Πίνακας Συμβόλων

Γεώργιος Μανής
Πανεπιστήμιο Ιωαννίνων
Πολυτεχνική Σχολή
Τμήμα Μηχανικών Η/Υ και Πληροφορικής

Απρίλιος 2022

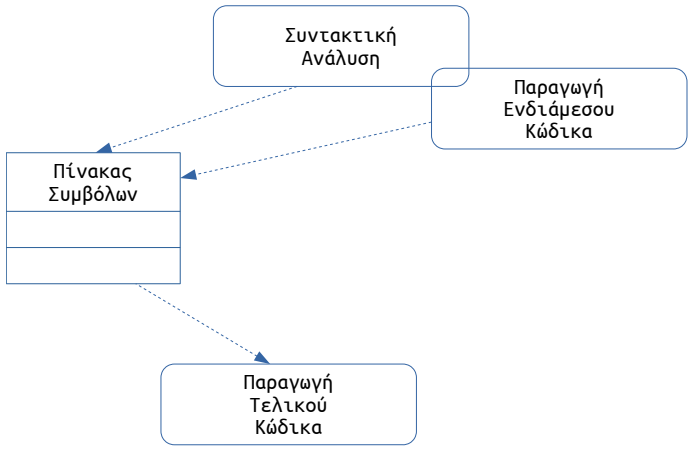
Ο πίνακας συμβόλων είναι δυναμική δομή στην οποία αποθηκεύεται πληροφορία σχετιζόμενη με τα συμβολικά ονόματα που χρησιμοποιούνται στο υπό μεταγλώττιση πρόγραμμα. Η δομή αυτή παρακολουθεί τη μεταγλώττιση και μεταβάλλεται δυναμικά, με την προσθήκη ή αφαίρεση πληροφορίας σε και από αυτήν, ώστε σε κάθε σημείο της διαδικασίας της μεταγλώττισης να περιέχει ακριβώς την πληροφορία που εκείνη τη στιγμή πρέπει να έχει. Λέγοντας *πρέπει να έχει*, εννοούμε τις εγγραφές εκείνες και μόνο αυτές, που σύμφωνα με τους κανόνες εμβέλειας της γλώσσας, το υπό μεταγλώττιση πρόγραμμα έχει δικαίωμα να έχει πρόσβαση τη συγκεκριμένη στιγμή της μεταγλώττισης.

Σε ένα πίνακα συμβόλων διατηρούμε πληροφορία για τα συμβολικά ονόματα που εμφανίζονται στο πρόγραμμα. Έτσι, σε έναν πίνακα συμβόλων αποθηκεύεται πληροφορία που σχετίζεται με τις μεταβλητές του προγράμματος, τις διαδικασίες και τις συναρτήσεις, τις παραμέτρους και με τα ονόματα των σταθερών. Για κάθε ένα από αυτά υπάρχει διαφορετική εγγραφή στον πίνακα και στην εγγραφή αυτή αποθηκεύεται διαφορετική πληροφορία, ανάλογα με το είδος του συμβολικού ονόματος. Η πληροφορία αυτή είναι χρήσιμη για έλεγχο σφαλμάτων, αλλά είναι διαθέσιμη να ανακτηθεί κατά τη φάση της παραγωγής του τελικού κώδικα.

Η μορφή του πίνακα συμβόλων δεν είναι αυτή την οποία φανταζόμαστε κρίνοντας από τη λέξη *πίνακας*. Συνήθως σαν πίνακα φανταζόμαστε κάποια τετραγωνική απεικόνιση πληροφορίας, κάτι δηλαδή καλά περιγεγραμμένο από μία δισδιάστατη, ομοιόμορφη αναπαράσταση. Ανάλογα με τη γλώσσα που υλοποιούμε, ο πίνακας συμβόλων μπορεί να είναι από μία απλή δομή, ως μία οργάνωση δεδομένων σε επίπεδα, με (πιθανά φωλιασμένες) λίστες και λεξικά ή αντικείμενα σε κάθε επίπεδο. Η *C-imple* έχει αυξημένες απαιτήσεις σε αυτό το σημείο και ο πίνακας συμβόλων παρουσιάζει ενδιαφέρουσα πολυπλοκότητα.

Πέρα από την αρωγή στην παραγωγή του τελικού κώδικα, με τον πίνακα συμβόλων είναι συνυφασμένη η σημασιολογική ανάλυση. Ο πίνακας συμβόλων παρέχει την πληροφορία που απαιτείται για τη σημασιολογική ανάλυση, ενώ μέρος της σημασιολογικής ανάλυσης υλοποιείται μέσα στον πίνακα συμβόλων.

Σε σχέση με τις φάσεις ανάπτυξης ενός μεταγλωττιστή, τις οποίες περιγράψαμε στο εισαγωγικό κεφάλαιο, ο πίνακας συμβόλων αντλεί πληροφορία από τις φάσεις της συντακτικής ανάλυσης και της παραγωγής ενδιάμεσου κώδικα και διαθέτει την πληροφορία που έχει συλλέξει για την παραγωγή τελικού κώδικα. Στο σχήμα 1.1 εικονίζεται η αλληλεπίδραση του πίνακα συμβόλων με τις φάσεις της συντακτικής ανάλυσης και της παραγωγής ενδιάμεσου και τελικού κώδικα.



Σχήμα 1.1: Ο πίνακας συμβόλων

1.1 Οι εγγραφές στον πίνακα συμβόλων

Ο πίνακας συμβόλων διατηρεί διαφορετική πληροφορία για κάθε είδος συμβολικού ονόματος. Σκοπός του είναι να συγκεντρώσει όλη την πληροφορία που θα απαιτηθεί να αντληθεί για το κάθε συμβολικό όνομα μέχρι το τέλος της μεταγλώττισης. Έτσι, για παράδειγμα, στην εγγραφή που αντιστοιχίζεται σε μία μεταβλητή θα αποθηκεύσει, πέρα από το όνομά της, τον τύπο της. Στην εγγραφή μιας διαδικασίας ο τύπος δεν έχει καμία αξία, έχει όμως αντίθετα αξία να αποθηκεύσουμε, πέρα από οτιδήποτε άλλο, τη θέση της μνήμης στην οποία είναι αποθηκευμένη η διαδικασία. Στην εγγραφή μιας παραμέτρου χρειαζόμαστε να αποθηκεύσουμε, πάλι μεταξύ άλλων, αν πρόκειται για παράμετρο που περνάει με τιμή ή με αναφορά. Ας δούμε όμως καλύτερα, για κάθε περίπτωση χωριστά, ποια είναι η πληροφορία που αποθηκεύουμε για κάθε τύπο συμβολικού ονόματος.

Ας ξεκινήσουμε με τον πιο συχνά χρησιμοποιούμενο τύπο, αυτόν της *μεταβλητής*. Κάθε μεταβλητή έχει το όνομα που τη χαρακτηρίζει καθώς και τον τύπο της. Δεν απαιτούν όλες οι γλώσσες προγραμματισμού από τον προγραμματιστή να ορίσει τον τύπο μιας μεταβλητής. Ανάλογα με τη φιλοσοφία της γλώσσας, ο ορισμός του τύπου της μεταβλητής μπορεί να γίνει αυτόματα, μπορεί να οριστεί και να τροποποιηθεί αργότερα τόσο από τον προγραμματιστή ή τον μεταγλωττιστή. Ο πίνακας συμβόλων που υλοποιούμε οφείλει να παρακολουθήσει τις ανάγκες της γλώσσας και να προσαρμοστεί κατάλληλα. Στην *C-imple* οι τύποι των μεταβλητών ορίζονται στην αρχή του προγράμματος ή ενός υποπρογράμματος και διατηρούνται χωρίς μεταβολή όσο διαρκεί η ζωή της μεταβλητής.

Μία ακόμα πληροφορία πολύ σημαντική για κάθε μεταβλητή είναι η θέση στην οποία αυτή θα βρίσκεται στην μνήμη. Η πληροφορία αυτή είναι απαραίτητη και θα αναζητηθεί στον πίνακα συμβόλων κατά την παραγωγή τελικού κώδικα. Παρότι φαίνεται νωρίς να ασχοληθεί ο μεταγλωττιστής με κάτι τέτοιο, στην πραγματικότητα αυτή είναι η κατάλληλη στιγμή για να σχεδιαστεί ο τρόπος με τον οποίο θα γίνει η τοποθέτηση των μεταβλητών στη μνήμη.

Η φυσική θέση της μεταβλητής στη μνήμη δεν μπορεί να προσδιοριστεί με ακρίβεια, αλλά ούτε και θέλουμε να κάνουμε κάτι τέτοιο. Αρκεί να σκεφτούμε ότι κάθε φορά που καλείται μια συνάρτηση ή μία διαδικασία η ίδια τοπική μεταβλητή τοποθετείται σε διαφορετική θέση της μνήμης. Στην περίπτωση, μάλιστα, αναδρομικών ή φωλιασμένων κλήσεων υπάρχουν περισσότερες από ένα στιγμιότυπα της μεταβλητής ταυτόχρονα στη μνήμη.

