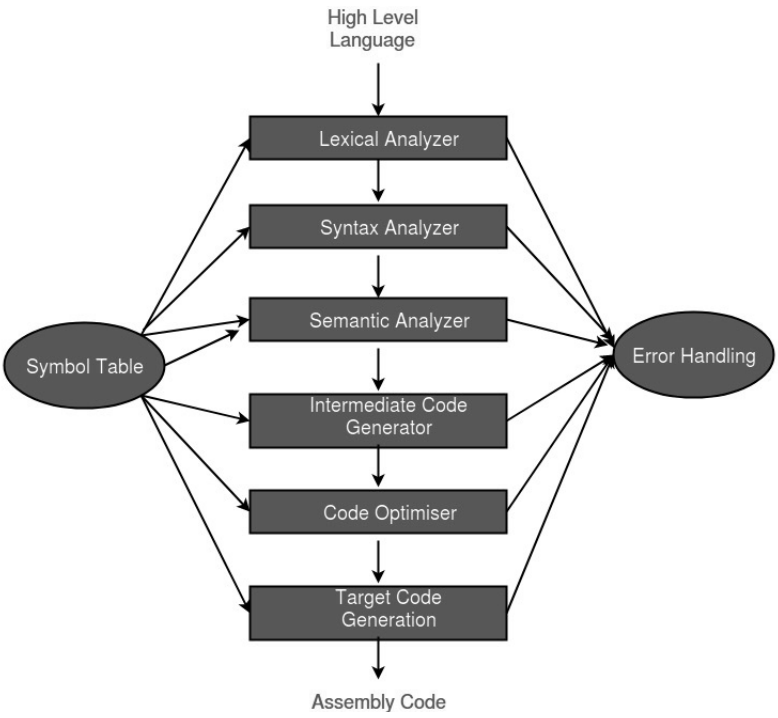
# Προαιρετική Εργασία - Σχεδίαση Γλωσσών Προγραμματισμού

Βασίλειος Παπαστέργιος (ΑΕΜ: 3651)

9 Σεπτεμβρίου, 2021




Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τμήμα Πληροφορικής

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάθημα: Σχεδίαση γλ. προγραμματισμού
Εξάμηνο: 4ο
Διδάσκων: Γεώργιος Μακρής

# Contents

# 1 Εισαγωγή

Το παρόν έγγραφο αποτελεί γραπτή αναφορά για την προαιρετική εργασία του μαθήματος της Σχεδίασης Γλωσσών Προγραμματισμού. Η εργασία είχε ως θέμα την κατασκευή ενός μεταγλωττιστή για μια γλώσσα, η γραμματική και η περιγραφή της οποίας ήταν γνωστές από τις απαιτήσεις της εκφώνησης. Σύμφωνα με τα δεδομένα της τελευταίας, η γλώσσα ονομάζεται `SimpleFortran` και μοιάζει με μια απλουστευμένη έκδοση της γλώσσας προγραμματισμού `Fortran`.

Τα αρχεία πηγαίου κώδικα που παραδίδονται από τον γράφοντα, καθώς και όσα αναγράφονται στο παρόν έγγραφο ικανοποιούν τη βασική εκδοχή της εργασίας, δηλαδή ολοκληρώνουν επιτυχώς **λεκτική, συντακτική και σημασιολογική ανάλυση, παράγοντας πίνακα συμβόλων και εκτυπώνοντας το συντακτικό δέντρο του προγράμματος**.

Η κατασκευή του μεταγλωττιστή πραγματοποιήθηκε από τον γράφοντα με τη βοήθεια του εργαλείου `Antlr`, και πιο συγκεκριμένα την τέταρτη έκδοσή του (`Antlr v4`). Η γλώσσα-στόχος του εργαλείου, η οποία και χρησιμοποιήθηκε από τον γράφοντα για την ανάπτυξη του πηγαίου κώδικα του μεταγλωττιστή είναι η `Java`. Η συγγραφή του πηγαίου κώδικα του μεταγλωττιστή πραγματοποιήθηκε στο περιβάλλον ανάπτυξης `IntelliJ IDEA`, αξιοποιώντας το αντίστοιχο πρόσθετο (plugin) εν ονόματι `Antlr`.

