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 الأول

مجدي عوض

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

مقدمة

بصفتي أعمل كمبرمج تطبيقات إنترنت منذ قرابة العشر سنوات فقد اعتدت على نمط معين من البرمجة نتيجة لتعاملي مع لغات كالـ Java و PHP و ASP و قواعد بيانات مثل الـ SQL و MYSQL و الـ Oracle و غيرها. لكن الواقع هو أن أول لغة برمجة كنت قد تعلمتها هي لغة البرولوغ "Prolog" و التي تختلف تماما عن أنماط الكتابة في لغات البرمجة الاعتيادية إذ أنها تعتمد على العلاقات المنطقية و تشبه الكتابة باللغة الإنجليزية المعروفة و لا تضم الكثير من التعقيدات.

و رغم أنني أرى فيها لغة برمجة بسيطة سهلة التعلم، إلا أن هذه اللغة التي ظهرت في العقد السابع من القرن المنصرم لا زالت تحافظ على مكانها في مجالات مختلفة أهمها الذكاء الاصطناعي و تعلم الآلة و الأنظمة الخبيرة.

حقيقة و رغم اهتمامي المبالغ به في هذه اللغة وكثرة بحثي عن مصادر و مراجع لتعلمها، إلا أنني لم أستطع إيجاد مرجع عربي يناقش هذه اللغة بنوع من الجدية رغم وجود بعض المحاولات المتواضعة من قبل عدد من الباحثين و و الأساتذة ولعل ذلك عائد لقلة استخدام البرمجة المنطقية في سوق العمل و قلة اقبال الطلبة على تعلمها. لذلك فقد آثرت أن أجعل من هذه الورقة البحثية شرحا مفصلا مبدئيا للغة البرولوغ "Prolog" مع تنفيذ و شرح بعض التطبيقات.

و في نهاية هذه المقدمة السريعة أشير ألى أنني لم أستعمل لغة البرولوغ خلال العشر سنوات الماضية من حياتي المهنية و إنما كنت أقوم ببعض المشاريع بين الفينة و الأخرى في سبيل تفعيل و تطوير المباديء المنطقية لدي.

مقدمة في لغة البرولوغ "Prolog"

لعل أبرز الاختلافات بين البرولوغ "Prolog" و لغات البرمجة الاعتيادية كالجافا "Java" مثلا هو أن لغة البرولوغ "Prolog" لغة ديناميكية على مستوى نصوصها البرمجية، أي أن النص البرمجي يمكن له أن يتغير لوحده في كل مرة يتعلم فيها البرنامج حقيقة أو أمرا جديدا.

حتى تتمكن من فهم آلية كتابة النصوص البرمجية بلغة البرولوغ "Prolog" عليك أن تزيح كل ما تعرفه عن البرمجة باللغات الاعتيادية و تعلم بأن هذه اللغة تعمل على أساس أمرين فقط و هما: الحقائق "Facts" و قواعد "Rules". أما الحقائق فهي أساس هذه اللغة و التي يمكننا من خلالها بناء قاعدة المعرفة "Knowledge Base" الخاصة بالبرنامج الذي نريد أن نصممه و القواعد باختصار هي الروابط بين العلاقات إن جاز التعبير و هي كقاعدة If باللغات الاعتيادية على سبيل المثال.

قواعد المعرفة KB (الجزء الأول)

هل يمكننا أن ننفي عملية إشراق الشمس من جهة الشرق و غروبها من جهة الغرب ؟؟ هل يمكننا أن أن نقول بأن زحل هو المسؤول عن عملية المد و الجزر ؟؟ بطبيعة الحال لسنا قادرين على فعل ذلك لأنها حقائق معروفة و متفق عليها و ثابتة و من الأمثلة على الحقائق أن اسمي هو "مجدي" و أن "مجدي" يكتب بحثا حول لغة "البرولوغ" و أن "مجدي" يجلس على سريره. قواعد المعرفة في لغة البرولوغ "Prolog" هي عبارة عن مجموعة من الحقائق التي نريد أن نخبر الآلة بها و لتمثيل الحقائق السابقة بلغة البرولوغ نكتبها كما يلي:

name(majdi).

work(cto).

workPlace(bed).

