ΕΘΝΙΚΟ & ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

# ΘΠ06 Μεταγλωττιστές

## Εργασία Εξαμήνου:
## Υλοποίηση ενός Μεταγλωττιστή για τη γλώσσα προγραμματισμού Robin

**Διδάσκων: Μανόλης Κουμπαράκης**
**Βοηθός: Χαράλαμπος Νικολάου (charnik)**

# Εισαγωγή

Η εργασία του μαθήματος των μεταγλωττιστών αφορά τη σχεδίαση και υλοποίηση ενός μεταγλωττιστή για τη γλώσσα **Robin** η οποία περιγράφεται αναλυτικά παρακάτω. Γλώσσα υλοποίησης πρέπει να είναι η **C++** ή η C (κατά προτίμηση και δική σας διευκόλυνση η C++) και τα εργαλεία ***flex*** και ***bison***. Περισσότερες πληροφορίες σχετικά με αυτά τα εργαλεία υπάρχουν στην ιστοσελίδα του μαθήματος (`http://www.di.uoa.gr/~thp06/tools`).

# 1. Περιγραφή της γλώσσας Robin

Η γλώσσα **Robin** είναι μια απλή γλώσσα προγραμματισμού. Τα χαρακτηριστικά της εν συντομία είναι τα εξής:

- βασικοί τύποι δεδομένων για ακέραιους, χαρακτήρες και κινητής υποδιαστολής απλής ακρίβειας
- σύνθετοι τύποι δεδομένων (οριζόμενοι από το χρήστη)
- μονοδιάστατοι πίνακες με περιορισμένη λειτουργικότητα
- σύνταξη εντολών παρόμοια με αυτή της γλώσσας C
- ορισμό και κλήση συναρτήσεων παρόμοια με αυτή της γλώσσας C
- εμφωλιασμό μπλοκ εντολών και κανόνες εμβέλειας σύμφωνα με τη γλώσσα C
- συμπερίληψη αρχείων

Περισσότερες λεπτομέρειες της γλώσσας δίνονται στις παραγράφους που ακολουθούν.

## *1.1 Αναγνωριστικά*

Τα αναγνωριστικά (*identifiers*) της γλώσσας **Robin** απαρτίζονται από γράμματα (**A**..**Z**, **a**..**z**), ψηφία (**0**..**9**) και τον ειδικό χαρακτήρα _ (underscore). Ένα αναγνωριστικό δεν μπορεί να αρχίζει από ψηφίο. Τα πεζά γράμματα διαφέρουν από τα αντίστοιχα κεφαλαία.

Τα παρακάτω αναγνωριστικά είναι δεσμευμένα και δεν επιτρέπεται να χρησιμοποιούνται ως κοινά αναγνωριστικά. Η σημασία τους εξηγείται σε επόμενες παραγράφους.

```
char, int, float, else, if, main, return, void, while,
record
```

## *1.2 Βασικοί τύποι δεδομένων*

Οι βασικοί τύποι δεδομένων που υποστηρίζει η γλώσσα **Robin** είναι τρεις: **char** (για χαρακτήρες), **int** (για ακεραίους) και **float** (για κινητής υποδιαστολής απλής ακρίβειας).

Η χώρος μνήμης σε bytes που καταλαμβάνει ένας χαρακτήρας, ακέραιος και κινητής υποδιαστολής απλής ακρίβειας είναι 1, 4 και 4 αντίστοιχα.

Οι ακέραιες σταθερές αποτελούνται από ένα προαιρετικό πρόσημο ακολουθούμενο από ένα ή

περισσότερα δεκαδικά ψηφία. Παραδείγματα ακέραιων σταθερών είναι τα εξής:

```
0       5       -10     7483    -25983
```

Τις πραγματικές σταθερές κινητής υποδιαστολής απλής ακρίβειας, που αποτελούνται από ένα ακέραιο μέρος, ένα κλασματικό μέρος και ένα προαιρετικό εκθετικό μέρος. Το ακέραιο μέρος αποτελείται από ένα ή περισσότερα δεκαδικά ψηφία. Το κλασματικό μέρος αποτελείται από το χαρακτήρα . της υποδιαστολής, ακολουθούμενο από ένα ή περισσότερα δεκαδικά ψηφία. Τέλος, το εκθετικό μέρος αποτελείται από το πεζό ή κεφαλαίο γράμμα E, ένα προαιρετικό πρόσημο + ή - και ένα ή περισσότερα δεκαδικά ψηφία. Παραδείγματα πραγματικών σταθερών είναι τα ακόλουθα:

```
42.0            4.2e1           0.420e+2        42000.0e-3
```

Οι σταθερές τύπου χαρακτήρα αποτελούνται από έναν απλό ή ειδικό χαρακτήρα που περικλείεται από απλά εισαγωγικά. Ως απλοί χαρακτήρες θεωρούνται όλοι οι εκτυπώσιμοι χαρακτήρες πλην των εισαγωγικών (απλών και διπλών) και της ανάστροφης καθέτου \ (backslash). Οι ειδικοί χαρακτήρες χρησιμοποιούνται όπως στη γλώσσα C. Παριστάνονται ως ζεύγη χαρακτήρων, με πρώτο την ανάστροφη κάθετο. Οι επιτρεπόμενοι ειδικοί χαρακτήρες είναι οι εξής:

| **Χαρακτήρας** | **Περιγραφή** |
|---|---|
| **\n** | χαρακτήρας νέας γραμμής (new line) |
| **\"** | χαρακτήρας **"** (διπλό εισαγωγικό) |
| **\'** | χαρακτήρας **'** (απλό εισαγωγικό) |
| **\0** | χαρακτήρας με ASCII κωδικό **0** |
| **\t** | χαρακτήρας αλλαγής στήλης (TAB) |
| **** | χαρακτήρας \ (backslash) |

### *1.3 Σύνθετοι τύποι δεδομένων*

Η γλώσσα Robin επιτρέπει τον ορισμό σύνθετων τύπων δεδομένων. Ένας σύνθετος τύπος δεδομένων είναι μία συλλογή από μία ή περισσότερες μεταβλητές, πιθανώς διαφορετικών τύπων, που ομαδοποιούνται με ένα μόνο όνομα για ευκολία στο χειρισμό τους. Στο εξής θα ονομάζουμε αυτές τις μεταβλητές *πεδία* (fields). Οι σύνθετοι τύποι δεδομένων της γλώσσας **Robin** είναι αντίστοιχοι των δομών (struct) της γλώσσας C και των εγγραφών (record) της γλώσσας Pascal. Τα πεδία ενός σύνθετου τύπου δεδομένου της γλώσσας **Robin** μπορεί να έχουν τύπο μόνο κάποιο από τους βασικούς τύπους ή να είναι πίνακες βασικών τύπων. Έτσι, ένας σύνθετος τύπος δεν μπορεί να περιέχει έναν άλλο σύνθετο τύπο. Ένας σύνθετος τύπος σηματοδοτείται με τη δεσμευμένη λέξη **record**. Ένα παράδειγμα ενός σύνθετου τύπου είναι ο εξής:

```
record point {
    int x;
    int y;
```

```
};
```

Ο παραπάνω σύνθετος τύπος αναπαριστά την έννοια του σημείου στο επίπεδο. Κάθε σημείο ορίζεται από τις συντεταγμένες του, την τετμημένη και την τεταγμένη. Στον παραπάνω ορισμό αυτές αντιστοιχούν στα πεδία x και y αντίστοιχα.

