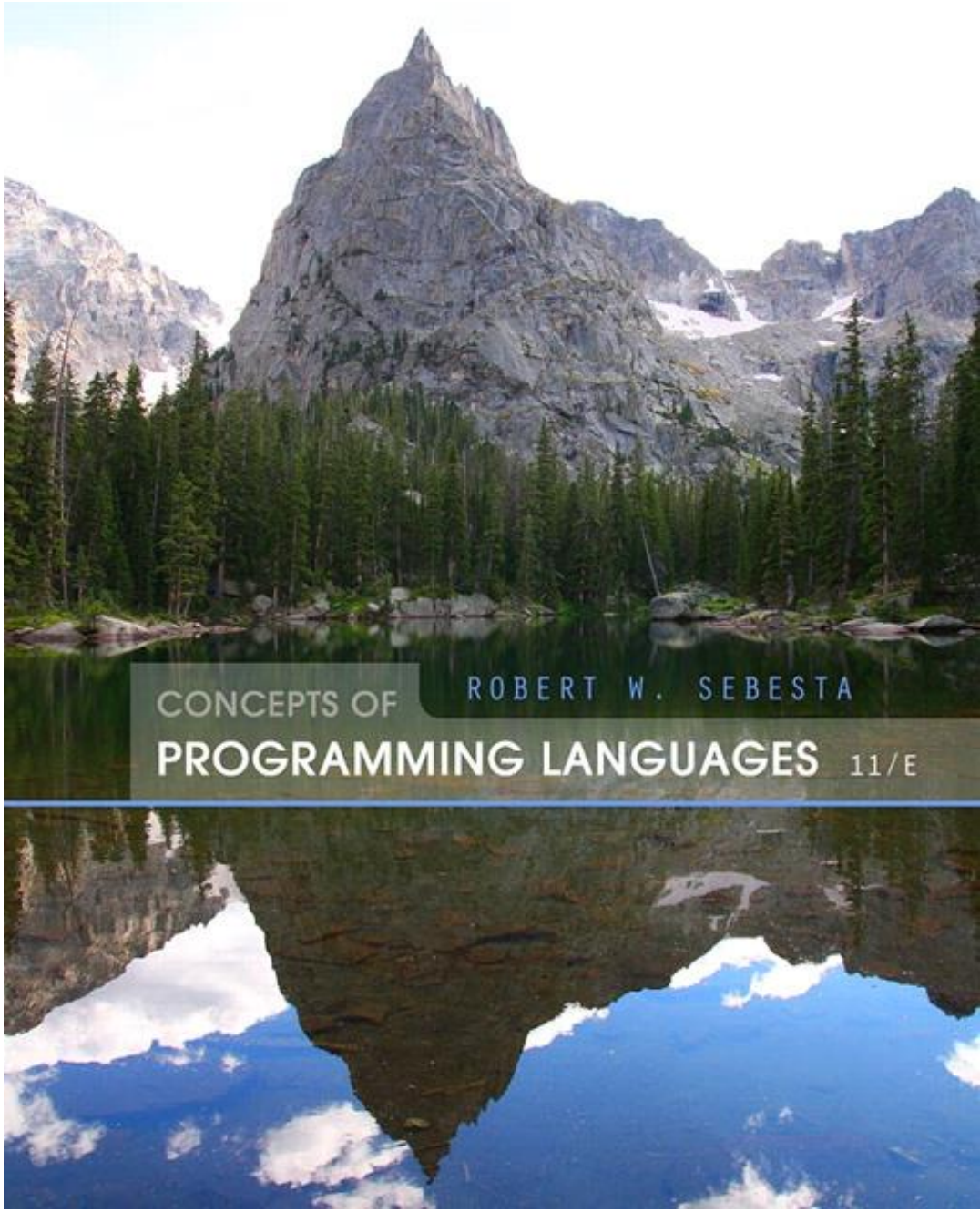
Έκδοση:
7/4/2025

Κεφάλαιο 9

Υποπρογράμματα

Γκόγκος Χρήστος
Τμήμα Πληροφορικής και Τηλεπικοινωνιών (Άρτα)
Πανεπιστήμιο Ιωαννίνων



ISBN 0-321-49362-1

Θέματα Κεφαλαίου 9

- Εισαγωγή
- Βασικές έννοιες Υποπρογραμμάτων
- Σχεδιαστικά θέματα Υποπρογραμμάτων
- Τοπικά Περιβάλλοντα Αναφοράς
- Μέθοδοι περάσματος παραμέτρων
- Παράμετροι που είναι Υποπρογράμματα
- Έμμεση κλήση Υποπρογραμμάτων
- Θέματα Σχεδίασης Συναρτήσεων
- Υπερφορτωμένα Υποπρογράμματα
- Γενικά Υποπρογράμματα
- Υπερφορτωμένοι Τελεστές ορισμένοι από τον Χρήστη
- Κλειστότητες (closures)
- Συρρουτίνες (coroutines)

Εισαγωγή

- Δύο θεμελιώδεις αφαιρέσεις
  - Αφαίρεση διεργασιών
    - Δόθηκε έμφαση από τις πρώτες ημέρες εμφάνισης των γλωσσών προγραμματισμού
  - Αφαίρεση δεδομένων
    - Δόθηκε έμφαση στη δεκαετία του 1980

Θεμελιώδεις έννοιες Υποπρογραμμάτων

- Κάθε υποπρόγραμμα έχει ένα μοναδικό σημείο εισόδου
- Το καλών υποπρόγραμμα αναστέλλεται κατά την εκτέλεση του κληθέντος υποπρογράμματος
- Ο έλεγχος πάντα επιστρέφει στο καλών υποπρόγραμμα όταν η εκτέλεση του κληθέντος υποπρογράμματος τερματίζει

Βασικοί ορισμοί

- Ο ορισμός (definition) υποπρογράμματος περιγράφει τη διεπαφή (interface) του υποπρογράμματος και τις λειτουργίες του
  - Στην Python, οι ορισμοί συναρτήσεων είναι εκτελέσιμοι, στις άλλες γλώσσες είναι μη-εκτελέσιμοι
  - Στην Ruby, οι ορισμοί συναρτήσεων function μπορούν να εμφανίζονται εντός ή εκτός του ορισμού των κλάσεων. Αν βρίσκονται εκτός, είναι μέθοδοι του `Object`. Μπορούν να καλούνται χωρίς object, όπως μια συνάρτηση
  - Στην Lua, όλες οι συναρτήσεις είναι ανώνυμες
- Μια κλήση υποπρογράμματος αποτελεί ένα ρητό αίτημα εκτέλεσης του υποπρογράμματος
- Μια κεφαλίδα υποπρογράμματος είναι το πρώτο μέρος του ορισμού, και περιλαμβάνει το όνομα, το είδος του υποπρογράμματος, και τις τυπικές παραμέτρους
- Το προφίλ παραμέτρων (γνωστό και ως *υπογραφή*) ενός υποπρογράμματος είναι το πλήθος, η σειρά, και οι τύποι των παραμέτρων του
- Το πρωτόκολλο είναι το προφίλ παραμέτρων του υποπρογράμματος και, αν το υποπρόγραμμα είναι συνάρτηση, ο τύπος επιστροφής

Βασικοί ορισμοί (συνέχεια)

