**Εργασία Επεξεργασίας Φυσικής Γλώσσας**

Διδάσκων: Θέμης Παναγιωτόπουλος

**A picture containing text, electronics

Description automatically generated**

**Σπουδάστρια: Καλογήρου Στυλιανή**

**Α.Μ: Π18181**

**Ιούλιος 2021**

# Περιεχόμενα

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

[Περιεχόμενα 2](#_Toc78839565)

[Γενικές Πληροφορίες άσκησης 3](#_Toc78839566)

[Θέμα 1 4](#_Toc78839567)

[Λύση Ερωτήματος α) 4](#_Toc78839568)

[Λύση Ερωτήματος β) 5](#_Toc78839569)

[Θέμα 2 6](#_Toc78839570)

[Λύση 7](#_Toc78839571)

[Θέμα 3 11](#_Toc78839572)

[Λύση 11](#_Toc78839573)

[Επεκτάσεις/Εργαλεία 40](#_Toc78839574)

[Πηγές 40](#_Toc78839575)

# Γενικές Πληροφορίες άσκησης

Η εργασία υλοποιήθηκε με την χρήση της Prolog

* Η **Prolog**είναι μια [γλώσσα](https://el.wikipedia.org/wiki/%CE%93%CE%BB%CF%8E%CF%83%CF%83%CE%B1_%CF%80%CF%81%CE%BF%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1%CF%84%CE%B9%CF%83%CE%BC%CE%BF%CF%8D) [λογικού προγραμματισμού](https://el.wikipedia.org/w/index.php?title=%CE%9B%CE%BF%CE%B3%CE%B9%CE%BA%CF%8C%CF%82_%CF%80%CF%81%CE%BF%CE%B3%CF%81%CE%B1%CE%BC%CE%BC%CE%B1%CF%84%CE%B9%CF%83%CE%BC%CF%8C%CF%82&action=edit&redlink=1) γενικής χρήσης που κυρίως χρησιμοποιείται στον τομέα της [τεχνητής νοημοσύνης](https://el.wikipedia.org/wiki/%CE%A4%CE%B5%CF%87%CE%BD%CE%B7%CF%84%CE%AE_%CE%BD%CE%BF%CE%B7%CE%BC%CE%BF%CF%83%CF%8D%CE%BD%CE%B7). Δημιουργήθηκε στις αρχές του 1970 από τους [Ρόμπερτ Κοβάλσκι](https://el.wikipedia.org/w/index.php?title=%CE%A1%CF%8C%CE%BC%CF%80%CE%B5%CF%81%CF%84_%CE%9A%CE%BF%CE%B2%CE%AC%CE%BB%CF%83%CE%BA%CE%B9&action=edit&redlink=1) και [Alain Colmeraue](https://el.wikipedia.org/w/index.php?title=Alain_Colmeraue&action=edit&redlink=1" \o "Alain Colmeraue (δεν έχει γραφτεί ακόμα)). Το όνομα Prolog το έβγαλε ο συνεργάτης του Kowalski' Philippe Roussel και είναι συντομογραφία του [γαλλικού](https://el.wikipedia.org/wiki/%CE%93%CE%B1%CE%BB%CE%BB%CE%B9%CE%BA%CE%AE_%CE%B3%CE%BB%CF%8E%CF%83%CF%83%CE%B1) «PROgramation et LOGique» («Προγραμματισμός και Λογική»).
* Η επεξεργασία φυσικής γλώσσας (NLP) είναι ένα υποπεδίο της γλωσσολογίας, της επιστήμης των υπολογιστών και της τεχνητής νοημοσύνης που ασχολείται με τις αλληλεπιδράσεις μεταξύ υπολογιστών και ανθρώπινης γλώσσας, ιδιαίτερα τον τρόπο προγραμματισμού υπολογιστών για την επεξεργασία και ανάλυση μεγάλων ποσοτήτων φυσικών γλωσσικών δεδομένων. Ο στόχος είναι ένας υπολογιστής ικανός να «κατανοήσει» το περιεχόμενο των εγγράφων, συμπεριλαμβανομένων των αποχρώσεων της γλώσσας μέσα σε αυτά. Η τεχνολογία μπορεί στη συνέχεια να εξαγάγει με ακρίβεια πληροφορίες και πληροφορίες που περιέχονται στα έγγραφα, καθώς και να κατηγοριοποιήσει και να οργανώσει τα ίδια τα έγγραφα. Οι προκλήσεις στην επεξεργασία φυσικής γλώσσας περιλαμβάνουν συχνά αναγνώριση λόγου, κατανόηση φυσικής γλώσσας και δημιουργία φυσικής γλώσσας.
* Η επεξεργασία της φυσικής γλώσσας έχει τις ρίζες της στη δεκαετία του 1950. Tο 1950, ο Alan Turing δημοσίευσε ένα άρθρο με τίτλο "Computing Machinery and Intelligence" το οποίο πρότεινε αυτό που σήμερα ονομάζεται δοκιμή Turing ως κριτήριο νοημοσύνης, ένα έργο που περιλαμβάνει την αυτοματοποιημένη ερμηνεία και δημιουργία φυσικής γλώσσας, αλλά εκείνη τη στιγμή δεν αρθρώνεται. ως πρόβλημα ξεχωριστό από την τεχνητή νοημοσύνη.

# Θέμα 1

Θέμα 1ο (20 μονάδες)

(α) Με ποιά γραμματική σε μορφή DCG μπορούμε να αναγνωρίσουμε την

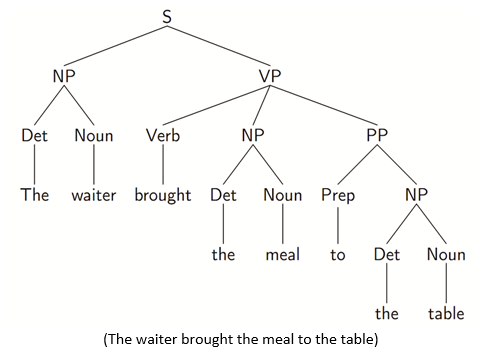
πρόταση : [the, waiter, brought, the, meal, to, the, table], σύμφωνα με το

παρακάτω σχήμα;

(β) Με ποιά γραμματική σε μορφή DCG μπορούμε να παράγουμε σε μορφή

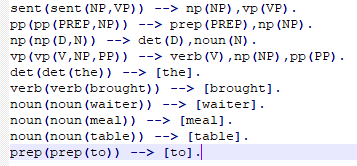
functor το συντακτικό δένδρο για την αναγνώριση της πρότασης :

[the, waiter, brought, the, meal, to, the, table], σύμφωνα με το παρακάτω σχήμα;



## Λύση Ερωτήματος α)

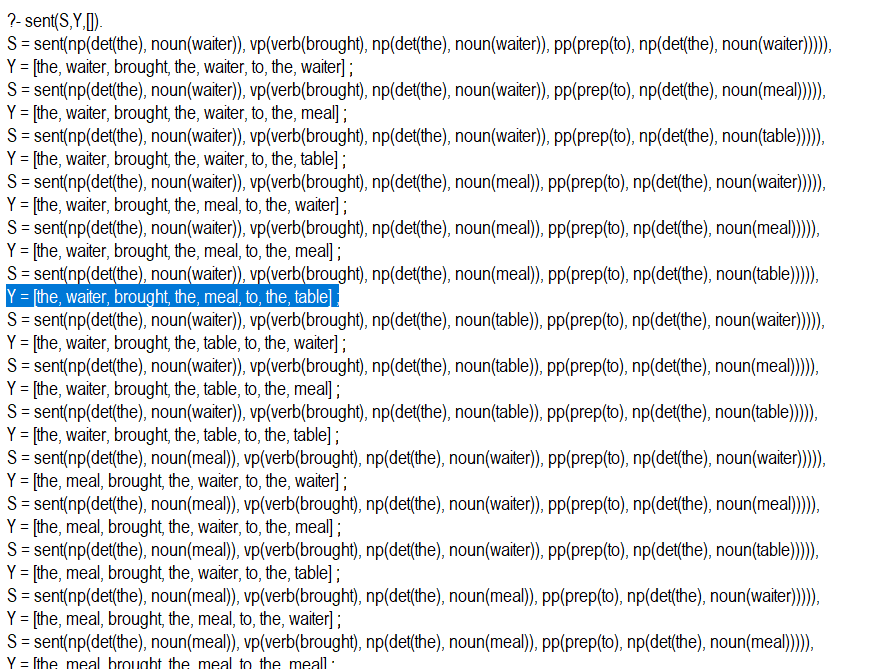
Μια Γραμματική, δηλαδή ένα σύνολο συντακτικών κανόνων, μπορεί εύκολα να διατυπωθεί σε DCGs. Για παράδειγμα έστω ότι έχουμε το παρακάτω σύνολο συντακτικών κανόνων ως ένα πρόγραμμα Prolog :



Δηλαδή, η γραμματική μας μπορεί να χρησιμοποιηθεί ως «γεννήτορας προτάσεων» που είναι συντακτικά ορθές, ανεξάρτητα αν βγάζουν νόημα, αυτό αφορά την σημασιολογία.

Επίσης η ίδια γραμματική μπορεί να χρησιμοποιηθεί για την αναγνώριση της συντακτικής ορθότητας μιας πρότασης :

**Αποτέλεσμα:**



## Λύση Ερωτήματος β)

Αν επιθυμούμε η Prolog να παράγει το ίδιο το συντακτικό δένδρο σαν έξοδο, πρέπει να διαμορφώσουμε λίγο τους κανόνες της γραμματικής μας ως εξής :

s(s(NP,VP)) --> np(NP), vp(VP).

Pp(pp(Rrep,NP)) --> prep(PREP), np(NP).

np(np(D,N)) --> det(D), noun(N).

vp(vp(V,NP,PP)) --> verb(V), np(NP),pp(PP).

det(det(the)) --> [the].

verb(verb(brought)) --> [brought].

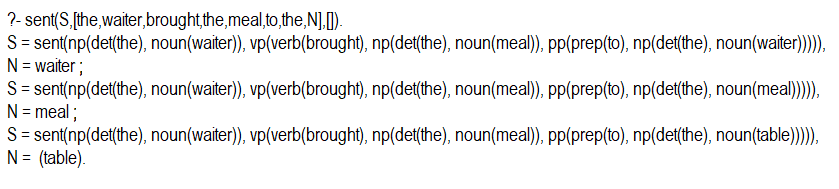
noun(noun(waiter)) --> [waiter].

noun(noun(meal)) --> [meal].

noun(noun(table)) --> [table].

Prep(prep(to)) --> [to].

Αποτέλεσμα:



# Θέμα 2

Θέμα 2ο (20 μονάδες)

Το παρακάτω πρόγραμμα αναγνωρίζει και υπολογίζει αριθμητικές εκφράσεις

όπως αναλύθηκε στο θεωρητικό μέρος. Να αναπτυχθεί ένα αντίστοιχο

πρόγραμμα όπου οι αριθμοί είναι διαδικοί και οι αριθμητικές εκφράσεις είναι

αντίστοιχα αριθμητικές εκφράσεις διαδικών αριθμών

expression(Value) --> number(Value).

expression(Value) --> number(X), [+], expression(V), {Value is X+V}.

expression(Value) --> number(X), [-], expression(V), {Value is X-V}.

expression(Value) --> number(X), [\*], expression(V), {Value is X\*V}.

expression(Value) --> number(X), [/], expression(V), {V/=0, Value is X/V}.

expression(Value) --> left\_parenthesis, expression(Value), right\_parenthesis.

left\_parenthesis --> ['('].

right\_parenthesis --> [')'].

number(X) --> digit(X).

number(Value) --> digit(X), number(Y),{numberofdigits(Y,N), Value is X\*10^N+Y}.

digit(0) --> [0].

digit(1) --> [1].

digit(2) --> [2].

digit(3) --> [3].

digit(4) --> [4].

digit(5) --> [5].

digit(6) --> [6].

digit(7) --> [7].

digit(8) --> [8].

digit(9) --> [9].

numberofdigits(Y,1) :- Z is Y/10, Z<1.

numberofdigits(Y,N) :-

Z is (Y - mod(Y,10))/10,

numberofdigits(Z,N1),

N is N1+1.

## Λύση

Το παράδειγμα της λογικής σύζευξης των δυαδικών αριθμών, μπορεί να επεκταθεί εύκολα και σε άλλες πράξεις του συγκεκριμένου πεδίου. Θα εξετάσουμε στα επόμενα, μερικές από αυτές, ξεκινώντας με ένα απλό

κατηγόρημα που εξασφαλίζει ότι ένας όρος είναι δυαδικός αριθμός.

Το κατηγόρημα binary\_number/1, δέχεται ως όρισμα έναν όρο και επιτυγχάνει όταν ο όρος είναι λίστα, τα στοιχεία της οποίας είναι μηδέν ή ένα.

