ΕΘΝΙΚΟ & ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΘΠ06 Μεταγλωττιστές

Εργασία Εξαμήνου:
Υλοποίηση ενός Μεταγλωττιστή για τη Γλώσσα Floop2009

Βοηθοί:
Χαράλαμπος Νικολάου(charnik)
Χρύσα Τζούμα(grad1005)

3/16/2009

ΘΕΜΑ:

Να σχεδιαστεί και να υλοποιηθεί από κάθε ομάδα φοιτητών ένας μεταγλωττιστής για τη γλώσσα Floop2009. Γλώσσα υλοποίησης μπορεί να είναι μία από τις C/C++. Η επιλογή κάποιας άλλης γλώσσας υλοποίησης μπορεί να γίνει κατόπιν συνεννόησης με το διδάσκοντα. Επίσης θα χρησιμοποιηθούν τα εργαλεία flex και bison.

1 Παραδοτέα, ημερομηνίες και βαθμολόγηση

Τα τμήματα του μεταγλωττιστή και η κατανομή μονάδων φαίνονται στον παρακάτω πίνακα. Οι ημερομηνίες παράδοσης αναγράφονται στη σελίδα του μαθήματος

Τμήμα του μεταγλωττιστή	***Μονάδες***	***Bonus***
Λεκτικός αναλυτής	0.5	-
Υλοποίηση συντακτικού αναλυτή με bison	1	-
Πίνακας συμβόλων και σημασιολογική ανάλυση	2	-
Ενδιάμεσος κώδικας	1.5	-
Τελικός κώδικας	-	2.0
Συνολική εργασία	**5.0**	**2.0**

Για τα διάφορα τμήματα της εργασίας πρέπει να παραδίδεται εμπρόθεσμα από κάθε ομάδα ο αντίστοιχος κώδικας σε ηλεκτρονική μορφή, με βάση τις οδηγίες που αναγράφονται στη σελίδα του μαθήματος.

2 Περιγραφή της γλώσσας Floop2009

Η γλώσσα Floop2009 βασίζεται σε ένα γνήσιο υποσύνολο της C++. Λόγω των πολλών ομοιοτήτων της Floop2009 με την C++, τόσο από πλευράς σύνταξης όσο και από πλευράς σημασιολογίας, η περιγραφή θα είναι σύντομη με εξαίρεση τα σημεία όπου οι γλώσσες διαφέρουν.

2.1 Λεκτικές Μονάδες

Οι λεκτικές μονάδες της γλώσσας Floop2009 χωρίζονται στις παρακάτω κατηγορίες:

- Τις *λέξεις κλειδιά*, οι οποίες είναι οι παρακάτω: `public`, `private`, `while`, `for`, `continue`, `goto`, `break`, `if`, `else`, `integer`, `float`, `boolean`, `char`, `void`, `class`, `true`, `false`, `new`, `static`, `extends`, `main`, `return`
- 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.
- Τις *ακέραιες σταθερές* χωρίς πρόσημο, που αποτελούνται από ένα ή περισσότερα δεκαδικά ψηφία. Παραδείγματα ακέραιων σταθερών είναι τα ακόλουθα:

  `0` `42` `1284` `00200`

- Τις *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`

Χαρακτήρας	Περιγραφή
`\n`	χαρακτήρας αλλαγής γραμμής (line feed)
`\t`	χαρακτήρας στηλοθέτησης (TAB)
`\r`	χαρακτήρας επιστροφής στην αρχή της γραμμής
`\0`	χαρακτήρας με ASCII κωδικό **0**
`\\`	χαρακτήρας **** (backslash)
`\'`	χαρακτήρας **'** (απλό εισαγωγικό)
`\"`	χαρακτήρας **"** (διπλό εισαγωγικό)

Πίνακας 1 Ακολουθίες διαφυγής (escape sequences)

- Τους *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 οι ακόλουθες:

```
'a'        '1 '        '\n'        '\' '
```

- Τις *σταθερές συμβολοσειρές (string)*, 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"      "Route 66"      "Hello world!\n"

