المحاضرة # ٤: الخوارزميات (Algorithms)

١ مقدمة

تتضمن عملية حل المسائل في الحاسوب أربعة مراحل: التحليل (Analysis) والتوصيف (Specification) وتطوير الخوارزمية (Algorithm Development) والتنفيذ (Implementation) والصيانة (Maintenance). ان الناتج من المرحلة الأولى (التحليل والتوصيف) هي جملة واضحة توصف المشكلة المطلوب حلها. اما الناتج من مرحلة تطوير الخوارزمية فهو خطة لحل عام للمشكلة التي تم تحديدها في المرحلة الأولى. ناتج المرحلة الثالثة (التنفيذ) هو برنامج حاسوب محدد والذي ينفذ الخوارزمية والذي يمثل الحل المحدد للمشكلة. لا يوجد هنالك إخراج من المرحلة الرابعة (الصيانة) مالم تكن هنالك أخطاء في البرنامج او تغييرات مطلوب إجرائها. فان وجدت هذه الأخطاء او التغيرات فسوف يتم إعادة إرسالها إلى المرحلة الأولى، الثانية او الثالثة حسب الحاجة. يوضح الشكل ٢ كيفية تفاعل هذه المراحل مع بعضها البعض. تمثل الأسهم باللون الأسود المسار العام خلال هذه المراحل بينما تمثل الأسهم باللون الأحمر المسارات العائدة الى المراحل السابقة نتيجة لحدوث خطأ.

التحليل والتوصيف (Analysis and Specification)	
فهم وتعريف للمسألة	التحليل
تحديد المسألة التي يجب على البرنامج حلها	التوصيف
تطوير الخوارزمية (Algorithm Development)	
تطوير سلسلة من الخطوات المنطقية لاستخدامها في حل المشكلة	تطوير الخوارزمية
تتبع هذه الخطوات لمعرفة قدرتها على حل المشكلة	اختبار الخوارزمية
التنفيذ (Implementation)	
ترجمة الخوارزمية (الحل العام) الى برنامج بإحدى لغات برمجة الحاسوب	كتابة البرنامج
تنفيذ البرنامج على الحاسوب وتتبع النتائج وأجراء التصحيحات الى ان تكون جميع النتائج صحيحة.	اختبار البرنامج
الصيانة (Maintenance)	
استخدام البرنامج	استخدام
تعديل البرنامج لموائمته مع المتطلبات او لتصحيح الأخطاء	صيانة

شكل ١: مراحل حل المسائل في الحاسوب

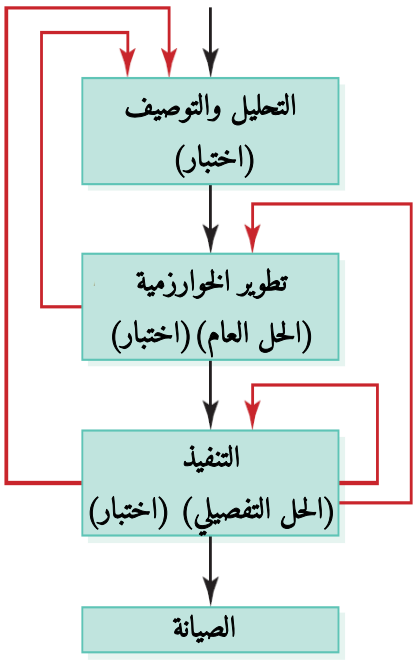
مصدر الشكل كتاب Computer Science Illuminated by Nell B. Dale

٢ الخوارزميات (Algorithms)

تُعّرف **الخوارزمية (Algorithm)** على إنها مجموعة من الإيعازات الواضحة لحل مسالة او جزء من مسألة في وقت محدود وباستخدام بيانات محدودة. تتكون منهجية تصميم الخوارزميات (وفق تصميم من أعلى إلى أدنى top-down design) من أربع خطوات: تحليل المسألة، كتابة الوحدة الرئيسية (Main Module)، كتابة الوحدات (Modules) المتبقية وإعادة الترتيب والتنقيح عند الضرورة.