Σύμφωνα με τα προηγούμενα, το κατηγόρημα θα διατρέχει

αναδρομικά την λίστα, εξασφαλίζοντας ότι κάθε στοιχείο είναι 0 ή 1. Η υλοποίηση του παραπάνω είναι σχετικά απλή:

binary\_number([X]):-

zero\_or\_one(X).

binary\_number([X|Rest]):-

zero\_or\_one(X),

binary\_number(Rest).

Tο παραπάνω κατηγόρημα, εξασφαλίζεται κάθε φορά ότι η μεταβλητή Χ είναι 0 ή 1, μέσω του κατηγορήματος zero\_or\_one/1. Το τελευταίο, υλοποιείται απλά σαν ένα ζεύγος, γεγονότων:

zero\_or\_one(0).

zero\_or\_one(1).

Η συμπεριφορά του κατηγορήματος είναι η ακόλουθη:

?- binary\_number([1,0,0,1,1]).

true

?- binary\_number([1,0]).

true .

?- binary\_number([1,0,3]).

false.

Το κατηγόρημα μπορεί να κληθεί με μια μεταβλητή σαν όρισμα:

?- binary\_number(Num).

Num = [0] ;

Num = [1] ;

Num = [0, 0] ;

Num = [0, 1] ;

Num = [0, 0, 0] ;

Num = [0, 0, 1] ;

Num = [0, 0, 0, 0] ;

Num = [0, 0, 0, 1] ;

Num = [0, 0, 0, 0, 0] ;

Num = [0, 0, 0, 0, 1] ;

Num = [0, 0, 0, 0, 0, 0] ;

...

Το σημείο αυτό χρήζει λίγο περισσότερης εξήγησης. Αν και θα περίμενε κάποιος να εξαντλούνται οι αριθμοί N-bits (για παράδειγμα 2 bits) πριν το κατηγόρημα επιστρέψει λύσεις με μεγαλύτερο αριθμό bits, επιστρέφονται όλο και μεγαλύτερου μήκους αριθμοί, οι οποίοι διαφέρουν μόνο στο τελευταίο ψηφίο. Αυτό οφείλεται στο γεγονός ότι ο μηχανισμός οπισθοδρόμησης, ερευνά συστηματικά το δένδρο αναζήτησης, ελέγχοντας εξαντλητικά, πρώτα τα αριστερότερα κλαδιά του δένδρου (αναζήτηση κατά βάθος). Καθώς, όμως είναι δυνατό να υπάρχουν (θεωρητικά) αριθμοί με άπειρα bits, το κατηγόρημα παράγει άπειρες λύσεις, επιστρέφοντας αριθμούς με όλο και μεγαλύτερο αριθμό ψηφίων.

Για να παράξει το κατηγόρημα όλους τους αριθμούς Ν bit, τότε θα έπρεπε πριν από την κλήση της binary\_number/1, να οριστεί ότι ή λίστα είναι Ν στοιχείων. Για παράδειγμα η ακόλουθη ερώτηση επιστρέφει διαδοχικά όλους τους δυαδικούς μήκους 3 bit.

**?- length(L,3),binary\_number(L).**

**L = [0, 0, 0] ;**

**L = [0, 0, 1] ;**

**L = [0, 1, 0] ;**

**L = [0, 1, 1] ;**

**L = [1, 0, 0] ;**

**L = [1, 0, 1] ;**

**L = [1, 1, 0] ;**

**L = [1, 1, 1] ;**

**false.**

Η πράξη της λογικής διάζευξης δυαδικών αριθμών (bitwise\_or/3) είναι παρόμοια με εκείνη της λογικής σύζευξης, που παρουσιάστηκε στη αρχή του κεφαλαίου. Μοναδική διαφορά είναι ότι θα πρέπει να οριστεί κατάλληλα η λογική διάζευξη μέσω του κατηγορήματος bit\_or/3, αντίστοιχο εκείνου του bit\_and/3:

**bit\_or(0,0,0).**

**bit\_or(0,1,1).**

**bit\_or(1,0,1).**

**bit\_or(1,1,1).**

Ακολουθώντας μια παρόμοια δηλωτική περιγραφή, η υλοποίηση της λογικής σύζευξης είναι η ακόλουθη:

**bitwise\_or([Bit1], [Bit2], [BitR]):-**

**bit\_or(Bit1,Bit2,BitR).**

**bitwise\_or([Bit1|RestBits1], [Bit2|RestBits2], [BitR|Result]):-**

**bit\_or(Bit1, Bit2, BitR),**

**bitwise\_or(RestBits1, RestBits2, Result).**

Η συμπεριφορά του κατηγορήματος είναι:

**?- bitwise\_or([1,1,1],[1,0,0],Res).**

**Res = [1, 1, 1] .**

**?- bitwise\_or([0,0,0],[1,0,1],Res).**

**Res = [1, 0, 1] .**

**?- bitwise\_or([0,1,0],[1,0,1],Res).**

**Res = [1, 1, 1] .**

**?- bitwise\_or([0,2,0],[1,0,1],Res).**

**false.**

Η τελευταία ερώτηση, δείχνει το πως η Prolog μπορεί να κάνει “έλεγχο τύπων”, περιορίζοντας την επιτυχία της κλήσης του κατηγορήματος της πράξης της λογικής σύζευξης μόνο σε λίστες οι οποίες αποτελούν από 0

και 1. Προφανώς με παρόμοιο τρόπο μπορεί να οριστεί η πράξη της αποκλειστικής λογικής διάζευξης (XOR). Ενδιαφέρον παρουσιάζουν οι πράξεις της λογικής ολίσθησης. Η λογική ολίσθηση Ν θέσεων προς τα αριστερά, αφορά την “μετατόπιση” των Ν ψηφίων του δυαδικού αριθμού προς τα αριστερά με ταυτόχρονη απόρριψη των Ν δεξιότερων ψηφίων και συμπλήρωση των κενών θέσεων που προκύπτουν με 0.

# Θέμα 3

Θέμα 3ο (60 μονάδες)

Χρησιμοποιείστε το σύνολο του κώδικα που σας δόθηκε σε Prolog για την

«κατανόηση» μιας μικρής ιστορίας. Πρέπει να παράγετε τα περιεχόμενα της

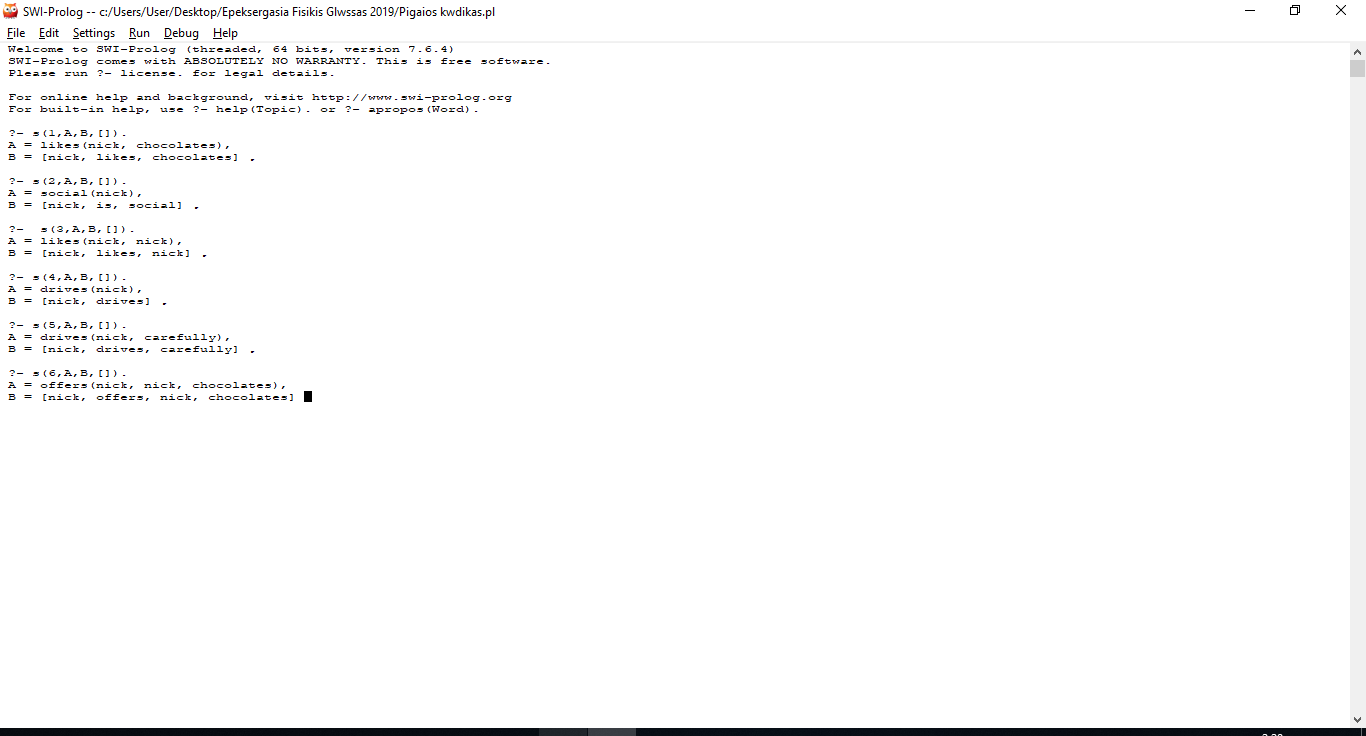
βάσης γνώσης και να μπορείτε να εισάγετε νέες πληροφορίες από το

πληκτρολόγιο, να κάνετε ερωτήσεις στην βάση γνώσης, κ.λπ., παρόμοιες με αυτές της πρότυπης λύσης.

## Λύση

Ο τύπος parser που χρησιμοποιήθηκε για την εργασία είναι ο **DCG**. Αντίθετα όμως με τις περισσότερες υλοποιήσεις, η συγκεκριμένη είναι λιγότερα ίσως επιστημονικά αποδεκτή και η δομή της είναι περισσότερο πρακτική, παρά λογική. Αντί λοιπόν να βασίζεται σε έναν τύπο πρότασης που ανάλογα με τις διακλαδώσεις παράγει όλες τις πιθανές προτάσεις, έχουμε έξι (6) τύπους γνώσης που αντιστοιχούν σε **6 τύπους προτάσεων**. Παρακάτω βλέπουμε τους **6 τύπους γνώσης** και τον τύπο πρότασης που την παράγει. Τα παραδείγματα είναι παρμένα κατευθείαν από το πρόγραμμα.

**Παραδείγματα παραγωγής όλων των τύπων προτάσεων:**

(Α=semantics, B= φυσική γλώσσα)

**\* s(Type, Semantics, X, []).**

Οι έξι τύποι γνώσεις είναι κατά κάποιον τρόπο ανθρωποκεντρικοί και αφορούν γεγονότα που γίνονται σε αόριστο χρόνο.

**Ας δούμε την γραμματική η οποία επιτρέπει αυτές τις προτάσεις:**

s(1,Sem) --> pn(N), vp(1,V,N1), {Sem=..[V,N,N1]}.

s(2,Sem) --> pn(N), vp(2,\_,A), {Sem=..[A,N]}.

s(3,Sem) --> pn(N), vp(3,V,N1), {Sem=..[V,N,N1]}.

s(4,Sem) --> pn(N), iv(V,s), {Sem=..[V,N]}.

s(5,Sem) --> pn(N), iv(V,s),adv(A), {Sem=..[V,N,A]}.

s(6,Sem) --> pn(N), tv(V,s),pn(N1),np(N2), {Sem=..[V,N,N1,N2]}.

Όπως φαίνεται, το verbal phrase ακολουθεί την λογική των προτάσεων και έχει τρεις (3) τύπους. Για να γίνουν κατανοητές οι παραπάνω προτάσεις δίνετε η επεξήγηση των όρων.

**Πρώτα, δίνονται οι απλοί όροι που αντιστοιχούν σε λέξεις και στην συνέχεια οι όροι που αντιστοιχούν σε ακολουθίες λέξεων:**

**2)1) Ορολογία:**

**a) Proper Nouns (pn):** Αναφέρεται σε ονόματα και τοποθεσίες. Μπορεί να είναι υποκείμενο αλλά και αντικείμενο σε μια πρόταση και δεν παίρνει άρθρο.

**Παραδείγματα προγράμματος:**

pn(nick) -->[nick].

pn(rebecca) -->[rebecca].

pn(alex) -->[alex].

pn(zina) -->[zina].

**b) Intransitive Verbs (iv):** Ρήμα που δεν δέχεται αντικείμενο. Αναφέρει πράξη του υποκειμένου και μπορεί από μόνο του να αποτελεί φράση (verbal phrase πιο συγκεκριμένα).