"Name:\t\"Douglas Adams\"\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 Floop2009 μπορεί επίσης να περιέχει τα παρακάτω, **τα οποία αγνοούνται** (δηλαδή τα αναγνωρίζετε αλλά δεν κάνετε τίποτα για αυτό):

- *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*, 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.**

  Στα παρακάτω δύο παραδείγματα με πλάγια γράμματα είναι τι θα πρέπει να αναγνωρίζετε σαν σχόλια, με bold είναι η αρχή και το τέλος τους ενώ με κανονικά γράμματα είναι τι μένει έξω από τα σχόλια:

  1. ***(* This is an example of comments*)*** This is not a comment
  2. ***(* This is an example (* of comment*)*** This is not a comment*)
  3. ***(* This is an (* example (* of comment*)*** This is not a comment*)

2.2 Τύποι δεδομένων

Η Floop2009 υποστηρίζει τέσσερις βασικούς τύπους δεδομένων:

- `integer`: ακέραιοι αριθμοί,
- `boolean`: λογικές τιμές,
- `char`: χαρακτήρες, και

- `float`: πραγματικοί αριθμοί.

Εκτός από τους βασικούς τύπους, η Floop2009 υποστηρίζει επίσης μονοδιάστατους πίνακες. Μια δήλωση πίνακα είναι της μορφής

```
Τύπος  Όνομα Πίνακα   [Μέγεθος(Integer)]
```

Έτσι για παράδειγμα οι παρακάτω δηλώσεις πινάκων θα πρέπει να υποστηρίζονται από την Floop2009:

```
integer spok[23]
integer spok[k]
```

Οι τύποι integer και float ονομάζονται *αριθμητικοί* τύποι. Ορίζουμε ότι ο τύπος float είναι «ευρύτερος» του τύπου integer και υποστηρίζουμε την μετατροπή τύπου (type casting) από «στενότερο» σε «ευρύτερο» τύπο.

Ο αριθμός των bytes που καταλαμβάνουν τα δεδομένα κάθε τύπου στη μνήμη του υπολογιστή, καθώς και ο ακριβής τρόπος παράστασης αυτών εξαρτώνται από την υλοποίηση της Floop2009. Στο πλαίσιο αυτής της εργασίας δεν θα ασχοληθούμε με το πρόβλημα.

2.3 Δομή του προγράμματος

Ένα πρόγραμμα Floop2009 μπορεί να αποτελείται από τα παρακάτω:

- Δηλώσεις μεταβλητών
- Ορισμούς συναρτήσεων
- Ορισμούς κλάσεων

Δεν υπάρχει κανένας περιορισμός στη σειρά με την οποία εμφανίζονται αυτά. Για παράδειγμα, ένα πρόγραμμα μπορεί να έχει στην αρχή δηλώσεις μεταβλητών, στη συνέχεια δηλώσεις συναρτήσεων, μετά ξανά μεταβλητές, μετά δηλώσεις κλάσεων, ξανά συναρτήσεις, κλάσεις, κτλ. Ένα πρόγραμμα μπορεί να έχει μηδέν ή περισσότερες δηλώσεις μεταβλητών και κλάσεων. Ένα πρόγραμμα πρέπει να έχει **μία** ή περισσότερες δηλώσεις συναρτήσεων. Σε κάθε πρόγραμμα θα πρέπει να υπάρχει ο ορισμός **μιας** συνάρτησης με επικεφαλίδα `void main()`. Από αυτή τη συνάρτηση ξεκινά η εκτέλεση του προγράμματος.

Επίσης υποστηρίζεται η υπερφόρτωση συναρτήσεων (overloading), δηλαδή μπορούν να υπάρχουν πολλές συναρτήσεις με το ίδιο όνομα αλλά με διαφορετικό αριθμό και τύπο ορισμάτων ώστε να μπορούμε να διαχωρίσουμε ποια συνάρτηση θα κληθεί κάθε φορά ανάλογα με τα ορίσματα που της δίνουμε. Σημειώνεται ότι η ύπαρξη δύο συναρτήσεων με ίδιο όνομα και ίδιο αριθμό και τύπο ορισμάτων, αλλά διαφορετικό τύπο επιστροφής δε θεωρείται υπερφόρτωση. Μία τέτοια περίπτωση θα πρέπει να θεωρείται ως λανθασμένη.