تهدف مرحلة تحليل المسألة الى فهم المسألة المطلوبة. وتبدأ بكتابة المعلومات المطلوب العمل عليها. ثم تحديد الشكل الذي سيبدو علية الحل. مع وضع قائمة بأي افتراضات تقوم بها حول المسألة أو المعلومات.

تدعى كتابة قائمة بالمهام الرئيسية **بالوحدة الرئيسة (Main Module)**. تستخدم اللغة البسيطة (العربية أو الإنكليزية) لكتابة وصياغة هذه المهام. تكتب هذه المهام بشكل موجز ومختصر وبالتأكيد فان بعض هذه المهام يتطلب تفصيلًا ومعالجة أكثر وهذا دور الوحدات الثانوية الأخرى.



شكل ٢: التفاعل بين مراحل حل المسائل

مصدر الشكل كتاب Computer Science Illuminated by Nell B. Dale

أن عملية كتابة الوحدات الثانوية الأخرى قد تكون بعدة مستويات. أي إن تفصيل المهام الرئيسية قد يليه تفصيل آخر للمهام الثانوية. في النهاية يجب مراجعة كل وحدة وأجراء التحسينات المطلوبة لها حتى تكتمل الخوارزمية. لضمان صحة ودقة الخوارزمية تتطلب مراجعة الوحدات الرئيسية والثانوية. قد تتضمن هذه المراجعة تغييرات في التسلسل الوحدات أو أنشاء وحدات ثانوية أخرى.

٢-١ ملخص المنهجية

يمكن تقسيم منهجية تصميم الخوارزميات إلى أربع خطوات رئيسية:

١. تحليل المشكلة

افهم المشكلة وأدرج المعلومات التي لديك للعمل بها. قد تتكون هذه المعلومات على الأرجح من البيانات المتعلقة بالمشكلة. حدد كيف سيبدو الحل. إذا كانت تقريرًا فحدد الشكل. اسرد أي افتراضات قد تقوم بها حول المشكلة أو المعلومات. فكر. كيف ستحل المشكلة يدويًا؟ طور خوارزمية عامة أو خُطَّة عامة.

٢. سرد المهام الرئيسية

يُطلق على سرد المهام الرئيسية اسم الوحدة الرئيسة. استخدم الإنجليزية (او العربية) أو رموز البرنامَج المستعار (Pseudocode) لإعادة صياغة المشكلة في الوحدة الرئيسة. استخدم أسماء المهام لتقسيم المشكلة إلى مجالات وظيفية. إذا كانت الوحدة الرئيسة طويلة جدًا فإنك تتضمنها تفاصيل كثيرة لها. قدم أي هياكل تحكم (الشروط وحلقات التكرار) تكون ضرورية في هذه المرحلة. أعد ترتيب الأجزاء الفرعية بشكل منطقي إذا كان ذلك ضروريًا. أجل التفاصيل للوحدات الفرعية الأخرى. كلّما عليك فعله في الوحدة الرئيسة هو إعطاء أسماء (ذات معنى) للوحدات على مستويات أقل تحل بعض المهام المحددة.

٣. كتابة الوحدات المتبقية

لا يوجد عدد ثابت من المستويات. يمكن للوحدات على مستوى واحد تحديد المزيد من الوحدات على مستويات أدنى. يجب أن تكون كل وحدة كاملة، حتى إذا كانت تشير إلى وحدات غير مكتوبة. حسن تدريجيًا كل وحدة حتى تكون كل عبارة خطوة ذات معنى.

٤. إعادة ترتيب وتنقيح عند الحاجة

للتغيير خطط. لا تخف من بَدْء من جديد. قد تكون هناك حاجة إلى عدة محاولات وتحسينات. حاول الحفاظ على الوضوح. عن نفسك عبر ببساطة وبشكل مباشر.

٢-٢ اختبار الخوارزمية

هدف حل المشكلات الرياضية هو إنتاج إجابة محددة لمشكلة لذا فإن التحقق من النتائج يعد ما يعادل اختبار العملية التي تم بواسطة استنتاج الإجابة. إذا كانت الإجابة صحيحة فإن العملية صحيحة. ومع ذلك يهدف حل مشكلات الحاسوب إلى إنشاء العملية الصحيحة. يمكن استخدام الخوارزمية التي تجسد هذه العملية مرة بعد مرة مع بيانات مختلفة لذلك يجب اختبار أو التحقق من العملية نفسها.