**Παραδείγματα προγράμματος:**

iv(drives,s) -->[drives]. iv(drives,q) -->[driving].

iv(swims,s) -->[swims]. iv(swims,q) -->[swimming].

Το ίδιο ρήμα (iv(drives,s)) & (iv(swims,s)) το βρίσκουμε σε δυο διαφορετικούς τύπους (iv(drives,s)) & (iv(swims,s) . Οι τύποι συνδέονται μεταξύ τους με το κοινό όνομα και είναι τα αρχικά για τα sentence και query όπου και χρησιμοποιούνται πιο συχνά.

(π.χ.: Nick drives / Who is **driving**? & Rebecca swims/ Who is **swimming**?)

**c) Auxiliary Verbs (av):** Auxiliary verbs ή αλλιώς helping verbs ονομάζονται τα ρήματα που λειτουργούν ως βοηθητικά στα ρήματα που τα ακολουθούν, δίνοντας τους ή συμπληρώνοντας το νόημα τους. **Παραδείγματα προγράμματος:**

av(is) -->[is].

av(does) -->[does].

**d) Verbs (v):** Ο όρος χρησιμοποιείτε για όλα τα ρήματα τα οποία δέχονται αντικείμενο. Μεταξύ τους υπάρχουν πολλοί τύποι ρημάτων (emotion verbs, action verbs, helping verbs). Δίνονται πάλι όπως στα iv δύο τύποι συνδεόμενοι για χρήση σε διαφορετικούς τύπους προτάσεων. Στην παραγωγή γνώσης παίρνουν δυο ορίσματα, το υποκείμενο και το αντικείμενο, σε αυτήν την σειρά.

**Παραδείγματα προγράμματος:**

v(likes,s) -->[likes]. v(likes,q) -->[like].

v(adores,s) -->[adores]. v(adores,q) -->[adore].

v(hates,s) -->[hates]. v(hates,q) -->[hate].

v(finds,s) -->[finds]. v(finds,q) -->[find].

v(loses,s) -->[loses]. v(loses,q) -->[lose].

**e) Transitive Verbs (tv):** Μεταβατικά ρήματα, δέχονται υποκείμενο, και πάνω από ένα αντικείμενο. Ένα άμεσο και ένα απλό.

**Παραδείγματα προγράμματος:**

1. «Η Ρεβέκκα προσφέρει μία σοκολάτα στον Νίκο»

σοκολάτα 🡪 αντικείμενο & Νίκο 🡪 άμεσο αντικείμενο

1. «Ο Άλεξ φωνάζει στην κουζίνα της Ζήνας»

κουζίνα 🡪 αντικείμενο & Ζήνα 🡪 άμεσο αντικείμενο

1. «Η Ρεβέκκα δίνει ένα κουτάλι στην Ζήνα»

κουτάλι 🡪 αντικείμενο & Ζήνα 🡪 άμεσο αντικείμενο

1. «Η Ζήνα επιτίθεται με το κουτάλι στον Άλεξ»

κουτάλι 🡪 αντικείμενο & Άλεξ 🡪 άμεσο αντικείμενο

1. «Ο Άλεξ κυνηγάει στο σπίτι την Ζήνα»

σπίτι 🡪 αντικείμενο & Ζήνα 🡪 άμεσο αντικείμενο

Χρησιμοποιείτε ξεχωριστά καθώς στην βάση γνώσης έχει τρία γνωρίσματα (1 υποκείμενο, 2 αντικείμενα) και χρησιμοποιούνται τρείς τύποι του.

**Παραδείγματα προγράμματος με τις i), ii), iii), iv), v) προτάσεις αντίστοιχα:**

tv(offers,s) -->[offers]. tv(offers,q) -->[offer]. tv(offers,q2) -->[offering].

tv(shouts,s) -->[shouts]. tv(shouts,q) -->[shout]. tv(shouts,q2) -->[shouting].

tv(gives,s) -->[gives]. tv(gives,q) -->[give]. tv(gives,q2) -->[giving].

tv(attacks,s) -->[attacks]. tv(attacks,q) -->[attack]. tv(attacks,q2) -->[attacking].

tv(chases,s) -->[chases]. tv(chases,q) -->[chase]. tv(chases,q2) -->[chasing].

**f) Adverb (adv):** Μέρος του λόγου που περιγράφει ένα ρήμα, χρησιμοποιείτε μετά από αυτόκαι αποτελείτε μόνο από μια λέξη. **Παραδείγματα προγράμματος:**

adv(carefully) -->[carefully].

adv(correctly) -->[correctly].

**g) Adjectives (adj):** Στην ελληνική γλώσσα τα επίθετα, χρησιμοποιούνται ως χαρακτηρισμός προσώπου στο συγκεκριμένο πρόγραμμα.

**Παραδείγματα προγράμματος:**

adj(social) -->[social].

adj(naughty) -->[naughty].

adj(fat) -->[fat].

adj(beautiful) -->[beautiful].

**h) Determiner (det):** Ονομάζονται οι λέξεις που είτε χρησιμοποιούνται για την κλήση ενός ουσιαστικού (άρθρα όπως a, the) είτε λέξεις που χρησιμοποιούνται για να δηλώσουν κτίση (my, his). Σε κάθε περίπτωση χρησιμοποιούνται πριν το ουσιαστικό σε φράση. Στο πρόγραμμα έχει τα άρθρα (a, the) και τις αντωνυμίες (who, what, how) αλλά μόνο τα πρώτα δηλώνονται ως det για πρακτικούς λόγους. Στο πρόγραμμα, determiners ορίζονται τα άρθρα. Ακόμα, δηλώνετε και η κενή λίστα, καθώς δεν χρειάζεται πάντα ένα άρθρο πριν ένα ουσιαστικό.

**Παραδείγματα προγράμματος:**

det(null) -->[].

det(a) -->[a].

det(the) -->[the].

**i) Noun (n):** Τα ουσιαστικά στο πρόγραμμα χρησιμοποιούνται ως αντικείμενα και μόνο. Τον ρόλο του υποκειμένου παίρνουν τα proper nouns, ένα υποείδος ουσιαστικών που εξηγείτε παραπάνω.

**Παραδείγματα προγράμματος:**

n(chocolates) -->[chocolates].

n(chocolate) -->[chocolate].

n(theater) -->[theater].

n(cinema) -->[cinema].

n(turtle) -->[turtle].

n(rabbit) -->[rabbit].

n(kitchen) -->[kitchen].

n(spoon) -->[spoon].

n(home) -->[home].

**j) Noun Phrase (np):** Φράση που δηλώνει το υποκείμενο σε μια πρόταση. Κανονικά μπορεί να αποτελείτε και από ένα proper noun αλλά αυτή η δήλωση (np(N):- pn(N)) δεν γίνετε για πρακτικούς λόγους και όχι επιστημονικής ακρίβειας.

**Ένας άλλος τύπος φράσης είναι o εξής και ο μόνος που χρησιμοποιείτε στο συγκεκριμένο πρόγραμμα, πέρα από το proper noun:**

np(N) --> det(\_),n(N).

**k) Verbal Phrase:**

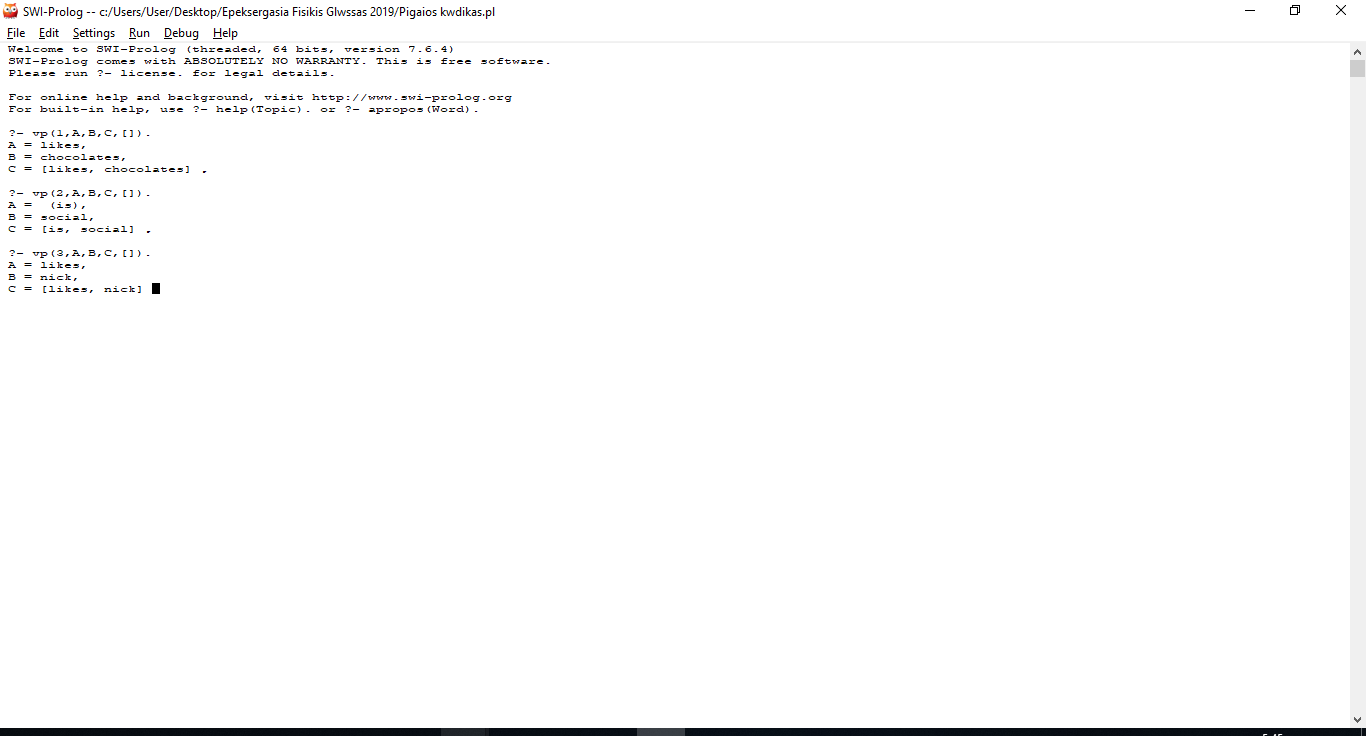
**Φράσεις που περιέχουν το ρήμα και το αντικείμενο (αν αυτό υπάρχει):**

vp(1,V,N) --> v(V,s),n(N).

vp(2,is,A) --> av(is),adj(A).

vp(3,V,N) --> v(V,s),pn(N).

**Σύμφωνα με τα παραπάνω είναι εύκολο να παράγουμε μερικά παραδείγματα. Χρησιμοποιείτε το πρόγραμμα για την παραγωγή. Τα παρακάτω αποτελέσματα είναι απλή μεταφορά του διαλόγου στην prolog.**



**Επεξήγηση αποτελεσμάτων για το ?- vp(1,A,B,C[])., ?- vp(2,A,B,C[])., ?- vp(3,A,B,C[]). αντίστοιχα:**

* To likes chocolates μπορεί να μην είναι σωστό συντακτικά, αλλά περιέχει πληροφορία. Η λεπτομέρεια που χρειάζεται εδώ είναι ο διαχωρισμός των emotion verbs από τα υπόλοιπα, καθώς πρέπει να ακολουθούνται μόνο από πληθυντικό ή proper nouns στον ενικό.
* Το is social προσδιορίζει προφανώς ένα όνομα και είναι σωστή φράση.
* Το likes nick προσδιορίζει προφανώς ένα όνομα και είναι σωστή φράση.

**2)2) Τύποι γνώσης:**

Στην εισαγωγή αναφέρονται έξι (6) τύποι γνώσης στους οποίους βασίζεται το πρόγραμμα. Αυτοί αντιστοιχούν άμεσα στους έξι τύπους προτάσεων. Σε αυτήν την παράγραφο παρουσιάζονται ένας προς ένας οι τύποι γνώσης.

* **Τύπος 1 (υποκείμενο – ρήμα – αντικείμενο):**

Αυτός ο τύπος γνώσης αφορά μια πράξη που πράττει ένα φυσικό πρόσωπο (pn) σε ένα αντικείμενο (n).

