ΠΑΡΑΔΕΙΓΜΑ ΣΥΝΤΑΚΤΙΚΟΥ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Ορίζουμε παρακάτω τη γλώσσα προγραμματισμού C-minus (C-).

Α. Λεκτικές Συμβάσεις της C-

1. Οι λέξεις-κλειδιά της γλώσσας είναι οι ακόλουθες:

   ```
   else if int return void while
   ```

2. Τα ειδικά σύμβολα της γλώσσας είναι:

   ```
   + - * / < <= > >= == != = ; , ( ) [ ] { } /* */
   ```

3. Άλλες λεκτικές μονάδες (tokens) της γλώσσας είναι οι `ID` και `NUM` που ορίζονται από τις ακόλουθες κανονικές εκφράσεις:

   ```
   ID = letter letter*
   NUM = digit digit*
   letter = a|..|z|A|..|Z
   digit = 0|..|9
   ```

   Τα κεφαλαία και τα μικρά γράμματα είναι διαφορετικά μεταξύ τους.

4. Στα προγράμματα της C- επιτρέπεται η ύπαρξη κενών, χαρακτήρων νέας γραμμής (newlines), και tabs. Οι χαρακτήρες αυτοί διαχωρίζουν τις λέξεις-κλειδιά και τα tokens, αλλά κατά τα άλλα αγνοούνται από το μεταγλωττιστή.

5. Τα σχόλια περικλείονται από `/*` και `*/`. Τα σχόλια μπορούν να επεκτείνονται σε περισσότερες από μία γραμμές αλλά δεν μπορούν να είναι φωλιασμένα (nested). Επίσης, σχόλια δεν μπορούν να εμφανιστούν μέσα σε tokens.

Β. Συντακτικό και Σημασιολογία της C-

Μια BNF γραμματική γιά τη C- είναι η ακόλουθη:

1. *program* → *prototypes declaration-list*
2. *prototypes* → *prototype prototypes* | *prototype*
3. *prototype* → *type-specifier* `ID ( params );`

4. *declaration-list* → *declaration-list declaration* | *declaration*
5. *declaration* → *var-declaration* | *fun-declaration*
6. *var-declaration* → *type-specifier* `ID ;` | *type-specifier* `ID [ NUM ] ;`
7. *type-specifier* → `int` | `void`
8. *fun-declaration* → *type-specifier* `ID (` *params* `)` *compound-stmt*
9. *params* → *param-list* | `void`
10. *param-list* → *param-list* `,` *param* | *param*
11. *param* → *type-specifier* `ID` | *type-specifier* `ID [ ]`
12. *compound-stmt* → { *local-declarations statement-list* }
13. *local-declarations* → *local-declarations var-declaration* | *empty*
14. *statement-list* → *statement-list statement* | *empty*
15. *statement* → *expression-stmt* | *compound-stmt* | *selection-stmt* | *iteration-stmt* | *return-stmt*
16. *expression-stmt* → *expression* `;` | `;`
17. *selection-stmt* → `if (` *expression* `)` *statement* | `if (` *expression* `)` *statement* `else` *statement*
18. *iteration-stmt* → `while (` *expression* `)` *statement*
19. *return-stmt* → `return ;` | `return` *expression* `;`
20. *expression* → *var* `=` *expression* | *simple-expression*
21. *var* → `ID` | `ID [` *expression* `]`
22. *simple-expression* → *additive-expression relop additive-expression* | *additive-expression*
23. *relop* → `<=` | `<` | `>` | `>=` | `==` | `!=`
24. *additive-expression* → *additive-expression addop term* | *term*
25. *addop* → `+` | `-`
26. *term* → *term mulop factor* | *factor*
27. *mulop* → `*` | `/`
28. *factor* → `(` *expression* `)` | *var* | *call* | `NUM`
29. *call* → `ID (` *args* `)`
30. *args* → *arg-list* | *empty*

31. *arg-list* → *arg-list* , *expression* | *expression*

Στη συνέχεια δίνουμε κάποιες επεξηγήσεις για τη σημασιολογία των διαφόρων χαρακτηριστικών της γλώσσας.

Κανόνες 1, 2, 3, 4, 5: Ένα πρόγραμμα αποτελείται από μιά ακολουθία πρωτοτύπων (prototypes) ακολουθούμενα από δηλώσεις συναρτήσεων ή μεταβλητών. Κάθε συνάρτηση που ορίζεται στο πρόγραμμα πρέπει να έχει ένα πρωτότυπο (prototype). Όλες οι μεταβλητές πρέπει να έχουν δηλωθεί πριν χρησιμοποιηθούν. Η τελευταία δήλωση σε ένα πρόγραμμα πρέπει να είναι η δήλωση μιας συνάρτησης με το όνομα `main`.

Κανόνες 6, 7: Μια δήλωση μεταβλητής αφορά είτε μια απλή μεταβλητή ακέραιου τύπου είτε ένα array ακεραίων του οποίου οι δείκτες παίρνουν τιμές από 0 μέχρι `NUM`-1. Στη C- οι μόνοι βασικοί τύποι είναι integer και void. Όταν δηλώνεται μια μεταβλητή, μόνο ο τύπος `int` μπορεί να χρησιμοποιηθεί. Ο τύπος `void` είναι μόνο για δηλώσεις συναρτήσεων. Αξίζει να σημειώσουμε ότι σε κάθε δήλωση μόνο μια μεταβλητή μπορεί να δηλωθεί.

