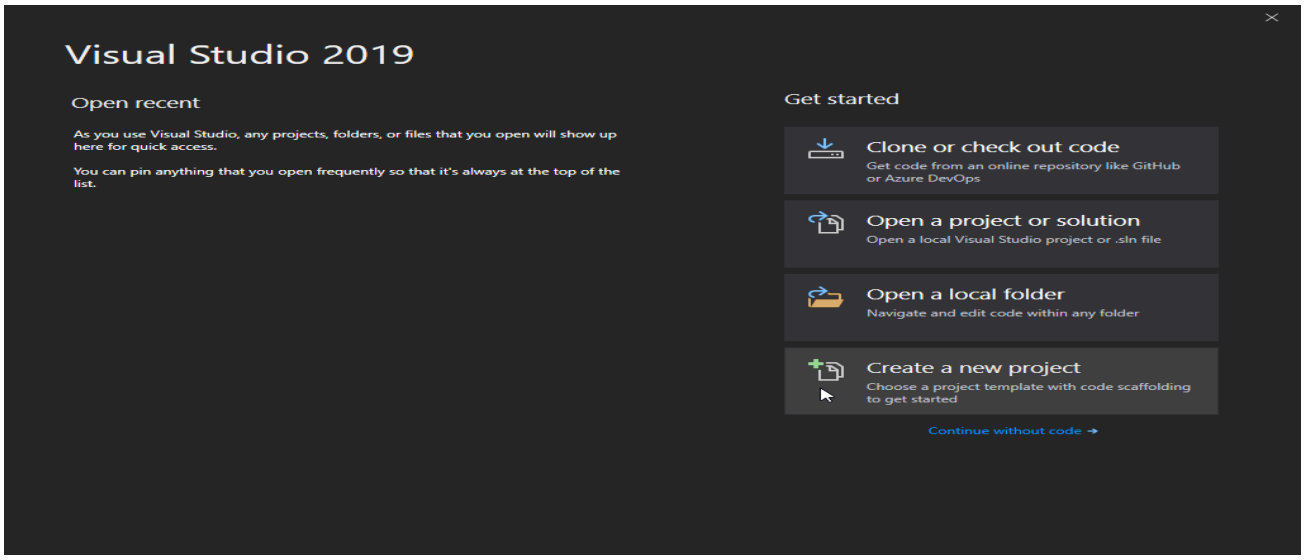

Visual Studio 2019
Open recent
As you use Visual Studio, any projects, folders, or files that you open will show up here for quick access.
You can pin anything that you open frequently so that it's always at the top of the list.
Get started
Clone or check out code
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code

الشكل رقم (٢)

الفصل الثاني
Console Application

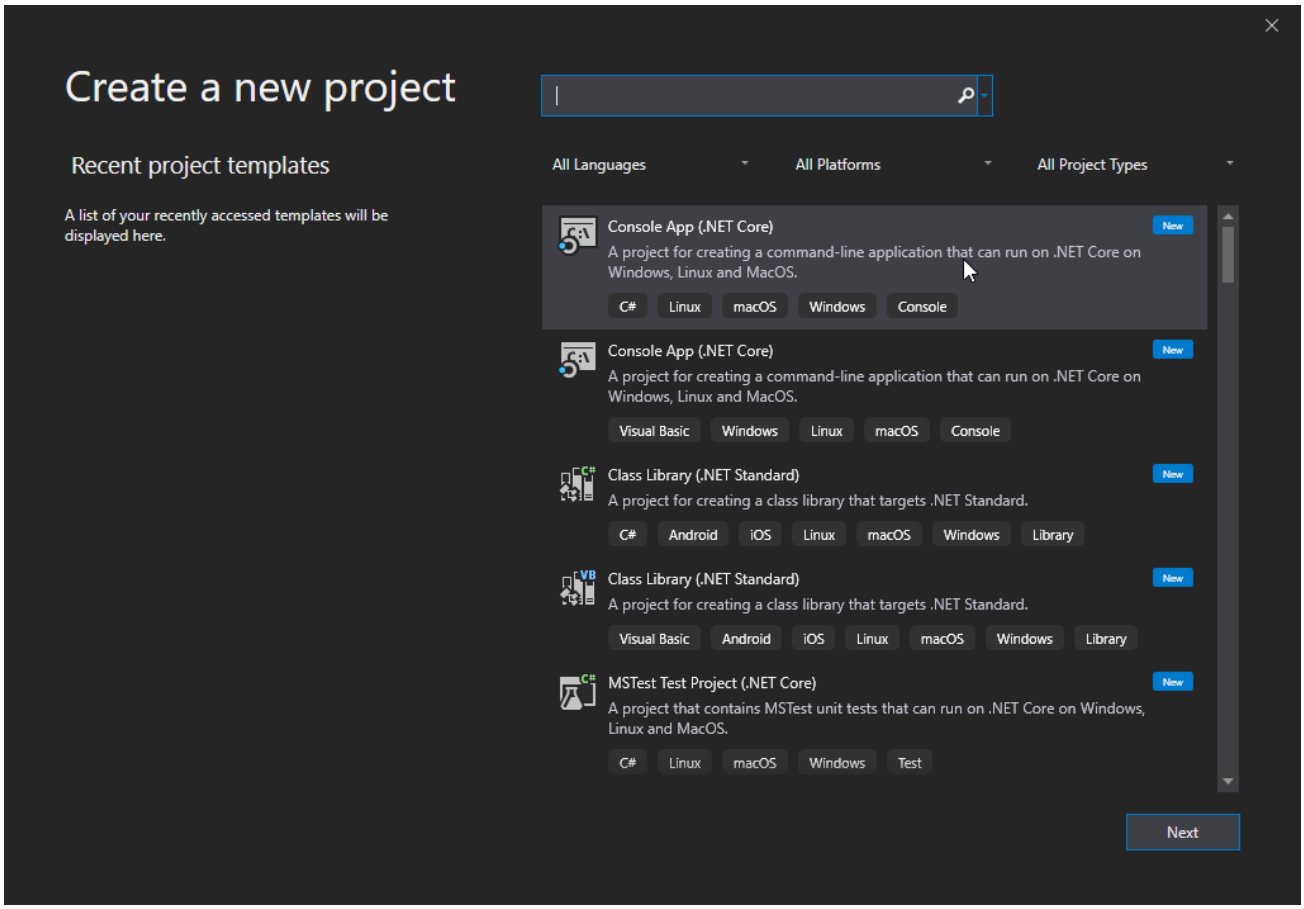
٨- انشاء مشروع جديد

أ- لأنشاء مشروع جديد يتم الذهاب الى فيجول استوديو ويتم اختيار Create a new project لاحظ الشكل (٣)



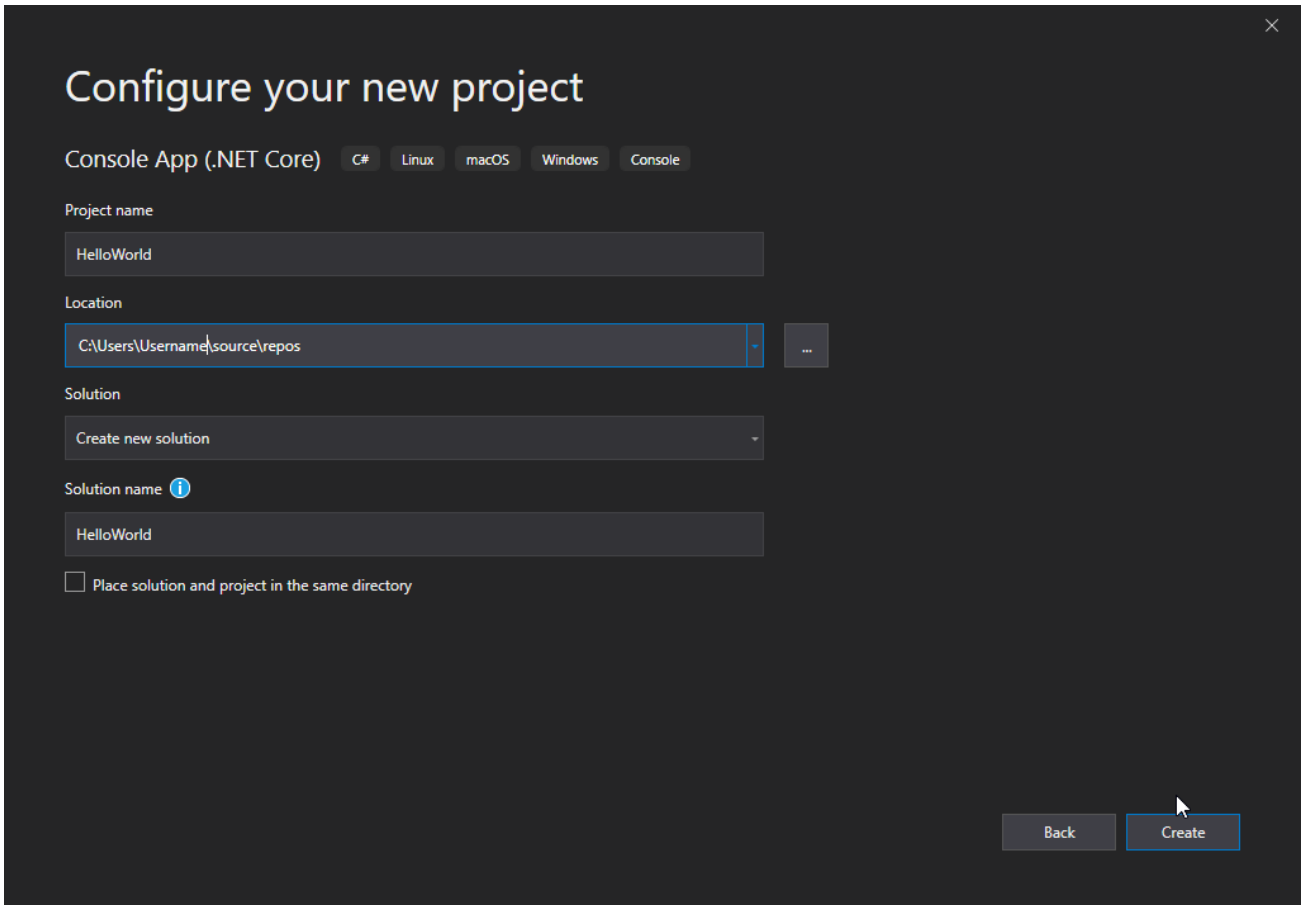
الشكل رقم (٣)

ب- يتم اختيار Console App (.NET Core) لاحظ الشكل رقم (٤)



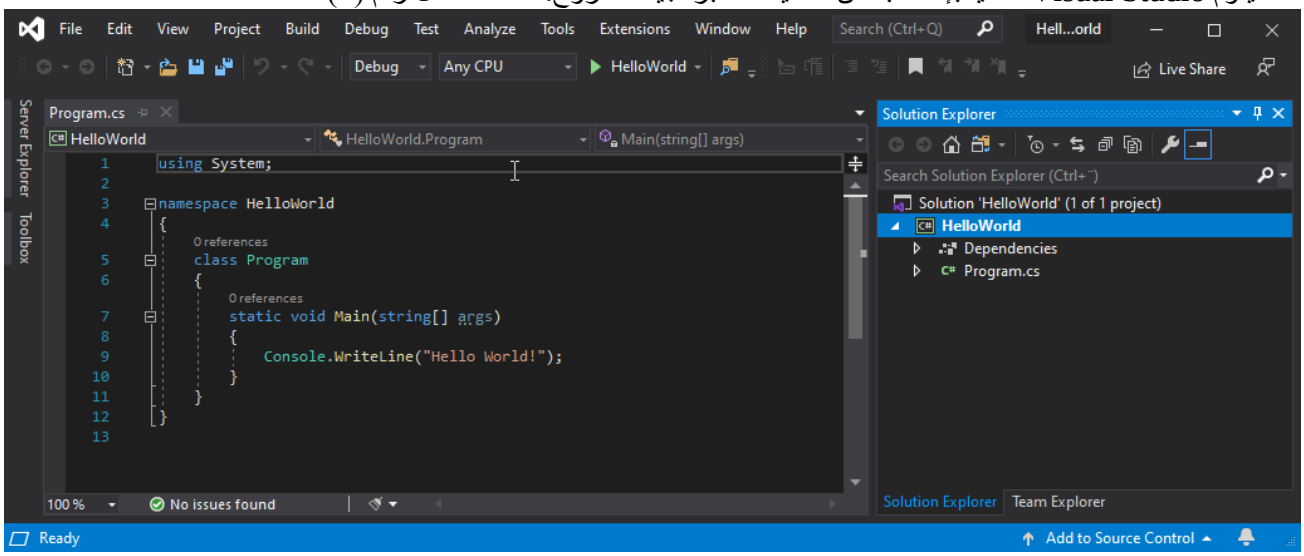
الشكل رقم (٤)

ج- بعدها يتم اختيار الاسم المطلوب للمشروع لاحظ الشكل رقم (٥)



الشكل رقم (٥)

د- سيقوم Visual Studio تلقائيا بإنشاء بعض التعليمات البرمجية لمشروع: لاحظ الشكل رقم (٦)



الشكل رقم (٦)

أكواد أساسية – الفئة (Console)

٩. وتنقسم الى :

أ. طباعة عبارة نصية

لإخراج القيم أو طباعة النص ، يمكن استخدام طريقة WriteLine() كما في المثال التالي
مثال ١/ طباعة عبارة نصية

```
Console.WriteLine("Hello World!");
```

يقوم الايعاز البرمجي بطباعة عبارة نصية، والانتقال للسطر التالي
كما يمكن طباعة أكثر من عبارة نصية، لاحظ المثال التالي
مثال٢/ طباعة اكثر من عبارة نصية

```
Console.WriteLine("Hello World!");

Console.WriteLine("I am Learning C#");
```

ب. التعليقات

يمكن استخدام التعليقات لشرح ايعاز برمجي ، وجعله أكثر قابلية للقراءة. يمكن استخدامه أيضا لمنع التنفيذ عند اختبار التعليمات البرمجية البديلة.
لكتابة تعليق مفرد كما في المثال التالي
مثال ٣/ كتابة تعليق مفرد

```
// This is a comment
```

ت. المتغيرات

المتغيرات هي حاويات لتخزين قيم البيانات. ، هناك أنواع مختلفة من المتغيرات (محددة بكلمات رئيسية مختلفة)

أولا. انواع المتغيرات

لاحظ الجدول رقم (١)

نوع المتغير	قيمته
Bool	قيمة منطقية trueأو false
Byte	عدد صحيح من ٠حتى ٢٥٥
DateTime	فئة خاصة بالتاريخ والوقت
Double	قيمة عشرية
Float	قيمة عشرية
Int	قيمة صحيحة
String	قيمة نصية

الجدول رقم (١)

ثانيا. التصريح عن المتغيرات

للتصريح عن المتغيرات يتم كتابة نوع المتغير يليه اسمه. هناك قواعد عديدة تحدد اسم المتغير أهمها ألا يكون كلمة محجوزة ولا يبدأ برقم ، ولا يحتوي على فراغات أو بعض الرموز مثل نقطة أو فاصلة منقوطة ، ولا يحتوي إلا على أحرف لاتينية ، كما أن (C#) حساسة لحالة الأحرف . هذا بالإضافة إلى قواعد أخرى كثيرة ، وان جميع لغات البرمجة لها ذات المبدأ بالنسبة لأسماء المتغيرات، مع فوارق بسيطة.