**Η ακολουθία φράσεων είναι η εξής:**

**s(1,Sem) 🡪**

pn(N), vp(1,V,N1)🡪

pn(N), v(V,s),n(N)🡪

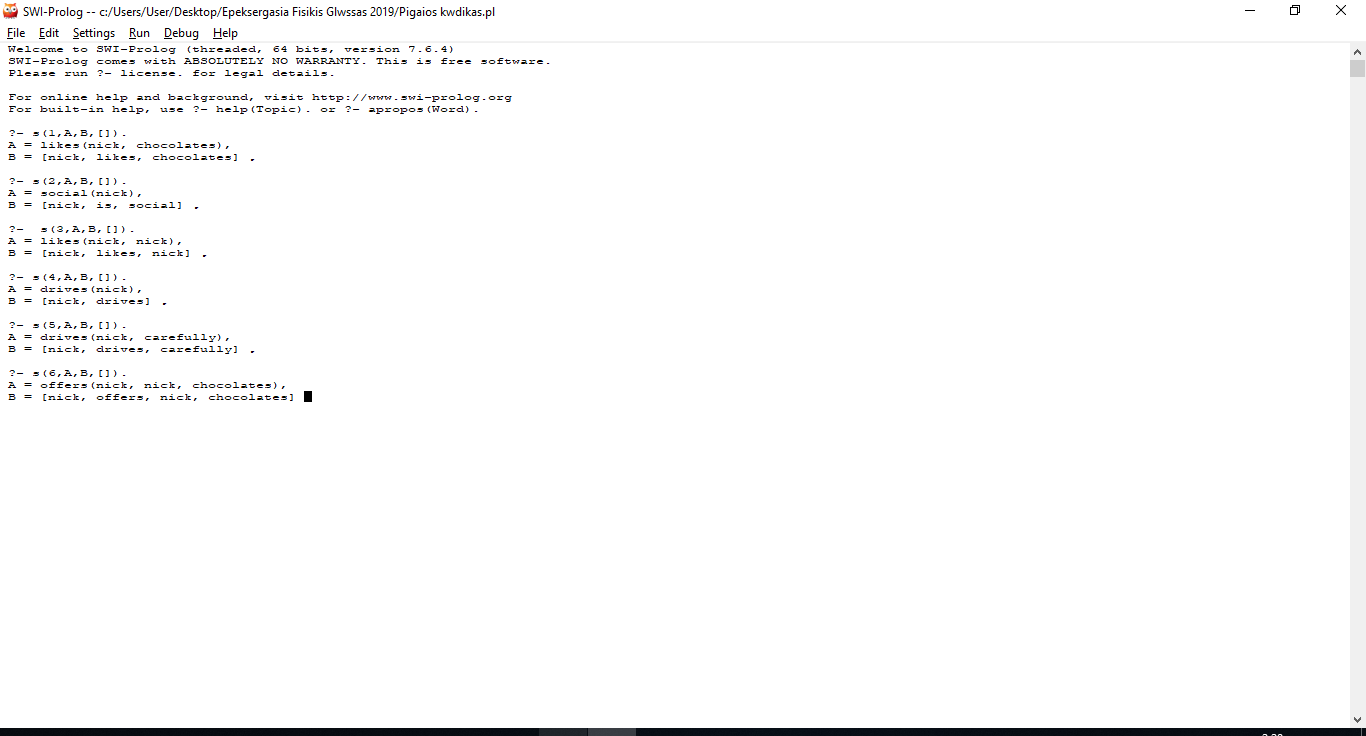
pn(N), v(V,s),n(N)

**Ο τύπος γνώσης που παράγεται είναι:**

**V(N,N1)** ή αλλιώς **ρήμα(υποκείμενο, αντικείμενο)**

Η γνώση περνάει στο Sem, το δεύτερο όρισμα του s.

**Η μετατροπή γίνετε με τον παρακάτω τρόπο:**

**Sem=..[V,N,N1]**

* **Τύπος 2 (υποκείμενο – επίθετο):**

Αυτός ο τύπος γνώσης αφορά στον προσδιορισμό χαρακτηριστικού σε ένα πρόσωπο.

**Η ακολουθία φράσεων είναι η εξής:**

**s(2,Sem) 🡪**

pn(N), vp(2,\_,A)🡪

pn(N), av(is),adj(A)🡪

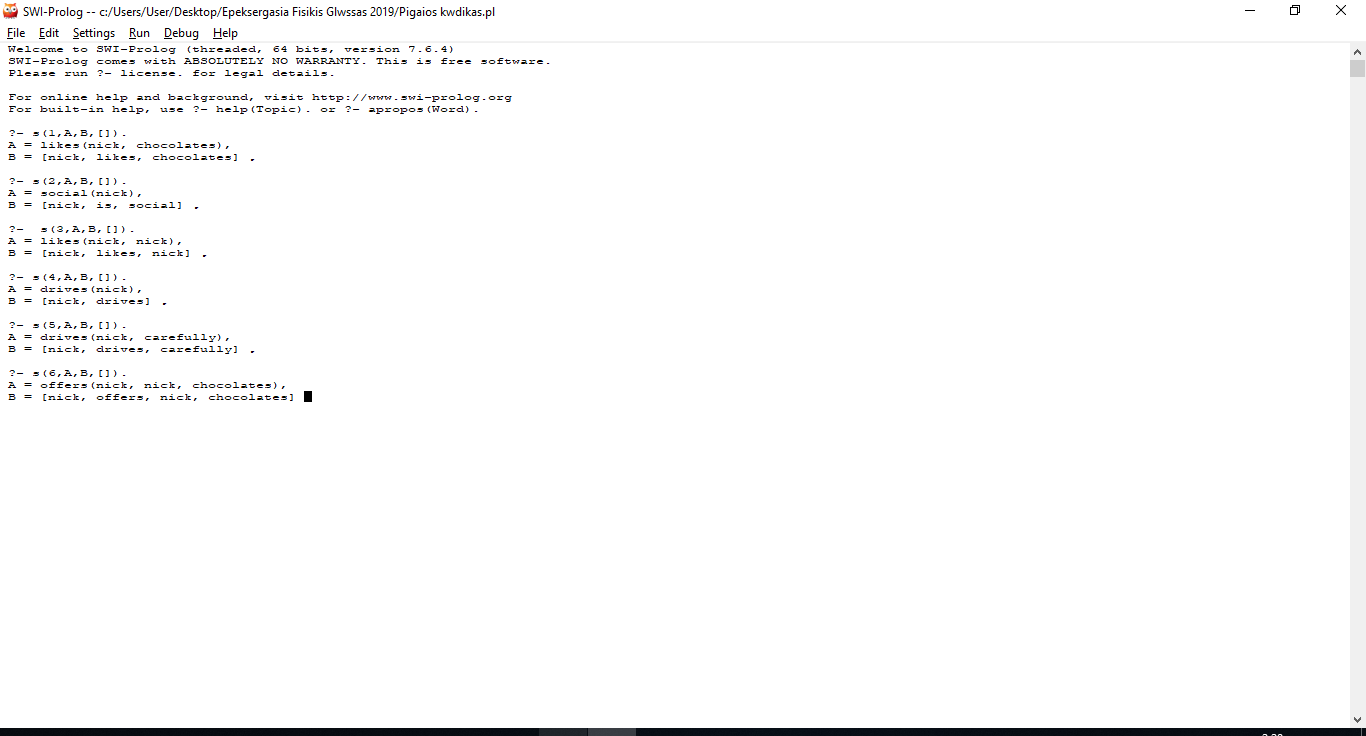
pn(N), av(is),adj(A)

**Ο τύπος γνώσης που παράγεται είναι:**

**A(N**) ή αλλιώς **επίθετο (πρόσωπο)**

**Η μετατροπή γίνετε με τον παρακάτω τρόπο:**

**Sem=..[A,N]**



* **Τύπος 3 (υποκείμενο – ρήμα – αντικείμενο):**

Αυτός ο τύπος γνώσης αφορά μια πράξη που πράττει ένα φυσικό πρόσωπο (pn) σε ένα άλλο.

**Η ακολουθία φράσεων είναι η εξής:**

**s(3,Sem) 🡪**

pn(N), vp(3,V,N1)🡪

pn(N), v(V,s),pn(N)🡪

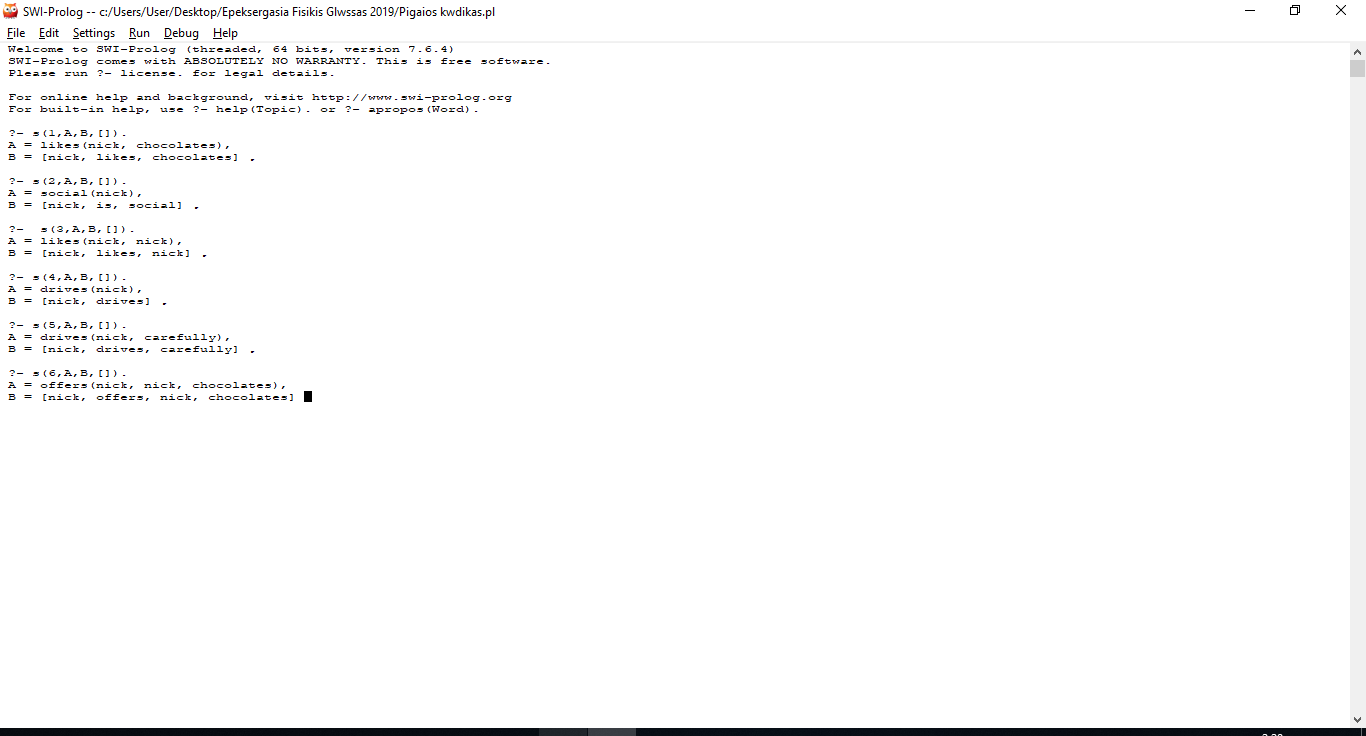
pn(N), v(V,s),pn(N)

**Ο τύπος γνώσης που παράγεται είναι:**

**V(N,N1)** ή αλλιώς **ρήμα(πρόσωπο1,πρόσωπο2)**

**Η μετατροπή γίνετε με τον παρακάτω τρόπο:**

**Sem=..[V,N,N1]**



* **Τύπος 4 (υποκείμενο – ρήμα):**

Αυτός ο τύπος γνώσης αφορά μια πράξη που πράττει ένα φυσικό πρόσωπο (pn) και δεν μεταφέρεται, αφορά δηλαδή το ίδιο.

