Εισαγωγή

Η γλώσσα Uni-C είναι ένα μικρό υποσύνολο βασισμένο στην διάσημη γλώσσα προγραμματισμού C**.** Η C είναι μια γενικού σκοπού διαδικαστική (procedural) γλώσσα προγραμματισμού.

Η Uni-C σχεδιάστηκε για να χρησιμοποιηθεί αποκλειστικά στο πλαίσιο του εργαστηρίου Μεταγλωττιστών και τα χαρακτηριστικά της περιγράφονται αναλυτικά στα ακόλουθα δύο μέρη του παρόντος εγγράφου.

ΜΕΡΟΣ 1ο: Λεκτική Ανάλυση

Ένα πηγαίο πρόγραμμα Uni-C διαβάζεται από έναν **λεκτικό αναλυτή – ΛΑ (lexical analyzer)** ως μια μεγάλη συμβολοσειρά εισόδου. Ο ΛΑ διαχωρίζει ένα-ένα τα λεξήματα που περιέχονται μέσα στη συμβολοσειρά εισόδου και τα αναγνωρίζει ως λεκτικές μονάδες κάθε φορά που του στέλνει ένα αίτημα ο **συντακτικός αναλυτής – ΣΑ (parser)**. Αυτό το κεφάλαιο περιγράφει πώς ο λεκτικός αναλυτής πραγματοποιεί τον διαχωρισμό και την αναγνώριση των λεξημάτων του πηγαίου κώδικα.

1.1 Το αλφάβητο

Το αλφάβητο της γλώσσας αποτελείται από τους ακόλουθους ASCII χαρακτήρες:

Πεζοί και κεφαλαίοι λατινικοί χαρακτήρες: a-z A-Z

Αριθμητικά ψηφία: 0–9

Ειδικοί χαρακτήρες: ! " # % & ' ( ) * + , - . / : ; < = > ? [\ ] ^ _ { | } ~

Whitespace χαρακτήρες: κενό (space \s), tab (\t), νέα-γραμμή (NEWLINE \n), τέλος-αρχείου (EOF)

Η γλώσσα κάνει διάκριση μεταξύ πεζών και κεφαλαίων (case sensitive language).

1.2 Διαχωριστές μεταξύ λέξεων

Εκτός από την αρχή μιας γραμμής, ή ενός string, οι διαχωριστικοί χαρακτήρες, δηλ. ο κενός χαρακτήρας, το tab και το ελληνικό ερωτηματικό (semi-colon), μπορούν να χρησιμοποιηθούν σε οποιοδήποτε συνδυασμό για να διαχωρίσουν τα τις λέξεις (λεξήματα-lexemes) μέσα στο πηγαίο πρόγραμμα. Το κενό είναι απαραίτητο μεταξύ δυο λεξημάτων μόνο όταν η ένωσή τους θα μπορούσε να αποτελέσει ένα άλλο λέξημα (π.χ., το `ab` είναι ένα λέξημα, αλλά τα `a` και `b` είναι δυο λεξήματα ενώ το `month <= 12` μπορεί να γραφεί και χωρίς τα ενδιάμεσα κενά του ως month<=12).

1.3 Λεκτικές μονάδες (tokens)

Όταν οι πηγαίες λέξεις διαχωρίζονται και στη συνέχεια αναγνωρίζονται επιτυχώς, ικανοποιώντας ένα πρότυπο αναγνώρισης, προσδιορίζουν μια λεκτική μονάδα (token) που περιγράφεται από το αναγνωριστικό της λέξης και άλλες πληροφορίες που την αφορούν (βλέπε θεωρία).

