مقدمة في البرمجة
Introduction To Programming

IPG101

الفصل السابع
بنى التحكم (التكرار)
Control Structure (Loops)

الكلمات المفتاحية

تدفق، تحكم، تكرار، حلقة، حلقة لا نهائية، while، do while، for، break، continue.

ملخص الفصل

يستكمل هذا الفصل عرض مفهوم التحكم في مسار تنفيذ البرنامج المكتوب بلغة C#، حيث يركز على عمليات التكرار، ويركز على استعراض أساليب استخدام الأوامر الثلاثة للتكرار while، do while، for. كما يلقي الضوء على مفهومي الحلقات المتداخلة والحلقات اللامنتهية وكيفية استخدام أوامر التحكم في العملية التكرارية break و continue.

أهداف الفصل

بنهاية هذا الفصل سيكون الطالب قادراً على:

- فهم أسلوب التحكم بالتدفق في لغة C#.
- وصف العملية التكرارية وتحديد مكوناتها.
- استخدام أوامر التكرار while، do while، for.
- معرفة كيفية بناء حلقات متداخلة.
- التحكم بسلوك الحلقات التكرارية من خلال الأوامر break و continue.

محتويات الفصل

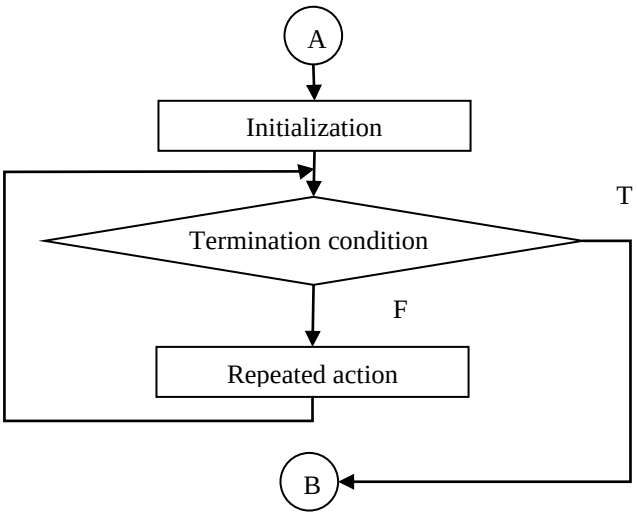
1. مقدمة
2. أمر التكرار while.
3. أمر التكرار do while.
4. أمر التكرار for.
5. تداخل الأوامر التكرارية.
6. الأمر التكراري غير المنتهي.
7. الأوامر break و continue.
8. تمارين وأنشطة.

1- مقدمة.

بتأمل العمليات المتكررة التي يمكن ملاحظتها في حياتنا اليومية نجد أن جميع هذه العمليات تشترك فيما بينها بثلاثة مكونات أساسية وهي:

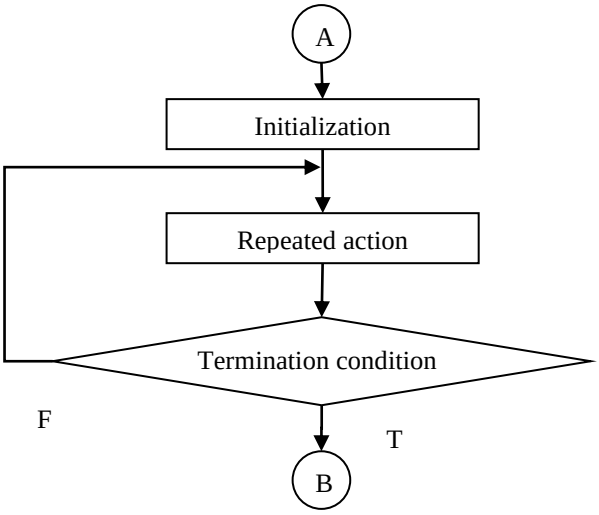
- نقطة بداية initialization، تمثل اللحظة التي يبدأ عندها تكرار العملية.
- شرط انتهاء termination، يمثل الأمر الذي سيؤدي تحققه إلى إيقاف تكرار العملية والانتقال إلى عملية جديدة.
- الأمر المتكرر repeated action، يمثل الفعل أو مجموعة الأفعال التي سيتكرر تنفيذها.