أريدك فقط أن تنتبه إلى أنني قد استخدمت الأحرف الصغيرة فقط Small letters في أول حقيقتين بينما لجأت لاستخدام حروف كبيرة في الحقيقة الثالثة و سف نتحدث عن هذا بعد قليل. أما الآن فدعنا نستفسر و نسأل البرولوغ حول الحقائق السابقة و ليكن السؤال حول الاسم لنكتبه بكل بساطة كما يلي:

?- name(majdi).

و ستأتي الإجابة من البرنامج

Yes

لاحظ فقط أن كافة الجمل بلغة البرلوج سواء كانت حقائق قواعد أو استفسارات فإنها تنتهي بنقطة "Full stop" (.) و في حال لم نضعها في الاستعلام السابق فإن البرنامج لن يجري عملية البحث في قاعدة المعارف ليتأكد من أن الاسم هو "مجدي" و هي حقيقة صريحة بالنسبة له. فلنتعمق قليلا الآن عبر المثال التالي:

male(majdi).

male(ahmed).

female(thuraia).

webDeveloper(majdi).

و الآن سنقوم بطرح بعض الأسئلة على البرولوغ:

?- male(majdi).

ستكون إجابة البرنامج yes كما هو الحال في المثال السابق. سنطرح السؤال التالي:

?- webDeveloper(majdi).

سوف يجيب البرنامج بـ yes مرة أخرى بما أن هذه الحقيقة واحدة من الحقائق في قاعدة المعرفة، ولكن ماذا لو طرحنا السؤال التالي على البرنامج:

?- webDeveloper(ahmed).

بما أن هذه الحقيقة ليست في قاعدة المعارف فإن الإجابة ستكون no هذا و تعتبر قاعدة المعارف التي نعمل عليها إلى الآن بسيطة للغاية و لا زلنا لا نتعامل مع برنامج قادر على الاستنتاج فحتى لو أن "أحمد" مبرمج ويب فإن البرولوج لن يكون قادرا على اكتشاف ذلك. ولكن يمكننا أن نطرح السؤال بطريقة مختلفة على البرولوغ:

?- webDeveloper(who).

من خلال الجملة السابقة فنحن نسأل البرنامج ((من هو مبرمج الويب؟)) ولكن الحقيقة هي أن إجابة البرنامج على هذا السؤال غير متوقعة إذ أنها ستكون no فالبرولوغ اعبرت أننا نسأل إذا كانت who ((كاسم أو كيان)) مبرمج ويب. ولكي نحصل على الإجابة التي نريدها فإن على البرولوغ أن تفهم السؤال الذي نطرحه عليها بشكل صحيح و لذلك فإن علينا أن نكتب who على شكل متغير و لتحقيق ذلك بلغة البرولوغ فإنه يجب علينا أن نكتب الحرف الأول من الكلمة بشكل كبير Capital letter كما يلي:

?- webDeveloper(Who).

و في هذه الحالة ستكون إجابة البرولوغ على سؤالنا majdi و هي الإجابة المتوافقة مع ما تحويه قاعدة المعارف من حقائق.

قواعد المعرفة KB (الجزء الثاني)

خلال بحثي عن مراجع لتعلم لغة البرولوغ وجدت أن أكثر الأمثلة المطروحة ذات صلة بالعلاقات الأسرية شجرة العائلة، ولست بصدد أن أشذ عن ذلك و إن كنت سأعمل على شرح هذه الأمثلة بأسلوب مفصل. سوف نعمل في هذا الجزء من قواعد المعرفة على المثال التالي:

son(majdi).

son(ahmed).

son(khalid).

father(mohammed).

و الآن ماذا لو أردنا أن نقول بأن "محمد" هو والد "مجدي"، في هذه الحالة سوف يكون النص البرمجي كما يلي:

Son(ahmed).

son(khalid).

father(mohammed,majdi).

و هنا أشرنا إلى حقيقة واضحة هي أن "محمد" والد "مجدي" باعتبار هيكلية الحقيقة كما يلي:

fact(x,y).

و الأن لنطرح السؤال التالي على البرولوغ:

?- father(Father,majdi).