ثالثا. إسناد القيم للمتغيرات

يمكن اسناد القيم الى المتغيرات فور التصريح عنها ويمكن لاحقاً عند الحاجة يجب ان تتطابق نوع القيمة المسندة مع نوع المتغير. كما في المثال التالي

مثال ٤/ إعطاء قيمة لمتغير ومن ثم طباعة المتغير

```
string name = "John";

Console.WriteLine(name);
```

مثال ٥/ إعطاء قيم لمجموعة من المتغيرات

```
int myNum = ٥;

double myDoubleNum = ٥.٩٩D;

char myLetter = 'D';

bool myBool = true;

string myText = "Hello";
```

رابعاً التصريح عن الثوابت

يمكن لتطبيق أن يحتاج إلى قيم ثابتة على طول فترة تنفيذه ويمكن التصريح عنها بإضافة الكلمة المحجوزة (const) إلى ما قبل نوع المتغير أثناء التصريح مع إسناد قيمة لاحظ المثال

مثال ٦/ إعطاء قيم ثابتة للمتغير

```
const int myNum = ١٥;
```

قراءة المعطيات

١٠.عند التصريح عن المتغيرات سيبدأ البرنامج بسلسلة من المجريات التي قد تؤدي إلى احتياج البرنامج لقيم خارجية يعطيها المستخدم للبرنامج، فمثلا بإمكان برمجة تطبيق يقوم بحساب مسائل بسيطة او معقدة بناءاً على قيم يعطيها له المستخدم. لاحظ المثال التالي

مثال ٧/ ادخال المستخدم اسمه ومن ثم يطبع الاسم

```
Console.WriteLine("Enter username:");

string userName = Console.ReadLine();

Console.WriteLine("Username is: " + username);
```

الروابط

١. تقسم الروابط إلى الأنواع التالية : روابط رياضية ، منطقية ، مقارنة وإسناد
٢. **استخدامات الروابط :**

أ. العمليات الأساسية كالجمع والطرح والقسمة،
ب. باقي القسمة، ويستخدم عند قسمة عددين بحيث لا يكون الناتج صحيحا، كقسمة ١٠على ٣
جـ.الروابط المنطقية تعيد قيمة (True) أو (False)، صح أو خطأ
د. المقارنة تستخدم للمقارنة بين المتغيرات، وتتعامل مع أنواع المتغيرات أو قيمها.
هـ. الإسناد تستخدم لإسناد القيم للمتغيرات.
لاحظ الجدول التالي

العملية	الوصف
+	يُستخدم هذا المعامل لجمع رقمين أو أكثر.
-	يُستخدم لطرح الرقم الثاني من الرقم الأول.
*	يُستخدم لإجراء عملية الضرب على الأعداد
/	يُستخدم لإجراء عملية القسمة.
%	يُستخدم لحساب باقي القسمة.
++	يُستخدم لإجراء زيادة بمقدار واحد على العدد الصحيح
--	يُستخدم لإنقاص قيمة العدد الصحيح بمقدار واحد.

مثال ٨/ جمع عددين

```
int sum١ = ١٠٠;

int sum٢ = ٢٥٠;

int sum٣ = sum٢ + sum٢;
```

مثال ٩/ التحقق من عددين

```
int x = ٥;

int y = ٣;

Console.WriteLine(x > y); // returns True because ٥ is greater than ٣
```

١١. بعض الدوال الرياضية

أ. Max() لإيجاد أعلى قيمة بين عددين

مثال ١٠/ إيجاد العدد الأكبر بين عددين

```
Math.Max(٥, ١٠);
```

ب.Min() لإيجاد اقل قيمة بين عددين

مثال ١١/ إيجاد العدد الاصغر بين عددين

```
Math.Min(٥, ١٠);
```

جـ. Sqrt لايجاد الجدذ التربيعي

مثال ١٢/ إيجاد الجذر التربيعي للعدد

```
Math.Sqrt(١٤);
```

د. Abs لايجاد القيمة المطلقة لعدد

مثال ١٣/ إيجاد القيمة المطلقة للعدد

```
Math.Abs(-٤.٧);
```

مثال ١٤/ ايجاد حاصل ضرب عددين؟

```
int x = 5;
int y = 6;
int z = x * y;
Console.WriteLine("x*y="+z);
```

مثال ١٥/ ايجاد حاصل طرح عددين؟

```
int x = 5;
int y = 6;
int z = x - y;
Console.WriteLine("x-y="+z);
```

مثال ١٦/ ايجاد حاصل جمع عددين المستخدم يقوم بادخال القيم؟

```
Console.WriteLine("Enter The Number one");
int x=Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Enter The Number two");
int y = Convert.ToInt32(Console.ReadLine());
int z = x + y;
Console.WriteLine("The sum x+y="+z);
```

مثال ١٧/ زيادة قيم متغير عدد واحد؟

```
int x = 5;
x++;
Console.WriteLine(x);
```

مثال ١٨/ نقصان قيم متغير عدد واحد؟

```
int x = 5;
x--;
Console.WriteLine(x);
```

١٢. **العمليات المنطقية**

هي العمليات التي تساعد في صناعة القيود والشروط على شيء معين وبالتالي تمنح تحكم اكبر في الايعاز البرمجي والقيم المنطقية ترمز الى الاشياء التي تحتمل اكثر من احتمالين اما صح او خطأ.

مثال ١٩/ تحقق هل العدد الاول اكبر من العدد الثاني؟

```
int x = 5;
int y = 7;
bool b = (x > y);
Console.WriteLine(b)
```

١٣. **بنية الشرط (If):** يتم من خلالها اختبار قيمة الحالة التي يمر بها برنـامج وفق بنية شرط If عندما يتحقق الشرط سيتنفذ . الصيغة العامة لبنية الشرط هي :

```
if (condition)
{
  // block of code to be executed if the condition is True
}
```

مثال ٢٠/ تحقق هل العدد الاول اكبر من العدد الثاني باستخدم If؟

```
int x = ٢٠;
int y = ١٨;
if (x > y)
{
  Console.WriteLine("x is greater than y");
}
```

مثال٢١ / تحقق من الاصغر بين رقمين يدخلهم المستخدم ؟

```
Console.WriteLine("Enter value x");
int x =Convert.ToInt32( Console.ReadLine());
Console.WriteLine("Enter value y");
int y = Convert.ToInt32(Console.ReadLine());
if(x<y)
{
    Console.WriteLine("x is small");
}
else
{
    Console.WriteLine("y is small");
}
```

مثال٢٢ / التحقق من العدد اذا كان سالب او موجب ؟

```
int x;
x = Convert.ToInt32( Console.ReadLine());
if (x >=  0)
{
    Console.WriteLine(" العدد موجب ");
}
else
{
    Console.WriteLine(" العدد سالب ");

}
Console.ReadKey();
```

١٤. **بنية الشرط (IF..Else) :** يتم من خلالها التعامل بشرطين متعاكسين، بحيث يحدد الشرط الأول ويحدد الايعازات البرمجية كما في بنية الشرط (If)، ثم يحدد الايعازات البرمجية الحالة المعاكسة ضمن بنية (Else) ، وذلك وفق الصيغة :

```
if (condition)
{
  // block of code to be executed if the condition is True
}
else
{
  // block of code to be executed if the condition is False}
```

مثال ٢٣ / تحقيق من الرقم اذا كان زوجي ام فردي.

```
int x;
x = Convert.ToInt32( Console.ReadLine());
if (x % 2==0)
{
    Console.WriteLine("number even ");
}else
{
    Console.WriteLine(" nember odd ");

}
Console.ReadKey();
```

١٥. **بنية الشرط المتعددة** : يتم من خلالها استخدام العدد الذي تريد من بنى شرط داخل بنية أساسية.
الصيغة العامة لها :

```
if (condition١)

{

  // block of code to be executed if condition١ is True

}

else if (condition٢)

{

  // block of code to be executed if the condition١ is false and condition٢
is True

}

else

{

  // block of code to be executed if the condition١ is false and condition٢
is False
```

مثال٢٤ / تحقيق بين ثلاث ارقام وإيجاد الاكبر؟

```
int x, y, z;
   Console.WriteLine("enter x");
   x = Convert.ToInt32( Console.ReadLine());
   Console.WriteLine("enter y");
   y = Convert.ToInt32(Console.ReadLine());
   Console.WriteLine("enter z");
   z = Convert.ToInt32(Console.ReadLine());
   if (x >y&&x>z)
   {
       Console.WriteLine("x langer ");
   }else if (y>x && y>z)
   {
       Console.WriteLine(" y longer ");

   }
   else
   {
       Console.WriteLine(" z longer ");
   }
   Console.ReadKey();
```

١٦. **بنية الاختيار (Switch)** : يتم من خلالها مقارنة لحالة متغير ما .والصيغة التالية توضح بنية الاختيار :

```
switch(expression)
{
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
    break;
}
```

مثال ٢٥/ باستخدام بنية (switch) طباعة أيام الأسبوع حسب الرقم ؟

```
int day = ٤;
switch (day)
{
  case ١:
    Console.WriteLine("Monday");
    break;
  case ٢:
    Console.WriteLine("Tuesday");
    break;
  case ٣:
    Console.WriteLine("Wednesday");
    break;
  case ٤:
    Console.WriteLine("Thursday");
```

```
    break;
  case ٥:
    Console.WriteLine("Friday");
    break;
  case ٦:
    Console.WriteLine("Saturday");
    break;
  case ٧:
    Console.WriteLine("Sunday");
    break;
```

١٧. حلقة (For)

```
for (statement ١; statement ٢; statement ٣)
{
  // code block to be executed
}
```

مثال٢٦ / طباعة الارقام من ١ الى ١٠ باستخدام (for)؟

```
for(int i= 0; i <= 10; i++)
{
 Console.WriteLine(i);
}
 Console.ReadKey();
```

مثال٢٧ / كتابة برنامج يقوم بطباعة الاعداد الزوجية فقط التي بين العددين ٠ الى ١٠؟

```
for(int i= 0; i <= 10; i++)
          {
              if (i % 2 == 0)
              {

                  Console.WriteLine(i);
              }
          }
          Console.ReadKey();
```

مثال٢٨ / طباعة جدول الضرب للعدد ٤؟

```
for(int i = 0;i <= 10; i++)
{
    Console.WriteLine("4*"+i+"="+ 4 * i);
}
```

مثال٢٩ / طباعة جدول الضرب ؟

```
for(int i = 0;i <= 10; i++)
{
    for (int j = 0; j <= 10; j++)
    {
        Console.WriteLine(i+"*" + j + "=" + i * j);
    }
}
```

١٨. حلقة do – while

لا تختلف كثيرا هذه الحلقة عن حلقة for، إلا أن هذه الحلقة تنفذ التعليمات مرة واحدة على الأقل
الصيغة العامة لها :

```
do
{
  // code block to be executed
}
while (condition);
```

مثال٣٠ / باستخدام حلقة do – while طباعة الأرقام من ١ الى ٥ ؟

```
int i = ٠;
do
{
  Console.WriteLine(i);
  i++;
}
```

```
while (i < ٥);
```

مثال٣١ / باستخدام حلقة do – while طباعة جدول الضرب للعدد ٥ ؟

```
int i = ١, n = ٥, product;

do
{
    product = n * i;
    Console.WriteLine(n+" *"+i +"="+ product);
    i++;
} while (i <= ١٠);
```

١٩.حلقة (while)

تشابه حلقة (for) كثيرا وتختلف عنها بطريقة التنفيذ، كما أنها تختلف عن حلقة (do – while) بأنها لا تنفذ التعليمات إلا إذا تحقق الشرط، ولها الصيغة :

```
while (condition)
{
  // code block to be executed
}
```

مثال٣٢ / باستخدام حلقة while طباعة الأرقام من ١ الى ٥ ؟

```
int i = ٠;
while (i < ٥)
{
  Console.WriteLine(i);
  i++;
}
```

مثال٣٣ / باستخدام حلقة while طباعة مجموع الأرقام من ١ الى ٥ ؟

```
int i = 1, sum = 0;

while (i <= 5)
{
    sum += i;
    i++;
}
Console.WriteLine("Sum = "+sum);
```

٢٠. **Foreach**
تستخدم للتنقل عبر العناصر الموجودة للمصفوفة .

```
foreach (type variableName in arrayName)
{
  // code block to be executed
}
```

المصفوفات

٢١. المصفوفة هي عبارة عن مجموعة من المتغيرات، تشترك فيما بينها بالنوع وتختلف بالقيمة. من الممكن أن تكون المصفوفات أحادية البعد أو ثنائية الأبعاد أو ثلاثية الأبعاد لكن كلما زاد عدد أبعادها أصبحت أصعب للتخيل والتعامل .

التصريح عن المصفوفات

٢٢. الصيغة التالية تستخدم لتعريف المصفوفات أحادية البعد

```
string[] cars = new string[٤];
```

إسناد القيم لعناصر المصفوفات

٢٣. لا تختلف كثيرا طريقة إسناد القيم لعناصر المصفوفات عن المتغيرات العادية، إلا بفروقات بسيطة كترتيب العنصر في المصفوفة.

مثال٣٤/ طباعة اسماء أسماء السيارات باستخدام المصفوفات؟

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (int i = ٠; i < cars.Length; i++)
{
  Console.WriteLine(cars[i]);
}
```

التعامل مع المصفوفات

٢٤. توفر اللغة امكانيات كبيرة للتعامل مع المصفوفات والتي تناقش في الامثلة الاتية .

أ. للحصول على طول المصفوفة استخدم الخاصية (Length)
مثال٣٥/ إيجاد طول المصفوفة ؟

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

Console.WriteLine(cars.Length);

// Outputs ٤
```

ب. للحصول على أكبر وأصغر والمجموع قيمة استخدم التابعين (Max) و(Min) (Sum)
مثال٣٦/ إيجاد الرقم الأكبر والرقم الأصغر وإيجاد المجموع في المصفوفة ؟

```
int[] myNumbers = {٥, ١, ٨, ٩};

      Console.WriteLine(myNumbers.Max());  // returns the largest value

      Console.WriteLine(myNumbers.Min());  // returns the smallest value

      Console.WriteLine(myNumbers.Sum());  // returns the sum of elements
```

ج.ترتيب العناصر أبجديا (للعناصر النصية) وتصاعديا (للعناصر الرياضية)، استخدم الطريقة (Sort)
مثال٣٧/ ترتيب المصفوفة ؟

```
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

Array.Sort(cars);

foreach (string i in cars)

{

  Console.WriteLine(i);

}
```

مثال٣٨/ يطلب البرنامج التالي من المستخدم إدخال درجات ٥ طلاب في إحدى المواد الدراسيّة ومن ثمّ يحسب معدّل هؤلاء الطلبة في هذه المادّة، على افتراض أنّ الدرجة العظمى هي ١٠٠ ومن ثمّ يطبع المعدّل مع أسماء الطلاب ودرجاتهم على الشاشة؟

```
int[] arrMarks = new int[5];
 string[] arrNames = new string[5];
   int sum = 0;

   Console.WriteLine("Input Students Marks");
   Console.WriteLine("====================");

   //input loop.
    for (int i = 0; i < arrMarks.Length; i++)
      {
       Console.Write("Input student {0} th name: ", i + 1);
        arrNames[i] = Console.ReadLine();

       Console.Write("Input student {0} th mark: ", i + 1);
                     string tmpMark = Console.ReadLine();
                     arrMarks[i] = int.Parse(tmpMark);

                     Console.WriteLine();
                 }

              Console.WriteLine();
             Console.WriteLine("Students Marks Table");
              Console.WriteLine("====================");
             Console.WriteLine("No\tName\tMark");

              //calculating sum and display output loop.
              for (int i = 0; i < arrMarks.Length; i++)
                  {
                      sum += arrMarks[i];
                      Console.WriteLine("{0}\t{1}\t{2}",     i     +     1,
arrNames[i], arrMarks[i]);
                  }

              Console.WriteLine("--------------------");
              Console.WriteLine("Sum\t\t{0}", sum);
              Console.WriteLine("Average\t\t{0}", sum /
(double)arrMarks.Length);
              Console.WriteLine();
```

مثال٣٩/ ايجاد مجموع درجات طالب والمعدل واكبر درجة واصغر درجة ؟

```
int[] arrMarks =  { 100, 80, 66, 89, 58, 55, 67 };
Console.WriteLine("The sum "+arrMarks.Sum());
Console.WriteLine("The Average"+arrMarks.Average());
Console.WriteLine("The Max"+arrMarks.Max());
Console.WriteLine("The Min"+arrMarks.Min());
```

اللوائح (Lists)

٢٥. اللوائح هي أبسط شكل للمصفوفات، والتي تمكن من اضافة او حذف العناصر بسهولة وذلك لان طولها ليس ثابتاً كما هو الحال في المصفوفات والتي لا يوجد فيها إمكانية إضافة أو حذف عناصر، فعناصر المصفوفة تبقى موجودة خلال فترة استخدامها في البرنامج مع إمكانية تغيير قيمها فقط.