Στις σελίδες που ακολουθούν, παρατίθεται παρουσίαση και τεκμηρίωση του πηγαίου κώδικα που αναπτύχθηκε από τον γράφοντα, καθώς και παραδείγματα και αποτελέσματα της εκτέλεσης του κώδικα.

# 2 Η γλώσσα `SimpleFortran`

Η γλώσσα `SimpleFortran` μοιάζει με τη γλώσσα υψηλού επιπέδου `FORTRAN`. Η `SimpleFortran` είναι δομημένη με σύνθετες εντολές, σε αντίθεση με την κλασική `FORTRAN`, η οποία δε διαθέτει τέτοιες εντολές. Εκτός από τους συνήθεις τύπους της `FORTRAN`, η `SimpleFortran` υποστηρίζει και έναν τύπο λίστας που θυμίζει την αντίστοιχη δομή της `LISP`, καθώς και έναν τύπο ορμαθού χαρακτήρων.

Η `SimpleFortran` δεν έχει αυστηρότητα στη μορφή των προγραμμάτων όπως η κλασική `FORTRAN`, και το κενό παίζει τον ίδιο ρόλο στη μορφή όπως αυτό παίζει στις πιο σύγχρονες γλώσσες. Ακόμα, η `SimpleFortran` επιτρέπει τον ορισμό υποπρογραμμάτων σε ένα - μόνο το εξωτερικό – επίπεδο, όπως και η `FORTRAN`, επιτρέπει κοινές περιοχές, αλλά όχι δηλώσεις ισοδυναμίας μεταβλητών. Τέλος, η `SimpleFortran` χρησιμοποιεί τη στοίβα για κλήση υποπρογραμμάτων, επιτρέποντας έτσι αναδρομή.

Περισσότερες λεπτομέρειες για την περιγραφή της γλώσσας είναι διαθέσιμες στην **επίσημη εκφώνηση της εργασίας**, μαζί με ειδικές αναφορές και παραδείγματα.

# 3 Η υλοποίηση του μεταγλωττιστή

Έχοντας παρουσιάσει τη γενική περιγραφή της γλώσσας `SimpleFortran`, η συγγραφή πηγαίου κώδικα για την υλοποίηση του μεταγλωττιστή έρχεται ως λογικό επόμενο της συλλογιστικής μας. Στην ενότητα αυτή, επομένως, θα αναλύσουμε διεξοδικά την υλοποίηση του μεταγλωττιστή, δίνοντας ιδιαίτερη έμφαση στα μοτίβα ανάπτυξης κώδικα που εφαρμόστηκαν.

Ξεκινώντας, το εργαλείο που χρησιμοποιήθηκε ήταν το `Antlr v4`, με το οποίο δημιουργήθηκε ένα αρχείο γραμματικής, εν ονόματι `SimpleFortran2.g4`. Το αρχείο αυτό, περιέχει συγκεντρωμένους τους κανόνες τόσο για τον λεκτικό (lexer), όσο και για τον συντακτικό αναλυτή (parser). Πιο συγκεκριμένα, οι κανόνες για τη συντακτική ανάλυση βρίσκονται στο πάνω μέρος του αρχείου κατάληξης `.g4` και έχουν ονόματα που αναγράφονται με **πεζά** γράμματα. Ακριβώς κάτω από τους κανόνες αυτούς, δηλαδή στο κάτω μέρος του αρχείου, βρίσκονται οι κανόνες και οι λεκτικές μονάδες που αναγνωρίζονται από το λεξικό αναλυτή της γλώσσας `SimpleFortran`.