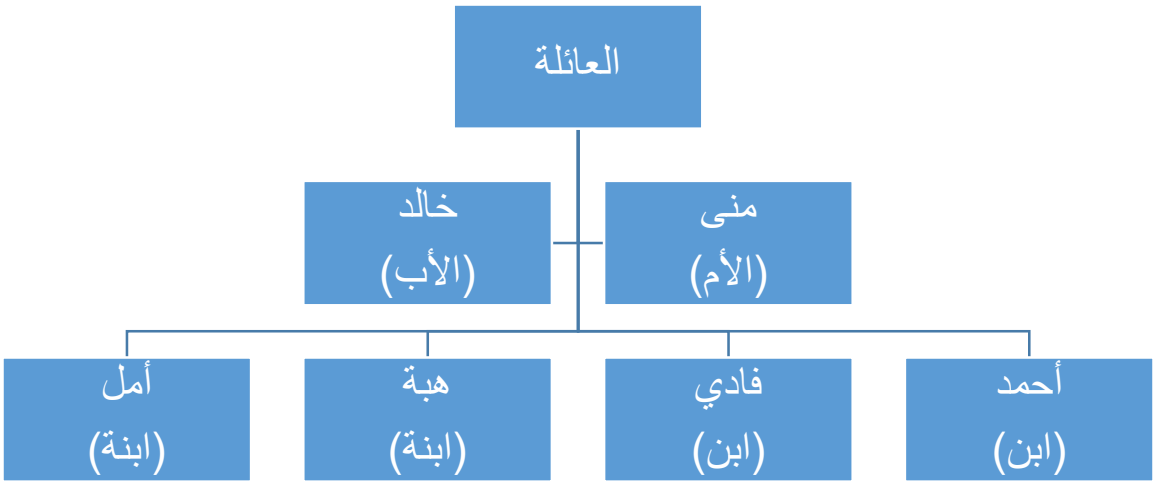
السؤال السابق باللغة العربية هو ((من هو والد "مجدي")) و في هذه الحالة سوف تأتي الإجابة من البرولوغ:

Father = mohammed.

و يمكنك أن تقوم بعكس الآية لتسأل ((من هو ابن "محمد")) فتكتب:

?- father(mohammed,Son)

ليأتيك الجواب المتوقع و هو "majdi". في الواقع فإن ما تملكه من معلومات حول كيفية كتابة الحقائق في لغة البرولوغ كاف لتقوم بكتابة شجرة العائلة الصغيرة الخاصة بك و كمثال على ذلك سوف نتخيل أننا نتعامل مع عائلة مؤلفة من أب و أم و أربع أطفال كما يوضح الشكل التالي



لتمثيل شجرة العائلة السابقة وفقا لما نملكه من معلومات حول كيفية كتابة الحقائق في لغة البرولوغ حتى الآن فأننا بحاجة إلى كتابة النص البرمجي التالي:

father(khalid,ahmed).

father(khalid,fadi).

father(khalid,hiba).

father(khalid,amal).

mother(muna,ahmed).

mother(muna,fadi).

mother(muna,hiba).

mother(muna,amal).

brother(fadi,ahmed).

brother(ahmed,fadi).

brother(ahmed,hiba).

brother(fadi,hiba).

brother(ahmed,amal).

brother(fadi,amal).

sister(hiba,ahmed).

sister(hiba,fadi).

sister(hiba,amal).

sister(amal,hiba).

sister(amal,ahmed).

sister(amal,fadi).

قواعد المعرفة KB (الجزء الثالث)

ليس من المنطقي أن نستخدم ما يعرف بالبرمجة المنطقية لنعبر عن شجرة العائلة السابقة من خلال الطريقة السابقة و إن بدت منطقية و سهلة للكثيرين، إلا أن العلاقات الأسرية في لغة البرولوغ يمكن التعبير عنها من خلال قواعد ثابتة و واضحة و بسيطة بحيث يصبح النص البرمجي السابق كما يلي (ليعطينا نفس النتائج تقريبا مع إضافة بسيطة)

male(khalid).

male(hamed).

male(fadi).

female(muna).

female(hiba).

