جامعة العث
الكلية التطبيقية
قسم تقنيات الحاسوب

برمجة متقدمة
السنة الثانية
عملي
محاضرة أولى

م. زكي نقولا م. بتول الليوس م. سارة معروف

مقدمة إلى لغة البرمجة#C

لغة البرمجة #C هي واحدة من لغات البرمجة الأكثر استخدامًا في عالم تطوير البرمجيات. تم تطويرها من قبل شركة مايكروسوفت وأطلقت لأول مرة في عام 2000 كجزء من منصة تطوير البرمجيات .NET.تم تصميم #C بغرض تطوير تطبيقات الويندوز وتطبيقات الويب وتطبيقات الألعاب والتطبيقات المتنقلة وغيرها من التطبيقات.

خصائص لغة البرمجة#C

- **بنية اللغة #C :**تعتمد على بنية برمجية مألوفة ومنظمة. تتبع #C هيكلًا شبيهًا بـ C و ++C و Java، مما يجعلها سهلة التعلم والفهم للمطورين الذين لديهم خبرة في هذه اللغات.
- **سكونية التنميط :**تُحدد أنواع المتغيرات في سي شارب قبل استخدامها، مما يساعد على منع الأخطاء وتجعل التعليمات البرمجية أكثر قابلية للقراءة والصيانة.
- **أمرية :**تُنفذ تعليمات سي شارب في تسلسل محدد، مما يجعلها سهلة الفهم والتنبؤ بسلوكها.
- **تعريفية :**تدعم سي شارب البرمجة المعنوية، مما يسمح للمطورين بكتابة التعليمات البرمجية بطريقة أكثر وضوحًا ووصفية.
- **وظيفية :**تدعم سي شارب البرمجة الوظيفية، مما يسمح للمطورين بكتابة التعليمات البرمجية بطريقة أكثر وضوحًا وكفاءة.
- **كائنية المنحى :**تدعم سي شارب البرمجة الشيئية، مما يسمح للمطورين بإنشاء برامج معقدة قابلة لإعادة الاستخدام والصيانة.
- **عامة الأغراض :**يمكن استخدام سي شارب لإنشاء مجموعة واسعة من التطبيقات، بدءًا من تطبيقات الويب وتطبيقات سطح المكتب، وصولًا إلى الألعاب والبرامج المضمنة.

مميزات لغة البرمجة #C سي شارب:

- **سهلة التعلم والاستخدام :**تشبه سي شارب لغات برمجة أخرى شائعة مثل C++ وJava ، مما يجعلها سهلة التعلم للمطورين الذين لديهم خبرة سابقة في هذه اللغات، توفر #C بنية برمجية قوية وسهلة الاستخدام، مما يسمح للمطورين بتطوير التطبيقات بشكل سريع وفعال. بفضل توفر العديد من المكتبات القياسية والأدوات التطويرية المتقدمة، يمكن للمطورين البدء في كتابة الكود بسرعة.
- **قوية ومتعددة الاستخدامات :**يمكن استخدام سي شارب لإنشاء مجموعة واسعة من التطبيقات، مما يجعلها لغة برمجة مثالية للمطورين الذين يرغبون في العمل على مشاريع متنوعة.
- **مُدارة بواسطة مايكروسوفت :**تدعم مايكروسوفت سي شارب بشكل قوي، مما يعني وجود العديد من الموارد المتاحة للمطورين، مثل وثائق شاملة وأدوات تطوير قوية.
- **مجتمع كبير ونشط :**يوجد لدى سي شارب مجتمع كبير ونشط من المطورين، مما يعني أنه يمكن للمطورين الحصول على المساعدة والدعم بسهولة عندما يواجهون مشاكل.
- **دعم منصة :.NET** سي شارب تعتمد على منصةNET. ، وهي بيئة تطوير قوية تقدم العديد من الخدمات والمكتبات القياسية لتطوير التطبيقات. يسمح لك استخدام #C بالوصول إلى مجموعة واسعة من الميزات مثل إدارة الذاكرة والتعامل مع البيانات والتواصل مع قواعد البيانات بسهولة.
- **دعم متعدد المنصات :**بالإضافة إلى تطوير تطبيقات الويندوز، يمكن استخدام #C لتطوير تطبيقات متعددة المنصات، بما في ذلك تطبيقات الويب والألعاب والتطبيقات المحمولة. هذا يتيح للمطورين القدرة على بناء تجارب متنوعة للمستخدمين عبر مختلف الأجهزة.
- **أدوات التطوير المتقدمة :**توفر مايكروسوفت ومجتمع المطورين العديد من الأدوات المتقدمة لتطوير التطبيقات باستخدام .#C من بين هذه الأدوات Visual Studio و Visual Studio Code، التي توفر بيئات تطوير متكاملة وقوية تساعد المطورين على كتابة واختبار وتصحيح البرمجيات بشكل فعال.