يتضمن اختبار الخوارزمية غالبًا تشغيل البرنامَج الذي تم تشفير الخوارزمية فيه تحت ظروف متنوعة وفحص النتائج لاكتشاف المشاكل. ومع ذلك يمكن إجراء هذا النوع من الاختبارات فقط عند اكتمال البرنامَج أو في الأقل جزئيًا مما عدّ متأخرًا جدًا للانتهاء من الاعتماد على هذا النوع من الاختبارات فقط. كلما تم اكتشاف المشاكل وإصلاحها في وقت أبكر كلما كانت التكلفة والسهولة في التعامل معها أرخص وأسهل. من الواضح إننا بحاجة إلى إجراء اختبارات في مراحل أوقات مبكرة في عملية التطوير وتحديدًا قبل تنفيذها.

٣ تمثيل الخوارزميات

تعرفنا على الخوارزمية على إنها خِطَّة عامة لحل المسائل وعادة ما تتكون من مجموعة من الخطوات. هنالك طريقتان للتعبير عن هذه الخطوات. أما بصورة نصوص تسمى بالبرنامج المستعار (Pseudocode) أو بصورة أشكال هندسية تسمى بالمخطط الانسيابي (Flowchart).

٣-١ البرنامج المستعار (Pseudocode)

إن البرنامَج المستعار هو ليس لغة برمجية ولكنة مختصرات نصيه كالتي يستخدمها الناس عادة لكتابة الملاحظات. ليست هنالك قواعد لغوية محددة لكتابة البرنامَج المستعار ولكن للتعبير عن خطوات الخوارزميات يجب أن نكون قادرين على التعبير عن المفاهيم التالية:

المتغيرات (variables)

الأسماء التي تظهر في خوارزميات البرنامَج المستعار تشير إلى أماكن في الذاكرة حيث تخزن القيم. لابد أن يظهر الاسم دور محتوياتها في الخوارزمية. مثلا المتغير (sum) يمكن استخدامه لتمثيل حاصل جمع مجموعة من القيم.

التعيين (Assignment)

إذا كان لدينا متغير لابد أن يكون لدينا طريقة لوضع القيم فيه. تستخدم الجملة التالية لخزن القيمة صفر في المتغير (sum):

```
Set sum to 0
```

يمكن استخدام طريقة أخرى للتعبير عن نفس المعنى السابق باستخدام السهم (<--):

```
sum <-- 0
```

عندما نريد الوصول إلى قيمة ذلك المتغير لاحقا نستخدم الاسم فقط. مثلا نقوم بقراءة القيم في المتغيرين (sum) و (num) في الجمل التالية:

```
Set sum to sum + num
```

او

```
sum <-- sum + num
```

ان القيمة المخزونة في المتغير (sum) جمعت مع القيمة المخزونة في المتغير (num) والنتيجة خزنت في المتغير (sum) مرة أخرى. وبالتالي عندما يكون المتغير على الجهة اليمنى من (to) او (<--) فان قيمة المتغير تقرأ فقط. أما إذا كان المتغير يتبع (set) او على الجانب الأيسر من (<--) فان القيم سوف تخزن في ذلك المتغير. أن القيمة التي يتم خزنها في المتغير اما انا تكون قيمة مفردة كما في الصفر او تعبير حسابي مكون من متغيرات وعمليات كما هو في (sum + num) .

الإدخال والإخراج

معظم برامج الحواسيب تعالج بيانات من نوع ما. لذا يجب علينا أن نكون قادرين على إدخال البيانات وإخراج النتيجة على الشاشة. بإمكاننا استخدام الكلمة (write) للإخراج و الكلمة (read) للإدخال.

```
Write "Enter the number of values to read and sum"
Read num
```