التعامل مع اللوائح

٢٦.بشكل أساسي تتميز اللوائح عن المصفوفات بإمكانية الإضافة والحذف، وذلك من خلال الطريق (Add) , (Remove) و (Remove At).

أ. إضافة عنصر

تختلف اللوائح عن المصفوفات شكل الاقواس ولا نقوم بإسناد القيم للعناصر وإنما بإضافتها لأنها غير موجودة أصلا، حيث نضيف العنصر وقيمته .

مثال ٤٠ / كتابة برنامج باستخدام (LIST) ادخال قيم والتحقق منها اذا كانت اكبر من ٥٠ طباعة ناجح واذا كانت اصغر من ٥٠ طباعة راسب .

```
List<int> mark = new List<int>() ;
mark.Add(29);
mark.Add(89);
mark.Add(40);
mark.Add(22);
foreach(int m in mark)
{
if(m>50)
Console.WriteLine("passed "+ m);
else
Console.WriteLine("faled " + m);

}

Console.ReadKey();
```

ب. حذف العناصر

ويكون باستخدام رتبة العنصر أو قيمته، وهذه الميزة غير موجودة بالمصفوفات وكما موضح في المثال التالي :

مثال ٤١ /

```
List<int> mark = new List<int>() ;
mark.Add(29);
mark.Add(89);
mark.Add(40);
mark.Add(22);
mark.Add(77);
mark.Remove(29);
foreach(int m in mark)
{

Console.WriteLine(+ m);

}

Console.ReadKey();
```

القاموس

٢٧. القاموس <TKey, TValue> عبارة عن مجموعة عامة تخزن أزواج القيمة الرئيسية بدون ترتيب معين.

٢٨. خصائص القاموس
أ. يقوم القاموس<TKey, TValue> بتخزين أزواج القيمة الرئيسية.
ب. يأتي ضمن System.Collections مساحة الاسم العامة.
ج. يطبق واجهة IDictionary<TKey, TValue>.
هـ. يجب أن تكون المفاتيح فريدة ولا يمكن أن تكون فارغة.
و. يمكن أن تكون القيم فارغة أو مكررة.
ز. يمكن الوصول إلى القيم عن طريق تمرير المفتاح المرتبط في المفهرس

٢٩. **أنشاء قاموس**
يمكنك إنشاء كائن عن طريق تمرير نوع المفاتيح والقيم التي يمكنه تخزينها. يوضح المثال التالي كيفية إنشاء قاموس وإضافة أزواج قيمة المفتاح.
مثال ٤٢/ إنشاء قاموس وإضافة عناصر

```
IDictionary<int, string> numberNames = new Dictionary<int, string>();
numberNames.Add(١,"One"); //adding a key/value using the Add() method
numberNames.Add(٢,"Two");
numberNames.Add(٣,"Three");
foreach(KeyValuePair<int, string> kvp in numberNames)
    Console.WriteLine("Key: {٠}, Value: {١}", kvp.Key, kvp.Value);
```

Stack

٣٠. Stack: هو نوع خاص من المجموعات التي تقوم بتخزين العناصر بأسلوب(LIFO (Last In First Out.
يتضمن C# فئات Stack<T> العامة وفئات مجموعة Stack غير العامة. يوصى باستخدام مجموعة< Stack<T العامة.
يعد المكدس مفيدًا لتخزين البيانات المؤقتة بأسلوب LIFO.
أ. يمكن إضافة العناصر باستخدام طريقة ()Push.
ب. يمكن استرجاع العناصر باستخدام طريقتي ()Pop و ()Peek.

يقوم المثال التالي بإنشاء وإضافة عناصر في Stack<T> باستخدام طريقة ()Push.
مثال ٤٣/

```
Stack<int> myStack = new Stack<int>();
myStack.Push(١);
myStack.Push(٢);
myStack.Push(٣);
myStack.Push(٤);

foreach (var item in myStack)
     Console.Write(item + ","); //prints ٤,٣,٢,١,
```

٣١. تقوم طريقة ()Pop بإرجاع العنصر الأخير وإزالته من المكدس. يتم التحقق دائمًا من عدد العناصر الموجودة في المكدس قبل استدعاء طريقة ()Pop.
مثال٤٤/ الوصول إلى المكدس باستخدام ()Pop.

```
Stack<int> myStack = new Stack<int>();
myStack.Push(١);
myStack.Push(٢);
myStack.Push(٣);
myStack.Push(٤);

Console.Write("Number of elements in Stack: {٠}", myStack.Count);

while (myStack.Count > ٠)
     Console.Write(myStack.Pop() + ",");

Console.Write("Number of elements in Stack: {٠}", myStack.Count);
```

٣٢. تقوم طريقة ()Peek بإرجاع القيمة المضافة الأخيرة من المكدس ولكنها لا تقوم بإزالتها. يتم التحقق دائمًا من العناصر الموجودة في المكدس قبل استرداد العناصر باستخدام طريقة ()Peek.

مثال٤٥/ استرداد العناصر باستخدام ()Peek.

```
Stack<int> myStack = new Stack<int>();
myStack.Push(١);
myStack.Push(٢);
myStack.Push(٣);
myStack.Push(٤);

Console.Write("Number of elements in Stack: {٠}", myStack.Count);//
prints ٤

if(myStack.Count > ٠){
     Console.WriteLine(myStack.Peek()); // prints ٤
}
```

٣٣. تتحقق الطريقة Contains () مما إذا كان العنصر المحدد موجودًا في مجموعة Stack أم لا. ويرجع true إذا كان موجودا، وإلا فهو يرجع false.

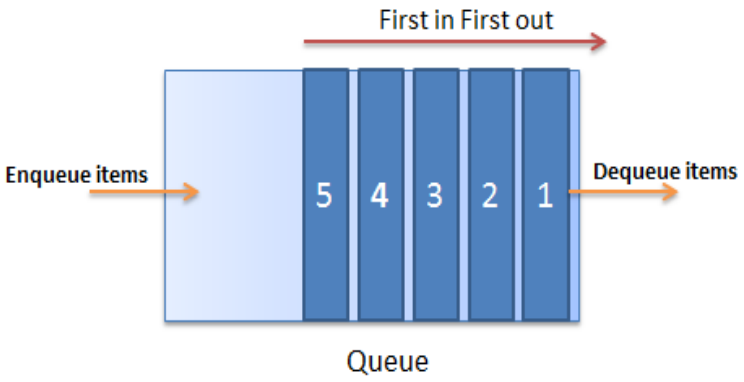
مثال ٤٦/ عن دالة Contains()

```
Stack<int> myStack = new Stack<int>();
myStack.Push(١);
myStack.Push(٢);
myStack.Push(٣);
myStack.Push(٤);

myStack.Contains(٢); // returns true
myStack.Contains(١٠); // returns false
```

Queue<T>

٣٤. هي نوع خاص من المجموعات التي تقوم بتخزين العناصر بأسلوب FIFO (First In First Out)، وهو عكس مجموعة Stack<T> تمامًا. أنه يحتوي على العناصر بالترتيب الذي تمت إضافتها. لاحظ الصورة التالية



٣٥. **Creating a Queue**

يمكنك إنشاء كائن Queue<T> عن طريق تحديد معلمة نوع لنوع العناصر التي يمكنه تخزينها. يقوم المثال التالي بإنشاء وإضافة عناصر في Queue باستخدام دالة Enqueue().

مثال ٤٧/

```
Queue<int> callerIds = new Queue<int>();
callerIds.Enqueue(١);
callerIds.Enqueue(٢);
callerIds.Enqueue(٣);
callerIds.Enqueue(٤);

foreach(var id in callerIds)
    Console.Write(id); //prints ١٢٣٤
```

٣٦. **دالة Peek()**

تقوم دالة Peek() دائمًا بإرجاع العنصر الأول . كما في المثال التالي

مثال ٤٨/

```
Queue<string> strQ = new Queue<string>();
strQ.Enqueue("H");
strQ.Enqueue("e");
strQ.Enqueue("l");
strQ.Enqueue("l");
strQ.Enqueue("o");
```

```
Console.WriteLine("Total elements: {٠}", strQ.Count); //prints ٥

if(strQ.Count > ٠){
    Console.WriteLine(strQ.Peek()); //prints H

}
```

٣٧ . **دالة Contains()**

تتحقق دالة Contains()من وجود عنصر في قائمة الانتظار أم لا. يتم إرجاعه صحيحًا إذا كان العنصر المحدد موجودًا، وإلا فإنه يُرجع خطأ. لاحظ المثال التالي

مثال ٤٨/

```
Queue<int> callerIds = new Queue<int>();
callerIds.Enqueue(١);
callerIds.Enqueue(٢);
callerIds.Enqueue(٣);
callerIds.Enqueue(٤);

callerIds.Contains(٢); //true
callerIds.Contains(١٠); //false
```

٣٨. **Exception Handling**

هو خطأ يقال في بعض الأحيان حدث ويقال استثناء- يحدث أثناء تشغيل البرنامج يقطع عملية تنفيذ الأوامر الخاصة بالبرنامج.
يجب معالجة الاستثناءات الموجودة في التطبيق لمنع تعطل البرنامج والنتيجة غير المتوقعة، وتسجيل الاستثناءات ومتابعة الوظائف الأخرى. يوفر C# دعمًا مدمجًا للتعامل مع الاستثناء باستخدام try , catch, finally.
الصيغة العامة لها

```
try
{
    // put the code here that may raise exceptions
}
catch
{
    // handle exception here
}
finally
{
    // final cleanup code
}
```

مثال ٥٠/ معالجة الاستثناءات باستخدام try, catch

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            Console.WriteLine("Enter a number: ");

            var num = int.parse(Console.ReadLine());

            Console.WriteLine($"Squre of {num} is {num * num}");
        }
       catch(Exception ex)
```

```
        {
            Console.Write("Error info:" + ex.Message);
        }
        finally
        {
            Console.Write("Re-try with a different number.");
        }
    }
}
```

الطرق والتوابع والإجراءات

٣٩.الطرق (Methods) :- هي مجموعه من الاوامر المجمعة تحت اسم معين وعند استدعاءها – ذكر اسمها يتم تنفيذ الأوامر التابعة لها الغاية منها تبسيط البرنامج باسهل شكل ممكن بحيث يسهل التعديل عليه وإيجاد الأخطاء واستخدامه في برامج أخرى .

الصيغة العامة للتصريح عن الدوال :

```
class Program
{
  static void MyMethod()
  {
    // code to be executed
  }
```

مثال٥١ / باستخدام الاجراءات كتابة برنامج يطبع كلمة معينة ؟

الحل/

```
static void print1()
{
Console.WriteLine("welcoome to c");

}
static void Main(string[] args)
{

print1();


}
```

مثال٥٢ / باستخدام الاجراء

```
    Console.WriteLine(fname + " is " + age);
}

static void Main(string[] args)
{
    MyMethod("Liam", 5);
    MyMethod("Jenny", 8);
    MyMethod("Anja", 31);
```

```
}
```

مثال٥٣ / باستخدام الاجراء كتابة برنامج يقوم بجمع رقمين ؟

```
static int add(int x , int y)
{
return x + y;

}
static void Main(string[] args)
{

int ans = (add(4, 10));
Console.WriteLine(ans);

Console.ReadKey();

}
```

مثال٥٤ / باستخدام الاجراء حساب مساحة المربع ؟

```
static void square(int nmbr)
{
    int sq = nmbr * nmbr;
    Console.WriteLine("Square of the given number is  " + sq);


}

static void Main(string[] args)
{


    square(2); //calling the method
}
```

٤٠.البرمجة كائنية التوجه

برمجة ذات نمط كائنية التوجُّه أو شيئية المنحو(يطلق عليها ايضًا برمجة موجهة نحو الكائنات) وهي نمط برمجة متقدم، وفيه يقسم البرنامج إلى وحدات تسمى الكائنات(Objects) ، كل كائن هو حزمة (تعليب) من البيانات المتغيرات والثوابت والطرق ووحدات التنظيم وواجهات الاستخدام. ويُبنى البرنامج بواسطة استخدام الكائنات وربطها مع بعضها البعض وواجهة البرنامج الخارجية باستخدام هيكلية البرنامج وواجهات الاستخدام الخاصة بكل كائن. من مميزات البرمجة الشيئية أنها تسمح بإعادة الاستخدام الايعازات البرمجية التي أُخْتُبِرَت وذلك باستدعائها في البرامج الأخرى دون إعادة برمجتها. إعادة الاستخدام يسهل بناء البرامج بشكل سريع في وقت قصير.

الفئات (Classes)

٤١. الكلاس يدل على مجموعة المتغيرات والخصائص والدوال التي تعبر **عن** كائن ، يتم تعريف الكلاس كفئة ويتم وضع المتغيرات ودوال التنفيذ ، هذا المفهوم اختصر وسهل البرمجة

مثال٥٥/ سيقوم المثال التالي بإنشاء كائن من فئة السيارة ، باسم myObj. ثم نطبع قيمة لون الحقول والسرعة القصوى:

```
class Car
{
  string color = "red";
  int maxSpeed = ٢٠٠;

  static void Main(string[] args)
  {
    Car myObj = new Car();
    Console.WriteLine(myObj.color);
    Console.WriteLine(myObj.maxSpeed);
  }
}
```

مثال٥٦/ مثال يوضّح كيفية إنشاء الصنف Employee وكيفيّة إنشاء كائنين منه ؟

```
class Employee      {
  public string FirstName;
  public string LastName;
  public double Salary;
  public string DisplayInfo()
          {
              string result = string.Format("{0} {1} - Salary: {2:N0}",
                  this.FirstName, this.LastName, this.Salary);

              return result;
          }
      }

      class Program
      {
              static void Main(string[] args)
          {
                      Employee employee1, employee2;

              employee1 = new Employee();
                      employee1.FirstName = "Mohammad";
                     employee1.LastName = "Mansoor";
```

```
                employee1.Salary = 1000;

            employee2 = new Employee();
                employee2.FirstName = "Saleh";
                employee2.LastName = "Mahmoud";
                employee2.Salary = 2500;