female(amal).

```
parent(khalid,ahmed).

parent(khalid,fadi).

parent(khalid,hiba).

parent(khalid,amal).

parent(muna,ahmed).

parent(muna,fadi).

parent(muna,hiba).

parent(muna,amal).

mother(X,Y):-parent(X,Y),female(X).

father(X,Y):-parent(X,Y),male(X).

sister(X,Y):-parent(_Z,X),parent(_Z,Y),female(X),X\==Y.

brother(X,Y):-parent(_Z,X),parent(_Z,Y),male(X),X\==Y.
```

عبر هذا النص البرمجي قمنا بداية بتعريف جنس أعضاء شجرة العائلة، ثم قمنا بتحديد الحقائق على أساس الأبوية، و بعد ذلك استخدمنا القواعد لنساعد البرولوغ على الاستنتاج و هذه القواعد ببساطة هي الآلية التي سوف يعرف البرولوغ من خلالها العلاقات بين أفراد هذه العائلة الكريمة.

1- القاعدة الأولى تشرح علاقة الأمومة بحيث تكون X والدة لـ Y، إذا كانت X في الحقيقة الأبوية هي والد أو والدة Y و إذا كانت X أنثى.
2- القاعدة الثانية تشرحة علاقة الأبوية بحيث تكون X والد لـ Y، إذا كانت X في الحقيقة الأبوية هي والد أو والدة لـY و إذا كانت X ذكر.
3- القاعدة الثالثة توضح كيف يجب على البرولوغ معرفة الأخت و فيها تكون X اختا لـ Y شرط أن لا تكون الأخت معرفة في الحقيقة الأبوية كوالد أو والدة لـ Y و أن تكون X أنثى و أن لا تساوي X قيمة Y.
4- تشرح القاعدة الرابعة كيف يجب على البرولوغ معرفة الأح و فيها تكون X أخ لـ Y شرط أن لا يكون الأخ معرفا في الحقيقة الأبوية كوالد أو والدة لـ Y و أن تكون X ذكرا و أن لا تساوي X قيمة Y.

أصبحت الآن قادرا على طرح الأسئلة على البرولوغ، يمكنك كتابة البرنامج السابق و تجربته و لكن بقي علينا في هذا الجزء أن نلخص بعض ما تعلمناه من المثال السابق.

1- يمكننا كتابة الحقيقة بالصيغة .fact(X) كما حدث حينما قمنا بتعريف الحقائق ذات الصلة بجنس أفراد العائلة.

2- يمكننا كتابة الحقيقة بالصيغة .fact(X,Y) كما حدث حينما قمنا بتعريف الحقائق ذات الصلة بالأبوية.

3- الرمز "-:" يستخدم للاستعاضة عن قاعدة If في اللغات العادية و يعني ببساطة "إذا" و قد سبق لنا أن تحدثنا عنه بالتفصيل.

4- الرمز "," يستخدم للإضافة المنطقية كأن نقول ((X أب أو أم لـ Y """و""""" X أنثى)).

5- جميع الجمل البرمجية بلغة البرولوغ تنتهي بالـ Full Stop "."

6- الرمز "==\" يعني "لا يساوي"

7- المتغير "_Z" يعني أن المتغير غير موجود أو غير معروف أو غير معرف أو غير صحيح و في المثال السابق قمنا باستعماله لنخبر البرولوغ أن X لا يرتبط بعلاقة أبوية مع Y لتحديد الأخ و الأخت و سوف نتطرق لهذا النوع من المتغيرات بشيء من التفصل فيما بعد.

قواعد المعرفة KB (الجزء الرابع)

خلال عملي على مثل هذه المراجع أحوال أن أقوم بإنشاء تطبيق قد لا يرتقي في أغلب الأحيان إلى المستوى المطلوب تجاريا أو مهنيا و لكنه غالبا ما يعود بالفائدة على طلبة الجامعات. لذلك سوف أقوم باستكمال البرنامج السابق في هذا الجزء.

سوف نقوم بإضافة أفراء جديدة لعائلتنا الموقرة و لذلك فإننا سوف نقوم بكتابة الحقائق التالية:

1- male(mohammed).
2- male(jihad).
3- Female(jihan).
4- parent(mohammed,khalid).
5- parent(mohammed,jihad).
6- parent(jihan,muna).