Βασικές κατηγορίες αναγνωριστικών που χρησιμοποιούνται στην αναγνώριση προτύπων στην Uni-C είναι οι ακόλουθες: αναγνωριστικά ή ονόματα (*identifiers ή names)*, λέξεις κλειδιά (*keywords)*, κυριολεκτικές συμβολοσειρές (*strings)* και τελεστές (*operators)*. Η συμβολοσειρά '\n' αναγνωρίζεται ως NEWLINE, ενώ το end-of-file ως EOF. Όσες οι πηγαίες λέξεις αναγνωρίζονται με τις παραπάνω κατηγορίες επιστρέφονται ως tokens με τα αντίστοιχα αναγνωριστικά ονόματα

Οι διαχωριστές (ακολουθίες κενών και ακολουθίες tabs) που χρησιμοποιούνται για το διαχωρισμό των λεξημάτων και τα σχόλια αναγνωρίζονται από πρότυπα αλλά δεν επιστρέφονται ως tokens, ενώ άλλοι χαρακτήρες διαχωρισμού που χρησιμοποιούνται για να διαχωρίζουν δομές σύνταξης, αναγνωρίζονται κανονικά και επιστρέφονται ως tokens με ένα όνομα αναγνωριστικού όπως το *delimiter*, π.χ. ο χαρακτήρας ';' στον κώδικα a = (b+c); θα μπορούσε να επιστραφεί ως *delimiter(;).*

Όπου υπάρχει ασάφεια κατά το διαχωρισμό λέξεων, ο διαχωρισμός περιλαμβάνει την μακρύτερη πιθανή συμβολοσειρά που σχηματίζει ένα αποδεκτό token όταν η συμβολοσειρά εισόδου διαβάζεται (κατά το διαχωρισμό της) από αριστερά προς τα δεξιά.

1.3.1 Αναγνωριστικά (ονόματα)

Τα αναγνωριστικά ή ονόματα (*identifiers* ή *names*) της Uni-C περιγράφονται από ένα πρότυπο αναγνώρισης με βάση τους λεκτικούς προσδιορισμούς που ακολουθούν.

Αναγνωρίζονται και επιστρέφονται ως tokens με το αναγνωριστικό όνομα ***identifier***:

Τα λεξήματα αυτής της κατηγορίας αποτελούνται από έναν ή περισσότερους κεφαλαίους ή/και μικρούς λατινικούς χαρακτήρες `a` έως `z` και `A` έως `Z`, ή/και από το underscore _ και, εκτός από τον αρχικό χαρακτήρα, από τα ψηφία `0` έως `9`.

Οι identifiers είναι θεωρητικά απεριόριστοι σε μήκος. Πρακτικά όμως μπορείτε να ορίσετε αυθαίρετα ένα εύλογο μήκος που να σας εξυπηρετεί στην υλοποίησή σας (πχ. 32 χαρακτήρες)

1.3.2 Λέξεις κλειδιά

Στην Uni-C, τα ακόλουθα ονόματα χρησιμοποιούνται ως λέξεις κλειδιά (*reserved words* ή *keywords)* και δεν απαιτούν πρότυπο αναγνώρισης γιατί είναι ήδη καταχωρημένα μέσα στον πίνακα συμβόλων, όμως επιστρέφονται κανονικά ως tokens με αναγνωριστικό όνομα ***keyword***, π.χ. keyword(if).

```
break   case    func    const   continue
do      double  else    float   for
if      int     long    return  short
sizeof  struct  switch  void    while
```

1.3.3 Λεκτικά κυριολεκτικά

Τα λεκτικά κυριολεκτικά ή απλώς συμβολοσειρές (*string literals* ή *strings*) περικλείονται μέσα σε διπλές " αποστρόφους και περιλαμβάνουν οποιονδήποτε χαρακτήρα εκτός του backslash \, της νέας γραμμής \n ή της διπλής αποστρόφου " που για τη χρήση τους απαιτούν χρήση σειράς διαφυγής.

Αναγνωρίζονται και επιστρέφονται ως tokens με το αναγνωριστικό όνομα ***string***.