Τότε, τελικά, τι είναι αυτό που αποθηκεύεται στον πίνακα συμβόλων; Στον πίνακα συμβόλων αποθηκεύεται η θέση της μεταβλητής μέσα στο εγγράφημα δραστηριοποίησης της συνάρτησης ή της διαδικασίας, η απόσταση δηλαδή από την αρχή του. Το *εγγράφημα δραστηριοποίησης* είναι ο χώρος που δίνεται σε μία συνάρτηση ή διαδικασία για να τοποθετήσει τα δεδομένα της στη μνήμη. Ο τρόπος με τον οποίο γίνεται ο υπολογισμός της απόστασης της μεταβλητής από την αρχή του θα συζητηθεί λίγο αργότερα στο κεφάλαιο αυτό.

Έτσι, ορίζουμε την κλάση `Variable` με την οποία μπορούμε να δημιουργούμε εισαγωγές στον πίνακα συμβόλων που αναπαριστούν εγγραφές. Τα πεδία της κλάσης θα μπορούσαν να είναι τα ακόλουθα:

- `name`: το όνομα της μεταβλητής
- `datatype`: ο τύπος δεδομένων της μεταβλητής
- `offset`: η απόστασή της μεταβλητής από την αρχή του εγγραφήματος δραστηριοποίησης

Ένας άλλος τύπος εγγραφών σε πίνακα συμβόλων είναι η *παράμετρος*. Πρόκειται για τις παραμέτρους που περνούμε στις διαδικασίες και τις συναρτήσεις. Η παράμετρος μοιάζει πολύ με τη μεταβλητή. Ο τρόπος με τον οποίο τελικά θα διαχειριστούμε τις μεταβλητές και τις παραμέτρους είναι πολύ διαφορετικός, αλλά όσον αφορά τον πίνακα συμβόλων η πληροφορία που διατηρούμε είναι παρόμοια. Έτσι, κάθε παράμετρος έχει το όνομά της, τον τύπο της και τον τρόπο με τον οποίο περνάει. Τα δύο πρώτα είναι τα ίδια με αυτά της μεταβλητής και δεν χρειάζεται να τα αναλύσουμε περισσότερο.

Οι τρόποι περάσματος διαφέρουν από γλώσσα σε γλώσσα, αλλά οι περισσότερο συνηθισμένοι τύποι περάσματος είναι το *πέρασμα με τιμή* και το *πέρασμα με αναφορά*, οι τύποι δηλαδή που υποστηρίζονται από την *C-imple*. Θα ονομάσουμε λοιπόν ένα πεδίο `mode` το οποίο θα μπορεί να παίρνει τις τιμές `in`, `ref` και `ret`. Το πρώτο αντιστοιχεί στο πέρασμα με τιμή, δεύτερο στο πέρασμα με αναφορά, ενώ το τρίτο είναι για επιστροφή τιμής συνάρτησης. Θυμάστε ότι είχαμε διαχειριστεί την επιστροφή τιμής συνάρτησης στον ενδιάμεσο κώδικα σαν μία παράμετρο. Δεν έχουμε λόγο να το διαχειριστούμε διαφορετικά εδώ, πρόκειται για έναν άλλο τρόπο να διαχειριστούμε επικοινωνία δεδομένων ανάμεσα στην κληθείσα και την καλούσα.

Το τελευταίο πεδίο που χρειαζόμαστε είναι το `offset`. Όπως και στις μεταβλητές, πρέπει να γνωρίζουμε την απόσταση της παραμέτρου από την αρχή του εγγραφήματος δραστηριοποίησης, ώστε να μπορούμε να εντοπίζουμε την παράμετρο στη μνήμη. Το τι πληροφορία κρατάμε στη μνήμη για μια μεταβλητή ή κάθε είδος παραμέτρου δεν μας ενδιαφέρει ακόμα, αν και μπορούμε να τη φανταστούμε. Αυτό που μας ενδιαφέρει τώρα είναι ότι μέσα από το `offset` μπορούμε να εντοπίσουμε την πληροφορία αυτή.

Έτσι, συνοψίζοντας, θα ορίσουμε την κλάση `Parameter`, η οποία περιγράφει μία παράμετρο και θα μπορούσε να έχει τα ακόλουθα πεδία:

- `name`: το όνομα της παραμέτρου
- `datatype`: ο τύπος δεδομένων της παραμέτρου
- `mode`: ο τρόπος περάσματος της παραμέτρου
- `offset`: η απόστασή της παραμέτρου από την αρχή του εγγραφήματος δραστηριοποίησης

Πολύ εύκολα μπορεί να παρατηρήσει κανείς ότι εδώ παρουσιάζονται ευκαιρίες οργάνωσης των κλάσεων σε ιεραρχία που εκφράζει κληρονομικότητα. Ας αναβάλλουμε όμως αυτή τη συζήτηση έως ότου αναλύσουμε όλους τους πιθανούς τύπους εγγραφών, οπότε και θα δούμε τις δυνατότητες ιεράρχησης των κλάσεων συνολικά.

Δύο ακόμα τύποι εγγραφής στον πίνακα συμβόλων είναι η *συνάρτηση* και η *διαδικασία*. Χρησιμοποιούνται για να σημειώσουν την ύπαρξη ενός υποπρογράμματος στον κώδικα. Όπως και στις προηγούμενες εγγραφές, αποθηκεύουμε στον πίνακα συμβόλων όλη την πληροφορία που σχετίζεται με το υποπρόγραμμα αυτό.

Έτσι, στο πρώτο πεδίο θα αποθηκεύσουμε το όνομα του υποπρογράμματος. Το δεύτερο πεδίο είναι μία λίστα με τις τυπικές παραμέτρους του υποπρογράμματος. Η σειρά με την οποία συναντούμε τις τυπικές παραμέτρους στο υποπρόγραμμα είναι και η σειρά με την οποία εμφανίζονται αυτές στη λίστα. Χρειαζόμαστε, λοιπόν, μία ακόμα κλάση, την κλάση των τυπικών παραμέτρων. Ας δώσουμε στην κλάση αυτή το προφανές όνομα: `FormalParameter` και ας τη μελετήσουμε αμέσως μετά.

Ένα ακόμα πεδίο που κάθε υποπρόγραμμα χρειάζεται είναι το `startingQuad`. Στο πεδίο αυτό αποθηκεύουμε την ετικέτα της πρώτης εκτελέσιμης τετράδας του ενδιάμεσου κώδικα που αντιστοιχεί στη συνάρτηση ή τη διαδικασία αυτή. Με άλλα λόγια, εκεί σημειώνουμε την τετράδα στην οποία πρέπει η καλούσα συνάρτηση να κάνει άλμα προκειμένου να εκκινήσει η εκτέλεση της κληθείσας.

Το τελευταίο πεδίο κοινό πεδίο ανάμεσα σε μία συνάρτηση και μία διαδικασία είναι το `framelength`. Πρόκειται, όπως ίσως υποψιάστηκε κανείς από την ονομασία του, για το μήκος (σε bytes) του εγγραφήματος δραστηριοποίησης της συνάρτησης. Ο τρόπος υπολογισμού του μήκους του εγγραφήματος δραστηριοποίησης θα συζητηθεί σε λίγο, παρακάτω, σε επόμενη ενότητα του κεφαλαίου αυτού.

Αν πρόκειται για συνάρτηση και όχι για διαδικασία, χρειαζόμαστε ακόμα ένα πεδίο: αυτό που περιγράφει τον τύπο δεδομένων που επιστρέφει η συνάρτηση. Βέβαια, στην *C-imple* έχουμε μόνο έναν τύπο δεδομένων, οπότε το πεδίο αυτό θα έχει πάντοτε την τιμή *integer*, αλλά στη γενικότερη περίπτωση το πεδίο αυτό θα μπορεί να πάρει κάθε τύπο δεδομένων που η περιγραφή της γλώσσας ορίζει και επιτρέπει.

Έτσι, η κλάση `Procedure` θα έχει τα εξής πεδία:

- `name`: το όνομα της διαδικασία
- `startingQuad`: η ετικέτα της πρώτης εκτελέσιμης τετράδας της διαδικασίας
- `framelength`: το μήκος του εγγραφήματος δραστηριοποίησης
- `formalParameters`: η λίστα με τις τυπικές παραμέτρους της διαδικασίας

Και η κλάση `Function` ακόμα ένα:

- `name`: το όνομα της συνάρτησης
- `datatype`: ο τύπος δεδομένων της συνάρτησης
- `startingQuad`: η ετικέτα της πρώτης εκτελέσιμης τετράδας της συνάρτησης
- `framelength`: το μήκος του εγγραφήματος δραστηριοποίησης
- `formalParameters`: η λίστα με τις τυπικές παραμέτρους της συνάρτησης

Αφήσαμε ένα υπόλοιπο, την κλάση `FormalParameter`. Όπως είπαμε, πρόκειται για την κλάση, στιγμιότυπα της οποία αποτελούν αντικείμενα που περιγράφουν τη δήλωση παραμέτρων της συνάρτησης ή της διαδικασίας. Αυτά θα τοποθετηθούν σε μία λίστα, πεδίο ενός στιγμιότυπου της κλάσης `Procedure` ή `Function`. Η σειρά που θα έχει στη λίστα αυτή θα υποδηλώνει και τη σειρά την οποία θα έχει αυτή η παράμετρος στη συνάρτηση. Άρα δεν χρειαζόμαστε πεδίο της `FormalParameter` που να κρατά την πληροφορία αυτή.

