كيف تبني مفسراً

تأليف

خليل الأمين عبد الجواد

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

الإهداء

إلى كل طالب علم يتمثل بقول الشاعر :

لذّتي في العلمِ إنه عسلِي والشَّهد يرْجى بالجِد لا الكسلِ

الفهرس

1- مقدمة .

2- السلاسل النصية والدوال الخاصة بها .

3- تمرينات حول معالجة النصوص .

4- بناء المفسر.

4-1- نظرة عامة عن كيفية عمل مفسر Learn .

4-2- شرح سطور Learn .

5- الملاحق

الملحق أ- شفرة Learn كاملة .

الملحق ب- كيفية تنفيذ البرامج.

مقدمة

يشرح هذا الكتاب كيفية بناء مفسر لغة برمجة ، والطريقة المتبعة في بناء المفسر فيه ليست الطريقة القياسية ، وإنما طريقة قمت بابتكارها للوصول إلى هذا الهدف ، أي بناء المفسر ، وأما عن الطريقة القياسية فمعلوماتي عنها قليلة جداً ولا يمكنني أن أبني مفسراً بناءً عليها ، ولأن كل الطرق تؤدي إلى روما والتي هنا هي المفسر ، فانسداد أحد الطرق أمامنا لا يعني أننا لا يمكننا الوصول ، بل علينا البحث واكتشاف طريق أخرى . وهذه هي الطريق التي اكتشفتها .

ولكي أقوم بتأليف هذا الكتاب كان عليَّ أن أقوم ببناء مفسر للغة برمجة لكي أشرح على ضوئه كيفية البناء ، ولهذا قمت ببرمجة لغة Learn لهذا السبب ، وهي لغة بسيطة جداً ولكنها تفي بالغرض ، وفي الحقيقة لم أكن أتوقع أن أصل بها إلى مستواها الحالي على الإطلاق ،لاسيما أني قد قمت ببرمجتها من الصفر وبدون الاستعانة بالمفسرات التي قمت ببرمجتها من قبل ، ومع ذلك فقد برمجتها في أقل من أسبوع ، وهي مبرمجة بلغة C# وهي اللغة المعتمِد عليها هذا الكتاب في شرحه لبناء المفسرات ، ومن ثم فعلى من يريد فهم الشرح والاستفادة من هذا الكتاب أن يكون على دراية بأساسيات السي # ، الأساسيات فقط لا أكثر ، وطريقتي في بناء المفسرات تعتمد على معالجة النصوص وعلى الأفكار التي يتم ابتكارها لحل ما يطرأ من مشاكل ومن ثم للوصول إلى المراد اعتماداً تاماً ، لهذا سيتم في البداية شرح السلاسل والمتغيرات النصية ودوال التعامل معها في لغة سي # ثم يتم شرح كيفية برمجة وعمل لغة Learn .

وأرجو أن يجد القارئ في الكتاب ما يتمناه وأن تتحقق له الاستفادة العظمى منه .

والسلام عليكم

خليل الأمين عبد الجواد

Khal_i_l@Yahoo.com

WWW.VC4ARAB.COM

النصوص (السلاسل النصية) :

النص عبارة عن سلسلة أو مصفوفة من الحروف أو العناصر الرمزية توضع بين علامتي تنصيص مزدوجتين ""، فأي كلمة وأي أرقام متتابعة أو أية رموز أخرى يمكن أن تكون سلسلة نصية.

ولأن النص كما أسلفنا هو مصفوفة من الحروف فإنه بمعرفتنا لدليل أو رقم ترتيب أي حرف في المصفوفة يمكننا التحكم فيه كيفما أردنا ، وفي لغة سي # فإن دليل أول عنصر في سلسلة نصية هو الصفر وثاني عنصر دليله الواحد وهكذا .

أما طول النص فهو يحدد بعدد العناصر المكونة له ، فمثلا كلمة "سلسلة" طولها 5 ، كما أن دليل آخر حرف فيها هو 4 أو هو طول الكلمة - 1 .

والمتغيرات النصية هي تلك المتغيرات التي تحمل قيما نصية ، وإذا افترضنا أن لدينا متغيراً نصياً بالاسم s ، فإن :

```
s="ABCDEFGHIJKL";// المتغير يحتوي على هذه القيمة
s[1]=> 'B';
s[7]=> 'H';
```

الدوال التي تتعامل مع النصوص :

في شرح الدوال سنفترض بأن s متغير نصي ، وأن i و j أرقام أو متغيرات صحيحة .

قبل شرح الدوال أود الإشارة إلى أن معامل الجمع + يمكننا من جمع سلسلتين نصيتين معاً ودمجهما في سلسلة واحدة ، فمثلاً "Learn language" = "Learn"+" language" .

الدالة ToString() :

هي دالة عامة تقوم بتحويل أي متغير أو تعبير إلى سلسلة نصية ، والمثال التالي يوضح ذلك :

```
int i=45,j=6;
string s=i.ToString() + j.ToString();
```

➔ s = 456

الفئة Convert :

فئة بها دوال أعضاء وظيفتها التحويل بين الأنواع ، وسنحتاجها للتحويل من النوع النصي إلى الصحيح والنوع النصي إلى نوع منطقي .

```
string s="123";
```

int i=Convert.ToInt32(s);

➔ i=123

والآن نأتي للدوال التابعة للمتغيرات النصية :

الدالة Endswith(str) :

ترجع قيمة منطقية ، إذا كانت السلسلة تنتهي بالنص str فإنها ترجع True وإلا ترجع False .

s="ABCD";

bool bo=Convert.ToBoolean(s.EndsWith("D").ToString());

➔bo = True

الدالة IndexOf(str) :

تقوم بإرجاع رقم ترتيب أول ظهور للنص str في النص الذي نتعامل معه ، فمثلاً :

s="Enjoy";

i=s.IndexOf("joy");

➔ i = 2

الدالة Insert(i,str) :

تقوم بإدخال أو حشر السلسلة str في السلسلة s عند رقم الترتيب i :

s="ABCDEF";

s=s.Insert(3,"123");

➔ s= "ABC123DEF"

المتغير العضو Length :

يقوم بإرجاع طول النص .

s="12345";

i=s.Length;

➔ i=5

الدالة Remove(i,j) :

تقوم بحذف أو إزالة مجموعة حروف من السلسة التي نتعامل معها بداية من الدليل i بعدد j من الحروف :

```
s="ABCDEF";
s=s.Remove(3,2);
```

➔ s= "ABCF"

وإذا أردنا أن نحذف من الحرف C إلى نهاية السلسلة نكتب التالي:

```
s=s.Remove(s.IndexOf("C"),s.Length-s.IndexOf("C"));
```

➔ s= "AB"

الدالة Replace(str1,str2) :

تقوم باستبدال النص str1 بالنص str2 أينما وجد في السلسلة s :

```
s="ABCD";
s=s.Replace("BC","XY");
```

➔ s= "AXYD"

الدالة Split(str) :

تقوم تقسيم السلسلة s إلى مجموعة سلاسل يفصل بينها رقم ترتيب str ، ويتم استقبال السلاسل الجزئية الناتجة في مصفوفة حجمها يساوي Split(str).Length ، كما في المثال :

```
s="ab,cd,ef,gh";
string[] ar=new string[s.Split(',').Length];
ar=s.Split(',');
```

➔ ar[2]= "ef"

الدالة StartsWith(str) :

ترجع قيمة منطقية ، إذا كانت السلسلة s تبدأ بالنص str فإنها ترجع True وإلا ترجع False .

```
s="ABCD";
bool bo=Convert.ToBoolean(s.StartsWith("CD").ToString());
```

➔bo = False

الدالة Substring(i,j) :

تقوم باستخلاص سلسلة فرعية من السلسلة s بطول j من الحروف ابتداء من الحرف ذي رقم الترتيب i .

```
s="ABCDEFGHIJKLMNOPQRSTU";
s=s.Substring(0,9);
```

➔s = ABCDEFGHI

الدالة ToLower() :

تقوم بتحويل الحروف الكبيرة في السلسلة إلى حروف صغيرة .

```
s="ABCDEFG";
s=s.ToLower();
```

➔s = "abcdefg"

والدالة ToUpper() عكسها .

الدالة Trim() :

تقوم بإزالة الفراغات من بداية ونهاية السلسلة ، والدالة TrimStart() تزيل الفراغات من البداية فقط ، و TrimEnd() من النهاية فقط .

```
s="  123  ";
s=s.Trim();
```

➔s = "123"

وبهذا ننتهي من الدوال ، وأشير إلى أنه يمكن ربط السلاسل باستخدام المعامل + :

```
s="ab"+" cd";
```

➔s="ab cd"

تمارين في معالجة النصوص :

تمرين 1:

في هذا التمرين سنقوم بتغيير جملة خبرية إلى جملة استفهامية :

```
string s="This book is good. ";
Console.WriteLine(s);
s=s.Remove(s.IndexOf('.'),s.Length-s.IndexOf('.'));
string str=s.Substring(s.IndexOf(" is "),4);
s=s.Remove(s.IndexOf(" is "),3);
```

```
s=str.Trim()+' '+s;
s=s.Trim();
s=s.ToLower();
s=s.Remove(0,1);
s=s.Insert(0,"I");
s+='?';
Console.WriteLine("\n"+s);
```

➔ This book is good.

➔ Is good this book?

الشرح :

في أول سطر عرفنا المتغير النصي s وأعطيناه القيمة المبينة .

في السطر الثاني قمنا بطباعة قيمة المتغير s .

السطر الثالث فيه نبدأ في تحويل الجملة ، إذ أن الجملة الخبرية تنتهي بالنقطة فإننا نقوم بالبحث عنها وحذفها من النص ، وهذا ما يفعله السطر الثالث ، وكان بالإمكان إزالتها بالجملة التالية لو كانت هي آخر حرف في الجملة – لاحظ أني تعمدت وضع فراغ في نهاية الجملة – .

```
s=s.Remove(s.Length-1,1);
```

في السطر الرابع عرفنا متغيراً نصياً جديداً str وأعطيناه قيمة ، والتي هي نص جزئي من المتغير s ، هذا النص يبدأ هو " is " ، ويلاحظ وضعنا " is " كمعامل للدالة IndexOf() ولم نضع "is" لأنه لو فعلنا فإن str ستبدأ من is التي تنتهي بها الكلمة This وليس هذا هو المطلوب ، وهذا سبب وجود الفراغ الأمامي ، أما الفراغ الاخير فلو أنه كانت ثمة كلمة تبدأ بـ is فإنها لن تؤخذ في الاعتبار ، ولأن الجملة التي لدينا لا تحتوي على كلمة مثل هذه فإنه غير ضروري .