Η Floop2009 ακολουθεί τους κανόνες εμβέλειας της C++, όσον αφορά στην ορατότητα των ονομάτων μεταβλητών, υποπρογραμμάτων και παραμέτρων.

2.3.1 Μεταβλητές

Οι δηλώσεις μεταβλητών γίνονται με την αναγραφή του τύπου ακολουθούμενου από ένα ή περισσότερα ονόματα. Παραδείγματα δηλώσεων είναι:

```
integer i;
float a,b,c;
```

Επιπλέον στην δήλωση των τοπικών μεταβλητών βασικού τύπου υποστηρίζεται και ο όρος `static`. Θυμίζεται ότι μία μεταβλητή τύπου `static` όταν δηλώνεται μέσα σε συνάρτηση, διατηρεί την τιμή της και έξω από τα όρια της συνάρτησης. Για παράδειγμα:

```
static integer i;
```

2.3.2 Συναρτήσεις

Κάθε συνάρτηση είναι μια δομική μονάδα και αποτελείται από την επικεφαλίδα και το σώμα της. Στην επικεφαλίδα αναφέρεται το όνομα της συνάρτησης, οι τυπικές της παράμετροι μέσα σε παρενθέσεις και ο τύπος του αποτελέσματος. Οι παρενθέσεις είναι υποχρεωτικές ακόμα και αν ένα υποπρόγραμμα δεν έχει τυπικές παραμέτρους. Επίσης αν η συνάρτηση δεν επιστρέφει τιμή τότε ο τύπος της ορίζεται ως `void`.

Κάθε τυπική παράμετρος χαρακτηρίζεται από το όνομά της, τον τύπο της και τον τρόπο περάσματος. Η Floop2009 υποστηρίζει πέρασμα παραμέτρων κατ' αξία (by value) και κατ' αναφορά (by reference). Εξ' ορισμού ολες οι παράμετροι στην Floop2009 περνιούνται κατ' αξία εκτός αν του ονόματος της τυπικής παραμέτρου προηγηθεί ο διαχωριστής `&` οπότε περνά κατ' αναφορά. Το σώμα μιας συνάρτησης περικλείεται μέσα σε άγκιστρα `{}`. Ακολουθούν παραδείγματα επικεφαλίδων συναρτήσεων.

```
void f1();
integer f2 (integer a, char b);
float f3 (integer &a);
```

Το σώμα μιας συνάρτησης μπορεί να αποτελείται από

- Δηλώσεις μεταβλητών.
- Εντολές.

Δεν υπάρχει κανένας περιορισμός στην σειρά με την οποία εμφανίζονται τα παραπάνω. Το σώμα μιας συνάρτησης μπορεί να περιέχει μηδέν ή περισσότερες δηλώσεις μεταβλητών. Αν ο τύπος επιστροφής της συνάρτησης είναι `void` τότε το σώμα της μπορεί να περιέχει μηδέν ή περισσότερες εντολές. Αν το τύπος επιστροφής της συνάρτησης δεν είναι `void` τότε το σώμα της θα πρέπει να περιέχει τουλάχιστον την εντολή `return`. Οι εντολές περιγράφονται στο κεφάλαιο 2.5.

Παράδειγμα:

```
integer foo(integer k, integer bound){
       integer p;
       p=34*k;
       integer i=0;
       integer z=0;
       for(i=1; i<=k; i++){
             if (z<bound)
                   z=p*i;
```

```
        }
        return z;
    }
```

2.3.3 Κλήση συναρτήσεων

Έστω *f* είναι το όνομα μιας συνάρτησης με αποτέλεσμα τύπου t και έστω η έκφραση $f(e_1, \ldots, e_n)$, τότε 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. Κατά την κλήση μιας συνάρτησης, οι πραγματικές παράμετροι αποτιμώνται από αριστερά προς τα δεξιά. Δείτε στο κεφάλαιο 2.7 για το πως καλούνται συναρτήσεις που έχουν οριστεί στο σώμα μιας κλάσης.