حيث يختلف ترتيب ورود هذه المكونات الثلاثة باختلاف العملية المراد تكرارها، فعلى سبيل المثال، قد نرغب بتكرار تنفيذ فعل (أو مجموعة أفعال) طالما أن سبب انتهائه لم يتحقق، عندها ستنفذ العملية التكرارية بالتتالي التالي (يدعى أحياناً التكرار ذو الشرط السابق):



الشكل 1- المخطط التدفقي للتكرار ذو الشرط السابق

أما إذا رغبنا بتكرار تنفيذ فعل (أو مجموعة أفعال) لحين تحقق سبب انتهائه، عندها ستنفذ العملية التكرارية بالتتالي التالي (يدعى أحياناً التكرار ذو الشرط اللاحق):



الشكل 2- المخطط التدفقي للتكرار ذو الشرط اللاحق.

تحتوي لغة C# ثلاثة أوامر (بنى) قياسية لتنفيذ عمليات التكرار وهذه الأوامر هي:

1- البنية while.

2- البنية do …while.

3- البنية for.

حيث يمكن اعتبار البنيتين while و for بنى تكرارية ذات شرط سابق، أما البنية do .. while فهي ذات شرط لاحق.

2- البنية التكرارية while

تعتبر البنية while من البنى التكرارية ذات الشرط المسبق pretest، وبالتالي فإن مخططها التدفقي يكون من النموذج المبين في الشكل 1 .

الصيغة العامة لهذه البنية تكون على النحو:

while (condition)
statement

حيث:

while : كلمة مفتاحية.

condition : تعبير منطقي.

statement : أمر بسيط أو مركب.

يعمل الأمر while كما يلي:

" **يتم تقييم الشرط condition، فإن كانت قيمته true (الشرط محقق) يتم تنفيذ الفعل المراد للمرة الأولى ومن ثم يتم العودة إلى نقطة تقييم الشرط مجدداً .... وهكذا، إلى أن تصبح قيمته false، عندها يتم إيقاف التكرار ونقل التحكم إلى الأمر التالي**".

ننوه هنا إلى أمرين هامين:

الأول: إن اختبار الشرط يتم قبل بتنفيذ الحلقة ومن هنا جاءت تسميتها بالبنية ذات الشرط المسبق.

الثاني: لابد أن يحتوي الفعل المراد تكراره على أمر يغير في حالة المتحول (أو المتحولات) التي يختبرها شرط التكرار حتى نصل إلى حالة يصبح فيها الشرط غير محقق لإنهاء العملية التكرارية.

مثال

بفرض أنه طلب إلينا كتابة برنامج يقوم بحساب مجموع الأعداد المحصورة بين 1 و 100 (ضمناً) وطباعته على الشاشة.

يكون البرنامج المطلوب من الشكل:

```
int number = 1;
int sum = 0;
while (number <= 100)
 {
        sum += number;
        number++;
  }
  Console.WriteLine("SUM OF NUMBERS FROM 1 to 100 is: "+sum);
```

يمثل الأمر int number=1 نقطة البداية، والشرط number<=100 شرط الاستمرار، والأمر المركب المحصور بين قوسي كتلة { } الفعل المراد تكراره، لاحظ وجود الأمر number++ الذي يغير في حالة المتحول المختبر في شرط البنية.

3- البنية التكرارية do while

بخلاف البنية while، تعتبر البنية do … while بنية التكرارية ذات شرط لاحق posttest، وبالتالي فإن مخططها التدفقي يكون وفق النموذج في الشكل 2.

الصيغة العامة لهذه البنية تكون على النحو:

```
do
   statement
while ( condition ) ;
```

حيث:

while : كلمة مفتاحية.

condition : تعبير منطقي.

statement : أمر بسيط أو مركب.

يعمل الأمر do … while كما يلي:

" يتم تنفيذ الفعل المراد للمرة الأولى ومن تقييم الشرط condition، فإن كانت قيمته true (الشرط محقق) يتم العودة إلى تكرار تنفيذ الفعل لمرة ثانية .... وهكذا، إلى أن تصبح قيمته false، عندها يتم إيقاف التكرار ونقل التحكم إلى الأمر التالي".

وبالمثل ننوه هنا إلى ثلاثة أمور هامة:

الأول: يتم تنفيذ الأمر statement قبل اختبار الشرط ومن هنا جاءت تسميتها بالبنية ذات الشرط اللاحق.