أن الرموز بين علامتي الاقتباس تدعى **بالنصوص (strings)** وهي تخبر المستخدم ماذا عليه أن يدُخل أو لوصف ما تمت كتابته. يمكن استخدام كلمات أخرى بديلة مثلا يمكن استخدام الكلمة (Display) او (Print) بدلا من (Write) للكتابة كذلك يمكن استخدام الكلمة (Get) أو (Input) بدلا من (Read) للقراءة. تذكر بان الخوارزميات المكتوبة بصيغة برنامَج مستعار كتب لكي تكون مفهومة من البشر لكي تتم ترجمتها في مرحة لاحقة إلى إحدى لغات البرمجة لذلك فلا توجد قواعد محددة لها. على إي حالة فان استخدام نفس الكلمات خلال كتابة خوارزمية معينة يسهل فهمها من قبل كاتب الخوارزمية و إي شخص أخر يعمل معه في كتابتها. أخيرًا فان عبارتي الأخراج التاليتين يوضحان نقطة مهمة:

```
Write "Err"
Write sum
```

العبارة الأولى تكتب الرموز ما بين علامتي الاقتباس على الشاشة. والثانية تكتب محتويات المتغير (sum) على الشاشة. أن قيمة المتغير (sum) لا تتغير خلال العبارة السابقة.

الاختيار (Selection)

يسمح بناء الاختيار بالاختيار بين تنفيذ الأجراء أو تخطيه أو بين تنفيذ إجراءين اعتمادًا على عبارة الشرط الموجودة بين قوسين. على سبيل المثال البرنامَج المستعار التالي يطبع قيمة المتغير (sum) او رسالة خطأ اعتمادًا على الشرط فيمَا إذا كانت قيمة المتغير (sum) اصغر من صفر أو لا.

```
// Read and sum three numbers
IF (sum < 0)
    Print error message
ELSE
    Print sum
// Whatever comes next
```

لتجميع الجمل البرمجية نترك مسافة فارغة عند بداية كتابه تلك الجُمل وكما هو في المثال أعلاة. حيث إن جملتي الطباعة (Print) كتبتا بعد مسافة بسيطة من بداية جملتي (IF) و (ELSE). كذلك نلاحظ في البرنامج السابق استخدام الرمز (//) في السطر الأول والأخير لكتابة الملاحظات على البرنامَج (تمتلك كل لغات البرمجة رمز معين لغرض كتابة الملاحظات على البرامج). أن الاختبار أو الشرط في

المثال السابق كُتب بطريقة إذا-أذن-آخر (if-then-else) إي أن هنالك عبارتين برمجيتين احدهما تنفذ عن تحقق الشرط والأخرى تنفذ عن عدم تحققه.

الصيغة الأبسط من ذلك هو وجود جملة برمجية واحدة تنفذ عند تحقق الشرط كما هو في المثال التالي:

```
// Read and sum three numbers
IF (sum < 0)
    Print error message
// Whatever comes next
```

أضافه إلى ذلك يمكننا إضافة عدة شروط و جمل برمجية مطلوب تنفيذها عند تحقق الشرط المرتبطة به.

التكرار (Repetition)

إن بناء التكرار يسمح بتكرار تنفيذ الإيعازات عدة مرات حسب شرط معين أو لعدد معين من المرات. فمثلا في مسائله الجمع, نحتاج لاستخدام عداد (Counter) بإعطائه قيمة أولية ثم نختبره تم نزيد قيمته. كما هو في الاختيار IF فان العبارة بين القوسين بجانب عبارة التكرار WHILE تمثل الاختبار أو الشرط. عند تحقق الشرط فان الجملة أو الجمل البرمجية داخل حلقة التكرار سوف تنفذ أما إذا لم يتحقق الشرط فسوف تنتهي حلقة التكرار.

```
Set limit to the number of values to sum
WHILE (counter < limit)
    Read num
    Set sum to sum + num
    Set counter to counter + 1
// Rest of program
```

أن التعابير بين الأقواس بجانب عبارتي WHILE و IF هي تعابير منطقية أي إن تحققها ينتج عنه نتيجة منطقية صحيحة (True) وعن عدم تحققها ينتج عنها نتيجة منطقية غير صحيحة (False). تم تمييز العبارات WHILE و IF و ELSE بكتابتها بالأحرف الكبيرة وذلك لإنها في الغالب تمثل الإيعازات في عدة لغات برمجية.