**Η ακολουθία φράσεων είναι η εξής:**

**s(4,Sem) 🡪**

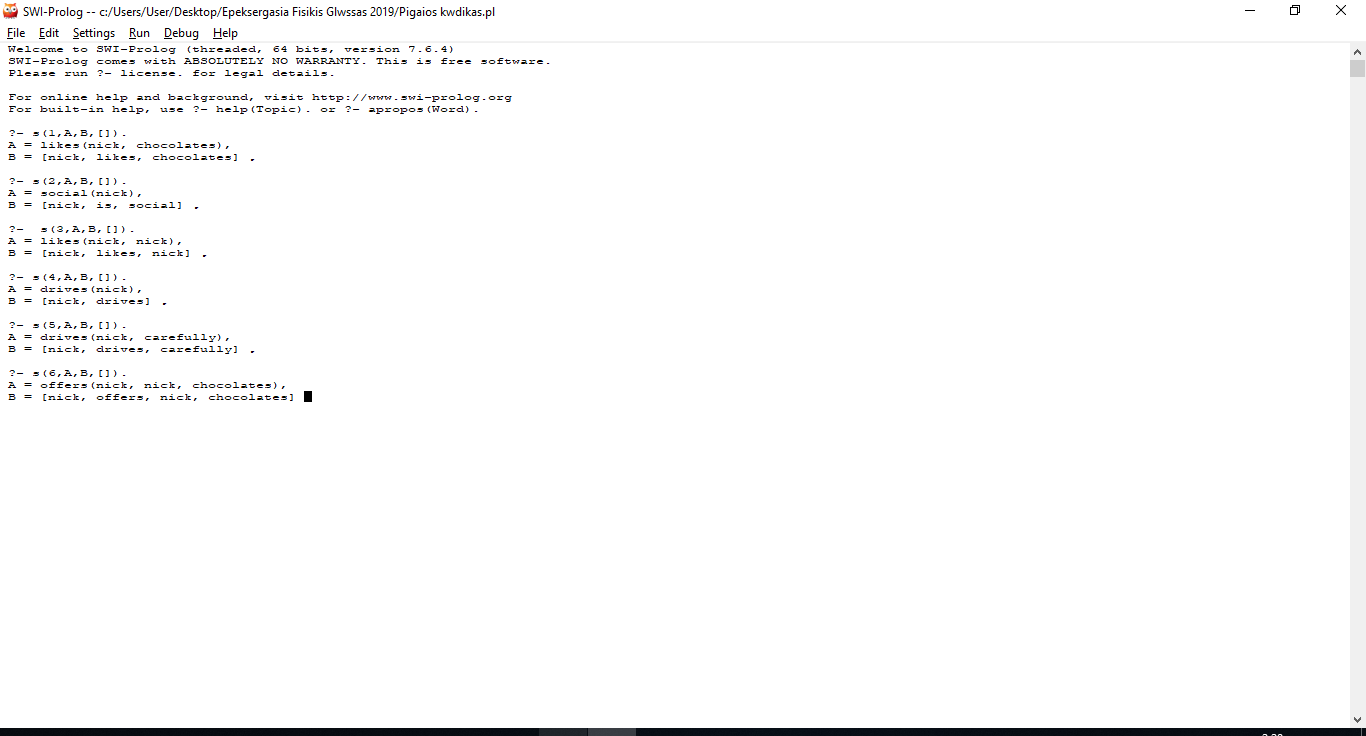
pn(N), iv(V,s)

**Ο τύπος γνώσης που παράγεται είναι:**

**V(N)** ή αλλιώς **ρήμα(υποκείμενο)**

**Η μετατροπή γίνετε με τον παρακάτω τρόπο:**

**Sem=..[V,N]**



* **Τύπος 5 (υποκείμενο – ρήμα – επίρρημα):**

Αυτός ο τύπος γνώσης αφορά μια πράξη που πράττει ένα φυσικό πρόσωπο (pn) και δεν μεταφέρεται, αφορά δηλαδή το ίδιο αλλά ακολουθείτε από ένα όρισμα-επίρρημα.

**Η ακολουθία φράσεων είναι η εξής:**

**s(5,Sem) 🡪**

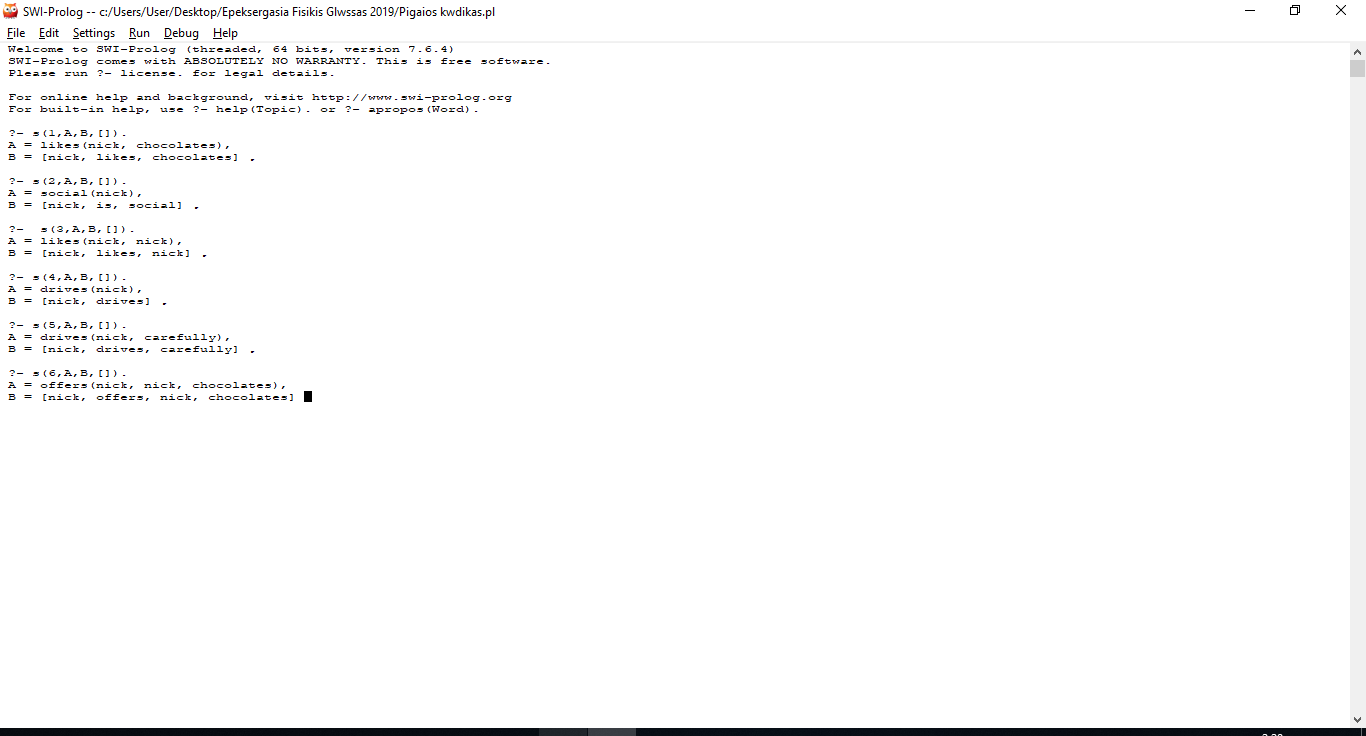
pn(N), iv(V,s),adv(A)

**Ο τύπος γνώσης που παράγεται είναι:**

**V(N,Α)** ή αλλιώς **ρήμα(υποκείμενο, επίρρημα)**

**Η μετατροπή γίνετε με τον παρακάτω τρόπο:**

**Sem=..[V,N,Α]**

****

* **Τύπος 6 (υποκείμενο – ρήμα – άμεσο αντικείμενο – αντικείμενο):**

Αυτός ο τύπος γνώσης αφορά μια πράξη που πράττει ένα φυσικό πρόσωπο (pn) σε ένα άλλο. Η γνώση περιέχει και ένα ακόμη αντικείμενο με το οποίο ενεργεί το πρώτο πρόσωπο στο δεύτερο. Εναλλακτικά η πράξη ορίζεται με το ρήμα και το αντικείμενο (π.χ. ρίχνει πέτρες) και ενεργείτε από το υποκείμενο στο άμεσο αντικείμενο (Ο Πέτρος ρίχνει πέτρες στον Γιάννη). Η ακολουθία φράσεων είναι η εξής:

**s(6,Sem) 🡪**

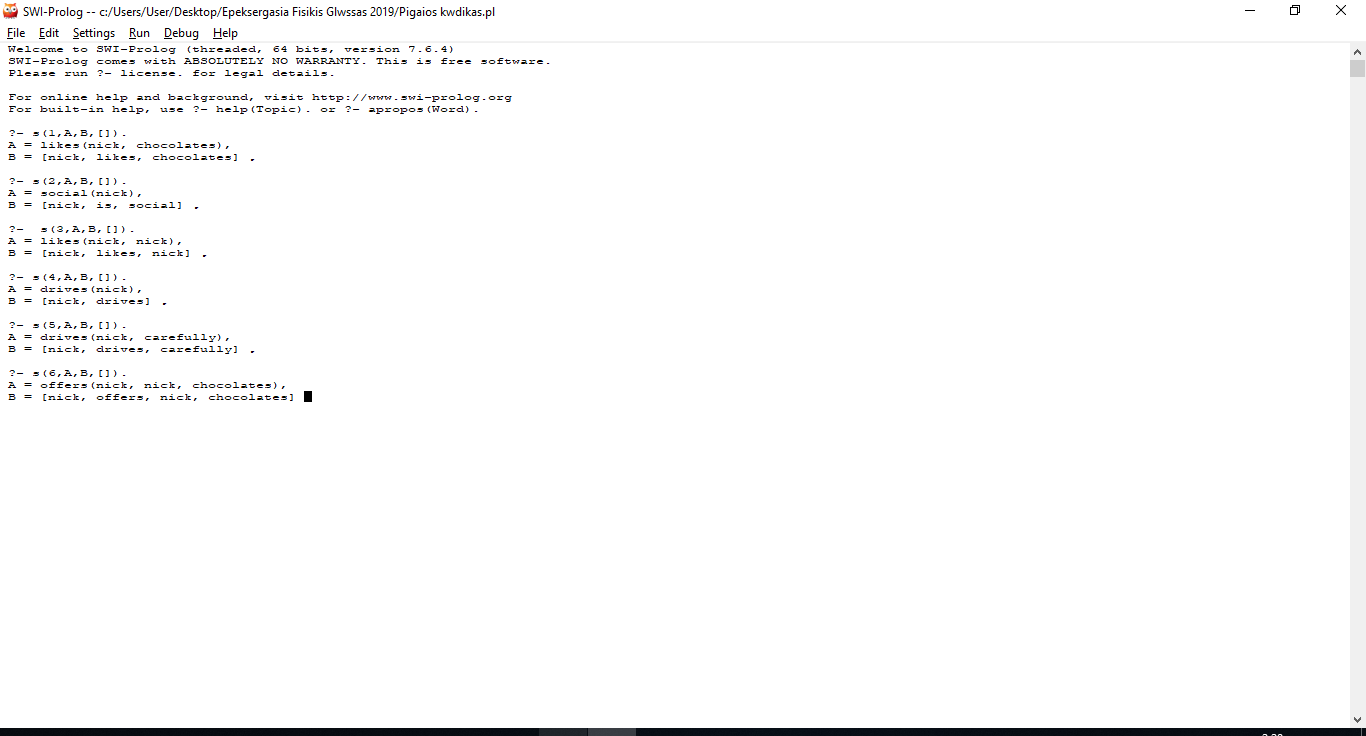
pn(N), tv(V,s),pn(N1),np(N2)

**Ο τύπος γνώσης που παράγεται είναι:**

**V(N,N1,N2)** ή αλλιώς **ρήμα (υποκείμενο, άμεσο αντικείμενο, αντικείμενο)**

**Η μετατροπή γίνετε με τον παρακάτω τρόπο:**

**Sem=..[V,N,N1,N2]**

****

Περιληπτικά, υπάρχουν έξι τύποι γνώσης που αφορούν σε σχέσεις μεταξύ ατόμων, αλλά και προσωπικές πράξεις και χαρακτηριστικά προσώπων. Ο κώδικας που έχει παρουσιαστεί ως τώρα, έχει την δυνατότητα με την παρακάτω εντολή να διαβάσει προτάσεις με τις παραπάνω δομές, αλλά και να αποθηκεύσει την πληροφορία σε μια knowledge base στην οποία μπορούν να τεθούν ερωτήματα.