Αναγνωρισμένοι συνδυασμοί σειρών διαφυγής (escape sequence) μέσα στα strings είναι:

Escape Sequence	Meaning	Notes
\\	Backslash (\)	
\"	Double quote (")	
\n	ASCII Linefeed (LF)	NEWLINE

Παραδείγματα:

```
""
"Test"
"Mark said, \"Boo!\"\n"
```

1.3.4 Αριθμητικά κυριολεκτικά

Τα αριθμητικά κυριολεκτικά (*numerical literals*) περιέχουν ακεραίους αριθμούς (*integers*) και αριθμούς κινητής υποδιαστολής (*floating point numbers*).

Σημειώστε ότι τα αριθμητικά δεν περιλαμβάνουν πρόσημο: η έκφραση -1 στην ουσία είναι σύνθεση του τελεστή '-' και του αριθμητικού 1.

1.3.4.1 Ακέραιοι

Οι ακέραιοι περιλαμβάνουν δεκαδικούς, δεκαεξαδικούς και οκταδικούς ακεραίους και περιγράφονται ως εξής: Ένας δεκαδικός ακέραιος ξεκινάει με ΜΗ μηδενικό ψηφίο (1-9) και ακολουθεί προαιρετικά οποιοσδήποτε αριθμός ψηφίων 0-9. Ένας δεκαεξαδικός ακέραιος ξεκινάει με 0x ή 0X και ακολουθούν ένα ή περισσότερα δεκαεξαδικά (0-9, A-F) ψηφία. Ένας οκταδικός ακέραιος ξεκινάει με 0 και ακολουθεί ένα ή περισσότερα οκταδικά (0-7) ψηφία.

Σημειώστε ότι το 0 αποτελεί τη μόνη εξαίρεση δεκαδικού ακεραίου που ξεκινάει με 0.

Αναγνωρίζονται και επιστρέφονται ως tokens με το αναγνωριστικό όνομα ***integer*** π.χ. integer(3).

Παραδείγματα **δεκαδικών**, **δεκαεξαδικών** και **οκταδικών** ακεραίων:

```
0       3       214748          0x4F
0XFF42  063     0147            00
```

1.3.4.2 Αριθμοί κινούμενης υποδιαστολής

Τα κυριολεκτικά κινητής υποδιαστολής (*floating point literals*) περιγράφονται ως εξής: Ένας δεκαδικός αριθμός αποτελείται από το ακέραιο μέρος και το δεκαδικό μέρος που διαχωρίζονται με μία τελεία '.'. Τόσο το ακέραιο μέρος όσο και το δεκαδικό ορίζονται σύμφωνα με τους κανόνες που περιγράφηκαν παραπάνω για τους απλούς ακεραίους του δεκαδικού αριθμητικού συστήματος. Υπάρχει επίσης η δυνατότητα ορισμού δυνάμεων με χρήση του χαρακτήρα 'e' ή 'E'. Στην περίπτωση αυτή το ακέραιο ή/και δεκαδικό μέρος υψώνεται στην ακέραια δύναμη που ακολουθεί μετά τον χαρακτήρα 'e' ή 'E'.

Αναγνωρίζονται και επιστρέφονται ως tokens με το αναγνωριστικό όνομα ***float.***

Μερικά παραδείγματα αριθμών κινητής υποδιαστολής:

```
3.14    10.0    0.001    1e100    3.14e-10    0e0
```

1.3.5 Τελεστές

Τα παρακάτω tokens αναγνωρίζονται στην Uni-C ως τελεστές:

- αριθμητικοί: +, -, *, /, %
- ανάθεσης: =
- επαυξημένες ανάθεσης: +=, -=, *=, /=
- boolean λογική: !, &&, ||
- έλεγχος ισότητας: ==, !=
- προσαύξηση και μείωση κατά ένα: ++, --
- order relations: <, <=, >, >=
- διεύθυνση μνήμης: &

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

Βρίσκονται καταχωρημένοι μέσα στον πίνακα συμβόλων χωρίς να χρειάζονται πρότυπο αναγνώρισης. Επιστρέφονται ως tokens με το αναγνωριστικό όνομα ***operator,*** π.χ. operator(<=).

