المحاضرة #٥: لغات البرمجة

١ مقدمة

أن لغات البرمجة التي نستخدمها يجب أن تُظهر العمليات التي يستطيع الحاسوب إنجازها. بالعودة إلى تعريف الحاسوب: جهاز إلكتروني قابل للبرمجة بإمكانه خزن واسترجاع ومعالجة البيانات. فان الكلمات التي تمثل العمليات هي البرمجة والخزن والاسترجاع والمعالجة. أن **البرمجة** ببساطة هي التغيرات التي تجريها الإيعازات على البيانات. ولكي يغير الحاسوب البيانات يجب علينا تغيير الإيعازات. أن **الخزن والاسترجاع والمعالجة** هي العمليات التي تستطيع الحواسيب إنجازها على البيانات. بتعبير أخر فان **الإيعازات** التي تنفذها وحدة السيطرة (داخل وحدة المعالجة المركزية CPU) يمكن **خزنها** في ذاكرة الحاسوب ثم **استرجاعها** مرة أخرى من الذاكرة ثم **معالجة** هذه البيانات بطريقة ما في وحدة الحساب والمنطق (داخل وحدة المعالجة المركزية CPU). أن كلمة المعالجة هي كلمة عامة تتضمن تنفيذ العمليات الحسابية والمنطقية على قيم البيانات. أضافه إلى ذلك يجب أن توضع البيانات بدءًا في الذاكرة و يجب إن تكون هناك إيعازات لتفاعل أجهزة الإدخال مع المعالج ولتفاعل المعالج مع أجهزة الأخراج.

٢ لغات المستوى الأدنى (Low-Level Languages)

٢.١ لغة الآله (Machine code)

ذكرنا في المحاضرة الأولى أن الإيعازات البرمجية الوحيدة التي تستطيع الحواسيب فعليا تنفيذها هي الإيعازات المكتوبة بلغة الآله. وهي اللغة الوحيدة المتاحة للجيل الأول للحواسيب. لكل معالج (CPU) لغة آله خاصة به وهي الإيعازات الوحيدة التي يستطيع تنفيذها. هناك عدد محدود من الإيعازات في لغة الآله لذا فان مصممي المعالجات ببساطة يضعون رقم ثنائي (Binary Number) لتمثيل كل إيعاز. وهي مشابه لطريقة تمثيل الرموز والأحرف الموصوفة في المحاضرة الثانية (تمثيل البيانات).

٢-٢ لغة التجميع (Assembly Language)

اشرنا في محاضرة المقدمة الأولى إلى أن أول وسيلة مساعدة تم تطويرها لمبرمجي الحواسيب الأولى هي لغة التجميع. تمثل إيعازات لغة التجميع تشفير نصي (أي نصوص مكونة من احرف بدلا من الأرقام) سهل التذكر لكل من إيعازات لغة الآله. وبذلك فان المبرمج يستخدم هذه المختصرات النصية السهلة بدلا من الأرقام الثنائية أو السداسي عشر مما يسهل كتابة البرامج و يقلل احتمالية الخطأ فيها. على سبيل المثال من السهل تذكر الإيعاز على هيئة الكلمة (ADD) بمعنى اجمع بدلا من الرَّقَم الثنائي (0110).

في النهاية فان البرنامَج يجب أن يكون بلغة الآله لكي يستطيع الحاسوب تنفيذه. لذلك لابد من وجود برنامَج يترجم إيعازات لغة التجميع إلى لغة الآله. يدعى هذا البرنامَج بالمجمع (**Assembler**). يقوم برنامَج المجمع بترجمة كل إيعاز من البرنامَج المكتوب بلغة التجمع إلى ما يقابله من الأعداد الثنائية (لغة الآله). ولان كل معالج له لغة الآلة الخاصة به فان له لغة تجميع خاصة به أيضاً.