من خلال قراءتك للعلاقات السابقة يمكنك معرفة أننا سوف نعمل على إضافة العلاقات التالية:

1- الأجداد
2- العمومة

هذا و سنعمل على إضافة قاعدة لاستنتاج الجدة و الزوجة كنوع من ترسيخ المعلومة. و لنبدأ بكتابة القواعد:

- الأجداد: X هو جد Y إذا كان X والد أو والدة لـ _Z و كان _Z و الدا أو والدة لـ Y. جملة منطقية بسيطة يمككنا أن نعبر عنها بطريقة أبسط بأن نقول أن "محمد" هو جد "أحمد" إذا كان "محمد" والدا لـ "خالد" و كان "خالد" والدا لـ "أحمد". و نعبر عن ذلك بلغة البرولوغ من خلال السطر البرمجي التالي:

grandparent(X,Y):-parent(X,_Z),parent(_Z,Y).

- العمومة: X هو عم لـ Z_ إذا كان X أخ لـ Y (سبق أن وضعنا قانون الأخوة) و كان Y والدا لـ X كأن نقول بأن "جهاد" هو عم "أحمد" إذا كان "جهاد" أخا لـ "محمد" و كان "محمد" والدا لـ "أحمد" و نعبر عن هذا بلغة البرولوغ عبر السطر البرمجي التالي:

uncle(X,_Z):-brother(X,Y),parent(Y,_Z).

- بما أنك إنسان يتشكل دماغه من 100 مليار خلية عصبية فإنك تعلم سلفا أن "جهان" هي جدة "أحمد" و "أمل" و "فادي" و "هبة" ولكن و للأسف فإن البرولوغ لا تمتلك قدرة دماغك البيولوجية المعقدة لذلك فإن عليك أن تضع للبرولوغ قاعدة لمعرفة الجدة و يكون ذلك عبر أن تخبر البرولوغ بأن X هي جدة Z_ إذا كانت X أما لـ Y (سبق و قمنا بتعريف قانون الأمومة) و كانت Y أما لـ Z_ و يكون ذلك عبر كتابة السطر البرمجي التالي:

grandmother(X,_Z):-mother(X,Y),parent(Y,_Z).

- إذا ما أردنا تعريف الزوجة بحيث تكون X زوجة لـ Y فإن X يجب أن ترتبط بعلاقة أبوة مع Z_ و كذلك يجب على Y أن ترتبط بعلاقة أبوة مع Z_ و يجب أن تكون X أنثى و بصيغة أخرى "منى" هي زوجة "خالد" إذا كانت "منى" أما لـ "أحمد" و كان "خالد" أبا لـ "أحمد" و كانت "منى" أنثى هذا و نعبر عما سبق برمجيا من خلال السطر التالي:

wife(X,Y):-parent(X,_Z),parent(Y,_Z),female(X),male(Y).

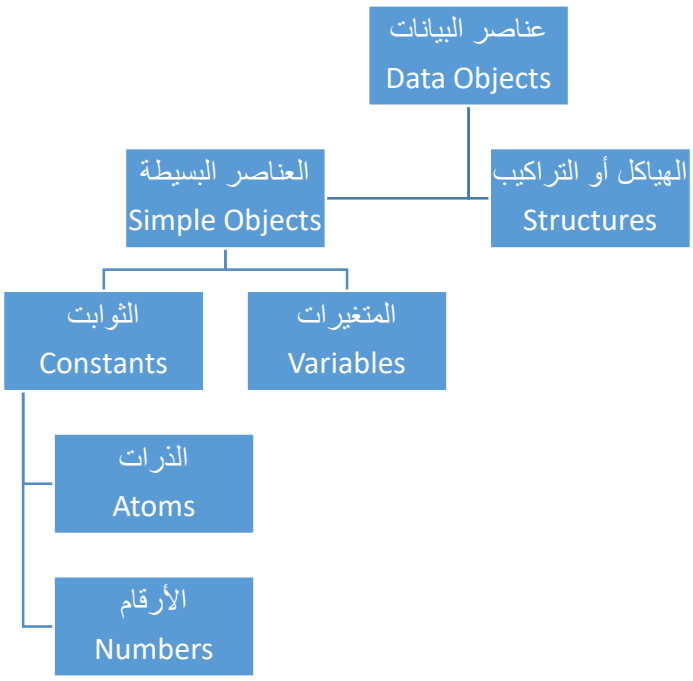
و بهذا نكون قد استطعنا إنجاز أول برنامج بلغة البرولوغ من خلال هذا المرجع بحيث يستطيع هذا البرنامج معرفة العلاقات العائلية من خلال قواعد برمجية كتبت بلغة البرولوغ. (يمكنك تحميل النص البرمجي من خلال الرابط في المراجع)

