Πανεπιστημιο Ιωαννινων
Τμήμα Πληροφορικής

Μεταφραστές
Ακαδημαϊκό Έτος 1999-2000

Εγχειρίδιο της Γλώσσας C−−

Η γλώσσα C−− είναι μια διαδικαστική γλώσσα που η σύνταξη της και κυρίως η δομή και οι κανόνες εμβέλειας της μοιάζουν με αυτούς της C. Η C−− σχεδιάστηκε για να χρησιμοποιηθεί στο εργαστήριο του μαθήματος των Μεταφραστών. Υποστηρίζει μόνο ένα υποσύνολο της C που κρίθηκε επαρκές για να κατανοηθούν οι βασικές αρχές και τεχνικές της διαδικασίας της μετάφρασης και ταυτόχρονα εφικτό να υλοποιηθεί στα χρονικά πλαίσια του μαθήματος.

Τα παρόν εγχειρίδιο παρέχει μια συνοπτική παρουσίαση της γλώσσας.

Προσοχή! Η γλώσσα έχει διαφορές από τη γλώσσα που δόθηκε την προηγούμενη ακαδημαϊκή χρονιά.

Συμβολισμός

Η γραμματική της γλώσσας δίνεται σε μορφή EBNF, όπου κάθε κανόνας είναι του τύπου:

<non_terminal> ::= definition

Τα σύμβολα που χρησιμοποιούνται έχουν την εξής σημασία:

<foo>	foo είναι ένα μη τερματικό σύμβολο
foo	foo είναι τερματικό σύμβολο που εμφανίζεται ακριβώς έτσι στο πρόγραμμα
foo	foo είναι τερματικό σύμβολο
r?	προαιρετική εμφάνιση του r (0 ή 1 φορά)
r*	0 ή περισσότερες επαναλήψεις του r
r+	1 ή περισσότερες επαναλήψεις του r
$a\vert b\vert$ ...	εμφάνιση ενός από τα a, b, ...

1 Δομή του Προγράμματος

Ένα πρόγραμμα σε C−− έχει την παρακάτω μορφή:

<program> ::= (< declaration>)*
<declaration> ::= <var_decl> | <function_decl> | <forw_function_decl>

Η περιγραφή των δηλώσεων των μεταβλητών (<var_decl>) δίνεται στην Παράγραφο 2.4 και των συναρτήσεων (<function_decl> και <forw_function_decl>) στην Παράγραφο 4. Μια από τις συναρτήσεις που ορίζονται στο <function_decl> πρέπει να έχει όνομα *main* που είναι και δεσμευμένη λέξη. Η συνάρτηση με όνομα *main* είναι η πρώτη που εκτελείται.

Οι κανόνες ορατότητας (visibility) για τις μεταβλητές και τις συναρτήσεις βασίζονται στη λεκτική εμβέλεια (lexical scope) όπως ακριβώς στην Pascal.

Σχόλια

Ο,τιδήποτε περικλείεται ανάμεσα σε (* και *) θεωρείται σχόλιο και αγνοείται. Δεν υποστηρίζονται φωλιασμένα σχόλια. Επίσης, δε μπορεί να εμφανίζονται σχόλια μέσα σε σταθερές συμβολοσειρές (*string_constant*). Τέλος, υπάρχει ένα ειδικό σύμβολο για σχόλια μιας γραμμής. Συγκεκριμένα, ό,τι ακολουθεί // και μέχρι το τέλος της γραμής θεωρείται σχόλιο και αγνοείται.

2 Τύποι, Εντολές Δηλώσεων Μεταβλητών και Τελεστές

2.1 Ονόματα Μεταβλητών

Επιτρεπτά γράμματα είναι τα a, …, z, A, …, Z και επιτρεπτά ψηφία τα 0, …, 9. Ένα όνομα (identifier) είναι μια ακολουθία από γράμματα, ψηφία και κάτω παύλες (_). Ο πρώτος και ο τελευταίος χαρκτήρας του ονόματος πρέπει να είναι γράμμα. Δεν επιτρέπεται να υπάρχουν δύο συνεχόμενες κάτω παύλες στο όνομα. Ένα όνομα μπορεί να έχει μέχρι 16 χαρακτήρες.

Δε γίνεται διάκριση ανάμεσα στα κεφαλαία και στα μικρά γράμματα.

Οι λέξεις κλειδιά είναι δεσμευμένες.

2.2 Τύποι Δεδομένων

Οι βασικοί τύποι δεδομένων που υποστηρίζει η γλώσσα είναι: ακέραιοι (integers) και χαρακτήρες (characters).

<simple_type> ::= char | int

Εκτός από τους βασικούς τύπους η γλώσσα υποστηρίζει και πίνακες. Τα στοιχεία ενός πίνακα έχουν έναν από τους βασικούς τύπους.