Έτσι, η κλάση `FormalParameter` θα πρέπει να έχει μόνο δύο πεδία τα:

- `name`: το όνομα της τυπικής παραμέτρου
- `datatype`: ο τύπος δεδομένων της τυπικής παραμέτρου
- `mode`: ο τρόπος περάσματος της τυπικής παραμέτρου

Ο προσεκτικός αναγνώστης μπορεί να παρατηρήσει ότι στην κλάση αυτή δεν χρειάζεται να αποθηκεύσουμε το όνομα της παραμέτρου. Ο λόγος που διατηρούμε τη λίστα με τις τυπικές παραμέτρους είναι για να μπορέσουμε να ελέγξουμε, αργότερα, ότι μία συνάρτηση ή διαδικασία κλήθηκε όπως ακριβώς δηλώθηκε. Ως γνωστόν, το όνομα της τυπικής παραμέτρου δεν είναι υποχρεωτικό να είναι το ίδιο με το όνομα της πραγματικής παραμέτρου. Για λόγους ομοιομορφίας αλλά και πληρότητας της πληροφορίας που τοποθετούμε στον πίνακα συμβόλων, θα επιλέξουμε να κρατήσουμε και το όνομα της τυπικής παραμέτρου σε αυτόν.

Θα ορίσουμε ακόμα μία κλάση, την `TemporaryVariable`. Αυτή η κλάση, δεν διαφέρει στη βασική λειτουργικότητά της από την `Variable`, ίσως στον κατασκευαστή της. Σχεδιαστικά είναι σωστό να την διαφοροποιήσουμε από την `Variable`, και θα το κάνουμε, όπως είπαμε, για πληρότητα. Τα πεδία της είναι τα ακόλουθα:

- `name`: το όνομα της προσωρινής μεταβλητής

- `datatype`: ο τύπος δεδομένων της προσωρινής μεταβλητής
- `offset`: η απόστασή της προσωρινής μεταβλητής από την αρχή του εγγραφήματος δραστηριοποίησης

Τέλος χρειαζόμαστε ακόμα μία κλάση, την `SymbolicConstant` για να κρατήσουμε εκεί τις τιμές των συμβολικών σταθερών. Οι συμβολικές σταθερές πρέπει να είναι διαθέσιμες στη φάση της μετάφρασης. Όταν ολοκληρωθεί η μετάφραση αυτές δεν χρειάζονται πια, αφού στον τελικό κώδικα τα ονόματα των σταθερών έχουν αντικατασταθεί με τις τιμές τους. Συνεπώς δεν υπάρχει κάποιος λόγος να τοποθετηθούν στη στοίβα και να καταλάβουν άσκοπα χώρο εκεί. Για το λόγο αυτό, οι συμβολικές σταθερές δεν χρειάζεται να έχουν πεδίο `offset`. Έτσι, τα πεδία της κλάσης `SymbolicConstant` είναι τρία. Στην *C-imple* δεν χρειαζόμαστε το `datatype`. αφού όλες οι συμβολικές σταθερές έχουν ακέραιες τιμές.

- `name`: το όνομα της συμβολικής σταθεράς
- `datatype`: ο τύπος δεδομένων της συμβολικής σταθεράς
- `value`: η τιμή της συμβολικής σταθεράς

Πριν κλείσουμε την ενότητα αυτή, θα πρέπει να σημειώσουμε ότι τα στιγμιότυπα κάθε κλάσης δημιουργούνται και εισάγονται στον πίνακα συμβόλων κατά την δήλωση τους στο αρχικό πρόγραμμα, ενώ τα πεδία τους συμπληρώνονται όταν η πληροφορία αυτή γίνει γνωστή. Για παράδειγμα, το αντικείμενο της κλάσης `Function` δημιουργείται όταν συναντούμε τη δήλωση συνάρτησης, αλλά το πεδίο `startingQuad` του αντικειμένου αυτού συμπληρώνεται όταν μεταφραστεί η πρώτη εκτελέσιμη τετράδα ενδιάμεσου κώδικα της συνάρτηση.

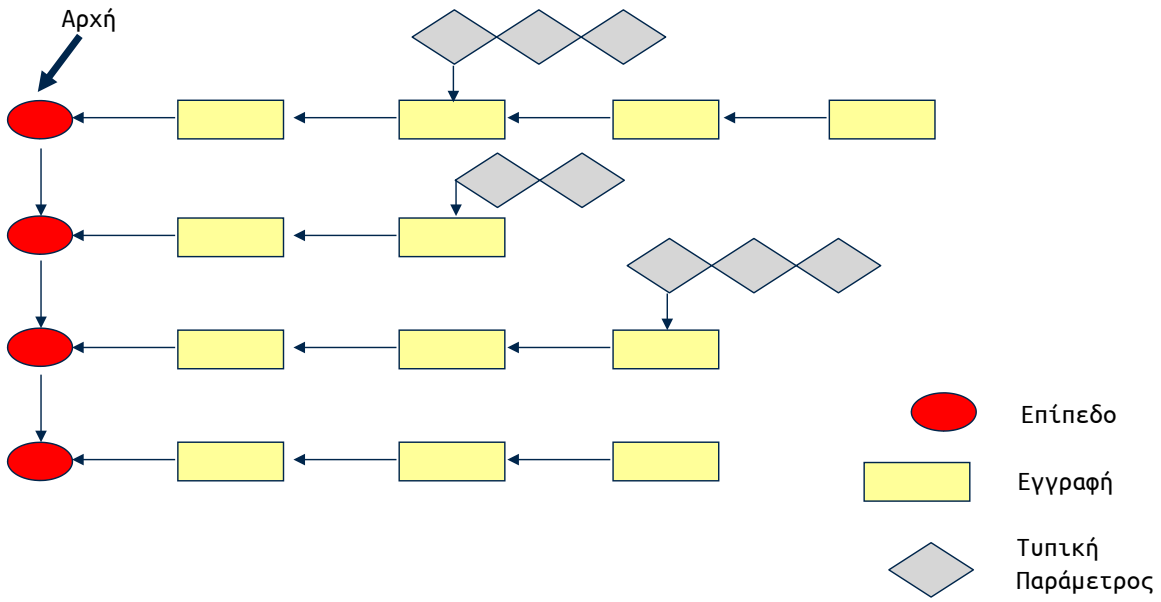
1.2 Η δομή του πίνακα συμβόλων

Η δομή του πίνακα συμβόλων εξαρτάται από τη γλώσσα που υλοποιούμε. Εδώ θα δούμε τη δομή ενός πίνακα συμβόλων κατάλληλο για την *C-imple*. Η *C-imple* υποστηρίζει φωλιασμένες συναρτήσεις και διαδικασίες, κάτι που κάνει την δομή του πίνακα συμβόλων πολύπλοκη και ενδιαφέρουσα. Αν κατανοήσετε τις απαιτήσεις και τον τρόπο που υλοποιούμε και χρησιμοποιούμε τον πίνακα συμβόλων της *C-imple*, εύκολα θα μπορέσετε να υλοποιήσετε πίνακα συμβόλων για οποιαδήποτε γλώσσα, ενώ το πιθανότερο είναι οι λειτουργίες αυτές και η δομή του πίνακα να είναι κάποιο υποσύνολο αυτών της *C-imple*.

Ο πίνακας συμβόλων αποτελείται από επίπεδα τα οποία ας ονομάσουμε *scope*. Ο όρος σημαίνει *εμβέλεια* και κάθε τέτοιο επίπεδο αντιστοιχεί στη μετάφραση μίας συνάρτησης. Όταν ξεκινάει η μετάφραση μίας συνάρτησης, τότε δημιουργείται ένα νέο επίπεδο (scope). Όταν τερματίζεται η μετάφραση μίας συνάρτησης, τότε αφαιρείται το επίπεδο της από τον πίνακα συμβόλων. Όταν δύο συναρτήσεις `f1,f2` είναι φωλιασμένες η μία μέσα στην άλλη με εσωτερικότερη την `f2`, τότε κατά τη μετάφραση της `f2` στον πίνακα συμβόλων υπάρχουν τα επίπεδα και για την `f1` και για την `f2`.