            Console.WriteLine("First Employee: {0}",
employee1.DisplayInfo());
                Console.WriteLine("Second Employee: {0}",
employee2.DisplayInfo());
            }
```

حدود تعريف الكائنات البرمجية

٤٢. يتم ملاحظة استعمال الكلمات (Public) و (Static) وغيرها فهي تعني هذه الكلمات تحدد مدى قدرتنا الى الوصول الى الكائن البرمجي سواء كان متغير ام دالة

أ- Public الكائنات البرمجية تكون معروفة على مستوى المشروع

ب- private الكائنات البرمجية تكون معروفة على مستوى الفئة التي تنتمي اليها

ج- protected الكائنات البرمجية تكون معروفة على مستوى الفئة التي تنتمي اليها والتي ترث منها

المجمعات

٤٣.المجمع عبارة عن ملف يضم كل ما يوجد في المشروع من ملفات او فئات ويكون امتداده اما (dll) او (exe) حسب نوع المشروع اذا كان تطبيق يكون امتداده (exe) واذا كان نوع المشروع مكتبة كان امتداده (dll) .

مجالات الأسماء

٤٤. هي تلك الاسماء التي تكون مسبوقة بالموجهة (using) وهي مجالات تضم العديد من الفئات والانواع كما يمكن للمجال ان يضم مجموعة من مجالات الاسماء كما هو الحال في مجال (system) الذي نجد به فئات كثرة منها (console) وكذلك يضم مجالات واسماء فرعية مثل (IO) واول سطر يكون في البرنامج هو سطر التأشير في مجالات الاسماء ويكون باستعمال الموجه (using) .

استعمال كلمة static

٤٥. وتعني المتغيرات والدوال التي تكون مسبوقة بهذه الكلمة يمكن استعمالها من غير استنساخ للفئة استعمالها مباشرة كما في المثال التالي .

مثال٥٧/

```
{
    class persone
    {
        static public int age;
        static public int reternage()
        {
            return age;
        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            persone.age = 21;
            int age = persone.reternage();
            Console.WriteLine(age);
            Console.ReadKey();
        }}}
```

المشيدات

٤٦. المشيد عبارة عن دالة تحمل نفس اسم الكلاس لكنها لا تعيد شيئا وهي اول جزء ينفذ عند انشاء كائن من فئة معينة دوره يكون في اعطاء قيم ابتدائية لمتغيرات الفئة وهو لا يحتاج الاعلان عن نوعه كما هو الحال في الدوال الاعتيادية بل لا يقبل حتى استعمال (void) كما في المثال التالي

مثال٥٨/

```
    class article
    {
        private int Code;
        private string Type;
        private double Prix;

         public article (int code,string type,double prix)
        {
            this.Code = code;
            this.Type = type;
            this.Prix = prix;

        }
        public double clicprix (int qunt)
```

```
        {
            double montant;
            montant = qunt * Prix;
            return montant;

        }
    }
    class Program
    {
        static void Main(string[] args)
        {
            article myart = new article(1,"ordinateur",4500);
            double motant = myart.clicprix(4);
            Console.WriteLine(motant);
            Console.ReadKey();

        }
    }
```

مثال٥٩/ باستخدام المشيدات انشاء صنف من نوع موظف واظهار اسمه وراتبه ؟

```
class Employee
       {
              public string FirstName;
           public string LastName;
          public double Salary;

      public Employee(string firstName, string lastName, double salary)
          {
                  this.FirstName = firstName;
                  this.LastName = lastName;
                  this.Salary = salary;
              }

          public Employee()
          {

          }

          public string DisplayInfo()
          {
                 string result = string.Format("{0} {1} - Salary:
{2:N0}",
                     this.FirstName, this.LastName, this.Salary);

              return result;
              }


     }


     class Program
```

```
    {
            static void Main(string[] args)
        {
                Employee employee1, employee2;

            employee1 = new Employee("Mohammad", "Mansoor", 1000);
                    employee2 = new Employee("Saleh", "Mahmoud", 2500);

           Console.WriteLine("First Employee: {0}",
employee1.DisplayInfo());
                    Console.WriteLine("Second Employee: {0}",
employee2.DisplayInfo());
                }
```

خصائص الفئات

٤٧. وهي طرق تستعمل من اجل التعامل مع المتغيرات الداخلية للفئة كما لو انها عامة بحيث الامكان للدخول اليها لقراءة قيمتها وللتعديل عليها معتمدة على الامرين (get) ويستعمل لأرجاع قيمة المتغير وكذلك الامر (set) يستعمل من اجل التعديل على المتغير كما في المثالين ٣١,٣٢ .

مثال٦٠/

```
// Create a Car class
class Car
{
  public string model;  // Create a field

  // Create a class constructor for the Car class
  public Car()
  {
    model = "Mustang"; // Set the initial value for model
  }

  static void Main(string[] args)
  {
    Car Ford = new Car();  // Create an object of the Car Class (this will
call the constructor)
```

```
    Console.WriteLine(Ford.model);  // Print the value of model
  }
}
```

مثال ٦١/

```
class Car
{
  public string model;
  public string color;
  public int year;

  // Create a class constructor with multiple parameters
  public Car(string modelName, string modelColor, int modelYear)
  {
    model = modelName;
    color = modelColor;
    year = modelYear;
  }

  static void Main(string[] args)
  {
    Car Ford = new Car("Mustang", "Red", ١٩٦٩);
    Console.WriteLine(Ford.color + " " + Ford.year + " " + Ford.model);
  }
}
```

تعدد التعاريف

٤٨. يتم الحديث عن تعدد التعاريف عندما تتوفر احدى الفئات على مجموعة من الدوال والاجراءات التي لها نفس الاسم ولكنها تختلف في المتغيرات الداخلية واحياناً النوع الذي تعيده وهذا المفهوم يستعمل بكثرة في البرمجة المتطورة حيث يسمح للمطور ان يوفر عليه عناء تغيير الاسماء .

الوراثة

٤٩. هي عملية إنشاء واشتقاق تصنيف جديد أو فئة جديدة (New class) بناءاً على فئة (class)التي تم تعريفها مسبقاً داخل الايعاز البرمجي، ويرتكز هذا المفهوم على أنه بإمكان كائن معين (object) من الاستفادة من خصائص وميزات الكائن الأعلى منه كما بإمكانه إضافة بعض الميزات الجديدة التي يرغب باستخدامها كما في المثالين التالين.

مثال٦٢/

```
class Vehicle  // base class (parent)
{
  public string brand = "Ford";  // Vehicle field
  public void honk()             // Vehicle method
  {
    Console.WriteLine("Tuut, tuut!");
  }
}

class Car : Vehicle  // derived class (child)
{
  public string modelName = "Mustang";  // Car field
}

class Program
{
```

```
  static void Main(string[] args)
  {
    // Create a myCar object
    Car myCar = new Car();
    // Call the honk() method (From the Vehicle class) on the myCar object
    myCar.honk();
    // Display the value of the brand field (from the Vehicle class) and the
value of the modelName from the Car class
    Console.WriteLine(myCar.brand + " " + myCar.modelName);
  }
}
```

مثال٦٣/

```
class Father
      {
           public Father()
           {
               Console.WriteLine("Father: In Constructor");
          }
          public void MyMethod()
          {
            Console.WriteLine("Father: In MyMethod");
          }
     }

     class Child : Father
     {
         public Child()
        {
             Console.WriteLine("Child: In Constructor");
         }

         public new void MyMethod()
         {
                 Console.WriteLine("Child: In MyMethod");
             }
      }

      class Program
      {
             static void Main(string[] args)
          {
                     Child c = new Child();
```

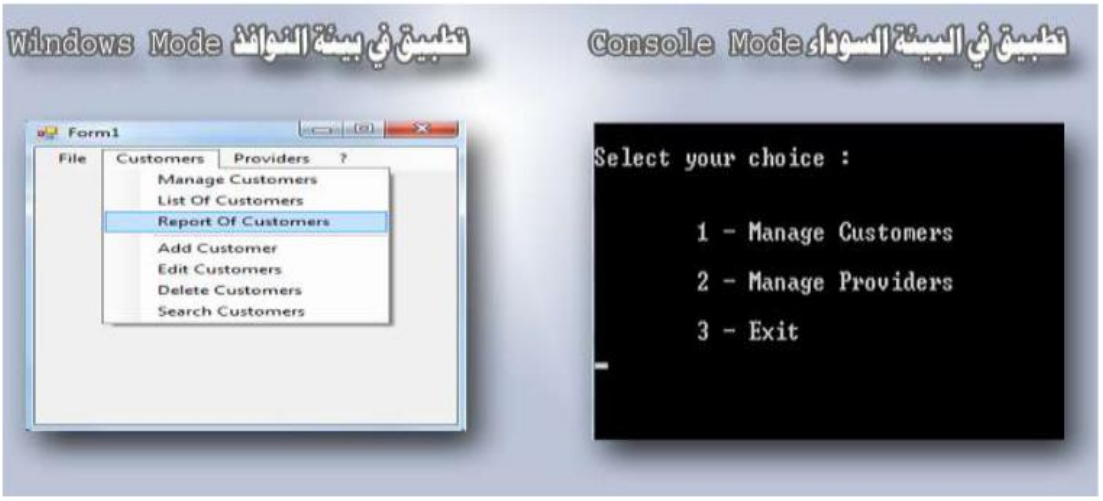
```
        c.MyMethod();

