كلية العلوم

القسم : الرياضيات

السنة : الرابعة

المادة : ذكاء صنعي

المحاضرة : الثالثة /عملي/

{{ مكتبة A to Z }}

Facebook Group : مكتبة A to Z

عملي ذكاء صنعي

الجلسة الثالثة

م.ريم بصل

م.ريم رقبو

تعريف الاستدعاء الذاتي :
هو قابلية البرنامج استدعاء نفسه، أي سوف تعيد الدالة تكرار نفسها (استدعاء نفسها) داخل نفسها.

وفي كل استدعاء ذاتي يجب إن يتوفر الشرطين الأساسيين وهما:
١- **شرط التوقف،** إذ يتوقف البرنامج عند الوصول إلى شرط التوقف.
٢- **شرط** الاستدعاء **الذاتي،** إذ نقوم بعملية التكرار ولكن بمدخلات جديدة إلى أن نصل شرط التوقف.

وليس من الضروري أن يكون هناك شرط توقف أو استدعاء ذاتي واحد ربما يكون هناك أكثر من شرط توقف وعملية استدعاء ذاتي.
وهنالك نوعين من الاستدعاء الذاتي هما:

النوع الاول:
في هذا النوع كما في أي استدعاء ذاتي؛ يكون هناك شرط التوقف وشرط الاستدعاء الذاتي، وتكون ميزته انه بعد عملية الاستدعاء الذاتي يكون هناك جملة أو كثر برمجية توضع بعد عملية الاستدعاء هذه الجمل لا تنفذ بل توضع في المكدس (stack) وتنفذ عندما نصل إلى شرط التوقف، وتكون عملية تنفيذها بصورة معكوسة.

مثال:

أكتب برنامج لحساب (M) للأس (N) وان كل من (M) و(N) أعداد صحيحة.

أن كل رقم للأس (0) يكون قيمته (1) فلو أردنا حساب M^N فإننا نبدأ بتقليل (N) إلى أن تصبح قيمته (0) فلو كان لدينا 4^3 فحسابه يكون بالشكل الآتي::

$$4^3=4*4^2$$
$$4^2=4*4^1$$
$$4^1=4*4^0$$
$$4^0=1$$

إذ أن $4^0=1$ نبدأ بعدها باستخراج العمليات من المكدس بصورة معكوسة إلى أن نصل إلى أول عملية موضوعة في المكدس، إذ تكون النتيجة النهائية.

البرنامج:

شرط التوقف **power (_, 0, 1).**
power (M, N, P):-Z is N-1, power (M, Z, L), P is M*L.

حيث أن العبارة power (M, Z, L), هي الاستدعاء الذاتي . وهذه العملية P is M*L توضع في المكدس.

فلو كان الاستدعاء بالشكل الآتي:

power (4, 3, X) .

فان تتبع البرنامج يكون بالشكل الآتي سوف نطلق على شرط التوقف R1 وعلى الاستدعاء الذاتي R2.

Goal	R1	R2	Stack
Call power (4, 3, X)	Fail	M=4 N=3 Z=2 call power (4, 2, L)	P=4*L
Call power (4, 2, L)	Fail	M=4 N=2 Z=1call power (4, 1, L)	P=4*L
Call power (4, 1, L)	Fail	M=4 N=1 Z=0 call power(4, 0, L)	P=4*L
Call power (4, 0, L)	هنا يتم التطابق L يأخذ قيمة ١		

بعدما يتم تطابق شرط التوقف مع power(4, 0, L) إذ أن الإشارة (_) تعني بغض النظر عن فان (0) يتطابق مع (0) في شرط التوقف فان (L) تصبح قيمته (1) بعدها يتم استخراج العمليات من المكدس إلى أن نصل إلى أول عملية موضوعه داخل المكدس.

فبالعودة للمكدس:
نبدأ من العملية الاخيرة ونعوض

P=4*L (قيمة L هي واحد نعوض)
P=4*1=4

الان نعوض القيمة الناتجة بالعملية التالية :

P=4*4=16

نعوض النتيجة بالعملية الاولى بالمكدس :

P=4*16=64

اذا وصلنا للنتيجة النهائية (P=64)

مثال٢ :

أكتب برنامج لإيجاد حاصل قسمة عددين صحيحين بدون استخدام الدالة (div).

أن قسمة عددين صحيحين باستخدام إيعاز (div) لا يحتوي على كسور الناتج يكون أعداد صحيحة. فمثلا حاصل قسمة (9) على (4) تكون (2) والباقي (1). نلاحظ أن عملية تتضمن أنه نقوم بطرح الرقم الثاني من الأول ومن ثم نستخدم الرقم الناتج وهكذا إلى أن نصل إلى نتيجة يكون الرقم الناتج أصغر من الرقم الثاني.