Αν θεωρήσουμε ότι η `f2` ανήκει στο κυρίως πρόγραμμα τότε θα δημιουργηθεί πρώτα ένα επίπεδο για για το κυρίως πρόγραμμα, όταν ξεκινήσει η μετάφραση της `f1` θα δημιουργηθεί ένα επίπεδο για αυτήν, όταν ξεκινήσει η μετάφραση της `f2` θα δημιουργηθεί και ένα επίπεδο για την `f2`. Έτσι, τη στιγμή της μετάφρασης της `f2` θα υπάρχουν τρία επίπεδα στον πίνακα συμβόλων. Όταν τελειώσει η μετάφραση της `f2` θα αφαιρεθεί ένα επίπεδο από τον πίνακα συμβόλων, αυτό που αντιστοιχούσε στην `f2`. Τη στιγμή της μετάφρασης της `f1` θα υπάρχουν δύο επίπεδα στον πίνακα συμβόλων. Όταν τελειώσει η μετάφραση της `f1` θα αφαιρεθεί ένα επίπεδο από τον πίνακα συμβόλων, αυτό που αντιστοιχεί στην `f1`. Τη στιγμή της μετάφρασης του κυρίως προγράμματος θα υπάρχει μόνο ένα επίπεδο στον πίνακα συμβόλων. Με την προσθαφαίρεση επιπέδων επιτυγχάνουμε να έχουμε μέσα στον πίνακα συμβόλων, σε κάθε στιγμή της μετάφρασης, ακριβώς την πληροφορία που πρέπει να έχει, με βάση την περιγραφή και τους κανόνες της γλώσσας.

Η δομή ενός πίνακα συμβόλων εικονίζεται στο σχήμα 1.2.



Σχήμα 1.2: Η δομή του πίνακα συμβόλων

Όλα θα ξεκαθαρίσουν με το παράδειγμα μετατροπής του πίνακα συμβόλων που υπάρχει στο τέλος του κεφαλαίου. Αλλά πριν φτάσουμε ως εκεί, έχουμε ακόμα κάποια πράγματα που πρέπει να κατανοήσουμε, αλλά και να δούμε πώς μπορούμε να εκμεταλλευτούμε την αντικειμενοστρεφή σχεδίαση για να υλοποιήσουμε έναν πίνακα συμβόλων με βάση τις αρχές της.

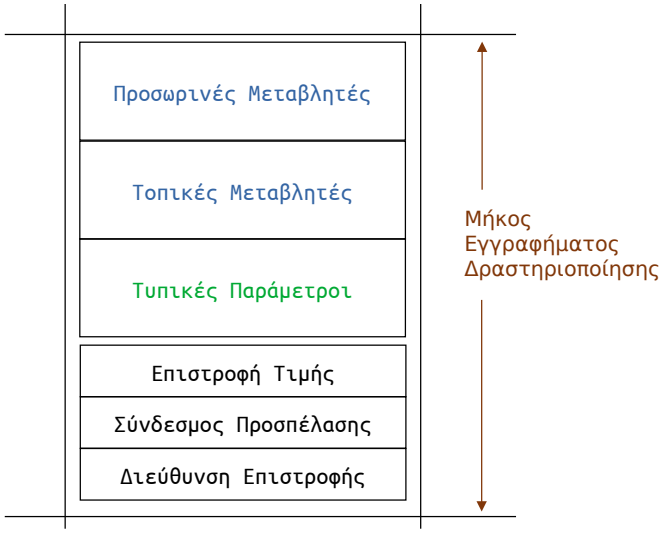
1.3 Το εγγράφημα δραστηριοποίησης

Το *εγγράφημα δραστηριοποίησης* (activation record) δημιουργείται για κάθε συνάρτηση που πρόκειται να κληθεί και καταστρέφεται με την ολοκλήρωση της εκτέλεσης της συνάρτησης. Πρόκειται για το χώρο ο οποίος παραχωρείται στη συνάρτηση αυτή στη στοίβα, προκειμένου να τοποθετήσει εκεί ζωτικές πληροφορίες για την εκτέλεσή της. Στο εγγράφημα δραστηριοποίησης αποθηκεύονται επίσης οι πραγματικές παράμετροι (actual parameters) της συνάρτησης, οι τοπικές μεταβλητές και οι προσωρινές μεταβλητές. Ας δούμε περισσότερο αναλυτικά τις πληροφορίες αυτές.

Στην πρώτη θέση του εγγραφήματος δραστηριοποίησης αποθηκεύεται η *διεύθυνση επιστροφής* της συνάρτησης. Όχι βέβαια με κάποιο μαγικό τρόπο. Εμείς είμαστε υπεύθυνοι να το κάνουμε αυτό. Στην ενότητα αυτή δεν μας ενδιαφέρει που θα βρεθεί, πότε και με ποιο τρόπο θα αξιοποιηθεί αυτή η πληροφορία. Το μόνο που μας ενδιαφέρει είναι πόσος χώρος πρέπει να δεσμευτεί για αυτήν. Έτσι, η πρώτη θέση του εγγραφήματος δραστηριοποίησης καταλαμβάνεται από τη διεύθυνση επιστροφής και έχει μέγεθος που απαιτείται για να αποθηκεύσουμε μία διεύθυνση στη μνήμη.

Στο σημείο αυτό θα εκμεταλλευτούμε τον επεξεργαστή που θα χρησιμοποιήσουμε και την περιγραφή της *C-imple* για να απλοποιήσουμε λίγο τις περιγραφές. Χωρίς βλάβη της γενικότητας, θα παρατηρήσουμε ότι στον επεξεργαστή μας μίας διεύθυνση είναι 4 bytes και ο μοναδικός τύπος της *C-imple*, ο ακέραιος, είναι επίσης 4 bytes. Έτσι κάθε θέση στο εγγράφημα δραστηριοποίησης θα αποτελείται από 4 bytes. Αυτό μας βοηθάει μόνο στην απλοποίηση της περιγραφής, αφού αν χρειαζόμασταν θέσεις στο εγγράφημα δραστηριοποίησης μικρότερες ή μεγαλύτερες των 4 bytes, αυτό δεν θα ήταν πρόβλημα, απλά θα έπρεπε να το λαμβάνουμε υπόψη όταν υπολογίζουμε τη διεύθυνση μιας μεταβλητής.

Στη δεύτερη θέση του εγγραφήματος δραστηριοποίησης τοποθετείται ο *σύνδεσμος προσπέλασης*. Ο σύνδεσμος προσπέλασης είναι ένα δείκτης ο οποίος δείχνει στο εγγράφημα δραστηριοποίησης στο οποίο πρέπει να αναζητήσει η συνάρτηση μία μεταβλητή ή παράμετρο η οποία δεν της ανήκει, αλλά στην οποία, σύμφωνα με τους κανόνες εμβέλειας της γλώσσας, έχει πρόσβαση. Όταν λέμε *δεν της ανήκει* εννοούμε ότι αυτή δεν



Σχήμα 1.3: Η δομή του εγγραφήματος δραστηριοποίησης

βρίσκεται στο δικό της εγγράφημα δραστηριοποίησης αλλά σε κάποιο άλλο, στο οποίο όμως ξανατονίζουμε ότι έχει πρόσβαση με βάση την περιγραφή της γλώσσας. Η αναζήτηση είναι αναδρομική, αν δηλαδή η πληροφορία αυτή δε βρεθεί στο εγγράφημα δραστηριοποίησης που δείχνει ο σύνδεσμος προσπέλασης, τότε θα αναζητηθεί με βάση το δικό του σύνδεσμο προσπέλασης και ούτω καθεξής. Στην *C-imple* κάθε συνάρτηση ή διαδικασία έχει πρόσβαση στις δικές της μεταβλητές και παραμέτρους, στις τοπικές μεταβλητές και παραμέτρους του γονέα της και σε κάθε άλλη συνάρτηση ή διαδικασία που ανήκει στο γενεαλογικό της δέντρο, συμπεριλαμβανομένων και των καθολικών μεταβλητών. Ο αναδρομικός αυτός τρόπος πρόσβασης δείχνει να ταιριάζει απόλυτα στη *C-imple*.

Στην τρίτη θέση του εγγραφήματος δραστηριοποίησης δεσμεύουμε χώρο για την *επιστροφή τιμής της συνάρτησης*. Εκεί θα αποθηκευτεί η διεύθυνση της μεταβλητής στην οποία θα επιστραφεί η τιμή της συνάρτησης. Αν το εγγράφημα δραστηριοποίησης ανήκει σε διαδικασία, τότε η τρίτη θέση του θα μείνει κενή. Φυσικά αν θέλαμε να κάνουμε οικονομία στη μνήμη, θα μπορούσαμε να αναζητήσουμε σχεδιασμό στον οποίο δεν θα δεσμεύαμε καθόλου χώρο στη στοίβα για την επιστροφή τιμής, όταν πρόκειται για διαδικασία. Θα προτιμήσουμε να διατηρήσουμε την ομοιομορφία στη σχεδίαση και όχι την οικονομία στη μνήμη, αν και κάθε λύση έχει τα δικά της πλεονεκτήματα.

Στις τρεις θέσεις που έχουμε συζητήσει μέχρι τώρα έχουμε αποθηκεύσει τρεις διευθύνσεις, συνολικά 12 bytes. Μετά από αυτά τα 12 bytes αρχίζουμε να τοποθετούμε τις παραμέτρους και τις μεταβλητές. Πρώτα τις παραμέτρους.