Το αρχείο αυτό της γραμματικής (κατάληξης `.g4`) δίνεται ως είσοδος στο εργαλείο `Antlr v4`. Πρόκειται για ένα εργαλείο το οποίο, με βάση το αρχείο γραμματικής, μπορεί να παράγει λεκτικό και συντακτικό αναλυτή σε μια πληθώρα γλωσσών προγραμματισμού. Μερικές από αυτές είναι ενδεικτικά οι γλώσσες `Java`, `C#`, `C++`, `JavaScript`, `Perl`, `Python`. Στο πλαίσιο της παρούσης εργασίας, επιλέχθηκε η γλώσσα `Java`, στην οποία το εργαλείο παρήγαγε αυτόματα έναν λεκτικό και έναν συντακτικό αναλυτή για τη γλώσσα ενδιαφέροντος, την `SimpleFortran`.

Εκτός από τους δύο αναλυτές, ωστόσο, το εργαλείο προσφέρει τη δυνατότητα για αυτόματη παραγωγή και μιας επιπλέον κλάσης, η οποία διαδραματίζει ιδιαίτερα σημαντικό ρόλο στην ανάπτυξη του μεταγλωττιστή. Η κλάση αυτή ονομάζεται «επισκέπτης» (visitor) και προσφέρει τη δυνατότητα στον προγραμματιστή να προσθέσει με γρήγορο και εύκολο τρόπο τμήματα κώδικα της επιλογής του, τα οποία μπορούν να εκτελεστούν κατά τη διάρκεια του parsing.

Ειδικότερα, η κλάση–επισκέπτης διαθέτει μία μέθοδο για κάθε έναν κανόνα της γραμματικής, από εκείνους που βρίσκονται στο πάνω τμήμα του αρχείου της γραμμτικής και αναφέρονται στον συντακτικό αναλυτή. Κάθε μία από αυτές τις μεθόδους καλείται αυτόματα από το εργαλείο, όταν ο συντακτικός αναλυτής αντιστοιχήσει μια συμβολοσειρά της στοίβας με τον αντίστοιχο κανόνα της γραμματικής στον οποίο αναφέρονται. Από προεπιλογή, οι μέθοδοι αυτές προσφέρονται κενές (χωρίς τμήμα κώδικα που εκτελείται).

Έγκειται, επομένως, στην επιθυμία και την ικανότητα του προγραμματιστή να προσθέσει τμήματα κώδικα τα οποία θα εκτελούνται κατά την επίσκεψη σε κάθε έναν κόμβο του αφηρημένου συντακτικού δέντρου (abstract syntax tree – AST), **κατά τη διάρκεια κατασκευής του από τον συντακτικό αναλυτή**. Ο τρόπος με τον οποίο ο προγραμματιστής μπορεί να προσθέσει τμήματα κώδικα της επιθυμίας του είναι δημιουργώντας παράγωγες κλάσεις (υποκλάσεις στην γλώσσα `Java`), στις οποίες καλείται να παρέχει **υπερφορτωμένες (overridden)** υλοποιήσεις των μεθόδων του «κενού» επισκέπτη.

Με αυτόν ακριβώς τον τρόπο είναι δομημένος και ο πηγαίος κώδικας της παρούσας εργασίας. Για περισσότερες λεπτομέρειες επί του τρόπου υλοποίησης του μεταγλωττιστή μη διστάσετε να ρωτήσετε τον γράφοντα κατά τη διάρκεια της παρουσίασης της εργασίας.

# 4 Παραδείγματα εκτέλεσης του μεταγλωττιστή

Στην ενότητα αυτή θα παρουσιάσουμε παραδείγματα εκτέλεσης του μεταγλωττιστή.

```
integer n,x,y(3),i,number
real f
write number
read n, (y(i), i=1,n) ,x
write "Squares: ", (y(i)**2, ",", i=1,99), y(100)**2
write fibonacci(n)
end

integer function z(integer number)
  if(number .eq. 3) then
       integer number
      number = 5
      z = number
  else
      z = number
  endif
end

subroutine number
  write 3
   return
end

integer function fibonacci(integer number)
    if(number .eq. 0) then
        fibonacci = 0
        return
    endif
    if(number .eq. 1) then
       fibonacci = 1
       return
    endif

    fibonacci = fibonacci(number - 1) + fibonacci(number-2)
end
```