<array_id> ::= *identifier* ([<array_range>])+
<array_range> ::= *int_constant* .. *int_constant*

όπου *int_constant* είναι μια θετική ακέραια σταθερά. Το πάνω όριο πρέπει να είναι μεγαλύτερο ή ίσο του κάτω ορίου.

2.3 Σταθερές

Μια σταθερά είναι τύπου ακεραίου ή συμβολοσειρά.

<constant> ::= *int_constant* | *string_constant*

Οι αριθμητικές σταθερές αποτελούνται από ένα προαιρετικό πρόσημο ακολουθούμενο από ένα ή περισσότερα ψηφία. Αν μια αριθμητική σταθερά έχει παραπάνω από ένα ψηφίο, το πρώτο ψηφίο δε μπορεί να είναι 0.

Οι σταθερές συμβολοσειρές είναι μια ακολουθία από χαρακτήρες ανάμεσα σε μονά εισαγωγικά. Για να συμπεριλάβουμε ένα εισαγωγικό σε μια συμβολοσειρά πρέπει να προηγείται το σύμβολο escape (\). Για να συμπεριληφθεί το escape πρέπει να γραφτεί δυο φορές. Μια συμβολοσειρά δεν περνά τα όρια μιας γραμμής. Κάθε συμβολοσειρά πρέπει να έχει τουλάχιστον έναν χαρακτήρα.

Μια σταθερά τύπου χαρακτήρα είναι μια συμβολοσειρά με ένα μόνο χαρακτήρα.

2.4 Δηλώσεις Μεταβλητών

Καθε μεταβλητή πρέπει να δηλώνεται πριν χρησιμοποιηθεί. Η σύνταξη των δηλώσεων είναι:

<var_decl> ::= <id> (, <id>)* : <simple_type> ;
<id> ::= *identifier* | <array_id>

2.5 Τελεστές

Οι αριθμητικοί τελεστές έχουν ως τελεσταίους ακεραίους και δίνουν ως αποτέλεσμα ακέραιο. Είναι οι εξής:

+ (πρόσθεση)

- (αφαίρεση)

∗ (πολλαπλασιασμός)

/ (ακέραιο πηλίκο διαίρεσης)

mod (ακέραιο υπόλοιπο διαίρεσης)

- (αρνητικό πρόσημο)

Δεν υπάρχει ειδικός τύπος λογικός (boolean), γενικά θεωρούμε ότι η τιμή true αντιστοιχεί σε ένα θετικό ακέραιο και η τιμή false αντιστοιχεί σε έναν ακέραιο μικρότερο μικρότερο ή ίσο με το 0. Οι λογικοί τελεστές έχουν ως τελεσταίους λογικούς και δίνουν ως αποτέλεσμα true ή false. Είναι οι εξής:

! (άρνηση)

&& (λογικό AND)

|| (λογικό OR)

Οι τελεστές σύγκρισης εφαρμόζονται πάνω σε τελεσταίους ίδιου τύπου. Ο τύπος των τελεσταίων μπορεί να είναι άκεραιος ή συμβολοσειρά. Το αποτέλεσμα είναι true ή false. Οι σχεσιακοί τελεστές είναι οι εξής:

<rel_op> ::= < | <= | > | >= | != | =

Για τις συμβολοσειρές ισχύει η λεξικογραφική διάταξη, όπου η διάταξη μεταξύ χαρακτήρων γίνεται με βάση την συνάρτηση ord της Pascal.

Προτεραιότητα και Προσεταιριστικότητα

Οι τελεστές είναι προσεταιριστικοί από τα αριστερά προς τα δεξιά.

Η προτεραιότητα είναι ξεκινώντας από τη μεγαλύτερη:

!, − (πρόσημο)

&&, ∗, /

||, +, − (αφαίρεση)

σχεσιακοί τελεστές

3 Εντολές

Οι εντολές έχουν τν παρακάτω μορφή:

<stmt> ::= <simple_stmt> | <compound_stmt>
<compound_stmt> ::= { <stmt_list> }
<stmt_list> ::= (<simple_stmt> (; <simple_stmt>)*)?

Μια απλή εντολή μπορεί να είναι μια εντολή ανάθεσης, μια εντολή ελέγχου ροής for, μια εντολή ελέγχου ροής repeat, ή μια κλήση μιας συνάρτησης τύπου void.

<simple_stmt> ::= <assgnmnt_stmt> |
<if_stmt> |
<for_stmt> |
<repeat_stmt> |
<function_call> |

3.1 Εντολή Ανάθεσης

Μια εντολή ανάθεσης έχει την μορφή:

<assngnmnt_stmt> ::= <l_expr> = <r_expr>

Οι τύποι των <l_expr> και <r_expr> πρέπει να συμφωνούν.