Για κάθε μία παράμετρο δεσμεύουμε 4 bytes. Ανάλογα με το αν θα τοποθετηθεί εκεί παράμετρος με τιμή ή με αναφορά, στη δεσμευμένη για την παράμετρο θέση θα τοποθετηθεί η τιμή της ή η διεύθυνσή της. Αυτό όμως δεν μας απασχολεί τώρα. Αυτό που μας νοιάζει είναι ο χώρος που θα καταλάβει αυτή, ποιος θα είναι και πόσος θα είναι. Οι παράμετροι τοποθετούνται στη στοίβα με τη σειρά εμφάνισης τους στις τυπικές παραμέτρους της συνάρτησης ή της διαδικασίας και για κάθε μία απαιτούνται 4 bytes στη *C-imple*, αφού υποστηρίζει μόνο ακέραιους αριθμούς. Στη γενική περίπτωση, θα χρειαζόταν να δεσμεύσουμε τόσα bytes, όσα απαιτούνται ανάλογα με την τύπο δεδομένων της παραμέτρου.

Τα ίδια ισχύουν και στη δέσμευση χώρου για τις μεταβλητές. Οι μεταβλητές τοποθετούνται στη στοίβα με τη σειρά εμφάνισης τους στη δήλωση των τοπικών μεταβλητών της συνάρτησης ή της διαδικασίας και για κάθε μία μεταβλητή απαιτούνται 4 bytes, όταν πρόκειται για τη *C-imple*.

Η τελευταία ομάδα που καταλαμβάνει χώρο στη στοίβα είναι οι προσωρινές μεταβλητές. Μπορεί να δημιουργούνται από τον μεταγλωττιστή, χρειάζονται όμως χώρο για να αποθηκευτεί η τιμή τους. Καθότι είναι και αυτές μεταβλητές όπως και οι τοπικές, τις διαχειριζόμαστε με τον ίδιο τρόπο.

Στο σχήμα 1.3 μπορείτε να δείτε την δομή του εγγραφήματος δραστηριοποίησης.

Στο σημείο θα εισάγουμε δύο νέους όρους που θα χρησιμοποιήσουμε αργότερα. Ο πρώτος είναι το *μήκος του εγγραφήματος δραστηριοποίησης*. Πρόκειται για τον συνολικό χώρο, σε bytes, που καταλαμβάνει το εγγράφημα δραστηριοποίησης της στοίβα. Το μήκος αυτό γίνεται γνωστό αφότου γνωρίζουμε τον αριθμό των παραμέτρων, των μεταβλητών και των προσωρινών μεταβλητών μίας συνάρτησης ή διαδικασίας, όταν δηλαδή τελειώσει η μετάφραση του ενδιάμεσου κώδικα για το υποπρόγραμμα αυτό. Το μήκος του εγγραφήματος δραστηριοποίησης σημειώνεται επίσης στο σχήμα 1.3.

Ο δεύτερος όρος είναι η *απόσταση από την αρχή του εγγραφήματος δραστηριοποίησης* (offset) και δεν είναι τίποτε άλλο από ό,τι λέει ο όρος. Πρόκειται για την απόσταση σε bytes μίας μεταβλητής από την αρχή του εγγραφήματος. Αν πρόκειται για τη *C-imple*, για μία συνάρτηση με 2 παραμέτρους, 2 τοπικές μεταβλητές και μία προσωρινή, τα offset που αντιστοιχούν στις οντότητες αυτές είναι 12, 16, 20, 24 και 28 αντίστοιχα, ενώ το μήκος του εγγραφήματος δραστηριοποίησης είναι 32. Ο λόγος που η πρώτη παράμετρος τοποθετήθηκε στο 12, είναι ότι τα 12 πρώτα bytes είναι δεσμευμένα για τη διεύθυνση επιστροφής, τον σύνδεσμο προσπέλασης και την επιστροφή τιμής.

1.4 Οι λειτουργίες του πίνακα συμβόλων

Ας δούμε τι ενέργειες κάνει ένας πίνακας συμβόλων. Όπως είπαμε παραπάνω, στον πίνακα συμβόλων προσθαφαιρούμε πληροφορία, ώστε να πετύχουμε το ζητούμενο, σε κάθε σημείο της παραγωγής τελικού κώδικα, ο πίνακας συμβόλων να περιέχει ακριβώς εκείνες τις εγγραφές τις οποίες το μεταφραζόμενο πρόγραμμα έχει δικαίωμα να προσπελάσει τη συγκεκριμένη χρονική στιγμή, με βάση τους κανόνες της γλώσσας.

Οι βασικές λειτουργίες του πίνακα συμβόλων είναι οι ακόλουθες:

- *πρόσθεση νέας εγγραφής*: Γίνεται όταν συναντάμε τη δήλωση μιας σταθεράς, μεταβλητής, διαδικασίας ή συνάρτησης, παραμέτρου ή δημιουργούμε μια νέα προσωρινή μεταβλητή. Η νέα εγγραφή προστίθεται στην τελευταία θέση, στο ανώτερο επίπεδο του πίνακα συμβόλων, στο επίπεδο δηλαδή της συνάρτησης ή της διαδικασίας που μεταφράζεται τη στιγμή αυτή.

- *πρόσθεση νέου επιπέδου*: Ένα νέο επίπεδο δημιουργείται όταν ξεκινάει η μετάφραση μιας συνάρτησης, διαδικασίας ή του κυρίως προγράμματος. Το νέο επίπεδο δημιουργείται με τη λογική της στοίβας, πάνω από το επίπεδο που μέχρι στιγμής έχει δημιουργηθεί.

- *αφαίρεση επιπέδου*: γίνεται όταν ολοκληρώνουμε τη μετάφραση μίας συνάρτησης ή διαδικασίας ή του κυρίως προγράμματος. Ολόκληρο το επίπεδο που αντιστοιχεί στη συνάρτηση ή τη διαδικασία ή το κυρίως πρόγραμμα που μόλις μεταφράστηκε αφαιρείται, μαζί φυσικά με όλες τις εγγραφές του επιπέδου αυτού.

- *ενημέρωση πεδίων*: γίνεται όταν πρέπει να συμπληρώσουμε πληροφορία στον πίνακα συμβόλων η οποία δεν ήταν διαθέσιμη κατά τη δημιουργία της εγγραφής, για παράδειγμα τα `framelength` και `startingQuad`. Όταν δημιουργείται η εγγραφή για μία συνάρτηση, διαδικασία ή του κυρίως προγράμματος, δεν γνωρίζουμε το framelength ή το startingQuad. Το framelength το γνωρίζουμε όταν ολοκληρώσουμε τη μετάφραση της συνάρτησης, της διαδικασίας ή του κυρίως προγράμματος, οπότε και πρέπει να το συμπληρώσουμε στο αντίστοιχο πεδίο της εγγραφής. Το ίδιο συμβαίνει και με το startingQuad, το οποίο γίνεται γνωστό και μπορεί να συμπληρωθεί μόλις μεταφραστεί η πρώτη εκτελέσιμη εντολή της συνάρτησης.

- *πρόσθεση τυπικής παραμέτρου*: θα μπορούσε να ανήκει και στην κατηγορία της ενημέρωσης πεδίων, αφού πρακτικά ενημερώνουμε κάποια εγγραφή, αλλά θα προτιμούμε να τη διαχωρίσουμε. Γίνεται όταν συναντάμε τη δήλωση μίας τυπικής παραμέτρου μίας συνάρτησης ή διαδικασίας η οποία προστίθεται στη λίστα με τις τυπικές παραμέτρους της εγγραφής που αντιστοιχεί στη συνάρτηση ή τη διαδικασία.

- *αναζήτηση εγγραφής*: γίνεται όταν αναζητούμε πληροφορία στον πίνακα συμβόλων. Η αναζήτηση στον πίνακα συμβόλων γίνεται με το όνομα της εγγραφής. Η αναζήτηση ξεκινά από το υψηλότερο επίπεδο και οι εγγραφές του διαπερνιούνται μία προς μία. Αν το προς αναζήτηση όνομα βρεθεί, τότε επιστρέφεται ως το αποτέλεσμα της αναζήτησης. Αν στο επίπεδο αυτό δεν βρεθεί η εγγραφή που αναζητείται, τότε η αναζήτηση συνεχίζεται στο αμέσως επόμενο επίπεδο. Η διαδικασία αυτή συνεχίζεται μέχρι να βρεθεί η ζητούμενη εγγραφή σε κάποιο επίπεδο ή να εξαντληθούν όλα τα επίπεδα στον πίνακα συμβόλων. Στην περίπτωση αυτή έχουμε αναγνωρίσει σφάλμα και πρέπει να καλέσουμε τον διαχειριστή σφαλμάτων.

Με τον μηχανισμό αυτό αναζήτησης υλοποιούμε και την υπερκάλυψη μεταβλητών. Όταν αναζητούμε μία εγγραφή με ένα συγκεκριμένο όνομα και υπάρχουν στον πίνακα συμβόλων περισσότερες από μία εγγραφές με το όνομα αυτό, τότε η αναζήτηση με βάση τα επίπεδα θα επιστρέψει την εγγραφή που αντιστοιχεί στο υψηλότερο επίπεδο. Έτσι, οι τοπικές μεταβλητές και οι παράμετροι που ανήκουν στην υπό μετάφραση συνάρτηση έχουν προτεραιότητα από τις αντίστοιχες εγγραφές που έχουν δηλωθεί στον γονέα, όπως και αυτές που έχουν δηλωθεί στον γονέα, σε σχέση με αυτές που έχουν δηλωθεί σε κάποιον άλλο πρόγονο. Η ιεράρχηση αυτή σταματάει στις καθολικές μεταβλητές, στις οποίες θα ανατρέξουμε αν σε κανένα από τα προηγούμενα επίπεδα δεν υπάρχει η προς αναζήτηση εγγραφή.