**Ο κώδικας που χρησιμοποιείτε για την εισαγωγή πληροφορίας, αν αυτή είναι έγκυρη, είναι ο εξής:**

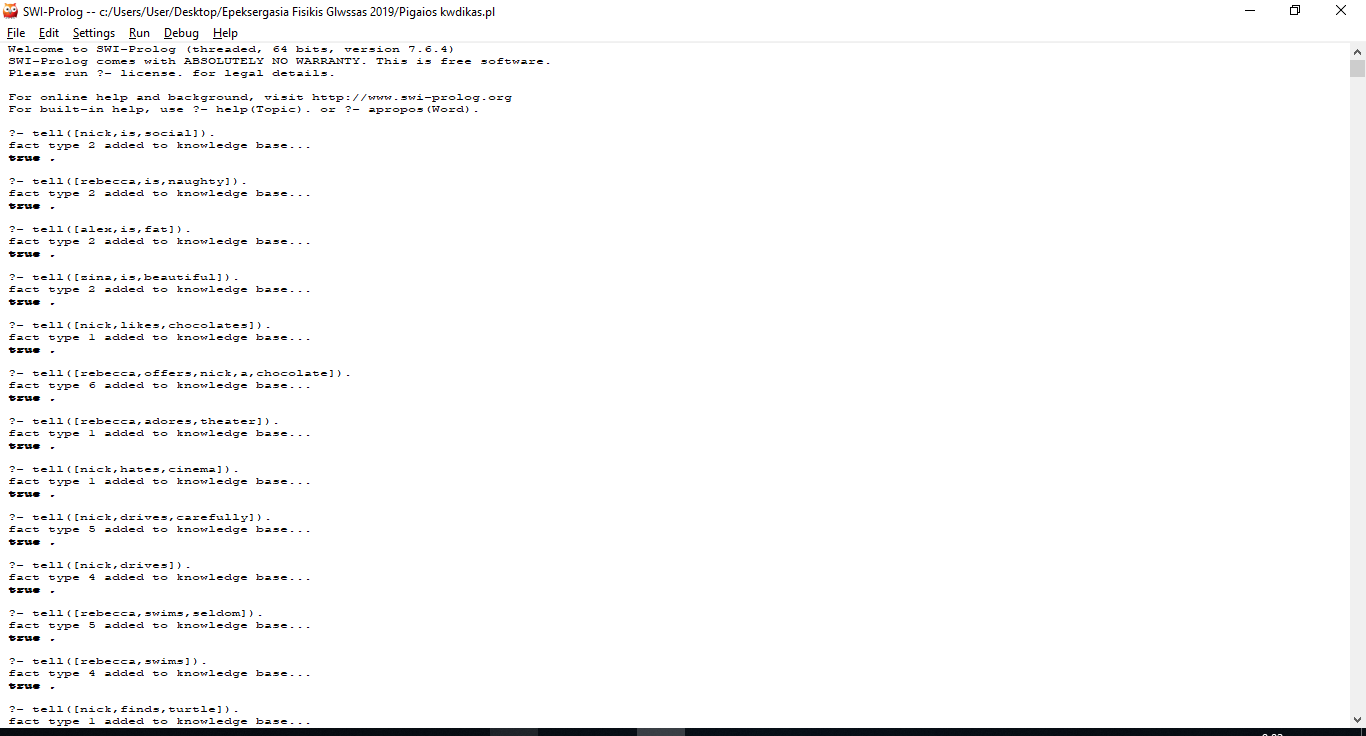
**tell(X):-**

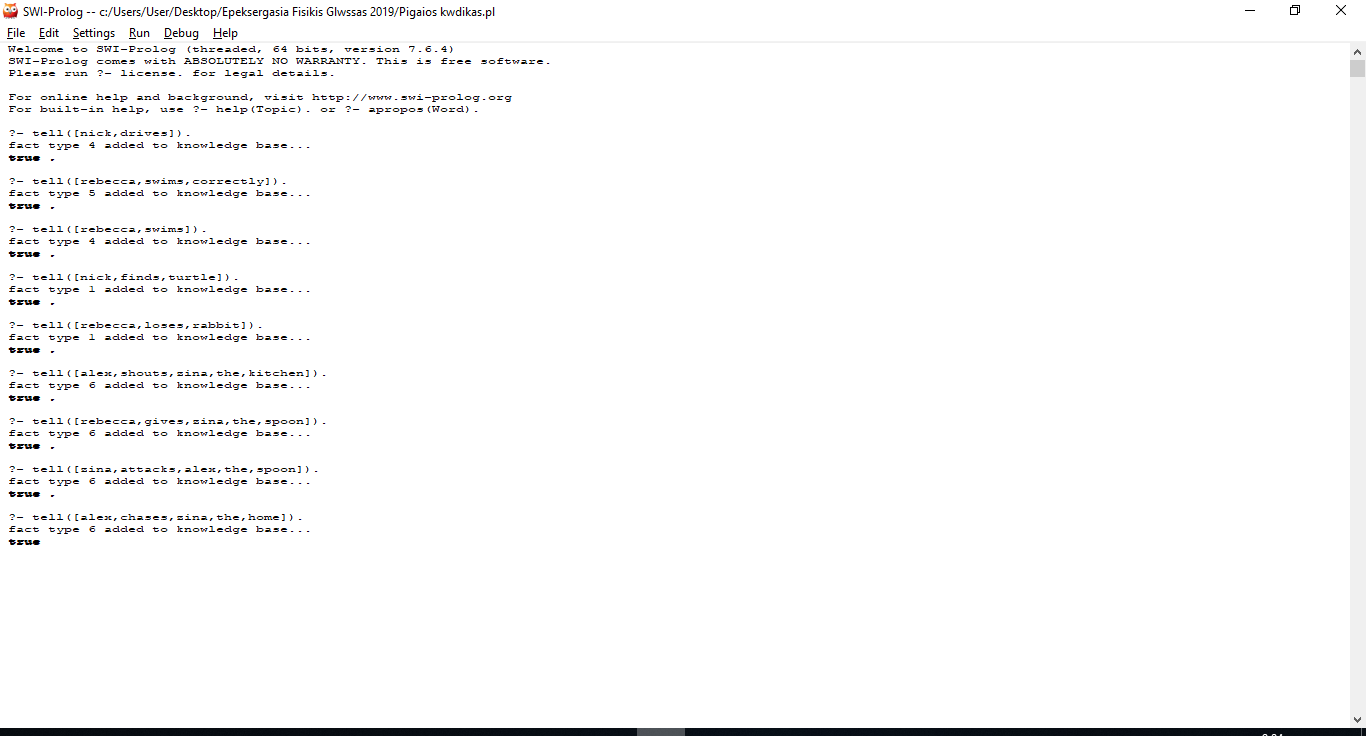
**s(TYPE,Sem,X,[]),assert(Sem),write('fact type '),write(TYPE),write(' added to knowledge base…'),nl.**

**Ο χρήστης εισάγει μόνο το κείμενο (Χ) και αυτό δοκιμάζεται από την Prolog σε ένα ερώτημα της μορφής:**

**s(TYPE,Sem,X,[])**

Αν υπάρχει συνδυασμός TYPE σημαίνει ότι η φυσική γλώσσα είναι έγκυρη με βάση κάποιον από τους τύπους των προτάσεων. Ο τύπος περνάει στο TYPE και η λογική σημασία του περνάει δομημένη στο Sem. Εν συνεχεία η prolog εκτελεί το **assert(Sem)** και περνάει την γνώση στο πρόγραμμα. Οι εντολές που ακολουθούν απλά δηλώνουν τον τύπο πρότασης που διαβάστηκε και δίνουν μια κενή γραμμή.





**2)3) Τύποι ερωτήσεων:**

Ενώ οι τύποι γνώσης είναι περιορισμένοι, οι τύποι ερωτήσεων που εξετάζονται είναι όλοι οι πιθανοί πάνω στην βάση γνώσης που έχουμε. Είναι δε δυνατόν να χωριστούν σε δύο υποκατηγορίες. Στην εξακρίβωση της ορθότητας μιας πληροφορίας και στην αναζήτηση πληροφορίας.

**2)3)α) Εξακρίβωση πληροφορίας:**

Στα σχόλια του προγράμματος ονομάζονται *yes/no queries* καθώς απαιτούν μια καταφατική ή μια αρνητική απάντηση. Οι ερωτήσεις παράγουν μια σημασία / γνώση με παρόμοιο τρόπο, όπως γίνετε και στις προτάσεις, και στην συνέχεια αντί να προστεθεί αυτή η γνώση, εξετάζεται αν υπάρχει στην βάση γνώσης.

**Όπως και στις προτάσεις, έξι τύποι ορίζουν τις έγκυρες ερωτήσεις:**

1. **q(1,tf,Sem)** 🡪 **av(does),pn(N),v(V,q),n(N1), {Sem=..[V,N,N1]}.**

Τίθεται το ερώτημα εάν ένα πρόσωπο ενεργεί σε ένα αντικείμενο.

**Ο τύπος ερώτησης σε φυσική γλωσσά είναι:**

1. Does Rebecca adore theater?
2. Does Nick hate cinema?
3. **q(2,tf,Sem)** 🡪 **av(is),pn(N),adj(A), {Sem=..[A,N]}.**

Τίθεται το ερώτημα εάν ένα χαρακτηριστικό ταιριάζει σε ένα πρόσωπο

**Ο τύπος ερώτησης σε φυσική γλωσσά είναι:**

1. Is Nick social?
2. Is Rebecca naughty?
3. Is Alex fat?
4. Is Zina beautiful?
5. **q(3,tf,Sem)** 🡪 **av(does),pn(N),v(V,q),n(N1),{Sem=..[V,N,N1]}.**

Τίθεται το ερώτημα εάν ένα πρόσωπο ενεργεί σε ένα αντικείμενο.

**Ο τύπος ερώτησης σε φυσική γλωσσά είναι:**

1. Does Rebecca adore theater?
2. Does Nick hate cinema?
3. **q(4,tf,Sem)** 🡪 **av(is),pn(N),iv(V,q),{Sem=..[V,N]}.**

Ερωτάτε εάν ένα πρόσωπο ενεργεί κατά κάποιον τρόπο.

**Ο τύπος ερώτησης σε φυσική γλωσσά είναι:**

1. Is Nick driving?
2. Is Rebecca swimming?
3. **q(5,tf,Sem) --> av(is),pn(N),iv(V,q),adv(A),{Sem=..[V,N,A]}.**

Ερωτάτε εάν ένα πρόσωπο πράττει μια ενέργεια κατά συγκεκριμένο τρόπο.

**Ο τύπος ερώτησης σε φυσική γλωσσά είναι:**

1. Is Nick driving carefully?
2. Is Rebecca swimming correctly?
3. **q(6,tf,Sem) --> av(does),pn(N), tv(V,q),pn(N1),np(N2), {Sem=..[V,N,N1,N2]}.**

Ερευνάτε μια ενέργεια ενός ατόμου σε ένα άλλο. Γίνετε η χρήση δυο αντικειμένων όπως περιγράφεται στον έκτο τύπο γνώσης.

**Ο τύπος ερώτησης σε φυσική γλωσσά είναι:**

1. Does Rebecca offer Nick, a chocolate?

ii) Does Rebecca give Zina, the spoon?

1. Does Zina attack Alex, with the spoon?

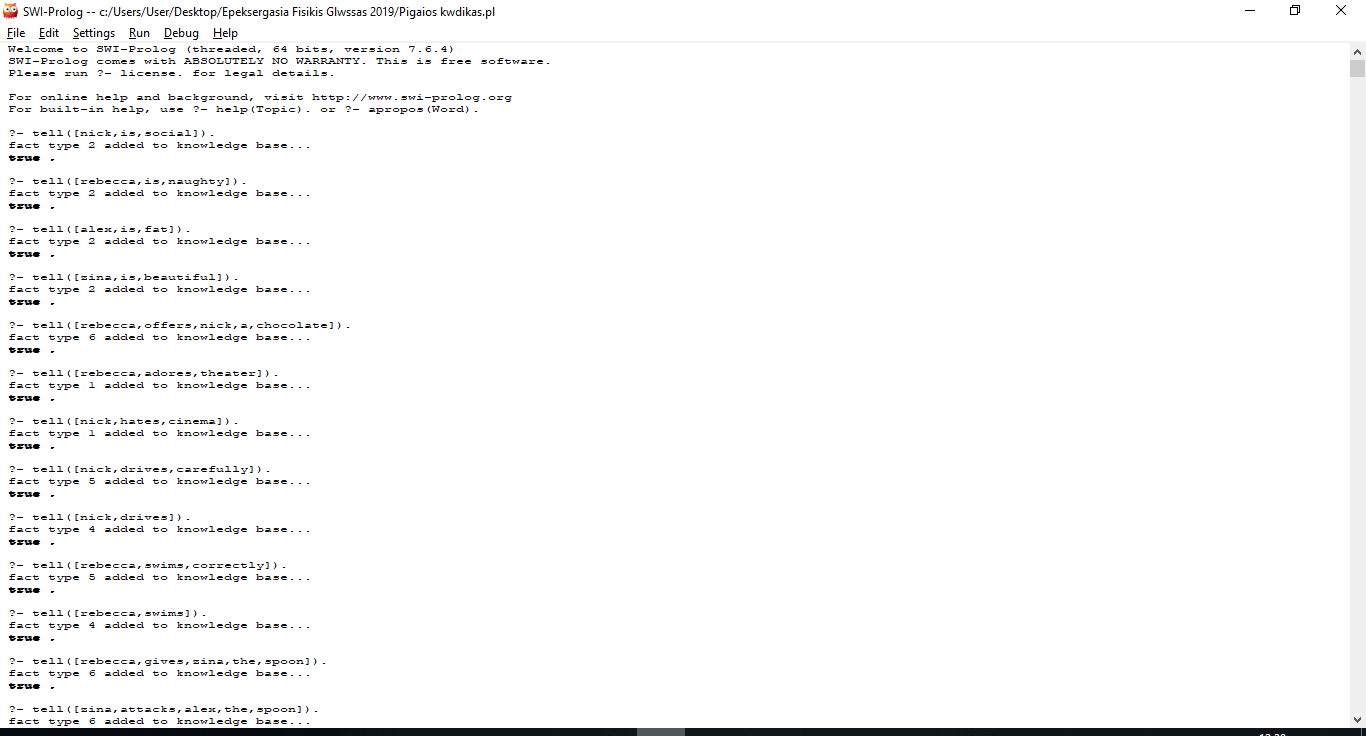
**Εδώ ολοκληρώνονται οι τύποι ερωτήσεων yes/no.**