Κανόνες 8, 9, 10, 11: Η δήλωση μιας συνάρτησης αποτελείται από τον τύπο επιστροφής, από το όνομα της συνάρτησης, από τη λίστα των τυπικών παραμέτρων μέσα σε παρενθέσεις, και από τον κώδικα της συνάρτησης. Αν ο τύπος επιστροφής είναι `void`, τότε η συνάρτηση δεν επιστρέφει τιμή. Οι παράμετροι της συνάρτησης μπορεί να είναι `void` ή μια λίστα που αναπαριστά τις παραμέτρους της συνάρτησης. Οι παράμετροι που ακολουθούνται από `[]` είναι arrays των οποίων το μέγεθος μπορεί να ποικίλει. Η δήλωση μιας συνάρτησης πρέπει να συμφωνεί με το πρωτότυπό της. Δεν επιτρέπεται να υπάρχουν συναρτήσεις που να έχουν το ίδιο όνομα. Η εμβέλεια των παραμέτρων μιας συνάρτησης είναι το σώμα της συνάρτησης. Οι συναρτήσεις μπορεί να είναι αναδρομικές.

Κανόνας 12: Μια σύνθετη εντολή εκτελείται εκτελώντας μια-μια με τη σειρά της εντολές που την απαρτίζουν. Οι τοπικές δηλώσεις έχουν εμβέλεια που εκτείνεται στη λίστα των εντολών (*statement-list*). Η εμβέλεια αυτή υπερκαλύπτει οποιεσδήποτε άλλες ολικές (global) δηλώσεις.

Κανόνες 13, 14: Το μη τερματικό σύμβολο `empty` είναι το γνωστό μας ϵ.

Κανόνες 15, 16: Μια έκφραση-εντολή (*expression-stmt*) συνήθως αποτιμάται για τις παρενέργειες (side-effects) που επιφέρει (για παράδειγμα, μια εντολή ανάθεσης ή η κλήση μιας συνάρτησης).

Κανόνας 17: Το `if-else` έχει τη γνωστή σημασιολογία. Ο κανόνας αυτός της γραμματικής περιέχει το γνωστό dangling-else πρόβλημα το οποίο ξεπερνιέται με τον "most closely nested" κανόνα.

Κανόνας 18: Το `while` έχει τη γνωστή σημασιολογία (όπως και στη C).

Κανόνας 19: Οι συναρτήσεις που δεν έχουν δηλωθεί ως `void` πρέπει να επιστρέφουν μια τιμή με το `return`. Το `return` επιστρέφει τον έλεγχο στο σημείο που κλήθηκε η συνάρτηση (ή τερματίζει το πρόγραμμα αν εκτελεστεί μέσα από το `main`).

Κανόνες 20, 21: Η εντολή ανάθεσης έχει τη γνωστή σημασιολογία. Η τιμή που ανατίθεται στη

μεταβλητή, επιστρέφει και ως τιμή ολόκληρης της έκφρασης. Η μεταβλητή *var* μπορεί να είναι είτε μια απλή ακέραια μεταβλητή είτε μια μεταβλητή που αντιστοιχεί σε κάποια θέση ενός array.

Κανόνες 22-28: Οι κανόνες αυτοί αφορούν εκφράσεις των οποίων η σημασιολογία είναι η γνωστή. Το σύμβολο / αναπαριστά τη διαίρεση κατά την οποία το υπόλοιπο δεν λαμβάνεται υπόψιν. Το *factor* μπορεί να είναι και κλήση συνάρτησης.

Κανόνες 29-31: Στην κλήση συναρτήσεων, ο αριθμός και ο τύπος των παραμέτρων που χρησιμοποιούνται πρέπει να είναι ίδια με αυτά που υπάρχουν στον ορισμό της συνάρτησης. Ειδικώτερα, όταν μια παράμετρος μιας συνάρτησης είναι τύπου array, τότε στην κλήση της συνάρτησης αυτής πρέπει να εμφανίζεται μια μεταβλητή που να αντιστοιχεί στό όνομα ενός array.

Οι παραπάνω κανόνες δεν υποστηρίζουν εντολές για input-output. Για το σκοπό αυτό υποθέτουμε την ύπαρξη δύο συναρτήσεων τις οποίες θα τις θεωρούμε σα να είναι ορισμένες σε κάθε πρόγραμμα που γράφουμε:

```
int input(void){...}
void output(int x){...}
```

Οι δύο αυτές συναρτήσεις θα διαβάζουν και θα γράφουν στο standard input και στο standard output αντίστοιχα.

Γ. Παράδειγμα Προγράμματος

Το παρακάτω είναι ένα παράδειγμα προγράμματος στη C-:

```
/* A program that performs Euclid's Algorithm
   to compute the gcd of two numbers */
int gcd (int a, int b);
void main(void);

int gcd (int u, int v)
{ if (v == 0) return u ;
  else return gcd(v, u-u/v*v);
    /* u - u/v*v == u mod v */
 }

 void main(void)
 { int x; int y;
   x = input(); y = input();
   output(gcd(x,y));
 }
```