1.5 Αντικειμενοστρεφής σχεδίαση

Όπως υπονοήθηκε παραπάνω, αλλά όπως σίγουρα σκεφτήκατε και εσείς διαβάζοντας την προηγούμενη ενότητα, οι κλάσεις των διαφόρων εγγραφών του πίνακα συμβόλων δίνουν ευκαιρίες ιεραρχικής σχεδίασης και εκμετάλλευσης της κληρονομικότητας. Οι λύσεις που μπορούν να δοθούν είναι πολλές, αλλά θα περιοριστούμε στο να παρουσιάσουμε μία, η οποία μας φαίνεται πιο καλά δομημένη. Κάθε άλλη πιθανή λύση μπορεί να έχει θετικά και αρνητικά στοιχεία.

Μία αφηρημένη κλάση, η `Entity` μπορεί να τοποθετηθεί υψηλότερα στην ιεραρχία, ώστε να ενοποιήσει εννοιολογικά όλες τις εγγραφές. Υπάρχει μόνο ένα πεδίο κοινό σε όλες τις κλάσεις και αυτό είναι το `name`, οπότε η αφηρημένη κλάση `Entity` θα έχει μόνο ένα πεδίο.

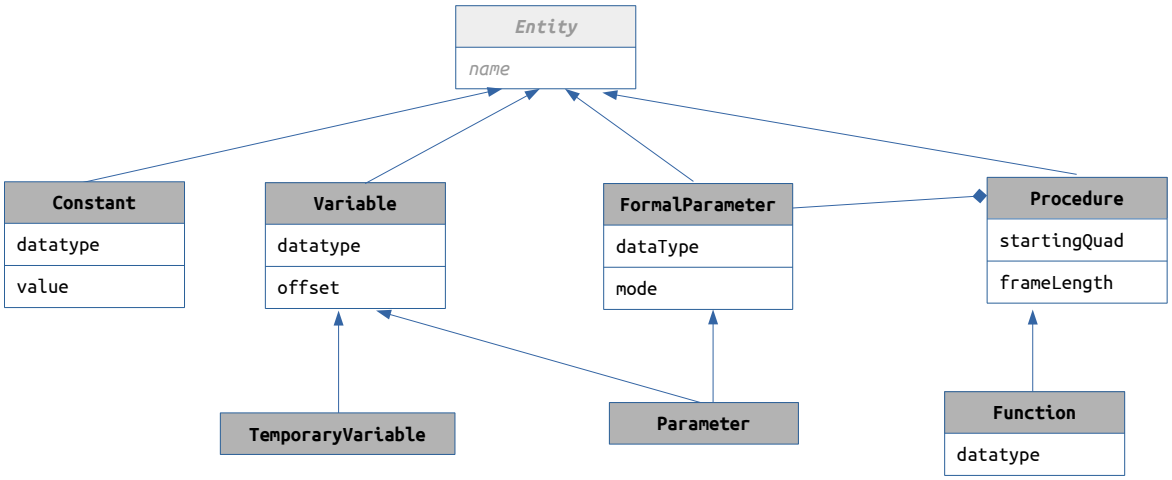
Από την `Entity` είναι λογικό να κληρονομήσει κάθε εγγραφή που αντιστοιχεί σε κάποια βασική έννοια. Δύο τέτοιες έννοιες είναι οι `Variable` αλλά και τα υποπρογράμματα. Κοιτώντας προσεκτικά τα πεδία των κλάσεων `Function` και `Procedure`, θα αντιληφθούμε ότι η `Function` περιέχει όλα τα πεδία της `Procedure` και ακόμα ένα. Άρα είναι λογικό η `Procedure` να είναι αυτή που θα κληρονομήσει από την `Entity` και στη συνέχεια η `Function` από την `Procedure`. Αν το δούμε σε υψηλότερο επίπεδο, μία συνάρτηση είναι μια διαδικασία η οποία μπορεί να επιστρέφει τιμή.

Στη συνέχεια, πρέπει να τοποθετήσουμε κάπου στο δέντρο της ιεραρχίας των κλάσεων τις κλάσεις `Parameter` και `FormalParameter`. Τα πεδία της `FormalParameter` περιέχονται στην `Parameter`, άρα η δεύτερη είναι αυτή που φαίνεται να πρέπει να κληρονομήσει από την πρώτη, η οποία με τη σειρά της να κληρονομήσει από την `Entity`.

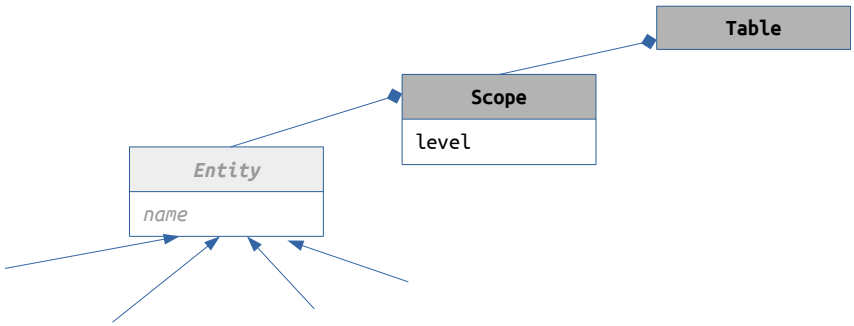
Ιδιαίτερο ενδιαφέρον έχει η `Parameter`. Μία παράμετρος έχει όλα τα χαρακτηριστικά μίας μεταβλητής και έχει ακόμα και έναν τρόπο περάσματος. Μία καλή σχεδιαστική επιλογή είναι να χρησιμοποιήσουμε πολλαπλή κληρονομικότητα και να δώσουμε στην κλάση `Parameter` τα πεδία της `FormalParameter` και της `Variable`. Αν τώρα η γλώσσα μας δεν υποστηρίζει πολλαπλή κληρονομικότητα, θα πρέπει η `Parameter` να κληρονομήσει την `FormalParameter` και στη συνέχεια να προστεθούν σε αυτήν τα πεδία που λείπουν.

Όπως είχαμε αποφασίσει στην προηγούμενη ενότητα, θα ορίσουμε μία κλάση για τις προσωρινές μεταβλητές, την `TemporaryVariable`. Οι προσωρινές μεταβλητές θα μπορούσαν να ανήκουν στην κλάση `Variable`, αλλά για λόγους καλύτερης σχεδίασης δεν το κάναμε αυτό. Μία εύκολη και σχεδιαστικά σωστή επιλογή είναι οι προσωρινές μεταβλητές να κληρονομούν από την κλάση `Variable`.

Μας απομένει ακόμα η `SymbolicConstant`. Είναι κάτι αρκετά διαφορετικό από τις υπόλοιπες κλάσεις, αφού ναι μεν θυμίζει μεταβλητή, αλλά δεν τοποθετείται στη στοίβα. Η τιμή της είναι γνωστή σε χρόνο με-



Σχήμα 1.4: Η ιεραρχία των κλάσεων των εγγραφών του πίνακα συμβόλων



Σχήμα 1.5: Οι σχέσεις ανάμεσα στις κλάσεις `Table`, `Scope` και `Entity`

τάφρασης και τότε μας χρειάζεται και όχι σε χρόνο εκτέλεσης. Όπως μας οδηγούν τα πεδία των κλάσεων `SymbolicConstant` και `Variable` αλλά και οι προδιαγραφές κάθε κλάσης, η `SymbolicConstant` θα θεωρηθεί κληρονομικά μη σχετιζόμενη με κάθε άλλη μη αφηρημένη κλάση και θα κληρονομήσει απευθείας από την `Entity`.

Η προτεινόμενη σχεδίαση απεικονίζεται στο σχήμα 1.4.