Μία δήλωση μεταβλητής ενός σύνθετου τύπου γίνεται ως εξής:

```
record point p;
```

Τα πεδία ενός σύνθετου τύπου προσπελαύνονται με τον τελεστή **.** εφαρμοζόμενο πάνω σε μία μεταβλητή ενός σύνθετου τύπου και ακολουθούμενο από το όνομα του πεδίου (π.χ. **p.x ή p.y**). Όλοι οι σύνθετοι τύποι ενός προγράμματος θα πρέπει να ορίζονται πριν το κυρίως πρόγραμμα (βλ. επόμενη ενότητα). Ο χώρος μνήμης που καταλαμβάνει ένας σύνθετος τύπος εξαρτάται και υπολογίζεται αθροιστικά από τον αριθμό και το μέγεθος που καταλαμβάνουν τα πεδία του. Η **Robin**, όπως και η **C**, επιτρέπει τη δήλωση μεταβλητών με όνομα αυτό ενός σύνθετου τύπου δεδομένων.

### *1.4 Πίνακες*

Εκτός από τους βασικούς και τους σύνθετους τύπους δεδομένων, η γλώσσα **Robin** υποστηρίζει πίνακες (arrays) αποτελούμενους από στοιχεία των βασικών ή σύνθετων τύπων. Οι πίνακες όμως έχουν περιορισμένη λειτουργικότητα, όπως θα εξηγηθεί στις επόμενες παραγράφους.

Κατά τον ορισμό μιας μεταβλητής τύπου πίνακα, πρέπει να ορίζεται το μήκος του, δηλαδή το πλήθος των στοιχείων που τον αποτελούν. Ένας πίνακας π.χ. 20 χαρακτήρων με όνομα **name** ορίζεται ως εξής:

```
char name [20];
```

Τα στοιχεία του πίνακα μπορούν να προσπελαύνονται τοποθετώντας ένα δείκτη (έκφραση που επιστρέφει ακέραιο αριθμό) ανάμεσα σε αγκύλες, π.χ.

```
name[5]
```

Το πρώτο στοιχείο κάθε πίνακα αντιστοιχεί σε δείκτη **0**, ενώ το τελευταίο σε δείκτη ίσο με το μήκος του μείον ένα. Για τον πίνακα που ορίστηκε πιο πάνω, οι επιτρεπτές τιμές του δείκτη είναι οι ακέραιοι αριθμοί μεταξύ **0** και **19**.

Όπως σε κάθε γλώσσα προγραμματισμού, έτσι και στη γλώσσα **Robin** ιδιαίτερα χρήσιμοι είναι οι πίνακες χαρακτήρων, που χρησιμοποιούνται για την αποθήκευση συμβολοσειρών (strings). Η **Robin** επιτρέπει τη χρήση σταθερών συμβολοσειρών, δηλαδή σταθερών πινάκων αποτελούμενων από χαρακτήρες. Οι συμβολοσειρές είναι ακολουθίες απλών και ειδικών χαρακτήρων που περικλείονται από διπλά εισαγωγικά. Παραδείγματα συμβολοσειρών είναι τα εξής:

```
"This is a string!"
"Hello world!\n"
"My phone number is: 1234567."
```

Για την παράσταση των συμβολοσειρών, η γλώσσα **Robin** χρησιμοποιεί την ίδια σύμβαση με τη C, δηλαδή οι συμβολοσειρές είναι ακολουθίες χαρακτήρων που τερματίζονται με τον

ειδικό χαρακτήρα **`\0`** και δεν επιτρέπεται να εκτείνονται σε παραπάνω από μία γραμμή. Κατ' αυτό τον τρόπο το πραγματικό μήκος μιας συμβολοσειράς είναι κατά ένα μεγαλύτερο από το πλήθος των χαρακτήρων που την αποτελούν και πρέπει να γίνεται πρόβλεψη για τον επιπλέον χαρακτήρα.

### *1.5 Μονάδες προγράμματος*

Ένα πρόγραμμα της γλώσσας **Robin** έχει τη δομή ενός προγράμματος C. Η γλώσσα **Robin** δε διαφοροποιεί τις μονάδες προγράμματος: το κυρίως πρόγραμμα, οι διαδικασίες και οι συναρτήσεις αντιμετωπίζονται από κοινού ως συναρτήσεις, και ως τέτοιες δηλώνονται. Το κυρίως πρόγραμμα πρέπει να ονομάζεται **`main`** και να επιστρέφει τύπο **`void`**. Σε περίπτωση που κάποια μονάδα προγράμματος δεν επιστρέφει αποτέλεσμα, τότε δηλώνεται με τύπο **`void`**. Επισημαίνεται ότι μια συνάρτηση δεν μπορεί να επιστρέφει αποτέλεσμα τύπου πίνακα.

Οι συναρτήσεις (όπως άλλωστε και οι μεταβλητές) πρέπει να έχουν δηλωθεί πριν χρησιμοποιηθούν. Η δήλωση αυτή μπορεί να γίνει με δυο τρόπους:

- με πλήρη ορισμό (επικεφαλίδα και κώδικας)
- με απλή δήλωση (επικεφαλίδα μόνο)

Στην περίπτωση δήλωσης συνάρτησης με απλή δήλωση, ο ορισμός της θα πρέπει να παρουσιασθεί κάποια στιγμή στη συνέχεια του προγράμματος. Σε κάθε περίπτωση η δήλωση και ο ορισμός κάθε συνάρτησης πρέπει να παρουσιάζεται μία μόνο φορά.

Οι παράμετροι των συναρτήσεων μπορούν να περνούν με δύο τρόπους: ***κατ' αξία*** ή ***κατ' αναφορά***. Το πέρασμα κατ' αναφορά δηλώνεται με το σύμβολο **`&`** που προηγείται του ονόματος της τυπικής παραμέτρου στην επικεφαλίδα της συνάρτησης, π.χ.

```
void f (int by_value, int & by_reference)
```

Στην περίπτωση που κάποια τυπική παράμετρος μιας συνάρτησης είναι πίνακας, στην επικεφαλίδα της συνάρτησης δεν αναγράφεται το μήκος του πίνακα. Αν είναι απαραίτητο αυτό να είναι γνωστό στη συνάρτηση, πρέπει να περνάει ως ανεξάρτητη παράμετρος. Πίνακες μπορούν να περνούν και με τους δύο τρόπους (αξία και αναφορά). Παραδείγματα συναρτήσεων με τυπικές παραμέτρους τύπου πίνακα είναι οι εξής:

```
void print (char s[])
int sum (int & values[], int size)
```

### *1.6 Εντολές*

Οι εντολές της γλώσσας **Robin** έχουν ακριβώς την ίδια μορφή με τις αντίστοιχες εντολές της γλώσσας C. Με εξαίρεση τη σύνθετη εντολή (compound statement), πρέπει να τερματίζονται υποχρεωτικά με το χαρακτήρα **`;`** (semicolon).