    }
```

الفصل الثالث
Windows Form Applications

مدخل الى برمجة الواجهات

٥٠. تختلف برامج البيئة السوداء على برمجة بيئة النوافذ الويندوز في طريقة العمل وفي النتائج وفي البيئة السوداء تسعى الى تسهيل الامر بالنسبة للمستخدم الا ان ذلك يبقى بدائياً بالمقارنة مع هيئة تطبيقات الويندوز اذ ان للمستخدم له امكانيات عديدة متاحة امامه هي تحريك الماوس وضغط وجذب واختيار وليس كالبيئة السوداء التي تحتم عليه استعمال لوحة المفاتيح فقط وتقييده بتنفيذ سطر قبل مروره الى غيره . كما سنعرض مثال يوضح الفرق بين البيئتين . لاحظ الشكل (٧) .



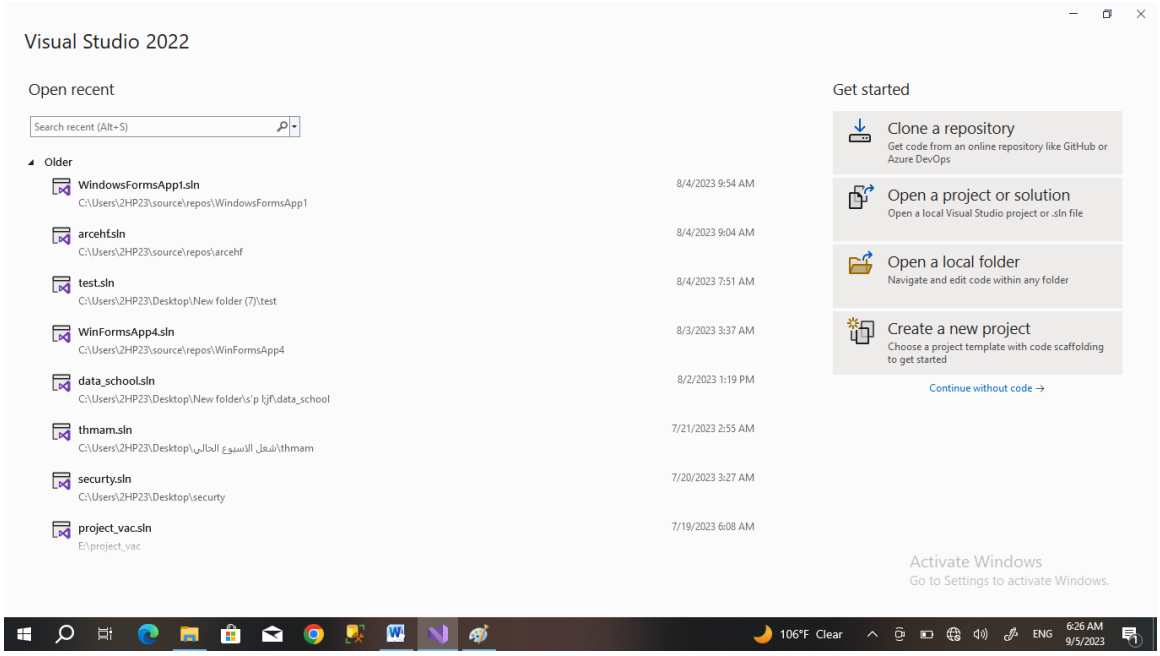
شكل رقم (٧)

بيئة التصميم (Design environment)

٥١. ويقصد بها ان كل البرامج التي تحتوي على بيئة الاشتغال وهي بمثابة ورشة تضم كل مكونات البرنامج التي قد يحتاجها المستخدم في عمله .

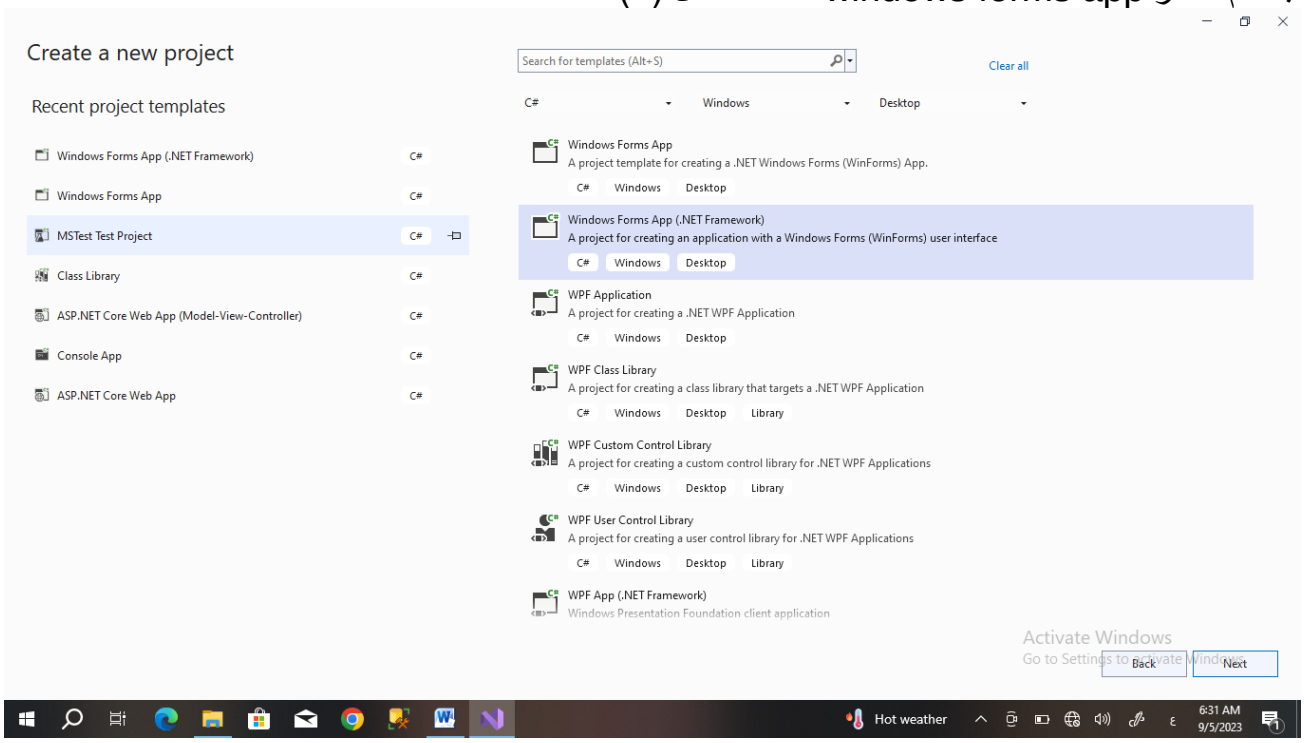
٥٢. لعمل مشروع جديد

أ- القيام بفتح برنامج فيجول استوديو واختيار (greate new project)
وبعد ذلك ستظهر النافذة . كما في الشكل رقم (٨)



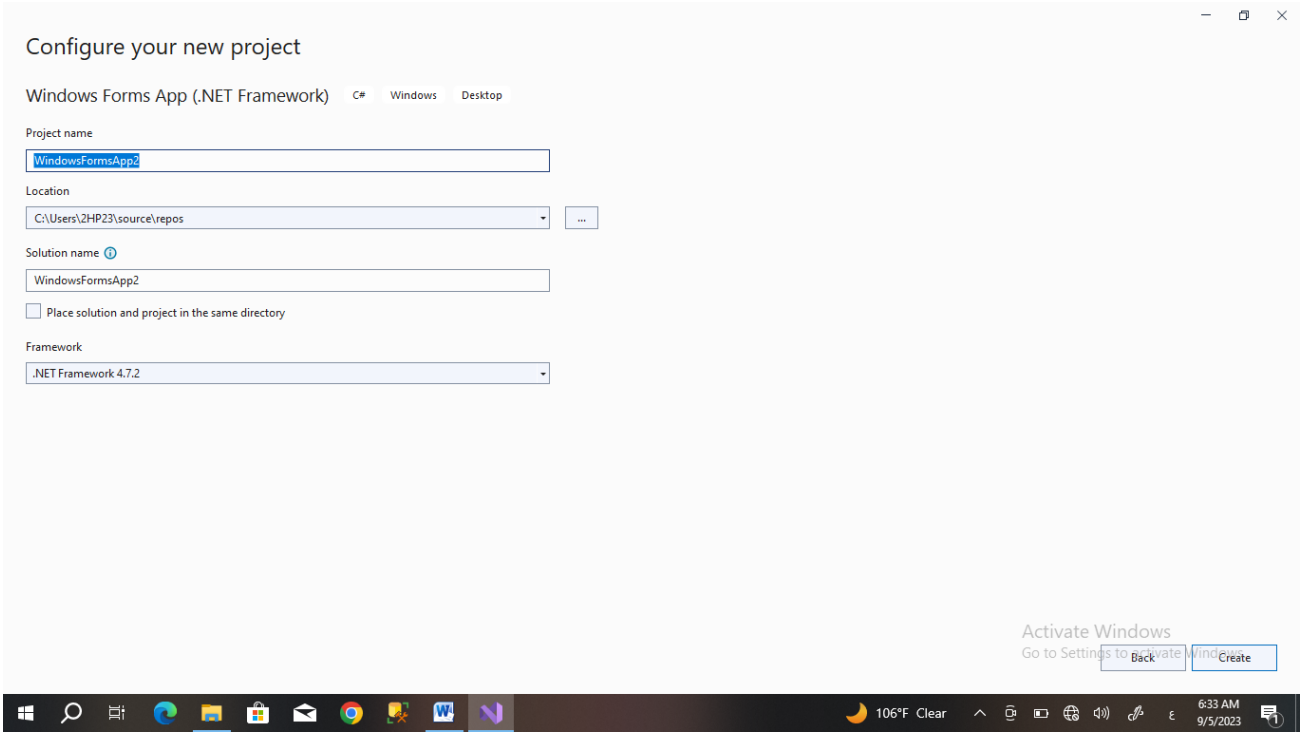
شكل رقم (٨)

ب- ثم نختار windows forms app لاحظ الشكل (٩)



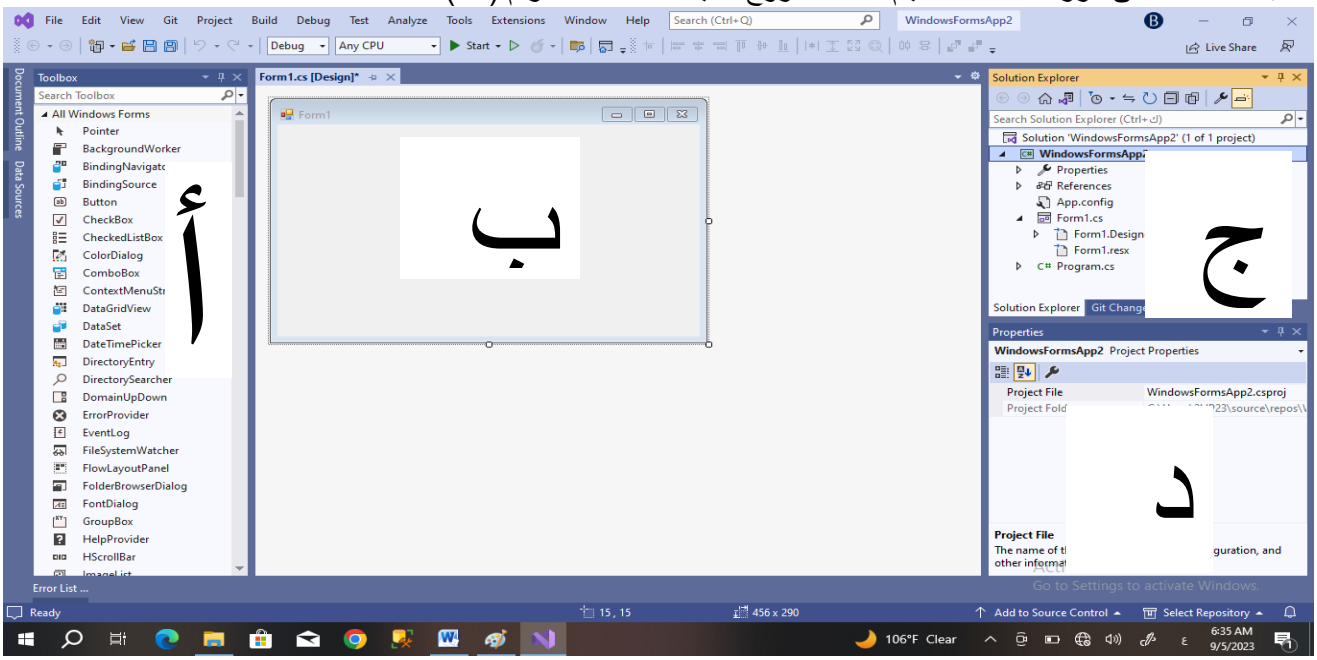
شكل رقم (٩)

ج. ثم يتم اعطاء اسم للمشروع وكذلك مكان حفظه لاحظ الشكل رقم (١٠)



شكل رقم (١٠)

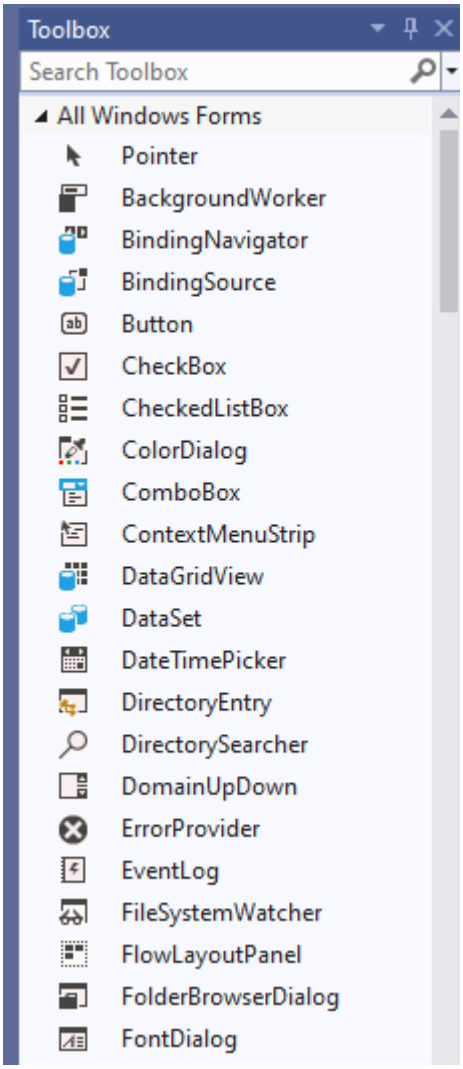
د . بعد الضغط على الزر create يتم انشاء مشروع جديد لا حظ الشكل رقم (١١)



شكل رقم (١١)

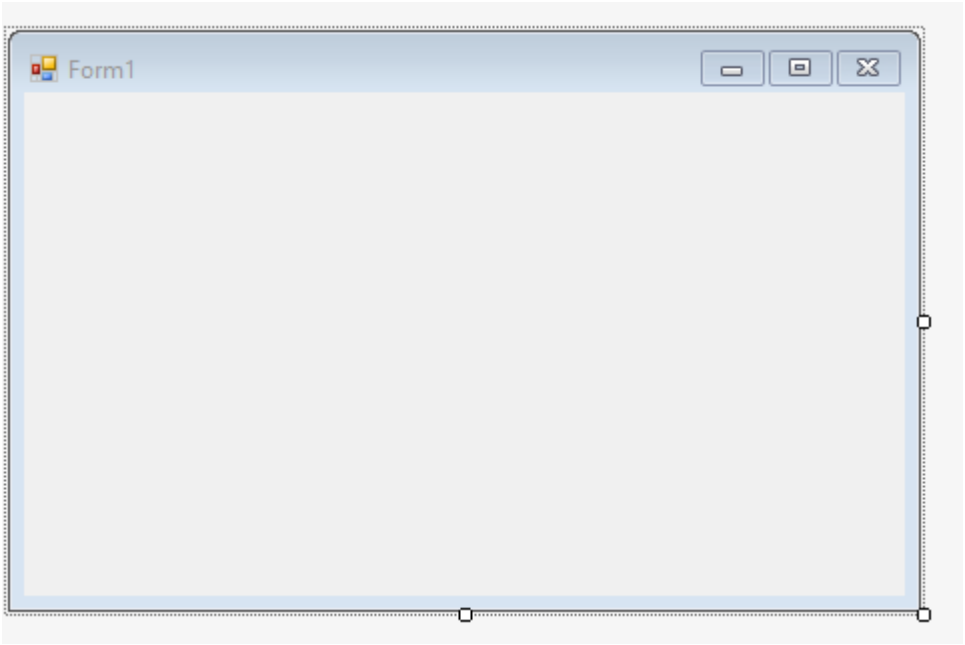
سيتم شرح لكل جزء من بيئة التصميم حسب الارقام الواردة في الصورة .

أ. تسمى هذه النافذة بعلبة الادوات (Toolbox) هي تضم كل الادوات التي يحتاجها البرنامج من (ازرار وقوائم). لاحظ الشكل رقم (١٢)



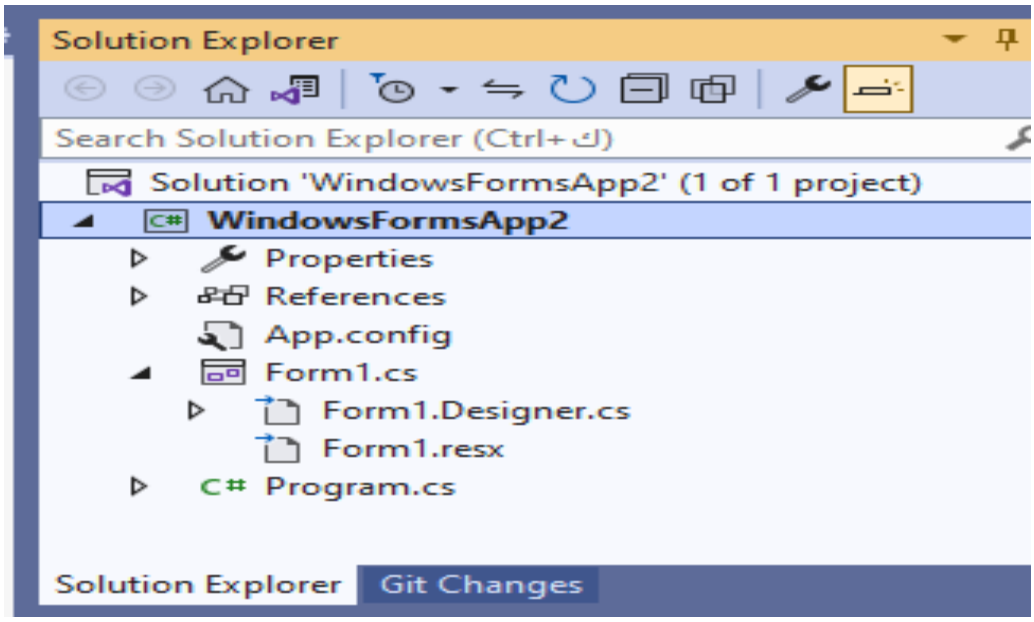
شكل رقم (١٢)

ب.الفورم (From) الذي سيتم وضع عليه الادوات اللازمة لبناء المشروع ويمكن اضافة العديد من الفورمات الى المشروع . كما في الشكل رقم (١٣)



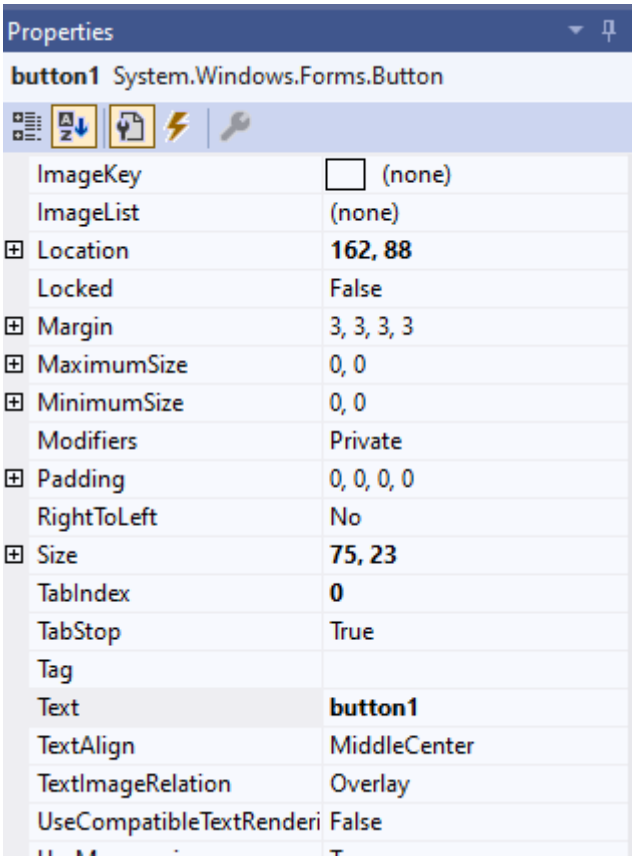
شكل رقم (١٣)

ج. يسمى هذا الجزء متصفح المشروع (Solution Explorer) يعرض كل ملفات المشروع كما في الشكل رقم (١٤)



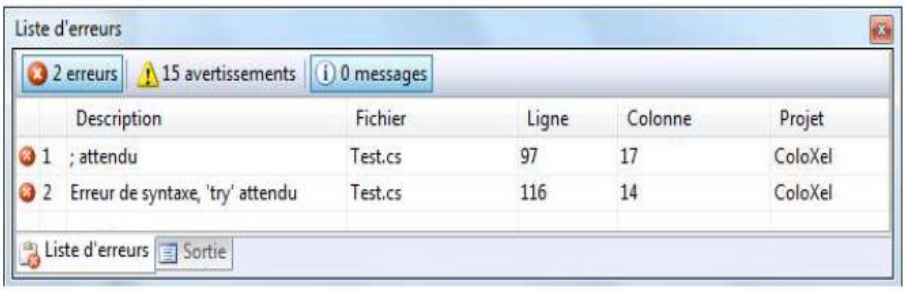
شكل رقم (١٤)

د. نافذه الخصائص (properties) وتحتوي على الخصائص التي يتم تحديدها من خلال هذه النافذة يمكن تغيير لون الخلفية والخط وغير الخصائص الكثيرة كما في الشكل رقم (١٥) .



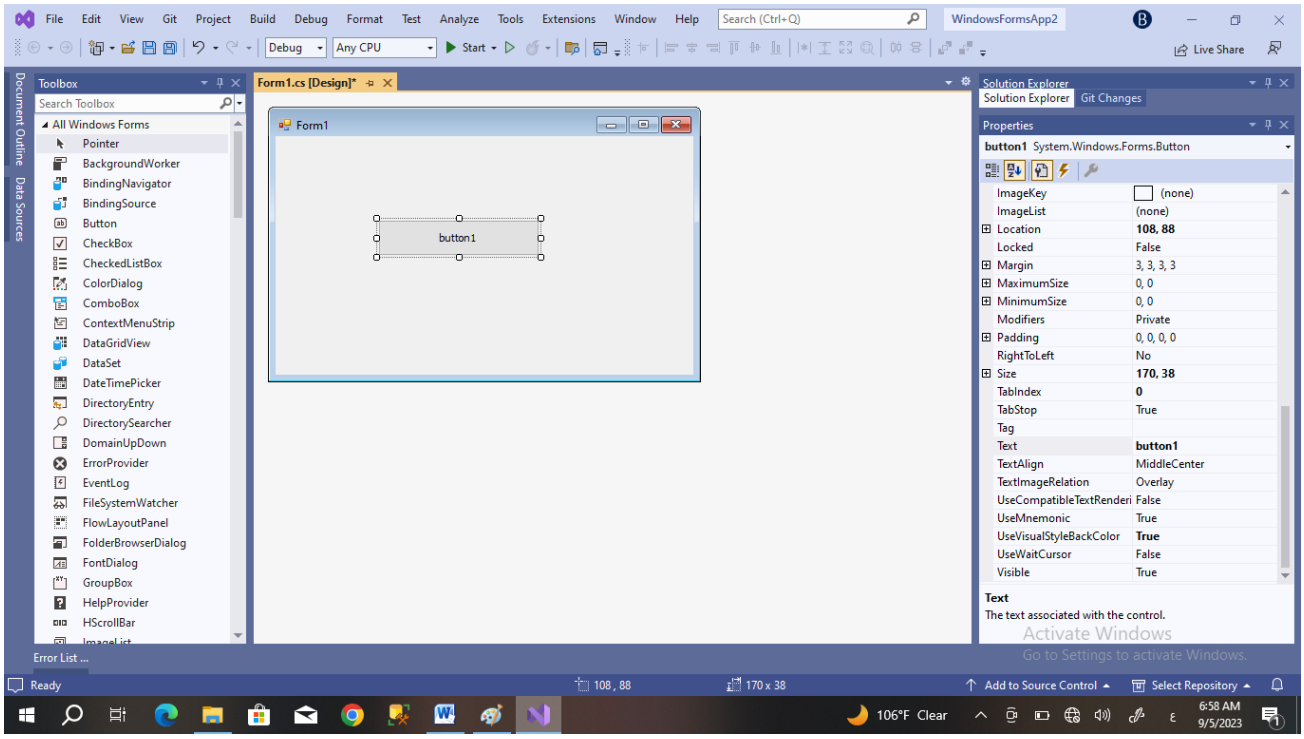
شكل رقم (١٥)

٥.قائمة الاخطاء (error list) وتعرض هذه النافذة الاخطاء الموجودة في الكود كما في الشكل رقم (١٦)



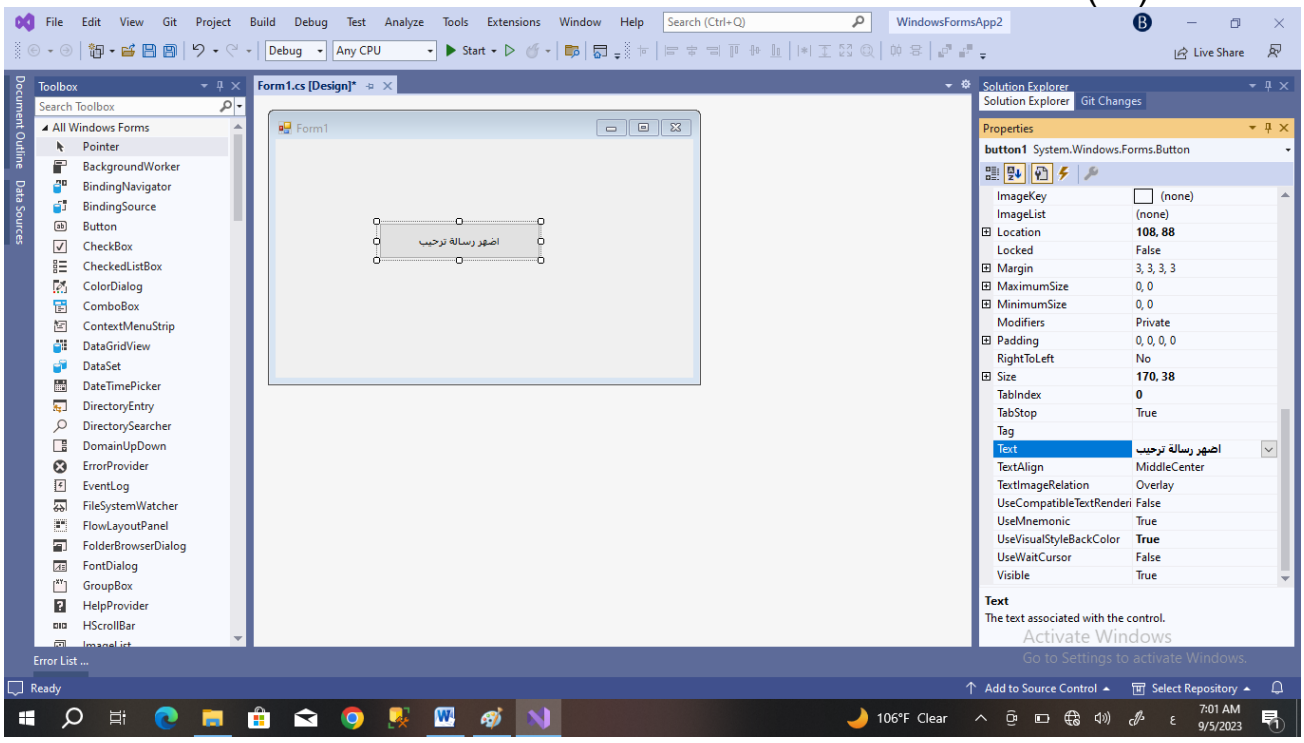
شكل رقم (١٦)

مثال ٦٤/ عمل برنامج باستخدام الواجهات عند الضغط على الزر يظهر رسالة ترحيب
من الادوات نسحب الاداة Button لاحظ الشكل (١٧)



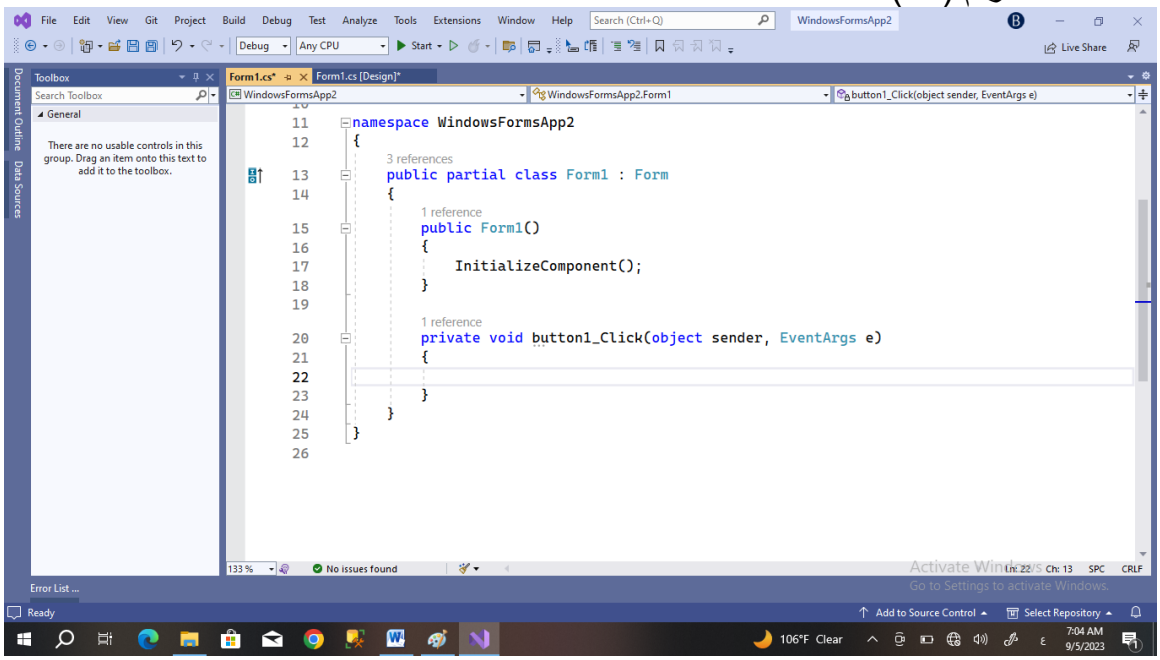
