

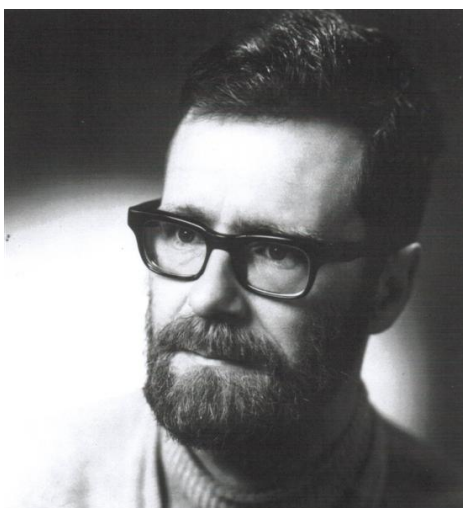


Εθνικό Μετσόβιο Πολυτεχνείο  
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών  
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών  
Εργαστήριο Τεχνολογίας Λογισμικού

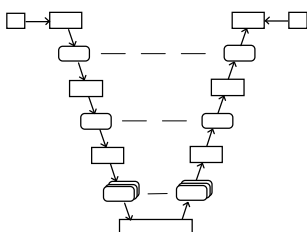
Μεταγλωττιστές 2022

Θέμα εργασίας

# Η γλώσσα Edsger



Edsger Wybe Dijkstra (1930–2002)



**Μεταγλωττιστές**

<https://courses.softlab.ntua.gr/compilers/>

Διδάσκων: Κωστής Σαγώνας

Αθήνα, Μάρτιος 2022



## ΘΕΜΑ:

Να σχεδιαστεί και να υλοποιηθεί από κάθε ομάδα, το πολύ δυο σπουδαστών, ένας μεταγλωττιστής για τη γλώσσα Edsger. Γλώσσα υλοποίησης μπορεί να είναι μια από τις C/C++, Java, SML, OCaml, Haskell ή Python, αλλά έχετε υπ' όψη σας ότι κομμάτια του μεταγλωττιστή και εργαλεία που μπορεί να σας φανούν χρήσιμα μπορεί να μην είναι διαθέσιμα σε κάποιες από τις παραπάνω γλώσσες και σε αυτήν την περίπτωση θα πρέπει να τα υλοποιήσετε μόνοι σας. Η επιλογή κάποιας άλλης γλώσσας υλοποίησης μπορεί να γίνει κατόπιν συνεννόησης με τον διδάσκοντα. Επιτρέπεται να χρησιμοποιηθούν και επίσης συνιστάται η χρήση εργαλείων, π.χ. flex/ML-Lex/ocamllexA1ex, bison/ML-Yacc/ocamlyaccHappy, JavaCC, κ.λπ., όπως και το LLVM. Περισσότερες πληροφορίες σχετικές με αυτά τα εργαλεία θα δοθούν στις παραδόσεις.

### Παραδοτέα, ημερομηνίες και βαθμολόγηση

Τα τμήματα του μεταγλωττιστή φαίνονται στον παρακάτω πίνακα. Οι ημερομηνίες παράδοσης θα ανακοινωθούν στη σελίδα του μαθήματος.

Τμήμα του μεταγλωττιστή	Μονάδες	Bonus
Λεκτικός αναλυτής	0.5	—
Συντακτικός αναλυτής	0.5	—
Σημασιολογικός αναλυτής	1.0	0.5
Ενδιάμεσος κώδικας	1.0	—
Βελτιστοποίηση	1.0	0.8
Τελικός κώδικας	1.0	0.7
Συνολική εργασία και έκθεση	5.0	2.0

Για τα διάφορα τμήματα της εργασίας πρέπει να παραδίδεται εμπρόθεσμα από κάθε ομάδα ο αντίστοιχος κώδικας σε ηλεκτρονική μορφή, καθώς και σαφείς οδηγίες για την παραγωγή ενός εκτελέσιμου προγράμματος επίδειξης της λειτουργίας του αντίστοιχου τμήματος, από τον κώδικα αυτόν. Καθυστερημένες ασκήσεις θα βαθμολογούνται με μικρότερο βαθμό, αντιστρόφως ανάλογα προς το χρόνο καθυστέρησης. Παρακαλούμε, **μην παραδίδετε τυπωμένες εργασίες!** Η μορφή και τα περιεχόμενα των παραδοτέων, συμπεριλαμβανομένης και της τελικής έκθεσης, πρέπει να συμφωνούν με τις οδηγίες που δίνονται στην ενότητα 4 του παρόντος.



### Προαιρετικά τμήματα και μονάδες bonus

Το σύνολο των μονάδων της παρούσας εργασίας είναι 7. Από αυτές, οι 2 μονάδες είναι bonus (που σημαίνει ότι το σύνολο μονάδων του μαθήματος είναι 12) και αντιστοιχούν στα παρακάτω τμήματα της εργασίας, που είναι προαιρετικά:

- Πραγματικοί αριθμοί (double) και μετατροπές τύπων (type casting).
- Βελτιστοποίηση ενδιάμεσου κώδικα, με ανάλυση ροής δεδομένων και ελέγχου.
- Δέσμευση καταχωρητών και βελτιστοποίηση τελικού κώδικα.



# Περιεχόμενα

<b>1</b>	<b>Περιγραφή της γλώσσας Edsger</b>	<b>5</b>
1.1	Λεκτικές μονάδες	5
1.2	Τύποι δεδομένων	6
1.3	Δομή του προγράμματος	7
1.3.1	Μεταβλητές	7
1.3.2	Συναρτήσεις	8
1.4	Εκφράσεις	8
1.4.1	L-values	8
1.4.2	Σταθερές	9
1.4.3	Τελεστές	9
1.4.4	Κλήση συναρτήσεων	11
1.4.5	Δυναμική διαχείριση μνήμης	12
1.5	Εντολές	12
1.6	Οδηγίες προς το μεταγλωττιστή	12
1.7	Βιβλιοθήκη έτοιμων συναρτήσεων	13
1.7.1	Είσοδος και έξοδος — #include “stdio.h”	13
1.7.2	Μαθηματικές συναρτήσεις — #include “math.h”	13
1.7.3	Συναρτήσεις μετατροπής — #include “stdlib.h”	14
1.7.4	Συναρτήσεις διαχείρισης συμβολοσειρών — #include “string.h”	14
<b>2</b>	<b>Πλήρης γραμματική της Edsger</b>	<b>15</b>
<b>3</b>	<b>Παραδείγματα</b>	<b>16</b>
3.1	Πες γεια!	16
3.2	Οι πύργοι του Hanoi	16
3.3	Πρώτοι αριθμοί	17
3.4	Αντιστροφή συμβολοσειράς	19
3.5	Ταξινόμηση με τη μέθοδο της φουσαλίδας	19
3.6	Μέση τιμή τυχαίας μεταβλητής	20
<b>4</b>	<b>Οδηγίες για την παράδοση</b>	<b>22</b>

# 1 Περιγραφή της γλώσσας Edsger

Η γλώσσα Edsger είναι μια απλή γλώσσα προστακτικού προγραμματισμού. Βασίζεται σε ένα γνήσιο υποσύνολο της ISO C99, είναι όμως εμπλουτισμένη με μερικές μικρές παραλλαγές. Τα κύρια χαρακτηριστικά της εν συντομία είναι τα εξής:

- Σύνταξη παρόμοια σε γενικές γραμμές με αυτή της C.
- Δομημένες συναρτήσεις με τους κανόνες εμβέλειας της Pascal.
- Βασικοί τύποι δεδομένων για ακέραιους αριθμούς, χαρακτήρες, λογικές τιμές και πραγματικούς αριθμούς.
- Τύποι δεικτών και (έμμεσα) πίνακες, συνδεδεμένοι σημασιολογικά όπως στη C.
- Βιβλιοθήκη συναρτήσεων.

Περισσότερες λεπτομέρειες της γλώσσας δίνονται στις παραγράφους που ακολουθούν. Λόγω των πολλών ομοιοτήτων της Edsger με τη γλώσσα C, τόσο από πλευράς σύνταξης όσο και από πλευράς σημασιολογίας, η περιγραφή θα είναι σύντομη με εξαίρεση τα σημεία όπου οι δυο γλώσσες διαφέρουν.