Η εντολή εκχώρησης έχει την ίδια σύνταξη και σημασιολογία με την αντίστοιχη εντολή της C. Επιτρέπεται μόνο η εκχώρηση απλών τύπων: απαγορεύεται η εκχώρηση ολόκληρων πινάκων, αλλά επιτρέπεται η εκχώρηση σε στοιχεία πίνακα. Επίσης, δεν επιτρέπεται η εκχώρηση τιμής μεταξύ διαφορετικών τύπων, εκτός της περίπτωσης της εκχώρησης της τιμής ενός ακεραίου σε μία μεταβλητή τύπου κινητής υποδιαστολής.

Οι εντολές ελέγχου που επιτρέπει η γλώσσα είναι οι εξής :

```
if ( λογική-έκφραση ) εντολή [ else εντολή ]
while ( λογική-έκφραση ) εντολή
```

Η σημασιολογία των εντολών αυτών είναι η ίδια όπως στη γλώσσα C.

Η εντολή κλήσης συνάρτησης έχει την ίδια σύνταξη και σημασιολογία με την αντίστοιχη εντολή της γλώσσας C. Η εντολή **`return`** χρησιμοποιείται για την επιστροφή από συνάρτηση. Όταν πρόκειται για συνάρτηση που επιστρέφει κάποιο τύπο, πρέπει να ακολουθείται από έκφραση του ίδιου τύπου με αυτόν που επιστρέφει η συνάρτηση.

## *1.7 Τελεστές*

Οι τελεστές της γλώσσας **Robin** δίνονται παρακάτω. Η ***προτεραιότητα*** (*precedence*) και η ***προσεταιριστικότητα*** (*associativity*) των τελεστών αυτών είναι η ίδια με αυτή των τελεστών της γλώσσας C (για λεπτομέρειες ανατρέξτε στο βιβλίο των Kernighan & Ritchie, "Η Γλώσσα Προγραμματισμού C"). Δεν υπάρχουν τελεστές που να επιστρέφουν αποτέλεσμα τύπου χαρακτήρα ή πίνακα.

### *1.7.1 Αριθμητικοί τελεστές*

Οι αριθμητικοί τελεστές με ένα τελούμενο της γλώσσας **Robin** είναι οι εξής:

| **Τελεστής** | **Σημασία** |
|---|---|
| **+** | Θετικό πρόσημο (ταυτότητα) |
| **-** | Αρνητικό πρόσημο (αντίθετος) |

Οι δυαδικοί αριθμητικοί τελεστές της γλώσσας είναι οι εξής:

| **Τελεστής** | **Σημασία** |
|---|---|
| **+** | Πρόσθεση ακεραίων |
| **-** | Αφαίρεση ακεραίων |
| ***** | Πολλαπλασιασμός ακεραίων |
| **/** | Πηλίκο ακέραιας διαίρεσης |
| **%** | Υπόλοιπο ακέραιας διαίρεσης |

Τα τελούμενα των αριθμητικών τελεστών πρέπει να είναι έγκυρες αριθμητικές εκφράσεις ακέραιου τύπου (**int**) ή τύπου κινητής υποδιαστολής (**float**). Στην περίπτωση των δυαδικών αριθμητικών τελεστών, αν και τα δύο είναι ακέραιου τύπου, τότε και το αποτέλεσμα είναι ακέραιου τύπου, διαφορετικά το αποτέλεσμα είναι τύπου κινητής υποδιαστολής, **float**. Στην περίπτωση του τελεστή υπολοίπου ακέραιας διαίρεσης, %, τα τελούμενα, όπως και το αποτέλεσμα της πράξης, θα πρέπει να είναι ακέραιου τύπου.

### *1.7.2 Σχεσιακοί τελεστές*

Οι σχεσιακοί τελεστές της γλώσσας **Robin** φαίνονται στον παρακάτω πίνακα. Τα τελούμενά τους πρέπει να είναι έγκυρες ακέραιες εκφράσεις ή χαρακτήρες. Πρέπει όμως τα δυο

τελούμενα να είναι του ίδιου τύπου. Απαγορεύονται συγκρίσεις ολόκληρων πινάκων, επιτρέπονται όμως οι συγκρίσεις στοιχείων πινάκων. Συγκρίσεις μεταξύ χαρακτήρων γίνονται με βάση τους αντίστοιχους κωδικούς ASCII (ακέραιοι από **`0`** ως **`255`**).

| Τελεστής | Σημασία |
|---|---|
| **`==`** | ίσο |
| **`!=`** | διάφορο |
| **`>`** | μεγαλύτερο |
| **`<`** | μικρότερο |
| **`>=`** | μεγαλύτερο ή ίσο |
| **`<=`** | μικρότερο ή ίσο |

Τα αποτελέσματα των σχεσιακών τελεστών είναι λογικές εκφράσεις. Οι εκφράσεις αυτές μπορούν να χρησιμοποιηθούν μόνο σε εντολές **`if`** και **`while`**. Δεν μπορούν να εκχωρηθούν σε μεταβλητές, να περάσουν ως παράμετροι ούτε να επιστραφούν ως αποτελέσματα συναρτήσεων.

#### *1.7.3 Λογικοί τελεστές*

Οι λογικοί τελεστές της γλώσσας **Robin** είναι τρεις:

| Τελεστής | Σημασία |
|---|---|
| **`\|\|`** | Λογική διάζευξη (ή) |
| **`&&`** | Λογική σύζευξη (και) |
| **`!`** | Λογική άρνηση (όχι) |

Οι τελεστές **`||`** και **`&&`** είναι δυαδικοί, ενώ ο τελεστής **`!`** δέχεται ένα τελούμενο. Τα τελούμενα πρέπει να είναι έγκυρες λογικές εκφράσεις. Το αποτέλεσμα είναι επίσης λογική έκφραση.

#### *1.7.4 Τελεστής προσπέλασης πεδίου*

Η **Robin** χρησιμοποιεί τον τελεστή προσπέλασης πεδίου, **`.`**, για την αναφορά σε πεδία σύνθετων τύπων δεδομένων. Ο τελεστής αναφοράς μπορεί να εφαρμοστεί μόνο σε τελούμενα σύνθετων τύπων δεδομένων.