الثاني: كما في حالة البنية while، لابد أن يحتوي الفعل المراد تكراره على أمر يغير في حالة المتحول (أو المتحولات) التي يختبرها شرط التكرار حتى نصل إلى حالة يصبح فيها الشرط غير محقق لإنهاء العملية التكرارية.

الثالث: سيتم تنفيذ الفعل المراد تكراره مرة واحدة على الأقل (حتى لو كان الشرط غير محقق) وذلك بسبب كون شرط الانتهاء يتم اختباره بعد التنفيذ أول مرة.

مثال

بما أن البنيتين while و do … while تملكان المكونات ذاتها ولكن بترتيب مختلف، سنحاول القيام بإعادة ترتيب أوامر حل مسألة حساب مجموع الأعداد المحصورة بين 1 و 100 ذاتها التي رأيناها في الفقرة السابقة بحيث نحقق الحل باستخدام البنية do…while.

يكون البرنامج المطلوب من الشكل:

```
int number = 1;
int sum = 0;
do
{
        sum += number;
        number++;
  }
while (number <= 100) ;
  Console.WriteLine("SUM OF NUMBERS FROM 1 to 100 is: "+sum);
```

ننبه إلى أن من الأخطاء الشائعة لدى استخدام البنية do…while نسيان وضع الفاصلة المنقوطة بعد الشرط في حين أن وضعها في البنية while يؤدي إلى تنفيذ حلقة فارغة. كما يجب الانتباه إلى أنه، ونظراً لكون البنية تنفذ الأمر statement قبل اختبار الشرط فإن كتابة أمر التهيئة في هذه الحالة يكتسب أهمية بالغة، فمثلاً لو كتب أمر التهيئة كما يلي:

```
int number=300;
```

فإن البنية ستنفذ الفعل المراد تكراره مرة واحدة قبل الوصول إلى شرط الانتهاء الذي يؤدي إلى إيقافها، وبالتالي سنحصل على نتيجة خاطئة.

في حين أن مثل هذا الأمر لا يحصل في البنى ذات الشرط السابق في مثل هذه الحالة.

4- البنية التكرارية for

من مزايا البنيتين while و do … while أن عدد المرات التي سيتم تنفيذ الفعل المتكرر ليس بالضرورة أن يكون معلوماً وثابتا دائماً في كل مرة يتم فيها تنفيذ البرنامج، فقد يحصل أن يختلف عدد مرات التنفيذ من مرة لأخرى.

على سبيل المثال، بفرض كان المطلوب تكرار إدخال قيم صحيحة موجبة من لوحة المفاتيح وطباعة مربعها، فإن شكل الحل باستخدام البنية while في هذه الحالة سيكون:

```
int number = Int32.Parse(Console.ReadLine()) ;
while (number>=0)
{
    Console.WriteLine(number*number) ;
    number = Int32.Parse(Console.ReadLine()) ;
}
```

إن عدد مرات التكرار في هذه المسألة غير ثابت، كونه مرتبط بإدخال قيمة سالبة، فقد يتم إدخال قيمة سالبة من المرة الأولى، أو بعد خمس قيم، أو مئة .... إلخ.

تتميز البنية التكرارية for بأن عدد مرات التكرار يكون معلوماً وثابتاً كل مرة يتم فيها تنفيذ البرنامج، ولهذا السبب فإنها تسمى أحياناً باسم حلقات العد counting loop أو الحلقات المقادة بالعداد counter-controlled.

ملاحظة: إن هذا التقسيم ليس دقيقاً دائماً، إذ أن من الممكن التعبير عن أي عملية تكرارية باستخدام أي من البنى الثلاث وذلك باستخدام أمر الإنهاء القسري للعملية التكرارية (أي الأمر break) أو أمر تجاهل التكرار الحالي (أي الأمر continue) الذين سنقوم بمناقشتهما لاحقاً في هذا الفصل.

الصيغة العامة لهذه البنية تكون على النحو:

for (init+expression ; Boolean_expression; step_expression)
statement

حيث:

for هي كلمة مفتاحية.

init_expression,boolean_expression,setp_expression تعابير البداية والانتهاء والخطوة الانتقالية

statement أمر بسيط أو مركب.