شكل ١: برنامَج المجمع (Assembler)

مصدر الشكل كتاب Computer Science Illuminated by Nell B. Dale

٣ لغات المستوى الأعلى (High-Level Languages)

٣-١ عملية الترجمة (Translation Process)

كما هو مذكور أعلاه فان البرنامَج المكتوب بلغة التجميع (Assembly) سيتم ترجمته إلى لغة الآله بواسطة برنامَج المجمع (Assembler) لكي يتم تنفيذ البرنامَج النهائي المكتوب بلغة الآله. في لغات المستوى الأعلى (High-Level Languages) تتطلب عملية تَرْجَمَة البرامج أدوات برمجية أضافية مثل المُصرِف (Compiler) والمفسر (Interpreter).

٣-١-١ المُصرِفات (Compilers)

توفر لغات المستوى الأعلى (High-Level Languages) مجموعة غنية من الإيعازات والتي تسهل عملية البرمجة وتوفر أمكانيات برمجية كبيرة. لذا فان عملية تَرْجَمَة هذه الإيعازات أصعب من لغة التجميع. تسمى البرمجيات التي تترجم لغات المستوى الأعلى بالمُصرِفات (Compilers). يستطيع البرنامَج المكتوب بلغات المستوى الأعلى أن يعمل على إي حاسوب لدية المُصرِف (Compiler) المناسب لهذه اللغة. في الأجيال الأولى للمُصرِفات كان الناتج عنها هو برنامَج بلغة التجميع. إلا أن المُصرِفات الحالية تطورت وأصبحت اكثر تعقيداً وتترجم البرنامَج إلى لغة الآله مباشرة.

٣-١-٢ المفسرات (Interpreters)

المفسر (Interpreter) هو برمجية تترجم وتنفيذ جمل البرنامَج بالتسلسل. بشكل مغاير للمجمع (Assembler) والمُصرِف (Compiler) الذي ينتج عنهما برنامَج بلغة الآله الذي يتم تنفيذه في خطوات أضافية. يقوم المفسر بترجمة الجملة البرمجية وتنفيذها مباشرة. تنقسم اللغات في الجيل الثاني للغات المستوى الأعلى إلى نوعين: لغات يتم تصريفها (Compiled) ولغات يتم تفسيرها (Interpreted). من أمثلة اللغات التي يتم تصريفها (Compiled) فورتران (FORTRAN) وكوبل (COBAL) والكول (ALGOL) بينما من أمثلة اللغات التي يتم تفسيرها (Interpreted) ليسب (Lisp) وسنابول٤ (SNOBOL4) واي بي ال (APL) وبايثون (Python). بسبب تعقيد برمجيات المفسرات (Interpreters) فان البرامج التي يتم تفسيرها (Interpreted) تعمل بشكل أبطأ من البرامج التي يتم تصريفها (Compiled). كنتيجة لذلك اصبح التوجه للغات المُصرفة (Compiled) اكثر حتى ظهور لغة جافا (Java).

طورت لغة جافا (Java) عام ١٩٩٦ وتم تبنيها بشكل كبير من قبل مجتمع المبرمجين. صُممت لغة جافا (Java) بشكل يعطي أولولية إلى قابلية النقل (Portability) وهي القدرة على العمل على مختلف أجهزة الحاسوب. لتحقيق ذلك يتم تصريفها (Compiled) إلى لغة آله قياسية (standard machine language) خاصة بـ جافا تسمى بشفره بايت (Bytecode). ثم يقوم برنامج مفسر (

(Interpreter) يسمى بـ آله جافا الافتراضية (Java Virtual Machine - JVM) بتنفيذ شفرة بايت (Bytecode). أن شفرة بايت (Bytecode) لا تمثل لغة آله لأي معالج حاسوب معين. لذا فان اي حاسوب لدية آله جافا الافتراضية (JVM) يستطيع تنفيذ برنامَج جافا.