Με τον όρο <l_expr> ορίζουμε ο,τιδήποτε μπορεί να βρεθεί στα αριστερά μιας εντολής ανάθεσης, δηλαδή ονόματα μεταβλητών απλού τύπου ή στοιχεία πίνακα:

<l_expr> ::= *identifier* |
identifier ([<r_expr>])+

όπου ο τύπος του <r_expr> πρέπει να είναι ακέραιος και οι διαστάσεις του πίνακα να συμφωνούν με τον ορισμό του.

Δεν επιτρέπονται αναθέσεις σε ολόκληρο πίνακα εκτός από την περίπτωση της ανάθεσης μιας σταθερής συμβολοσειράς σε ένα μονοδιάστατο πίνακα. Σε αυτήν την περίπτωση, τα στοιχεία του πίνακα πρέπει να είναι τύπου χαρακτήρα και οι διαστάσεις του να επαρκούν.

Ανάθεση στο όνομα μιας συνάρτησης επιτρέπεται μόνο στο body της συνάρτησης και μόνο αν η συνάρτηση δεν είναι τύπου void.

Μια <r_expr> μπορεί να είναι μια σταθερά, μια <l_expr>, η τιμή μιας συνάρτησης που δεν είναι void, ή αποτέλεσμα πράξεων.

<r_expr> ::= <l_expr> |
<constant> |
<function_call> |
<r_expr> (<binary_op> <r_expr>)* |
<unary_op> <r_expr> |
(<r_expr>)

όπου

<binary_op> ::= && | || | * | - | + | / | <rel_op>
<unary_op> ::= ! | -

3.2 Εντολές Ελέγχου Ροής

<if_stmt> ::= if (<r_expr>)
then <stmt>
(else <stmt>)?

Ο τύπος του <r_expr> πρέπει να είναι ακέραιος.

<repeat_stmt> ::= repeat
<stmt>
until (<r_expr>)

Ο τύπος του <r_expr> πρέπει να είναι ακέραιος.

<for_stmt> ::= for (<l_expr> = <r_expr> to <r_expr>)
<stmt>

Ο τύπος του <r_expr> και του <l_expr> πρέπει να είναι ακέραιος.

4 Συναρτήσεις

Η γλώσσα C−− είναι μια δομημένη γλώσσα (block-structured). Δεν υπάρχουν φωλιασμένες (nested) συναρτήσεις. Οι κανόνες εμβέλειας είναι ίδιοι με αυτούς στη C.

Δε γίνεται διάκριση ανάμεσα σε συναρτήσεις και διαδικασίες. Μια διαδικασία είναι απλώς μια συνάρτηση που δεν επιστρέφει τιμή. Αυτό δηλώνεται ορίζοντας τον τύπο της τιμής που επιτρέφει η συνάρτηση ως “void”.

Είναι επίσης δυνατόν να δίνεται ο ορισμός μιας συνάρτησης (< <forw_function_decl>) χωρίς το κυρίως μέρος της (body). Σε αυτήν την περίπτωση, το κυρίως μέρος της συνάρτησης θα πρέπει να δίνεται αργότερα. Αυτό αντιστοιχεί στη δήλωση “forward” που υποστηρίζει η Pascal.

Ειδικά για τη συνάρτηση “main” δεν επιτρέπεται “forward” δήλωση. Ο τύπος της είναι “void”.

<function> ::= *identifier* (<formal_par_list>) : <return_type>
{
<var_declaration>
<stmt>
}
<return_type> ::= <simple_type> | void

Οι δηλώσεις των forward function έχουν την εξής σύνταξη:

<forw_function_decl> ::= forward <forward_function> (, <forward_function>)* : <return_type>
;
<forward_function> ::= *identifier* (<formal_par_list>)

Το πέρασμα παραμέτρων γίνεται είτε με τιμή (by value) είτε με αναφορά (by name). Το πέρασμα με αναφορά δηλώνεται με χρήση της λέξης var.

<formal_par_list> ::= (<value_par> | <var_par>)*
<value_par> ::= <var_decl>
<var_par> ::= var <var_decl>

Δεν επιτρέπονται συναρτήσεις ως παράμετροι.

Η επιστροφή από τη συνάρτηση πραγματοποιείται στο τέλος του κώδικά της. Κατά την κλήση μιας συνάρτησης, οι πραγματικοί παράμετροι πρέπει να συμφωνούν με τον αριθμό και τον τύπο των τυπικών παραμέτρων.

<function_call> ::= *identifier*(<actual_par_list>?)
<actual_par_list> ::= <r_expr> (, <r_expr>)*

Στην περίπτωση περάσματος με αναφορά, η πραγματική παράμετρος πρέπει να είναι μεταβλητή.