في السطر الخامس قمنا بإزالة النص "is " من المتغير s ولم نزل النص " is " لأننا لو فعلنا فإن كلمة book ستلتصق بـ good .

في السطر السادس أعطينا للمتغير s قيمة المتغير str - مزالة منه الفراغات في البداية والنهاية - مع فراغ مع قيمة s الحالية .

في السطر السابع قمنا بإزالة الفراغات من بداية ونهاية السلسلة s هذه الفراغات تكون قد نتجت من عملية الاقتطاع والتبديل .

في السطر الثامن حولنا كل حرف كبير في السلسلة الجديدة الناتجة إلى حرف صغير .

في السطر التاسع نحذف أول حرف في السلسلة - i - وذلك لنستبدله بحرف I كبير، وهذا ما يفعله السطر العاشر .

في السطر الحادي عشر أضفنا الحرف "?" إلى نهاية السلسة كي تدل على الاستفهام .

وفي السطر الثاني عشر نقوم بطباعة السلسلة الناتجة .

تمرين 2 :

في هذا التمرين ستكون لدينا السلسلة التالية :"1+2+3+4+5+6+7+8+9+10" وسنقوم بمعالجتها لنحصل على ناتج الجمع :

```
string s="1+2+3+4+5+6+7+8+9+10";
Console.Write(s+" = ");
int sum=0;
while(s.IndexOf('+')>=0)
{
   sum+=Convert.ToInt32(s.Remove(s.IndexOf('+'),s.Length-s.IndexOf('+')));
   s=s.Remove(0,s.IndexOf('+')+1);
}
sum+=Convert.ToInt32(s);
Console.WriteLine(sum.ToString());
```

➔ 1+2+3+4+5+6+7+8+9+10 = 55

الشرح :

تتكرر الحلقة ما دام المتغير s يحتوي على إشارة الجمع .

أول سطر في الحلقة يستخلص نصاً جزئياً من المتغير s ، هذا النص يبدأ من أول حرف في المتغير s إلى أول ظهور لعلامة الجمع + مع عدم دخولها في النص الجزئي ، ثم يقوم بتحويله إلى عدد صحيح ثم يضيفه إلى المتغير الصحيح sum.

في السطر الثاني يقوم بإزالة النص الجزئي السابق من المتغير s مع علامة الجمع الأولى .

ثم تتكرر الحلقة إلى أن يصبح المتغير s غير محتوٍ على علامة الجمع وحينها تصبح قيمته "10"، وبالطبع هذه القيمة لم تضف إلى المتغير sum في الحلقة ، ولذا فإن أول سطر بعد نهاية الحلقة يقوم بإضافتها .

في التمرين السابق قمنا بالمطلوب ولكن الحل كان معقداً ، أو يمكن القول أنه لم يكن معقداً ولكن ثمة حل أبسط وأسهل منه .

وهنا أشير إلى أن معالجة النصوص ليست استخداماً للدوال فقط بل إنها أفكار أولاً ؛ مع معرفة أي دالة هي الأفضل في أداء المراد ومن ثم استخدامها هي هو الأولى .

فمثلاً في المثال السابق لو أننا قمنا بتقسيم السلسلة باستخدام الدالة Split() عند كل إشارة + لكان أبسط ولاستغنينا عن الدالتين Rmove() و IndexOf() والكود التالي يوضح ذلك :

```
string s="1+2+3+4+5+6+7+8+9+10";
Console.Write(s+" = ");
int sum=0;
string[] ar=new string[s.Split('+').Length];
ar=s.Split('+');
for(int i=0;i<ar.Length;i++)
  sum+=Convert.ToInt32(ar[i]);
Console.WriteLine(sum.ToString());
```

ومع ذلك فإن هذا لا يبين اختلافاً كبيراً ولكن في حالات أخرى فإن الاختلاف يكون كبيراً جداً، ويمكن أن ينتج تبسيط كان قبل الوصول إليه أقرب إلى المستحيل ،

تمرين 3:

في هذا التمرين سنحسب ناتج العبارة "1+23*4*5*6+7+8*9+10" بطريقتين :

```
string s="1+23*4*5*6+7+8*9+10",str;
Console.Write(s+" = ");
```

```
int sum=0,re,i,j;
for(i=0;i<s.Length;i++)
 {
       if(s[i]=='*')
       {
      int before=0,after=0;
        for(j = i;j>0;j--)
         if(s[j]=='+')
         {
              before=j;
        break;
         }
        for(j=i+1;j<s.Length;j++)
            if(s[j]=='+' || s[j]=='*')
            {
              after=j;
              break;
            }
            before++;
            str=s.Substring(before,after-before);
            string s1,s2;
            s1=str.Remove(str.IndexOf('*'),str.Length-str.IndexOf('*'));
            s2=str.Remove(0,str.IndexOf('*')+1);
            re=Convert.ToInt32(s1)*Convert.ToInt32(s2);
            s=s.Remove(before,after-before);
            s=s.Insert(before,re.ToString());
            i=0;
```

```
        }
}
string[] add=new string[s.Split('+').Length];
add=s.Split('+');
for(i=0;i<add.Length;i++)
sum+=Convert.ToInt32(add[i]);
Console.WriteLine(sum.ToString());
```

الكود السابق يقوم بالمطلوب ، ولكن انظر إلى بساطة الكود التالي :

```
string s="1+23*4*5*6+7+8*9+10";
Console.Write(s+" = ");
int sum=0;
string[] ar=new string[s.Split('+').Length];
ar=s.Split('+');
for(int i=0;i<ar.Length;i++)
{
 if(ar[i].IndexOf('*')>0)
 {
        int re=1;
        string[] mult=new string[ar[i].Split('*').Length];
        mult=ar[i].Split('*');
        for(int j=0;j<mult.Length;j++)
        re*=Convert.ToInt32(mult[j]);
        ar[i]=ar[i].Remove(0,ar[i].Length);
        ar[i]=ar[i].Insert(0,re.ToString());
 }
```

```
  sum+=Convert.ToInt32(ar[i]);
}
Console.WriteLine(sum.ToString());
```

واضح أن الكود الثاني عدد سطوره أقل بعشرة أسطر على الأقل من الكود الأول ، والأهم من ذلك أن كتابة فكرة الكود الثاني وبرمجتها أسرع بكثير من الكود الأول .
وأترك للقارئ فهم كيفية عمل الكودين .

بناء المفسر :

الآن نأتي للموضوع الأساسي وهو شرح كيفية بناء المفسرات وذلك بفهم الكيفية التي تمت بها كتابة لغة Enjoy وذلك بتشريح وشرح كل سطر كتبته بالتفصيل .

نظرة عامة عن كيفية عمل مفسر Learn :

يعمل المفسر باستقبال الكود سطراً سطراً من الملف المصدري ثم تتم معالجته ، والسطر يُقصد به الجملة أو مجموعة الجمل التي تنتهي بالفاصلة المنقوطة ، وتم برمجة السطر على هذا الأساس فقط ليكون شكل الكود عند كتابته أفضل تنسيقاً ، و إلا فإنه بالإمكان اعتبار السطر على أنه السطر الذي يتم كتابته في محرر النصوص ولا استغنينا عن الفاصلة المنقوطة ، ولكن بالفاصلة المنقوطة فإن الجملة :

```
if((a>b) And (a>c)) print a else if((b>a) And (b>c)) print b else if((c>a) And (c>b))print c
```

يمكننا كتابتها هكذا :

```
if((a>b) And (a>c))
print a
 else if((b>a) And (b>c))
print b
else if((c>a) And (c>b))
print c;
```

وهذا بالطبع أفضل شكلاً .

والإخراج يتم باستخدام الكلمة print ثم القيم المراد إخراجها على الشاشة ، أما تعريف المتغيرات الصحيحة فإنه يتم بتعريف مصفوفة نصية ، ثم بعد التعرف على جملة تعريف المتغيرات يتم إسناد عدد المتغيرات المعلن عنهم كطول للمصفوفة ، هذه المصفوفة يتم فيها تخزين أسماء المتغيرات ، ويتم تعريف مصفوفة أخرى بنفس الطول يتم فيها تخزين القيم التي ستعطى وتسند للمتغيرات ، وتسند لعناصرها القيمة 0 كقيمة ابتدائية، وهاتين المصفوفتين تعملان جنباً إلى جنب ، وفي جملة الإسناد للمتغيرات فإن القيمة المراد إسنادها سيتم وضعها في العنصر الموجود في مصفوفة القيم ، وهذا العنصر مناظر للعنصر الذي يحتوي على اسم المتغير في مصفوفة الأسماء ، أما في العمليات التي تقوم بمعالجة للمتغيرات مثل العمليات الحسابية والمنطقية وعملية الإخراج فإنه يتم استبدال أسماء المتغيرات بقيمها من المصفوفة الثانية ثم تتم عليها العملية المرادة .

وفي جملة الإدخال فإنه بعد معرفة المتغير المراد إدخال القيمة إليه ، يتم إدخال القيمة في العنصر المناظر له في مصفوفة القيم .

في جملة الشرط يتم التحقق من صحة الشرط من عدمها باستخدام الدالة Eval() المذكورة لاحقاً ولأن المفسر يقوم بقراءة الكود سطراً سطراً فإنه لا يمكن تنفيذ أكثر من جملة بعد جملة الشرط ، وكذلك الحال مع جملة التكرار .

وأنصح بشدة بقراءة الشرح التابع للغة Learn ومعرفة تركيبها قبل المضي في قراءة باقي الكتاب .

والآن نبدأ شرح سطور المفسر .

السطر :

```
using System.IO;
```

يقوم باستدعاء مكتبات الفئات المختصة بالتعامل مع الملفات من إنشاء وإدخال وإخراج وغيرها ، وسبب استخدامنا للملفات أن المفسر يقوم باستقبال شفرة المصدر على هيئة ملف وهو الملف المصدري طبعاً.