Programming Language	2024	2019	2014	2009	2004	1999	1994	1989
Python	1	3	7	7	7	24	23	-
C++	2	4	4	3	3	2	2	2
C	3	2	1	2	1	1	1	1
Java	4	1	2	1	2	3	-	-
C#	5	6	5	6	9	13	-	-
JavaScript	6	7	8	9	10	10	-	-
Go	7	18	35	-	-	-	-	-
Visual Basic .NET	8	20	234	-	-	-	-	-
SQL	9	9	-	-	100	-	-	-
Fortran	10	29	30	25	14	17	5	9
Ada	24	35	32	26	16	12	7	4
Lisp	30	31	17	17	13	19	6	3
Objective-C	35	10	3	27	38	-	-	-
Classic Visual Basic	-	-	41	5	4	4	3	5

شكل ٢: اكثر لغات البرمجة استخداماً عبر اربعة عقود

المصدر www.tiobe.com/tiobe-index/

٢.٣ أنماط لغة البرمجة (Programming Language Paradigms)

تصنف لغات البرمجة ذات المستوى الأعلى إلى نوعين رئيسيين: البرمجة بأسلوب الأوامر (imperative) والبرمجة بأسلوب التصريح (declarative).

١.٢.٣ نمط الأوامر (Imperative Paradigm)

إن غالبية لغات البرمجة التي تم تطويرها واستخدامها خلال تاريخ تطور البرمجيات تنتمي لهذا النوع. مثل لغة فورتران (FORTRAN) وبيسك (BASIC) وسي (C) وباسكال (Pascal) وسي++ (C++). يوصف البرنامَج المكتوب بلغات هذا النوع العمليات الضرورية لحل المسائل.أن ما يميز اللغات بنمط الأوامر هو التنفيذ المتتالي للإيعازات واستخدام المتغيرات (variables) لتمثيل مواقع الذاكرة واستخدام جمل التعيين لتغيير قيم تلك المتغيرات.

٣-٢-١-١ النمط الإجرائي (Procedural Paradigm)

أن جمل البرنامَج المكتوب بالنمط الأجرائي تكون مجمعة في برامج فرعية. يتكون البرنامَج من تسلسل هرمي من البرامج الفرعية. تقوم كل واحدة منها بإنجاز المهمة المحددة الضرورية لحل البرنامَج الكامل. من أمثلة اللغات الاجرائيه لغة سي++ (C++).

٣-٢-١-٢ النمط الشيئي (Object-Oriented Paradigm)

يتكون البرنامَج في هذا النوع من البرمجة من الأشياء (Objects) التي تتفاعل فيمَا بينها. تمُثل الأشياء البيانات والبرنامج الذي تعالجها مجتمعتًا سويتًا. كل "شيء" (Object) مسؤول عن معالجته بشكل منفصل. في البرمجة الاجرائية تعدّ "الأشياء" بيانات غير فعاله وتتم معالجتها من قبل البرنامَج. بينما في البرمجة الشيئية فان "الأشياء" هي بيانات فعالة. أن الأشياء (Objects) والبرامج التي تعالجها مجمعة معًا مما يجعل كل "شيء" (Object) مسؤول عن متغيراته. من الأمثلة الحديثة للغات البرمجة الشيئية هي جافا (Java) وبايثون (Python). بالرغم من اعتبار لغة جافا (Java) لغة شيئية إلا أنها تمتلك بعض الخصائص البرمجة الإجرائية. بالمقابل فان لغة سي++ (C++) هي لغة أجرائية ولكنها تمتلك بعض الخصائص البرمجة ألشيئية.

٣-٢-٢ نمط التصريح (Declarative Paradigm)

وهو الأسلوب الذي تكون فيه النتائج موصوفة لكن خطوات الحصول عليها غير مذكورة. هناك نموذجان أساسيان في هذا النوع هما الوظيفي والمنطقي.

