سلسلة من المقالات و الأبحاث العلمية المكتوبة باللغة العربية تبدأ من اليوم 2018/12/25 و تنتهي بنشر 150 إصدارا بتاريخ 2019/12/25.
نأمل منكم المساهمة في نشر هذه المقالات لدعم المحتوى العربي على شبكة الإنترنت ولدفعنا لتقديم المزيد.

البرولوغ

الجزء الثاني

مجدي عوض

بسم الله الرحمن الرحيم ((قل إن صلاتي و نسكي و محياي و مماتي لله رب العالمين)) صدق الله العظيم

مقدمة

استكمالا للجزء الأول من "مقدمة في لغة البرولوغ" التي نشرتها بتاريخ 2019/01/02 سوف أعمل هنا على تمثيل عدد من المعادلات الفيزيائية ثم سأتحدث قليلا عن الأنظمة الخبيرة و بعد ذلك سوف نعمل سويا على ما اصطلح عليه إعلاميا بلغز آينشتاين (ولا نزعم بأنه عائد لآينشتاين).

معادلات الحركة

كانت الفيزياء و لا تزال مصدر إلهام بالنسبة لي، و أذكر أن "قوانين الحركة" هي أكثر القوانين التي قمت بدراستها سواء خلال المرحلة الثانوية أو الجامعية. لذلك فإنني أفهم هذه القوانين أكثر من فهمي للغة البرولوغ التي أكتب عنها الأن و لذلك سوف أقوم معكم في هذا الجزء من سلسلة تعلم لغة البرولوغ بتصميم برنامج يمكنه التعامل مع معادلات السرعة.

سوف أفترض أن الطالب الذي يقرأ هذا المثال قادر على اشتقاق المعادلات الرياضية خاصة و أنه يسعى لتعلم البرمجة المنطقية لذلك فلن أقوم بشرح عمليات اشتقاق معادلات الحركة و سوف أركز على كيفية كتابتها بلغة البرولوغ. و لنبدأ بتحديد القوانين التي نحن بصدد التعامل معها:

- $v = v_i + at$
- $x = ½(v_i + v)t$
- $x = v_it + ½at^2$
- $v^2 = v_i^2 + 2ax$
- $x = vt - ½at^2$

و فيما يلي تفسيرا سريعا للرموز المستخدمة في المعادلات لمن لا يعرفها:

- V: السرعة الكلية
- X: المسافة بين نقطتين
- A: التسارع الثابت
- T: الزمن اللازم لقطع المسافة بين نقطتين

قبل متابعة القراءة، أرجو منك مراجعة "الجزء الأول – المثال التطبيقي 3" و محاولة كتابة البرنامج ثم مقارنتها بالنص البرمجي التالي:

```
speed(V,V_i,A,T):-V is (V_i+(A * T)).

distance(D,V,V_i,T):-D is 0.5 * (V_i + V) * T.

distance2(D2,V_i,T,A):-D2 is V_i * T + 0.5 * A * (T * T).

speed_squaring(V2,V_i,A,X):-V2 is (V_i * V_i) + 2 * A * X.
```

distance3(D3,V,T,A):-D3 is (V * T) - 0.5 * A * (T * T).

أهم مميزات البرمجة المنطقية أنك تقوم بكتابة برنامج ما بشكل صحيح و لكن و رغم صحته أستطيع كتابة برنامج يؤدي الوظيفة ذاتها بشكل صحيح و باللغة ذاتها ولكن بأسلوب مختلف تماما. قارن ما قمت بكتابته بما قمت أنا بكتابته و إن لم تصدق بأن البرنامج السابق صحيح، قم بتحميله من الرابط الخاص به في المراجع.

الأنظمة الخبيرة

في المثال التطبيقي السابق قمنا بتمثيل قوانين الحركة باستخدام لغة البرولوغ. ولعلنا لم نكن قادرين على تصميم هذا البرنامج لو لم نمتلك خلفية جيدة و واضحة عن قوانين الحركة. ببساطة لو لم نكن خبراء في التعامل مع قوانين الحركة لما كنا قادرين على تمثيلها بلغة البرولوغ بحيث يقوم البرنامج باستخدام خبرتنا في هذه المعادلات و نقلها لأشخاص آخرين. و من هنا يمكننا تعريف الأنظمة الخبير على أنها أنظمة قادرة على محاكاة الخبير البشري في مجال معين و ذلك من خلال استخلاص و تجميع خبراته في هذا المجال.

فلو كنت خبيرا في مجال الطب على سبيل المثال لتمكنت من تصميم برنامج باستخدام لغة البرولوغ قادر على تسمية الأمراض المحتملة وفقا للأعراض التي يدخلها المستخدم. و بما أنني من المتحيزين للغة البرولوغ على حساب نظيراتها كلغة LISP فأنا أعقد بأن اللغة الأمثل لإنجاز و تصميم الأنظمة و التطبيقات الخبيرة هي لغة الـ Prolog و التي نقوم بدراستها حاليا. هذا و سوف أحاول في الجزء التالي تصميم نظام خبير متكامل في مجال الطب (قسم الباطنية) من خلال الاستعانة بصديق (طبيب)خبير في هذا المجال.