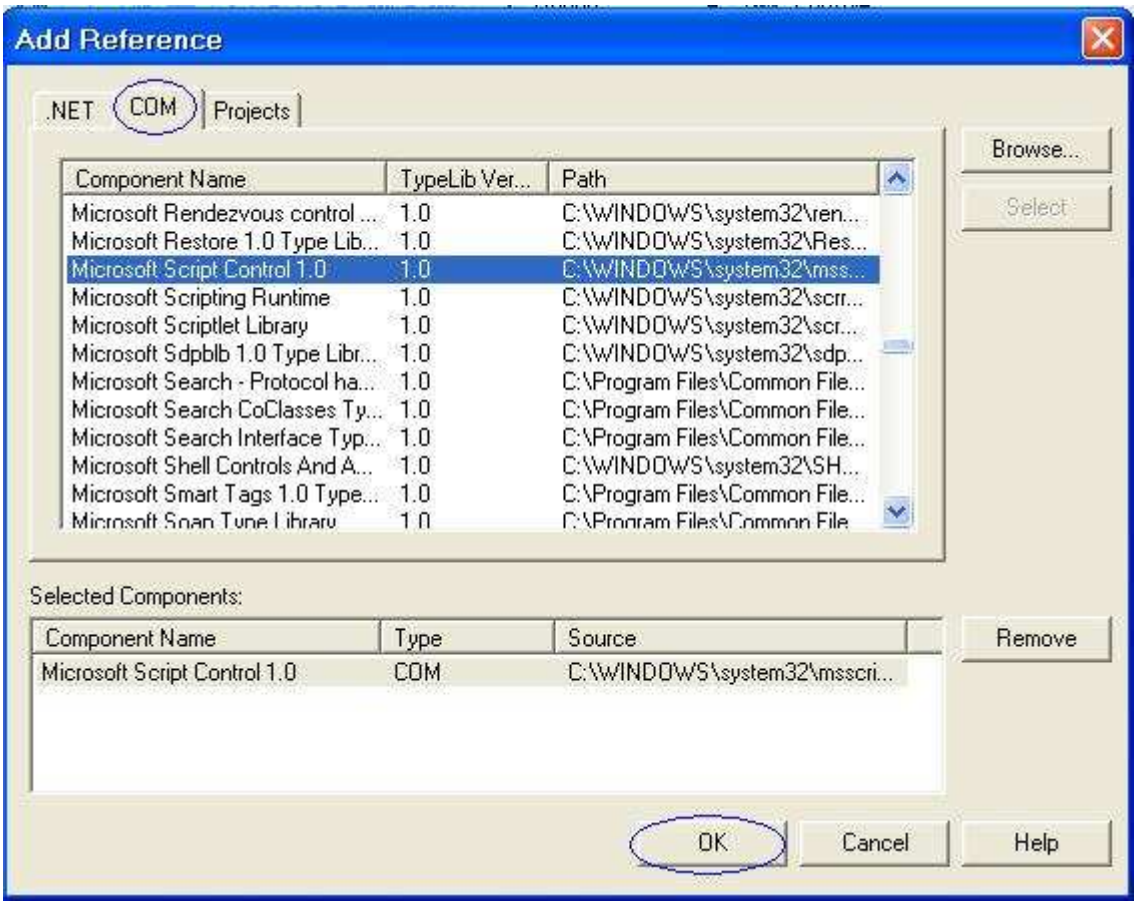
السطر :

```
static public MSScriptControl.ScriptControl re= new MSScriptControl.ScriptControl(
```

يقوم بتعريف كائن من الفئة ScriptControl التابعة لمكتبة الربط الديناميكي dll المسماة MSScriptControl وهي مكتبة عبارة عن مفسر للغة النصوص كـ VBScript وفيها الدالة Eval والتي

تستقبل سلسلة نصية عبارة عن عملية رياضية وتقوم بإرجاع ناتج العملية كما تتعرف على عمليات المقارنة ، وهذه الدالة هي مرادنا من هذه المكتبة وسميت الكائن re اختصاراً لـ result.

وقبل تعريف الكائن يجب إضافة المكتبة إلى مراجع المشروع والصور توضح كيفية القيام بذلك :



السطر:

```
static public string[] iname,ivalue,code;
```

في هذا السطر يتم تعريف ثلاث مصفوفات نصية كمتغيرات عامة Global ، المصفوفة iname سنخزن فيها أسماء المتغيرات الصحيحة التي سيتم تعريفها ، والمصفوفة ivalue سنخزن فيها القيم التي ستعطى

للمتغيرات الصحيحة ، والمصفوفة code سيتم فيها تخزين الكود المصدري المستقبل من الملف المصدري كما سيوضح في حينه.

السطر:

```
static bool bo=false;
```

يعرف متغيرا منطقيا . هذا المتغير سنعرف به إن تم الإعلان عن متغيرات صحيحة أم لا .

السطر :

```
static public int end=0;
```

يعرف المتغير العام الصحيح end وإعطائه القيمة 0 ، سيتم ذكر وظيفته فما بعد .

السطر:

```
static void readfile(string filename)
```

هو بداية الدالة readfile وهي التي تقوم باستقبال الكود من الملف المصدري وهي لا تقوم بإرجاع قيمة وتستقبل متغيرا نصياً هو اسم ملف المصدر .

السطر:

```
string allcode=null;
```

يعرف متغيراً صحيحاً ، هذا المتغير سنضع فيه كل الكود المستخرج من الملف المصدري.

السطر:

```
StreamReader read=new StreamReader(filename);
```

في هذا السطر تم تعريف الكائن read من الفئة StreamReader ، هذا الكائن هو الذي سيقوم بالقراءة من الملف المصدري .

السطر:

```
string line;
```

يعرف متغيراً نصياً لنقرأ به الملف المصدري سطرً سطراً.

الحلقة :

```
while( (line = read.ReadLine())!=null)
   allcode+=line+' ';
```

تقوم بقراءة الملف المصدري سطراً سطراً وإضافته مع فراغ إلى المتغير allcode مادامت القراءة لم تصل إلى نهاية الملف .

السطر:

```
allcode=allcode.TrimEnd();
```

وظيفته واضحة.

الشرط:

```
if(allcode[allcode.Length-1]==';')
  allcode=allcode.Remove(allcode.Length-1,1);
```

يقوم بحذف الفاصلة المنقوطة من نهاية المتغير allcode إن وجدت .

السطران:

```
code=new string[allcode.Split(';').Length];
code=allcode.Split(';');
```

يقومان بإعطاء المصفوفة code الحجم الناتج من تقسيم المتغير allcode عند أماكن وجود الفاصلة المنقوطة ، وإعطائها القيمة الناتجة من تجزيء المتغير كما شرح سابقاً عن الدالة Split() .

وفي الدالة الرئيسية يقوم السطر :

```
re.Language="VBscript";
```

بتهيئة re وذلك بإعطائها لغة VBscript كمقياس للتركيب اللغوي لتستعملها عند استخدامها.

الجملة :

```
if(args.Length>0)
  readfile(args[0]);
```

تختبر إن كانت طول المصفوفة args - والتي هي معامل للدالة الرئيسية – أكبر من صفر – أي تم تمرير معاملات للبرنامج – فإن كان كذلك فإنها تقوم بتمرير أول عنصر في المصفوفة إلى الدالة readfile أي أن أول عنصر سيكون اسم الملف المصدري الذي سيتم تفسيره وتنفيذه .

الدالة:

```
static public  string nospace(string s)
{
  for(int i=0;i<s.Length;i++)
      if(s[i]==' ')
        s=s.Remove(i,1);
```

```
  return s;
}
```

تقوم باستقبال سلسلة نصية وإعادتها بعد إزالة كل الفراغات منها .

السطر:

```
static void First()
```

بداية الدالة First وهي الدالة الرئيسية للمفسر والعمل كله يتم فيها وفي الدالة Second .

اللافتة : begin: سنعرف وظيفتها فيما بعد .

السطران:

```
int j=0,k;
string s=null;
```

نعرف فيه المتغيرين الصحيحين j و k ، والمتغير النصي s وتخصيص القيمة null له.

السطر:

```
while(j<code.Length)
```

بداية الحلقة الكبيرة والتي تقرأ كل جملة منتهية بالفاصلة المنقوطة في الملف الصدري لتفسيرها .

في السطرين :

```
int i;
s=code[j];
```

يتم تعريف المتغير i ، وإعطاء s العنصر ذي الدليل j في المصفوفة code والتي تم ذكرها سالفاً.

الحلقة :

```
for(i=0;i<s.Length;i++)
{
  if(s[i]=="")
      for(int jj=i+1;jj<s.Length-1;jj++)
          if(s[jj]=="")
          {
              i=jj+1;
              break;
```

```
            }
  if(s[i]==',')
  {
      s=s.Remove(i,1);
      s=s.Insert(i,"\\");
  }
}
```

تقوم باستبدال كل فاصلة ',' في المتغير s ليست ضمن نص مضمن بين علامتي تنصيص "" ، تقوم باستبدالها بالرمز '\' وذلك للفصل بين المتغيرات والنصوص وغيرها في جملة الإخراج ولو بقيت الفاصلة لحصل الالتباس للمفسر بين الفاصلة التي تفصل بين المتغيرات والنصوص وما سواها وبين الفاصلة المضمنة في نص ، ولأن كل فاصلة هكذا حالها فإن الرمز '\' سيفصل به أيضاً عند الإعلان عن المتغيرات الصحيحة وفي جملة الإدخال .

الـ for الداخلية تضمن أن الفاصلة التي سيتم استبدالها لا تقع في نص منصص (أي بين علامتي تنصيص ، وهي كلمة إيجازية بلاغية اشتققتها الآن !) .

هذه الـ for تفحص إذا كان العنصر s[i] قيمته " فإنها تقوم بزيادة عداد الحلقة وهو i إلى ما بعد علامة التنصيص الثانية والتي هي نهاية النص المنصص .

والسطر s=s.Trim(); وظيفته واضحة .

الكود :

```
if(!bo)
if(s.StartsWith("integer"))
{
  bo=true;
  s=s.Remove(0,8);
  iname=new string[s.Split('\\').Length];
  ivalue=new string[iname.Length];
  iname=s.Split('\\');
  string a=null;
```

```
  for(k=0;k<ivalue.Length;k++)a+="0,";
  a=a.Remove(a.Length-1,1);
  ivalue=a.Split(',');
}
```

أول سطر فيه يفحص قيمة المتغير المنطقي bo فإن كانت false فإنه ينفذ الجملة الشرطية التالية .

الجملة الشرطية تفحص ما إذا كانت قيمة المتغير s تبدأ بالكلمة integer ، فإن كانت كذلك ندخل إلى جسم الجملة ، وأول سطر يغير قيمة bo كما هو مبين.

السطر الثاني يحذف النص "integer " من المتغير s لكي يتم التعرف على المتغيرات التي تأتي بعده .

في السطر التالي يتم إعطاء المصفوفة iname الطول المبين والذي يساوي عدد مرات تكرار الرمز '\' في المتغير s .