لو اخذنا مثلا 14div3=4 يمكن أن نحصل على نفس الناتج بالطريقة الآتية:

14-3=11
11-3=8
8-3=5
5-3=2

وعندما نصل إلى (2) نتوقف لان (2) أصغر من (3) ونلاحظ عدد مرات تكرر الطرح هو (4) وهو ناتج تقسيم 14على 3.
مثال (٣٤): جد حاصل تقسيم 17على 5النتيجة تكون 3كما ياتي:

17-5=12
12-5=7
7-5=2

هنا نتوقف لان (2) أصغر من (5). وكما معروف فان حاصل قسمة رقم على رقم أكبر منه تكون (0) لذلك عند كتابه البرنامج يوضع هذا كشرط توقف.

البرنامج:

```
divno (M, N, 0):- N>M.
divno(M, N, Z):-L is M-N , divno(L, N, D) , Z is D+1.
```

حيث أن العبارة divno(L, N, D)هي عبارة الاستدعاء الذاتي وهذه العملية Z is D+1 توضع في المكدس.

فلو كان الاستدعاء بالشكل الآتي:

divno(11, 3, A).

فان تتبع البرنامج يكون بالشكل الآتي سوف نطلق على شرط التوقف R1 وعلى الاستدعاء الذاتي R2.

Goal	R1	R2	Stack
Call divno(11, 3, A)	M=11 N=3 Fail	M=11 N=3 L=8 call divno(8, 3, D)	Z=D+1
Call divno(8, 3, D)	M=8 N=3 Fail	M=8 N=3 L=5 call divno(5, 3, D)	Z=D+1
Call divno(5, 3, D)	M=5 N=3 Fail	M=5 N=3 L=2 call divno(2, 3, D)	Z=D+1
Call divno(2, 3, D)	M=2 N=3 هنا يتحقق شرط التوقف		

بعد تحقق شرط التوقف فان قيمة المتغير (D) سوف تصبح (0) لذلك نبدأ باستخراج العمليات الموجودة بالمكدس، إذ في كل مرة يزداد قيمته بمقدار (1) وبما انه هناك ثلاثة عمليات فالناتج يكون (3) وهو حاصل قسمة 11على 3.

مثال ٣:

اكتب برنامج لحساب مجموع الاعداد من 0 الى N بلغة prolog.

```
sum_to(0,0).
sum_to(N,S):- N>0 , N1 is N-1 , sum_to(N1,S1) , S is S1+N.
```

لو كان الاستدعاء

```
sum_to(5,S).
```

Goal	R1	R2	stack
Call sum_to(5,S).	N=5 Fail	N=5 5>0 N1=4 **sum_to(4,S1)**	S is S1+5
Call **sum_to(4,S1)**	N=4 Fail	N=4 4>0 N1=3 **sum_to(3,S1)**	S is S1+4
Call **sum_to(3,S1)**	N=3 Fail	N=3 3>0 N1=2 **sum_to(2,S1)**	S is S1+3
Call **sum_to(2,S1)**	N=2 Fail	N=2 2>0 N1=1 **sum_to(1,S1)**	S is S1+2
Call **sum_to(1,S1)**	N=1 Fail	N=1 1>0 N1=0 **sum_to(0,S1)**	S is S1+1
Call **sum_to(0,S1)**	N=0 هنا يتحقق الشرط نعود للمكدس ونعوض		

بعد تحقق شرط التوقف فان قيمة المتغير S1 سوف تصبح (0) لذلك نبدأ باستخراج العمليات الموجودة بالمكدس مع الانتباه ان اول عملية تدخل للمكدس هي اخر عملية تخرج منه.

S = 0 + 1=1

نعوض هذه القيمة بالعملية الأعلى

S = 2 + 1=3

نتابع هكذا:

S = 3 + 3=6

S = 4 + 6=10

S = 5 + 10=15

اذا مجموع الاعداد من ١ حتى ٥ يساوي ١٥.

مثال ٤:
اكتب برنامج لحساب عاملي أي عدد بلغة prolog.

```
fact(0,1).
fact(N ,F):- N1 is N-1 , fact(N1 , F1) , F is F1 * N.
```

لو استدعينا

```
fact(3 , F).
```

الخرج هو F=6

مثال ٥:

اكتب برنامج لحساب مضاعف أي عدد مدخل

```
double(0,0).
double(N , D):- N>0 , N1 is N-1 , double(N1 , D1) , D is D1 + 2.
```

لو استدعينا

```
double(4 , D).
```

الخرج هو D = 8