الاستخدامات الشائعة للغة البرمجة سي شارب#C

تطبيقات الويندوز

تطبيقات الوي:

تطبيقات المحمول

تطبيقات الألعاب

البرامج المضمنة

```
using System;


namespace HelloWorld
{
 class Program
 {
  static void Main(string[] args)
  {
   Console.WriteLine("Hello World!");
  }
 }
}
```

Line 1: using System means that we can use classes from the System namespace.

Line 2: A blank line. C# ignores white space. However, multiple lines makes the code more readable.

Line 3: namespace is used to organize your code, and it is a container for classes and other namespaces.

Line 4: The curly braces {} marks the beginning and the end of a block of code.

Line 5: class is a container for data and methods, which brings functionality to your program. Every line of code that runs in C# must be inside a class. In our example, we named the class Program.

Line 7: Another thing that always appear in a C# program is the Main method. Any code inside its curly brackets {} will be executed. You don't have to understand the keywords before and after Main. You will get to know them bit by bit while reading this tutorial.

Line 9: Console is a class of the System namespace, which has a WriteLine() method that is used to output/print text. In our example, it will output "Hello World!".

If you omit the using System line, you would have to write System.Console.WriteLine() to print/output text.

Note: Every C# statement ends with a semicolon ;.

Note: C# is case-sensitive; "MyClass" and "myclass" have different meaning.

C# Output

To output values or print text in C#, you can use the WriteLine() method:

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

You can add as many WriteLine() methods as you want. Note that it will add a new line for each method:

```
Console.WriteLine("Hello World!");
Console.WriteLine("I am Learning C#");
Console.WriteLine("It is awesome!");
```

You can also output numbers, and perform mathematical calculations:

```
Console.WriteLine(3 + 3);
```

The Write Method

There is also a Write() method, which is similar to WriteLine().

The only difference is that it does not insert a new line at the end of the output:

```
Console.Write("Hello World! ");
Console.Write("I will print on the same line.");
```

Single-line Comments

Single-line comments start with two forward slashes (//).

Any text between // and the end of the line is ignored by C# (will not be executed).

C# Multi-line Comments

Multi-line comments start with /* and ends with */.

Any text between /* and */ will be ignored by C#.

This example uses a multi-line comment (a comment block) to explain the code:

C# Variables

- int - stores integers (whole numbers), without decimals, such as 123 or -123
- double - stores floating point numbers, with decimals, such as 19.99 or -19.99
- char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- string - stores text, such as "Hello World". String values are surrounded by double quotes
- bool - stores values with two states: true or false

Constants

If you don't want others (or yourself) to overwrite existing values, you can add the const keyword in front of the variable type.

This will declare the variable as "constant", which means unchangeable and read-only

Display Variables

```
string name = "John";

Console.WriteLine("Hello " + name);


string firstName = "John ";

string lastName = "Doe";

string fullName = firstName + lastName;

Console.WriteLine(fullName);
```

The general rules for naming variables are:

- Names can contain letters, digits and the underscore character (_)
- Names must begin with a letter or underscore
- Names should start with a lowercase letter, and cannot contain whitespace
- Names are case-sensitive ("myVar" and "myvar" are different variables)
- Reserved words (like C# keywords, such as int or double) cannot be used as names

Get User Input

Console.ReadLine()

```
// Type your username and press enter
```

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

// Create a string variable and get user input from the keyboard and store it in the variable
string userName = Console.ReadLine();

// Print the value of the variable (userName), which will display the input value
Console.WriteLine("Username is: " + userName);
```

The Console.ReadLine() method returns a string. Therefore, you cannot get information from another data type, such as int. The following program will cause an error:

```
Console.WriteLine("Enter your age:");
int age = Console.ReadLine();
Console.WriteLine("Your age is: " + age);
```

Cannot implicitly convert type 'string' to 'int'