في السطر التالي يتم إعطاء المصفوفة ivalue طول المصفوفة iname .

السطر التالي يقوم بتخصيص السلاسل الناتجة من تجزيء المتغير s بالدالة Split('\\') .

السطر التالي يعرف متغيراً نصياً بالاسم a .

الحلقة التالية تقوم بتخزين القيمة "0," بحسب عدد المتغيرات الصحيحة المعلن عنها ، وذلك لكي نعطي القيمة 0 تلقائياً للمتغيرات عند الإعلان عنها .

السطر التالي يقوم بحذف آخر فاصلة تم تخزينها في المتغير a بفعل السطر السابق .

السطر التالي هو الذي يعطى للمتغيرات الصحيحة المعلن عنها القيمة 0 كقيمة ابتدائية .

السطر :

```
if(s=="end")break;
```

يفحص ما إذا كان المتغير s يحتوي على الكلمة المفتاحية end وهي تقوم بإنهاء البرنامج من مكان وجودها في الكود ، فإن تحقق الشرط فإنها تخرج من الحلقة الكبيرة الأولى .

الشرط :

```
if(s=="repeat")
  goto begin;
```

يذهب بتنفيذ البرنامج إلى اللافتة begin: والتي هي أول سطر في الدالة First() ، هذا إذا كانت قيمة المتغير s هي repeat .

الآن نأتي لبرمجة جملة الإخراج print :

```
1    if(s.StartsWith("print"))
2   {
3      bool line=true;
4      s=s.Remove(0,5);
5      if(s=="")goto here;
6      s=s.Trim();
```

السطر رقم 1 واضح ، والسطر 3 يعرف متغيراً منطقياً سيتم استخدامه لمعرفة إن كان السطر البرمجي الحالي يحتوي عنصره قبل الأخير على الفاصلة حتى لا يتم طباعة سطر جديد .

السطر 4 يقوم بإزالة الكلمة print من المتغير s .

السطر 5 يختبر ما إذا كانت قيمة المتغير s أصبحت تساوي "" ، فإن كانت كذلك فإن هذا يعني أن المراد طباعة سطر جديد لا غير ، ويتم تنفيذ ذلك بالذهاب للافتة here الآتي ذكرها ، والسطر6 واضح.

```
7      if(s[s.Length-1]=='\\')
8     {
9       line=false;
10        s=s.Remove(s.Length-1,1);
11    }
```

السطر 7 يختبر إن كان المتغير s ينتهي بالرمز '\' – أي أن السطر المناظر له في الملف المصدري ينتهي بفاصلة – فإن كان فإن قيمة المتغير line تصبح false ويتم إزالة '\' من نهاية المتغير .

```
12    if(s.StartsWith("print")){Second(s);  goto again;}
13    string[] ar=new string [s.Split('\\').Length];
14    ar=s.Split('\\');
```

في السطر 12 إن كانت s تبدأ بـ print فإنه يتم استدعاء الدالة Second() والتي سيتم ذكرها فيما بعد ، وبعد ذلك يتم الرجوع إلى بداية الحلقة الكبرى باستخدام بالذهاب إلى اللافتة again والواقعة في آخر جسم الحلقة ، وذلك لئلا يتم فحص شروط بعد جملة الإخراج لا فائدة من فحصها حينها ، وهذا يزيد من سرعة التفسير .

السطر 13 يعرف مصفوفة نصية ar بالطول المبين ، والسطر 14 يعطيها القيمة الموضحة .

```
15    for(i=0;i<ar.Length;i++)
```

```
16  {
17   if(ar[i][0]==""&& i<=ar.Length)
18   {
19          ar[i]=ar[i].Remove(0,1);
20        ar[i]=ar[i].Remove(ar[i].Length-1,1);
21   }
22  else
23  {
24    if(bo)
25
26      for(int h=0;h<ivalue.Length;h++)
27          ar[i]=ar[i].Replace(iname[h],ivalue[h]);
28    ar[i]=re.Eval(ar[i]).ToString();
29  }
30  Console.Write(ar[i]);
31 }
```

السطر 15 بداية الحلقة التي ستعمل على عناصر المصفوفة ar.

الجملة الشرطية التي تبدأ في السطر 17 تفحص ما إذا كان الرمز الموجود في ar[i][0] هو علامة تنصيص مزدوجة ، فإن كان فإنها تقوم بحذف العلامة من بداية النص ar[i] ومن نهايته ، بالطبع إن كان يحتوي على علامة تنصيص في البداية فينبغي أن يحتويها في النهاية وإن لم يكن فإنه سيحذف آخر حرف.

أما إن لم تكن ar[i] محتوية على علامة تنصيص فإنه يتم تنفيذ كتلة else .

السطر 24 يفحص قيمة bo فإن كانت true فهذا يعني أنه تم الإعلان عن متغيرات صحيحة ومن ثم يتم تنفيذ for التالية والتي تعمل على المصفوفتين iname و ivalue ، وعمل الحلقة في أنه إن كان موجوداً في قيمة العنصر ar[i] - والذي هو سلسلة نصية - إن كان يوجد فيه أحد أسماء المتغيرات الصحيحة التي تم الإعلان عنها فإنه يتم استبدال اسم المتغير بقيمته المخزنة في العنصر المناظر له في المصفوفة ivalue .

في السطر 28 يتم إيجاد قيمة التعبير الرياضي الذي هو قيمة العنصر ar[i] وذلك باستخدام الدالة Eval .

في السطر 30 يتم طباعة السلسلة المخزنة في ar[i] والتي أريد إخراجها بالجملة print ، وتتم الطباعة دون الانتقال إلى سطر جديد .

```
32    here:
33    if(line)
34    Console.WriteLine();
35   goto again;
36    }
```

السطر 32 يحتوي على اللافتة here المذكورة سابقاً والتي يتم المجيء إليها إن كان المراد طباعة سطر جديد فقط .

في السطر 33 إذا تحقق الشرط والذي سيكون متحققاً إن لم يكن الحرف ما قبل الأخير في جملة الإخراج هو الفاصلة أي أنه يراد الانتقال إلى سطر جديد ، فإن كان كذلك فإنه يتم طباعة سطر جديد .

السطر 35 مشروح مثيله سابقاً.

والآن نأتي لبرمجة جملة تخصيص القيم للمتغيرات :

```
if(s.IndexOf("")<0&&s.IndexOf("if")<0&&s.IndexOf('=')>0&&s.IndexOf("for")<0)
{
  s=nospace(s);
  string[] eq=new string[s.Split('\\').Length];
  eq=s.Split('\\');
  for(i=0;i<eq.Length;i++)
  {
      for(int h=0;h<iname.Length;h++)
        if(eq[i].StartsWith(iname[h]))
        {
            eq[i]=eq[i].Remove(0,eq[i].IndexOf('=')+1);
              for(int l=0;l<ivalue.Length;l++)
                 eq[i]=eq[i].Replace(iname[l],ivalue[l]);
            eq[i]=re.Eval(eq[i]).ToString();
```

```
            ivalue[h]=(Convert.ToInt32(Convert.ToDouble(eq[i]))).ToString();
          break;
        }
    }
  goto again;
}
```

السطر الأول يختبر ما إذا كانت قيمة s لا تحتوي على علامة تنصيص ولا تحتوي على الكلمة المفتاحية if ولا for وكانت تحتوي على علامة = ، فإن كانت s كذلك فهذا يعني أن الجملة جملة تخصيص ويتم الدخول في كتلة الشرط .

السطر الأول في الكتلة يزيل كل الفراغات من s ، والسطر الثاني يعرف مصفوفة نصية طولها عدد مرات تكرار '\' في s .

السطر الثالث يخصص القيمة المبينة لـ eq .

الحلقة التي في السطر التالي تعمل على عناصر المصفوفة eq ، و for الداخلية تعمل على عناصر المصفوفة iname .

جملة if تختبر ما إذا كان النص المخزن في العنصر eq[i] يبدأ بالنص المخزن في iname[h] ، أي أنه يراد تخصيص قيمة للمتغير المخزن اسمه في iname[h] . فإن كان ذلك يتم الدخول إلى كتلة الشرط والتي أول سطر فيها يقوم بحذف الحروف من البداية إلى علامة = مع العلامة أيضاً وهذا النص هو المحتوى في العنصر eq[i] ، أي أن هذه الجملة تبقي على القيمة الرقمية المارد تخصيصها فقط في العنصر eq[i] .

ولأنه ربما القيمة المخصصة ليست قيمة رقمية مباشرة بل قيمة متغير آخر لوحده أو مضافة له قيمة أخرى أو غيره فإن الحلقة التي في السطر الثاني تقوم باستبدال اسم كل متغير موجود في القيمة المخصصة بقيمته الموجودة في المصفوفة ivalue .

في السطر الثاني يتم حساب ناتج القيمة المخصصة وذلك لأنها كما ذكر ربما تكون تعبيراً رياضياً وليست قيمة مباشرة ، ويتم تخزينها في العنصر eq[i] نفسه .

ولأن ناتج التعبير الرياضي الذي تم حسابه في السطر السابق ربما يكون كسراً وليس عدداً صحيحاً ، ولأن متغيراتنا هي أعداد صحيحة فإن السطر التالي والذي قبل جملة break يقوم بتحويل القيمة من كسر إن كانت إلى عدد صحيح ويخزنها في العنصر ivalue[h] والذي يخزن قيمة المتغير المخزن اسمه في العنصر iname[h] ، وبهذه الكيفية تتم عملية التخصيص .

جملة break تقوم بالخروج من الحلقة الداخلية وذلك لتسرع العملية وذلك لأنه في العنصر eq[i] لن يوجد إلا متغير واحد يراد تخصيص قيمة له ، لذا فإن تم تخصيص القيمة له فلا حاجة لتكرار العملية مرة أخرى .

والآن إلى جملة الإدخال :

```
if(s.StartsWith("enter "))
{
    s=s.Remove(0,6);
    s=nospace(s);
    string[] eq=new string[s.Split('\\').Length];
    eq=s.Split('\\');
    for(i=0;i<eq.Length;i++)
   {
       for(int h=0;h<iname.Length;h++)
        {
         if(eq[i]==iname[h])
         {
       ivalue[h]=Console.ReadLine();
         for(int l=0;l<ivalue.Length;l++)
          ivalue[h]=ivalue[h].Replace(iname[l],ivalue[l]);
       ivalue[h]=re.Eval(ivalue[h]).ToString();
       ivalue[h]=Convert.ToInt32(Convert.ToDouble(ivalue[h])).ToString();
         break;
           }
       }
  }
  goto again;
}
```

الشرط واضح وكذلك الأسطر الأربعة الأولى في كتلة الشرط.