- Οι δηλώσεις (declarations) συναρτήσεων στη C και στη C++ συχνά ονομάζονται *πρωτότυπα*
- Η *δήλωση ενός υποπρογράμματος* παρέχει το πρωτόκολλο, αλλά όχι το σώμα του υποπρογράμματος
- Μια *τυπική παράμετρος (formal parameter)* είναι μια μεταβλητή που περιλαμβάνεται σε λίστα στην κεφαλίδα του υποπρογράμματος και χρησιμοποιείται στο υποπρόγραμμα
- Μια πραγματική παράμετρος (*actual parameter)* αναπαριστά μια τιμή ή μια διεύθυνση που χρησιμοποιείται στην εντολή κλήσης του υποπρογράμματος

Αντιστοίχιση Πραγματικών/Τυπικών Παραμέτρων

- Βάσει θέσεων
  - Η πρόσδεση πραγματικών παραμέτρων σε τυπικές παραμέτρους γίνεται με βάση τη θέση: η πρώτη πραγματική παράμετρος προσδένεται στην πρώτη τυπική παράμετρο κ.οκ.
  - Ασφαλές και αποδοτικό
- Βάσει λέξεων κλειδιών (keywords)
  - Το όνομα της τυπικής παραμέτρου στην οποία πρόκειται να προσδεθεί μια πραγματική παράμετρος καθορίζεται μαζί με την πραγματική παράμετρο
  - *Πλεονέκτημα*: Οι παράμετροι μπορούν να εμφανιστούν με οποιαδήποτε σειρά, συνεπώς αποφεύγονται λάθη αντιστοίχισης μεταβλητών
  - *Μειονέκτημα*: Ο χρήστης θα πρέπει να γνωρίζει τα ονόματα των τυπικών παραμέτρων

Προκαθορισμένες τιμές τυπικών παραμέτρων

- Σε ορισμένες γλώσσες (π.χ., C++, Python, Ruby, PHP), οι τυπικές παράμετροι μπορούν να έχουν προκαθορισμένες τιμές (εάν δεν μεταβιβαστεί πραγματική παράμετρος)
  - Στην C++, οι προκαθορισμένες παράμετροι θα πρέπει να εμφανίζονται τελευταίες, διότι οι αντιστοίχιση παραμέτρων γίνεται με βάση τη θέση (δεν διαθέτει παραμέτρους λέξεων κλειδιών)

- Μεταβλητό πλήθος παραμέτρων
  - Οι μέθοδοι της C# μπορούν να δέχονται μεταβλητό αριθμό παραμέτρων αρκεί να είναι του ίδιου τύπου — οι αντίστοιχη τυπική παράμετρος είναι ένα διάνυσμα πριν από το οποίο βρίσκεται η λέξη **`params`**
  - Στην Ruby, οι πραγματικές παράμετροι καταγράφονται ως στοιχεία ενός hash κυριολεκτικού ενώ η αντίστοιχη τυπική παράμετρος περιέχει έναν αστερίσκο στην αρχή της

Μεταβλητό πλήθος παραμέτρων (συνέχεια)

- Στην Python, η πραγματική παράμετρος είναι μια λίστα τιμών και η αντίστοιχη τυπική παράμετρος είναι ένα όνομα με έναν αστερίσκο

- Στην Lua, ένας μεταβλητός αριθμός παραμέτρων αναπαρίσταται ως μια τυπική παράμετρος με τρεις τελείες – προσπελαύνονται με μια εντολή **`for`** ή με πολλαπλές αναθέσεις στις τρεις τελείες

Διαδικασίες και Συναρτήσεις

- Υπάρχουν δύο κατηγορίες υποπρογραμμάτων, οι διαδικασίες και οι συναρτήσεις.
  - *Οι διαδικασίες είναι συλλογές εντολών που ορίζουν παραμετροποιημένους υπολογισμούς*
  - *Οι συναρτήσεις δομικά προσομοιάζουν στις διαδικασίες αλλά σημασιολογικά* αποτελούν μοντέλα μαθηματικών συναρτήσεων
    - Από μια συνάρτηση αναμένεται να μην προκαλεί παράπλευρες συνέπειες (side effects)
    - Ωστόσο, στην πράξη οι συναρτήσεις συχνά προκαλούν side effects

Θέματα Σχεδίασης Υποπρογραμμάτων

- Είναι οι τοπικές μεταβλητές στατικές ή δυναμικές;
- Μπορούν οι ορισμοί των υποπρογραμμάτων να εμφανίζονται σε ορισμούς άλλων υποπρογραμμάτων;
- Ποιες μέθοδοι μεταβίβασης παραμέτρων παρέχονται;
- Παρέχεται έλεγχος τύπων για τις παραμέτρους;
- Αν τα υποπρογράμματα μπορούν να μεταβιβαστούν ως παράμετροι και μπορούν να είναι ένθετα, ποιο είναι το περιβάλλον αναφοράς ενός υποπρογράμματος που μεταβιβάζεται;
- Επιτρέπονται side effects σε συναρτήσεις;
- Τι τύποι τιμών μπορούν να επιστραφούν από συναρτήσεις;
- Πόσες τιμές μπορούν να επιστραφούν από συναρτήσεις;
- Μπορούν τα υποπρογράμματα να υπερφορτωθούν;
- Μπορεί ένα υποπρόγραμμα να είναι γενικό;
- Αν μια γλώσσα επιτρέπει ένθετα υποπρογράμματα, επιτρέπονται οι κλειστότητες (closures);

Τοπικά περιβάλλοντα αναφοράς

- Οι τοπικές μεταβλητές μπορούν να είναι δυναμικές-στοίβας
  - Πλεονεκτήματα
    - Υποστήριξη αναδρομής
    - Ο χώρος αποθήκευσης τοπικών μεταβλητών μοιράζεται ανάμεσα σε υποπρογράμματα
  - Μειονεκτήματα
    - Δέσμευση/αποδέσμευση, χρόνος αρχικοποίησης
    - Έμμεση διευθυνσιοδότηση
    - Τα υποπρογράμματα δεν μπορούν να είναι ευαίσθητα στο ιστορικό κλήσεων τους
- Οι τοπικές μεταβλητές μπορούν επίσης να είναι στατικές
  - Τα πλεονεκτήματα και μειονεκτήματα είναι τα αντίθετα από εκείνα των τοπικών δυναμικών-στοίβας μεταβλητών