### *1.8 Σχόλια*

Τα σχόλια στη γλώσσα **Robin** είναι δυο ειδών. Τα κοινά σχόλια είναι ίδια με αυτά της γλώσσας C. Περικλείονται από τις ακολουθίες χαρακτήρων **`/*`** και **`*/`** και δεν μπορούν να είναι φωλιασμένα (nested). Η γλώσσα **Robin,** όμως, υποστηρίζει και ένα δεύτερο τύπο σχολίων, που αρχίζουν με την ακολουθία χαρακτήρων **`//`** και εκτείνονται ως το τέλος της γραμμής (όπως στη γλώσσα C++).

### *1.9 Οδηγίες προς το μεταγλωττιστή*

Η μοναδική οδηγία, που επιτρέπει ο μεταγλωττιστής της γλώσσας **Robin** είναι η οδηγία **`#include`**. Η οδηγία αυτή έχει την ίδια σημασιολογία όπως και στη γλώσσα C, δηλαδή επιτρέπει την ανάγνωση ενός εξωτερικού αρχείου σαν αυτό να ήταν τμήμα του προγράμματος. Πρέπει να βρίσκεται υποχρεωτικά στην αρχή του προγράμματος. Η σύνταξή της είναι η εξής:

```
#include "όνομα αρχείου"
```

Η οδηγία αυτή απευθύνεται ουσιαστικά στο λεκτικό αναλυτή. Σε περίπτωση που αυτός συναντήσει αυτή την οδηγία, θα πρέπει να σταματήσει την ανάγνωση του αρχείου προγράμματος, και να συνεχίσει με την επεξεργασία του αρχείου που ζητείται να συμπεριληφθεί. Μετά το τέλος αυτού του αρχείου, ο λεκτικός αναλυτής πρέπει να συνεχίσει από το σημείο του αρχείου προγράμματος, στο οποίο είχε σταματήσει. Για δική σας ευκολία, τα αρχεία που διαβάζονται με την οδηγία **`#include`** δεν μπορούν να περιέχουν αντίστοιχες οδηγίες. Φυσικά μεμονωμένες λεκτικές μονάδες καθώς και σχόλια πρέπει να περιέχονται πλήρως σε ένα αρχείο προγράμματος (δεν επιτρέπεται να αρχίζουν σε ένα αρχείο προγράμματος και να τελειώνουν σε κάποιο άλλο).

## 2. Η γραμματική της γλώσσας Robin

Παρακάτω δίνεται η γραμματική της γλώσσας **Robin** σε μορφή extended BNF. Τα τερματικά σύμβολα αναγράφονται με έντονους χαρακτήρες και μέσα σε διπλά εισαγωγικά. Μέσα σε γωνιακές αγκύλες αναγράφονται τα μη τερματικά σύμβολα και κατ' εξαίρεση οι λεκτικές μονάδες:

```
<integer-constant> : ακέραια σταθερά (χωρίς πρόσημο)
<char-constant>    : σταθερή τύπου χαρακτήρα
<string-constant>  : σταθερή συμβολοσειρά
<id>               : αναγνωριστικό
```

Ακολουθεί η γραμματική της γλώσσας.

```
<program>                  ::=  (<global-declaration-definition>)*
                                <program-header>
                                <compound-statement>

                                (<function-definition>)*

<global-declaration-definition>     ::=  (<function-prototype>)
                                    |    (<record-definition>)
                                    |    (<function-definition>)
                                    |    (<variable-definition>)

<record-definition> ::= <composite_type> “{“
                            (<primitive-type-def>)+ “}” “;”

<composite_type> ::= “record” <id>
```

```
<program-header> ::=  “void” “main” “(” “)”

<function-prototype> ::=  <function-header> “;”

<function-definition>::=  <function-header>
                      <compound-statement>

<function-header> ::=<return-type> <id>
                          “(” [<formal-params>] “)”

<return-type>    ::=  <data-type> | <composite-type> | “void”

<formal-params>  ::=  <formal-parameter>
                          (“,” <formal-parameter>)*

<formal-parameter>   ::=  <data-type> [“&”] <id> [“[” “]”]

                     |    <composite-type> [“&”] <id> [“[” “]”]

<data-type>      ::=  “int” | “float” | “char”

<variable-definition>::=  <primitive-type-def>

                     |    <composite-type-def>

<primitive-type-def> ::=  <data-type> <def-one-variable>
                          (“,” <def-one-variable>)* “;”

<composite-type-def> ::=  <composite_type> <def-one-variable>
                          (“,” <def-one-variable>)* “;”

<def-one-variable>   ::=  <id> [“[” <integer-constant> “]”]

<statement>      ::=  <assignment>
                   |  <if-statement>
                   |  <while-statement>
                   |  <void-function-call>
                   |  <return-statement>
                   |  <compound-statement>
                   |  <empty-statement>

<assignment>     ::=  <comp_l-value> “=” <expression> “;”

<comp_l-value>   ::=  <id> “.” <l-value>

                   |  <l-value>

<l-value>        ::=  <id> [“[” <expression> “]”]

<if-statement>   ::=  “if” “(” <b-expression> “)”
                          <statement>
                      [“else” <statement>]

<while-statement> ::=“while” “(” <b-expression> “)”
                          <statement>

<void-function-call> ::=  <function-call> “;”

<function-call>  ::=  <id> “(” [ <actual-params> ] “)”

<actual-params>  ::=  <actual-parameter>
                          (“,” <actual-parameter>)*

<actual-parameter>   ::=  <expression>
                     |    <string-constant>

<return-statement>   ::=  “return” [ <expression> ] “;”

<empty-statement> ::=“;”

<compound-statement> ::=  “{” (<variable-definition>)* (<statement>)* “}”

<b-expression>   ::=  <b-expression> “||” <b-expression>
                   |  <b-expression> “&&” <b-expression>
```

```
                    |   “!” <b-expression>
                    |   <expression> “==” <expression>
                    |   <expression> “!=” <expression>
                    |   <expression> “>” <expression>
                    |   <expression> “<” <expression>
                    |   <expression> “>=” <expression>
                    |   <expression> “<=” <expression>

<expression>     ::=  <expression> “+” <expression>
                    |   <expression> “-” <expression>
                    |   <expression> “*” <expression>
                    |   <expression> “/” <expression>
                    |   <expression> “%” <expression>
                    |   “+” <expression>
                    |   “-” <expression>
                    |   <comp_l-value>
                    |   <integer-constant>
                    |   <char-constant>
                    |   <function-call>
```

## 3. Η βιβλιοθήκη έτοιμων συναρτήσεων

Όπως και στη C, προκειμένου να υλοποιηθούν λειτουργίες που δεν υποστηρίζονται άμεσα από τη γλώσσα **Robin** (π.χ., είσοδος/έξοδος), χρειαζόμαστε βιβλιοθήκες έτοιμων συναρτήσεων. Στην περίπτωση της **Robin** θα υποθέσουμε ότι έχουμε μια standard input-output library που καθορίζεται από το αρχείο **`“robin_io.rob”.`** Το αρχείο αυτό πρέπει να περικλείεται σε κάθε πρόγραμμα που χρησιμοποιεί λειτουργίες input/output χρησιμοποιώντας την οδηγία **`#include`**.