الحلقتين تقومان مع جملة الشرط باختبار قيمة eq[i] فإن كانت أحد المتغيرات المعلن عنها فإنه يتم إدخال قيمةٍ له وتخزينها في العنصر المناظر وهو ivalue[i] والكود واضح .

مع ملاحظة أن القيمة المدخلة يمكن أن تكون تعبيراً رياضياً بل حتى متغيراً أو متغيراً في تعبير رياضي وهذه الخاصية لا توجد في اللغات الأخرى – يا لمرونة هذه اللغة ! –.

جملة if :

```
if(s.StartsWith("if"))
{
  int l=0;
 for(i=2;i<s.Length;i++)
 {
   if(s[i]=='(')
   l++;
   if(s[i]==')')l--;
   if(s[i]==')'& l==0)
     {
            i++;
      break;
     }
 }
```

سأشرح الكود السابق بافتراض أن s تحتوي عل جملة شرطية لأبين بصورة واضحة ما الذي يحدث ، وهذه الجملة هي :

```
if ((a>b) and (b>c)) print a else print c,"  ",b;
```

الجملة الشرطية تختبر ما إذا كانت قيمة s تبدأ بـ if فإن كانت كذلك يتم الدخول إلى كتلة الشرط وفي أولها يتم تعريف متغير صحيح وإعطائه القيمة الابتدائية 0 ، أما في السطر التالي فإنه يتم بناء حلقة تكرارية قيمة عدادها i تبدأ من 2 وهو رقم ترتيب أول عنصر في s بعد كلمة if .

أول سطر في الحلقة يختبر ما إذا كانت s[i]='(' فإن كانت فإنه يزيد قيمة المتغير l بمقدار واحد صحيح ، وما دامت قيمة المتغير l لا تساوي 0 فإن هذا يعني أنه ثمة قوس '(' ناقص وبالتالي يجب وجوده .

وفي الجملة التي لدينا فإن قيمة l ستزداد أو تصبح واحد إذ لا فرق في أول مرة عندما i = 3 لأن العنصر s[3]='(' ، وفي الجملة الشرطية التالية يتم اختبار ما إن كانت قيمة s[3]=')' وبالطبع هذا خاطئ ، والجملة الشرطية التالية هي أيضاً لن تنفّذ ، وفي هذه الجملة يتم اختبار ما إذا كانت قيمة s[i]=')' وقيمة l=0 وهذا لن يحدث إلا إذا وصلنا إلى آخر قوس في الشرط الذي في الجملة .

والآن لنعد إلى الحلقة ، في التكرار التالي للتكرار السابق أصبحت قيمة i= 4 ومن ثم في أول جملة شرطية يتم يتم زيادة المتغير l لتصبح 2 وذلك لأن s[4]=')' أما باق الجمل الشرطية فلن تنفذ ويتم الرجوع إلى الحلقة .

عندما تصبح قيمة i = 8 فإن شرط الجملة الشرطية الأولى لن يتحقق أما الجملة الثانية فإن شرطها متحقق وذلك لأن s[8]=')' وبالتالي فإن قيمة المتغير l ستنقص بمقدار 1 ،أما شرط الجملة الأخيرة فلن يتحق لأن قيمة i الآن هي 1 وليس 0 .

عندما تصبح قيمة i = 14 فإن شرط الجملة الأولى صحيح وبالتالي تزداد قيمة l وتصبح 2 ، وعندما تصبح قيمة i= 18 فإن شرط الجملة الثانية سيتحقق وبذلك تنقص قيمة i لتصبح 1 ، أما عندما تصبح i = 19 فإن الجملة الشرطية الثانية ستنقص قيمة i لتصبح 0 وذلك لأن s[19]=')' وبالتالي فإن شرط جملة الشرط الثالثة أو الأخيرة سيتحقق وعند تحققه فإننا نزيد قيمة i لتصبح 20 ثم نخرج من الحلقة باستخدام break .

والكود التالي للسابق هو :

```
string aelse="",g=s.Remove(0,i);
s=s.Remove(i,s.Length-i);
s=s.Remove(0,2);
s=s.TrimStart();
```

في أول سطر يتم تعريف متغيرين نصيين المتغير الأول aelse سنستخدمه فيما بعد وقد أعطيناه القيمة الابتدائية "" ، أما المتغير g فقد أعطيناه قيمة s مزال منها 20 من البداية أي أن قيمة g هي print a else . print c," ",b;

في السطر الثالث يتم إزالة الجملة التي بعد آخر قوس من الشرط في الجملة ، وفي السطر التالي يتم إزالة الكلمة if من s ، وفي السطر التالي يتم إزالة الفراغات من بداية s ، أي أن قيمة s الآن هي ((a>b) and (b>c))

```
if(bo==true)
for(k=0;k<s.Length;k++)
  for(int h=0;h<ivalue.Length;h++)
    s=s.Replace(iname[h],ivalue[h]);
bool lo=Convert.ToBoolean(re.Eval(s).ToString());
```

في الأسطر الأربعة الأولى يتم التحقق إن تم الإعلان عن متغيرات صحيحة مسبقاً أم لا فإن تم فإنه يتم استبدال اسم كل متغير موجود في s بقيمته .

في السطر الخامس يتم إيجاد نتيجة الشرط هل هي صح أم خطأ باستخدام الدالة Eval() ، ويتم تحويل النتيجة إلى نص يتم تحويله إلى قيمة منطقية يتم إسنادها إلى المتغير lo المعلن عنه .

```
if(g.IndexOf("else")>0)
{
    aelse=g.Remove(0,g.IndexOf("else")+4);
    g=g.Remove(g.IndexOf("else"),g.Length-g.IndexOf("else"));
}
if(lo)
Second(g);
else
if(aelse!="")
Second(aelse);
if(end==1)goto end;
goto again;
}
```

في السطر الأول يتم اختبار وجود كلمة else في المتغير g ولأنها موجودة في الجملة التي لدينا فإنه يتم الدخول إلى كتلة الشرط والتي تقوم بإسناد النص الموجود بعد كلمة else إلى المتغير aelsae لتصبح قيمته : print c," ",b .

وفي السطر التالي يتم إزالة جزء من قيمة g بداية من else إلى النهاية ، لتصبح قيمة g : print a

السطر if(lo) يختبر قيمة lo والتي هي قيمة الشرط الأصلي التابع لـ if التي لدينا ، فإن كان الشرط صحيحاً فإنه يتم تمرير قيمة g إلى الدالة Second والتي سنتحدث عنها بعد قليل ، أما إن لم يتحقق الشرط فيتم التأكد من أن الكلمة else موجودة في الجملة الشرطية وهذا لا يكون إلا إن كانت قيمة المتغير aelse لا تساوي "" أي القيمة المسندة ابتدائياً له ، والتي لن تتغير إلا إن كانت else موجودة كما شرح قبل قليل ، فإن كان aelse لا تساوي "" فإنه يتم تمرير قيمته هو إلى الدالة Second .

السطر التالي يختبر قيمة end فإن كانت تساوي 1 فإنه يتم الذهاب إلى آخر سطر في الدالة First() لإنهاء تنفيذ البرنامج ، والقيمة تكون 1 في حالة ما إذا كانت قيمة g أو قيمة aelse هي "end" والتي ستمرر إلى Second() التي ستجعل القيمة 1 حيث أن السطر if(s=="end")break; في First() يقابله السطر if(s=="end")end=1; في Second()

الدالة Second() مكتوبة بعد الدالة First() ، وفكرتها أتتني أثناء برمجة اللغة عندما وصلت إلى برمجة if ، وهي فكرة مهمة جداً ، بل لولاها لم تكن لغة Learn على هذا المستوي على الإطلاق ، ولكان الكود معقداً إلى درجة لا تطاق مع نقص كبير في المرونة .

دعنا نفترض أنه ليس لدينا الدالة Second() ، ونفترض أن الشرط صحيح ونريد تنفيذ الجملة التي لدينا وهي هاهنا print a فما الذي سنفعله ، وبالطبع الجملة التي لدينا قد برمجنا مثلها من قبلة ، أي عندما تحدثنا عن جملة الإخراج ، لذا فإنه بإمكاننا باستخدام goto مع الجمل الشرطية أن نذهب إلى بداية كود جملة الإخراج بالجملة print a – وهذه الطريقة سيئة جداً ولا تُحتمل حتَى ، والتعقيد فيها لا يخفى وما هو أعقد في الطريق – ، وهكذا سنفعل إن لم يتحقق الشرط ووجدت else ، ولكن ماذا سيحدث إن كانت لدينا الجملة :

if((a>b) and (a>c)) print a else if((b>a) and (b>c)) print b else print c;

الذي سيحدث أنه إذا لم يتحقق الشرط فإننا سنذهب بالجملة التي بعد else ليس إلى الجزء المتعلق بجملة الإدخال ؛ ولكن إلى المتعلق بجملة الشرط ، أي الجزء الذي نحن فيه الآن ، والنتيجة أنه سيحدث تداخل يؤدي إلى إرباك المفسر المسكين وإلى ارتفاع ضغطه ارتفاعاً حاداً سيفقده وعيه، وبالطبع لن يتم تنفيذ الكود وستظهر رسالة خطأ من مكان ما ، ربما إطار دوت نت وربما النظام ، وقد استعملت هذه الطريقة في

الإصدار الأخير من لغة سماء والنتيجة أن عدد أسطر كودها تقارب ضعف عدد أسطر Learn ، أما في المرونة فنصف Learn أو أقل ، والأهم من ذلك أن تطوير Learn سهل جداً.

الآن لنفكر في طريقة أخرى ، هيّا فكر قليلاً ، ولأنشط لك عقلك سأسألك سؤالاً يحتاج إلى تشغيل مخ ألا وهو : 1 + 2 = ؟ !!!.