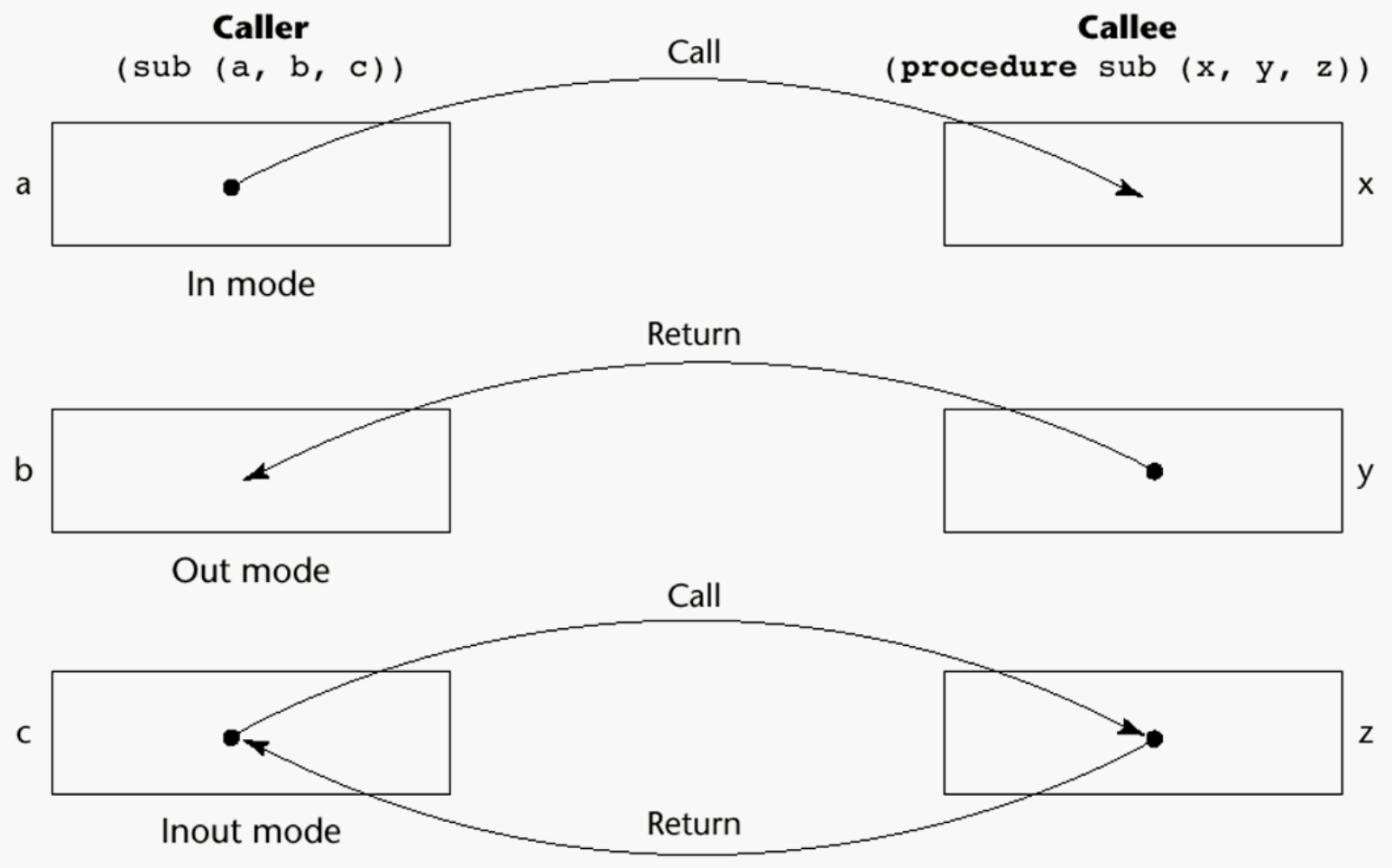
Τοπικά περιβάλλοντα αναφοράς: Παραδείγματα

- Στις περισσότερες σύγχρονες γλώσσες, οι τοπικές μεταβλητές είναι δυναμικές-στοίβας
- Στις C-based γλώσσες, οι τοπικές μεταβλητές προκαθορισμένα είναι δυναμικές-στοίβας, αλλά μπορούν να δηλωθούν και ως `static`
- Οι μέθοδοι στις C++, Java, Python, και C# διαθέτουν μόνο δυναμικές-στοίβας τοπικές μεταβλητές
- Στην Lua, όλες οι έμμεσα δηλωμένες μεταβλητές είναι καθολικές – οι δε τοπικές μεταβλητές δηλώνονται με το `local` και είναι δυναμικές-στοίβας

Σημασιολογικά Μοντέλα Μεταβίβασης Παραμέτρων

- Λειτουργία Εισόδου (in mode)
- Λειτουργία Εξόδου (out mode)
- Λειτουργία Εισόδου–Εξόδου ή διπλή λειτουργία (inout mode)

Μοντέλα Μεταβίβασης Παραμέτρων



Εννοιολογικά Μοντέλα Μεταφοράς

- Φυσική μετακίνηση (αντιγραφή) μιας τιμής
- Μετακίνηση μιας διαδρομής πρόσβασης προς μια τιμή (δείκτης ή αναφορά)

Μεταβίβαση κατά τιμή (Λειτουργία Εισόδου)

- Η τιμή της πραγματικής παραμέτρου χρησιμοποιείται για την αρχικοποίηση της αντίστοιχης τυπικής παραμέτρου
  - Κανονικά υλοποιείται με αντιγραφές
  - Μπορεί να υλοποιηθεί μεταδίδοντας μια διαδρομή πρόσβασης, αλλά δεν συνίσταται (η επιβολή προστασίας εγγραφής γίνεται δύσκολη)
  - *Μειονεκτήματα* (εάν πρόκειται για φυσική μετακίνηση): απαιτείται επιπλέον αποθηκευτικός χώρος (αποθήκευση δύο φορές) και η πραγματική μετακίνηση μπορεί να είναι ακριβή (για μεγάλες παραμέτρους)
  - *Μειονεκτήματα* (εάν η πρόσβαση γίνεται μέσω διαδρομής πρόσβασης): πρέπει να επιβάλλεται προστασία εγγραφής στο κληθέν υποπρόγραμμα και οι προσπελάσεις κοστίζουν περισσότερο (έμμεση διευθυνσιοδότηση)

Μεταβίβαση κατ' αποτέλεσμα (Λειτουργία εξόδου)

- Όταν μια παράμετρος μεταβιβάζεται κατ' αποτέλεσμα, δεν μεταφέρεται τιμή στο υποπρόγραμμα – η αντίστοιχη τυπική παράμετρος ενεργεί ως τοπική μεταβλητή – η τιμή της μεταδίδεται στην πραγματική παράμετρο του καλούντος όταν ο έλεγχος επιστρέφεται σε αυτόν, με φυσική μετακίνηση
  - Απαιτεί επιπλέον χώρο αποθήκευσης και λειτουργία αντιγραφής
- Πιθανά προβλήματα:
  - `sub(p1, p1);` η τυπική παράμετρος που θα αντιγραφεί πίσω θα αποτελεί την τρέχουσα τιμή του `p1`
  - `sub(list[sub], sub);` ο υπολογισμός της διεύθυνσης του list[sub] θα γίνεται στην αρχή ή στο τέλος του υποπρογράμματος;

Μεταβίβαση κατά Τιμή και Αποτέλεσμα (Λειτουργία Εισόδου–Εξόδου)

- Πρόκειται για συνδυασμό μεταβίβασης με τιμή και μεταβίβασης με αποτέλεσμα
- Μερικές φορές αναφέρεται ως μεταβίβαση με αντιγραφή
- Οι τυπικές παράμετροι αποθηκεύονται σε τοπικό χώρο αποθήκευσης
- Μειονεκτήματα:
  - Τα μειονεκτήματα της μεταβίβασης κατ' αποτέλεσμα
  - Τα μειονεκτήματα της μεταβίβασης κατά τιμή

Μεταβίβαση κατ' αναφορά (Λειτουργία Εισόδου-Εξόδου)

- Μεταβιβάζει μια διαδρομή πρόσβασης
- Ονομάζεται επίσης και μεταβίβαση με διαμοιρασμό
- Πλεονέκτημα: Η διαδικασία μεταβίβασης είναι αποδοτική (δεν συμβαίνουν αντιγραφές και δεν υφίστανται αντίγραφα αποθηκευμένα στη μνήμη)
- Μειονεκτήματα
  - Βραδύτερες προσπελάσεις (σε σύγκριση με τη μεταβίβαση κατά τιμή) στις τυπικές παραμέτρους
  - Πιθανότητα για μη-επιθυμητά side effects (συγκρούσεις)
  - Μη-επιθυμητά ψευδώνυμα (η πρόσβαση διευρύνεται)

```
fun(total, total);  fun(list[i], list[j];  fun(list[i], i);
```