## 1.1 Λεκτικές μονάδες

Οι λεκτικές μονάδες της γλώσσας Edsger χωρίζονται στις παρακάτω κατηγορίες:

- Τις λέξεις κλειδιά, οι οποίες είναι οι παρακάτω:

bool	break	byref	char	continue	delete
double	else	for	false	if	int
new	NULL	return	true	void	

- Τα ονόματα, τα οποία αποτελούνται από ένα πεζό ή κεφαλαίο γράμμα του λατινικού αλφαβήτου, πιθανώς ακολουθούμενο από μια σειρά πεζών ή κεφαλαίων γραμμάτων, δεκαδικών ψηφίων ή χαρακτήρων υπογράμμισης (*underscore*). Τα ονόματα δεν πρέπει να συμπίπτουν με τις λέξεις κλειδιά που αναφέρθηκαν παραπάνω.
- Οι *ακέραιες σταθερές* χωρίς πρόσημο, που αποτελούνται από ένα ή περισσότερα δεκαδικά ψηφία. Παραδείγματα ακέραιων σταθερών είναι τα ακόλουθα:

0      42      1284      00200

- Οι *πραγματικές σταθερές* χωρίς πρόσημο, 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 *σταθεροί χαρακτήρες*, που αποτελούνται από ένα χαρακτήρα μέσα σε απλά εισαγωγικά. Ο χαρακτήρας αυτός μπορεί να είναι οποιοσδήποτε κοινός χαρακτήρας ή ακολουθία διαφυγής (*escape sequence*). Κοινοί χαρακτήρες είναι όλοι οι εκτυπώσιμοι χαρακτήρες πλην των απλών και διπλών εισαγωγικών και του χαρακτήρα `\` (backslash). Οι ακολουθίες διαφυγής ξεκινούν με το χαρακτήρα `\` (backslash) και περιγράφονται στον πίνακα 1. Παραδείγματα σταθερών χαρακτήρων είναι τα ακόλουθα:

'a'      '1'      '\n'      '\"'      '\x1d'

Πίνακας 1: Ακολουθίες διαφυγής (escape sequences).

Ακολουθία διαφυγής	Περιγραφή
<code>\n</code>	ο χαρακτήρας αλλαγής γραμμής (line feed)
<code>\t</code>	ο χαρακτήρας στηλοθέτησης (tab)
<code>\r</code>	ο χαρακτήρας επιστροφής στην αρχή της γραμμής (carriage return)
<code>\0</code>	ο χαρακτήρας με ASCII κωδικό 0
<code>\\</code>	ο χαρακτήρας <code>\</code> (backslash)
<code>\'</code>	ο χαρακτήρας <code>'</code> (απλό εισαγωγικό)
<code>\"</code>	ο χαρακτήρας <code>"</code> (διπλό εισαγωγικό)
<code>\xnn</code>	ο χαρακτήρας με ASCII κωδικό <code>nn</code> στο δεκαεξαδικό σύστημα

- Οι σταθερές συμβολοσειρές, που αποτελούνται από μια ακολουθία κοινών χαρακτήρων ή ακολουθιών διαφυγής μέσα σε διπλά εισαγωγικά. Οι συμβολοσειρές δεν μπορούν να εκτείνονται σε περισσότερες από μια γραμμές προγράμματος. Παραδείγματα σταθερών συμβολοσειρών είναι οι ακόλουθες:

```
"abc"           "Route66"           "Helloworld!\n"
"Name:\t\"DouglasAdams\""\nValue:\t42\n"
```

- Τους συμβολικούς τελεστές, οι οποίοι είναι οι παρακάτω:

```
=      ==      !=      >      <      >=      <=
+      -      *      /      %      &      !
&&     ||      ?      :      ,      ++      --
+=      -=      *=      /=      %=
```

- Τους διαχωριστές, οι οποίοι είναι οι παρακάτω:

```
;      (      )      [      ]      {      }
```

Εκτός από τις λεκτικές μονάδες που προαναφέρθηκαν, ένα πρόγραμμα Edsger μπορεί επίσης να περιέχει τα παρακάτω, τα οποία διαχωρίζουν λεκτικές μονάδες και αγνοούνται:

- *Κενούς χαρακτήρες*, δηλαδή ακολουθίες αποτελούμενες από κενά διαστήματα (space), χαρακτήρες στηλοθέτησης (tab), χαρακτήρες αλλαγής γραμμής (line feed) ή χαρακτήρες επιστροφής στην αρχή της γραμμής (carriage return).
- *Σχόλια μιας γραμμής*, τα οποία αρχίζουν με την ακολουθία χαρακτήρων `//` και τερματίζονται με το τέλος της τρέχουσας γραμμής.
- *Σχόλια πολλών γραμμών*, τα οποία αρχίζουν με την ακολουθία χαρακτήρων `/*` και τερματίζονται με την πρώτη μετέπειτα εμφάνιση της ακολουθίας χαρακτήρων `*/`. Κατά συνέπεια, τα σχόλια δεν επιτρέπεται να είναι φωλιασμένα. Στο εσωτερικό τους επιτρέπεται η εμφάνιση οποιουδήποτε χαρακτήρα.

## 1.2 Τύποι δεδομένων

Η Edsger υποστηρίζει τέσσερις βασικούς τύπους δεδομένων:

- `int` : ακέραιοι αριθμοί,
- `char` : χαρακτήρες,

- `bool` : λογικές τιμές, και
- `double` : πραγματικοί αριθμοί.

Οι τύποι `int` και `double` ονομάζονται *αριθμητικοί* τύποι.

Εκτός από τους βασικούς τύπους, η Edsger υποστηρίζει επίσης τύπους δεικτών, που συμβολίζονται με  $t^*$ , όπου ο  $t$  πρέπει να είναι έγκυρος τύπος. Κάθε δείκτης τύπου  $t^*$  μπορεί να χρησιμοποιηθεί ως ένας μονοδιάστατος πίνακας, αποτελούμενος από στοιχεία τύπου  $t$ . Στην περίπτωση αυτή, η αρίθμηση των στοιχείων του πίνακα ακολουθεί τη σύμβαση της C: το πρώτο στοιχείο βρίσκεται στη θέση 0, το δεύτερο στη θέση 1, ενώ το τελευταίο βρίσκεται σε θέση κατά ένα μικρότερη από το πλήθος των στοιχείων του πίνακα.

Ο αριθμός των bytes που καταλαμβάνουν τα δεδομένα κάθε τύπου στη μνήμη του υπολογιστή, καθώς και ο τρόπος παράστασης των δεδομένων περιγράφεται παρακάτω. Η περιγραφή διευκολύνει την υλοποίηση μεταγλωττιστών για υπολογιστές βασισμένους σε επεξεργαστές της σειράς x86 της Intel, με το μοντέλο προγραμμάτων COM.

- `int` : μέγεθος 2 bytes, παράσταση συμπληρώματος ως προς 2.
- `char` : μέγεθος 1 byte, παράσταση σύμφωνα με τον πίνακα ASCII.
- `bool` : μέγεθος 1 byte, τιμή 0 για `false` και  $\neq 0$  για `true`.
- `double` : μέγεθος 10 bytes, παράσταση σύμφωνα με το πρότυπο IEEE 754.
- $t^*$  : μέγεθος 2 bytes.

### 1.3 Δομή του προγράμματος

Ένα πρόγραμμα Edsger αποτελείται από μια ή περισσότερες δηλώσεις. Οι δηλώσεις ανήκουν σε τρεις κατηγορίες:

- Δηλώσεις μεταβλητών (variable declarations),
- Ορισμοί συναρτήσεων (function definitions).
- Δηλώσεις συναρτήσεων (function declarations), οι ορισμοί των οποίων θα ακολουθήσουν.

Σε κάθε πρόγραμμα πρέπει να υπάρχει ο ορισμός μιας συνάρτησης με επικεφαλίδα:

```
void main ()
```

από την οποία ξεκινά η εκτέλεση του προγράμματος.