Επειδή δεν θα δημιουργήσουμε τελικό κώδικα δεν μας ενδιαφέρει να έχουμε μια υλοποίηση των συναρτήσεων που περιλαμβάνονται στο αρχείο **`robin_io.rob`**. Για τις ανάγκες της εργασίας σας λοιπόν, απλά δημιουργείστε ένα αρχείο **`robin_io.rob`** που να περιέχει τα πρωτότυπα των παρακάτω συναρτήσεων που υλοποιούν τις λειτουργίες input/output για τους τύπους δεδομένων της γλώσσας **Robin**.

### `void put_char (char c)`

Εκτυπώνει το χαρακτήρα **`c`** στο τρέχον αρχείο εξόδου.

### `void put_int (int i)`

Εκτυπώνει τον ακέραιο **`i`** στο τρέχον αρχείο εξόδου σε δεκαδική μορφή.

### `void put_float (float f)`

Εκτυπώνει τον δεκαδικό **`f`** στο τρέχον αρχείο εξόδου.

### `void put_string (char s[])`

Εκτυπώνει τη συμβολοσειρά **`s`** στο τρέχον αρχείο εξόδου. Η συμβολοσειρά πρέπει να τερματίζεται με τον ειδικό χαρακτήρα **`\0`**.

### `char get_char ()`

Διαβάζει και επιστρέφει ένα χαρακτήρα από το αρχείο εισόδου. Σε περίπτωση που δεν

υπάρχουν άλλοι χαρακτήρες (τέλος αρχείου) επιστρέφεται ο ειδικός χαρακτήρας **`\0`**.

### `int get_int ()`

Διαβάζει και επιστρέφει έναν ακέραιο αριθμό από το τρέχον αρχείο εισόδου σε δεκαδική μορφή. Αγνοούνται κενοί χαρακτήρες που πιθανώς προηγούνται του ακέραιου αριθμού.

### `float get_float ()`

Διαβάζει και επιστρέφει ένα δεκαδικό αριθμό από το τρέχον αρχείο εισόδου. Αγνοούνται κενοί χαρακτήρες που πιθανώς προηγούνται του δεκαδικού αριθμού.

### `void get_string (char & s[], int size)`

Διαβάζει μια συμβολοσειρά από το τρέχον αρχείο εισόδου και την αποθηκεύει στη μεταβλητή **`s`**. Η μεταβλητή **`size`** περιέχει το μήκος της συμβολοσειράς **`s`** και χρησιμεύει ως όριο χαρακτήρων που μπορούν να τοποθετηθούν σε αυτή. Διαβάζονται χαρακτήρες μέχρι να εξαντληθεί το μήκος της συμβολοσειράς **`s`** ή να συναντηθεί ο ειδικός χαρακτήρας **`\n`** (τέλος γραμμής). Ο χαρακτήρας αυτός αφαιρείται από το αρχείο εισόδου, δεν τοποθετείται όμως στη συμβολοσειρά **`s`**. Η συμβολοσειρά **`s`** πάντα τερματίζεται με τον ειδικό χαρακτήρα **`\0`**.

## 4. Παραδείγματα

Στην παράγραφο αυτή δίνονται τέσσερα παραδείγματα προγραμμάτων στη γλώσσα **Robin**. Δίνεται επίσης ο ενδιάμεσος κώδικας, που πρέπει να παράγεται από το μεταγλωττιστή της **Robin**. Σημειώνεται ότι σε καμιά φάση της μεταγλώττισης δεν έχουν γίνει βελτιστοποιήσεις.

### *4.1 Πες γεια!*

Το παρακάτω παράδειγμα είναι το απλούστερο πρόγραμμα στη γλώσσα **Robin**, που παράγει κάποιο αποτέλεσμα ορατό στο χρήστη. Το πρόγραμμα αυτό τυπώνει απλώς το μήνυμα: **`"Hello world!"`**.

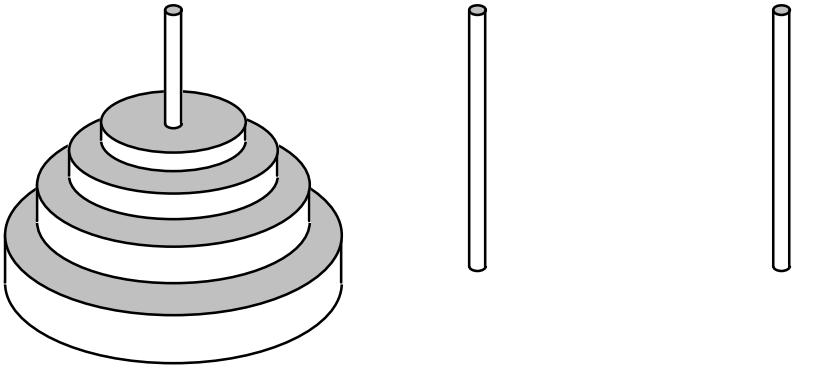
```
#include "robin_io.rob"

void main ()
{
   put_string("Hello world!\n");
}
```

### *4.2 Οι πύργοι του Hanoi*

Το πρόγραμμα που ακολουθεί λύνει το πρόβλημα των πύργων του Hanoi. Μια σύντομη περιγραφή του προβλήματος δίνεται παρακάτω.

Υπάρχουν τρεις στύλοι, στον πρώτο από τους οποίους είναι περασμένοι n το πλήθος δακτύλιοι. Οι εξωτερικές διάμετροι των δακτυλίων είναι διαφορετικές και αυτοί είναι περασμένοι από κάτω προς τα πάνω σε αύξουσα σειρά εξωτερικής διαμέτρου, όπως φαίνεται στο Σχ.1. Ζητείται να μεταφερθούν οι δακτύλιοι από τον πρώτο στον τρίτο στύλο




**Σχ.1. Οι πύργοι του Hanoi.**

(χρησιμοποιώντας το δεύτερο ως βοηθητικό χώρο), ακολουθώντας όμως τους εξής κανόνες :

- Κάθε φορά επιτρέπεται να μεταφερθεί ένας μόνο δακτύλιος, από κάποιο στύλο σε κάποιον άλλο στύλο.
- Απαγορεύεται να τοποθετηθεί δακτύλιος με μεγαλύτερη διάμετρο πάνω από δακτύλιο με μικρότερη διάμετρο.