شكل رقم (١٧)

أ- يتم تغير تسمية Button عن طريق الذهاب الى properties يتم تغيرها عن طريق text لاحظ الشكل (١٨)



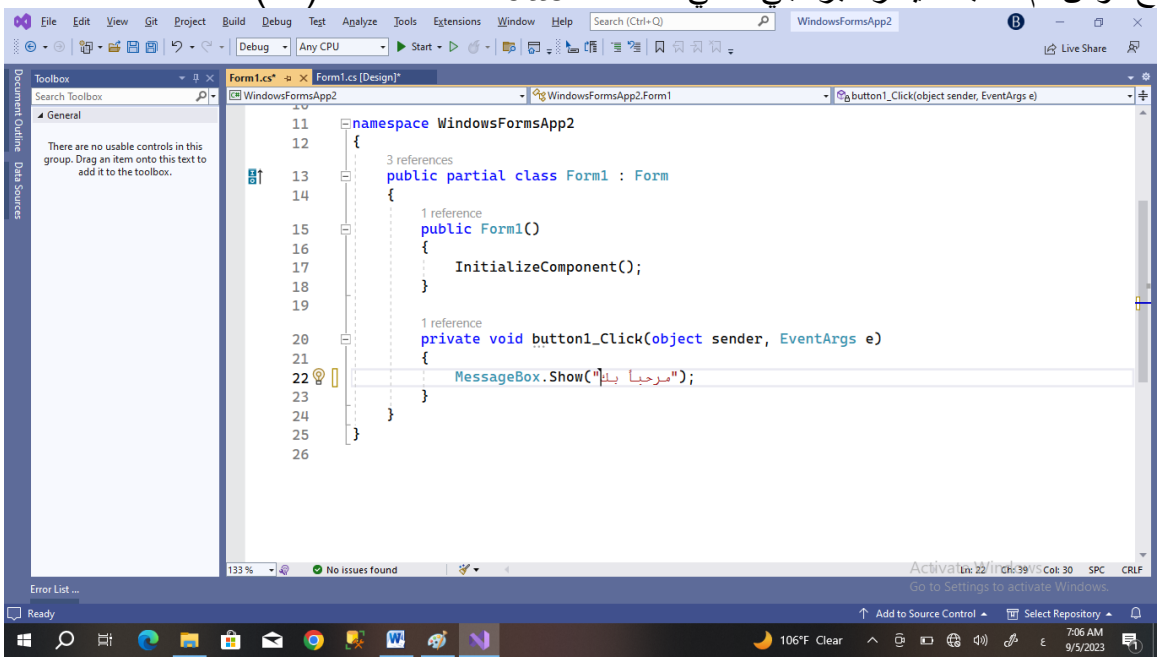
شكل رقم (١٨)

ب-بعد الانتهاء من التصميم نضغط على Button مرتين يتم الدخول الى واجهة كتابة الايعازات البرمجية لاحظ الشكل رقم (١٩)



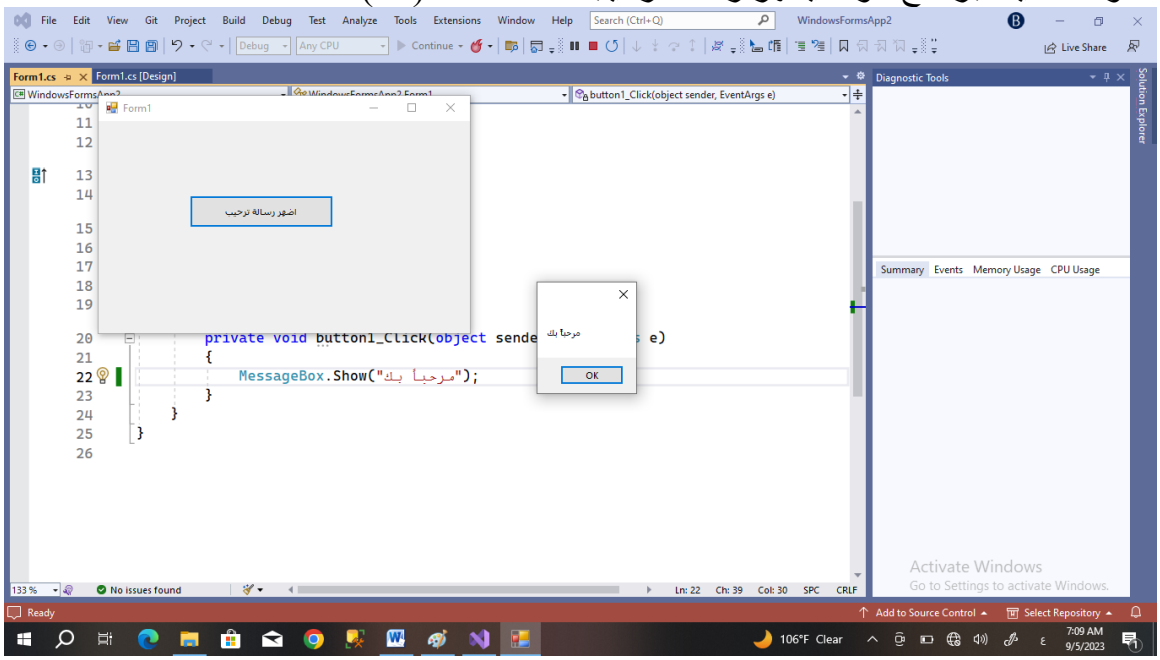
شكل رقم (١٩)

ج- ومن ثم نكتب الايعاز البرمجي التالي داخل Button لاحظ الشكل (٢٠)



شكل رقم (١٩)

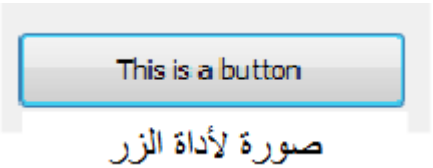
د- وعند تنفيذ البرنامج سوف يظهر رسالة ترحيب لاحظ الشكل (٢٠)



شكل رقم (٢٠)

سيتم الشرح بعض الادوات المهمة :

٥٣. اداة الزر (Button) : وهو عبارة عن اداة تستعمل للقيام بعمل ما عند الضغط عليها تماما مثل زر التلفاز او زر الجرس الذي يتم الضغط عليه ليبدئ بالرنين وتعد هذه الاداة من اهم الادوات لاحظ الشكل (٢١) .



صورة لأداة الزر

شكل رقم (٢١)

٥٤.علبة النص (TextBox): هذه الاداة لا تقل اهمية عن (Button) بحيث توجد في اغلب البرامج وهي عبارة عن اداة لأدخال النصوص

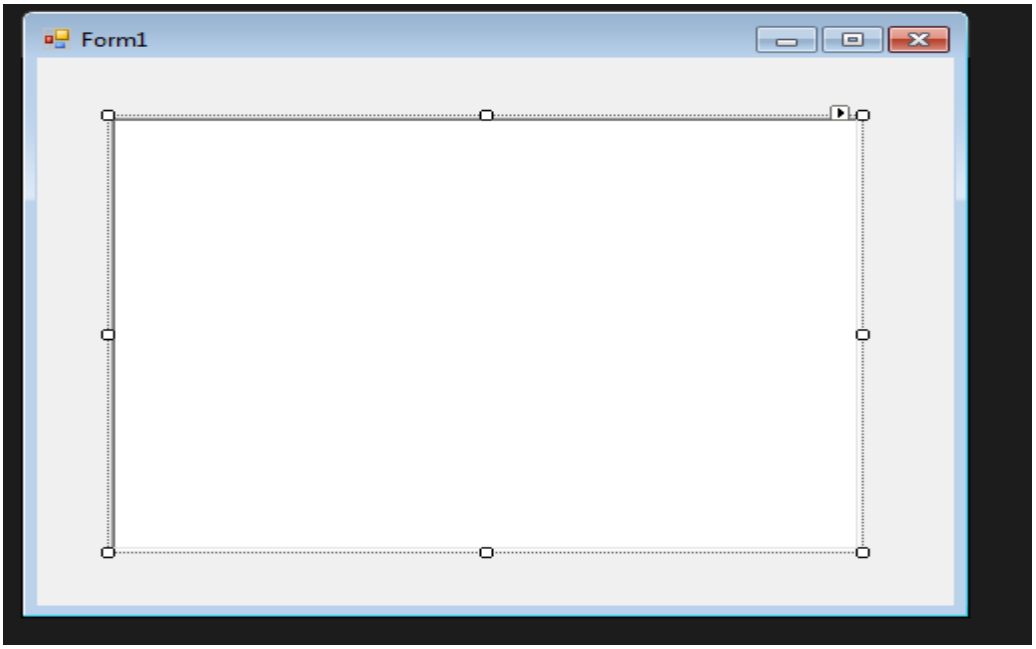
مثال ٦٥/ مثال يتم استخدام فيه (TextBox), (Button) وعند الضغط على (Button) تظهر رسالة ترحيبية مع الاسم (لاحظ الشكل رقم ٢٢) .

شكل رقم (٢٢)

ثم نقوم بكتابة الايعاز البرمجي التالي داخل Button

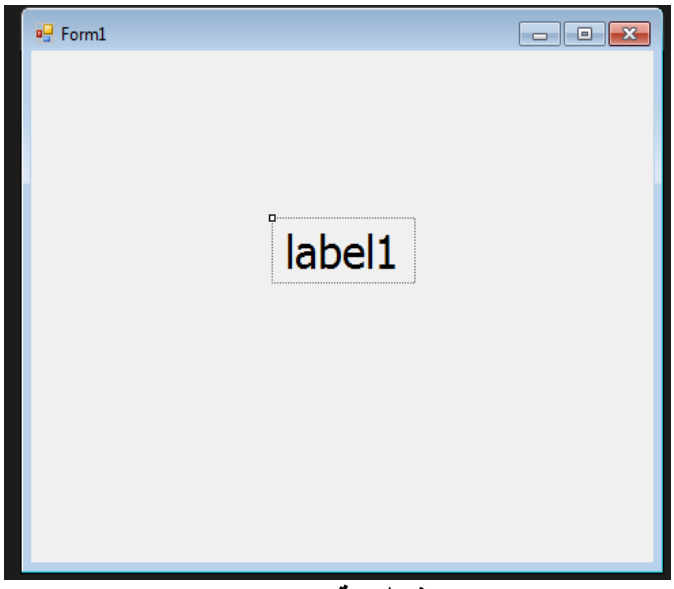
```
private void button1_Click(object sender, EventArgs e)
{
MessageBox.Show("Hellow : " + textBox1.Text);
}
```

٥٥.علبة النص الغنية (Rich textbox): هذه الاداة شبيه (textbox) كونها تسطيع إظهار النص الا انها تتميز بمزايا تنعدم في الاولى وابرز هذه المزايا احتوائها على النص متعدد الالوان والخطوط والحجم . تستعمل هذه الاداة في برامج معالجة النصوص كبرنامج الورد (word) الشهير لاحظ الشكل (٢٠)



شكل رقم (٢٣)

٥٦. ادوات اظهار النص (Label): وهي الاداة التي تلعب دور الملصق الذي يكون مطبوعاً على الملابس او على الاجهزة بغرض تعريفها دورها اظهار النص فقط لاحظ الشكل رقم (٢٤) .



شكل رقم (٢٤)

٥٧.علبة الاختيار Checkbox: لاحظ الشكل (٢٥)

شكل رقم (٢٥)

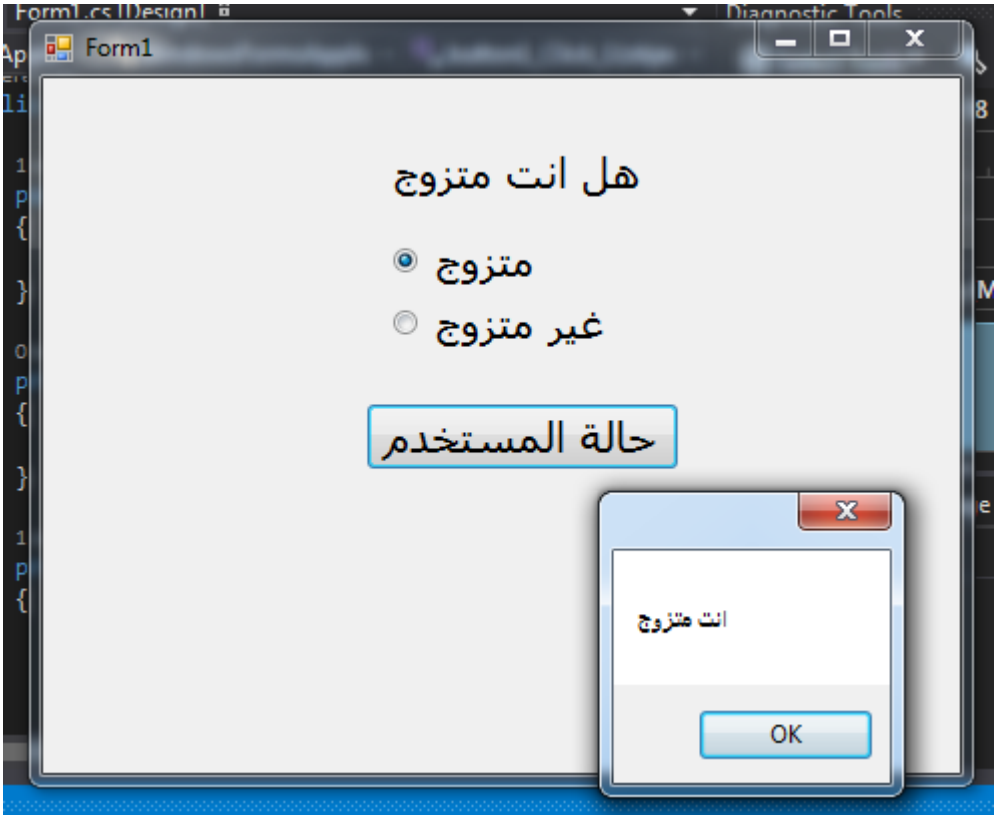
هذه الاداة تسمح للمستخدم بالاختيار المتعدد وتوجد هذه في اكثر من برنامج هذه الاداة تقبل حالتين لا ثالث لها ويمكن اضافة حالة ثالثة اذا قمنا بتغير الخصيصة (Three) (state) الى (True)

٥٨.ازرار الاختيار(Radio Button) : لاحظ الشكل (٢٦)

شكل رقم (٢٦)

هذه الاداة شبيه بالاداة (check box) لكنها تمتاز عنها لا تتيح للمستخدم امكانية تعدد الاختيارات بحيث يصير له حق الاختيار الواحد

مثال ٦٦/ تحقق من المستخدم هل متزوج ام لا ؟
يتم عمل واجهة فيها (check box), Button لاحظ الشكل (٢٧) .



شكل رقم (٢٧)

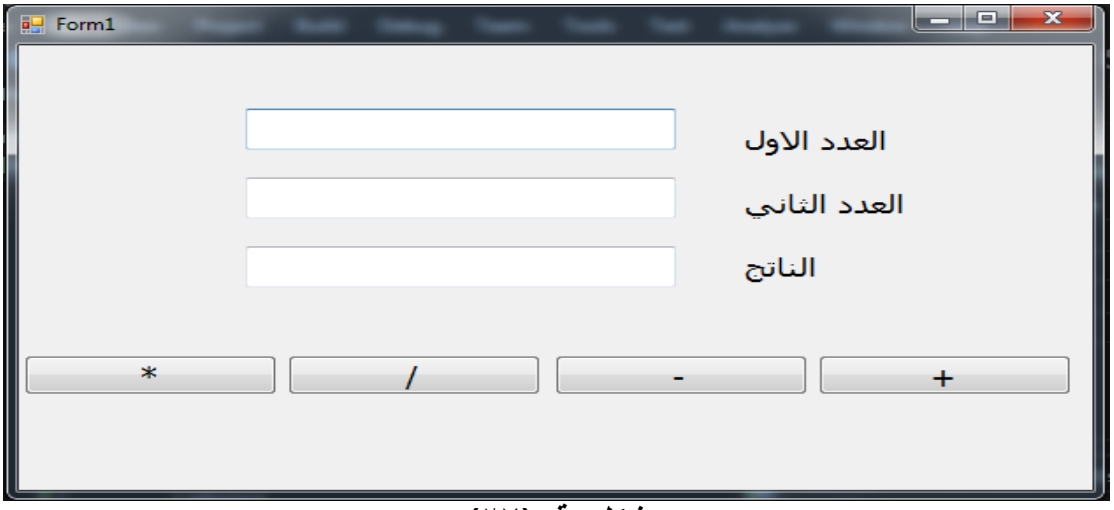
نكتب الايعاز البرمجي التالي داخل الـ (Button)