٣-٢-٢-١ النموذج الوظيفي (Functional Model)

يعتمد هذا الأسلوب على مبدأ الدالة (function) في الرياضيات. إي أن العمليات الحسابية يتم التعبير عنها بدلالة حساب ألدالة وأن حلول المسائل يتم التعبير عنها بدلالة استدعاء ألدوال. لا يوجد متغيرات ولا جمل تعيين قيم لهذه المتغيرات. فمثلا عملية جمع قيمتين يمكن التعبير عنها بهذا الشكل (30 40 +). من الأمثلة الشائعة للغات برمجة للنموذج الوظيفي هي ليسب (Lisp) وسكيم (Scheme) وهي لغة مشتقة من ليسب ولغة ام ال (ML). مثل حساب مضروب العدد (factorial) باستخدام لغة سكيم (Scheme):

```
#;> (define factorial
#;> (lambda(n)
#;>  (if
#;>  (= n 0)
#;> 1
#;> (* n (factorial (- n 1))))))
#;> (factorial 7)
5040
```

٢-٢-٣ البرمجة المنطقية (Logic Programming)

يعتمد هذا الأسلوب على مبادئ المنطق الرمزي التي تشتمل على مجموعة من الحقائق عن الكائنات ومجموعة من القواعد حول العلاقات التي تربط تلك الكائنات. يتكون البرنامَج من مجموعة من الأسئلة حول الكائنات والعلاقات التي تربطها و التي يمكن استنتاج إجاباتها من الحقائق والقواعد. من أمثلة هذا النوع هي لغة برولوك (Prolog). يتكون البرنامَج المكتوب بلغة برولوك (Prolog) من ثلاث أنواع من الجمل: النوع الأول للتصريح الحقائق عن الكائنات والعلاقات التي تربطهم. النوع الثاني لتعريف القواعد عن الكائنات والعلاقات بينهم. أما النوع الثالث فللسؤال عن الكائنات والعلاقات التي تربطهم.

```
owns(mary,bo).
owns(ann,kitty).
owns(bob,riley).
owns(susy,charlie).
```

التصريح عن حقائق الكائنات والعلاقات التي تربطها

```
?-owns(mary,bo) Yes
?-owns(bo,mary) No
?-owns(susy,bo) No
```

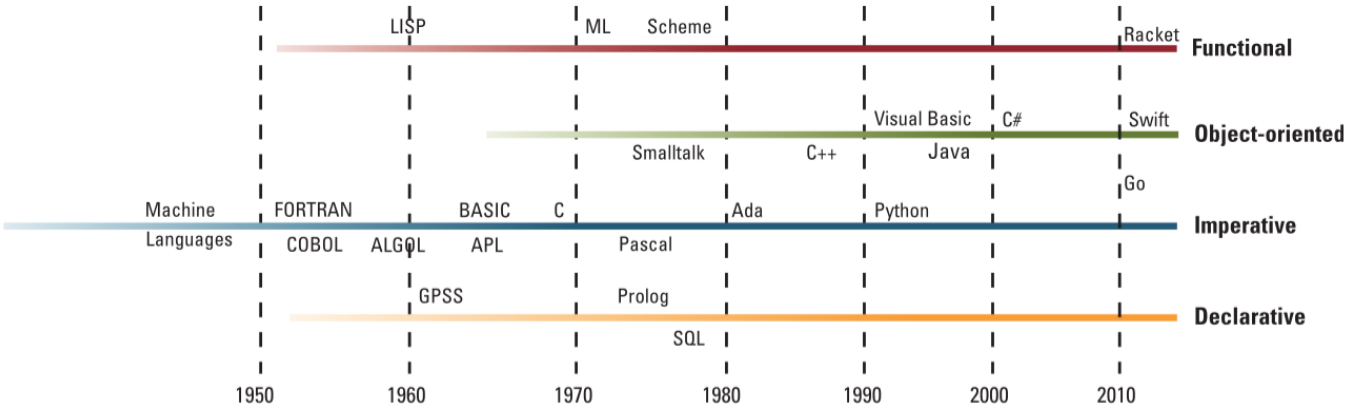
```
?-owns(ann,Cat).
?-owns(Name,charlie).
```

أسئلة عن الكائنات والعلاقات التي تربطها

```
Cat = kitty
Name = susy
```

شكل ٣: مثال يوضح اجزاء لغة برولوك (Prolog)

مصدر الشكل كتاب Computer Science Illuminated by Nell B. Dale



شكل ٤: تطور الأنماط البرمجية

مصدر الشكل كتاب Computer Science: An Overview by J. Glenn Brookshear and Dennis Brylow