#### 1.3.1 Μεταβλητές

Οι δηλώσεις μεταβλητών γίνονται με αναγραφή του τύπου, ακολουθούμενου από έναν ή περισσότερους *δηλωτές* (declarators) μεταβλητών. Η απλούστερη μορφή δηλωτή είναι ένα αναγνωριστικό (identifier), που αντιστοιχεί στο όνομα της μεταβλητής. Παραδείγματα τέτοιων δηλώσεων είναι:

```
int i;
double x, y;
int * p, q;
```

**Ας σημειωθεί ότι, αντίθετα με τη γλώσσα C, στην τελευταία δήλωση και οι δύο μεταβλητές `p` και `q` έχουν τύπο `int*`.**

Για τη διευκόλυνση της δήλωσης μεταβλητών τύπου δείκτη που χρησιμοποιούνται ως μονοδιάστατοι πίνακες, η γλώσσα Edsger υποστηρίζει ένα δεύτερο τύπο δηλωτή που αποτελείται από ένα αναγνωριστικό και την επιθυμητή διάσταση του πίνακα μέσα σε αγκύλες. Παραδείγματα τέτοιων δηλώσεων είναι:

```
int a[100];
double x[5], y[7];
int * p[12];
```

Στις παραπάνω δηλώσεις, η μεταβλητή *a* έχει τύπο `int*` (δείκτης σε ακέραιο, δηλαδή ισοδύναμα μονοδιάστατος πίνακας ακεραίων), οι μεταβλητές *x* και *y* έχουν τύπο `double*` ενώ η μεταβλητή *p* έχει τύπο `int**` (δείκτης σε δείκτη σε ακέραιο, δηλαδή ισοδύναμα μονοδιάστατος πίνακας δεικτών σε ακέραιο).

Οι μεταβλητές τύπου δείκτη που δηλώνονται με δηλωτές της δεύτερης μορφής ονομάζονται **αμετάβλητες** (immutable) και η τιμή τους δεν μπορεί να μεταβληθεί κατά την εκτέλεση του προγράμματος. Για παράδειγμα, το παρακάτω είναι σφάλμα:

```
int a[100], b[200];

a = b; // wrong!
```

### 1.3.2 Συναρτήσεις

Κάθε συνάρτηση είναι μια δομική μονάδα και αποτελείται από την επικεφαλίδα της και το σώμα της. Στην επικεφαλίδα αναφέρεται ο τύπος επιστροφής της συνάρτησης, το όνομά της και οι τυπικές της παράμετροι μέσα σε παρενθέσεις. Οι παρενθέσεις είναι υποχρεωτικές ακόμα και για συναρτήσεις που δεν έχουν τυπικές παραμέτρους. Ο τύπος αποτελέσματος `void` υποδηλώνει ότι μια συνάρτηση δεν επιστρέφει αποτέλεσμα.

Κάθε τυπική παράμετρος χαρακτηρίζεται από το όνομά της, τον τύπο της και τον τρόπο περάσματος. Σε αντίθεση με τη C, η γλώσσα Edsger υποστηρίζει **πέρασμα παραμέτρων κατ' αξία (by value) και κατ' αναφορά (by reference)**. Αν στη δήλωση μιας τυπικής παραμέτρου έχει προηγηθεί η λέξη κλειδί `byref`, τότε αυτή περνά κατ' αναφορά, διαφορετικά περνά κατ' αξία.

Ακολουθούν παραδείγματα επικεφαλίδων συναρτήσεων.

```
void p1 ()
void p2 (int n)
void p3 (int a, int b, byref bool b)
double f1 (double x)
int f2 (byref char * s)
double * f3 (double x)
```

Το σώμα μιας συνάρτησης περικλείεται μέσα σε άγκιστρα { και }. Μπορεί να περιέχει δηλώσεις, ακολουθούμενες από εντολές. Η Edsger ακολουθεί τους **κανόνες εμφάνισης της Pascal**, όσον αφορά στην ορατότητα των ονομάτων μεταβλητών, συναρτήσεων και παραμέτρων.

## 1.4 Εκφράσεις

Κάθε έκφραση της Edsger διαθέτει ένα μοναδικό τύπο<sup>1</sup> και μπορεί να αποτιμηθεί δίνοντας ως αποτέλεσμα μια τιμή αυτού του τύπου. Οι εκφράσεις διακρίνονται σε δύο κατηγορίες: αυτές που δίνουν l-values, οι οποίες περιγράφονται στην ενότητα 1.4.1, και αυτές που δίνουν r-values, που περιγράφονται στις ενότητες 1.4.2 ως 1.4.4. Τα δυο αυτά είδη τιμών έχουν πάρει το όνομά τους από τη θέση τους σε μια εντολή ανάθεσης: οι l-values εμφανίζονται στο αριστερό μέλος της ανάθεσης ενώ οι r-values στο δεξιό.

Τόσο οι l-values όσο και οι r-values μπορούν να εμφανίζονται μέσα σε παρενθέσεις, που χρησιμοποιούνται για λόγους ομαδοποίησης.

### 1.4.1 L-values

Οι l-values αντιπροσωπεύουν αντικείμενα που καταλαμβάνουν χώρο στη μνήμη του υπολογιστή κατά την εκτέλεση του προγράμματος και τα οποία μπορούν να περιέχουν τιμές. Τέτοια αντικείμενα είναι οι μεταβλητές, οι παράμετροι των συναρτήσεων και οι μεταβλητές που κατασκευάζονται με δυναμική παραχώρηση μνήμης. Συγκεκριμένα:

<sup>1</sup>Εξάιρεση αποτελεί η σταθερά μηδενικού δείκτη NULL, η οποία δεν έχει μοναδικό τύπο. Περιγράφεται στην ενότητα 1.4.2.



- Το όνομα μιας μη μόνιμης μεταβλητής ή μιας παραμέτρου συνάρτησης είναι l-value και αντιστοιχεί στο εν λόγω αντικείμενο. Ο τύπος της l-value είναι ο τύπος του αντίστοιχου αντικειμένου.
- Αν  $p$  είναι μια έκφραση τύπου  $t^*$ , τότε  $*p$  είναι μια l-value με τύπο  $t$ , που αντιστοιχεί στο αντικείμενο όπου δείχνει ο δείκτης που παριστάνει η τιμή της  $p$ .
- Αν  $p$  είναι μια έκφραση τύπου  $t^*$  και  $e$  είναι μια έκφραση τύπου `int`, τότε  $p[e]$  είναι μια l-value με τύπο  $t$ . Αυτή η l-value είναι ισοδύναμη με την l-value  $*(p+e)$ , όπως αυτή περιγράφεται στην ενότητα 1.4.3.

Αν μια l-value χρησιμοποιηθεί ως έκφραση, η τιμή αυτής της έκφρασης είναι ίση με την τιμή που περιέχεται στο αντικείμενο που αντιστοιχεί στην l-value.

### 1.4.2 Σταθερές

Στις r-values της γλώσσας Edsger συγκαταλέγονται οι ακόλουθες σταθερές:

- Οι ακέραιες σταθερές χωρίς πρόσημο, όπως περιγράφονται στην ενότητα 1.1. Έχουν τύπο `int` και η τιμή τους είναι ίση με τον μη αρνητικό ακέραιο αριθμό που παριστάνουν.
- Οι λέξεις κλειδιά `true` και `false`, με τύπο `bool` και προφανείς τιμές.
- Οι πραγματικές σταθερές χωρίς πρόσημο, όπως περιγράφονται στην ενότητα 1.1. Έχουν τύπο `double` και η τιμή τους είναι ίση με τον μη αρνητικό πραγματικό αριθμό που παριστάνουν.
- Οι σταθεροί χαρακτήρες, όπως περιγράφονται στην ενότητα 1.1. Έχουν τύπο `char` και η τιμή τους είναι ίση με το χαρακτήρα που παριστάνουν.
- Η λέξη κλειδί `NULL` που έχει τύπο  $t^*$ , για κάθε έγκυρο τύπο  $t$ , και η τιμή της οποίας είναι ο μηδενικός δείκτης. Είναι η μόνη έκφραση της Edsger που δεν έχει μοναδικό τύπο. Ο μηδενικός δείκτης απαγορεύεται να αποδεικτοδοτηθεί με χρήση του τελεστή  $*$ .
- Οι σταθερές συμβολοσειρές, όπως περιγράφονται στην ενότητα 1.1. Έχουν τύπο `char*`. Κάθε τέτοια r-value είναι ένας δείκτης σε έναν πίνακα αντικείμενων τύπου `char`, όπου βρίσκονται αποθηκευμένοι με τη σειρά οι χαρακτήρες της συμβολοσειράς. Στο τέλος του πίνακα αποθηκεύεται αυτόματα ο χαρακτήρας `'\0'`, σύμφωνα με τη σύμβαση που ακολουθεί η γλώσσα C για τις συμβολοσειρές.

