أنواع البيانات :

1- بيانات حرفية

Symbol : كلمه واحده فقط أو حرف أو جمله بس بدون مسافات

String: " نـــص "

2- بيانات رقمية

الأوامر

الأمر	وظيفته	مثال
bind	تخصيص قيمه لـ Variable	(bind ? a 7) قيمه مفرده (bind $?a (create$ a b c d)) قيم متعدده
create$	انشاء مصفوفه او مجموعه من القيم	(create$ cairo "ali" 1 2 3)
member$	الفحص . يقوم باختبار قيمه معينه اذا كانت موجوده او لا ويرجع بترتيبها فى المجموعه او false اذا لم توجد	(member$ red (create$ red green blue)) Out put => 1 (member$ white (create$ red green blue)) Output => false
nth$	يقوم بكتابه قيمة العنصر المراد البحث عنه بترتيبه فى المجموعه	(nth$ 3 (create$ red green blue yellow)) Output => blue
assert	اضافه fact فى الذاكره	(assert (a 10))
read	يسمح بقراءة قيمة اثناء الـ run time	(bind ?*a* (read)) input => 7 Result => 7

كتابه Rule فى لغة الـ CLIPS

- ☒ File Menu => New
- ☒ تعتمد على Left hand side و Right hand side
- ☒ ازاى بقى اكتب Rule

ملحوظة :
عند تعديل Rule ويتم عمل Loadbuffer
يتم عمل overwrite ع الاقدم ويتم تسجيل الاحدث

ملحوظة :
اذا لم يوجد الشرط
معنى ذلك انه يعتمد على Intial facts

```
(defrule  name
   Condition  الشرط
  =>
Action  النتيجه
)
```

☒ مثال لطباعة متغير لناتج جمع الارقام (2-3-4)

```
(defrule  r1
 =>
  (bind  ?a (+ 2 3 4))
  (printout t  t  ?a  crlf)
)
```

كيفية تنفيذ وتشغيل الـcode
- ☒ الصفحه الخاصه بالـRule اعمل لها Load buffer
- ☒ الصفحه الخاصه بالـDialog اعمل لها Run

كتابة Rule لقيم متعددة

```
(defrule  r1
 =>
  (bind  $?a (Create$  red green blue))
  (printout t  t   $?a  crlf)
)
```

اوعى تنسى تعمل
- ☒ Load buffer
- ☒ Run

كتابة Rule للتأكد من الأمر member$

```
(defrule  r1
 =>
  (bind  $?a (Create$  red green blue))
  (bind   ?t   (member $ 5  $?a))
  (printout t  t   ?t  crlf)
)
```

خلى بالك هنا من t و ?t
- ☒ الاولى الخاصه بالطباعه ع الشاشه
- ☒ التانية متغير

تعريف متغير معين (عام) بالأمر (defglobal)

```
(defglobal   ?*a*   =   10)
(defglobal   ?*a*   =  (create$  1 4 5 7 8))
```

(single variable)
(multi variables)

ملحوظه
- ☒ القيمه فى الحاله دى بتتخزن فى global main
- ☒ Window => Global main

للتعديل على المتغير الـ global

يتم التعديل بالامر bind

```
(bind   ?*a*   4)
(bind   ?*a*   (+  ?*a*   30))
(bind   ?*a*  (*  2  4  5))
(bind    ?*a*    )        =>   يعود للقيمه الاولى له
```

☒ مثال لطباعة متغير عام والتعديل فيه واضافه قيم :

```
(defglobal   ?*a*   =  (create$  red green blue))
(printout   t  ?*a*  crlf )
(bind    ?*a*    (create$  1 2 3 4))
(bind     ?*a*     )
(bind    ?*a*    (create$  ?*a*   yellow ))
```

ملحوظه

اول سطرين
- ☒ تم تحديد متغير عام وفيه قيم الوان
- ☒ طباعته ع الشاشه

تالت سطر
- ☒ تم التعديل ع الامرر global وغيرت قيمة المتغير العام لارقام

رابع سطر
- ☒ يعود لقيمة الdefult اللى هى اول قيمه خالص

خامس سطر
- ☒ تم تعديل فى القيم وتم اضافة لون اضافى ع الالوان السابقه

كتابه كود باستخدام الامر (read)

```
(defrule   r1
   =>
 (printout   t    "Insert value : " )
(bind    ?a  (read))
(bind     ?b ( *  ?a   2))
 (printout  t  "Result is : "   ?b   crlf)
)
```

اوعى تنسى تعمل
Load buffer ☒
Run ☒

Example : write program to calculate The area and Circum of

- Circle
- Rectangle

كتابة Facts

انواع الـFacts

Ordered : بيانات غير مصنفة لايمكن تعديلها

Non ordered : بيانات مصنفة حسب الحقول لايهم الترتيب يمكن تعديلها

Facts	
Ordered facts	**Non Oredered Facts**
(stu ahmed 20 100)	(stu (name ahmed)(age 20) (deg 100))
(stu Zyad 19 80)	(stu (age 19)(deg 80)(name Zyad))
(stu Nadeen 70 18)	(stu (name Nadeen)(deg 70)(age 18))