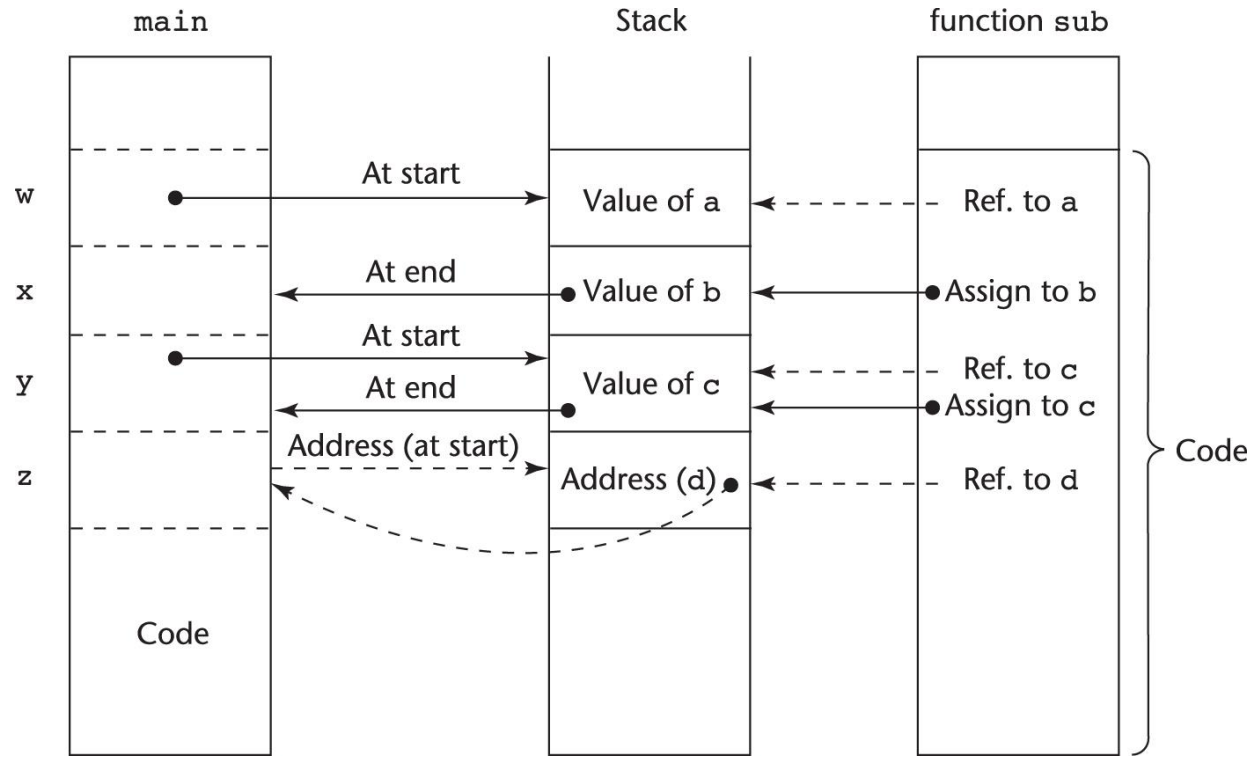
Μεταβίβαση κατ' Όνομα (Λειτουργία Εισόδου-Εξόδου)

- Με αντικατάσταση κειμένου
- Οι τυπικές παράμετροι προσδένονται σε μια μέθοδο πρόσβασης τη στιγμή της κλήσης, αλλά η πραγματική πρόσδεση σε μια τιμή ή σε μια διεύθυνση συμβαίνει κατά τη χρονική στιγμή της αναφοράς ή της ανάθεσης
- Επιτρέπει ευελιξία στην καθυστερημένη πρόσδεση (late binding)
- Η υλοποίηση απαιτεί το περιβάλλον αναφοράς του καλούντος να περάσει με την παράμετρο, έτσι ώστε να μπορεί να υπολογιστεί η διεύθυνση της πραγματικής παραμέτρου

Υλοποίηση Μεθόδων Μεταβίβασης Παραμέτρων

- Στις περισσότερες γλώσσες η επικοινωνία παραμέτρων γίνεται μέσω της στοίβας χρόνου εκτέλεσης
- Η μεταβίβαση κατ' αναφορά είναι η απλούστερη στην υλοποίηση – τοποθετείται μόνο μια διεύθυνση στην στοίβα

Υλοποίηση Μεθόδων Μεταβίβασης Παραμέτρων



Επικεφαλίδα Συνάρτησης: `void sub(int a, int b, int c, int d)`
Κλήση της συνάρτησης στην main: `sub(w, x, y, z)`
(μεταβίβαση `w` κατά τιμή, `x` κατ' αποτέλεσμα,
`y` κατά τιμή-αποτέλεσμα, `z` κατ' αναφορά)

Μέθοδοι Μεταβίβασης Παραμέτρων στις Κύριες Γλώσσες

- C
  - Μεταβίβαση κατά τιμή
  - Η μεταβίβαση κατ' αναφορά επιτυγχάνεται χρησιμοποιώντας δείκτες ως παραμέτρους
- C++
  - Ένας ειδικός τύπος δείκτη, ο τύπος αναφοράς μπορεί να χρησιμοποιηθεί για μεταβίβαση κατ' αναφορά
- Java
  - Όλες οι παράμετροι μεταβιβάζονται κατά τιμή
  - Οι παράμετροι αντικειμένων μεταβιβάζονται κατ' αναφορά

Μέθοδοι Μεταβίβασης Παραμέτρων στις Κύριες Γλώσσες (συνέχεια)

- Fortran 95+
  - Οι παράμετροι μπορούν να δηλωθούν ότι είναι in, out, ή inout
- C#
  - Προκαθορισμένη μέθοδος: μεταβίβαση κατά τιμή
  - Η μεταβίβαση κατ' αναφορά καθορίζεται τοποθετώντας τόσο πριν από την τυπική παράμετρο όσο και πριν από την πραγματική παράμετρο το `ref`
- PHP: παρόμοια με τη C#, με τη διαφορά ότι είτε η πραγματική είτε η τυπική παράμετρος μπορεί να ορίσει το ref
- Perl: όλες οι πραγματικές παράμετροι υπονοούμενα τοποθετούνται σε ένα προκαθορισμένο πίνακα με όνομα `@_`
- Η Python και η Ruby χρησιμοποιούν μεταβίβαση κατ' εκχώρηση (όλες οι τιμές δεδομένων είναι αντικείμενα) – οι πραγματικές παράμετροι εκχωρούνται στις τυπικές παραμέτρους (ουσιαστικά είναι μεταβίβαση κατ' αναφορά)

Έλεγχος Τύπων στις Παραμέτρους

- Θεωρείται πολύ σημαντικό χαρακτηριστικό για αξιοπιστία
- Στην FORTRAN 77 και στην αρχική C: απουσία
- Στην Pascal και στην Java: απαιτείται πάντα
- ANSI C και C++: Η επιλογή γίνεται από τον χρήστη
  - Πρωτότυπα
- Οι σχετικά νέες γλώσσες Perl, JavaScript, και PHP δεν απαιτούν έλεγχο τύπων
- Στην Python και στην Ruby, οι μεταβλητές δεν έχουν τύπους (τα αντικείμενα έχουν), συνεπώς ο έλεγχος τύπων παραμέτρων δεν είναι εφικτός