Αν θέλουμε να προχωρήσουμε λίγο περισσότερο στη σχεδίαση, θα διακρίναμε μία κλάση για τον πίνακα, την οποία θα ονομάζαμε `Table` και μία ακόμα για τα επίπεδα, την `Scope`. Η `Scope` και η `Entity` έχουν σχέση συναρμολόγησης/σύνθεσης, αφού ένα επίπεδο αποτελείται από πολλές εγγραφές, ενώ για τον ίδιο λόγο υπάρχει η ίδια σχέση ανάμεσα στην `Table` και την `Scope`. Στην `Scope` χρειαζόμαστε και ένα πεδίο το οποίο θα κρατά τον αύξοντα αριθμό του επιπέδου, θα μπορούσαμε βέβαια να υλοποιήσουμε το πρόγραμμά μας και χωρίς αυτό. Οι σχέσεις αυτών των κλάσεων εικονίζονται στο σχήμα 1.5

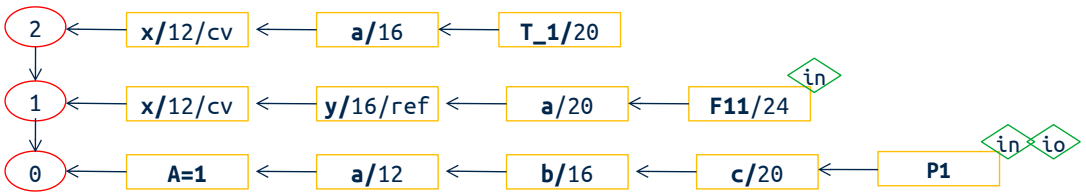
1.6 Παράδειγμα λειτουργίας του πίνακα συμβόλων

Στην ενότητα αυτή θα δούμε ένα πλήρες παράδειγμα λειτουργίας ενός πίνακα συμβόλων. Ως είσοδος θα χρησιμοποιηθεί το παράδειγμα σε γλώσσα *C-imple* που φαίνεται στο σχήμα 1.6. Το παράδειγμα είναι κατάλληλα κατασκευασμένο ώστε να γίνουν κατανοητές οι λεπτομέρειες λειτουργίας, κυρίως ο τρόπος και ο χρόνος που εισάγονται οι εγγραφές, ο τρόπος και ο χρόνος που αφαιρούνται τα επίπεδα, ο τρόπος και ο χρόνος που συμπληρώνεται πληροφορίας στον πίναα συμβόλων και οι χρονικές στιγμές κατά τις οποίες γίνεται αναζήτηση πληροφορίας. Το παράδειγμα δεν υλοποιεί κάτι χρήσιμο, αλλά αυτό δεν μας ενδιαφέρει καθόλου.

Ξεκινώντας από την αρχή του αρχικού προγράμματος του σχήματος 1.6, ο συντακτικός αναλυτής περνάει από το `program`, όπου και αντιλαμβάνεται ότι πρέπει να δημιουργήσει το πρώτο επίπεδο, το επίπεδο 0. Προ-

```
program symbol
{   const A=1;
    declare a,b,c;

    procedure P1(in x, inout y)
    {   declare a;

        function F11(in x);
        {
            declare a;
            # body of F11 #
            b = a;
            a = x;
            c = F11(in x);
            return (c);
        }

        function F12(in x)
        {   # body of F12 #
            c = F11(in x);
            return (c);
        }

        # body of P1 #
        y = x;
    }

    procedure P2(inout x)
    {   declare x;
        # body of P2 #
        y = A;
        call P1(in x, inout y);
    }

    # main program #
    call P1(in a, inout b);
    call P2(in c);
}
```

Σχήμα 1.6: Αρχικός κώδικας παραδείγματος λειτουργίας του πίνακα συμβόλων



Σχήμα 1.7: Στιγμιότυπο του πίνακα συμβόλων μετά την ολοκλήρωση της μετάφρασης της `F11`

χωρώντας παρακάτω θα συναντήσει το `const` το οποίο θα εισάγει στον πίνακα συμβόλων σαν την πρώτη εγγραφή του επιπέδου 0. Στη *C-imple* επιλέξαμε οι σταθερές να εισάγονται στον πίνακα συμβόλων. Μία διαφορετική προσέγγιση μπορεί να συναντήσει κανείς στην προγραμματιστική γλώσσα C, όπου οι σταθερές (`#define`) διαχειρίζονται από έναν προεπεξεργαστή, ο οποίος διαπερνά όλο το πρόγραμμα και αντικαθιστά τα ονόματα των σταθερών με τις τιμές τους. Αφού ολοκληρωθεί αυτή η διαδικασία και δεν υπάρχουν πια σταθερές στο πρόγραμμα C, τότε καλείται ο μεταφραστής της γλώσσας. Στο δικό μας παράδειγμα, μία εγγραφή σταθεράς, με όνομα `A` και τιμή 1 θα εισαχθεί στον πίνακα συμβόλων. Οι σταθερές δεν τοποθετούνται στη στοίβα, για το λόγο αυτόν δεν έχουν offset. Οι σταθερές διαχειρίζονται από τον μεταγλωττιστή σε χρόνο μετάφρασης και δεν υπάρχει κανένας λόγος να τοποθετηθούν στη στοίβα. Όσο διαρκεί η μετάφραση, ο πίνακας συμβόλων είναι διαθέσιμος για να ανατρέξουμε σε αυτόν και να αναζητήσουμε την τιμή (και τον τύπο για τις γλώσσες που έχει νόημα) της σταθεράς.

Στη συνέχεια συναντώνται οι δηλώσεις για τις μεταβλητές `a`,`b` και `c`. Τοποθετούνται στον πίνακα συμβόλων και υπολογίζονται για τις θέσεις με offset 12,16 και 20, οι οποίες σημειώνονται και στο αντίστοιχο πεδίο.

Στο σχήμα 1.7 μπορούμε να παρακολουθήσουμε τον σχηματισμό του πίνακα συμβόλων. Μέχρι στιγμής έχουμε δημιουργήσει ένα επίπεδο, το επίπεδο 0 και έχουμε τοποθετήσει σε αυτό τέσσερις εγγραφές. Η συντομογραφία που ακολουθείται για τις μεταβλητές μέσα στο κίτρινο παραλληλόγραμμο που υποδηλώνει την εγγραφή είναι *name/offset*. Η συντομογραφία που χρησιμοποιούμε για τις παραμέτρους είναι *name/offset/-mode*, ενώ για τις συναρτήσεις και τις διαδικασίες `name` ή `name/framelength`, όταν γνωρίζουμε το framelength. Στο σχήμα αυτό δεν σημειώνουμε το startingQuad, για λόγους διαχείρισης χώρου. Πάνω δεξιά από το παραλληλόγραμμο που συμβολίζει τις συναρτήσεις και τις διαδικασίες, τοποθετούμε με ένα ρόμβο τις τυπικές παραμέτρους της συνάρτησης και μέσα σε αυτό σημειώνουμε τον τρόπο περάσματος. Για λόγους επίσης διαχείρισης χώρου, δεν σημειώνουμε το όνομα της τυπικής παραμέτρου. Η σειρά με την οποία εμφανίζονται οι ρόμβοι αντιστοιχεί στη σειρά με την οποία εμφανίζονται οι τυπικές παράμετροι στη δήλωση της συνάρτησης.

Στη συνέχεια συναντάμε την διαδικασία `P1`. Με το που συναντάμε την έναρξη της διαδικασίας, προσθέτουμε την εγγραφή για τη διαδικασία στο τρέχον επίπεδο και δημιουργούμε ένα νέο επίπεδο για τη μετάφραση της διαδικασίας αυτής. Στη συνέχεια συναντάμε την παράμετρο *in x*. Για την παράμετρο αυτή, όπως και για κάθε άλλη παράμετρο, εισάγουμε μία εγγραφή στο ανώτερο επίπεδο που έχουμε δημιουργήσει, το επίπεδο 1 στη συγκεκριμένη περίπτωση και τοποθετούμε την τυπική παράμετρο στην εγγραφή που αντιστοιχεί στη διαδικασία. Στο παράδειγμά μας πρόκειται για τον πρώτο από τους δύο ρόμβους στην εγγραφή `P1`, αυτόν με την ένδειξη `in` ως τρόπο περάσματος. Συμπληρώνουμε και τα υπόλοιπα πεδία στην εγγραφή, άρα έχουμε τις τιμές x/12/cv για την εγγραφή αυτή. Όμοια για την παράμετρο `y` θα εισάγουμε την εγγραφή για την παράμετρο, θα τοποθετήσουμε τις τιμές y/16/ref σε αυτήν και θα προσθέσουμε και τον ρόμβο για την τυπική παράμετρο, στη λίστα με τις τυπικές παραμέτρους της συνάρτησης.

Μετά ακολουθεί η δήλωση της τοπικής μεταβλητής `a`. Οι δύο παράμετροι έχουν καταλάβει τις θέσεις 12 και 16, άρα η επόμενη διαθέσιμη θέση, την οποία και θα καταλάβει η `a` είναι η 20. Εισάγουμε στον πίνακα και την εγγραφή αυτή και έχουμε τελειώσει με τις τοπικές μεταβλητές. Αν κοιτάξουμε στο σχήμα 1.7, έχουμε φτάσει στο σημείο που έχουν εισαχθεί οι πέντε εγγραφές του επιπέδου 0 και οι τρεις πρώτες εγγραφές του επιπέδου 1.

Φωλιασμένη μέσα στην `P1` υπάρχει η συνάρτηση `F11`. Θα ακολουθήσουμε την ίδια διαδικασία. Θα δημιουργηθεί η εγγραφή για την `F11` στο τρέχον επίπεδο του πίνακα συμβόλων, θα δημιουργηθεί ένα νέο επίπεδο για τη μετάφραση της `F11`, θα δημιουργηθεί στο νεό επίπεδο η εγγραφή για την παράμετρο x, με offset=12 και mode=cv, και θα τοποθετηθεί και στη λίστα με τις παραμέτρους στην εγγραφή για τη συνάρτηση

F11. Η μία και μοναδική τοπική μεταβλητή θα έχει offset=16 και θα πάρει και αυτή τη θέση της στον πίνακα συμβόλων.