### 1.4.3 Τελεστές

Οι τελεστές της Edsger διακρίνονται σε τελεστές με ένα, δύο και τρία τελούμενα. Από τους τελεστές με ένα τελούμενο, ορισμένοι γράφονται πριν το τελούμενο (prefix) και ορισμένοι μετά (postfix). Οι τελεστές με δύο τελούμενα γράφονται πάντα μεταξύ των τελουμένων (infix) και η αποτίμηση των τελουμένων γίνεται από αριστερά προς τα δεξιά. Ο μοναδικός τελεστής με τρία τελούμενα είναι ο `?:` και έχει ειδική σύνταξη και σημασιολογία, όπως θα εξηγηθεί παρακάτω.

Ήδη στην ενότητα 1.4.1 περιγράφηκε ο τελεστής αναφοράς σε στοιχείο πίνακα, η σύνταξη του οποίου αποτελεί εξαίρεση στα παραπάνω, και ο τελεστής αποδεικτοδότησης  $*$ . Οι δύο αυτοί τελεστές είναι οι μοναδικοί που έχουν ως αποτέλεσμα l-value. Στη συνέχεια περιγράφονται οι υπόλοιποι τελεστές της Edsger, που έχουν ως αποτέλεσμα r-value. Οι τελεστές αυτοί χωρίζονται σε δύο κατηγορίες: τους τελεστές υπολογισμού και τους τελεστές ανάθεσης. Οι πρώτοι χρησιμοποιούνται για τον υπολογισμό τιμών, ενώ οι δεύτεροι για την ανάθεση τιμών σε μεταβλητές. Η αποτίμηση μιας έκφρασης που περιέχει έναν ή περισσότερους τελεστές ανάθεσης, όπως στη C, εκτός από το να υπολογίζει μια τιμή αποτελέσματος έχει ως παρενέργεια τη μεταβολή των τιμών κάποιων μεταβλητών.

Οι τελεστές υπολογισμού είναι οι ακόλουθοι:

- Ο τελεστής & επιστρέφει τη διεύθυνση ενός αντικειμένου. Αν  $l$  είναι μια l-value τύπου  $t$ , τότε  $\&l$  είναι μια r-value τύπου  $t^*$ . Η τιμή της είναι η διεύθυνση του αντικειμένου που αντιστοιχεί στην  $l$  και είναι πάντα διαφορετική του μηδενικού δείκτη.
- Οι τελεστές με ένα τελούμενο  $+$  και  $-$  υλοποιούν τους τελεστές προσήμου. Το τελούμενο πρέπει να είναι αριθμητικού τύπου και το αποτέλεσμα είναι του ίδιου τύπου με το τελούμενο.
- Ο τελεστής  $!$  υλοποιεί τη λογική άρνηση. Το τελούμενό του πρέπει να είναι τύπου `bool`, και τον ίδιο τύπο έχει και το αποτέλεσμα.
- Οι τελεστές με δύο τελούμενα  $+$ ,  $-$ ,  $*$ ,  $/$  και  $\%$  υλοποιούν τις αριθμητικές πράξεις. Τα επιτρεπτά τελούμενα αυτών των τελεστών περιγράφονται παρακάτω:
  - Αν τα τελούμενα των  $+$ ,  $-$ ,  $*$ ,  $/$  και  $\%$  είναι τύπου `int` τότε και το αποτέλεσμα είναι τύπου `int`. Στην περίπτωση αυτή, οι τελεστές  $/$  και  $\%$  υλοποιούν την ακέραια διαίρεση.
  - Αν τα τελούμενα των  $+$ ,  $-$ ,  $*$  και  $/$  είναι τύπου `double` τότε και το αποτέλεσμα είναι τύπου `double`. Στην περίπτωση αυτή, ο τελεστής  $/$  υλοποιεί την πραγματική διαίρεση.
  - Αν το πρώτο τελούμενο των  $+$  και  $-$  είναι τύπου  $t^*$  και το δεύτερο τελούμενο είναι τύπου `int`, τότε το αποτέλεσμα είναι τύπου  $t^*$ . Η τιμή του αποτελέσματος είναι ένας δείκτης μετατοπισμένος ως προς την τιμή του πρώτου τελούμενου κατά τόσες θέσεις αντικειμένων τύπου  $t$  όσες ορίζει η τιμή του δεύτερου τελούμενου (pointer arithmetic).
- Οι τελεστές `==` και `!=` υλοποιούν αντίστοιχα την ισότητα και την ανισότητα. Το αποτέλεσμα είναι τύπου `bool`. Τα τελούμενα πρέπει να είναι του ίδιου τύπου.
- Οι τελεστές `<`, `>`, `<=` και `>=` υλοποιούν τις σχέσεις ανισότητας μεταξύ αριθμών ή δεικτών. Το αποτέλεσμα είναι τύπου `bool`. Τα τελούμενα πρέπει να είναι του ίδιου τύπου. Αν είναι αριθμητικά, η σύγκριση γίνεται βάσει των αριθμητικών τους τιμών. Αν είναι τύπου `bool`, η σύγκριση γίνεται θεωρώντας ότι `false < true`. Τέλος, αν είναι δείκτες, η σύγκριση έχει νόημα μόνο όταν τα τελούμενα περιέχουν διευθύνσεις αντικειμένων που βρίσκονται στον ίδιο μονοδιάστατο πίνακα και γίνεται βάσει της σχετικής θέσης αυτών των αντικειμένων μέσα στον πίνακα.
- Οι τελεστές `&&` και `||` υλοποιούν αντίστοιχα τις πράξεις της λογικής σύζευξης και διάζευξης. Τα τελούμενα πρέπει να είναι τύπου `bool` και τον ίδιο τύπο έχει και το αποτέλεσμα. Η αποτίμηση εκφράσεων που χρησιμοποιούν αυτούς τους τελεστές γίνεται με *βραχυκύκλωση* (short-circuit). Δηλαδή, αν το αποτέλεσμα της έκφρασης είναι γνωστό από την αποτίμηση και μόνο του πρώτου τελούμενου, το δεύτερο τελούμενο δεν αποτιμάται καθόλου.
- Ο τελεστής **, (κόμμα)** υπολογίζει τις τιμές των δύο τελουμένων και επιστρέφει την τιμή του δεύτερου, αγνοώντας αυτήν του πρώτου. Τα δύο τελούμενα μπορούν να είναι οποιουδήποτε έγκυρου τύπου. Το αποτέλεσμα είναι του ίδιου τύπου με αυτόν του δεύτερου τελούμενου.
- Ο τελεστής με τρία τελούμενα  $e_1 ? e_2 : e_3$  έχει την ειδική σύνταξη  $e_1 ? e_2 : e_3$ . Το πρώτο τελούμενο πρέπει να είναι τύπου `bool`, ενώ τα άλλα δύο τελούμενα πρέπει να είναι του ίδιου τύπου. Το αποτέλεσμα έχει τον ίδιο τύπο με τα δύο τελευταία τελούμενα. Η αποτίμηση της έκφρασης  $e_1 ? e_2 : e_3$  γίνεται ξεκινώντας από την αποτίμηση της  $e_1$ . Αν η τιμή αυτής είναι `true`, τότε αποτιμάται η  $e_2$  και η τιμή αυτής είναι το αποτέλεσμα. Διαφορετικά, αποτιμάται η  $e_3$  και η τιμή αυτής είναι το αποτέλεσμα. Σε κάθε περίπτωση, αποτιμάται μόνο ένα από τα τελούμενα  $e_2$  και  $e_3$ .

Οι τελεστές ανάθεσης είναι οι ακόλουθοι:

- Ο τελεστής `=` αναθέτει την τιμή του δεύτερου τελουμένου στη μεταβλητή που αντιστοιχεί στο πρώτο τελούμενο. Το πρώτο τελούμενο πρέπει να είναι μια l-value οποιουδήποτε έγκυρου τύπου  $t$ , ενώ το δεύτερο τελούμενο πρέπει να είναι του ίδιου τύπου  $t$ . Το αποτέλεσμα είναι και αυτό τύπου  $t$  και η τιμή του είναι η τιμή που ανατίθεται στη μεταβλητή.

Πίνακας 2: Προτεραιότητα και προσηταιριστικότητα των τελεστών της Edsger.