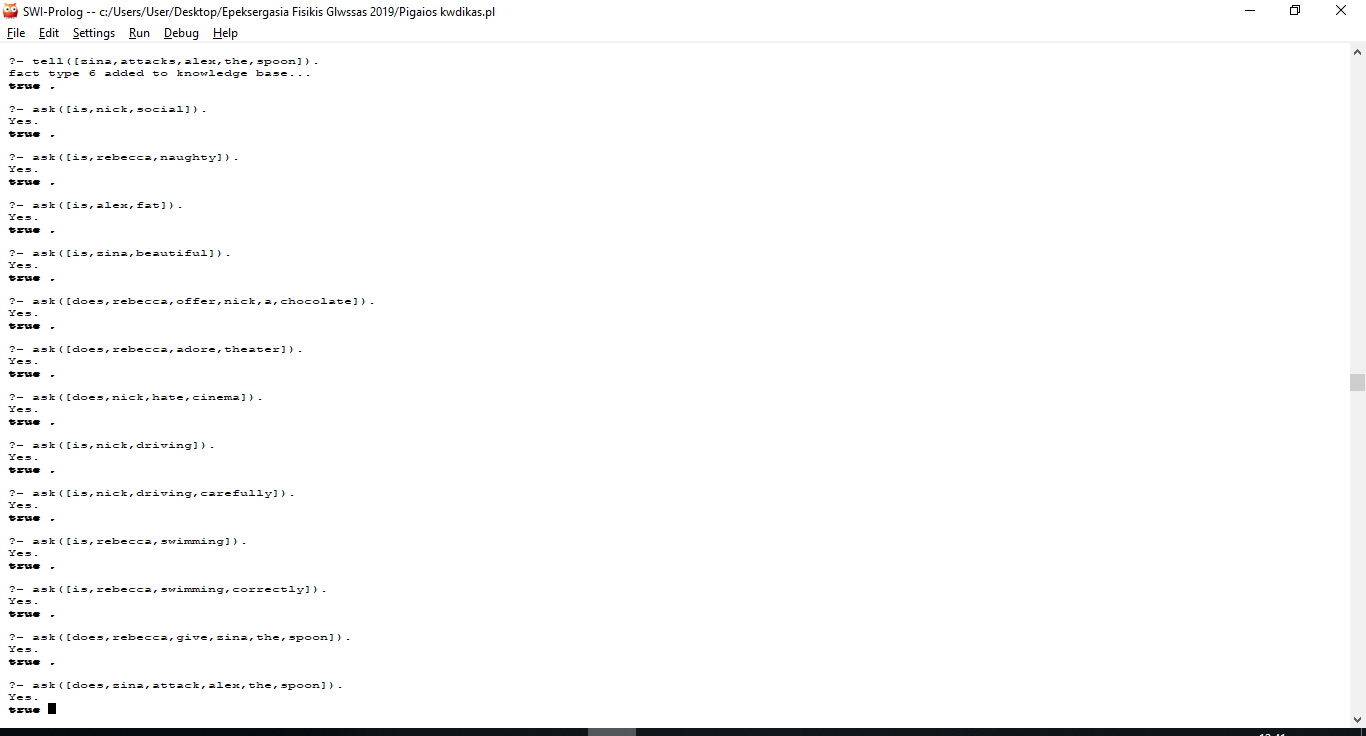
**Η συνάρτηση που χρησιμοποιείτε για να τεθούν οι ερωτήσεις είναι η εξής:**

**ask(X):- q(\_,tf,Sem,X,[]),Sem,write('Yes.').**

Εξετάζει την φυσική γλώσσα σε έναν τύπο ερώτησης **tf** και αν είναι επιτυχής γυρίζει το **Sem**. Στην συνεχεία εξετάζεται αυτό και αν ισχύει, η prolog τυπώνει **Yes**.

**Πριν την κάθε ερώτηση, γίνετε καταχώρηση στην βάση γνώσης με την συνάρτηση tell.**

****

****

**2)3)β) Ερωτήσεις γεγονότων:**

Στα σχόλια του προγράμματος ονομάζονται *fact queries* καθώς απαιτούν ένα γεγονός σαν απάντηση. Η λογική με την οποία παρήχθησαν είναι απλή. Ένα γεγονός, μια καταχώρηση στην βάση γνώσης αποτελείτε από δύο έως τέσσερα μέλη. Σε κάθε ερώτηση μπορεί να λείπει ένα μέλος της σχέσης και να ελέγχετε εάν υπάρχει καταχώρηση που ταιριάζει. Οι ερωτήσεις γεγονότων χωρίζονται και αυτές σε έξι κατηγορίες που αντιστοιχούν στους τύπους γνώσης.

1. **Τύπος 1 (υποκείμενο – ρήμα – αντικείμενο):**

Στον τύπο αυτό έχει νόημα να εξεταστεί το αντικείμενο και το υποκείμενο.

**Στις προτάσεις «Nick likes chocolates.», «Rebecca adores theater.», «Nick hates cinema.», «Nick finds a turtle.», «Rebecca loses a rabbit.» μπορούν να τεθούν τα παρακάτω ερωτήματα:**

1. Who likes chocolates?
2. What does Nick like?
3. Who adores theater?
4. What does Rebecca adore?
5. Who hates cinema?
6. What does Nick hate?
7. Who finds turtle?
8. Who loses rabbit?

**Τα ερωτήματα ορίζονται από τους τύπους:**

**q(1,fact,F) --> [who],v(V,s),n(N1), {Sem=..[V,F,N1],Sem}.**

**q(1,fact,F) --> [what],av(does),pn(N),v(V,q), {Sem=..[V,N,F],Sem}.**

Στο **F** περνάει μια πιθανή τιμή για τον άγνωστο της εξίσωσης.

1. **Τύπος 2 (υποκείμενο – επίθετο):**

Στον τύπο αυτό έχει νόημα να εξεταστεί το υποκείμενο.

**Στις προτάσεις «Nick is social.», «Rebecca is naughty.», «Alex is fat.»,**

**«Zina is beautiful.» μπορούν να τεθούν τα παρακάτω ερωτήματα:**

1. Who is social?
2. Who is naughty?
3. Who is fat?
4. Who is beautiful?

**Τα ερωτήματα ορίζονται από τον τύπο:**

**q(2,fact,F) --> [who],av(is),adj(A), {Sem=..[A,F],Sem}.**

1. **Τύπος 3 (υποκείμενο – ρήμα – αντικείμενο):**

Ο τύπος 3 είναι όμοιος με τον τύπο 1.

**Στις προτάσεις «Nick likes chocolates.», «Rebecca adores theater.», «Nick hates cinema.», «Nick finds a turtle.», «Rebecca loses a rabbit.» μπορούν να τεθούν τα παρακάτω ερωτήματα:**

1. Who likes chocolates?
2. What does Nick like?
3. Who adores theater?
4. What does Rebecca adore?
5. Who hates cinema?
6. What does Nick hate?
7. Who finds turtle?
8. Who loses rabbit?

**Τα ερωτήματα ορίζονται από τους τύπους:**

**q(3,fact,F) --> [who],vp(3,V,N1), {Sem=..[V,F,N1],Sem}.**

**q(3,fact,F) --> [who],av(does),pn(N),v(V,q), {Sem=..[V,N,F],Sem}.**

1. **Τύπος 4 (υποκείμενο – ρήμα):**

Στον τύπο αυτό έχει νόημα να εξεταστεί το υποκείμενο.

**Στις προτάσεις «Nick is driving.», «Rebecca is swimming» μπορούν να τεθούν τα παρακάτω ερωτήματα:**

1. Who is driving?
2. Who is swimming?

**Τα ερωτήματα ορίζονται από τον τύπο:**

**q(4,fact,F) --> [who],av(is),iv(V,q), {Sem=..[V,F],Sem}.**

1. **Τύπος 5 (υποκείμενο – ρήμα – επίρρημα):**

Στον τύπο αυτό έχει νόημα να εξεταστεί το υποκείμενο και το επίρρημα.

**Στις προτάσεις «Nick is driving carefully.», «Rebecca is swimming correctly.» μπορούν να τεθούν τα παρακάτω ερωτήματα:**

1. Who drives carefully?

ii) How does Nick drives?

1. How is Nick driving?
2. Who swims correctly?
3. How does Rebecca swims?
4. How is Rebecca swimming?

**Τα ερωτήματα ορίζονται από τους τύπους:**

**q(5,fact,F) --> [how],av(does),pn(N),iv(V,s), {Sem=..[V,N,F],Sem}.**

**q(5,fact,F) --> [how],av(is),pn(N),iv(V,q), {Sem=..[V,N,F],Sem}.**

**q(5,fact,F) --> [who],iv(V,s),adv(A), {Sem=..[V,F,A],Sem}.**

1. **Τύπος 6 (υποκείμενο – ρήμα – άμεσο αντικείμενο – αντικείμενο):**

Στον τύπο αυτό έχει νόημα να εξεταστεί το υποκείμενο και τα αντικείμενα.

**Στις προτάσεις «Rebecca is offering Nick a chocolate.», «Alex is shouting Zina in the kitchen.», «Rebecca is giving Zina the spoon.», «Zina is attacking Alex with the spoon.», «Alex is chasing Zina in the home.» μπορούν να τεθούν τα παρακάτω ερωτήματα:**

1. Who offers Nick a chocolate?
2. Who is Rebecca offering a chocolate to?
3. What is Rebecca offering to Nick?
4. Who shouts Zina in the kitchen?
5. Who is Alex shouting in the kitchen to?
6. Who gives Zina the spoon?
7. Who is Rebecca giving the spoon to?
8. What is Rebecca giving to Zina?
9. Who attacks Alex with the spoon?
10. Who is Zina attacking with the spoon to?
11. What is Zina attacking to Alex?
12. Who chases Zina in the home?
13. Who is Alex chasing in the home to?

**Τα ερωτήματα ορίζονται από τους τύπους:**

**q(6,fact,F) --> [who],tv(V,s),pn(N1),np(N2), {Sem=..[V,F,N1,N2],Sem}.**

**q(6,fact,F) --> [who],av(is),pn(N),tv(V,q2),np(N2),[to], {Sem=..[V,N,F,N2] ,Sem}.**

**q(6,fact,F) --> [what],[is],pn(N),tv(V,q2),[to],pn(N1), {Sem=..[V,N,N1,F] ,Sem}.**

Εδώ ολοκληρώνονται οι τύποι ερωτημάτων γεγονότων. Όπως στις προτάσεις και στον άλλο τύπο ερωτημάτων, χρησιμοποιείτε συνάρτηση για την εξαγωγή απάντησης. Παρακάτω η διατύπωση της.

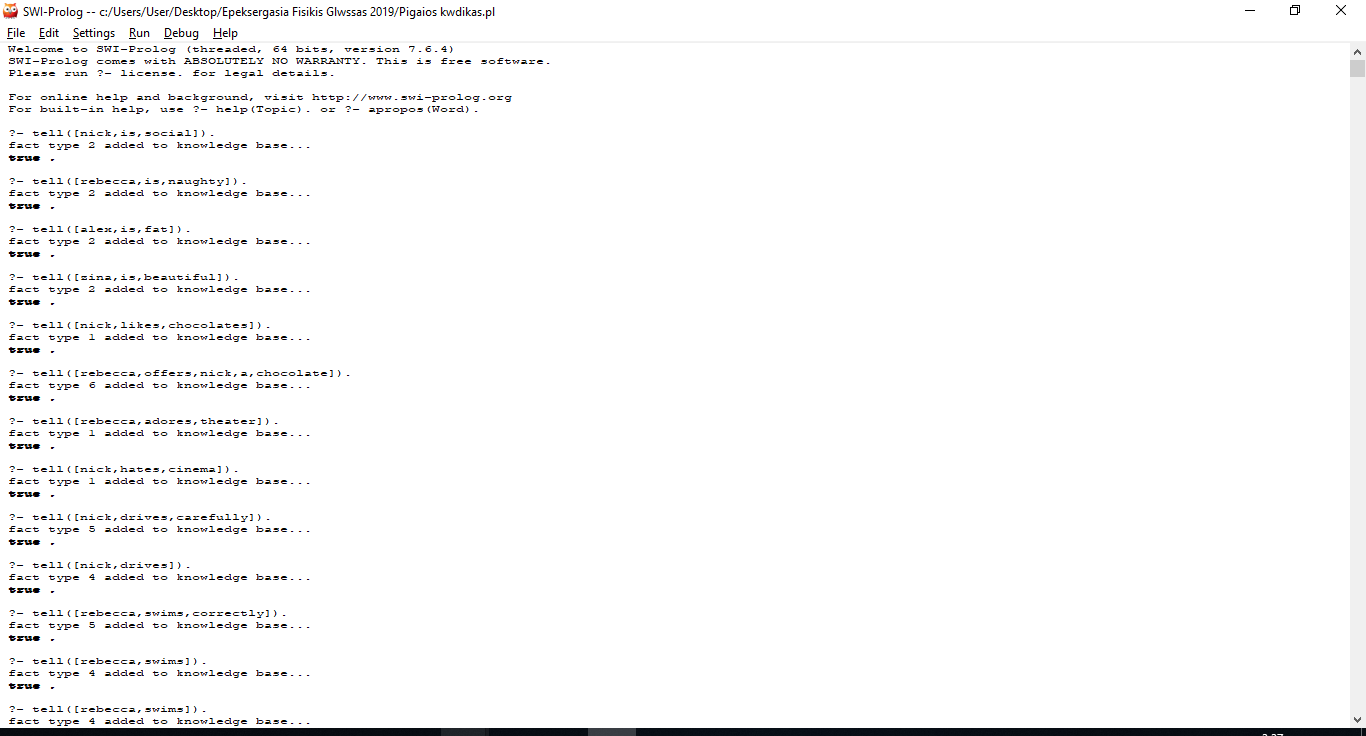
**ask(X):- q(\_, fact, Fact, X, []), write(Fact).**