1.4 Σχόλια

Στην Uni-C υπάρχουν δύο ειδών σχόλια: τα σχόλια μιας γραμμής και τα σχόλια πολλών γραμμών. Ένα σχόλιο μιας γραμμής αρχίζει με δύο χαρακτήρες '/' στη σειρά (δηλαδή //), που δεν αποτελούν μέρος της συμβολοσειράς του σχολίου, και ολοκληρώνεται με το τέλος της γραμμής (\n). Ένα σχόλιο πολλών γραμμών ξεκινάει με τους χαρακτήρες /* και ολοκληρώνεται μονάχα με τους χαρακτήρες */ ανεξάρτητα με τον αριθμό των γραμμών που έχουν μεσολαβήσει.

Αναγνωρίζονται βάσει προτύπου με το αναγνωριστικό όνομα ***comment*** αλλά **ΔΕΝ επιστρέφονται** ως tokens (αγνοούνται από τον συντακτικό αναλυτή).

1.5 White_spaces χαρακτήρες

Οι ακολουθίες διαχωριστικών χαρακτήρων (κενά, tabs ή συνδυασμοί τους), που χρησιμοποιούνται για το διαχωρισμό των λεξημάτων, αναγνωρίζονται από πρότυπα με το αναγνωριστικό όνομα ***white_spaces*** αλλά δεν επιστρέφονται ως tokens (αγνοούνται από τον συντακτικό αναλυτή).

Μέρος 2ο: Συντακτική Ανάλυση

Ο ρόλος του συντακτικού αναλυτή – ΣΑ (syntax analyser-parser) είναι να διαπιστώσει τη συντακτική ορθότητα των δομών του πηγαίου κώδικα βάσει της γραμματικής της γλώσσας. Η είσοδος στον συντακτικό αναλυτή - ΣΑ είναι μια ακολουθία από λεκτικές μονάδες (***tokens***), που παράγονται από τον λεκτικό αναλυτή - ΛΑ (lexical analyzer-scanner), μετά το διαχωρισμό και την αναγνώριση των λεξημάτων της συμβολοσειράς εισόδου. Τα tokens επιστρέφονται από τον ΛΑ ένα-ένα μετά από συνεχή αιτήματα του ΣΑ κατά τη διάρκεια των ελέγχων που πραγματοποιεί. Πολλές φορές, αντί λεκτικών μονάδων ο ΣΑ ζητά από τον ΛΑ να εντοπίσει συγκεκριμένες λέξεις που στη γραμματική που ακολουθεί ο ΣΑ καλούνται με τον γενικό όρο ***atoms.*** Παράδειγμα atom αποτελεί η συμβολοσειρά '<=' όταν ο ΣΑ ζητά από τον ΛΑ να εντοπίσει την ακολουθία χαρακτήρων **<=** αντί να του ζητήσει η επόμενη λέξη να είναι το token **operator(<=)**. Συνήθως η γραμματική χρησιμοποιεί atoms αντί tokens όταν πρόκειται για κάποια λέξη της οποίας η αναγνώριση δεν απαιτεί πρότυπο (λέξεις-κλειδιά, τελεστές, διαχωριστές κλπ).

Αυτό το κεφάλαιο περιγράφει τη γραμματική για τη δομή ενός πηγαίου κώδικα και τη σύνταξη των ***εκφράσεων (expressions)*** της γλώσσας Uni-C. Οι εκφράσεις μέσα στον πηγαίο κώδικα χρησιμοποιούνται για να δηλωθεί ένα αναγνωριστικό, να υπολογιστεί μια αξία, ή για να κληθεί μια συνάρτηση (function). Οι εκφράσεις που θα περιγραφούν αποτελούν ένα μικρό υποσύνολο των εκφράσεων της γλώσσας C σε απλοποιημένη μορφή.