Τελεστές	Περιγραφή	Αριθμός τελουμένων	Θέση και προσηταιριστικότητα
[ ] ( )	Στοιχείο πίνακα, κλήση συνάρτησης	2 ή >	ειδική
++ --	Αύξηση και μείωση	1	postfix
& * + - !	Διεύθυνση αντικειμένου, αποδεικτοδό- τηση, πρόσημα, λογική άρνηση	1	prefix
new delete	Δυναμική παραχώρηση μνήμης	1	prefix
++ --	Αύξηση και μείωση	1	prefix
( )	Μετατροπή τύπου	2	ειδική
* / %	Πολλαπλασιαστικοί τελεστές	2	infix, αριστερή
+ -	Προσθετικοί τελεστές	2	infix, αριστερή
== != > < <= >=	Σχεσιακοί τελεστές	2	infix, καμία
&&	Λογική σύζευξη	2	infix, αριστερή
	Λογική διάζευξη	2	infix, αριστερή
?:	Τελεστής συνθήκης	3	ειδική
= += -= *= /= %=	Τελεστές ανάθεσης	2	infix, δεξιά
,	Τελεστής παράθεσης	2	infix, αριστερή

- Ο τελεστής  $op=$ , όπου  $op \in \{+, -, *, /, \%\}$ , συνδυάζει υπολογισμό και ανάθεση. Η έκφραση  $l\ op= e$  είναι σημασιολογικά ισοδύναμη με  $l = l\ op\ e$ , με τη διαφορά ότι το τελούμενο  $l$  θα υπολογιστεί μόνο μια φορά.
- Οι τελεστές  $++$  και  $--$  δέχονται ένα τελούμενο και υπάρχουν σε δύο μορφές, μια prefix και μια postfix. Το τελούμενό τους πρέπει να είναι μια l-value αριθμητικού τύπου ή τύπου δείκτη. Το αποτέλεσμα είναι του ίδιου τύπου με το τελούμενο. Οι τελεστές προκαλούν την αύξηση ή τη μείωση της τιμής της l-value. Στην περίπτωση της μορφής prefix το αποτέλεσμα που υπολογίζεται είναι η τιμή της l-value μετά την αύξηση ή τη μείωση, ενώ στην περίπτωση της μορφής postfix το αποτέλεσμα είναι η τιμή της l-value πριν την αύξηση ή τη μείωση.

Στον πίνακα 2 ορίζεται η προτεραιότητα και η προσηταιριστικότητα των τελεστών της Edsger. Οι γραμμές που βρίσκονται υψηλότερα στον πίνακα περιέχουν τελεστές μεγαλύτερης προτεραιότητας. Τελεστές που βρίσκονται στην ίδια γραμμή έχουν την ίδια προτεραιότητα. Στον πίνακα συμπεριλαμβάνονται η κλήση συναρτήσεων, η μετατροπή τύπου και οι τελεστές δυναμικής παραχώρησης μνήμης, που περιγράφονται σε επόμενες ενότητες.

#### 1.4.4 Κλήση συναρτήσεων

Αν  $f$  είναι το όνομα μιας συνάρτησης με αποτέλεσμα τύπου  $t$ , τότε η έκφραση  $f(e_1, \dots, e_n)$  είναι μια r-value με τύπο  $t$ . Ο αριθμός των πραγματικών παραμέτρων  $n$  πρέπει να συμπίπτει με τον αριθμό των τυπικών παραμέτρων της  $f$ . Επίσης, ο τύπος και το είδος κάθε πραγματικής παραμέτρου πρέπει να συμπίπτει με τον τύπο και τον τρόπο περάσματος της αντίστοιχης τυπικής παραμέτρου, σύμφωνα με τους παρακάτω κανόνες.

- Αν η τυπική παράμετρος είναι τύπου  $t$  και περνά κατ' αξία, τότε η αντίστοιχη πραγματική παράμετρος πρέπει να είναι τύπου  $t$ .
- Αν η τυπική παράμετρος είναι τύπου  $t$  και περνά κατ' αναφορά, τότε η αντίστοιχη πραγματική παράμετρος πρέπει να είναι l-value τύπου  $t$ .

Κατά την κλήση μιας συνάρτησης, οι πραγματικές παράμετροι αποτιμώνται από αριστερά προς τα δεξιά.

### 1.4.5 Δυναμική διαχείριση μνήμης

Οι τελεστές new και delete χρησιμοποιούνται για τη δυναμική διαχείριση μνήμης.

- Η έκφραση new  $t$  [ $e$ ], όπου  $t$  οποιοσδήποτε έγκυρος τύπος και  $e$  έκφραση τύπου int, προκαλεί τη δυναμική παραχώρηση μνήμης. Η έκφραση  $e$  καθώς και οι αγκύλες είναι προαιρετικές. Αν δίνονται, τότε η τιμή της  $e$  πρέπει να είναι ένας θετικός ακέραιος αριθμός  $n$ . Διαφορετικά, θεωρείται ότι  $n = 1$ . Το αποτέλεσμα είναι τύπου  $t^*$ . Ο δείκτης αυτός τοποθετείται να δείχνει προς το πρώτο από  $n$  νέα δυναμικά αντικείμενα τύπου  $t$ .
- Η έκφραση delete  $e$ , όπου  $e$  έκφραση τύπου  $t^*$ , προκαλεί την αποδέσμευση της μνήμης στην οποία δείχνει η τιμή της  $e$ . Η μνήμη αυτή πρέπει να έχει προηγουμένως παραχωρηθεί δυναμικά με χρήση του τελεστή new. Το αποτέλεσμα είναι τύπου  $t^*$  και η τιμή του είναι πάντοτε NULL.

### 1.5 Εντολές

Οι εντολές που υποστηρίζει η γλώσσα Edsger είναι οι ακόλουθες:

- Η κενή εντολή ; που δεν κάνει καμία ενέργεια.
- Η εντολή έκφρασης  $e$  ; που αποτιμά την έκφραση  $e$  και αγνοεί την τιμή του αποτελέσματος.
- Η σύνθετη εντολή, που αποτελείται από μια σειρά έγκυρων εντολών ανάμεσα σε άγκιστρα { και }. Οι εντολές αυτές εκτελούνται διαδοχικά, εκτός αν κάποια είναι εντολή άλματος.
- Η εντολή ελέγχου if ( $e$ )  $s_1$  else  $s_2$ . Η έκφραση  $e$  πρέπει να έχει τύπο bool και τα  $s_1, s_2$  να είναι έγκυρες εντολές. Το τμήμα else είναι προαιρετικό.
- Η εντολή βρόχου for ( $e_1$  ;  $e_2$  ;  $e_3$ )  $s$ . Οι εκφράσεις  $e_1, e_2$  και  $e_3$  είναι προαιρετικές. Αν δίνεται, η έκφραση  $e_2$  πρέπει να έχει τύπο bool, διαφορετικά θεωρείται ίση με true. Το  $s$  πρέπει να είναι έγκυρη εντολή. Η σημασιολογία της εντολής βρόχου είναι η ίδια με αυτή της C. **Μπροστά από την εντολή βρόχου μπορεί να υπάρχει μια ετικέτα της μορφής  $I$** : το όνομα της οποίας πρέπει να είναι μοναδικό μέσα στην τρέχουσα συνάρτηση.
- Οι εντολές άλματος break  $I$ ; και continue  $I$ ;, όπου το  $I$  είναι προαιρετικό. Αν δίνεται, το  $I$  πρέπει να είναι το όνομα μιας ετικέτας εντολής βρόχου μέσα στο σώμα της οποίας βρίσκεται η εντολή άλματος. Αν το  $I$  παραλείπεται, οι εντολές αυτές έχουν την ίδια σημασιολογία με τη γλώσσα C, δηλαδή προκαλούν τον τερματισμό ή τη συνέχιση του βρόχου μέσα στον οποίο βρίσκονται. Αν το  $I$  δίνεται, προκαλείται ο τερματισμός ή η συνέχιση του βρόχου που φέρει την ετικέτα  $I$ .
- Η εντολή άλματος return  $e$  ; που τερματίζει την εκτέλεση της τρέχουσας συνάρτησης και επιστρέφει την τιμή της  $e$  ως αποτέλεσμα της συνάρτησης. Αν η τρέχουσα συνάρτηση έχει τύπο αποτελέσματος void τότε η έκφραση  $e$  πρέπει να παραλείπεται. Διαφορετικά, η έκφραση  $e$  πρέπει να έχει τύπο ίδιο με τον τύπο αποτελέσματος της τρέχουσας συνάρτησης.

### 1.6 Οδηγίες προς το μεταγλωττιστή

Η μοναδική οδηγία, που επιτρέπει ο μεταγλωττιστής της γλώσσας Edsger 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 της γραμμής (να μην προηγούνται κενά διαστήματα). Μετά την οδηγία #include ακολουθεί μια σταθερή συμβολοσειρά που περιέχει το όνομα του αρχείου το οποίο θα συμπεριληφθεί σε αυτό το σημείο.