Το πρόγραμμα στη γλώσσα **Robin** που λύνει αυτό το πρόβλημα δίνεται παρακάτω. Η συνάρτηση **`hanoi`** είναι αναδρομική.

```
#include "robin_io.rob"

void hanoi (char source[], char target[],
                  char auxiliary[], int rings);
void move (char source[], char target[]);

void main () {
   int NumberOfRings;

   put_string("Please, give the number of rings : ");
   NumberOfRings = get_int();
   put_string("\nHere is the solution :\n\n");
   hanoi("left", "right", "middle", NumberOfRings);
}

void hanoi (char source[], char target[],
                  char auxiliary[], int rings)  {
      if (rings < 1)
         return;
      else {
         hanoi(source, auxiliary, target, rings-1);
         move(source, target);
         hanoi(auxiliary, target, source, rings-1);
      }
}

void move (char source[], char target[]) {
     put_string("Move from ");
     put_string(source);
     put_string(" to ");
     put_string(target);
     put_string(".\n");
}
```

Ο ενδιάμεσος κώδικας, που πρέπει να παράγεται από το μεταγλωττιστή της **Robin** είναι ο εξής:

```
1:   unit, move, -, -
2:   par, "Move from ", V, -
3:   call, -, -, put_string
4:   par, source, V, -
5:   call, -, -, put_string
6:   par, " to ", V, -
7:   call, -, -, put_string
8:   par, target, V, -
9:   call, -, -, put_string
10:  par, ".\n", V, -
11:  call, -, -, put_string
12:  endu, move, -, -

13:  unit, hanoi, -, -
14:  <, rings, 1, 16
15:  jump, -, -, 17
16:  ret, -, -, -
17:  par, source, V, -
18:  par, auxiliary, V, -
19:  par, target, V, -
20:  -, rings, 1, $1
21:  par, $1, V, -
22:  call, -, -, hanoi
23:  par, source, V, -
24:  par, target, V, -
25:  call, -, -, move
26:  par, auxiliary, V, -
27:  par, target, V, -
28:  par, source, V, -
29:  -, rings, 1, $2
30:  par, $2, V, -
31:  call, -, -, hanoi
32:  endu, hanoi, -, -

33:  unit, main, -, -
34:  par, "Please, give the number of rings : ", V, -
35:  call, -, -, put_string
36:  par, $3, RET, -
37:  call, -, -, get_int
38:  :=, $3, -, NumberOfRings
39:  par, "\nHere is the solution :\n\n", V, -
40:  call, -, -, put_string
41:  par, "left", V, -
42:  par, "right", V, -
43:  par, "middle", V, -
44:  par, NumberOfRings, V, -
45:  call, -, -, hanoi
46:  endu, main, -, -
```

### *4.3 Πρώτοι αριθμοί*

Το παρακάτω παράδειγμα προγράμματος στη γλώσσα **Robin** είναι ένα πρόγραμμα που υπολογίζει τους πρώτους αριθμούς μεταξύ **1** και **n**, όπου το **n** καθορίζεται από το χρήστη. Το πρόγραμμα αυτό χρησιμοποιεί έναν απλό αλγόριθμο για τον υπολογισμό των πρώτων αριθμών. Μια διατύπωση αυτού του αλγορίθμου σε ψευδογλώσσα δίνεται παρακάτω. Λαμβάνεται υπόψη ότι οι αριθμοί **2** και **3** είναι πρώτοι, και στη συνέχεια εξετάζονται μόνο οι αριθμοί της μορφής **6k±1**, όπου **k** ακέραιος αριθμός.

***Κύριο πρόγραμμα***

τύπωσε τους αριθμούς 2 και 3
t := 6
όσο t <= n κάνε τα εξής :
    αν ο αριθμός t-1 είναι πρώτος τότε τύπωσέ τον
    αν ο αριθμός t+1 είναι πρώτος τότε τύπωσέ τον
    t := t+6

***Αλγόριθμος ελέγχου (είναι ο αριθμός t πρώτος;)***

αν t < 0 τότε έλεγξε τον αριθμό -t
αν t < 2 τότε ο t δεν είναι πρώτος
αν t = 2 τότε ο t είναι πρώτος
αν ο t διαιρείται με το 2 τότε ο t δεν είναι πρώτος
i := 3
όσο i <= t/2 κάνε τα εξής :
    αν ο t διαιρείται με τον i τότε ο t δεν είναι πρώτος
    i := i + 2
ο t είναι πρώτος

Το αντίστοιχο πρόγραμμα σε γλώσσα **Robin** είναι το εξής:

```
#include "robin_io.rob"

int prime (int n);

void main () {
   int limit, number, counter;

   put_string("Please, give the upper limit : ");
   limit = get_int();
   put_string("Prime numbers between 0 and ");
   put_int(limit);
   put_string("\n\n");
   counter = 0;
   if (limit >= 2) {
      counter = counter + 1;
      put_string("2\n");
   }
   if (limit >= 3) {
      counter = counter + 1;
      put_string("3\n");
   }
   number = 6;
   while (number <= limit) {
      if (prime(number-1) > 0) {
         counter = counter + 1;
         put_int(number-1);
         put_string("\n");
      }
      if (number != limit && prime(number+1) > 0) {
         counter = counter + 1;
         put_int(number+1);
         put_string("\n");
      }
      number = number + 6;
   }
   put_string("\n");
   put_int(counter);
   put_string(" prime number(s) were found.\n");
}

int prime (int n) {
   int i;
```

```
        if (n < 0)
           return prime(-n);
        if (n < 2)
           return 0;
        if (n == 2)
           return 1;
        if (n % 2 == 0)
           return 0;
        i = 3;
        while (i <= n/2) {
           if (n % i == 0)
              return 0;
           i = i+2;
        }
        return 1;
    }
```