2.1 Δομή πηγαίου κώδικα

2.1.1 Λογικές γραμμές

Ο κώδικας ενός προγράμματος Uni-C αποτελείται από λογικές γραμμές. Κάθε έκφραση του προγράμματος πρέπει να περιορίζεται μέσα σε μια λογική γραμμή. Κάθε λογική γραμμή, ολοκληρώνεται με τον χαρακτήρα semi-colon (;) και δομείται από μια ή περισσότερες φυσικές γραμμές που συνδέονται μεταξύ τους βάσει συντακτικών κανόνων σύνδεσης.

2.1.2 Φυσικές γραμμές

Μια φυσική γραμμή του πηγαίου αρχείου αποτελείται από μια ακολουθία χαρακτήρων που ολοκληρώνεται με τον ειδικό χαρακτήρα NEWLINE.

2.1.2.1 Κανόνες σύνδεσης φυσικών γραμμών

Δυο φυσικές γραμμές συνδέονται μεταξύ τους με τον ειδικό χαρακτήρα backslash (\) στο τέλος της πρώτης γραμμής.

Μια γραμμή που περιέχει σχόλιο δεν επιτρέπεται να περιέχει το backslash ούτε στην αρχή ούτε στο τέλος του σχολίου γιατί το σχόλιο από μόνο του θα αποσυνδέσει τη γραμμή που περιέχεται από την επόμενη.

Απαγορεύεται η χρήση του backslash σε οποιοδήποτε άλλο σημείο μιας γραμμής εκτός και αν εμφανίζεται μέσα σε ένα string.

Παράδειγμα λογικής γραμμής με 4 φυσικές γραμμές:

```
if (year < 1900 && month <= 12 \
   && day <= 31 && hour < 24 \
   && minute < 30 && second < 60)
        return 1;
```

2.1.3 Κενές γραμμές

Μια λογική γραμμή που περιέχει μόνο spaces, tabs και πιθανώς ένα σχόλιο αλλά όχι κάποια δήλωση της γλώσσας, αγνοείται, δηλαδή ο λεκτικός αναλυτής δεν επιστρέφει τίποτα στον συντακτικό αναλυτή.

2.2 Δηλώσεις μεταβλητών

Οι δηλώσεις μεταβλητών γίνονται με την αναγραφή του τύπου δεδομένων (double, int, long, short, float) ακολουθούμενου από ένα ή περισσότερα ονόματα μεταβλητών.

Παραδείγματα:

```
int a;
double var;
float c;
```

2.3 Πίνακες (arrays)

Ένας πίνακας περιέχει στοιχεία που διαχωρίζονται με κόμματα και περικλείονται μέσα σε αγκύλες [ ]. Τα στοιχεία μέσα σε έναν πίνακα πρέπει αυστηρά να έχουν τον ίδιο τύπο δεδομένων. Αναφορά σε συγκεκριμένο στοιχείο ενός πίνακα γίνεται με τη χρήση αριθμητικού δείκτη μέσα σε αγκύλες.

Παραδείγματα πινάκων:

```
pin1 = [1, 2, 3, 4, 5 ];
pin2 = ["a", "b", "c", "d"];
print(pin1[0]);   // Θα εκτυπώσει το πρώτο στοιχείο ("a") του pin1
```

2.4 Ενσωματωμένες (Built–in) απλές συναρτήσεις

Στην Uni-C ορίζονται οι ενσωματωμένες συναρτήσεις: scan, len, cmp και print.

2.4.1 Η συνάρτηση scan

Η συνάρτηση `scan` διαβάζει μια τιμή από το πληκτρολόγιο και την αποδίδει σε μία μεταβλητή

Η σύνταξη της περιλαμβάνει το αναγνωριστικό μέσα σε παρενθέσεις.