**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. Τα αρχεία που διαβάζονται με την οδηγία `#include` μπορούν να περιέχουν και αυτά τέτοιες 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).

## 1.7 Βιβλιοθήκη έτοιμων συναρτήσεων

Η Edsger υποστηρίζει ένα σύνολο προκαθορισμένων συναρτήσεων, οι οποίες είναι υλοποιημένες σε assembly του 80x86 ως μια βιβλιοθήκη έτοιμων συναρτήσεων (run-time library). Οι επικεφαλίδες αυτών των συναρτήσεων βρίσκονται σε κατάλληλα αρχεία επικεφαλίδων (header files), τα οποία διαβάζονται με χρήση της οδηγίας `#include`. Εφόσον το αντίστοιχο αρχείο επικεφαλίδας έχει συμπεριληφθεί σε ένα πρόγραμμα και η βιβλιοθήκη έτοιμων συναρτήσεων χρησιμοποιηθεί κατά τη συνένωση (linking) του τελικού κώδικα, οι συναρτήσεις που περιέχει βρίσκονται στη διάθεση του προγραμματιστή. Είναι ορατές σε κάθε δομική μονάδα, εκτός αν επισκιάζονται από μεταβλητές, παραμέτρους ή άλλες συναρτήσεις με το ίδιο όνομα.

Παρακάτω δίνονται οι δηλώσεις τους, όπως γράφονται στα αρχεία επικεφαλίδας της Edsger, και εξηγείται η λειτουργία τους.

### 1.7.1 Είσοδος και έξοδος — `#include "stdio.h"`

```
void writeInteger (int n);
void writeBoolean (bool b);
void writeChar    (char c);
void writeReal    (double d);
void writeString  (char * s);
```

Οι συναρτήσεις αυτές χρησιμοποιούνται για την εκτύπωση τιμών που ανήκουν στους βασικούς τύπους της Edsger, καθώς και για την εκτύπωση συμβολοσειρών.

```
int    readInteger ();
bool   readBoolean ();
char   readChar    ();
double readReal    ();
void   readString  (int size, char * s);
```

Αντίστοιχα, οι παραπάνω συναρτήσεις χρησιμοποιούνται για την εισαγωγή τιμών που ανήκουν στους βασικούς τύπους της Edsger και για την εισαγωγή συμβολοσειρών. Η συνάρτηση `readString` χρησιμοποιείται για την ανάγνωση μιας συμβολοσειράς μέχρι τον επόμενο χαρακτήρα αλλαγής γραμμής. Οι παράμετροι της καθορίζουν το μέγιστο αριθμό χαρακτήρων (συμπεριλαμβανομένου του τελικού `'\0'`) που επιτρέπεται να διαβαστούν και τον πίνακα χαρακτήρων στον οποίο αυτοί θα τοποθετηθούν. Ο χαρακτήρας αλλαγής γραμμής δεν αποθηκεύεται. Αν το μέγεθος του πίνακα εξαντληθεί πριν συναντηθεί χαρακτήρας αλλαγής γραμμής, η ανάγνωση θα συνεχιστεί αργότερα από το σημείο όπου διακόπηκε.

### 1.7.2 Μαθηματικές συναρτήσεις — `#include "math.h"`

```
int    abs (int n);
double fabs (double d);
```

Η απόλυτη τιμή ενός ακέραιου ή πραγματικού αριθμού.

```
double sqrt (double d);
double sin  (double d);
double cos  (double d);
double tan  (double d);
```

```
double atan (double d);
double exp  (double d);
double ln   (double d);
double pi   ();
```

Βασικές μαθηματικές συναρτήσεις: τετραγωνική ρίζα, τριγωνομετρικές συναρτήσεις, εκθετική συνάρτηση, φυσικός λογάριθμος, ο αριθμός  $\pi$ .

### 1.7.3 Συναρτήσεις μετατροπής — #include “stdlib.h”

```
int trunc (double d);
int round (double d);
```

Η `trunc` επιστρέφει τον πλησιέστερο ακέραιο αριθμό, η απόλυτη τιμή του οποίου είναι μικρότερη από την απόλυτη τιμή του `d`. Η `round` επιστρέφει τον πλησιέστερο ακέραιο αριθμό. Σε περίπτωση αμφιβολίας, προτιμάται ο αριθμός με τη μεγαλύτερη απόλυτη τιμή.

```
int  ord (char c);
char chr (int n);
```

Μετατρέπουν από ένα χαρακτήρα στον αντίστοιχο κωδικό ASCII και αντίστροφα.

### 1.7.4 Συναρτήσεις διαχείρισης συμβολοσειρών — #include “string.h”

```
int  strlen (char * s);
int  strcmp (char * s1, char * s2);
void strcpy (char * trg, char * src);
void strcat (char * trg, char * src);
```

Οι συναρτήσεις αυτές έχουν ακριβώς την ίδια λειτουργία με τις συνώνυμές τους στη βιβλιοθήκη συναρτήσεων της γλώσσας C.