حسناً الطريقة هي كالتالي : ماذا لو قمنا باستدعاء الدالة First() ، أعلم أننا في داخلها ولكن بالتأكيد لا مشكلة في استدعائها لنفسها، ونقوم باستدعائها لأن الدالة كما مر بنا تقوم بمعالجة كل الجمل المشابهة للتي لدينا ، ولكن الدالة First() تقوم بالعمل على عناصر المصفوفة code أما الذي لدينا فهو جملة برمجية واحدة ، لذا فإننا سنقوم ببناء دالة أخرى كودها مماثل لكود First() ، والفرق أنها تقوم باستقبال متغير نصي كمعامل لها ثم تقوم بما سبق شرحه من معالجة له ، وبالتالي يتم تفسير الجملة وتنفيذها ، وفي جملة if الطويلة السابقة فإنه على افتراض إرسال الجملة :

```
if((b>a) and (b>c)) print b else print c;
```

إلى الدالة Second فإنه سيتم معالجة جملة if في داخل Second() كما شرح قبل قليل في الدالة First() ، فإذا تحقق الشرط فسيتم استدعاء الدالة Second لنفسها مستقبلة النص print b والذي سيتم تنفيذه بلا أي مشاكل ، ومما سبق يتضح لنا لماذا جملة برمجية مثل print print print "Welcome to Learn language"; هي جملة صحيحة ، لأنه بعد إزالة print الأولى سيتم استدعاء Second() وإرسال ما تبقى من الجملة إليها وهي بالتالي ستزيل print الثانية ثم ستستدعي نفسها مرسلة ما تبقى مرة أخرى ، ومن ثم سيتم طباعة الجملة المراد طباعتها ،وهذه هي فكرة وعمل الدالة Second() .

كما أن الفرق الثاني لـ Second() عن First() هو أنها لا تحتوي على جمل تعريف المتغيرات الصحيحة وهذا شيء بدهي .

جملة for :

سنفترض هنا أيضاً أن لدينا الجملة البرمجية التالية :

```
for n=a to b step 2 do print n;
```

باعتبار أن a و b متغيرات صحية وأن قيمة a = 1 و b = 10 وسنتتبع الكود وفقاً لهذه الجملة :

```
if(s.StartsWith("for"))
{
  s=s.Remove(0,3);
  string b=s;
  b=b.Remove(0,b.IndexOf('=')+1);
```

```
  string now=s.Remove(s.IndexOf('='),s.Length-s.IndexOf('='));
  now=nospace(now);
  int know;
  for(i=0;i<iname.Length;i++)
  if(iname[i]==now)
 {
    know=i;
    break;
}
```

السطر الأول واضح ، وإذا تحقق الشرط فإنه بعد الدخول إلى الكتلة يتم إزالة for من s ، وفي السطر التالي يتم تعريف متغير نصي ويتم إسناد قيمة s إليه ، ثم في السطر التالي يتم إزالة جزء من النص الذي هو قيمة b ، هذا الجزء من بداية النص إلى علامة = ، أي أن قيمة b الآن هي:

"a to b step 2 do print n;"

في السطر التالي يُعرف المتغير النصي now مسندةً له قيمة s مزال منها من علامة = إلى النهاية ، أي أن قيمة now هي : " n" ، وفي السطر التالي يتم إزالة كل فراغ منها لتصبح "n" ، أي أن قيمة now هي اسم المتغير الذي سيعمل كعداد للحلقة .

في السطر التالي نعرف المتغير know.

في الحلقة يتم البحث في المصفوفة iname المخزنة فيها أسماء المتغيرات ، فإن كان أحد الأسماء يساوي قيمة المتغير now فإنه يتم تخصيص قيمة i إلى know ، ثم يتم الخروج من الحلقة .

```
string one=b.Remove(0,b.IndexOf(" do"));
b=b.Remove(b.IndexOf("do "),b.Length-b.IndexOf("do "));
for(k=0;k<iname.Length;k++)
   if(b.IndexOf(iname[k]+' ')>=0)
     b=b.Replace(iname[k]+' ',ivalue[k]+' ');
b+=one;
```

في الكود السابق يتم إسناد قيمة b إلى المتغير one بعد إزالة جزء منها من البداية إلى " do" ، أي قيمة one هي : " do print n;"، وفي السطر التالي يتم إزالة نص من b من كلمة do إلى نهاية النص المخزن في b لتصبح b = "a to b step 2 "، أما حلقة التكرار التالية فتقوم باستبدال كل اسم متغير موجود في b بقيمته ، أن قيمة b ستصبح "1 to 10 step 2 " .

في السطر الأخير يتم إضافة قيمة one إلى b ، و عملية تقسيم b ثم إعادة ربطها تمت لكي لا يتم تغيير أسماء المتغيرات بعد do إن وُجدوا بقيمهم ، ولو تم ذلك فإن الحلقة التي لدينا لن تطبع الأعداد من 1 إلى 10 ، بل ستطبع قيمة ثابتة في كل تكرار .

```
int incdec=1,del=0;
if(s.IndexOf("step")>0)
{
   string step=s.Remove(0,s.IndexOf("step")+4);
  step=step.TrimStart();
  step=step.Remove(step.IndexOf(' '),step.Length-step.IndexOf(' '));
  incdec=Convert.ToInt32(step);
  for(i=b.IndexOf("step");i<b.IndexOf("do");i++)
        del++;
  b=b.Remove(b.IndexOf("step"),del);
 }
```

السطر الأول واضح ، ووظيفة المتغير incdec حفظ مقدار الزيادة والذي يأتي بعد الكلمة step وقيمته الافتراضية هي 1 .

الجملة الشرطية في السطر الثاني واضحة ، وإذا تحقق الشرط فإنه يتم تعريف المتغير step وإسناد القيمة المبينة له ،وكذلك بقية الأسطر واضحة ، ووظيفة حلقة التكرار معرفة عدد الحروف من كلمة step في الجملة إلى كلمة do ليتم في السطر التالي إزالة هذه الحروف من b ، ومن ثم فإن قيمة b الآن : "1 to 10 do print a;" .

```
if(b.IndexOf("do")>0)
  b=b.Remove(b.IndexOf("do"),2);
else {Console.WriteLine("Error...(Missing \" do \")");goto end;}
s=s.Remove(s.IndexOf('=')+1,s.Length-(s.IndexOf('=')+1));
s+=b;
string a=s;
a=a.Remove(a.IndexOf("to"),a.Length-a.IndexOf("to"));
a=nospace(a);
```

الغرض من جملة الشرط التأكد من وجود كلمة "do" في جملة التكرار التي نعالجها الآن ، فإن وجدت تتم حذف الكلمة من قيمة b ، وإن لم توجد فإنه يتم إظهار رسالة خطأ تبين السبب ثم يتم إنهاء البرنامج ، وهذه طريقة يمكن بها معالجة الأخطاء الموجودة في الكود المصدري ، وهذه الجملة الوحيدة في لغة Learn تعالج الأخطاء ولم أبرمج جملاً ، إذ أن المهم أن تكون الصورة واضحة في كيفية التعامل مع مثل هذه الحالات ، كما

أن استعمال الجمل الخاصة بالاستثناءات المبينة في اللغة التي يتم بها بناء المفسر طريقة أخرى للتعامل مع الأخطاء.

في السطر الذي يتلو سطر else يتم حذف جزء من قيمة s والتي هي :

" n=a to b step 2 do print n;"

من الحرف الذي بعد علامة = إلى نهاية s لتصبح : " n=" ، وفي السطر التالي تضاف إليها قيمة b، لتصبح s= "n=1 to 10 print n" .

في السطر التالي تُسند قيمة s إلى المتغير النصي a الذي تم تعريفه ، والسطر التالي والذي بعده واضحان ، ومحصلتهما أن قيمة a ستصبح "n=1".

```
for(int h=0;h<iname.Length;h++)
  if(a.StartsWith(iname[h]))
 {
   a=a.Remove(0,a.IndexOf('=')+1);
   ivalue[h]=a;
   a=s;
   s=s.Remove(0,s.IndexOf("to")+2);
   a=s;
   s=s.TrimStart();
   s=s.Remove(s.IndexOf(' '),s.Length-s.IndexOf(' '));
   a=a.TrimStart();
   a=a.Remove(0,a.IndexOf(' '));
```

حلقة التكرار تبحث أي المتغيرات الصحيحة المعرفة مسبقاً والموجود اسمه في أحد عناصر المصفوفة iname هو عداد الحلقة ، فإذا وجدته فإن أول سطر في كتلة الشرط يقوم بحذف اسم المتغير مع علامة = من a لتبح قيمته = "1" ، ثم يتم إسنادها لقيمة المتغير العداد المناظرة له في المصفوفة ivalue والتي هي الآن ivalue[h] والدليل h سبقى معنا إلى حين الانتهاء من حلقة التكرار، أما باقي السطور فهي لا شك واضحة وفي نهايتها فإن قيمة a تصبح "print n" وهي الجملة المراد تكرارها .

```
   int loop;
  for(loop=Convert.ToInt32(ivalue[h]);loop<=Convert.ToInt32(s);loop+=incdec)
 {
    Second(a);
    ivalue[h]=ivalue[h].Remove(0,ivalue[h].Length);
    ivalue[h]=ivalue[h].Insert(0,(loop+incdec).ToString());
  }
  incdec=1;
```

```
 ivalue[h]=ivalue[h].Remove(0,ivalue[h].Length);
 ivalue[h]=ivalue[h].Insert(0,(loop-1).ToString());
 break;
}

}
```

في أول سطر يتم تعريف المتغير الصحيح loop هذا المتغير سيكون عداد الحلقة التي ستكرر الجملة المراد تكرارها ، ولم يتم تعريفه داخل الحلقة لأننا نريد استعماله خارجها بعد انتهائها في أول سطر يتم استدعاء الدالة Second وإرسال قيمة a إليها وذلك لأن الجملة التي نريد تكرارها لا بد أن تكون إحدى الجمل التي مرت بنا كجملة إخراج أو جملة إدخال أو جملة شرطية أو حلقة تكرارية أخرى ، وهي هاهنا جملة إخراج .

السطر الثاني في الحلقة يقوم بإفراغ المتغير المناظر للعداد n من قيمته الحالية 1 ليصبح متغيراً فارغاً، وفي السطر الثالث يتم إسناد قيمة عداد الحلقة أي قيمة loop مضافاً إليها متغير مقدار الزيادة إلى ivalue[h] الذي قمنا بإفراغه في السطر السابق ، فتصبح قيمته الآن 3 ،وعند تكرار الحلقة وإرسال الجملة "print a" إلى الدالة Second يتم طباعة 3 وهي القيمة الجديدة لـ ivalue[h] وذلك لأن مقدار الزيادة 2 ، وهذه هي طريقتي في برمجة عداد الحلقات التكرارية ، والحلقة التكرارية بشكل كامل .

السطر الأول خارج الحلقة السابقة يقوم بإعادة القيمة الافتراضية 1 إلى متغير مقدار الزيادة فإن لم يتم إعادتها وكان في البرنامج جملة تكرار أخرى ولم يتم إيضاح مقدار الزيادة لها فهذا يعني أن المبرمج يريدها 1 ، ولكن من الحلقة الحالية فإنها 2 ولن ترجع 1 إلا بهذا السطر .