Παραδείγματα:

```
scan(x);
scan(MyVariable);
```

2.4.2 Η συνάρτηση len

Η συνάρτηση `len` επιστρέφει το μήκος ενός πίνακα ή μιας συμβολοσειράς η οποία πρέπει να αναφέρεται μέσα σε παρενθέσεις (). Η σύνταξη μπορεί να περιέχει αναγνωριστικά.

Παραδείγματα:

```
len([10, 20, 30, 40, 50]);
len("This is a string");
len(StringVariable);
```

2.4.3 Η συνάρτηση cmp

Η συνάρτηση `cmp` συγκρίνει δυο συμβολοσειρές.

Η σύνταξη μπορεί να περιέχει αναγνωριστικά.

Παραδείγματα:

```
cmp("test", "best");
cmp(str1, str2);
```

2.4.4 Η συνάρτηση print

Στην Uni-C η συνάρτηση print χρησιμοποιείται για να τυπώσει κάποιο μήνυμα στην οθόνη. Όταν πρέπει να εκτυπωθούν παραπάνω από ένα στοιχεία με την ίδια εντολή, θα πρέπει να διαχωρίζονται με τον ειδικό χαρακτήρα κόμμα ','.

Παραδείγματα:

```
print("Hello World");
print(x, "=", 100);
```

Υπάρχει επίσης η δυνατότητα της απ' ευθείας εκτύπωσης του αποτελέσματος κλήσης μιας άλλης συνάρτησης.

Παραδείγματα:

```
print(cmp(str1, str2));
print(len("This is a string"));
```

2.5 Δήλωση συναρτήσεων χρήστη

Εδώ παρουσιάζονται οι απλοί κανόνες για να ορίσουμε μια συνάρτηση χρήστη στην Uni-C.

Κάθε συνάρτηση είναι μια δομική μονάδα και αποτελείται από την επικεφαλίδα και το σώμα της. Στην επικεφαλίδα αναφέρεται η δεσμευμένη λέξη func, το όνομα της συνάρτησης και οι παράμετροί της μέσα σε παρενθέσεις. Κάθε παράμετρος περιλαμβάνει τύπο δεδομένων και όνομα ενώ πολλαπλές παράμετροι διαχωρίζονται με κόμμα. Το σώμα της συνάρτησης περικλείεται μέσα σε άγκιστρα. Για την κλήση της συνάρτησης, αρκεί η χρήση του όνόματός της μαζί με τις παραμέτρους μέσα σε παρένθεση.

Παράδειγμα:

```
func myfunc(int varA, int varB)
{
        print("a = ", varA);
        print("b = ", varB);
}
```

```
// Κλήση της myfunc
myfunc(10, 20);
```

2.6 Δηλώσεις απλών εκφράσεων

Μια απλή δήλωση περιλαμβάνεται σε μια ενιαία λογική γραμμή. Αρκετές απλές δηλώσεις μπορούν να συμβούν σε μια ενιαία γραμμή διαχωριζόμενες με ερωτηματικά ';'. Απλές εκφράσεις θεωρούνται οι αριθμητικές εκφράσεις και οι αναθέσεις.

2.6.1 Αριθμητικές εκφράσεις

Η σύνταξη των αριθμητικών εκφράσεων στην Uni-C περιλαμβάνει δύο αναγνωριστικά ή αριθμούς με κάποιον τελεστή (+ - * /) ανάμεσά τους καθώς και προαιρετικά τα πρόσημα + -.

Παραδείγματα:

```
a1 * a2
a3 + a4
-5 + 10
15 + a5 - 9
```

2.6.2 Αναθέσεις τιμών σε αναγνωριστικά

Η ανάθεση τιμών σε αναγνωριστικά γίνεται μέσω του ειδικού χαρακτήρα '=' για μια μεμονωμένη ή για ομαδικές αναθέσεις. Η ομαδική ανάθεση γίνεται διαχωρίζοντας τα αναγνωριστικά με τον χαρακτήρα ',' ενώ ο ίδιος χαρακτήρας χρησιμοποιείται και για το διαχωρισμό των τιμών ανάθεσης.