2.4 Τελεστές

Οι τελεστές της Floop2009 διακρίνονται σε τελεστές με ένα όρισμα και τελεστές με δύο ορίσματα. Από τους πρώτους, ορισμένοι γράφονται πριν το όρισμα (prefix) και ορισμένοι μετά (postfix), ενώ οι δεύτεροι γράφονται πάντα μεταξύ των ορισμάτων (infix). Η αποτίμηση των ορισμάτων των τελεστών με δυο ορίσματα γίνεται από αριστερά προς τα δεξιά.

Στη συνέχεια περιγράφονται οι τελεστές της Floop2009:

- Οι τελεστές με ένα όρισμα + και − υλοποιούν τους τελεστές προσήμου. Το όρισμα πρέπει να είναι αριθμητικού τύπου και το αποτέλεσμα είναι του ίδιου τύπου με το όρισμα.
- Ο τελεστής `!` υλοποιεί τη λογική άρνηση. Το τελούμενό του πρέπει να είναι τύπου `boolean`, και τον ίδιο τύπο έχει και το αποτέλεσμα.
- Οι τελεστές με δύο ορίσματα `+`, `-`, `*`, `/` και `%` υλοποιούν αντίστοιχα τις αριθμητικές πράξεις της πρόσθεσης, της αφαίρεσης, του πολλαπλασιασμού, της πραγματικής διαίρεσης και του υπόλοιπου της ακέραιας διαίρεσης. Τα ορίσματά τους πρέπει να είναι εκφράσεις αριθμητικών τύπων. Στην περίπτωση των τελεστών `+`, `-`, `*`και `/`, αν και τα δύο ορίσματα είναι τύπου `integer` τότε και το αποτέλεσμα είναι τύπου `integer`, διαφορετικά το αποτέλεσμα είναι τύπου `float`. Στην περίπτωση του τελεστή `%`, τα ορίσματα πρέπει να είναι τύπου `integer` και το αποτέλεσμα είναι επίσης τύπου `integer`.
- Οι τελεστές `==` και `!=` υλοποιούν αντίστοιχα την ισότητα και την ανισότητα. Το αποτέλεσμα είναι τύπου `boolean`. Τα ορίσματα πρέπει να είναι είτε και τα δύο αριθμητικά, οπότε συγκρίνονται οι αριθμητικές τιμές τους, είτε του ίδιου τύπου, ο οποίος δεν πρέπει να είναι τύπος πίνακα. Στη δεύτερη περίπτωση, συγκρίνονται οι δυαδικές αναπαραστάσεις των τιμών των ορισμάτων.
- Οι τελεστές `<`, `>`, `<=` και `>=` υλοποιούν τις σχέσεις ανισότητας μεταξύ αριθμών. Τα ορίσματα πρέπει να είναι και τα δύο αριθμητικά και το αποτέλεσμα είναι τύπου `boolean`.
- Οι τελεστές `&&` και `||` υλοποιούν αντίστοιχα τις πράξεις της λογικής σύζευξης και διάζευξης. Τα ορίσματα πρέπει να είναι τύπου `boolean` 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 ορίσματος, το δεύτερο όρισμα δεν αποτιμάται καθόλου.

Οι τελεστές ανάθεσης είναι οι ακόλουθοι:

- Ο τελεστής = αναθέτει την τιμή του δεύτερου τελούμενου στο πρώτο. Το πρώτο το τελούμενο πρέπει να είναι μια μεταβλητή οποιουδήποτε έγκυρου τύπου t, ενώ το δεύτερο τελούμενο μπορεί να είναι μια τιμή ή μεταβλητή τύπου t. Επιπλέον το δεύτερο τελούμενο μπορεί να είναι μια συνάρτηση που επιστρέφει μια τιμή τύπου t.
- Ο τελεστής `op=`, όπου `op` in {`+`, `-`, `*`, `/`, `%`}, συνδυάζει υπολογισμό με ανάθεση. Η έκφραση `k op=a;` είναι σημασιολογικά ισοδύναμη με την `k=k op a;` με την διαφορά ότι το `k` θα υπολογιστεί μόνο μια φορά.
- Οι τελεστές ++ και −− δέχονται ένα τελούμενο και υπάρχουν σε δύο μορφές, μια prefix και μια postfix. Το τελούμενο πρέπει να είναι μια μεταβλητή οποιουδήποτε έγκυρου τύπου t, ενώ το αποτέλεσμα είναι του ίδιου τύπου t. Οι τελεστές προκαλούν την αύξηση ή την μείωση της τιμής του τελούμενου. Στην περίπτωση της prefix το αποτέλεσμα που υπολογίζεται είναι η τιμή του τελούμενου μετά την αύξηση ή την μείωση, ενώ στην περίπτωση της postfix το αποτέλεσμα που υπολογίζεται είναι η τιμή του τελούμενου πριν την αύξηση ή την μείωση.

