الجلسة الأولى: تصميم مترجمات

الهدف من الجلسة

- التعريف بمفهوم لغات البرمجة وأشهر أنواعها.
- التعريف بمفهوم المترجمات والمفسرات.
- التعريف بمراحل التحليل اللفظي والقواعدي والمعنوي وتوليد الشيفرة.

لغة البرمجة

مجموعة تعليمات توجه عمل الحاسب لأداء مهمة معينة. هناك العديد من لغات البرمجة كل منها يناسب تطبيق مختلف.

يوضح الجدول التالي مقارنة بين أشهر لغات البرمجة الحالية:

لغة البرمجة	سرعة التنفيذ	الذاكرة	التطبيق العملي	طريقة الترجمة	دعم غرضية التوجه
Java	1.89 sec	6.01 mb	تطبيقات الهاتف : android applications تطبيقات الويب: spring framework تطبيقات الشركات الكبرى.	Compiled /interpreted	تدعم غرضية التوجه
JavaScript	6.52 secs	4.59 mb	تطبيقات الويب: NodeJS	Compiled /interpreted	تدعم غرضية التوجه
PHP	27.64 secs	2.57 mb	تطبيقات الويب:framework Laravel		تدعم غرضية التوجه
Python	71.90secs	2.80mb	واسعة الانتشار. تطبيقات الويب: Django,flask,fastAPI	interpreted	تدعم غرضية التوجه
C	1.00 secs	1.17 mb	نظم التشغيل. النظم الموزعة. بناء المترجمات.	Compiled	لغة إجرائية
C#	3.14 secs	2.85mb	تطبيقات الويب: ASP.net محركات الألعاب: unity	Compiled	تدعم غرضية التوجه
Ruby	59.34secs	3.97mb	تطبيقات الويب بشكل خاص: Ruby on Rails	Compiled /interpreted	تدعم غرضية التوجه
Perl	65.79 secs	6.62 mb	testing databases تطبيقات الويب: Mojolicious	Compiled /interpreted	تدعم غرضية التوجه
SQL	-	-	Database applications	-	-

المترجمات compiler

برنامج يقوم بتحويل لغة مصدرية إلى لغة هدف.
أشهر أمثلتها مترجم يقوم بتحويل لغة عالية المستوى مثل C أو java إلى لغة assembly خاصة بالحاسب.

يوجد نوع أخرمن البرامج المستخدمة في ترجمة لغات البرمجة وهي المفسرات interpreters تختلف عن المترجمات في ألية العمل فالمترجمات تقول بترجمة الكود المصدري كاملة ثم تنفيذه في حين أن

المفسرات تقوم بترجمة الكود سطراً سطراً ثم تنفيذ كل سطرمباشرة بعد ترجمته.
أشهر أنواع المفسرات، مفسر لغة الـPython.

مراحل الترجمة

تمر عملية الترجمة بالمراحل التالية:

مرحلة تحليل المفردات:

يقوم بها المحلل اللفظيlexical analyzer حيث يقوم بتحويل سلسلة المفردات إلى سلسلة من الرموز tokens.

مثال:

السطر البرمجي التالي:

'print 3 + 1'

سينتج عنه مجموعة الرموز:

['print','1','+','3']

في الواقع، كل رمز سيحتوي نوع وقيمة موافقة (يتحدد في جدول الرمز).

يتم تحديد نوع الرمز وقيمته بناءً على قوالب محددة مسبقاً تم تعريف المترجم عليها تدعى بالتعابير النظامية **regular expressions**.

إذا تمثل هذه المرحلة مسحاً Scanning لتسلسل الدخل لتحديد المفردات Tokens الداخلة في تركيب العبارة، ولا يهتم المحلل اللفظي بترتيب الـ Tokens وبالتالي السطر التالي صحيح تماما بالنسبة له:

For (i=1;i<3;i++)

تعد هذه العبارة صحيحة لفظياً lexically correct لكنها خاطئة قواعدياً، وهنا تأتي مرحلة التحليل القواعدي لتحديد هذه الأخطاء.

مثال عن التعابير النظامية:

```
import re

#discove integer digits in a script
text1="a12t6k3"
#discove ID in a script
text2="1d"

pattern_digit = r"\d+" #digits
pattern_ID=r"[a-zA-Z_][a-zA-Z_0-9]*"

print(r"digits are:",re.findall(pattern_digit, text1))
print(r"ID:",re.findall(pattern_ID, text2))
```

النتيجة:

```
digits are: ['12', '6', '3']
ID: ['d']
PS D:\compilers> py regex.py
digits are: ['12', '6', '3']
ID: ['d']
```