- العلاقات التي يستطيع البرنامج استنتاجها هي:
  - الأم
  - الأب
  - الأخت
  - الأخ
  - الأجداد
  - الجدة
  - الزوجة
  - العم

يمكنك الآن أن تستمتع بإنجازك في حال كنت قد قمت بكتابة البرنامج مع الشرح عبر طرح الأسئلة على البرولوغ و إعادة سؤاله فرحا بإنجازك ولكني سأنتقل للعنوان التالي.

عناصر البيانات Data Objects

أقصد بـ "عناصر البيانات" البنية التي تشكل النصوص البرمجية بلغة البرولوغ و لعل الشكل التالي يوضح هذه البنية بشكل بسيط:



و لكي تتضح الصورة بشكل أكبر فهذه مجموعة من الأمثلة على عناصر البيانات:

Atoms: majdi – khalid – ahmed – course101 – course_101

Numbers: 101 – 102 – 101.1

Variables: X, Y, Z, _Z, Majdi

Structures: point(25,10) – date(28, nov, 1988)

كيف تكتب الذرات بلغة البرولوغ:

- تتشكل الذرات من حروف شريطة أن تبدأ بحرف صغير
- يمكن للذرات أن تحوي أرقاما كما في الأمثلة السابقة
- يمكن أن تبدأ الذرات بحرف كبير شريطة أن توضع بين علامتي تنصيص منفردة مثل: 'Majdi'
- يمكننا استخدام الرموز في الذرات كأن نكتب :::::::: أو ---------- أو -:-_-:: ولكن علينا الانتباه إلى أن بعض هذه الرموز محجوزة في لغة البرولوغ مثل "," أو "-:" و هي ذات معنى.

كيف تكتب الأرقام بلغة البرولوغ:

- الأرقام الصحيحة و تكتب كما يلي:
  - 100
  - 10
  - -20
  - 20

- المعدل الطبيعي للأرقام الصحيحة هو 16383- و حتى 16383
- الأرقام الحقيقية و تكتب كما يلي:
  - 16.2050
  - -0.10001
  - 100.205
- أشير فقط إلى أن استعمال الأرقام بلغة البرولوغ غير منتشر ولكن الإشارة لها كانت ضرورية من باب العلم بالشيء.

كيفية كتابة المتغيرات في لغة البرولوغ:

- تكتب على شكل رموز مثل X أو Y أو Z أو V شريطة أن يكون الحرف كبيرا Capital Letter
- تكتب على شكل كلمات مفردة شريطة أن تبدأ الكلمة بحرف مفرد مثل Mother أو Father
- تكتب على كلمتين أو أكثر بحيث يفصل بينهما الرمز "_" underscore مثل Old_address
- كما يمكن أن يحتوي المتغير رقما كأن نقول Course101
- يمكن أن يبدأ المتغير برمز الـ underscore فنكتب course101_
- و قد يكون رقما فقط شريطة أن يسبق بعلامة الـ underscore مثل 101_

كيفية كتابة الهياكل أو التراكيب بلغة البرولوغ:

- تكتب وفقا للصيغة التالية:
  - functor(arguments)
  - كأن نحدد تاريخا معينا فنقول:
    - date(28,11,1988)
    - date(28,November,1988)

مثال تطبيقي 1

غالبا ما يطلب من طلاب الجامعات كتابة تطبيقات بسيطة في الامتحانات و رغم أن الأمر يبدو غير منطقي إذ يقضي المحاضر أربعة أشهر تقريبا و هو يعطي الطالب كل ما يعرفه عن المادة التي يدرسها له ثم يتوقف نجاحه فيها من عدمه على برنامج بسيط. و لكن نظرتي هذه و التي أعتقد أنها منطقية بالنسبة للغات البرمجة الاعتيادية خاصة لغة C و C++ التي تدرس في الجامعات ليست منطقية على الإطلاق إذا ما تحدثنا عن البرولوج خاصة و أنني أفترض أن الطالب يلجأ للتعلم الذاتي من خلال هذا المرجع. لذا فسوف أسألك أن تكتب لنا برنامجا بلغة البرولوغ دون أن تكمل قراءة هذا العنوان و البرنامج رغم بساطته إلا أنه يعتمد على مدى قدرتك على التفكير بشكل منطقي و بسيط.