تقوم الخوارزمية التالية المكتوبة بصيغة البرنامَج المستعار (Pseudocode) بقراءة وجمع عددين وتطبع عبارة خطأ في حاله كان المجموع عدد سالب.

```
Set sum to 0
Read num1
Set sum to sum + num1
Read num2
Set sum to sum + num2
If (sum < 0)
     Write "Error"
ELSE
     Write sum
```

أما الخوارزمية التالية فتقرأ عدد محدد من الأعداد وتجمع وتطبع النتيجة:

```
Set counter to 0
Set sum to 0
Read limit
While (counter < limit)
     Read num
     Set sum to sum + num
     Set counter to counter + 1
Print sum
```

٣-٢ المخطط الانسيابي (Flowchart)

أن المخطط الانسيابي هو تمثيل رسومي للخوارزمية. غالبًا ما يستخدمه المبرمجون كأداة لتخطيط البرامج لحل مشكلة ما. يَستخدم الأشكال المرتبطة فيمَا بينها للإشارة إلى تدفق المعلومات والمعالجة. تُعرف عملية رسم مخطط انسيابي للخوارزمية باسم مخطط الانسياب (Flowchart).

٣-٢-١ فوائد استخدام المخططات الانسيابية

الوضوح: تبسط المخططات الانسيابية الخوارزميات المعقدة بصريًا مما يجعلها أسهل للفهم.

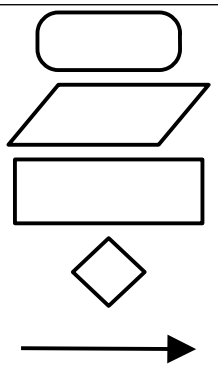
التواصل: تعمل كلغة عالمية لمناقشة الخوارزميات بين أعضاء الفريق.

تصحيح الأخطاء وإصلاحها: تساعد المخططات الانسيابية في تحديد الأخطاء المنطقية في الخوارزميات.

التوثيق: توفر سجلاً واضحًا للعمليات للرجوع إليها مستقبلاً.

٣-٢-٢ مكونات المخطط الانسيابي

تستخدم المخططات الانسيابية رموزًا قياسية لتمثيل أنواع مختلفة من الإجراءات والقرارات. الجدول التالي فيه الأشكال الهندسة الشائعة الاستخدام في رسم المخطط الانسيابي.

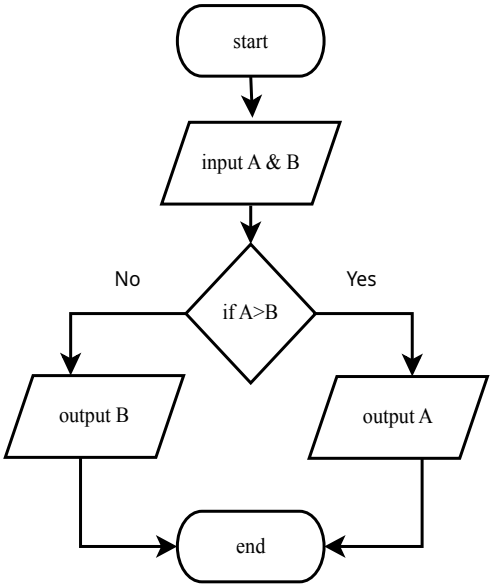
الشكل	الاستخدام
	يشير إلى بداية أو نهاية الخوارزمية
	يشير إلى عملية الإدخال أو الأخراج
	يشير إلى العمليات الحسابية مثل الجمع والقسمة
	يشير إلى الاختبار (الشروط)
	يشير إلى اتجاه انسياب الخوارزمية

٣-٢-٣ كيفية إنشاء مخطط انسيابي

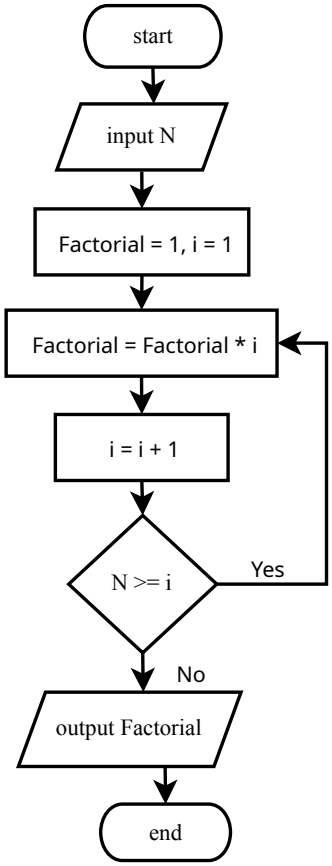
١. تحديد خطوات الخوارزمية.
٢. اختيار الرموز المناسبة لكل خطوة.
٣. ترتيب الرموز بترتيب منطقي.
٤. توصيل الرموز بالأسهم لإظهار التدفق.
٥. مراجعة المخطط للتأكد من اكتماله ووضوحه.