Πολυδιάστατοι Πίνακες ως Παράμετροι

- Αν ένας πολυδιάστατος πίνακας περάσει σε ένα υποπρόγραμμα και το υποπρόγραμμα μεταγλωττιστεί ξεχωριστά, ο μεταγλωττιστής πρέπει να γνωρίζει το δηλωθέν μέγεθος του πίνακα έτσι ώστε να δημιουργήσει τη συνάρτηση αντιστοίχισης στη μνήμη

Πολυδιάστατοι Πίνακες ως παράμετροι: C and C++

- Ο προγραμματιστής απαιτείται να συμπεριλαμβάνει τα δηλωθέντα μεγέθη όλων πλην του πρώτου δείκτη στην πραγματική παράμετρο
- Δεν επιτρέπει τη συγγραφή ευέλικτων υποπρογραμμάτων
- Λύση: Πέρασμα ενός δείκτη στον πίνακα και των μεγεθών των διαστάσεων ως επιπλέον παραμέτρους – ο χρήστης θα πρέπει να συμπεριλαμβάνει συναρτήσεις αντιστοίχισης στη μνήμη με βάση τις διαστάσεις του πίνακα

Πολυδιάστατοι Πίνακες ως παράμετροι: Java and C#

- Παρόμοια με την Ada
- Οι πίνακες είναι αντικείμενα – είναι όλα μονής-διάστασης, αλλά τα στοιχεία τους μπορούν να είναι πίνακες
- Κάθε πίνακας κληρονομεί μια σταθερά με όνομα `length` στην Java, `Length` στην C# που ορίζεται όταν το αντικείμενο δημιουργείται να είναι ίση με το μήκος του πίνακα

Θέματα Σχεδίασης για τη Μεταβίβαση Παραμέτρων

- Δύο σημαντικά ζητούμενα
  - Αποδοτικότητα
  - Μεταφορά μιας κατεύθυνσης ή δύο κατευθύνσεων
- Αλλά τα παραπάνω ζητούμενα είναι αντικρουόμενα
  - Μια καλή αρχή προγραμματισμού είναι να περιορίζεται η πρόσβαση σε μεταβλητές, που υπονοεί ότι η μεταβίβαση τιμών θα πρέπει να είναι μιας κατεύθυνσης όπου αυτό είναι εφικτό
  - Αλλά η μεταβίβαση κατ' αναφορά είναι αποδοτικότερη στο πέρασμα μεγάλων δομών

Παράμετροι που είναι ονόματα υποπρογραμμάτων

- Μερικές φορές μπορεί να είναι βολικό να περάσουν ονόματα υποπρογραμμάτων ως παράμετροι
- Θέματα:
  1. Ελέγχονται οι τύποι των παραμέτρων;
  2. Ποιο είναι το σωστό περιβάλλον αναφοράς για ένα υποπρόγραμμα το οποίο στέλνεται ως παράμετρος;

Παράμετροι που είναι ονόματα υποπρογραμμάτων: Περιβάλλον αναφοράς

- Αφορά μόνο τις γλώσσες που υποστηρίζουν ένθετα υποπρογράμματα.
- Επιλογές:
  - *Ρηχή πρόσδεση*: Το περιβάλλον της εντολής κλήσης του υποπρογράμματος που έχει μεταβιβαστεί
    – Φυσική επιλογή για γλώσσες δυναμικής εμβέλειας
  - *Βαθιά πρόσδεση*: Το περιβάλλον του ορισμού του υποπρογράμματος που έχει μεταβιβαστεί
    – Φυσική επιλογή για γλώσσες στατικής εμβέλειας
  - *Ad hoc πρόσδεση*: Το περιβάλλον της εντολής κλήσης που μεταβίβασε το υποπρόγραμμα

Έμμεση Κλήση Υποπρογραμμάτων

- Συνήθως όταν υπάρχουν πολλά πιθανά υποπρογράμματα που μπορούν να κληθούν και αυτό που πρέπει να κληθεί σε μια συγκεκριμένη εκτέλεση του προγράμματος δεν είναι γνωστό παρά μόνο στο χρόνο εκτέλεσης (π.χ., χειρισμός γεγονότων και GUIs)
- Στην C και στην C++, οι κλήσεις αυτές γίνονται μέσω δεικτών σε συναρτήσεις (function pointers)

Έμμεση Κλήση Υποπρογραμμάτων (συνέχεια)

- Στη C#, οι δείκτες σε μεθόδους υλοποιούνται με αντικείμενα που ονομάζονται αντιπρόσωποι *(delegates)*
  - Η δήλωση ενός delegate:

```
public delegate int Change(int x);
```

  – Ο τύπος delegate, με όνομα `Change`, μπορεί να αρχικοποιηθεί με οποιαδήποτε μέθοδο που δέχεται μια παράμετρο **`int`** και επιστρέφει μια τιμή **`int`**

  Η μέθοδος: `static int fun1(int x) { … }`

  Αρχικοποίηση: `Change chgfun1 = new Change(fun1);`

  Μπορεί να κληθεί ως: `chgfun1(12);`

  – Ένας αντιπρόσωπος μπορεί να αποθηκεύσει περισσότερες από μια διευθύνσεις, οπότε ονομάζεται *αντιπρόσωπος πολυδιανομής* (*multicast delegate)*

Θέματα Σχεδίασης Συναρτήσεων

- Επιτρέπονται τα side effects;
  - Οι παράμετροι θα πρέπει πάντα να είναι σε λειτουργία εισόδου έτσι ώστε να μειώνονται τα side effects (όπως στην Ada)
- Ποιοι τύποι τιμών επιστροφής επιτρέπονται;
  - Οι περισσότερες προστακτικές γλώσσες περιορίζουν τους τύπους επιστροφής
  - Η C επιτρέπει οποιοδήποτε τύπο εκτός από πίνακες και συναρτήσεις
  - Η C++ όπως και η C, αλλά επιπλέον επιτρέπει τύπους ορισμένους από το χρήστη
  - Οι μέθοδοι της Java και της C# μπορούν να επιστρέφουν οποιοδήποτε τύπο (αλλά καθώς οι μέθοδοι δεν είναι τύποι, δεν μπορούν να επιστρέφονται)
  - Η Python και η Ruby αντιμετωπίζουν τις μεθόδους ως αντικείμενα πρώτης τάξης, συνεπώς μπορούν να επιστρέφονται, όπως οποιαδήποτε άλλη κλάση
  - Η Lua επιτρέπει σε συναρτήσεις να επιστρέφουν πολλαπλές τιμές

Υπερφορτωμένα Υποπρογράμματα

- Ένα υπερφορτωμένο υποπρόγραμμα είναι ένα υποπρόγραμμα που έχει το ίδιο όνομα με κάποιο άλλο υποπρόγραμμα στο ίδιο περιβάλλον αναφοράς
  - Κάθε έκδοση ενός υπερφορτωμένου υποπρογράμματος έχει ένα μοναδικό πρωτόκολλο.
- Οι C++, Java, C#, και Ada περιέχουν προκαθορισμένα υπερφορτωμένα υποπρογράμματα
- Στην Ada, ο τύπος επιστροφής μιας υπερφορτωμένης συνάρτησης μπορεί να χρησιμοποιηθεί για να προσδιορίσει τι συνάρτηση που καλείται (έτσι, δύο υπερφορτωμένες συναρτήσεις μπορούν να έχουν τις ίδιες παραμέτρους)
- Οι Ada, Java, C++, και C# επιτρέπουν στους χρήστες να γράψουν πολλαπλές εκδόσεις υποπρογραμμάτων με το ίδιο όνομα