Στον Πίνακα 2 ορίζεται η προτεραιότητα και η προσεταιριστικότητα των τελεστών της Floop2009:

Τελεστές	**Περιγραφή**	**Αριθμός ορισμάτων**	**Θέση και προσεταιρι-στικότητα**
`[ ]()`	Αναφορά σε στοιχείο πίνακα, κλήση συνάρτησης, if (boolean expression), for (boolean expression)	2	ειδική
`++ --`	Αύξηση, μείωση	1	prefix, postfix
`+ -`	Πρόσημα	1	prefix
`!`	Λογική άρνηση	1	prefix
`* / %`	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
`= += -= /= %= *=`	Τελεστές ανάθεσης	2	infix, δεξιά

Πίνακας 2 Προτεραιότητα και προσεταιριστικότητα των τελεστών της Floop2009

2.5 Εντολές

Οι εντολές που υποστηρίζει η γλώσσα Floop2009 είναι οι ακόλουθες:

- Η κενή εντολή, που δεν κάνει καμία ενέργεια.
- Η εντολή ανάθεσης `l = e`, όπου `l` είναι τύπου t και `e` μια έκφραση τύπου t'. Ο τύπος t' πρέπει να είναι συμβατός για ανάθεση με τον τύπο t. Αυτή η σχέση συμβατότητας, που πρέπει να τονιστεί ότι δεν είναι συμμετρική, ορίζεται ως εξής:
  - Κάθε πλήρης τύπος είναι συμβατός για ανάθεση με τον εαυτό του.
  - Ο τύπος `integer` είναι συμβατός για ανάθεση με τον τύπο `float`.
- Η σύνθετη εντολή, που αποτελείται από μια σειρά έγκυρων εντολών χωρισμένων με το διαχωριστή `;`, ανάμεσα σε άγκιστρα `{}`. Οι εντολές αυτές εκτελούνται διαδοχικά, εκτός αν κάποια από αυτές είναι εντολή άλματος.
- Η εντολή ελέγχου `if ( e ) s1 else s2`. Η έκφραση `e` πρέπει να έχει τύπο boolean και τα `s1, s2` να είναι έγκυρες εντολές. Το τμήμα `else` είναι προαιρετικό.
- Η εντολή βρόχου `while ( e ) s`. Η έκφραση `e` πρέπει να έχει τύπο `boolean` και το `s` να είναι έγκυρη εντολή.
- Η εντολή βρόχου `for(e1;e2;e3) s`. Οι εκφράσεις `e1,e2,e3` είναι προαιρετικές. Αν δίνεται η έκφραση `e2` πρέπει να έχει τύπο `boolean`, διαφορετικά θεωρείται ίση με `true`. Η σημασιολογία της εντολή βρόχου `for` είναι όπως ακριβώς και στη C++.
- Η εντολή με ετικέτα `I:s`, όπου `I` το όνομα μιας ετικέτας και `s` μια έγκυρη εντολή. Κάθε ετικέτα πρέπει να ορίζεται το πολύ μια φορά στη σύνθετη εντολή που ορίζει το σώμα μιας συγκεκριμένης δομικής μονάδας.
- Η εντολή άλματος `goto I`, όπου `I` το όνομα μιας ετικέτας που πρέπει να εμφανίζεται στην ίδια δομική μονάδα. Πέραν αυτού, δεν υπάρχουν άλλοι περιορισμοί ως προς τη θέση των εντολών αλμάτων ή των ετικετών όπου αυτά οδηγούν.
- Η εντολή `break`, που σε βγάζει από τον πιο εσωτερικό βρόχο όπως ακριβώς και στη C++.
- Η εντολή `continue` προκαλεί την συνέχιση του βρόχου μέσα στον οποίο βρίσκεται όπως ακριβώς και στη C++.
- Η εντολή άλματος `return e;` που τερματίζει την εκτέλεση της τρέχουσας συνάρτησης και επιστρέφει την τιμή e ως αποτέλεσμα της συνάρτησης. Αν η τρέχουσα συνάρτηση έχει ως τύπο αποτελέσματος void τότε η έκφραση `return e;` θα πρέπει να παραλείπεται. Διαφορετικά η έκφραση `e` θα πρέπει να έχει τον ίδιο τύπο με τον τύπο αποτελέσματος της συνάρτησης.