٣-٢-٤ امثلة

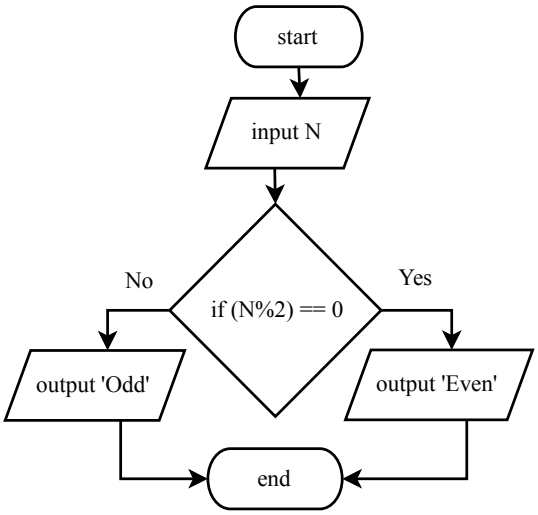
المثال ١: خوارزمية لإيجاد العدد الأكبر بين رقمين



المثال ٢: خوارزمية لحساب المضروب لعدد ما



المثال ٣: خوارزمية لتحديد العدد الزوجي والفردي



٤ الخوارزميات ذات المتغيرات البسيطة

المتغيرات البسيطة (الذرية) هي تلك التي لا يمكن تقسيمها وهي قيمة مخزنة في مكان ما.

٤-١ الخوارزمية مع الاختيار

في المثال التالي سنحاول كتابة خوارزمية بسيطة لغرض فهم عملية تصميمها بالطريقة السابقة الذكر. افترض باننا نريد كتابة خوارزمية لغرض اقتراح الملابس المناسبة لدرجات حرارة جو مختلفة. فمثلا سنقترح لبس قميص نصف كم للجو الحار جدا وقميص كم كامل لجو معتدل ومعطف خفيف لجو بارد ومعطف سميك لجو بارد جدا وأخيرًا التوصية بالبقاء بالمنزل لجو متجمد.

وعلية فان الوحدة الرئيسة (المهام الرئيسية) تتضمن المهام التالية:

```
Write "Enter the temperature"
Read temperature
Determine dress
```

أن الجملتين الأولى والثانية لا تتطلب المزيد من المهام ولكن الجملة الثالثة تتطلب المزيد من التفصيل لكي يكتمل حل المسألة. يجب ربط درجات الحرارة بتوصيفات الجو المطلوبة أي إن الجو الحار جدا يمثل أي درجة حرارة أكبر من ٣٥ مئوية والجو الحار يمثل أي درجة حرارة أكبر من ٢٠ مئوية والجو البارد يمثل أي درجة حرارة أكبر من ١٠ مئوية والجو بارد جدا يمثل أي درجة حرارة أكبر من صفر مئوية وأخيرًا فان الجو المتجمد يمثل أي درجة حرارة أصغر من صفر مئوية. علية يمكننا

تفصيل (تحديد المهام الثانوية) الوحدة الرئيسة "حدد الملبس المناسب" (Determine dress) بالوحدة الثانوية التالية:

```
Determine dress
IF (temperature > 35)
    Write "Texas weather: wear shorts"
ELSE IF (temperature > 20)
    Write "Ideal weather: short sleeves are fine"
ELSE IF (temperature > 10)
    Write "A little chilly: wear a light jacket"
ELSE IF (temperature > 0)
    Write "Philadelphia weather: wear a heavy coat"
ELSE
    Write "Stay inside"
```

نلاحظ من الجمل أعلاه بان إي عبارة شرط (إذا) لن تكون صحيحه أن لم تكن العبارات التي تسبقها خاطئة فمثلا إذا كانت عبارة الشرط (درجة الحرارة > ١٠) صحيحة فهذا يعني أن درجة الحرارة التي ادخلها المستخدم هي بين ١١ و٢٠ مئوية.