Γενικά Υποπρογράμματα

- Ένα *γενικό ή πολυμορφικό υποπρόγραμμα* λαμβάνει παραμέτρους διαφορετικών τύπων σε διαφορετικές ενεργοποιήσεις
- Τα υπερφορτωμένα υποπρογράμματα παρέχουν *ad hoc πολυμορφισμό*
- *Πολυμορφισμός υποτύπων* σημαίνει ότι μια μεταβλητή τύπου T μπορεί να προσπελάσει οποιοδήποτε αντικείμενο τύπου T ή οποιοδήποτε τύπο που παράγεται από τον T (στις Αντικειμενοστραφείς Γλώσσες Προγραμματισμού)
- Ένα υποπρόγραμμα που λαμβάνει μια γενική παράμετρο που χρησιμοποιείται σε εκφράσεις τύπων που περιγράφει τον τύπο των παραμέτρων του υποπρογράμματος παρέχει *παραμετρικό πολυμορφισμό*
  – Πρόκειται για υποκατάστατο της δυναμικής πρόσδεσης που συμβαίνει κατά το χρόνο μεταγλώττισης

Γενικά Υποπρογράμματα (συνέχεια)

- C++
  - Δημιουργούνται έμμεσα εκδόσεις των γενικών υποπρογραμμάτων, όταν το υποπρόγραμμα κατονομάζεται σε μια κλήση ή όταν λαμβάνεται η διεύθυνσή του με τον τελεστή &
  - Τα γενικά υποπρογράμματα περιέχουν στην αρχή τους μια πρόταση **template** με μια λίστα γενικών μεταβλητών, που μπορούν να είναι ονόματα τύπων ή ονόματα κλάσεων

```
template <class Type>
Type max(Type first, Type second) {
  return first > second ? first : second;
}
```

Γενικά Υποπρογράμματα (συνέχεια)

- Java 5.0
  - Διαφορές γενικών υποπρογραμμάτων στη Java 5.0 και στη C++:
    1. Οι γενικές παράμετροι στην Java 5.0 πρέπει να είναι κλάσεις
    2. Οι γενικές μέθοδοι στην Java 5.0 αρχικοποιούνται μια φορά ως πραγματικές γενικές μέθοδοι
    3. Στη Java μπορούν να καθοριστούν περιορισμοί στο εύρος των κλάσεων που μπορούν να περάσουν ως γενικές παράμετροι στις γενικές μεθόδους
    4. Η Java μπορεί να χρησιμοποιεί τύπους μπαλαντέρ

Γενικά Υποπρογράμματα (συνέχεια)

- Java 5.0 (συνέχεια)

```
public static <T> T doIt(T[] list) { … }
```

  - Η παράμετρος είναι ένας πίνακας με γενικά στοιχεία (`T` είναι το όνομα του τύπου)
  - Η κλήση:

```
doIt<String>(myList);
```

Οι γενικές παράμετροι μπορούν να έχουν όρια:

```
public static <T extends Comparable> T
  doIt(T[] list) { … }
```

Ο γενικός τύπος θα πρέπει να είναι μιας κλάσης που υλοποιεί τη διεπαφή `Comparable`

Γενικά Υποπρογράμματα (συνέχεια)

- Java 5.0 (συνέχεια)
  - Τύποι μπαλαντέρ

    `Collection<?>` είναι ένας τύπος μπαλαντέρ για τις κλάσεις συλλογών

```
void printCollection(Collection<?> c) {
   for (Object e: c) {
     System.out.println(e);
   }
 }
```

  - Λειτουργεί με οποιαδήποτε κλάση συλλογής

Γενικά Υποπρογράμματα (συνέχεια)

- C# 2005
  - Υποστηρίζει γενικές μεθόδους παρόμοιες με εκείνες της Java 5.0
  - Μια διαφορά: Ο τύπος των πραγματικών παραμέτρων στην κλήση μπορεί να παραληφθεί αν ο μεταγλωττιστής μπορεί να συμπεράνει τους μη καθορισμένους τύπους
    - Άλλη μια διαφορά – Η C# 2005 δεν υποστηρίζει τύπους μπαλαντέρ

Γενικά Υποπρογράμματα (συνέχεια)

- F#
  - Ο τύπος συμπεραίνεται ότι είναι γενικός αν δεν μπορεί να καθοριστεί ο τύπος μιας παραμέτρου ή ο τύπος επιστροφής μιας συνάρτησης – *αυτόματη γενίκευση*
  - Η τύποι αυτής της μορφής υποδηλώνονται με μια απόστροφο και ένα απλό γράμμα, π.χ., `'a`
  - Οι συναρτήσεις μπορούν να οριστούν ότι έχουν γενικές παραμέτρους

```
let printPair (x: 'a) (y: 'a) =
    printfn "%A %A" x y
```

    - `%A` είναι ο κωδικός μορφοποίησης για οποιοδήποτε τύπο
    - Οι παράμετροι δεν είναι περιορισμένοι με βάση τον τύπο

Γενικά Υποπρογράμματα (συνέχεια)

- F# (συνέχεια)
  - Αν οι παράμετροι μιας συνάρτησης χρησιμοποιούνται με αριθμητικούς τελεστές, τότε είναι περιορισμένοι σε σχέση με τον τύπο, ακόμα και αν οι παράμετροι έχουν οριστεί ως γενικοί
  - Λόγω της εξαγωγής τύπων και της απουσίας μετατροπών τύπων, οι γενικές συναρτήσεις της F# είναι λιγότερο χρήσιμες από εκείνες των C++, Java 5.0+, και C# 2005+

Υπερφορτωμένοι τελεστές ορισμένοι από τον χρήστη

- Οι τελεστές μπορούν να υπερφορτωθούν στην Ada, C++, Python, και στην Ruby
- Παράδειγμα σε Python

```
def __add__ (self, second) :
  return Complex(self.real + second.real,
                 self.imag + second.imag)
```

Χρήση: για τον υπολογισμό `x + y, x.__add__(y)`

Κλειστότητες

- Μια κλειστότητα (*closure)* είναι ένα υποπρόγραμμα και το περιβάλλον αναφοράς στο οποίο ορίστηκε
  - Το περιβάλλον αναφοράς χρειάζεται αν το υποπρόγραμμα μπορεί να κληθεί από μια οποιαδήποτε θέση του προγράμματος
  - Μια γλώσσα με στατική εμβέλεια που δεν επιτρέπει εμφωλιασμένα υποπρογράμματα δεν χρειάζεται κλειστότητες
  - Οι κλειστότητες χρειάζονται μόνο αν ένα υποπρόγραμμα μπορεί να προσπελάσει μεταβλητές στις εμβέλειες στις οποίες είναι εμφωλευμένο και το υποπρόγραμμα μπορεί να κληθεί από οποιοδήποτε σημείο
  - Για να υποστηρίξει κλειστότητες, μια υλοποίηση μπορεί να χρειάζεται να παρέχει απεριόριστη έκταση σε ορισμένες μεταβλητές (διότι ένα υποπρόγραμμα μπορεί να προσπελαύνει μη τοπικές μεταβλητές που δεν είναι πλέον ζωντανές)

Κλειστότητες (συνέχεια)

- Μια κλειστότητα στην JavaScript:

```
function makeAdder(x) {
  return function(y) {return x + y;}
}
...
var add10 = makeAdder(10);
var add5 = makeAdder(5);
document.write("add 10 to 20: " + add10(20) +
               "<br />");
document.write("add 5 to 20: " + add5(20) +
               "<br />");
```