Listing 1: Παράδειγμα προγράμματος `SimpleFortran` χωρίς συντακτικά και σημασιολογικά λάθη

Σύμφωνα με τις απαιτήσεις της εκφώνησης της εργασίας, ο μεταγλωττιστής αναλύει σημασιολογικά το πρόγραμμα, δημιουργεί πίνακα συμβόλων και εκτυπώνει το αφηρημένο συντακτικό δέντρο του προγράμματος:

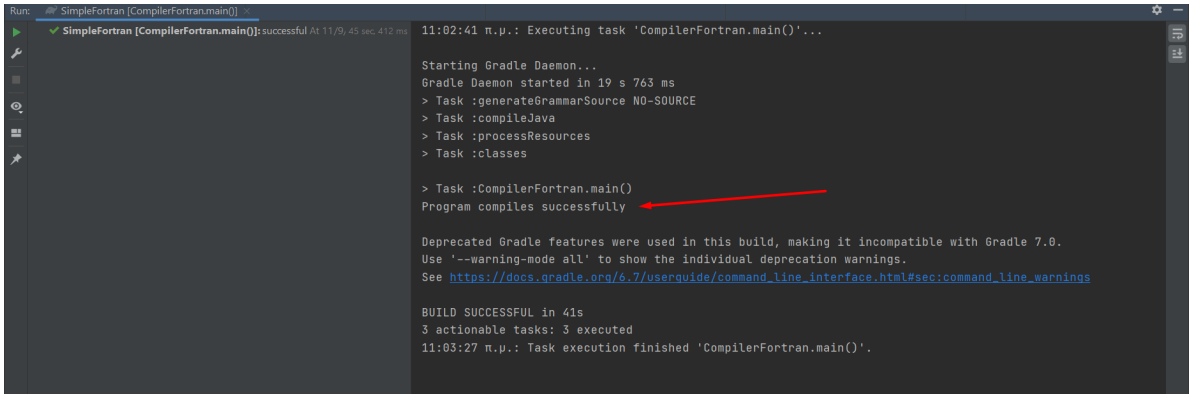



Figure 1: Η έξοδος του μεταγλωττιστή στην κονσόλα

Figure 2: Το συντακτικό δέντρο του προγράμματος

# 5 Association Rules – Κανόνες Συσχέτισης

**Ερώτημα:** Εφαρμόστε τον αλγόριθμο `Apriori` για τα χαρακτηριστικά 10 (action) εώς καί 35 (western) για να βρείτε τους 5 κανόνες με τη μεγαλύτερη εμπιστοσύνη (**confidence**) που έχουν υποστήριξη (**support**) $\geq$ 0.1.

**Απάντηση:** Ακολουθούμε τα βήματα της εκφώνησης και εκτελούμε τον αλγόριθμο `Apriori` από την καρτέλα “Associate” του Weka. Παρακάτω παρουσιάζεται η έξοδος του αλγορίθμου, με τους 5 κανόνες με τη μεγαλύτερη εμπιστοσύνη (**confidence**) που έχουν υποστήριξη (**support**) $\geq$ 0.1:

```
1. reality-tv=0 5041 ==> game-show=0 5041   <conf:(1)> lift:(1) lev:(0) [0] conv:(1)
2. news=0 reality-tv=0 5038 ==> game-show=0 5038   <conf:(1)> lift:(1) lev:(0) [0] conv:(1)
3. reality-tv=0 short=0 5036 ==> game-show=0 5036   <conf:(1)> lift:(1) lev:(0) [0] conv:(1)
4. film-noir=0 reality-tv=0 5035 ==> game-show=0 5035   <conf:(1)> lift:(1) lev:(0) [0] conv:(1)
5. news=0 reality-tv=0 short=0 5033 ==> game-show=0 5033   <conf:(1)> lift:(1) lev:(0) [0] conv:(1)
```