Ο ενδιάμεσος κώδικας, που πρέπει να παράγεται από το μεταγλωττιστή της **Robin** δίνεται παρακάτω.

```
1:   unit, prime, -, -
2:   <, n, 0, 4
3:   jump, -, -, 10
4:   -, n, -, $1
5:   par, $1, V, -
6:   par, $2, RET, -
7:   call, -, -, prime
8:   retv, $2, -, -
9:   ret, -, -, -
10:  <, n, 2, 12
11:  jump, -, -, 14
12:  retv, 0, -, -
13:  ret, -, -, -
14:  =, n, 2, 16
15:  jump, -, -, 18
16:  retv, 1, -, -
17:  ret, -, -, -
18:  %, n, 2, $3
19:  =, $3, 0, 21
20:  jump, -, -, 23
21:  retv, 0, -, -
22:  ret, -, -, -
23:  :=, 3, -, i
24:  /, n, 2, $4
25:  <=, i, $4, 27
26:  jump, -, -, 35
27:  %, n, i, $5
28:  =, $5, 0, 30
29:  jump, -, -, 32
30:  retv, 0, -, -
31:  ret, -, -, -
32:  +, i, 2, $6
33:  :=, $6, -, i
34:  jump, -, -, 24
35:  retv, 1, -, -
36:  ret, -, -, -
37:  endu, prime, -, -

38:  unit, main, -, -
39:  par, "Please, give the upper limit : ", V, -
40:  call, -, -, put_string
41:  par, $7, RET, -
42:  call, -, -, get_int
43:  :=, $7, -, limit
44:  par, "Prime numbers between 0 and ", V, -
```

```
45:  call, -, -, put_string
46:  par, limit, V, -
47:  call, -, -, put_int
48:  par, "\n\n", V, -
49:  call, -, -, put_string
50:  :=, 0, -, counter
51:  >=, limit, 2, 53
52:  jump, -, -, 57
53:  +, counter, 1, $8
54:  :=, $8, -, counter
55:  par, "2\n", V, -
56:  call, -, -, put_string
57:  >=, limit, 3, 59
58:  jump, -, -, 63
59:  +, counter, 1, $9
60:  :=, $9, -, counter
61:  par, "3\n", V, -
62:  call, -, -, put_string
63:  :=, 6, -, number
64:  <=, number, limit, 66
65:  jump, -, -, 97
66:  -, number, 1, $10
67:  par, $10, V, -
68:  par, $11, RET, -
69:  call, -, -, prime
70:  >, $11, 0, 72
71:  jump, -, -, 79
72:  +, counter, 1, $12
73:  :=, $12, -, counter
74:  -, number, 1, $13
75:  par, $13, V, -
76:  call, -, -, put_int
77:  par, "\n", V, -
78:  call, -, -, put_string
79:  <>, number, limit, 81
80:  jump, -, -, 94
81:  +, number, 1, $14
82:  par, $14, V, -
83:  par, $15, RET, -
84:  call, -, -, prime
85:  >, $15, 0, 87
86:  jump, -, -, 94
87:  +, counter, 1, $16
88:  :=, $16, -, counter
89:  +, number, 1, $17
90:  par, $17, V, -
91:  call, -, -, put_int
92:  par, "\n", V, -
93:  call, -, -, put_string
94:  +, number, 6, $18
95:  :=, $18, -, number
96:  jump, -, -, 64
97:  par, "\n", V, -
98:  call, -, -, put_string
99:  par, counter, V, -
100: call, -, -, put_int
101: par, " prime number(s) were found.\n", V, -
102: call, -, -, put_string
103: endu, main4.4, -, -
```

### *4.5 5.4 Ταξινόμηση με τη μέθοδο της φυσαλίδας*

Ο αλγόριθμος της φυσαλίδας (bubble sort) είναι ένας από τους πιο γνωστούς και απλούς αλγορίθμους ταξινόμησης. Το παρακάτω πρόγραμμα σε **Robin** ταξινομεί έναν πίνακα ακεραίων αριθμών κατ' αύξουσα σειρά, χρησιμοποιώντας αυτό τον αλγόριθμο. Αν **x** είναι ο

πίνακας που πρέπει να ταξινομηθεί και **n** είναι το μέγεθός του (θεωρούμε ότι τα στοιχεία του είναι τα **`x[0]`**, **`x[1]`**, ..., **`x[n-1]`**), ο αλγόριθμος περιγράφεται με ψευδοκώδικα ως εξής:

***Αλγόριθμος της φυσαλίδας (bubble sort)***

```
επανάλαβε το εξής
    για i από 0 ως n-2
        αν x[i] < x[i+1]
            αντίστρεψε τα x[i] και x[i+1]
όσο μεταβάλλεται η σειρά των στοιχείων του x
```

Το αντίστοιχο πρόγραμμα σε γλώσσα **Robin** είναι το εξής :

```
#include "robin_io.rob"

void swap (int & x, int & y);
void PrintArray (char msg[], int array[], int size);
void BubbleSort (int & array[], int size);

void main () {
     int i, x[16], seed;
     i = 0;
     seed = 65;
     while (i < 16) {
          seed = (seed * 137 + 221 + i) % 101;
          x[i] = seed;
          i = i + 1;
     }
     PrintArray("Initial array: ", x, 16);
     BubbleSort(x, 16);
     PrintArray("Sorted array: ", x, 16);
}

void swap (int & x, int & y) {
     int t;

     t = x;
     x = y;
     y = t;
}


void PrintArray (char msg[], int array[], int size) {
      int i;
      put_string(msg);
      i = 0;
      while (i < size) {
         if (i > 0)
            put_string(", ");
         put_int(array[i]);
         i = i+1;
      }
      put_string("\n");
}


void BubbleSort (int & array[], int size) {
     int i, changed;
     changed = 1;
     while (changed == 1) {
          i = 0;
          changed = 0;
          while (i < size-1) {
               if (array[i] < array[i+1]) {
                    swap(array[i], array[i+1]);
```

```
                    changed = 1;
                }
                i = i+1;
            }
        }
    }
```

Ο ενδιάμεσος κώδικας, που πρέπει να παράγεται από το μεταγλωττιστή της **Robin** δίνεται παρακάτω.

```
1:   unit, swap, -, -
2:   :=, x, -, t
3:   :=, y, -, x
4:   :=, t, -, y
5:   endu, swap, -, -

6:   unit, BubbleSort, -, -
7:   :=, 1, -, changed
8:   =, changed, 1, 10
9:   jump, -, -, 27
10:  :=, 0, -, i
11:  :=, 0, -, changed
12:  -, size, 1, $1
13:  <, i, $1, 15
14:  jump, -, -, 26
15:  +, i, 1, $2
16:  <, array[i], array[$2], 18
17:  jump, -, -, 23
18:  par, array[i], R, -
19:  +, i, 1, $3
20:  par, array[$3], R, -
21:  call, -, -, swap
22:  :=, 1, -, changed
23:  +, i, 1, $4
24:  :=, $4, -, i
25:  jump, -, -, 13
26:  jump, -, -, 8
27:  endu, BubbleSort, -, -

28:  unit, PrintArray, -, -
29:  par, msg, V, -
30:  call, -, -, put_string
31:  :=, 0, -, i
32:  <, i, size, 34
33:  jump, -, -, 43
34:  >, i, 0, 36
35:  jump, -, -, 38
36:  par, ", ", V, -
37:  call, -, -, put_string
38:  par, array[i], V, -
39:  call, -, -, put_int
40:  +, i, 1, $5
41:  :=, $5, -, i
42:  jump, -, -, 32
43:  par, "\n", V, -
44:  call, -, -, put_string
45:  endu, PrintArray, -, -

46:  unit, main, -, -
47:  :=, 0, -, i
48:  :=, 65, -, seed
49:  <, i, 16, 51
50:  jump, -, -, 60
51:  *, seed, 137, $6
52:  +, $6, 221, $7
```

```
53:  +, $7, i, $8
54:  %, $8, 101, $9
55:  :=, $9, -, seed
56:  :=, seed, -, x[i]
57:  +, i, 1, $10
58:  :=, $10, -, i
59:  jump, -, -, 49
60:  par, "Initial array: ", V, -
61:  par, x, V, -
62:  par, 16, V, -
63:  call, -, -, PrintArray
64:  par, x, R, -
65:  par, 16, V, -
66:  call, -, -, BubbleSort
67:  par, "Sorted array: ", V, -
68:  par, x, V, -
69:  par, 16, V, -
70:  call, -, -, PrintArray
71:  endu, main, -, -
```