- Η κλήση μιας συνάρτησης.

2.6 Βιβλιοθήκη έτοιμων συναρτήσεων

Η Floop2009 υποστηρίζει ένα σύνολο προκαθορισμένων υποπρογραμμάτων, τα οποία βρίσκονται στη διάθεση του προγραμματιστή. Τα υποπρογράμματα αυτά είναι ορατά σε κάθε δομική μονάδα, εκτός αν επισκιάζονται από μεταβλητές, παραμέτρους ή υποπρογράμματα με το ίδιο όνομα. Παρακάτω δίνονται οι επικεφαλίδες τους, όπως θα γράφονταν αν τα ορίζαμε σε ένα πρόγραμμα Floop2009. Επίσης, εξηγείται η λειτουργία τους.

2.6.1 Είσοδος και έξοδος

```
void writeInteger (integer a);
void writeBoolean (boolean b);
void writeChar    (char c);
void writeReal    (float d);
void writeString  (char a[]);
```

Οι διαδικασίες αυτές χρησιμοποιούνται για την εκτύπωση τιμών που ανήκουν στους βασικούς τύπους της Floop2009, καθώς και για την εκτύπωση συμβολοσειρών.

```
integer readInteger();
boolean readBoolean ();
char readChar    ();
float readReal    ();
void readString (integer size, char string[]);
```

Αντίστοιχα, τα παραπάνω υποπρογράμματα χρησιμοποιούνται για την εισαγωγή τιμών που ανήκουν στους βασικούς τύπους της Floop2009 και για την εισαγωγή συμβολοσειρών. Η διαδικασία `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.

2.7 Κλάσεις

2.7.1 Ορισμός κλάσεων

Η Floop2009 είναι αντικειμενοστρεφής γλώσσα όπως και η C++. Μια κλάση στη Floop2009 ορίζεται με το keyword `class` ακολουθούμενο από το όνομα και το σώμα της κλάσης. Το σώμα μιας κλάσης πρέπει να βρίσκεται μέσα σε άγκιστρα. Οπότε η μορφή ορισμού μια κλάσης είναι η ακόλουθη:

```
class  name{
      body
}
```

Το σώμα μιας κλάσης μπορεί να περιέχει δηλώσεις μεταβλητών και/ή ορισμούς συναρτήσεων. Οι δηλώσεις των μεταβλητών και συναρτήσεων έχουν την ίδια σύνταξη όπως πριν με την προσθήκη ενός keyword (`private` ή `public`) στην αρχή. Τα keywords `private` και `public` έχουν την ίδια σημασιολογία όπως και στην C++. Οπότε αν μια μεταβλητή ή μια συνάρτηση μιας κλάσης έχει δηλωθεί/ορισθεί με το keyword `public` μπορεί να χρησιμοποιηθεί και έξω από το σώμα της κλάσης. Αν μια μεταβλητή ή μια συνάρτηση μιας κλάσης έχει δηλωθεί/ορισθεί με το keyword `private` δεν μπορεί να χρησιμοποιηθεί και έξω από το σώμα της κλάσης. Προαιρετικά το σώμα μιας κλάσης μπορεί να περιέχει ένα ή περισσότερους constructors όπως και στην C++ μια συνάρτηση δηλαδή που έχει ίδιο όνομα με την κλάση και καλείται όταν ορίζεται ένα στιγμιότυπο της κλάσης. Κάθε constructor είναι public οπότε δεν χρειάζεται να προηγείται το keyword `public` πριν τον ορισμό του. Οι συναρτήσεις-μέλη των κλάσεων επιτρέπεται να ορίζονται μόνο μέσα στον ορισμό της κλάσης, ταυτόχρονα με τη δήλωσή τους. Ακολουθεί ένα παράδειγμα ορισμού μιας κλάσης:

```
class  foo{
       private integer i,j;
       public k;
       private void add(integer a){
             (* body of add *)
       }
       public integer sasa(){
             (* body of sasa *)
       }
       foo(){
             (* body of foo *)
       }
       foo(integer i){
             (* body of foo *)
       }
}
```