```
private void button1_Click_1(object sender, EventArgs e)
{
if (radioButton1.Checked == true)
{
MessageBox.Show("متزوج انت");
}
else
{
MessageBox.Show("متزوج غير انت");

}
}
```

مثال ٦٧/ عمل حاسبة الكترونية تحتوي على عملية الجمع والطرح والقسمة والضرب (كما موضح في الشكل رقم (٢٥))؟

الحل/



شكل رقم (٢٧)

أ. في البداية نعمل مثل الفورمة اعلاه متكونة من (٣) label (٣) textbox (٤) button

ب. نكتب في داخل كل buttonالكود الخاص به

أولا. عملية الجمع

```
int x = Convert.ToInt32( textBox1.Text);
int y = Convert.ToInt32(textBox2.Text);
int z;
z = x + y;
textBox3.Text = z.ToString();
```

ثانيا. عملية الطرح

```
int x =
Convert.ToInt32(textBox1.Text);
int y =
Convert.ToInt32(textBox2.Text);
int z;
z = x - y;
textBox3.Text = z.ToString();
```

ثالثا. عملية القسمة

```
int x =
Convert.ToInt32(textBox1.Text);
int y =
Convert.ToInt32(textBox2.Text);
int z;
z = x / y;
textBox3.Text = z.ToString();
```

رابعا. عملية الضرب