  - Η κλειστότητα είναι η ανώνυμη συνάρτηση που επιστρέφεται από την `makeAdder`

Κλειστότητες (συνέχεια)

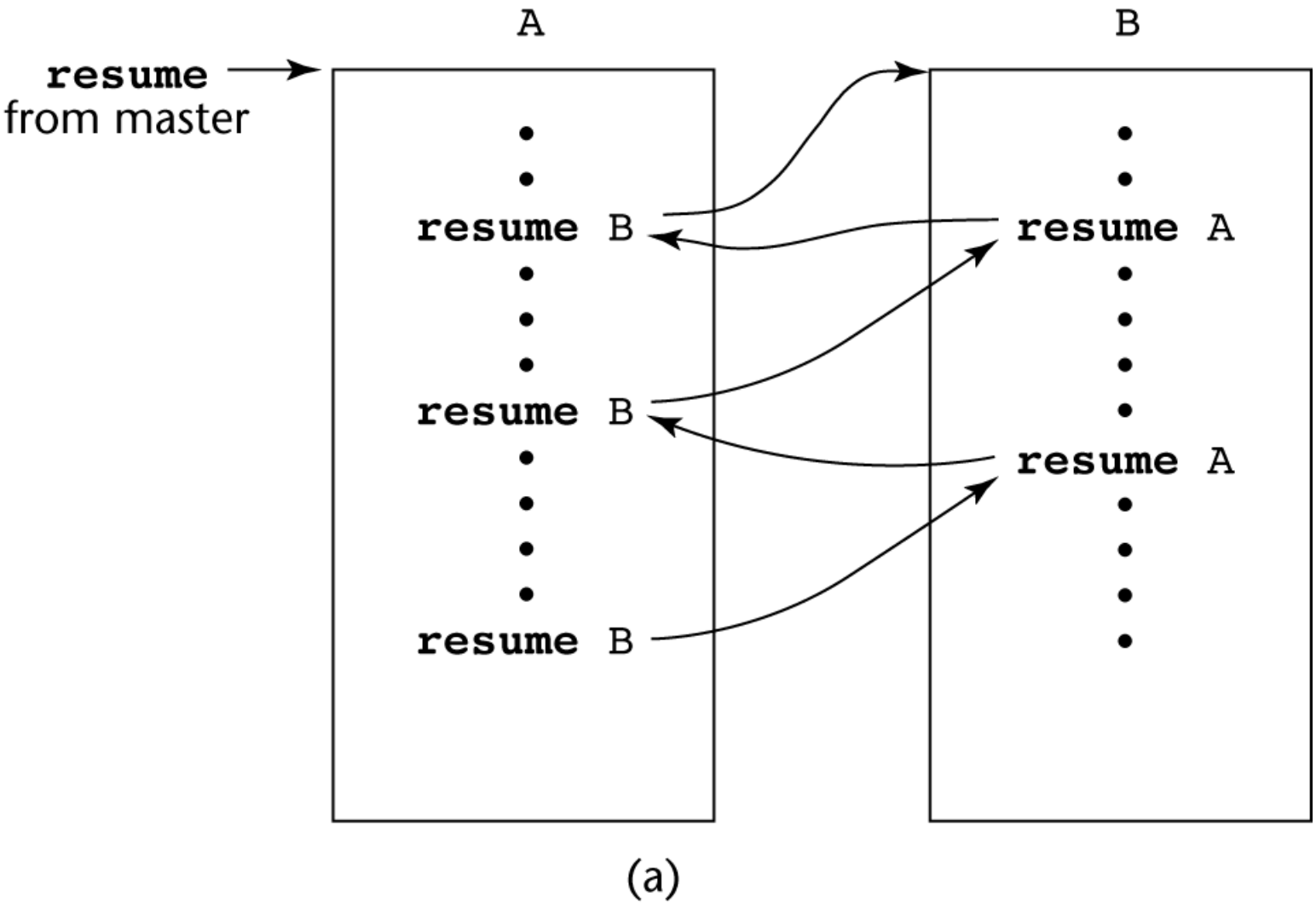
- C#
  - Μπορούμε να γράψουμε την ίδια κλειστότητα στην C# χρησιμοποιώντας ένα εμφωλευμένο ανώνυμο αντιπρόσωπο (delegate)
  - `Func<int, int>` (ο τύπος επιστροφής) καθορίζει έναν αντιπρόσωπο που παίρνει έναν `int` ως παράμετρο και επιστρέφει ένα `int`

```
static Func<int, int> makeAdder(int x) {
   return delegate(int y) {return x + y;};
}
...
Func<int, int> Add10 = makeAdder(10);
Func<int, int> Add5 = makeAdder(5);
Console.WriteLine("Add 10 to 20: {0}", Add10(20));
Console.WriteLine("Add 5 to 20: {0}", Add5(20));
```