2.7.2 Ορισμός στιγμιότυπων κλάσεων

Για να χρησιμοποιηθεί μια κλάση αρκεί να οριστεί μια μεταβλητή αυτού του τύπου, και στη συνέχεια να δημιουργηθεί ένα καινούριο στιγμιότυπο με το keyword `new`, για παράδειγμα

```
foo myclass ;
myclass  = new foo();
```

ή

```
foo myclass ;
myclass = new foo(34);
```

Για να προσπελαστούν οι public μεταβλητές και συναρτήσεις έξω από το σώμα μιας κλάσης χρησιμοποιείται ο τελεστής `.` (τελεία), για παράδειγμα

```
myclass.k;
myclass.sasa();
```

2.7.3 Κληρονομικότητα

Το keyword `extends` υλοποιεί την κληρονομικότητα όπως την υλοποιεί στη C++ η έκφραση `:public` . Για παράδειγμα:

```
class faa extends foo{
      (* body *)
}
```

Για να αναφερθεί κάποιο αντικείμενο μιας υποκλάσης (π.χ. της faa) σε μια συνάρτηση ή ένα δομικό στοιχείο μιας υπερκλάσης της (π.χ. της foo) χρησιμοποιείται ο τελεστής επίλυσης εμβέλειας ::

3 Παραδείγματα προγραμμάτων της Floop2009

3.1 Hello World!

```
void main() {
      writeString("Hello World!\n")
}
```

3.2 Πρώτοι αριθμοί

Το παρακάτω παράδειγμα προγράμματος στη γλώσσα Floop2009 είναι ένα πρόγραμμα που υπολογίζει τους πρώτους αριθμούς μεταξύ **1** και **n**, όπου το **n** καθορίζεται από το χρήστη. 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.

```
boolean prime(integer n){
      integer i;
      boolean isPrime, result;
      if (n < 0)
            result = prime(-n);
      else if (n < 2)
            result = false;
      else if (n == 2)
            result = true;
      else if (n % 2 == 0)
            result = false;
      else {
            i = 3;
            isPrime = true;
            while ( isPrime && i <= n / 2 ) {
                  isPrime = (n % i==0);
                  i = i+2;
            }
            result = isPrime;
      }
      return result;
```

```
 }


main( ){
       integer limit, number, counter;

       limit = readInteger();
       counter = 0;
       if (limit >= 2) {
            counter = counter + 1;
            writeInteger(2);
       }
       if (limit >= 3) {
            counter = counter + 1;
            writeInteger(3);
       }
       number = 6;
       while (number <= limit) {
            if (prime(number-1)) {
                 counter = counter + 1;
                 writeInteger(number-1);
            }
            if ((number != limit) && prime(number+1))
            {
                 counter = counter + 1;
                 writeInteger(number+1);
            }
            number = number + 6;
       }
       writeChar('\n');
       writeInteger(counter);
}
```

3.3 Κλάσεις

```
Class foo{
       private Integer sa;
       foo(integer k){
            sa=2*k;
       }
       foo(integer k, integer n){
            sa=(2*k)*(n+12);
       }
       public integer getSA(){
            return sa;
       }
}

void main(){
       integer p;
       foo test,test1;
       test = new foo(5);
       test1 = new foo(5,8);
```

```
        if (test.getSA() > test1.getSA)
              p=test.getSA();
        else p=test1.getSA();
        writeInteger(p);
}
```

4 Επίλογος

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