حيث أن التنفيذ يكون وفق الآلية والترتيب التالي:

1- **يتم تنفيذ أمر التهيئة init_expression.**
2- **يتم اختبار الشرط boolean_exression.**
3- **إذا كان ناتج تقييم الشرط true يتم تنفيذ statement.**
4- **يتم تنفيذ أمر الخطوة الانتقالية أي تغيير حالة متحول الشرط step_expression.**
5- **العودة إلى الخطوة رقم 2.**

يستمر هذا الأمر إلى أن يصبح ناتج تقييم الشرط false، يتم نقل التحكم إلى الأمر التالي للبنية for.

مثال

نكرر حل مسألة حساب مجموع الأعداد من 1 إلى 100، ولكن مع البنية for هذه المرة.

```
int sum = 0;
for (int number = 1; number <= 100 ; number++)
        sum += number;
  Console.WriteLine("SUM OF NUMBERS FROM 1 to 100 is: "+sum);
```

لاحظ أننا حققنا حلقة العد باستخدام الشكل التصاعدي ascending للبنية for الذي يتم فيه مزايدة قيمة متحول التحكم بالحلقة (number في مثالنا هذا).

من الممكن تحقيقها باستخدام الشكل التنازلي descending من خلال مناقصة قيمة متحول الحلقة كما يلي:

```
int sum = 0;
for (int number = 100; number >0 ; number--)
        sum += number;
 Console.WriteLine("SUM OF NUMBERS FROM 1 to 100 is: "+sum);
```

كما نود التنويه إلى أنه تعبير التهيئة، التعبير المنطقي و تعبير الخطوة الانتقالية يمكن أن تكون مركبة (ليس فقط بسيطة) وهذا الأمر ينطبق أيضاً على البنى while و do … while.

5- تداخل الأوامر التكرارية

من الممكن أن يكون الفعل التكراري المراد تنفيذه مؤلفاً من أي نوع من الأوامر بما في ذلك أوامر تكرارية، وفي هذه الحالة نقول بأن البنيتين التكراريتين متداخلتان.

مثال

```
for (int hours = 1; hours <= 12 ; hours ++)
     for (int minutes = 0; minutes < 60 ; minutes ++)
        Console.WriteLine(hours + " : "+ minutes);
```

يتم تنفيذ الحلقة الداخلية (minutes) ستين مرة من أجل كل تنفيذ للحلقة الخارجية (hours):

```
1:00
1:01
1:02
1:03
. . . . . .
1:59
2:00
2:01
2:02
. . . . .
2:59
3:00
.
```

وبالتالي، تتصرف الحلقة الداخلية مثل عقرب الدقائق، وتتصرف الحلقة الخارجية مثل عقرب الساعات.

في حال تم تداخل ثلاث بنى تتصرف البنية الداخلية الثالثة بشكل مشابه لعقرب الثواني

مثال

```
for (int hours = 1; hours <= 12 ; hours ++)
    for (int minutes = 0; minutes <= 60 ; minutes ++)
        for (int second = 0; second < 60 ; second ++)
                Console.WriteLine(hours + " : "+ minutes+ " : "+ second);
```

1:00:00
1:00:01
1:00:02
. . . . . .
1:00:59
1:01:00
2:01:01
2:01:02
. . . . . etc.

6- الأمر التكراري غير المنتهي

من المبادئ الذهبية في علم الخوارزميات لدى تصميم خوارزمية حل لمسألة ما باستخدام الحاسوب هو أن تكون هذه الخوارزمية " **منتهية finite** ".

نواجه أحياناً في البرمجة أحد أمرين:

الأول: الحاجة إلى بنية تكرارية يتحدد توقيت إنهائها أثناء التنفيذ بشكل يدوي من قبل المستخدم، كما في حالة حلقات الإدخال على سبيل المثال.

الثاني: قد نواجه في بعض الأحيان اختلالاً منطقياً في التعبير عن مكونات البنية التكرارية مثل شرط الانتهاء أو الخطوة الانتقالية، كأن تحذف هذه الشروط أو أن يتم التعبير عنها بطريقة تجعل تحقق شرط الانتهاء أمراً غير ممكن.

إن كلا الحالتين تقودنا إلى الدخول في حلقة لانهائية infinite loop.