في السطرين التاليين يتم تخزين قيمة آخر مدى للتكرار والتي هي عندنا 10 إلى قيمة المتغير n بعد الخروج من الحلقة ، وفي السطر الأخير يتم الخروج من حلقة التكرار الأولى بدون تكرارها وذلك لأنه لا يمكن أن يوجد أكثر من عداد للحلقة .

```
 again:;
 j++;
}
end:;
Console.WriteLine("\nPress Enter to exit...");
Console.Read();
}
```

وهذه الأسطر الأخيرة تبين موقع اللافتة again ،أما السطر التالي فهو مقدار الزيادة لحلقة التكرار الكبرى .

وفي السطر التالي موقع اللافتة end والتي بالذهاب إليها يتم انتهاء البرنامج ، والسطران التاليان واضحان .

وبهذا ننتهي من شرح بناء المفسرات على طريقتي المبيَّنة .

وفي النهاية أريد القول أن هذا المجال البرمجي واسع وعريض وليس له حدود ككل المجالات البرمجية بل هو أكثر اتساعاً وإمتاعاً ، كما أن هذه الطريقة في بناء المفسرات ليست بالمعقدة وتمنحنا كوداً بسيطاً سهل التطوير للوصول بالمفسر إلى درجة متقدمة،وهذا ما سأفعله مع لغة Learn حتى تصبح لغة مرموقة بإذن الله .

وللغة صفحة خاصة في موقع VC4ARAB في القسم الخاص ببناء مترجمات لغات البرمجة ، وبالتأكيد فلكل من يريدُ الحقَ في تطويرها والاستفادة منها بل أرحب بذلك كثيراً، بشرط أن يضع المطور التطوير الذي برمجه وتوصل إليه في الصفحة المذكورة ، وأهم التطويرات التي تحتاجها اللغة هي إضافة إمكانية تعريف متغيرات حقيقية ونصية ، ثم تأتي إضافة المصفوفات فالدوال .

الملحق أ

الكود الكامل للغة Learn

```
//////////////////////////////////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
using System;
using System.IO;

namespace Learn
{

	class Class1
	{
		static	public  MSScriptControl.ScriptControl re= new MSScriptControl.ScriptControl();
		static public string[] iname,ivalue,code;
		static bool bo=false;
		static public int end=0;
		/// <summary>
		/// Learn language .
		/// Version 0.1 .
		/// Programmed by: Kahlil Alamin Abduljawad
		/// Khal_i_l@Yahoo.com
		/// 7/15/2007
		/// </summary>
		[STAThread]
		static void readfile(string filename)
		{

			string allcode=null;
			StreamReader read=new StreamReader(filename);
			string line;
			while( (line = read.ReadLine())!=null)
				allcode+=line+' ';
			read.Close();
			allcode=allcode.TrimEnd();
			if(allcode[allcode.Length-1]==';')
				allcode=allcode.Remove(allcode.Length-1,1);
			code=new string[allcode.Split(';').Length];
			code=allcode.Split(';');

		}
		static void Main(string[] args)
		{
			re.Language="VBscript";
			if(args.Length>0)
			{
				readfile(args[0]);
				First();
			}
		}
		static public  string nospace(string s)
		{
			for(int i=0;i<s.Length;i++)
				if(s[i]==' ')
					s=s.Remove(i,1);
			return s;
		}


		// #################
```

```
static void First()
{
        begin:
        int j=0,k;
        string s=null;
        while(j<code.Length)
        {
                int i;
                s=code[j];
                for(i=0;i<s.Length;i++)
                {
                        if(s[i]=="")
                                for(int jj=i+1;jj<s.Length-1;jj++)
                                        if(s[jj]=="")
                                        {
                                                i=jj+1;
                                                break;
                                        }
                        if(s[i]==',')
                        {
                                s=s.Remove(i,1);
                                s=s.Insert(i,"\\");
                        }
                }
                s=s.Trim();
                if(!bo)
                if(s.StartsWith("integer"))
                {
                        bo=true;
                        s=s.Remove(0,8);
                        iname=new string[s.Split('\\').Length];
                        ivalue=new string[iname.Length];
                        iname=s.Split('\\');
                        string a=null;
                        for(k=0;k<ivalue.Length;k++)a+="0,";
                        a=a.Remove(a.Length-1,1);
                        ivalue=a.Split(',');
                }

                if(s=="end")break;
                if(s=="repeat")
                        goto begin;

                if(s.StartsWith("print"))
                {
                        bool line=true;
                        s=s.Remove(0,5);
                        if(s=="")goto here;
                        s=s.Trim();
                        if(s[s.Length-1]=='\\')
                        {
                                line=false;
                                s=s.Remove(s.Length-1,1);
                        }
                        if(s.StartsWith("print")){Second(s);goto again;}
                        string[] ar=new string [s.Split('\\').Length];
                        ar=s.Split('\\');
                        for(i=0;i<ar.Length;i++)
                        {
                                if(ar[i][0]==""&& i<=ar.Length)
                                {
                                        ar[i]=ar[i].Remove(0,1);
                                        ar[i]=ar[i].Remove(ar[i].Length-1,1);
```

```
                }
                else
                {
                        if(bo)
                                for(int h=0;h<ivalue.Length;h++)
                                  ar[i]=ar[i].Replace(iname[h],ivalue[h]);

                        ar[i]=re.Eval(ar[i]).ToString();
                }
                Console.Write(ar[i]);
        }
here:
        if(line)
        Console.WriteLine();
        goto again;
}

if(s.IndexOf("")<0 && s.IndexOf("if")<0 && s.IndexOf('=')>0 && s.IndexOf("for")<0)
{
        s=nospace(s);
        string[] eq=new string[s.Split('\\').Length];
        eq=s.Split('\\');
        for(i=0;i<eq.Length;i++)
        {
                for(int h=0;h<iname.Length;h++)
                        if(eq[i].StartsWith(iname[h]))
                        {
                                eq[i]=eq[i].Remove(0,eq[i].IndexOf('=')+1);
                                for(int l=0;l<ivalue.Length;l++)
                                        eq[i]=eq[i].Replace(iname[l],ivalue[l]);
                                eq[i]=re.Eval(eq[i]).ToString();
                                string hh=eq[i];
                        ivalue[h]=(Convert.ToInt32(Convert.ToDouble(eq[i]))).ToString();
                           break;
                        }
        }
        goto again;
}
if(s.StartsWith("enter "))
{
        s=s.Remove(0,6);
        s=nospace(s);
        string[] eq=new string[s.Split('\\').Length];
        eq=s.Split('\\');
        for(i=0;i<eq.Length;i++)
        {
                for(int h=0;h<iname.Length;h++)
                {
                        if(eq[i]==iname[h])
                        {
                                ivalue[h]=Console.ReadLine();
                                for(int l=0;l<ivalue.Length;l++)
                                  ivalue[h]=ivalue[h].Replace(iname[l],ivalue[l]);
                                ivalue[h]=re.Eval(ivalue[h]).ToString();

        ivalue[h]=Convert.ToInt32(Convert.ToDouble(ivalue[h])).ToString();
                                break;
                        }
                }
        }
        goto again;
}
if(s.StartsWith("if"))
{
```

```
                int l=0;
                for(i=2;i<s.Length;i++)
                {
                        if(s[i]=='(')
                                l++;
                        if(s[i]==')')l--;
                        if(s[i]==')'& l==0)
                        {
                                i++;
                                break;
                        }
                }
                string aelse="",g=s.Remove(0,i);
                s=s.Remove(i,s.Length-i);
                s=s.Remove(0,2);
                s=s.TrimStart();
                if(bo==true)
                        for(k=0;k<s.Length;k++)
                        {
                                for(int h=0;h<ivalue.Length;h++)
                                        s=s.Replace(iname[h],ivalue[h]);
                        }
                bool lo=Convert.ToBoolean(re.Eval(s).ToString());
                if(g.IndexOf("else")>0)
                {
                        aelse=g.Remove(0,g.IndexOf("else")+4);
                        g=g.Remove(g.IndexOf("else"),g.Length-g.IndexOf("else"));
                }

                if(lo)
                 Second(g);
                else
                 if(aelse!="")
                   Second(aelse);
                if(end==1)goto end;
        goto again;
        }
        if(s.StartsWith("for"))
        {
                s=s.Remove(0,3);
                string a,b=s;
                b=b.Remove(0,b.IndexOf('=')+1);
                string now=s.Remove(s.IndexOf('='),s.Length-s.IndexOf('='));
                now=nospace(now);
                int know;
                for(i=0;i<iname.Length;i++)
                        if(iname[i]==now)
                        {
                                know=i;
                                break;
                        }

                string one=b.Remove(0,b.IndexOf(" do"));
                b=b.Remove(b.IndexOf("do "),b.Length-b.IndexOf("do "));
                for(k=0;k<iname.Length;k++)
            if(b.IndexOf(iname[k]+' ')>=0)
                    b=b.Replace(iname[k]+' ',ivalue[k]+' ');
                b+=one;
                int incdec=1,del=0;
                if(s.IndexOf("step")>0)
                {
                        string step=s.Remove(0,s.IndexOf("step")+4);
                        step=step.TrimStart();
                        step=step.Remove(step.IndexOf(' '),step.Length-step.IndexOf(' '));
```

```
                    incdec=Convert.ToInt32(step);
                    for(i=b.IndexOf("step");i<b.IndexOf("do");i++)
                   del++;
                        b=b.Remove(b.IndexOf("step"),del);
                }
                if(b.IndexOf("do")>0)
                  b=b.Remove(b.IndexOf("do"),2);
                else {Console.WriteLine("Error...(Missing \" do \")");goto end;}
                s=s.Remove(s.IndexOf('=')+1,s.Length-(s.IndexOf('=')+1));
                s+=b;
                a=s;
                a=a.Remove(a.IndexOf("to"),a.Length-a.IndexOf("to"));
                a=nospace(a);
                for(int h=0;h<iname.Length;h++)
                        if(a.StartsWith(iname[h]))
                        {
                                a=a.Remove(0,a.IndexOf('=')+1);
                                ivalue[h]=a;
                                a=s;
                                s=s.Remove(0,s.IndexOf("to")+2);
                                a=s;
                                s=s.TrimStart();
                                s=s.Remove(s.IndexOf(' '),s.Length-s.IndexOf(' '));
                                a=a.TrimStart();
                                a=a.Remove(0,a.IndexOf(' '));
                                int loop;

                        for(loop=Convert.ToInt32(ivalue[h]);loop<=Convert.ToInt32(s);loop+=incdec)
                                {
                            Second(a);
                                ivalue[h]=ivalue[h].Remove(0,ivalue[h].Length);
                                        ivalue[h]=ivalue[h].Insert(0,(loop+incdec).ToString());
                                }
                                incdec=1;
                                ivalue[h]=ivalue[h].Remove(0,ivalue[h].Length);
                                ivalue[h]=ivalue[h].Insert(0,(loop-1).ToString());
                                break;
                        }
                }

        again:;
                j++;

        }
        end:;
        Console.WriteLine("\nPress Enter to exit...");
        Console.Read();
    }
//######################
    static void Second(string s)
    {

        int j=0,k;
        while(j<1)
        {
                int i;
                for(i=0;i<s.Length;i++)
                {
                        if(s[i]=="")
                                for(int jj=i+1;jj<s.Length-1;jj++)
                                        if(s[jj]=="")
                                        {
                                                i=jj+1;
                                                break;
```

```
                        }
        if(s[i]==',')
        {
                s=s.Remove(i,1);
                s=s.Insert(i,"\\");
        }
}
bool line=true;
s=s.TrimStart();
s=s.TrimEnd();
if(s=="end")end=1;
if(s.StartsWith("print"))
{
        s=s.Remove(0,5);
        if(s=="")goto here;
        s=s.TrimStart();
        if(s[s.Length-1]=='\\')
        {
                line=false;
                s=s.Remove(s.Length-1,1);
        }
        if(s.StartsWith("print")){Second(s);goto end;}
        string[] ar=new string [s.Split('\\').Length];//s.Split('+');
        ar=s.Split('\\');
        for(i=0;i<ar.Length;i++)
        {
                if(ar[i][0]==""&& i<=ar.Length)
                {
                        ar[i]=ar[i].Remove(0,1);
                        ar[i]=ar[i].Remove(ar[i].Length-1,1);
                }
                else
                {
                        if(bo)
                                for(k=0;k<ar.Length;k++)
                                {
                                        string d=ar[k];
                                        for(int h=0;h<ivalue.Length;h++)
                                                ar[k]=ar[k].Replace(iname[h],ivalue[h]);
                                }

                        ar[i]=re.Eval(ar[i]).ToString();
                }
                Console.Write(ar[i]);
        }
here:
        if(line)
                Console.WriteLine();
        goto end;
}

if(s.IndexOf("")<0 && s.IndexOf("if")<0 && s.IndexOf('=')>0 && s.IndexOf("for")<0)
{
        s=nospace(s);
        string[] eq=new string[s.Split('\\').Length];
        eq=s.Split('\\');
        for(i=0;i<eq.Length;i++)
        {
                for(int h=0;h<iname.Length;h++)
                        if(eq[i].StartsWith(iname[h]))
                        {
                                eq[i]=eq[i].Remove(0,eq[i].IndexOf('=')+1);
                                eq[i]=re.Eval(eq[i]).ToString();
                                ivalue[h]=(Convert.ToInt32(eq[i])).ToString();
```

```
                                        string hh=ivalue[h];
                                        break;
                                }
                        }
                }
                if(s.StartsWith("enter "))
                {
                        s=s.Remove(0,6);
                        s=nospace(s);
                        string[] eq=new string[s.Split('\\').Length];
                        eq=s.Split('\\');
                        for(i=0;i<eq.Length;i++)
                        {
                                for(int h=0;h<iname.Length;h++)
                                {
                                        string q=iname[h],o=eq[i];
                                        if(eq[i]==iname[h])
                                        {
                                                string d=Console.ReadLine();
                                                d=re.Eval(d).ToString();
                                                ivalue[h]=(Convert.ToInt32(d)).ToString();
                                                break;
                                        }
                                }
                        }
                        goto end;
                }
                if(s.StartsWith("if"))
                {
                        int l=0;
                        for(i=2;i<s.Length;i++)
                        {
                                if(s[i]=='(')
                                        l++;
                                if(s[i]==')')l--;
                                if(s[i]==')'& l==0)
                                {
                                        i++;
                                        break;
                                }
                        }
                        string aelse="",g=s.Remove(0,i);;
                        s=s.Remove(i,s.Length-i);
                        s=s.Remove(0,2);
                        if(bo==true)
                                for(k=0;k<s.Length;k++)
                                {
                                        for(int h=0;h<ivalue.Length;h++)
                                        {
                                                s=s.Replace(iname[h],ivalue[h]);
                                                string w=s,q=iname[h],m=ivalue[h];}
                                }
                        string res=re.Eval(s).ToString();
                        bool lo=Convert.ToBoolean(res);
                        if(g.IndexOf("else")>0)
                        {
                                aelse=g.Remove(0,g.IndexOf("else")+4);
                                g=g.Remove(g.IndexOf("else"),g.Length-g.IndexOf("else"));
                        }
                        if(lo)
                                Second(g);
                        else
                                if(aelse!="")
                                Second(aelse);
```

```
                goto end;
        }
        if(s.StartsWith("for"))
        {
                s=s.Remove(0,3);
                string a,b=s;
                b=b.Remove(0,b.IndexOf('=')+1);
                string now=s.Remove(s.IndexOf('='),s.Length-s.IndexOf('='));
                now=nospace(now);
                int know;
                for(i=0;i<iname.Length;i++)
                        if(iname[i]==now)
                        {
                                know=i;
                                break;
                        }

                for(i=s.IndexOf('=');i<s.IndexOf("do");i++)
                {
                        for(k=0;k<iname.Length;k++)
                                if(iname[k]!=now)
                                        if(b.IndexOf(iname[k]+' ')>=0)
                                                b=b.Replace(iname[k],ivalue[k]);

                }
                int incdec=1,del=0;
                if(s.IndexOf("step")>0)
                {
                        string step=s.Remove(0,s.IndexOf("step")+4);
                        step=step.TrimStart();
                        step=step.Remove(step.IndexOf(' '),step.Length-step.IndexOf(' '));
                        incdec=Convert.ToInt32(step);
                        int o=s.IndexOf("step"),ip=s.IndexOf("do");
                        for(i=b.IndexOf("step");i<b.IndexOf("do");i++)
                                del++;
                        b=b.Remove(b.IndexOf("step"),del);
                }
                if(b.IndexOf("do")>0)
                        b=b.Remove(b.IndexOf("do"),2);
                else {Console.WriteLine("Error...(Missing \" do \")");goto end;}
                s=s.Remove(s.IndexOf('=')+1,s.Length-(s.IndexOf('=')+1));
                s+=b;
                a=s;
                a=a.Remove(a.IndexOf("to"),a.Length-a.IndexOf("to"));
                a=nospace(a);
                for(int h=0;h<iname.Length;h++)
                        if(a.StartsWith(iname[h]))
                        {
                                a=a.Remove(0,a.IndexOf('=')+1);
                                ivalue[h]=a;
                                a=s;
                                s=s.Remove(0,s.IndexOf("to")+2);
                                a=s;
                                s=s.TrimStart();
                                s=s.Remove(s.IndexOf(' '),s.Length-s.IndexOf(' '));
                                a=a.TrimStart();
                                a=a.Remove(0,a.IndexOf(' '));
                                int loop;

for(loop=Convert.ToInt32(ivalue[h]);loop<=Convert.ToInt32(s);loop+=incdec)
                                {
                                        string[] ar =new string[s.Split('\\').Length];
                                        Second(a);
```

```
                                                    ivalue[h]=ivalue[h].Remove(0,ivalue[h].Length);
                                                    ivalue[h]=ivalue[h].Insert(0,(loop+incdec).ToString());
                                                    incdec=1;
                                                }
                                                incdec=1;
                                                ivalue[h]=ivalue[h].Remove(0,ivalue[h].Length);
                                                ivalue[h]=ivalue[h].Insert(0,(loop-1).ToString());
                                                break;
                                            }
                                        }
                                        goto end;
                                    }
                                    end:;
                                }

                            }
                        }
                        ////////////////////////////\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
```