لغز آينشتاين

خلال بحثي الأول في سلسلة "أفكار" و الذي حاولت خلاله إيجاد تعريف للذكاء مررت بمسألة منطقية منسوبة لآينشتاين (و لا أدعي أنها له) و يقال بأنك ان استطعت حل هذه المسألة فأنت من أذكى أذكياء العالم، إذ أن 98% من الناس غير قادرين على حلها.

و بما أننا بصدد دراسة أقدم و أجمل لغات البرمجة المنطقية فإن من المفترض بنا حل هذا اللغز بوصفه لغزا قائما على المنطق، إلا أننا سوف نعمل على حل هذا اللغز من خلال تمثيله بلغة البرولوغ و سوف ندع إيجاد الحل له.

في هذا اللغز يقوم آينشتاين (افتراضا) بإعطائنا مجموعة من المعلومات حول عدد من الأشخاص كما يلي:

- The Brit lives in the red house
- The Swede keeps dogs as pets.
- The Dane drinks tea
- The green house is on the left of the white house
- The green house's owner drinks coffee
- The person who smokes Pall Mall rears birds
- The owner of the yellow house smokes Dunhill
- The man living in the center house drinks milk
- The Norwegian lives in the first house
- The man who smokes Blends lives next to the one who keeps cats

- The man who keeps horses lives next to the man who smokes Dunhill
- The owner who smokes Bluemaster drinks beer
- The German smokes Prince
- The Norwegian lives next to the blue house
- The man who smokes Blend has a neighbor who drinks water

ثم يطلب منا تحديد اللون، الجنسية، الحيوان، المشروب، و نوع السجائر لكل واحد من هؤلاء الأشخاص مع توضيح من منهم يمتلك السمكة كحيوان أليف.

لحل هذا اللغز ببساطة يمكننا رسم الجدول التالي و الذي سيجعلنا قادرين على حل اللغز من دون الحاجة لاستخدام الحاسوب أو لغة البرولوغ أو غيرها من التقنيات و هذا ما يدفعني للتشكيك بنسبة المسألة لآينشتاين. فهي تذكرني بالأسئلة التافهة التي تنتشر على موقع اللينكد إن. على أي حال ها هو الحل ببساطة:

	اللون	الجنسية	الشراب	السجائر	الحيوان الأليف
البيت الأول	أصفر	نرويجي	الكحول	دنهل	قطة
البيت الثاني	أزرق	دينيماركي	القهوة	بليندز	حصان
البيت الثالث	أحمر	بريطاني	الحليب	بال مال	طيور
البيت الرابع	أخضر	ألماني	الشاي	برينس	السمكة
البيت الخامس	أبيض	سويدي	الماء	بلو ماستر	كلب

إذا و بدون الحاجة للغة البرولوغ أو غيرها استطعنا أن نحل ما اصطلح على تسميته إعلاميا بـ "لغز آينشتاين" ولكن الأمر قد استغرق منا بعض الوقت لتحديده. و في الواقع أجد أن حل هذا اللغز بدون العودة للغة البرولوغ أسهل بكثير، إذ أن حله بلغة البرولوغ سيتطلب منا تعلم آليات و تقنيات جديدة لذا سوف نبدأ بتعلمها الآن ثم نقوم في نهاية هذه الورقة بكتابة اللغز بلغة البرولوغ.

القوائم بلغة البرولوغ Lists

باختصار و كما يمكننا أن نستنتج من العنوان فالقوائم هي عبارة عن تسلسل محدد للعناصر في لغة البرولوغ كأن نقول:

- [majdi, ahmed, khalid, saeed]
- [majdi, ahmed(abu_jihad), Y, 2, majdi]
- []
- [majdi, [amal, khalid], [saeed, ahmed(abu_jihad)]]
- [[], dead(z), [2, [b, c]], [], Z, [2, [b, c]]]

و كما أن لغة البرولوغ تتصف بقدرتها على الاستنتاج، فمن باب أولى أن يكون دماغك البيولوجي قادر كذلك على الاستنتاج. فمن خلال قائمة الأمثلة السابقة نستنتج ما يلي:

- يمكن كتابة القوائم من خلال إحاطتها بالرمز "[]" و يتم فصل العناصر من خلال "," فواصل عادية كما أن طول القائمة يعتمد على عدد عناصرها. طول القائمة في المثال الأول 4 إذ أنها تحتوي على أربعة عناصر.
- ومن خلال المثال الثاني يمكننا معرفة ما يلي:
  - يمكن أن يكون عنصر القائمة عبارة عن ذرة.
  - يمكن أن يكون العنصر في القائمة عبارة عن مصطلح معقد
  - يمكن أن يكون العنصر في القائمة عبارة عن متغير
  - يمكن أن يكون العنصر في القائمة عبارة عن رقم
  - يمكن أن يتكرر العنصر أكثر من مرة في القائمة
- في المثال الثالث يمكننا ملاحظة وجود قائمة فارغة، و بما أن طول القائمة يعتمد على عدد عناصرها فإن طول القائمة الفارغة هو صفر.
- في المثال الرابع يمكننا استنتاج أن عناصر قائمة ما يمكن أن تتشكل من قوائم أخرى