## 5. Στάδια της Εργασίας και Βαθμολογία

Τα στάδια της εργασίας, τα αντίστοιχα τμήματα του μεταγλωττιστή, η κατανομή των μονάδων (6 από 10 που αντιστοιχούν στο μάθημα), και η ημερομηνία παράδοσης του κάθε σταδίου φαίνονται στον παρακάτω πίνακα.

| **Στάδιο** | **Τμήμα του μεταγλωττιστή** | **Μονάδες** | **Ημερομηνία Παράδοσης** |
|---|---|---|---|
| 1 | Υλοποίηση λεκτικού αναλυτή με flex | 0.5 | 15/03/2010 |
| 2 | Υλοποίηση συντακτικού αναλυτή με bison | 1 | 12/04/2010 |
| 3 | Υλοποίηση πίνακα συμβόλων και σημασιολογικής ανάλυσης | 2.5 | 03/05/2010 |
| 4 | Παραγωγή ενδιάμεσου κώδικα | 2 | 24/05/2010 |
| 5 | Παραγωγή τελικού κώδικα | 2 (bonus) | |
| | **Συνολική εργασία** | 6 (max 8) | |

**Παρατηρήσεις:**

1. Η εργασία θα εκπονηθεί από ομάδες φοιτητών που θα αποτελούνται από το **πολύ 2 άτομα.**

2. **Η εξέταση της εργασίας και η βαθμολογία θα είναι ατομική!** Η εξέτασης της εργασίας θα γίνει λίγες μέρες μετά την **24/05/2010** (λεπτομέρειες θα δοθούν εν καιρώ).

3. Ο κώδικας που θα παραδώσετε για κάθε στάδιο θα πρέπει να περιέχει και τον κώδικα για όλα τα προηγούμενα στάδια. Στις **24/05/2010** εκτός από τον κώδικα θα πρέπει να παραδώσετε και μια **αναλυτική έκθεση** ανά ομάδα η οποία θα καλύπτει την εργασία **στο σύνολο της** και θα αναλύει τις τεχνικές που χρησιμοποιήθηκαν. Οι εκθέσεις σας **δεν θα γίνουν δεκτές** αν είναι γραμμένες με το χέρι (θα πρέπει να χρησιμοποιήσετε κάποιο κειμενογράφο π.χ. Microsoft Word, LaTeX κλπ.). Το περιεχόμενο της έκθεσης περιγράφεται παρακάτω (Κεφ. 6.1).

4. Εκπρόθεσμες εργασίες δεν θα γίνονται δεκτές. Αν δεν παραδώσετε κάποιο στάδιο της εργασίας, χάνετε όλους τους βαθμούς που αναλογούν σ' αυτό το στάδιο.

5. Η υλοποίηση της οδηγίας συμπερίληψης αρχείων `#include` είναι προαιρετική.

## *5.1 Περιεχόμενα Έκθεσης*

Κατά την εξέταση της εργασίας, κάθε ομάδα θα πρέπει να παραδώσει μια έκθεση (10-15 σελίδες + κώδικας) που καλύπτει όλα τα λειτουργικά τμήματα του μεταγλωττιστή που υλοποιήθηκε:

1. **Λεκτικός Αναλυτής.** Να περιγράψετε προσεκτικά την επικοινωνία του λεκτικού αναλυτή με το συντακτικό αναλυτή και να δώσετε τον κώδικα σε flex που υλοποιεί τον λεκτικό αναλυτή.

2. **Συντακτικός αναλυτής.** Γι αυτό το τμήμα του μεταγλωττιστή πρέπει να κάνετε τα εξής:
   - Να δώσετε τη γραμματική της Robin σε μορφή γραμματικής χωρίς συμφραζόμενα.
   - Να εξηγήσετε ποιες συγκρούσεις μετάθεσης/αναγωγής ή αναγωγής/αναγωγής υπάρχουν και πως τις αποφεύγετε στην υλοποίηση σας με bison.
   - Να δώσετε τον κώδικα σε bison που υλοποιεί την συντακτική ανάλυση.

3. **Πίνακας συμβόλων.** Να περιγράψετε προσεκτικά τι πληροφορίες αποθηκεύονται στον πίνακα συμβόλων για κάθε είδος ονόματος της Robin, ποια μέθοδος υλοποίησης χρησιμοποιείται, και πως αντιμετωπίζεται το πρόβλημα της εμβέλειας ονομάτων.

4. **Δηλώσεις ονομάτων.** Να περιγράψετε προσεκτικά τι κάνει ο μεταγλωττιστής σας για κάθε περίπτωση δήλωσης ονόματος και να δώσετε το σχετικό κώδικα σε bison.

5. **Ενδιάμεσος κώδικας για τις υπόλοιπες εντολές.** Να περιγράψετε προσεκτικά τον ενδιάμεσο κώδικα που δημιουργεί ο μεταγλωττιστής σας για κάθε άλλη εντολή και να δώσετε τον σχετικό κώδικα σε bison.

6. **Έλεγχος ορθότητας προγραμμάτων και μηνύματα λάθους.** Να περιγράψετε αναλυτικά τα λάθη που εντοπίζονται από τον μεταγλωττιστή σας και τις τεχνικές που χρησιμοποιείτε για να τα αντιμετωπίσετε. Αυτή η συζήτηση είναι δυνατόν να ενσωματωθεί στην περιγραφή των διαφόρων λειτουργικών τμημάτων του μεταγλωττιστή.

7. **Παραδείγματα.** Θα πρέπει επίσης να δώσετε τουλάχιστον 1 πρόγραμμα της Robin που έχετε χρησιμοποιήσει σαν test για κάθε χαρακτηριστικό της γλώσσας που ελέγχετε, καθώς και την έξοδο που δίνει ο μεταγλωττιστής σας γι' αυτά τα προγράμματα. Οι έλεγχοι θα πρέπει να συμπεριλαμβάνουν και περιπτώσεις λαθών και πώς ο μεταγλωττιστής σας τις αντιμετωπίζει.

Αν δεν έχετε υλοποιήσει κάποιο μέρος του μεταγλωττιστή, θα πρέπει να το εξηγήσετε καθαρά στην έκθεση σας. Αν δεν το κάνετε αυτό, θα χάσετε περισσότερες μονάδες κατά την βαθμολόγηση.

## 6. Επίλογος

Στη διάρκεια του εξαμήνου θα δοθούν και άλλες διευκρινίσεις και αναλυτικά παραδείγματα μεταγλώττισης. Να παρακολουθείτε τις παραδόσεις, τις ασκήσεις και την ιστοσελίδα του μαθήματος. Καλή επιτυχία!