كأمثلة عن هذا النوع من الحلقات:

```
for ( ; ; )
      Console.WriteLine(" HELLO ");
```

سيتم طباعة الرسالة HELLO عدداً لانهائياً من المرات، لعدم وجود شرط انتهاء في هذه الحلقة.

```
int counter =1;
while (counter < 100)
        Console.WriteLine(" HELLO ");
```

سيتم طباعة الرسالة HELLO عدداً لانهائياً من المرات، كون شرط الانتهاء لن يتحقق أبداً لعدم وجود أي تغيير ضمن الحلقة لحالة المتحول المختبر ضمن الشرط.

```
int counter =1;
do
{
      Console.WriteLine(" HELLO ");
      Counter++;
}
while (counter > 0) ;
```

سيتم طباعة الرسالة HELLO عدداً لانهائياً من المرات، كون شرط الاستمرار محقق دائماً.

إن استخدام مثل هذه الحلقات أمر غير مسموح به في البرمجة ويتناقض مع مبادئ علوم الخوارزميات، لذلك قدمت لغة C# عدة طرق للتحكم بمثل هذه الحلقات أهمها استخدام الأمر break.

7- الأمران break و continue

تتيح لغة البرمجة C# تعليمات تحكم بإمكانها كسر قواعد البرمجة المهيكلة بحيث تسمح بالخروج من كتلة تعليمات قبل انتهائها. ولكن ينصح دوماً باستخدامها في الحالات الاستثنائية فقط (الحلقات اللامنتهية على سبيل المثال).

الأمر break

يؤدي الأمر break إلى الإنهاء القسري لكتلة تعليمات والخروج خارجها.

مثال:

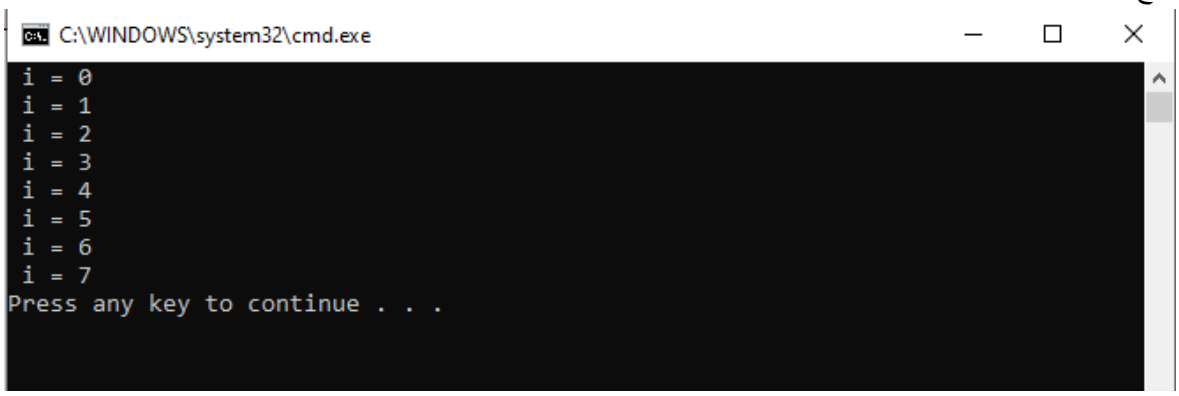
يهدف هذا المثال فقط إلى توضيح استخدام التعليمة break للخروج من كتلة تعليمات.

المطلوب من البرنامج تكرار تنفيذ تعليمات مادامت قيم i أصغر أو يساوي قيمة معينة (القيمة 13 في مثالنا). وللتبسيط أكثر في مثالنا، تقتصر التعليمات المطلوب تكرار تنفيذها على تعليمة إظهار قيمة i.

هذا هو المسار الأساسي لعمل برنامجنا. أما الاستثناء فهو قيم خاصة للمتحول i، نريد إيقاف تنفيذ التعليمات بسببها. القيم الخاصة في مثالنا هي i من مضاعفات 4. وما نقصده هنا بمضاعفات 4 ، هو أي عدد أكبر من 4، ويقسم 4 بلا باقي (باقي قسمته على 4 تساوي 0).

```
int i =0, n=13;
while ( i <= n)
{
      if ( ( i%4 == 0 ) && ( i > 4 ))
          break ;
       Console.WriteLine(" i = "+i);
      i++;
}
```

ناتج التنفيذ:



ملاحظات:

- لاحظ أن تكرار تنفيذ مجموعة/كتلة تعليمات while توقف عند القيمة 8 ، لأنها من مضاعفات 4، فقد وضعنا ذلك شرطاً ننفذ عنده تعليمة break للخروج إلى مابعد كتلة التعليمات.
- لاحظ أنه بإمكاننا تحقيق النتيجة نفسها بكتابة البرنامج دون استخدام تعليمة كسر الحلقة break، كيف؟