# 6 Visualize – Οπτικοποίηση

**Ερώτημα:** ) Χρησιμοποιήστε το menu Visualize του Weka για να σχεδιάσετε το διάγραμμα διασποράς (scatterplot) X:title_year, Y:imdb_score. Τι παρατηρείτε σχετικά με τις ταινίες που έχουν τιμή “bad”;

**Απάντηση:** Ακολουθούμε τα βήματα της εκφώνησης και παράγουμε το ακόλουθο διάγραμμα διασποράς:

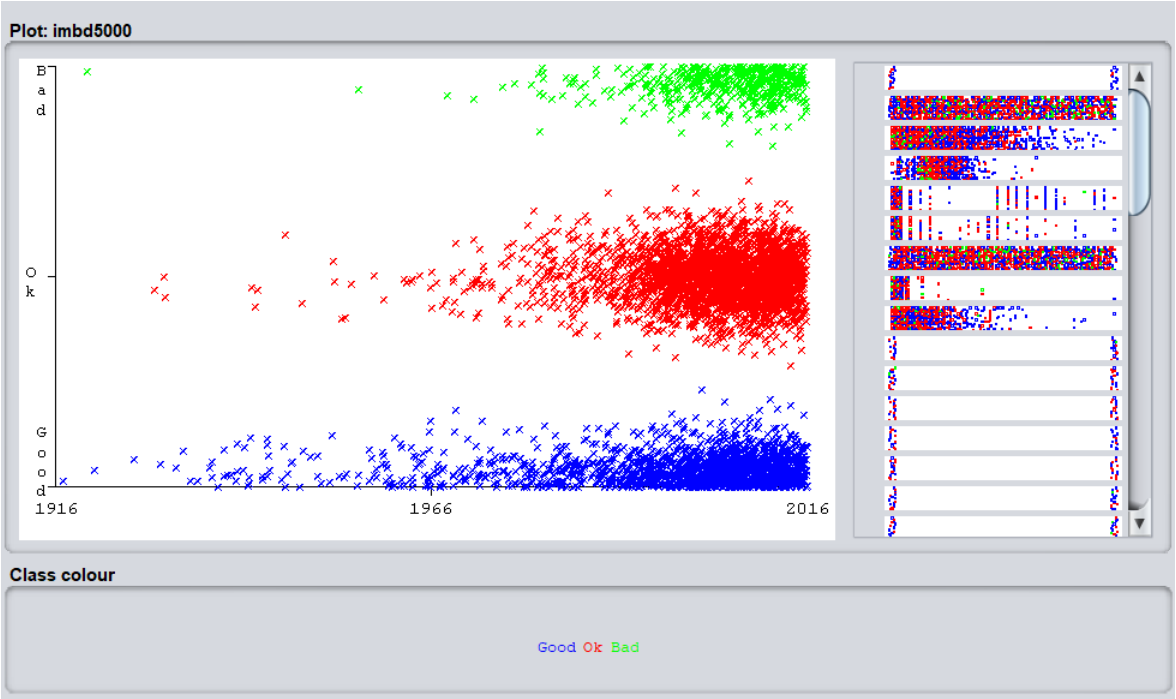



Figure 3: Διάγραμμα διασποράς X:title_year, Y:imdb_score

Παρατηρούμε ότι οι κακές ταινίες (πράσινο χρώμα) είναι συγκριτικά λιγότερες, σε σχέση με τις υπόλοιπες δύο κατηγορίες. Αναφορικά με την ημερομηνία δημοσίευσης των ταινιών, παρατηρούμε ότι οι ταινίες με τον χαρακτηρισμό “κακές” είναι ελάχιστες μέχρι και το 1966, οπότε και αρχίζουν να πληθαίνουν έως και το 2016, που είναι η ανώτερη τιμή του χαρακτηριστικού.