مرحلة التحليل القواعدي Syntax Analysis

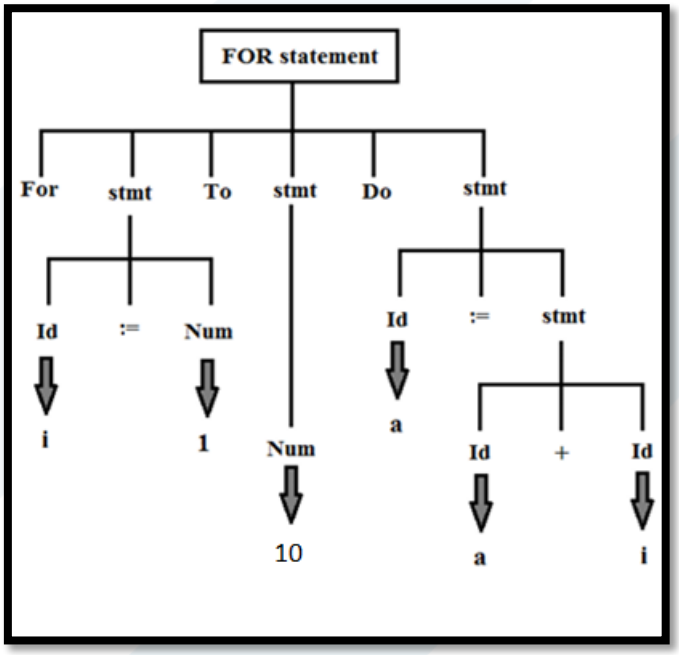
يتم خلال هذه المرحلة التحقق من النحو الخاص باللغة ويكون هذا وفقا لمجموعة من القواعد والتي تمثل الضابط الذي يحكم صحة تسلسل المفردات Tokens التي تم مسحها من قبل المحلل اللفظي.

يقوم المحلل القواعدي ببناء الشجرة باستخدام مجموعة Tokens التي تم توفيرها بالجدول السابق، مع الأخذ بعين الاعتبار قواعد اللغة التي ستحدد فيما إذا كانت العبارة الممسوحة صحيحة قواعدياً أو لا.

قد تكون العبارة صحيحة لفظياً وقواعدياً، لكنها ليست ذات معنى.

شجرةالإعراب للكود التالي:

for i:=1 to 10 do a:=a+i



مرحلة تحليل المعاني Semantic Analysis:

تعتمد هذه المرحلة بشكل أساس على جدول الرموز المنشأ سابقاً والذي يتضمن أسماء الرموز التي صادفها الماسح خلال مسحه لسلسلة الدخل.

هناك العديد من الشروط التي يتم التأكد منها في هذه المرحلة مثل:

- التأكد من تطابق أنواع المتغيرات مثل إسناد رقم حقيقي real لمتغير من النوع الصحيح integer.
- التأكد من تطابق نوع المتغير مع قيمته.
- التأكد من عدم تكرار اسم متغير معرفاً مسبقاً
- التأكد من عدم إسناد قيمة لمتغير غير مصرح عنه سابقاً

مرحلة توليد الكود Code generation

وهنا نقوم بعملية إنتاج كود بلغة الآلة أو لغة التجميع.

الأدوات الأساسية لبناء المترجمات

يوجد عدد من الأدوات البرمجية التي يمكن استعمالها لبناء المترجمات ولكن يوجد هناك أداتين أساسيتين هما الأكثر شهرة لذلك هما:

- **LEX**
- **BISON**

الأداة البرمجية (LEX):

وهي أداة تقوم بتوليد محلل مفردات أو ما يتم تسميته عادة بـ Scanner مكتوب بلغة C حيث يتم استخدام قوالب جاهزة لمطابقة السلاسل المحرفية الموجودة في الكود المصدر وتحويلها إلى tokens

مثلاً عندما يصادف الماسح (Scanner) المتغير x في سلسلة الدخل فإنه يرسل المفردة (token) التي تمثل هذا المتغير إلى المحلل القواعدي مثلاً <id> ، كما يرسل له أيضا اسم المتغير ويقوم بإدخاله إلى جدول الرموز وتعيين خصائصه كنوعه

وقيمته وغيرها من الخصائص.

الأداة البرمجية (BISON or YACC)

تولد هذه الأداة شيفرة بلغة C لمحلل قواعدي ويعرف أيضا بالمعرب (parser). يستعمل المعرب قواعد اللغة لتحليل الـ Token القادمة من الـ Scanner ويبني منها شجرة الإعراب الموافقة لتسلسل الـ Token.

في حالة عدم مطابقة العبارة للنحو، يصدر الـ BISON رسالة خطأ اعتماداً على تابع الخطأ المزود به (سيتم استعراضه لاحقاً عند شرح بنية ملف parser).

سنقوم في هذا المقرر باستخدام لغة البايثون مع مكتبة PLY وهو تطبيق في لغة البايثون لأداة LEX وYACC وهو يتكون من نموذجين منفصلين lex.py لعملية المسح وتحليل المفردات.

ونموذج yacc.py لعملية الإعراب.

تمارين:

ابحث في الانترنت عن التعابير النظامية، واكتب كود بلغة البايثون للتعرف عن القوالب التالي:

- قالب الأعداد العشرية.
- قالب الإيميل يقبل الأشكال التالية:

- http://www.[name of the website].com
- https://www.[name of the website].com

انتهت الجلسة