ببساطة لا نريد استمرار تكرار التنفيذ عندما يكون الشرط الذي استخدمناه لكسر الحلقة صحيحاً، نضيف ذلك كشرط إلى شرط تكرار الحلقة.

```
int i =0, n=13;
while (( i <= n) && ( i%4 == 0 ) && ( i > 4 ))
{
       Console.WriteLine(" i = "+i);
      i++;
}
```

الأمر continue

يستخدم الأمر continue للاستمرار في التنفيذ ضمن الحلقة التكرارية، ولكن بالانتقال إلى التكرار التالي.

مثال:

نريد القيام بنفس العمل الذي قمنا به عند استخدام break في المثال السابق، أي تجنب إظهار مضاعفات 4، ولكن أريد لبرنامجي ألا يتوقف عن التكرار، أي لا أريد الخروج من تعليمة التكرار، بل أريد منه الانتقال بالتنفيذ إلى التكرار التالي مباشرةً دون تنفيذ بقية كتلة التعليمات.

```
int i , n=13;
for (i =0; i <= n ; i++)
{
      if ( ( i%4 == 0 ) && ( i > 4 ))
         contiue ;
       Console.WriteLine(" i = "+i);
}
```

ناتج التنفيذ:

ملاحظات:

- لاحظ أن تنفيذ تعليمات الحلقة (إظهار قيمة i) جرى من أجل القيم من 1 حتى 7، ولكن عند القيمة 8 (مضاعفات 4) لم يتم التنفيذ/الإظهار بل تابعنا مباشرةً اختبار شرط التكرار نتيجة تنفيذ continue، هنا صارت i=9 ، شرط الحلقة صحيح، إذن ننفذ تعليمة الحلقة أي إظهار i، وهكذا يتكرر الإظهار للقيم 9 ، 10، 11 .ثم، تأخذ i القيمة 12(عدد من مضاعفات 4)، إذن تنفَّذ continue، فننتقل مباشرةً إلى اختبار شرط الحلقة مع القيمة 13...

8- تمارين وأنشطة

تمرين 1- ما قيمة المتغير s بعد تنفيذ كل من التعليمات التالية:

```
int s = 0;
for (int i = 1; i <= 5; i++)
{
    s = s + i;
 }
```

```
int s = 0;
for (int i = 1; i <= 5; i++)
{
    s = s * i;
 }
```

```
int s = 0;
for (int i = 1; i <= 5; i++)
{
    if ( i % 2 ==1)
        s = s + i;
 }
```

```
int s = 0;
for (int i = 1; i <= 5; i=i+2)
{
    s = s + i;
 }
```

```
int s = 0, i=1;
while (i <= 5)
{
    s = s + i;
    i = i+1;
 }
```

```
int s = 0, i=0;
while (i <= 5)
{
    i = i+1;
    if  ( i % 2 ==0 ) break;
        s = s + i;
 }
```

```
int s = 0, i=0;
while (i <= 5)
{
    i = i+1;
    if  ( i % 2 ==0 ) continue;
        s = s + i;
 }
```

```
int s = 0, i=0;
do
{
    i = i+1;
    s = s + i;
 }
while (i <= 5);
```

تمرين 2– قم بكتابة تطبيق يطلب من المستخدم إدخال عدد صحيح. يقوم التطبيق باختبار فيما إذا كان العدد أولي أم لا ويُظهر رسالة موافقة.

تمرين 3– قم بكتابة تطبيق يطلب من المستخدم إدخال عددين صحيحين. يقوم التطبيق بحساب القاسم المشترك الأعظم للعددين ومن ثم طباعته.

ملاحظة: يُمكنك استخدام الخوارزمية التالية لحساب القاسم المشترك الأعظم: كرر طرح العدد الكبير من العدد الصحيح حتى يُصبح العددين متساويين.

تمرين 4– قم بكتابة تطبيق يقوم بطباعة الشكل التالي:

```
*
**
***
****
*****
******
*******
********
*********
**********
```

تمرين 5– قم بكتابة تطبيق يقوم بطباعة الشكل التالي:

```
**********
*********
********
*******
******
*****
****
***
**
*
```