البرنامج المطلوب تصميمه، هو عبارة عن برنامج قادر على إيجاد أعلى رقم من رقمين كأن نقول 100 و 200 فإن على البرنامج أن يعرف أن 200 هي الرقم الأعلى و أن يتمكن كذلك من إيجاد الرقم الأصغر ضمن رقمين كأن نقول 5000 و 1000 فإن على البرنامج أن يعلم أن 1000 هو الرقم الأصغر في هذه المجموعة. (ملاحظة: أنت حر تماما بآلية كتابة البرنامج بلغة البرولوغ) – حاول أن تجرب وحدك ثم انتقل معي إلى الصفحة التالية.

لكتابة هذا البرنامج سوف نعمل على تحديد أربعة قواعد وفقا للصيغة التالية:

max(Variable1,Variable2,TheAnswer).

بحيث يقوم المستخدم (نحن) بتوزيد البرولوغ بقيمة المتغير الأول، و قيمة المتغير الثاني و بهذا فإن النص البرمجي لهذا البرنامج البسيط سيكون:

max(X,Y,X):-X>=Y.

max(X,Y,Y):-X<Y.

min(X,Y,X):-X=<Y.

min(X,Y,Y):-X>Y.

يمكنك تحميل النص البرمجي من خلال الرابط الخاص به في المراجع.

مثال تطبيقي 2

إن كنت تمتلك القليل من الرغبة في استغلال ما تتعلمه لكسب المال، فإن هذا المثال سيشكل بالنسبة لك إلهاما لا مثيل له و سيفتح شهيتك لتعلم البرمجة عموما و البرمجة المنطقية على وجه الخصوص. فمع القليل من الخيال يمكنك أن تجد نفسك .... اختر انت.

على عكس المثال السابق، فرغم بساطة هذا المثال إلا أنه سيخرج لنا تطبيقا عمليا يمكنك بيعه لزملائك الكسالى في مادة الدوائر الكهربائية. و سيعتمد مثالنا هذا على المعادلتين التاليتين:

$R_{total} = R_1 + R_2$

$R_{total} = (R_1 * R_2) / (R_1 + R_2)$

باختصار سوف نعمل على كتابة برنامج يقوم بحساب المقاومة المكافئة لدارة كهربائية على التوالي و التوازي. حاول كتابة البرنامج لوحدك وفقا للمعادلات الرياضية السابقة ثم أكمل متابعة الشرح لترى كيف قمت بكتابته. هذا و يمكنك تحميل النص البرمجي من خلال الرابط الخاص به في المراجع.

البرنامج بلغة البرولوغ أبسط بكثير مما يمكنك أن تتخيل، عمليا كل ما سنقوم به هو تمثيل المعادلات السابقة بلغة البرولوغ و كتابتها كقواعد و لنبدأ بالمعدلة الأولى على التولي:

series(R1,R2,R_total):-R_total is (R1+R2).

و الآن على التوازي:

parallel(R1,R2,R_total):-R_total is ((R1 * R2) / (R1 + R2)).

نعم لقد انتهى البرنامج و هو جاهز لحساب المقاومة المكافئة رغم أن الرموز المستخدمة هنا غير صحيحة من الناحية العلمية المتبعة في التعامل مع الدوائر الكهربائية.

قم فقط بتعديل الرموز، جرب حساب المقاومة المكافئة وفقا لأمثلة من الإنترنت، إن كانت إجابات البرنامج صحيحة، قم ببيعه لزملائك الكسالى في كليات الهندسة.

و بالمثال التطبيقي السابق أكون قد انتهيت من مقدمتي حول لغة البرولوغ و التي أسعى قريبا لإصدار كتاب خاص بها أعمل عليه حاليا. آمل أن يكون فيما كتبت فائدة للطالب العربي.

تاريخ النشر: 2019/01/02

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/prolog.zip