## 2 Πλήρης γραμματική της Edsger

Η σύνταξη της γλώσσας Edsger δίνεται παρακάτω σε μορφή EBNF. Η γραμματική που ακολουθεί είναι *διφορούμενη*, οι αμφισημίες όμως μπορούν να ξεπεραστούν αν λάβει κανείς υπόψη τους κανόνες προτεραιότητας και προσεταιριστικότητας των τελεστών, όπως περιγράφονται στην ενότητα 1.4.3. Τα σύμβολα  $\langle I \rangle$ ,  $\langle \text{int-const} \rangle$ ,  $\langle \text{double-const} \rangle$ ,  $\langle \text{char-const} \rangle$  και  $\langle \text{string-literal} \rangle$  είναι τερματικά σύμβολα της γραμματικής.

```
1   $\langle \text{program} \rangle ::= ( \langle \text{declaration} \rangle )^+$ 
2   $\langle \text{declaration} \rangle ::= \langle \text{variable-declaration} \rangle \mid \langle \text{function-declaration} \rangle \mid \langle \text{function-definition} \rangle$ 
3   $\langle \text{variable-declaration} \rangle ::= \langle \text{type} \rangle \langle \text{declarator} \rangle ( \text{“,”} \langle \text{declarator} \rangle )^* \text{“;”}$ 
4   $\langle \text{type} \rangle ::= \langle \text{basic-type} \rangle ( \text{“*”} )^*$ 
5   $\langle \text{basic-type} \rangle ::= \text{“int”} \mid \text{“char”} \mid \text{“bool”} \mid \text{“double”}$ 
6   $\langle \text{declarator} \rangle ::= \langle I \rangle [ \text{“[”} \langle \text{constant-expression} \rangle \text{“]”}]$ 
7   $\langle \text{function-declaration} \rangle ::= \langle \text{result-type} \rangle \langle I \rangle \text{“ (“} [ \langle \text{parameter-list} \rangle ] \text{“) ” “;”}$ 
8   $\langle \text{result-type} \rangle ::= \langle \text{type} \rangle \mid \text{“void”}$ 
9   $\langle \text{parameter-list} \rangle ::= \langle \text{parameter} \rangle ( \text{“,”} \langle \text{parameter} \rangle )^*$ 
10  $\langle \text{parameter} \rangle ::= [ \text{“byref”} ] \langle \text{type} \rangle \langle I \rangle$ 
11  $\langle \text{function-definition} \rangle ::= \langle \text{result-type} \rangle \langle I \rangle \text{“ (“} [ \langle \text{parameter-list} \rangle ] \text{“) ”}$ 
12  $\text{“{”} ( } \langle \text{declaration} \rangle )^* ( \langle \text{statement} \rangle )^* \text{“} \text{”}$ 
13  $\langle \text{statement} \rangle ::= \text{“;”} \mid \langle \text{expression} \rangle \text{“;”} \mid \text{“{”} ( } \langle \text{statement} \rangle )^* \text{“} \text{”}$ 
14  $\mid \text{“if” “ (“} \langle \text{expression} \rangle \text{“) ” } \langle \text{statement} \rangle [ \text{“else” } \langle \text{statement} \rangle ]$ 
15  $\mid [ \langle I \rangle \text{“:”} ] \text{“for” “ (“} [ \langle \text{expression} \rangle ] \text{“;” } [ \langle \text{expression} \rangle ] \text{“;”}$ 
16  $\mid [ \langle \text{expression} \rangle ] \text{“) ” } \langle \text{statement} \rangle$ 
17  $\mid \text{“continue”} [ \langle I \rangle ] \text{“;”} \mid \text{“break”} [ \langle I \rangle ] \text{“;”} \mid \text{“return”} [ \langle \text{expr} \rangle ] \text{“;”}$ 
18  $\langle \text{expression} \rangle ::= \langle I \rangle \mid \text{“ (“} \langle \text{expression} \rangle \text{“) ”} \mid \text{“true”} \mid \text{“false”} \mid \text{“NULL”}$ 
19  $\mid \langle \text{int-const} \rangle \mid \langle \text{char-const} \rangle \mid \langle \text{double-const} \rangle \mid \langle \text{string-literal} \rangle$ 
20  $\mid \langle I \rangle \text{“ (“} [ \langle \text{expression-list} \rangle ] \text{“) ”} \mid \langle \text{expression} \rangle \text{“[”} \langle \text{expression} \rangle \text{“]”}$ 
21  $\mid \langle \text{unary-operator} \rangle \langle \text{expression} \rangle \mid \langle \text{expression} \rangle \langle \text{binary-operator} \rangle \langle \text{expression} \rangle$ 
22  $\mid \langle \text{unary-assignment} \rangle \langle \text{expression} \rangle \mid \langle \text{expression} \rangle \langle \text{unary-assignment} \rangle$ 
23  $\mid \langle \text{expression} \rangle \langle \text{binary-assignment} \rangle \langle \text{expression} \rangle \mid \text{“ (“} \langle \text{type} \rangle \text{“) ” } \langle \text{expression} \rangle$ 
24  $\mid \langle \text{expression} \rangle \text{“?”} \langle \text{expression} \rangle \text{“:”} \langle \text{expression} \rangle$ 
25  $\mid \text{“new”} \langle \text{type} \rangle [ \text{“[”} \langle \text{expression} \rangle \text{“]”}] \mid \text{“delete”} \langle \text{expression} \rangle$ 
26  $\langle \text{expression-list} \rangle ::= \langle \text{expression} \rangle ( \text{“,”} \langle \text{expression} \rangle )^*$ 
27  $\langle \text{constant-expression} \rangle ::= \langle \text{expression} \rangle$ 
28  $\langle \text{unary-operator} \rangle ::= \text{“\&”} \mid \text{“*”} \mid \text{“+”} \mid \text{“-”} \mid \text{“!”}$ 
29  $\langle \text{binary-operator} \rangle ::= \text{“*”} \mid \text{“/”} \mid \text{“\%”} \mid \text{“+”} \mid \text{“-”} \mid \text{“<”} \mid \text{“>”} \mid \text{“<=”} \mid \text{“>=”} \mid \text{“==”} \mid \text{“!=”}$ 
30  $\mid \text{“\&\&”} \mid \text{“||”} \mid \text{“,”}$ 
31  $\langle \text{unary-assignment} \rangle ::= \text{“++”} \mid \text{“-”}$ 
32  $\langle \text{binary-assignment} \rangle ::= \text{“=”} \mid \text{“*=”} \mid \text{“/=”} \mid \text{“\%=”} \mid \text{“+=”} \mid \text{“-=”}$ 
33
```



## 3 Παραδείγματα

Στην ενότητα αυτή δίνονται έξι παραδείγματα προγραμμάτων στη γλώσσα Edsger.

### 3.1 Πες γεια!

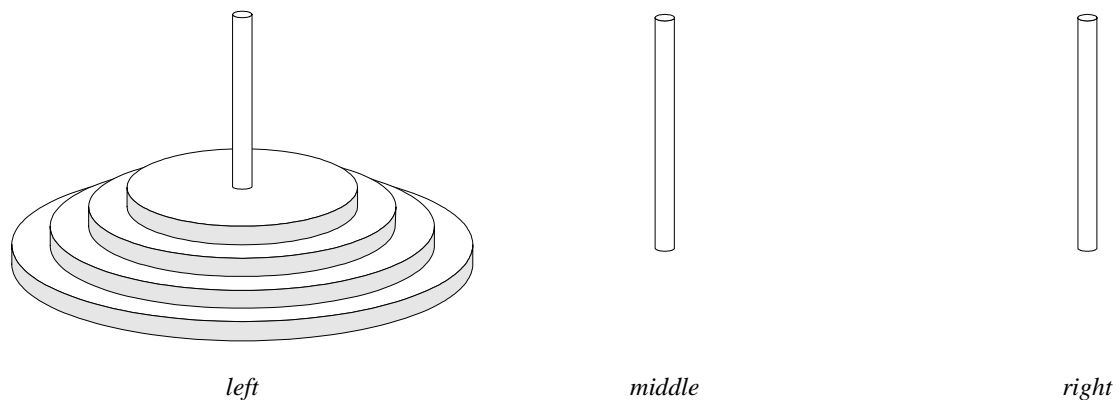
Το παρακάτω παράδειγμα είναι το απλούστερο πρόγραμμα στη γλώσσα Edsger που παράγει κάποιο αποτέλεσμα ορατό στο χρήστη. Το πρόγραμμα αυτό τυπώνει απλώς ένα μήνυμα.

```
1  #include "stdio.h"
2
3  void main ()
4  {
5      writeString("Hello world!\n");
6  }
```

### 3.2 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 διαμέτροι των δακτυλίων είναι διαφορετικές και αυτοί είναι περασμένοι από κάτω προς τα πάνω σε φθίνουσα σειρά εξωτερικής διαμέτρου, όπως φαίνεται στο σχήμα 1. Ζητείται να μεταφερθούν



Σχήμα 1: Οι πύργοι του Hanoi.

οι δακτύλιοι από τον πρώτο στον τρίτο στύλο (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 Edsger που λύνει αυτό το πρόβλημα δίνεται παρακάτω. Η συνάρτηση `hanoi` είναι αναδρομική.



```

1  #include "stdio.h"
2
3  void main ()
4  {
5      int numberOfRings;
6
7      void hanoi (char * source, char * target, char * auxiliary, int rings)
8      {
9          void move (char * source, char * target)
10         {
11             writeString("Move from ");
12             writeString(source);
13             writeString(" to ");
14             writeString(target);
15             writeString(".\n");
16         }
17
18         if (rings >= 1) {
19             hanoi(source, auxiliary, target, rings-1);
20             move(source, target);
21             hanoi(auxiliary, target, source, rings-1);
22         }
23     }
24
25     writeString("Please, give the number of rings: ");
26     numberOfRings = readInteger();
27     writeString("\nHere is the solution:\n\n");
28     hanoi("left", "right", "middle", numberOfRings);
29 }

```

### 3.3 Πρώτοι αριθμοί

Το παρακάτω παράδειγμα προγράμματος στη γλώσσα Edsger 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  $6k \pm 1$ , όπου  $k$  φυσικός αριθμός.

#### Κύριο πρόγραμμα

τύπωσε τους αριθμούς 2 και 3  
για  $t := 6$  μέχρι  $n$  με βήμα 6 κάνε τα εξής:  
    αν ο αριθμός  $t - 1$  είναι πρώτος τότε τύπωσέ τον  
    αν ο αριθμός  $t + 1$  είναι πρώτος τότε τύπωσέ τον

#### 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  
για  $i := 3$  μέχρι  $t/2$  με βήμα 2 κάνε τα εξής:  
    αν ο  $t$  διαιρείται με τον  $i$  τότε ο  $t$  δεν είναι πρώτος  
ο  $t$  είναι πρώτος

Το αντίστοιχο πρόγραμμα στη γλώσσα Edsger είναι το εξής:

```

1  #include "stdio.h"
2
3  void main ()
4  {
5      int limit, number, counter;
6
7      bool prime (int n)
8      {
9          int i;
10
11         if (n < 0)
12             return prime(-n);
13         if (n < 2)
14             return false;
15         if (n == 2)
16             return true;
17         if (n % 2 == 0)
18             return false;
19         for (i = 3; i <= n / 2; i += 2)
20             if (n % i == 0)
21                 return false;
22         return true;
23     }
24
25     writeString("Please, give the upper limit: ");
26     limit = readInteger();
27     writeString("Prime numbers between 0 and ");
28     writeInteger(limit);
29     writeString("\n\n");
30     counter = 0;
31     if (limit >= 2) {
32         counter++;
33         writeString("2\n");
34     }
35     if (limit >= 3) {
36         counter++;
37         writeString("3\n");
38     }
39     for (number = 6; number <= limit; number += 6) {
40         if (prime(number - 1)) {
41             counter++;
42             writeInteger(number - 1);
43             writeString("\n");
44         }
45         if (number != limit && prime(number + 1)) {
46             counter++;
47             writeInteger(number + 1);
48             writeString("\n");
49         }
50     }
51     writeString("\n");
52     writeInteger(counter);
53     writeString(" prime number(s) were found.\n");
54 }