كتابة Facts فى الذاكرة باستخدام الامر (assert)

```
(assert  (stu  ahmed  20  100 ))
(assert (a 10))
(assert  (a  10 20  30 ))
(assert ( a 20) (a 10 )  (a blue ))
```

F0 intial
F1 (stu ahmed 20 100)
F2 (a 10)
F3 (a 10 20 30)
F4 (a 20)
F5 (a blue)

لايجوز وضع اكثر من fact متطابقه

الا اذا تم التعديل الاختيار fact duplication

عن طريق

Execution => option => fact duplication

تطبيق Facts و Rule

```
(defrule  r1
   =>
 (assert  (a) )
)
(defrule  r2
   (a)
   =>
 (assert  (b) )
)
(defrule  r3
   (b)
   =>
 (printout  t  ok  crlf )
)
```

F0 initial
F1 (a)
F2 (b)

اوعى تنسى تعمل
Load buffer ☒
Run ☒

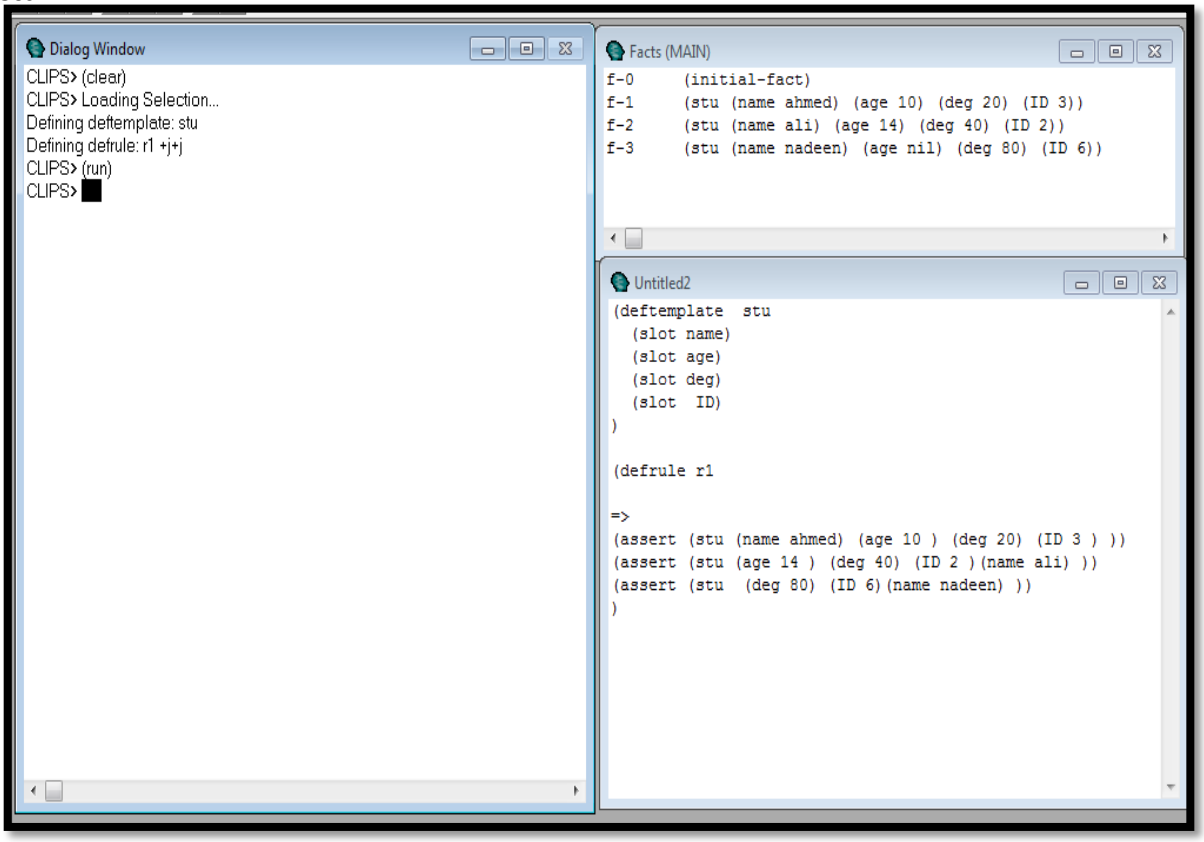
الأمر Clear والأمر Reset

Clear : بتسمح كل حاجه فى الذاكره ماعدا Initial facts

Reset : بتمسح كل حاجه وبتحمل الdeffacts

تعريف الـ Non ordered facts

```
(def template  stu
  (slot name )
  (slot age )
  (slot deg )
  (slot  ID  )
)
```



الترقيم التلقائى

```
(deftemplate  stu
  (slot name)
  (slot age)
  (slot deg)
  (slot  ID (default-dynamic (gensym*)))
)

(defrule r1

=>
(assert (stu (name ahmed) (age 10 ) (deg 20)))
(assert (stu (age 14 ) (deg 40) (name ali) ))
(assert (stu  (deg 80) (name nadeen) ))
)
```