تتألف القوائم في لغة البرولوغ من رأس و ذيل، أما رأس القائمة فهو العنصر الأول فيها و ذيل القائمة هو كافة العناصر التي تليه و للاستفسار عن رأس و ذيل القائمة في المثال الأول يمكننا أن نكتب:

```
?- [X|Y] = [majdi, ahmed, khalid, saeed].

X = majdi

Y = [ahmed, khalid, saeed]
```

العضو Member في لغة البرولوغ

الـ "List Operations Membership" تستخدم لتحديد ما إذا كانت X على سبيل المثال عنصرا من عناصر القائمة التي نعمل عليها. و لإيضاح الأمر سوف نقوم بكتابة برنامج قادر على معرفة ما إذا كانت X عنصرا في القائمة L و ذلك يتم في حال كانت X هي رأس القائمة أو ذيلها و يكتب النص البرمجي كما يلي:

```
list_member(X,[X|_]).

list_member(X,[_|TAIL]):-list_member(X,TAIL).
```

و الآن سنقوم بسؤال البرولوغ:

```
?- list_member(a, [a,b,c,d]).
```

و سوف يأتينا الجواب على شكل yes أو true أما إذا سألنا:

```
?- list_member(f, [a,b,c,d]).
```

فإن البرولوغ سيجيب بـ no. و الآن دعنا نطرح السؤال بطريقة مختلفة:

```
?- list_member(X, [a,b,c,d]).
```

في هذه الحالة سوف يقوم البرنامج بسرد عناصر القائمة كاملة واحدة تلو الأخرى.

```
list_length([],0).

list_length([_|TAIL],N):-list_length(TAIL,N1),N is N1+1.
```

في السطر الأول من النص البرمجي السابق، نقول للبرولوغ أن طول القائمة الفارغة يساوي صفر أما في السطر البرمجي الثاني فإننا نقول للبرولوغ أن يقوم بإضافة 1 إلى ذيل القائمة لحساب طول القائمة مع التذكير بأن طول القائمة هو مصطلح يستخدم لتحديد عدد عناصر القائمة.

عودة للغز آينشتاين

سوف أسترسل في الحديث عن القوائم في لغة البرولوغ في الورقة التالية أما الآن فإليكم تمثيل لغز آينشتين بلغة البرولوغ مع الإشارة إلى أن بإمكانكم تحميل النص البرمجي من خلال الرابط الخاص به في المراجع.

```
next_to(X, Y, List) :- iright(X, Y, List).

next_to(X, Y, List) :- iright(Y, X, List).

iright(L, R, [L | [R | _]]).

iright(L, R, [_ | Rest]) :- iright(L, R, Rest).

einstein(Houses, Fish_Owner) :- =(Houses, [[house, norwegian, _, _, _, _], _, [house, _, _, _, milk, _], _, _]),

member([house, brit, _, _, _, red], Houses),

member([house, swede, dog, _, _, _], Houses),

member([house, dane, _, _, tea, _], Houses),

iright([house, _, _, _, _, green], [house, _, _, _, _, white], Houses),

member([house, _, _, _, coffee, green], Houses),

member([house, _, bird, pallmall, _, _], Houses),

member([house, _, _, dunhill, _, yellow], Houses),

next_to([house, _, _, dunhill, _, _], [house, _, horse, _, _, _], Houses),

member([house, _, _, _, milk, _], Houses),

next_to([house, _, _, marlboro, _, _], [house, _, cat, _, _, _], Houses),
```

```
next_to([house, _, _, marlboro, _, _], [house, _, _, _, water, _], Houses),

member([house, _, _, winfield, beer, _], Houses),

member([house, german, _, rothmans, _, _], Houses),

next_to([house, norwegian, _, _, _, _], [house, _, _, _, _, blue], Houses),

member([house, Fish_Owner, fish, _, _, _], Houses).
```

بعد كتابة النص البرمجي السابق يمكنك أن تطرح السؤال التالي على البرولوغ:

```
?- einstein(Houses, Fish_Owner).
```

ولن تستغرب الإجابة.

تاريخ النشر: 2019/01/04

تأليف: مجدي عوض

المصادر:

تذكر المراجع بتقنية الـ MLA مع بعض التصرف.

المؤلف	عنوان المصدر	عنوان الموضوع	المساهمون	الإصدار	رقم المجلد	الناشر	تاريخ النشر	تاريخ الاطلاع على المادة	الرابط
Patrick Blackburn, Johan Bos, and Kristina Striegnitz	موقع إلكتروني Learn Prolog Now!	Learn Prolog Now!	Patrick Blackburn, Johan Bos, Kristina Striegnitz	الطبعة الأولى	-	-	2012	2018/12/28	http://www.learnprolognow.org/

لتحميل النصوص البرمجية الواردة في هذا البحث قم بالضغط على هذا الرابط:

www.webtech-internet.com/prolog2.zip