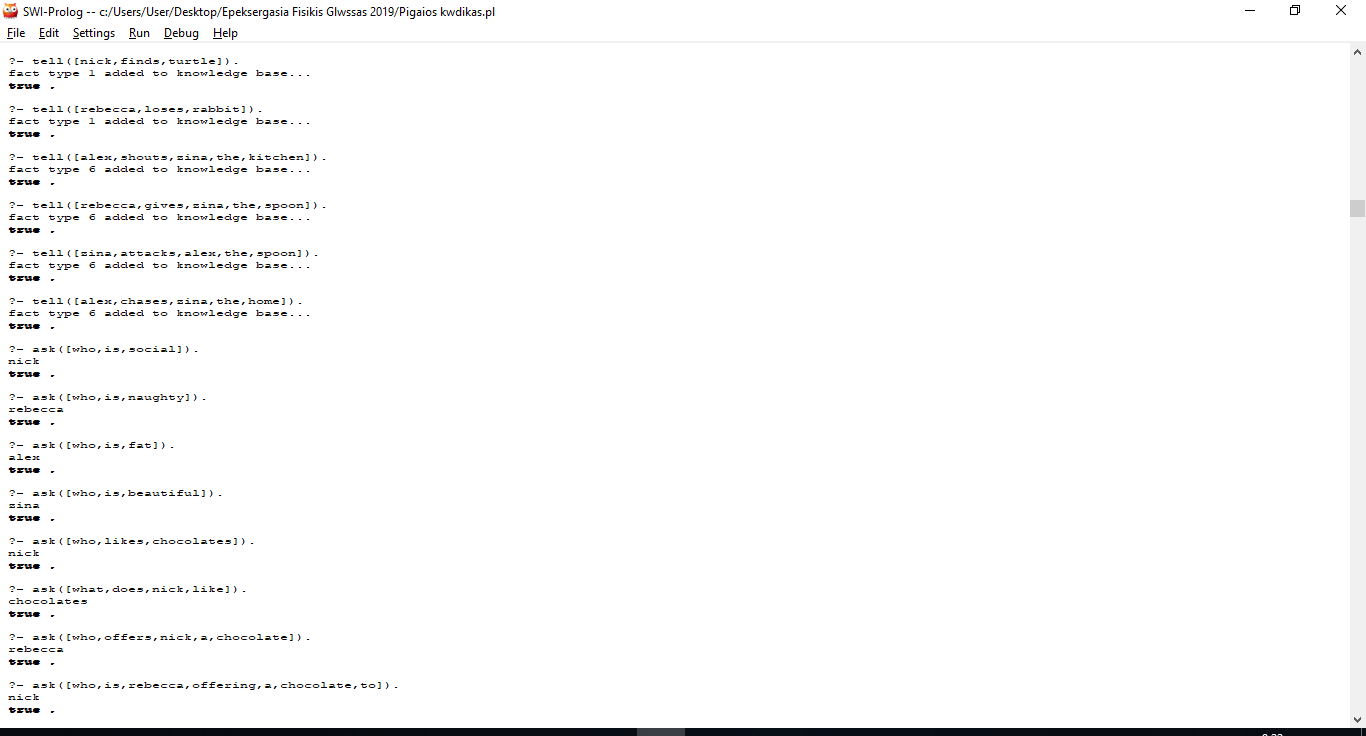
Όπως αναφέρθηκε παραπάνω, το τρίτο όρισμα του q αποτελεί τον άγνωστο.

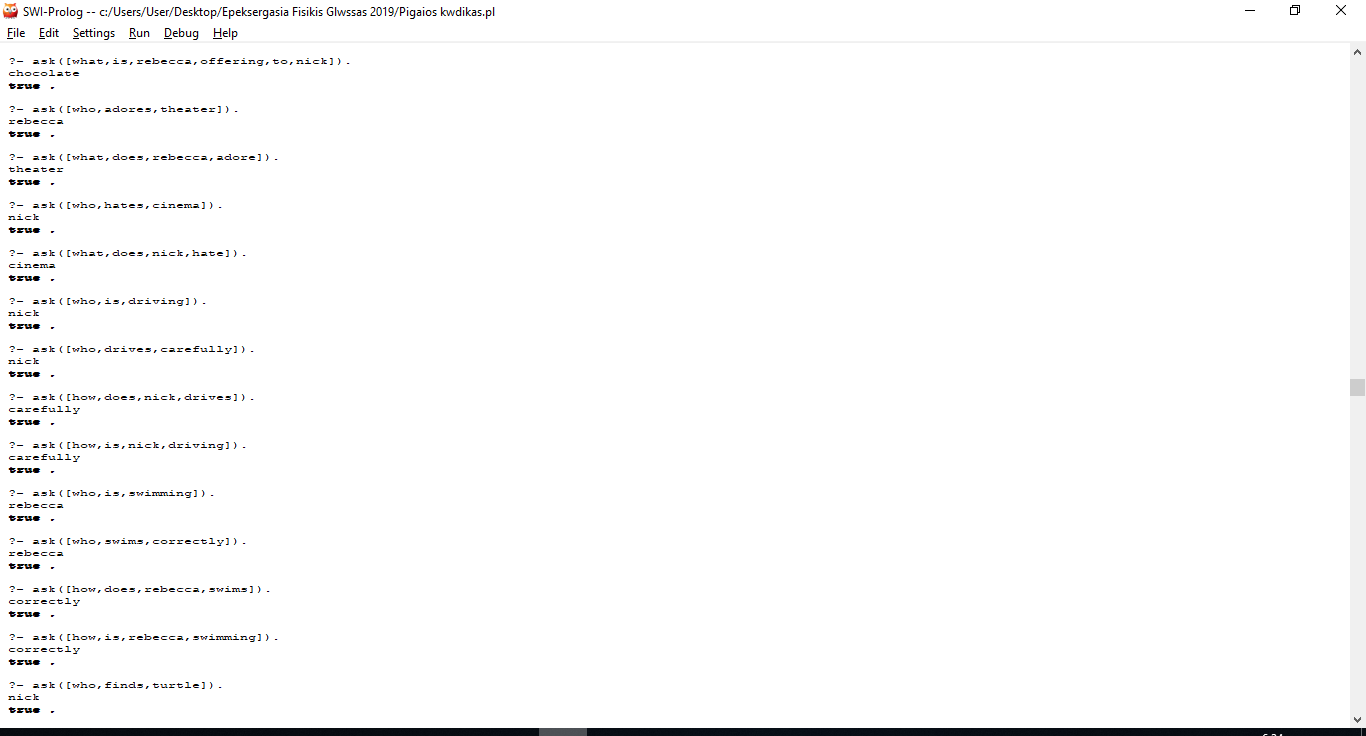
**Ωστόσο δίνετε η πλήρης διαδικασία:**

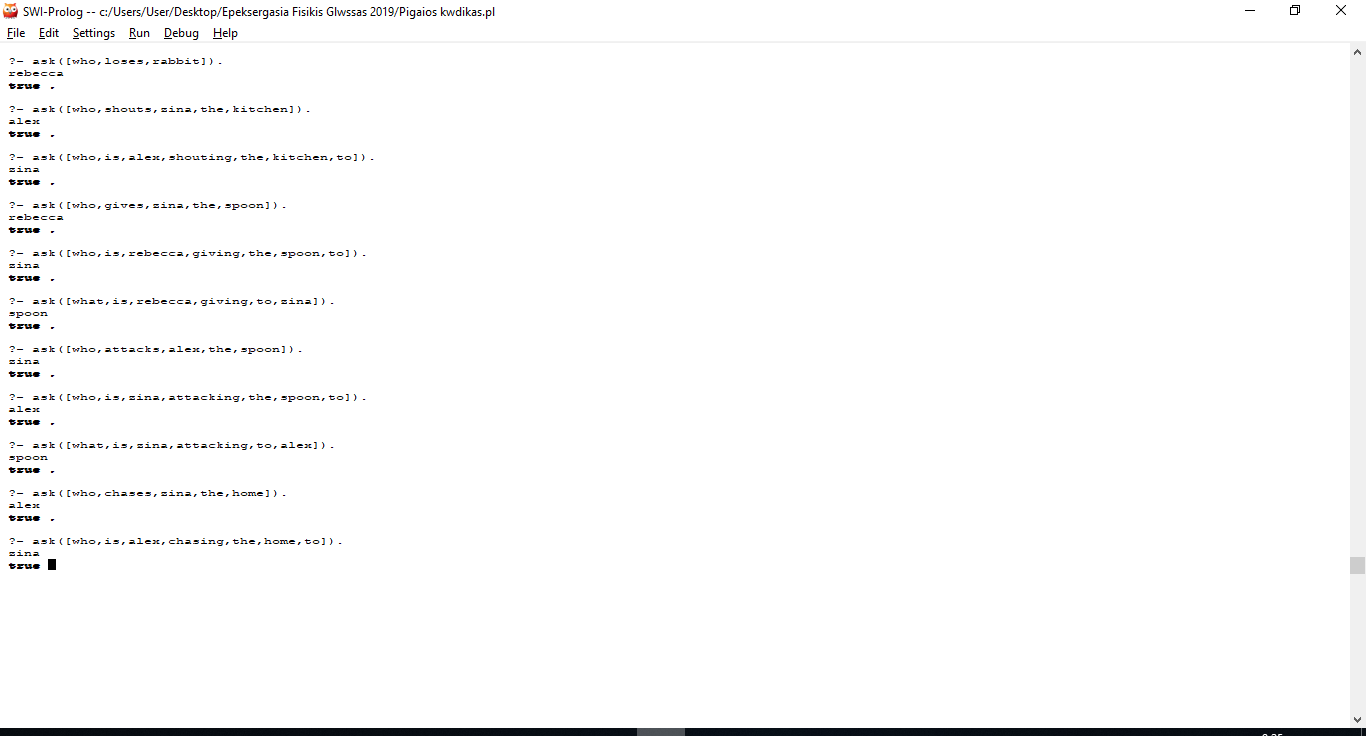
* Το q() τροφοδοτείτε με το Χ, την φυσική γλώσσα.
* Αυτή παρέχει όλους τους όρους που χρειάζονται, και παράγεται η μορφή γνώσης με έναν άγνωστο, το Fact.
* Όταν δημιουργηθεί το Sem, θέτετε ως στόχος γεμίζοντας το F με πιθανή απάντηση, αν αυτή υπάρχει.
* Γυρνώντας στην ask γράφεται η πιθανή απάντηση.

**Πριν την κάθε ερώτηση, γίνετε καταχώρηση στην βάση γνώσης με την συνάρτηση tell.**



****

****

****

## Επεκτάσεις/Εργαλεία

* SWI-Prolog, Version 8.2.1

## Πηγές

* <https://en.wikipedia.org/wiki/Natural_language_processing>
* <https://gepris.dfg.de/gepris/projekt/255344147?language=en>
* <https://gepris.dfg.de/gepris/projekt/238697821?language=en&selectedSubTab=2>
* <https://www.ic.unicamp.br/~meidanis/courses/mc336/problemas-prolog/>
* <https://stackoverflow.com/questions/29380105/prolog-binary-addition>
* <https://gunet2.cs.unipi.gr/modules/document/file.php/TMC113/%ce%a3%ce%b7%ce%bc%ce%b5%ce%b9%cf%8e%cf%83%ce%b5%ce%b9%cf%82/DCG.PDF>
* Βιβλίο ΤΕΧΝΙΚΕΣ ΛΟΓΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ, Η γλώσσα Prolog.