```

### 3.4 Αντιστροφή συμβολοσειράς

Το πρόγραμμα που ακολουθεί στη γλώσσα Edsger εκτυπώνει το μήνυμα “Hello world!” αντιστρέφοντας τη δοθείσα συμβολοσειρά.

```
1  #include "stdio.h"
2  #include "string.h"
3
4  void main ()
5  {
6      void reverse (char * s, char * r)
7      {
8          int i, l;
9
10         for (i = 0, l = strlen(s); i < l; i++)
11             r[i] = s[l-i-1];
12         r[i] = '\0';
13     }
14
15     char p [20];
16
17     reverse("\n!dlrow olleH", p);
18     writeString(p);
19 }
```

### 3.5 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 Edsger τον χρησιμοποιεί για να ταξινομήσει έναν πίνακα ακεραίων αριθμών κατ’ αύξουσα σειρά. Αν  $x$  είναι ο πίνακας που πρέπει να ταξινομηθεί και  $n$  είναι το μέγεθός του (θεωρούμε σύμφωνα με τη σύμβαση της Edsger 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 Edsger είναι το εξής:

```
1  #include "stdio.h"
2
3  void bsort (int * x, int n)
4  {
5      int i;
6      bool changed;
7
8      void swap (byref int x, byref int y)
9      {
10         int t;
11
12         t = x;
13         x = y;
14         y = t;
15     }
```

```

16
17     for (changed = true; changed;)
18         for (i = 0, changed = false; i < n-1; i++)
19             if (x[i] > x[i+1]) {
20                 swap(x[i], x[i+1]);
21                 changed = true;
22             }
23 }
24
25 void main ()
26 {
27     void printArray (char * msg, int * x, int n)
28     {
29         int i;
30
31         writeString(msg);
32         for (i = 0; i < n; i++) {
33             if (i > 0)
34                 writeString(", ");
35             writeInteger(x[i]);
36         }
37         writeString("\n");
38     }
39
40     int i, x[16], seed;
41
42     for (i = 0, seed = 65; i < 16; i++)
43         x[i] = seed = (seed * 137 + 221 + i) % 101;
44     printArray("Initial array: ", x, 16);
45     bsort(x, 16);
46     printArray("Sorted array: ", x, 16);
47 }

```

### 3.6 Μέση τιμή τυχαίας μεταβλητής

Το πρόγραμμα που ακολουθεί υπολογίζει τη μέση τιμή μιας αέραςιας τυχαίας μεταβλητής, που μεταβάλλεται ομοιόμορφα στο διάστημα από 0 έως  $n - 1$ . Η τιμή του  $n$  καθορίζεται από το χρήστη, όπως και το πλήθος  $k$  των δειγμάτων που χρησιμοποιούνται για τον υπολογισμό.

```

1  #include "stdio.h"
2
3  void main ()
4  {
5      int    n, k, i, seed;
6      double sum;
7
8      writeString("Give n: ");
9      n = readInteger();
10     writeString("Give k: ");
11     k = readInteger();
12
13     for (i = 0, sum = 0.0, seed = 65; i < k; i++)
14         sum += (double) (seed = (seed * 137 + 221 + i) % n);
15
16     if (k > 0) {
17         writeString("Mean: ");
18         writeReal(sum / (double) k);
19         writeString("\n");

```

20 }  
21 }

Από την εκτέλεση του προγράμματος διαπιστώνει κανείς ότι η μέση τιμή που προκύπτει διαφέρει σημαντικά από τη θεωρητική μέση τιμή  $(n - 1)/2$ , που θα έπρεπε να προκύπτει όταν το  $k$  γίνει αρκετά μεγάλο. Αυτό οφείλεται στον απλοϊκό αλγόριθμο που έχει χρησιμοποιηθεί για τη γέννηση ψευδοτυχαίων αριθμών, λόγω του οποίου η μεταβλητή απέχει αρκετά από το να είναι τυχαία.

## 4 Οδηγίες για την παράδοση

Ο τελικός μεταγλωττιστής πρέπει να μπορεί να εξάγει κατά βούληση ενδιάμεσο και τελικό κώδικα. Εφόσον δεν έχουν καθορισθεί οι παράμετροι λειτουργίας `-f` ή `-i`, που εξηγούνται παρακάτω, ο μεταγλωττιστής θα δέχεται το πηγαίο πρόγραμμα από ένα αρχείο με οποιαδήποτε κατάληξη (π.χ. `*.eds`) που θα δίνεται ως το μοναδικό του όρισμα. Ο ενδιάμεσος κώδικας θα τοποθετείται σε αρχείο με κατάληξη `*.imm` και ο τελικός σε αρχείο με κατάληξη `*.asm`. Τα αρχεία αυτά θα βρίσκονται στον ίδιο κατάλογο και θα έχουν το ίδιο κυρίως όνομα. Π.χ. από το πηγαίο αρχείο `/tmp/hello.eds` θα παράγονται τα `/tmp/hello.imm` και `/tmp/hello.asm`.

Το εκτελέσιμο του τελικού μεταγλωττιστή θα πρέπει να δέχεται τις παρακάτω παραμέτρους:

- o σημαία βελτιστοποίησης (προαιρετική).
- f πρόγραμμα στο standard input, έξοδος τελικού κώδικα στο standard output.
- i πρόγραμμα στο standard input, έξοδος ενδιάμεσου κώδικα στο standard output.

Επίσης, η τιμή που θα επιστρέφεται στο λειτουργικό σύστημα από το μεταγλωττιστή θα πρέπει να είναι μηδενική στην περίπτωση επιτυχούς μεταγλώττισης και μη μηδενική σε αντίθετη περίπτωση.

Για την εύκολη ανάπτυξη του μεταγλωττιστή προτείνεται η χρήση ενός αρχείου `Makefile` με τα εξής (τουλάχιστον) χαρακτηριστικά:

- Με απλό `make`, θα δημιουργεί το εκτελέσιμο του τελικού μεταγλωττιστή.
- Με `make clean`, θα σβήνει όλα ανεξαιρέτως τα αρχεία που παράγονται αυτόματα (π.χ. αυτά που παράγουν `bison` και `flex`, τα `object files`) εκτός από το τελικό εκτελέσιμο.
- Με `make distclean`, θα σβήνει όλα τα παραπάνω και το τελικό εκτελέσιμο.

Ένα παράδειγμα ενός τέτοιου `Makefile`, υποθέτοντας ότι γλώσσα υλοποίησης είναι η C και γίνεται χρήση των εργαλείων `flex` και `bison`, είναι το παρακάτω.

```
CC=gcc
CFLAGS=-Wall

compiler: compiler.lex.o compiler.tab.o symbol.o symbolc.o
    $(CC) $(CFLAGS) -o compiler compiler.lex.o compiler.tab.o \
        symbol.o symbolc.o

compiler.lex.c: compiler.l compiler.tab.h
    flex -it compiler.l > compiler.lex.c

compiler.tab.c compiler.tab.h: compiler.y
    bison -dv compiler.y

clean:
    $(RM) *.o compiler.tab.c compiler.tab.h compiler.lex.c \
        compiler core
```

Η μορφή εμφάνισης των τετράδων και του τελικού κώδικα θα πρέπει να είναι αυτή που προτείνεται στο βιβλίο και στις διαλέξεις. Επίσης να ληφθούν υπόψη οι παρακάτω οδηγίες:

- Ενδιάμεσος κώδικας: να υιοθετηθεί μορφοποίηση ισοδύναμη με  

```
printf("%d: %s, %s, %s, %s\n", ...)
```
- Τελικός κώδικας: προσοχή να δοθεί στη στηλοθέτηση (indentation). Να ακολουθηθεί το παρακάτω υπόδειγμα:

```
ετικέττα: <tab> εντολή <tab> όρισμα-1, όρισμα-2
          <tab> εντολή <tab> όρισμα-1, όρισμα-2
```