Αφού ο συντακτικός αναλυτής περάσει και από τη δήλωση της a, τότε θα συναντήσει την πρώτη εκτελέσιμη εντολή της F11. Τώρα είναι η κατάλληλη χρονική στιγμή για να συμπληρωθεί το startingQuad της F11. Για οικονομία χώρου δεν το σημειώνουμε στο σχήμα 1.7, αλλά η ενέργεια αυτή πρέπει να γίνει. Ακολουθώντας γραμμή προς γραμμή τον κώδικα της *C-imple* παράγεται ο αντίστοιχος ενδιάμεσος κώδικας για την F11. Κάθε φορά που απαιτείται η δημιουργία μιας νέας μεταβλητής, αυτή εισάγεται στην επόμενη διαθέσιμη θέση του πίνακα συμβόλων. Στο παράδειγμά μας, υπάρχει ανάγκη μονάχα για μία τέτοια στην F11, την T_1, η οποία προκύπτει από την ανάγκη της επιστροφής της τιμής της συνάρτησης, στη γραμμή με την αναδρομική κλήση. Αφού η a κατέλαβε τη θέση 16, η T_1 αντιστοιχίζεται με τη θέση 20.

Με την ολοκλήρωση της μετάφρασης του ενδιάμεσου κώδικα για την F11, γνωρίζουμε και το framelength της. Με 12 bytes για τη διεύθυνση επιστροφής, τον σύνδεσμο προσπέλασης και την επιστροφή της τιμής, με 4 bytes για την παράμετρο, 4 για την τοπική μεταβλητή και 4 για την προσωρινή, σχηματίζεται ένα εγγράφημα δραστηριοποίησης 24 bytes. Άρα σημειώνουμε για την F11 framelength=24. Η εικόνα που έχει τη στιγμή αυτή ο πίνακας συμβόλων είναι ακριβώς αυτή του σχήματος 1.7.

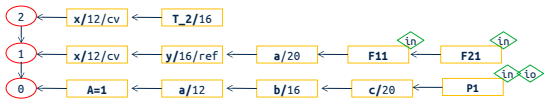
Το σημείο αυτό της μετάφρασης είναι πολύ χαρακτηριστικό για την F11. Αν παρατηρήσουμε τον πίνακα συμβόλων, τη στιγμή αυτή, έχει όλη την πληροφορία που έχει δικαίωμα να δει η F11 και μόνο αυτήν. Έτσι, μπορεί να δει τη δική της παράμετρο x, τη δική της τοπική μεταβλητή a και την προσωρινή μεταβλητή T_1. Από τον γονέα της μπορεί να δει την παράμετρο y που έχει περαστεί εκεί σαν παράμετρος με αναφορά. Θα μπορούσε να δει και την παράμετρο x και την τοπική, στον γονέα, μεταβλητή a, αν αυτές δεν είχαν το ίδιο όνομα με τις δηλωμένες στο δικό της χώρο x και a, οι οποίες σύμφωνα με τους κανόνες εμβέλειας υπερκαλύπτουν τις ομώνυμες δηλωμένες στον γονέα. Φυσικά, προσωρινές μεταβλητές δηλωμένες στον γονέα δεν έχει δικαίωμα να δει, άλλωστε αυτές δεν έχουν ακόμα εισαχθεί στον πίνακα συμβόλων. Τέλος, σύμφωνα πάντα με τον πίνακα συμβόλων, η F11 έχει δικαίωμα να δει και τις μεταβλητές στο επίπεδο 0, τις καθολικές μεταβλητές δηλαδή. Η a υπερκαλύπτεται από την τοπική μεταβλητή a, αλλά οι b και c μπορεί να προσπελαστούν κανονικά. Το ίδιο ισχύει και για την σταθερά A την τιμή της οποίας μπορεί να διαβάσει χωρίς κανένα πρόβλημα η F11

Αφήσαμε έξω από τη συζήτηση τις εγγραφές για τις F11 και P1. Σύμφωνα με τον πίνακα, η F11 μπορεί να δει ότι αυτές έχουν δηλωθεί. Όμως, στο σημείο αυτό της μετάφρασης, ο πίνακας συμβόλων είναι πλήρως συμπληρωμένος μόνο για την F11, όπου και το framelength αλλά και το startingQuad είναι ήδη γνωστά. Αυτό είναι λογικό, αφού μόνο για την F11 έχει παραχθεί ο ενδιάμεσος κώδικας. Έτσι, η μόνη συνάρτηση ή διαδικασία που μπορεί να κληθεί από την F11 είναι η ίδια η F11, με αναδρομική κλήση.

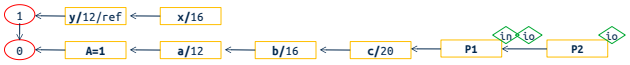
Να σημειώσουμε εδώ, αν και αφορά επόμενο κεφάλαιο, ότι στο σημείο αυτό θα παραχθεί ο τελικός κώδικας για την F11. Αφού γίνει αυτό, το επίπεδο 2, το επίπεδο που έχει δηλαδή δημιουργηθεί για τη μετάφραση της F11, αφαιρείται από τον πίνακα συμβόλων. Άρα καμία άλλη συνάρτηση ή διαδικασία που θα μεταφραστεί αργότερο δεν θα έχει δικαίωμα να δει τις παραμέτρους και τις μεταβλητές, τοπικές ή προσωρινές, της F11, κάτι που είναι σε αρμονία με την περιγραφή της γλώσσας.

Μετά την ολοκλήρωση της μετάφρασης της F11 συνεχίζουμε με την μετάφραση τη P1. Σύμφωνα με τον κώδικα, μέσα στην P1 ακολουθεί η δήλωση της F12, αδελφού της F11. Δημιουργούμε, λοιπόν, μία εγγραφή για την F12 και ένα επίπεδο μετάφρασης για αυτήν. Έχει πάλι το νούμερο 2, αφού το επίπεδο με το ίδιο νούμερο αφαιρέθηκε μετά την ολοκλήρωση της F11. Η F12 έχει μία παράμετρο την x, η οποία θα τοποθετηθεί στο επίπεδό της με offset 12, ως παράμετρος cv. Επίσης θα τοποθετηθεί και σαν τυπική παράμετρος στην εγγραφή της συνάρτησης στο επίπεδο 1. Από το σώμα, και πιο συγκεκριμένα για την επιστροφή τιμής της F11, θα χρειαστούμε μία προσωρινή μεταβλητή, η οποία και αυτή θα εισαχθεί στον πίνακα συμβόλων, στο επίπεδο 2 και με το επόμενο διαθέσιμο offset που είναι το 16.

Τη στιγμή αυτή, μετά την παραγωγή του ενδιάμεσου κώδικα για την F12, πάλι βρισκόμαστε στο σημείο που ο πίνακας είναι συμπληρωμένος για την F12 και μπορεί να παραχθεί ο τελικός κώδικας για αυτήν. Η εικόνα του πίνακα συμβόλων είναι αυτή που φαίνεται στο σχήμα 1.8.



Σχήμα 1.8: Στιγμιότυπο του πίνακα συμβόλων μετά την ολοκλήρωση της μετάφρασης της F12



Σχήμα 1.9: Στιγμιότυπο του πίνακα συμβόλων μετά την ολοκλήρωση της μετάφρασης της P2

Με το τέλος της μετάφρασης της F12, αφαιρείται το επίπεδο 2 από τον πίνακα συμβόλων. Παρατηρείστε ότι παρότι έχουν αφαιρεθεί τα επίπεδα που αντιστοιχούν στη μετάφραση των συναρτήσεων F11 και F12, οι εγγραφές τους από το επίπεδο 1 δεν έχουν αφαιρεθεί. Άρα, αν η P1, θελήσει να τις καλέσει, μπορεί να βρει ό,τι πληροφορία χρειάζεται για αυτές.

Η μετάφραση θα συνεχίσει με την P1. Δεν χρειαζόμαστε προσωρινές μεταβλητές εδώ. Όταν ολοκληρωθεί η μετάφραση του ενδιάμεσου και του τελικού κώδικά της, το επίπεδο 1 θα αφαιρεθεί από τη μνήμη. Γυρίζοντας στις F11 και F12 βλέπουμε ότι κανείς δε μπορεί πια να τις καλέσει, σύμφωνα με την περιγραφή της γλώσσας.

Η μετάφραση της P2 θα δημιουργήσει νέα εγγραφή στο επίπεδο 0 και νέο επίπεδο 1, για τη μετάφραση. Στο τέλος της παραγωγής του ενδιάμεσου κώδικα για την P2, η εικόνα του πίνακα συμβόλων θα είναι αυτή του σχήματος 1.9.

Με την ολοκλήρωση της μετάφρασης της P2, θα αφαιρεθεί το επίπεδό της από τον πίνακα. Η μετάφραση του κυρίως προγράμματος δεν θα δημιουργήσει προσωρινές μεταβλητές. Παρατηρήστε ότι το κυρίως πρόγραμμα μπορεί να δει και να καλέσει μόνο τις διαδικασίες P1 και P2. Με το τέλος της μετάφρασης και του κυρίως προγράμματος, ο πίνακας συμβόλων θα μείνει άδειος.