```
int x = Convert.ToInt32(textBox1.Text);
int y = Convert.ToInt32(textBox2.Text);
int z;
z = x* y;
textBox3.Text = z.ToString();
```

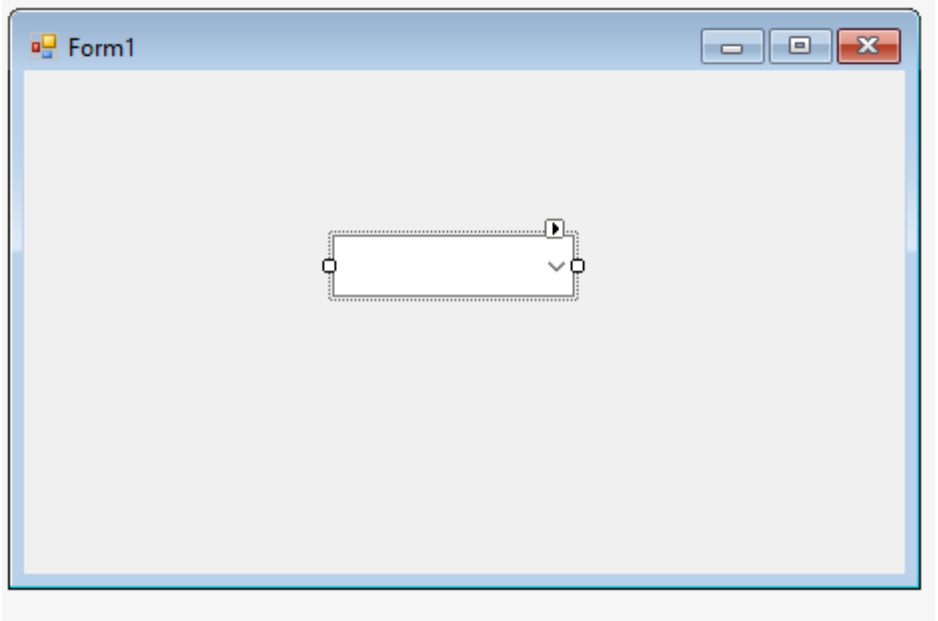
٥٩.علبة الكومبو (Combo box): وهي أداة تستخدم لاظهار القوائم المنسدلة لاحظ الشكل رقم (٢٨)

شكل رقم (٢٨)

هذه الاداة تمكن المستخدم من اختيار بعض البيانات مثلاً حينما القيام بالتسجيل في الفيسبوك او أي موقع اخر فانه يطلب منك الدولة التي تنتمي اليها يتم املائها يدويا عن طريق (Items) او عن طريق الدالة (add range) .

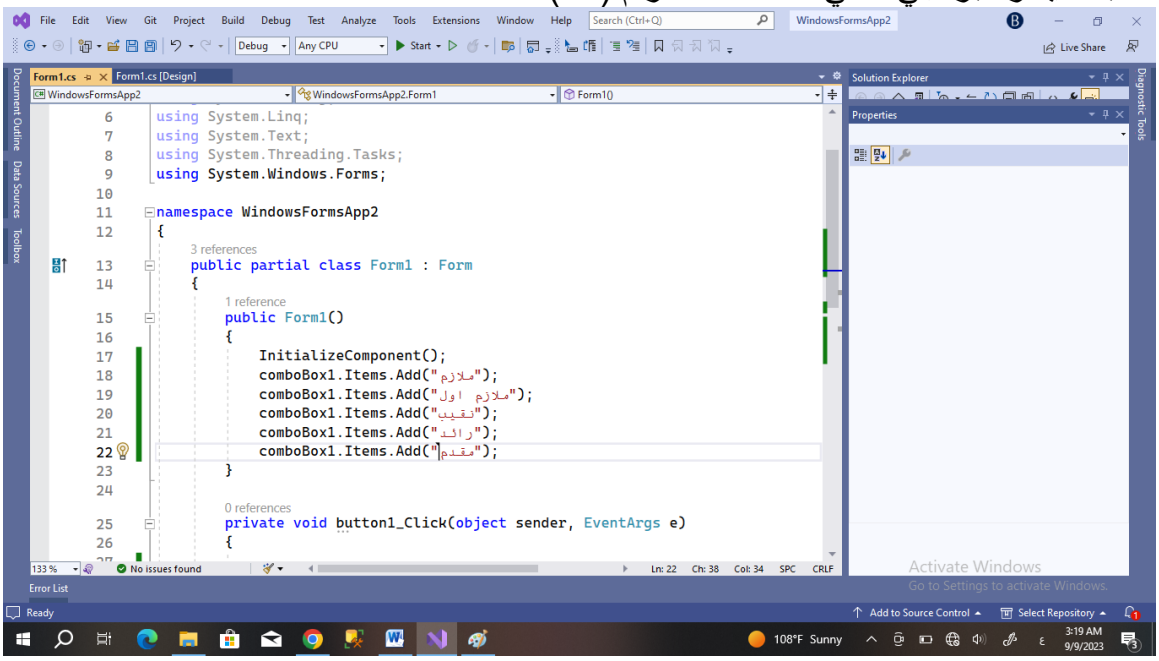
مثال ٦٨/ عمل (Combo box) واملاءه يدويا ؟

أ- يتم عمل واجهة ووضع عليها Combo box لاحظ الشكل (٢٩)



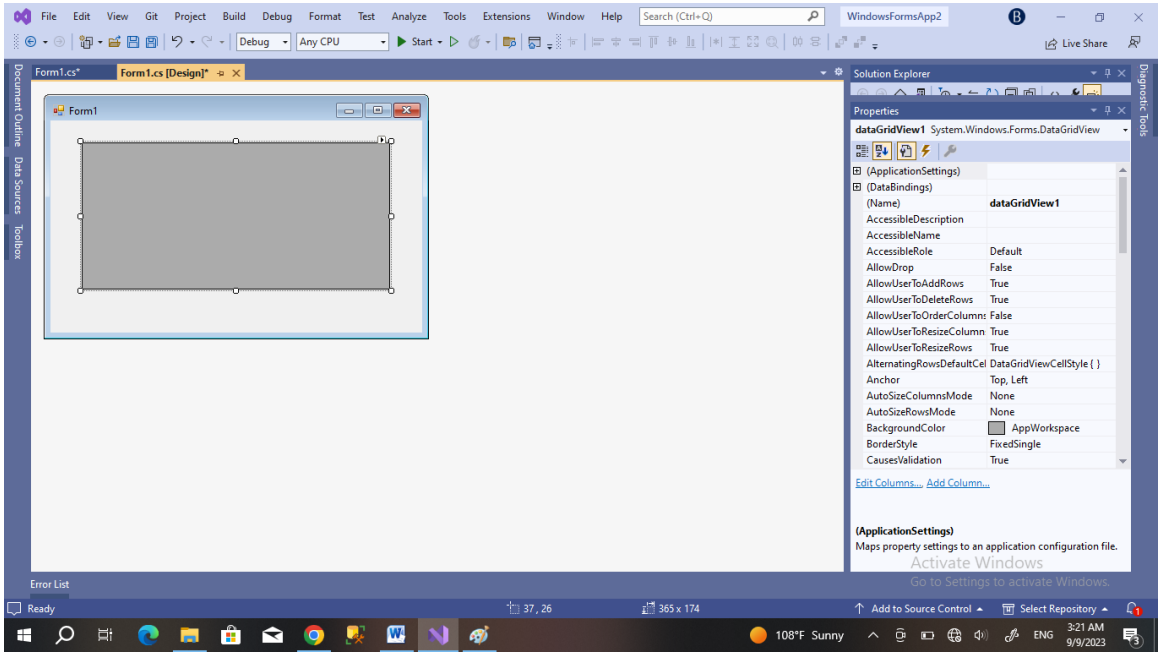
شكل رقم (٢٩)

ب- نكتب الايعاز البرمجي التالي لاحظ الشكل رقم (٣٠)



شكل رقم (٣٠)

٦٠. قائمة عرض البيانات (DataGridView): تستعمل هذه الاداة لعرض معلومات قادمة من قاعدة البيانات هي شبيه بالاكسل حيث تحتوي على خلايا واعمدة الجزء الاهم في قواعد البيانات لاحظ الشكل (٣١)

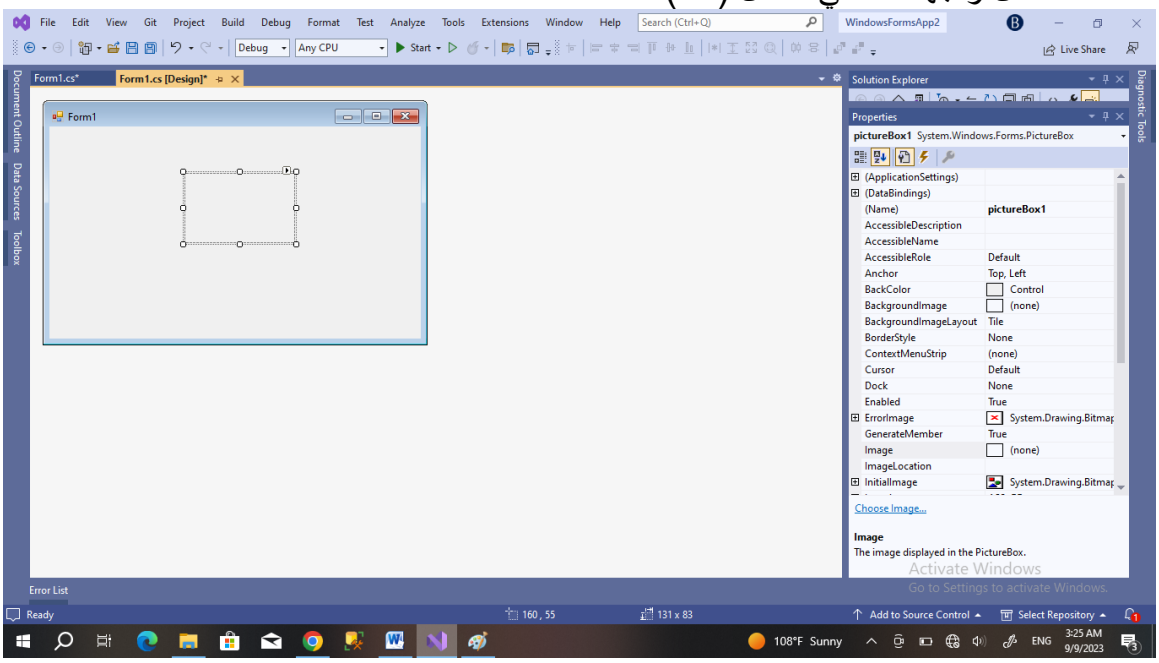


شكل رقم (٣١)

٦١.اداة التجميع (Group Box): وتستعمل هذه الاداة لتجميع الادوات داخل مربع وغالبا ما تستعمل مع (Radio Button) وتضيف الى الفورمة لمسه جمالية لاحظ الشكل رقم (٣٢)

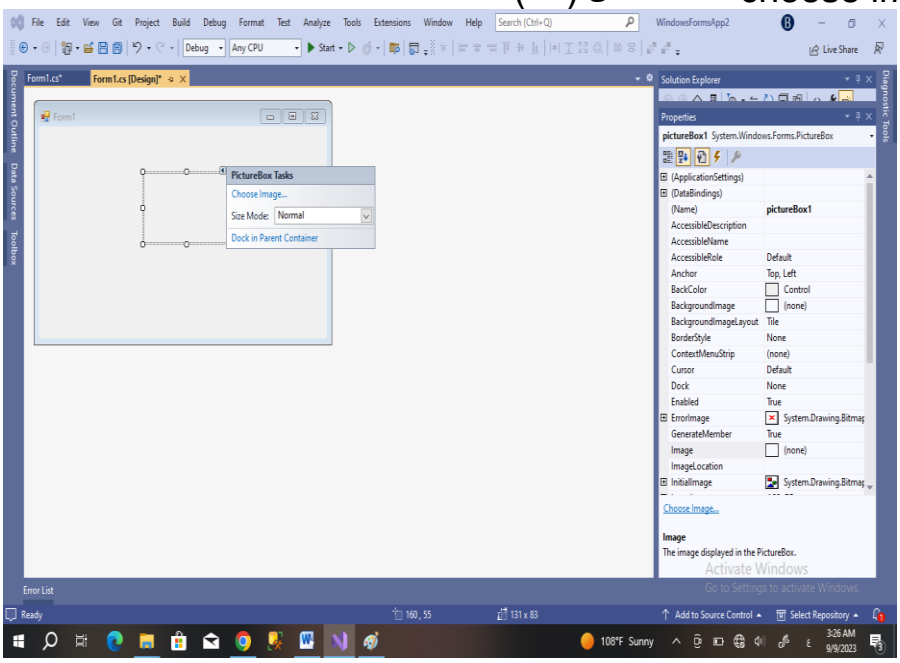
شكل رقم (٣٢)

٦٢. **علبة الصور Picture box** : تستعمل هذه الاداة لأظهار الصور .
مثال ٦٩/ اضافة صورة عن طريق الاداة Picture box ؟
أ- نعمل واجهة كما في الشكل (٣٣)



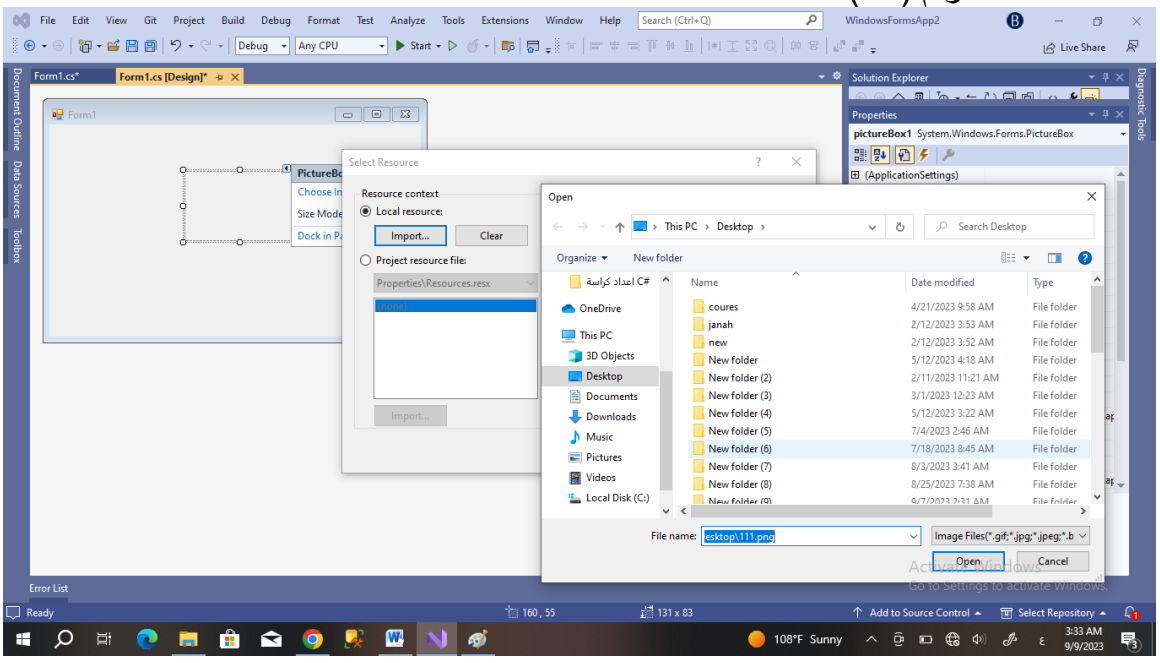
شكل رقم (٣٣)

ب- نضغط choese image لاحظ الشكل (٣٤)



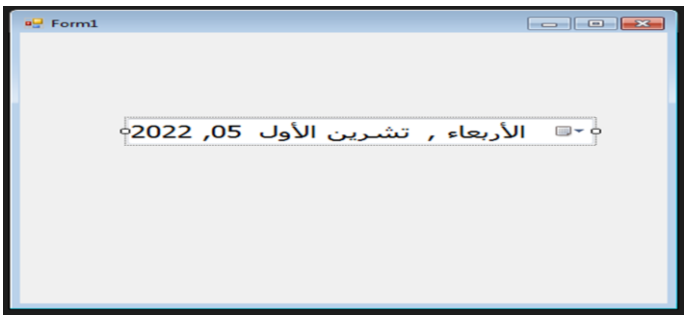
شكل رقم (٣٤)

ج- نضغط على Loacl resource ثم نضغط Import ونختار الصورة المطلوبة لاحظ الشكل رقم (٣٥)



شكل رقم (٣٥)

٦٣.اداة التاريخ **Date Time picker** : تتيح هذه الاداة للمستخدم اختيار تاريخ معين وتكون قيمتها الافتراضية تاريخ اليوم لاحظ الشكل (٣٦).

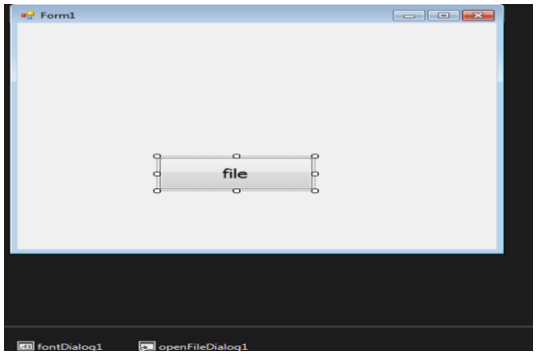


شكل رقم (٣٦)

وحين الضغط على السهم سيظهر تفاصيل الايام للشهر الحالي ويمكن التنقل بين الاشهر لاحظ الشكل (٣٧) .

شكل رقم (٣٧)

٦٤.اداة فتح الملفات **(Open File Dialog)**: تستخدم هذه الاداة لفتح ملف او مجموعة من الملفات القيام بأضافة (Open File Dialog) واضافة زر لفتح الأداة لاحظ الشكل رقم (٣٨)

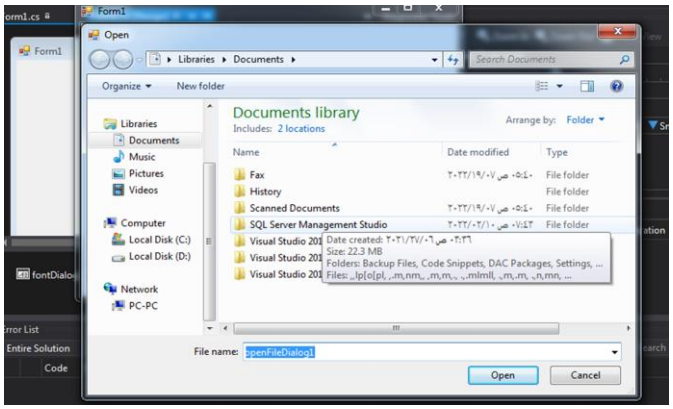


شكل رقم (٣٨)

والدخول الى الزر وكتابة الايعاز البرمجي

```
openFileDialog1.ShowDialog();
```

ثم التنفيذ لاحظ الشكل رقم (٣٩)



شكل رقم (٣٩)