Παραδείγματα δηλώσεων είναι τα παρακάτω:

```
x1=0;
x1, x2 = 0, 1;
x1, list1 = 0, [A, B, C];
x1, list1, string  = 0, [A, B, C], "HELLO";
```

2.6.3 Συγκρίσεις

Η σύγκριση τιμών σε αναγνωριστικά γίνεται μέσω των τελεστών >, <, <=, >=, == και !=.

Παραδείγματα συκγρίσεων είναι τα παρακάτω:

```
x1 > x2
a <= b
myvar == 52
```

2.6.4 Συνένωση πινάκων

Δυο πίνακες μπορούν να συνενωθούν με τη βοήθεια του τελεστή + .

Παραδείγματα:

```
[1, 2, 3] + [4, 5, 6]          -->   [1, 2, 3, 4, 5, 6]
[1, 3] + [5, 7, 9, 11]         -->   [1, 3, 5, 7, 9, 11]
```

2.7 Σύνθετες δηλώσεις

Οι σύνθετες δηλώσεις περιέχουν άλλες δηλώσεις. Σε γενικές γραμμές, οι σύνθετες δηλώσεις εκτείνονται σε πολλές γραμμές, αν και σε απλές υλοποιήσεις μια ολόκληρη δήλωση μπορεί να περιέχεται σε μία γραμμή. Σύνθετες γνωστές δηλώσεις είναι οι if, while και for που υλοποιούν παραδοσιακές ροές ελέγχου.

2.7.1 Η δήλωση if

Η δήλωση if χρησιμοποιείται για την εκτέλεση εντολών κώδικα εάν κάποια συνθήκη (που δίνεται εντός παρένθεσης μετά τη δεσμευμένη λέξη if) είναι αληθής.

Παραδείγματα:

```
if (var  == 100) print("Value of expression is 100");
if (x < y < z) { print(x); print(y); print(z); }
```

2.7.2. Η δήλωση while

Η δήλωση while χρησιμοποιείται για επαναλαμβανόμενη εκτέλεση εάν κάποια συνθήκη (που δίνεται εντός της παρένθεσης μετά τη δεσμευμένη λέξη while) είναι αληθής.

Παράδειγμα:

```
// Print i as long as i is less than 10
int i;
i = 1;
while (i < 10)
{
    print("i is: ", i);
    i += 1;
}
```

2.7.3. Η δήλωση for

Η δήλωση for χρησιμοποιείται για επαναλαμβανόμενη εκτέλεση. Στην παρένθεση μετά τη δεσμευμένη λέξη for υπάρχουν τρία ορίσματα χωρισμένα με semi-colon: Το πρώτο είναι μια ανάθεση τιμής σε μεταβλητή που γίνεται μονάχα την πρώτη φορά που η εκτέλεση του προγράμματος φτάνει στην for. Το δεύτερο όρισμα είναι μια συνθήκη που ελέγχεται σε κάθε επανάληψη και που όσο ισχύει η εκτέλεση της for εξακολουθεί. Τέλος, το τρίτο όρισμα είναι μια ανάθεση τιμής που εκτελείται μετά το πέρας κάθε επανάληψης της for.

Παράδειγμα:

```
// Print i, starting from 0 as long as i is less than 10
int i;
for (i = 0; i < 10; i++)
{
    print("i = ", i);
}
```

Παραδείγματα κώδικα της Uni-C

Παρακάτω δίνονται δύο απλά παραδείγματα συγγραφής προγραμμάτων πηγαίου κώδικα της Uni-C:

```
func main()
{
      int a;
      int b;
      scan(a);
      scan(b);
      myAdd(a,b);
}
```

```
func myAdd(int num1, int num2)
{
      int sum;
      sum = a + b;
      print("Sum is: ", sum)
}
```

Τελευταία ενημέρωση: 2024.3.9