٤-٢ الخوارزمية مع التكرار

هناك نوعان أساسيان من الحلَقات: التحكم في العدد والتحكم في الأحداث.

حلَقات يتم التحكم فيها بالعد

الحلقة التي يتم التحكم فيها بالعد هي حلقة تكرر العملية لعدد محدد من المرات. تقوم آلية التكرار ببساطة باحتساب كل مرة تتكرر فيها العملية وتختبر ما إذا كانت قد انتهت قبل البَدْء مرة أخرى.

هناك ثلاث أجزاء مميزة لهذا النوع من الحلَقات التي تستخدم متغيرًا خاصًا يسمى متغير التحكم في الحلقة. الجزء الأول هو التهيئة: تتم تهيئة متغير التحكم في الحلقة إلى قيمة البداية. الجزء الثاني هو الاختبار: هل وصل متغير التحكم في الحلقة إلى قيمة محددة مسبقًا؟ الجزء الثالث هو الزيادة: يتم زيادة متغير التحكم في الحلقة بمقدار ١. تكرر الخوارزمية التالية عدد مرات محددة للعملية:

```
Set count to 0                  // Initialize count to 0
WHILE (count < limit)           // Test
    . . .                       // Body of the loop
    Set count to count + 1      // Increment
. . .                           // Statement(s) following loop
```

يتم ضبط متغير التحكم في الحلقة count to 0 خارج حلقة التكرار. يتم اختبار count < limit ويتم تنفيذ الحلقة طالمَا أن التعبير صحيح. العبارة الأخيرة في الحلقة تزيد من متغير التحكم في الحلقة، count. كم مرة يتم تنفيذ الحلقة؟ يتم تنفيذ الحلقة عندما يكون متغير التحكم (count) مساوي الى ٠، ١، ٢، . . . ,Limit – 1. وبذلك يتم تنفيذ الحلقة بعدد مرات محددة بالقيمة الأولية لمتغير التحكم (count) في الحلقة والعامل المحدد (limit) المستخدم في التعبير المنطقي (count < limit).

تسمى حلقة while حلقة الاختبار المسبق لأن الاختبار يتم قبل تنفيذ الحلقة. إذا كان الشرط خاطئًا في البداية فلن يتم الدخول في الحلقة. ماذا يحدث إذا تم حذف عبارة الزيادة (Set count to count + 1)؟ لن يتغير التعبير المنطقي أبدًا. إذا كان التعبير خاطئًا في البداية فلن يحدث شيء لم يتم تنفيذ الحلقة فقط. إذا كان التعبير صحيحًا في البداية فلن يتغير التعبير أبدًا لذلك يتم تنفيذ الحلقة إلى الأبد. في الواقع تحتوي معظم أنظمة الحوسبة على مؤقت لذلك لن يعمل البرنامَج إلى الأبد. وبدلًا من ذلك سيتوقف البرنامَج مع ظهور رسالة خطأ. الحلقة التي لا تنتهي أبدًا تسمى حلقة لا نهائية.

حلَقات التحكم بالحدث

الحلَقات التي يتم فيها التحكم في عدد التكرارات من خلال حدث يقع داخل جسم الحلقة نفسها تسمى حلَقات يتم التحكم فيها بالحدث.

عند تنفيذ حلقة يتم التحكم فيها بالحدث باستخدام عبارة while، هناك مرة أخرى ثلاث أجزاء للعملية: يجب تهيئة الحدث، ويجب اختبار الحدث، ويجب تحديث الحدث. تحتوي خوارزمية التالية على حلقة يتم التحكم فيها بالحدث:

```
Write "Enter the new base"
Read newBase
Write "Enter the number to be converted"
Read decimalNumber
Set answer to 0
Set quotient to 1
WHILE (quotient is not zero)
    Set quotient to decimalNumber DIV newBase
    // Rest of loop body
Write "The answer is ", answer
```