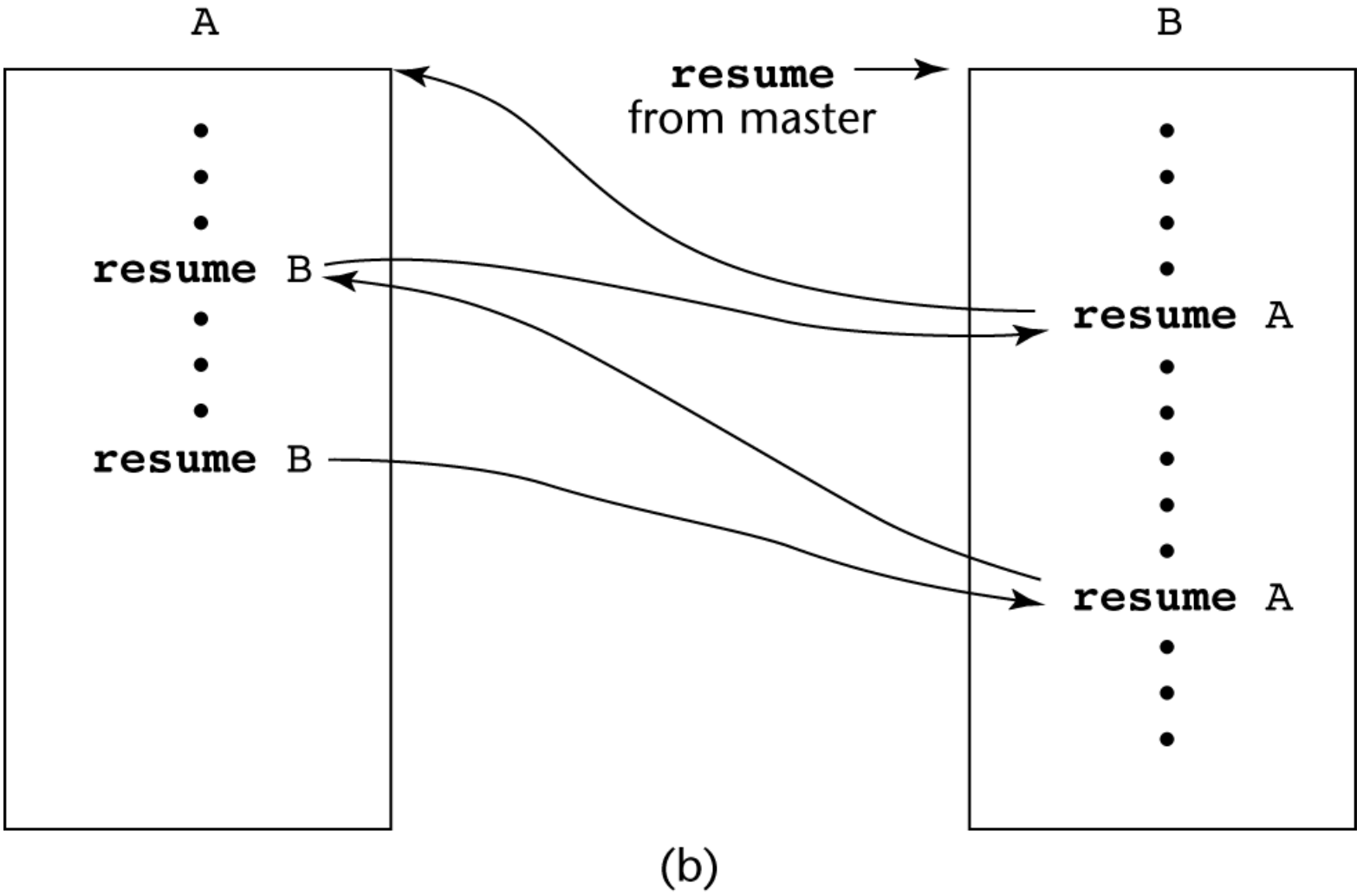
Συρρουτίνες

- Μια συρρουτίνα (*coroutine)* είναι ένα υποπρόγραμμα που έχει πολλά σημεία εισόδου τα οποία ελέγχονται από την ίδια τη συρρουτίνα – υποστηρίζεται απευθείας στην Lua
- Επίσης καλείται και συμμετρικός έλεγχος *(symmetric control):* οι καλούσες και οι καλούμενες συρρουτίνες λειτουργούν σε ισότιμη βάση
- Η κλήση μιας συρρουτίνας ονομάζεται συνέχιση (*resume)*
- Η πρώτη συνέχιση μιας συρρουτίνας γίνεται στην αρχή της, αλλά οι επόμενες κλήσεις εισέρχονται στο σημείο αμέσως μετά την τελευταία εντολή που εκτελέστηκε στην συρρουτίνα
- Οι συρρουτίνες επαναληπτικά συνεχίζουν η μια την άλλη, πιθανά για πάντα
- Οι συρρουτίνες παρέχουν *ημι-ταυτόχρονη* εκτέλεση των προγραμματιστικών μονάδων (των συρρουτινών) – η εκτέλεσή τους είναι παρεμβαλόμενη στο χρόνο (interleaved), αλλά όχι επικαλυπτόμενη στο χρόνο

Συρρουτίνες: Πιθανές αλληλουχίες ελέγχου εκτέλεσης

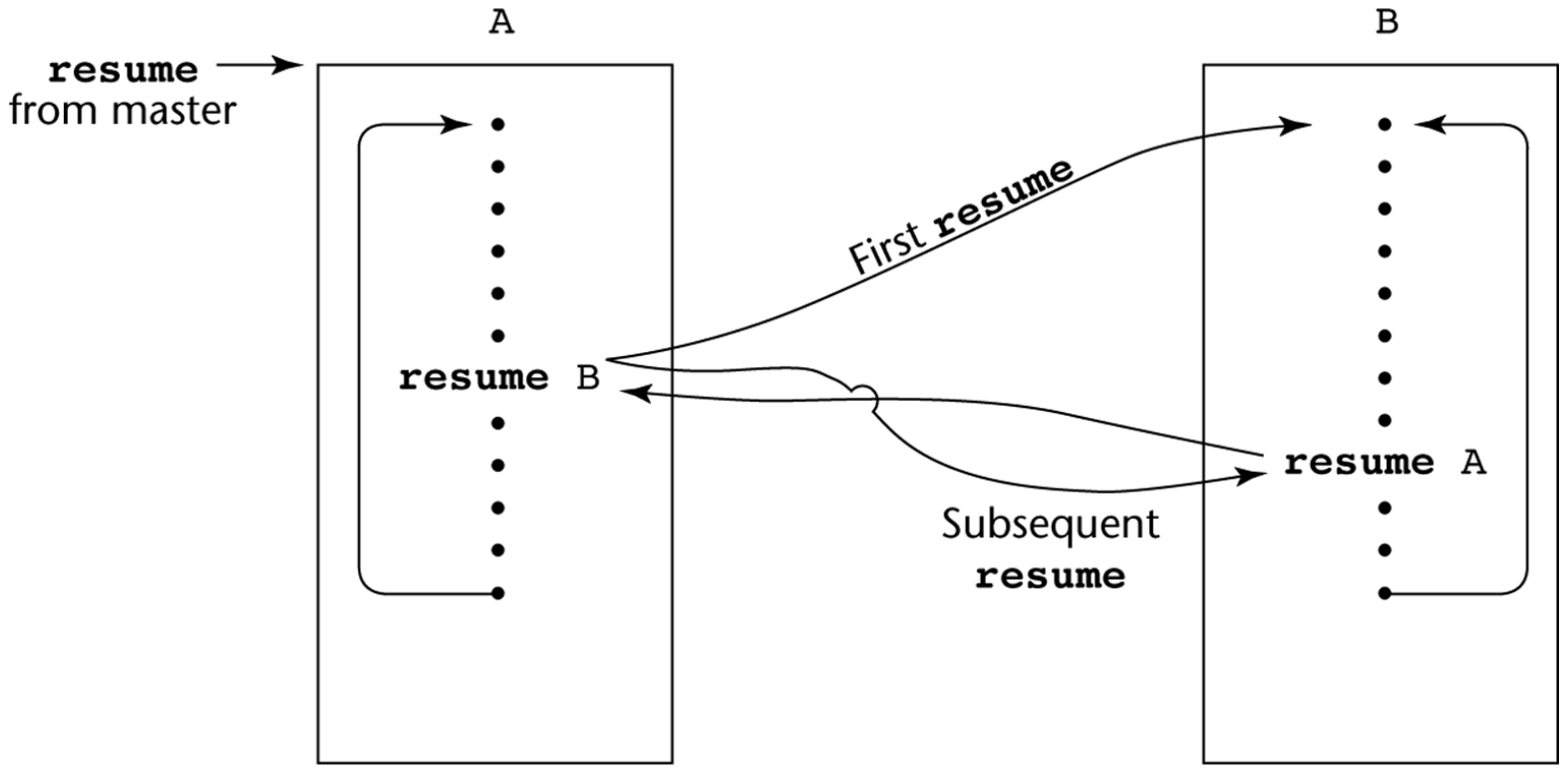


Συρρουτίνες: Μια πιθανή αλληλουχία εκτέλεσης



(b)

Συρρουτίνες: Αλληλουχία εκτέλεσης συρρουτινών με βρόχους



Σύνοψη

- Ο ορισμός ενός υποπρογράμματος περιγράφει τις ενέργειες που αναπαρίστανται από το υποπρόγραμμα
- Τα υποπρογράμματα μπορούν να είναι είτε συναρτήσεις είτε διαδικασίες
- Οι τοπικές μεταβλητές σε υποπρογράμματα μπορεί να είναι δυναμικές–στοίβας ή στατικές
- Τρία μοντέλα περάσματος παραμέτρων: λειτουργία εισόδου, λειτουργία εξόδου, και διπλή λειτουργία
- Μερικές γλώσσες επιτρέπουν την υπερφόρτωση τελεστών
- Τα υποπρογράμματα μπορούν να γίνουν γενερικά
- Μια κλειστότητα είναι ένα υποπρόγραμμα και το περιβάλλον αναφοράς του
- Μια συρρουτίνα είναι ένα ειδικό υποπρόγραμμα με πολλαπλές εισόδους