```
Console.WriteLine("Enter your age:");
int age = Convert.ToInt32(Console.ReadLine());
Console.WriteLine("Your age is: " + age);
```

جملة اتخاذ القرار أو الجملة الشرطية Conditional statement

الجملة الشرطية if...else هي مفهوم أساسي وقوي في لغة البرمجة C#. تسمح هذه الجملة بتحقيق فحص مشروط لقيمة معينة وتنفيذ بعض الكود بناءً على نتيجة الشرط. كما تتيح لنا **الجملة الشرطية if...else** التحكم في تدفق تنفيذ البرنامج بناءً على مجموعة متنوعة من الشروط المحددة.

الجملة الشرطية if...else هي مفهوم أساسي وقوي في لغة البرمجة C#. تسمح هذه الجملة بتحقيق فحص مشروط لقيمة معينة وتنفيذ بعض الكود بناءً على نتيجة الشرط. كما تتيح لنا **الجملة الشرطية if...else** التحكم في تدفق تنفيذ البرنامج بناءً على مجموعة متنوعة من الشروط المحددة.

```
If  ( grade >= 60)
    Console.WriteLine(“passed”);
```

ويمكن كذلك استخدام الصيغة المختصرة **للجملة الشرطية if… else** والتي تكون كالتالي :

```
Console.WriteLine( grade >= 60 ? “passed” : “failed”);
```

وتتألف الصيغة المختصرة **للشرط if… else** من :

- العامل الأول عبارة عن تعبير منطقي يمثل الشرط يتم تقييمه إلى صواب أو خطأ وفي المثال هو " grade >= 60"
- العامل الثاني ما بعد علامة الاستفهام (?) وهي الجملة التي تنفذ إذا كان الشرط محقق "صحيحًا) "جملة (if وهي المثال."Passed "
- العامل الثالث ما بعد النقطتين الرأسيتين (:) وهي الجملة التي تنفذ إذا كان الشرط غير محقق "خاطئ) "جملة (else وهي في المثال."Failed"

باستخدام **الجمل الشرطية if...else**، يمكننا تحقيق تحكم دقيق في سير تنفيذ برامجنا بناءً على شروط محددة. تعد هذه الجملة أداة قوية لاتخاذ القرارات وتنظيم تدفق البرنامج في لغة C# . يمكن استخدامها في مجموعة متنوعة من السيناريوهات، مثل التحقق من صحة البيانات المدخلة، تنفيذ سلوك مختلف بناءً على حالة معينة، أو تحديد القيم الافتراضية للمتغيرات.

تعليمات if…else والـ Block {...}

في C# يقوم الـ Compiler بربط تعليمة برمجية واحد فقط مع جملة if، تلك التي تأتي مباشرة بعد **شرط if** لتكون هي فقط جملة الشرط. وكذلك تعليمة واحد فقط بعد else أي أنه في if…else يقوم بربط تعليمة واحدة فقط بعد if وواحد فقط بعد else ولا يكترث للتعليمات التي تتبعها سواء كانت تتبع للشرط او لا.

في هذه الحالة عندما تريد تحديد مجموعة من التعليمات البرمجية لترتبط بشرط معين نستخدم الأقواس {...}وهو مايعرف بـ Block ، فيسكون هنالك Block لتعليمات if و أخر للتعليمات else لنلاحظ معاً ا النتائج في الأمثلة التالية:

```
int x = 6 , y= 3 ;

    if ( x < y )

        Console.WriteLine(“x > y”);

         Console.WriteLine(“ok”);
```

في هذا المثال لم يتم تحديد اي تعليمات ضمن block لذا سيكون الـ Output في هذه الحاله:

```
ok
```

لنرى كيف يكون الـ Output عند استخدام الـ Block

```
 Int x = 6 , y = 3;

If ( x > y )

{

    Console.WriteLine(“ x > y”);

    Console.WriteLine(“ x != y”);

}

Else
```

```
{
    Console.WriteLine(“ok”);
}
```

Output:

```
x > y

x != y
```

عبارة الاختيار Switch Statement في C#

```
Console.Write(“please enter grade between 1 and 100”)
Int grade = Convert.ToInt32(Console.ReadLine());
If (grade>=90)
  Console.WriteLine(‘A’);
else If (grade>=80)
    Console.WriteLine(‘B’);
else If (grade>=70)
    Console.WriteLine(‘C’);
 else If (grade>=60)
    Console.WriteLine(‘D’);
else
   Console.WriteLine(‘F’);
```

هذه الخوارزمية يمكن تمثيلها بأكثر من طريقة في C# وهذا عن طريق if…else كما نعرف او عن طريق عبارة switch فهذه العبارات تحدد التعليمات التالية المطلوب تنفيذها من عدد من الخيارات المحتملة بناءً على قيمة تعبير ما :

- عبارة **if…else** تحدد بيانًا ليتم تنفيذه بناءً على قيمة التعبير المنطقي
- عبارة **switch** تحدد قائمة جمل لتنفيذها بناءً على تطابق قيمة مع متغير أو تعبير معيين .

لاحظ /ي طريقة التنفيذ في حال if…else في المثال التالي :

في **سي شارب تعمل جملة الـ "Switch"** على المقارنة المتعددة لقيمة معينة مع مجموعة متنوعة من القيم المحتملة. و باستخدام هذه المقارنة، يمكننا تحديد التكوين الصحيح لتنفيذ الكود الموافق لكل قيمة. يتم تحقيق ذلك باستخدام "حالات" (Cases) تحوي كودًا يتم تنفيذه عندما تطابق قيمة معينة.

في **جملة الاختيار switch** يتم اختيار قيمة معينة لتنفيذ حالة من حالات متعددة فتنفذ هذه الإجراءات بناءً على هذه القيمة. هذه القيمة تكون تكون مرجعة من متغير أو تعبير برمجي. **تعبير switch** ينفذ الإجراءات لأول حاله قيمتها مطابقة للقيمة المرجعة ويتجاهل التنفيذ لبقية الحالات ويجب أن تكون القيمة في الحالة مطابقة تماما للقيمة المرجعة من نوع المتغير أو حتى حالة الأحرف فـ Compiler ينشئ خطأ عندما تحتوي عبارة التبديل على حالة غير قابلة للوصول.

```
Console.Write(“please enter grade between 1 and 100”)

Int grade = Convert.ToInt32(Console.ReadLine());

 switch(grade/10){

case 9:

case 10:

Console.WriteLine(‘A’);

break;

case 8:
```

```
Console.WriteLine('B');

break;

case 7:

Console.WriteLine('C');

break;

case 6:

Console.WriteLine('D');

break;

default:

   Console.WriteLine('F');

break;

}
```

الفرق بين عبارة الاختيارSwitch وعبارة الشرط if-else

هناك عدة **اختلافات بين عبارة الاختيار Switch Statement وعبارة الشرط if-else** في لغة C# . فيما يلي بعض الاختلافات الرئيسية بينهما:

- **هيكل الكود:** في عبارة الشرط if-else، يتم استخدام تعبير مشروط واحد (مثل if) يتم تقييمه لتحديد مسار التنفيذ. أما في عبارة الاختيار Switch Statement، يتم تحديد عدة حالات (cases) تستند إلى قيمة متغير التحكم.
- **قيمة المقارنة:** في عبارة الشرط if-else ، يمكن استخدام أي تعبير مشروط يعيد قيمة منطقية (true أو false) لتحديد التنفيذ. أما في عبارة الاختيار Switch Statement ، يتم مقارنة قيمة واحدة فقط بين قيمة متغير التحكم والحالات المحددة.

- **التعقيد والقراءة:** عبارة الشرط if-else تستخدم عادة لاختبار شروط معقدة ومتعددة، حيث يمكن استخدام مجموعة من العبارات المشروطة المتداخلة. بالمقابل، عبارة الاختيار Switch Statement تستخدم لاختبار قيمة واحدة وتوفر هيكلًا أكثر تنظيمًا وسهولة في القراءة.
- **الأداء:** في بعض الحالات، عبارة الاختيار Switch Statement يمكن أن تكون أكثر كفاءة من عبارة الشرط if-else ، خاصة عندما يكون هناك عدد كبير من الحالات. هذا يعود إلى طريقة تنفيذ عبارة الاختيار Switch Statement التي تستخدم جدولًا للانتقال (jump table)، مما يؤدي إلى تنفيذ أسرع للكود.
- **القيود:** عبارة الاختيار Switch Statement تفرض بعض القيود على نوع البيانات التي يمكن استخدامها كقيمة لحالاتها. على سبيل المثال، يجب أن تكون قيمة متغير التحكم من الأنواع القابلة للتحويل (convertible) إلى نوع البيانات المستخدم في كل حالة.

حلقة التكرار While في C#

من **حلقات التكرار** التي تتوفر في لغة سي شارب **حلقة التكرار while** وفيها تستخدم الكلمة المحجوزة **while**للإعلان عن حلقة التكرار ثم الأقواس التي تحتوي شرط الحلقة والصيغة العامة لحقلة تكرار While هي:

```
while ( booleanExpression )

  statement
```

مثال

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

(i <= 10) هو شرط التكرار حيث i هو متغير التحكم هنا و i++ هو التعبير المسؤول عن التغير في قيمة i .

do..whileحلقة التكرار

من **حلقات التكرار** المتوفرة في لغة سي شارب هي **حلقة التكرار do…while** وتستخدم معها الكلمتين المحجوزتين do and while للإعلان عن حلقة تكرار من نوع 'do…while hgav و**الصيغة العامة لحلقة التكرار do…while** تكون:

```
do {

statement

} while ( booleanExpression );
```

وهنا مثال لطريقة تنفيذ حلقة التكرار do…while في لغة C#.

```
 int  i = 0 ;

do{

  Console.WriteLine(i);

 i++;

} while (i <= 10);
```

نلاحظ في المثال ان فى **حلقة التكرار while** يتم التحقق من الشرط قبل تنفيذ تعليمات الحلقة ولكن مع **حلقة التكرار do…while** يتم التحقق من الشرط بعد تنفيذ تعليمات الحلقة لأول دورة للحلقة أي أن في do…while سيتم تنفيذ تعليمات الـ loop لمرة واحدة على الأقل

حلقة التكرار for في C#

وتدعم لغة **سي شارب حلقة التكرار for** التي تعتبر for أكثر حلقات التكرار استخداماً خصوصاً مع المصفوفات Array حيث يمكننا مع for الإعلان عن متغير التحكم عند الإعلان جملة for مما يجعل العمل مع **الحلقات التكرارية** المتداخلة أفضل وتستخدم الكلمة المحجوزة for للإعلان عن حلقة for وتكون جملة for من ثلاث أجزاء يفصل بينها بفاصلة منقوطة (;) وهي:

- لقيمة الاولية لمتغير التحكم (يمكن الاعلان عن متغير داخل او خارج حلقة for ولكن القيمة الأولية لابد تعطى للمتغير عند الإعلان عن الحلقة).
- الشرط الحلقة التكرارية.
- مقدار التغير في قيمة متغير الحلقة.

وتكون **الصيغة العامة للحلقة التكرار for في C#**

```
for (initialization; Boolean expression; update control variable)

statement
```

ويكون **تنفيذ حلقة التكرار for في لغة سي شارب** كتالي :

```
for( i=0 ; i<=10; i++) {

  Console.WriteLine(i);

}
```

عبارتي break و continue مع حلقة التكرار

يمكن التحكم في تنفيذ **حلقات التكرار loops** بواسطة عبارات التحكم
مثل **break** و **continue** التي ستؤثر على في طريقة تنفيذ حلقة التكرار بشكل معين لنرى طريقة تأثير كلا من break و continue على حلقات التكرار.

break

استخدام **عبارة break مع حلقة التكرار** يؤدي لإنهاء تنفيذ التكرار loop والخروج من حلقة التكرار ويوضح المثال التالي طريقة استخدام break مع حلقة التكرار، في المثال سوف ينتهي التنفيذ عند قيمة معينة وينقل التنفيذ إلى التعليمات التي تلي حلقة التكرار loop مباشرةً.لاحظ /ي النتائج في المثال التالي:

```
for( i=0 ; i<=10; i++){

if(i==5)

break;

  Console.WriteLine(“{0}” , i); }
```

لتكن النتيجة

1

2

3

4

continue

استخدام **continue مع حلقة التكرار** يؤدي إلى تخطي التنفيذ لدورة معينة من التكرار مع استكمال التنفيذ للدورات التالية من حلقة التكرار. ويوضح المثال التالي طريقة تأثير استخدام continue مع حلقة التكرار، في المثال سوف يتم تحطي التنفيذ عند قيمة معينة وفي المثال سنقوم بتخطي الدورة رقم 5 من التكرار ومن ثم يتابع تنفيذ حلقة التكرار Loop لاحظ /ي نتيجة المثال التالي :

```
for( i=0 ; i<=10; i++){

if(i==5)

break;

  Console.WriteLine(“{0}” , i); }
```

لتكن النتيجة

1

2

3

4

6

7

8

9

10