الملحق ب

كيفية تنفيذ البرنامج المبرمج بـ Learn

الفرق بين المفسر ومحرر النص في فتح الملفات أن محرر النص يقوم بفتح الملف ثم عرض محتوياته على الشاشة ، أما المفسر فيقوم بفتح الملفات وتنفيذ التعليمات المكتوبة فيه ، وما يهمنا هو كيف نجعل المفسر يفتح الملف المصدري ثم ينفذه تلقائياً ، يتم ذلك بأمرين من أوامر نظام دوس وهما الأمران ASSOC و FTYPE والسطران التاليان يقومان بجعل كل ملف مصدري للغتنا يتم فتحه تلقائياً بالمفسر عند النقر عليه :

```
ASSOC .lan=open
FTYPE open="C:\Learn.exe""%%1"
```

في السطر الأول نبين الامتداد .lan يتم تشغيله بالمشغل open ويمكن تسمية أي اسم ، في السطر نعطي مسار المشغل أي مسار البرنامج الذي سيقوم بتشغيل وفتح الملفات ذات الامتداد المذكور.

ويتم كتابة السطرين السابقين في نافذة موجه الأوامر ، أو بكتابتهما في ملف دفعي ثم تشغيله ، وهذه الطريقة هي الأفضل لاسيما أنه بالإمكان تنفيذها تلقائياً عند تنصيب البرنامج .

والآن بعد تعريف النظام على مشغل الملفات ذات الامتداد .lan فإنه يتم فتح أي محرر نص ويكتب فيه كود لغة Learn ويتم حفظه بالامتداد .lan ، وعند النقر على الملف لفتحه سيتم تنفيذ البرنامج .

كما يمكن تنفيذ البرنامج من شاشة موجه الأوامر ، فبافتراض أن مسار المفسر هو C:\Learn.exe وكان أحد ملفات Learn المصدرية في نفس المسار فإنه بكتابة الأمر التالي في شاشة موجه الأوامر سيتم تنفيذ البرنامج :

```
C:\Learn first.lan
```

وعلينا التذكر بأن اسم الملف في الأمر السابق هو معامل للدالة الرئيسة main() .

وإذا عُرف مشغل الملفات بالطريقة سالفة الذكر ، فيكفي الذهاب إلى مسار الملف ثم كتابة اسمه فقط كي يتم تنفيذه مباشرة ،فمثلاً الأمر التالي يكفي لذلك :

```
C:\first.lan
```