ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΙΚΑ
ΜΑΘΗΜΑΤΑ

Σχεδίαση Γλωσσών & Μεταγλωττιστές

Ενότητα 2: Εμβέλεια

Επ. Καθ. Π. Κατσαρός
Τμήμα Πληροφορικής

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδεια χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο«Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Συναρτήσεις και υποπρογράμματα

Η σύνταξη υποπρογραμμάτων στην Algol και στη C:

procedure P (< pars >)	< type > f(< pars >)
begin	{
< local vars >	< local vars >
< procedure_body >	< procedure_body >
end;	}

Οι εγγραφές ενεργοποιήσης πρέπει να περιλαμβάνουν χώρο για:

- παραμέτρους

Συναρτήσεις και υποπρογράμματα

Η σύνταξη υποπρογραμμάτων στην Algol και στη C:

```
procedure P ( < pars > )
begin
        < local vars >
        < procedure_body >
end;
```

```
< type > f( < pars > )
{
        < local vars >
        < procedure_body >
}
```

Οι εγγραφές ενεργοποιήσης πρέπει να περιλαμβάνουν χώρο για:

- παραμέτρους
- διεύθυνση επιστροφής μετά την εκτέλεση

Συναρτήσεις και υποπρογράμματα

Η σύνταξη υποπρογραμμάτων στην Algol και στη C:

procedure P (< pars >)	< type > f(< pars >)
begin	{
< local vars >	< local vars >
< procedure_body >	< procedure_body >
end;	}

Οι εγγραφές ενεργοποιήσης πρέπει να περιλαμβάνουν χώρο για:

- παραμέτρους
- διεύθυνση επιστροφής μετά την εκτέλεση
- τοπικές μεταβλητές & ενδιάμεσα αποτελέσματα

Συναρτήσεις και υποπρογράμματα

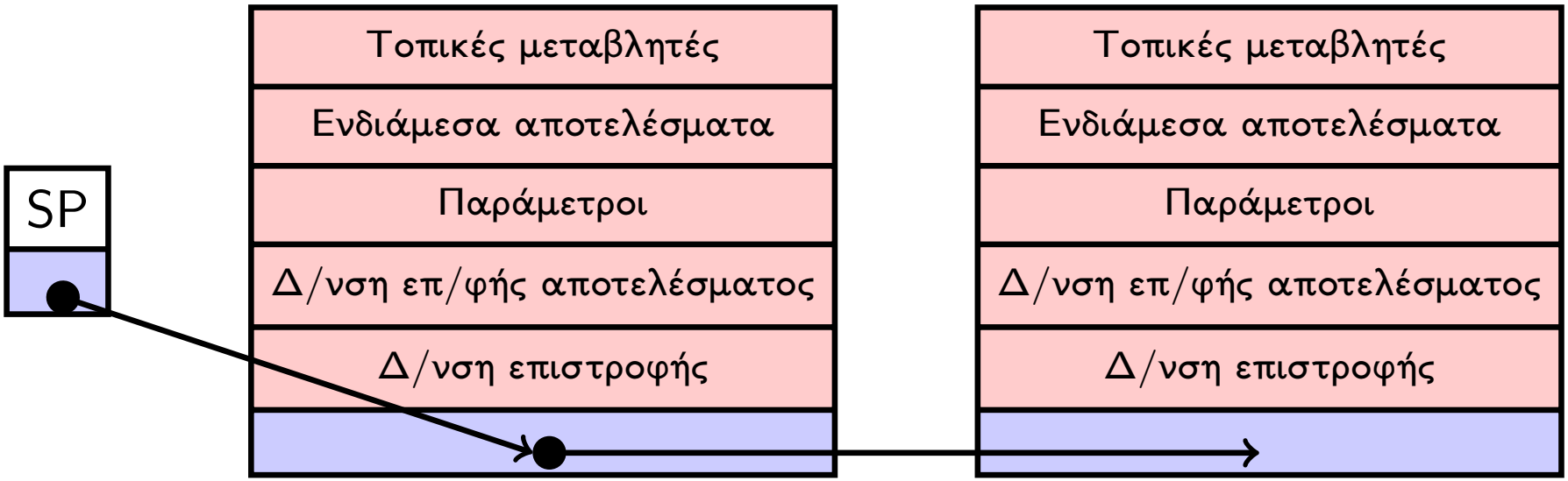
Η σύνταξη υποπρογραμμάτων στην Algol και στη C:

procedure P (< pars >)	< type > f(< pars >)
begin	{
< local vars >	< local vars >
< procedure_body >	< procedure_body >
end;	}

Οι εγγραφές ενεργοποιήσης πρέπει να περιλαμβάνουν χώρο για:

- παραμέτρους
- διεύθυνση επιστροφής μετά την εκτέλεση
- τοπικές μεταβλητές & ενδιάμεσα αποτελέσματα
- επιστρεφόμενη τιμή: θέση εναπόθεσης επιστρεφόμενης τιμής κατά την έξοδο

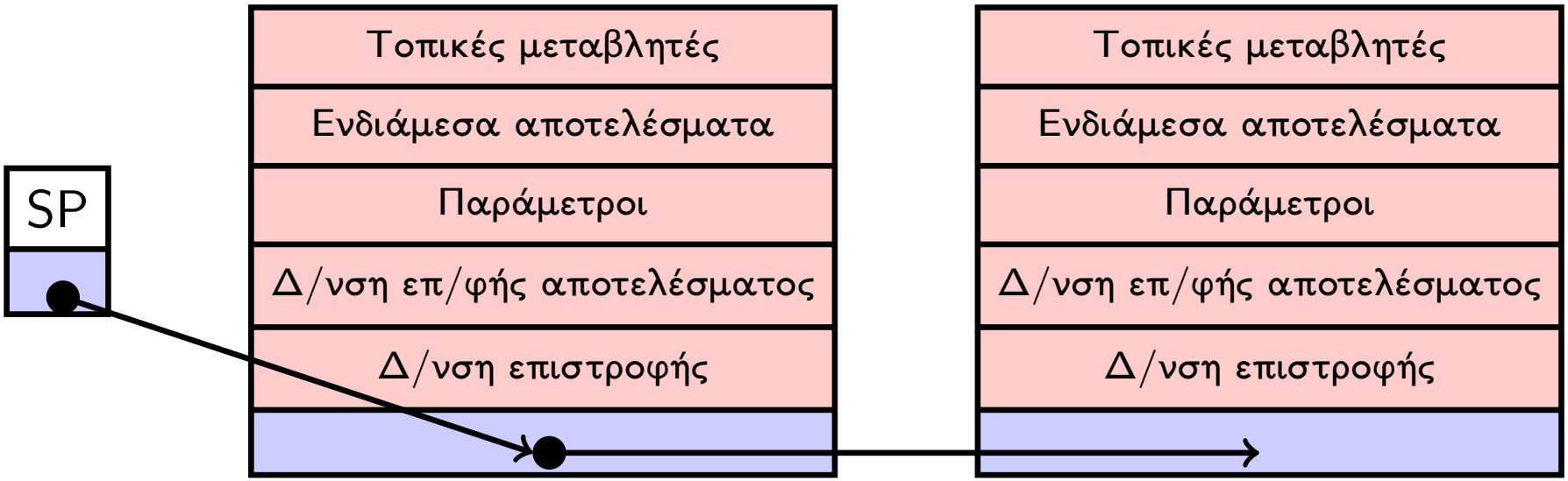
Εγγραφές ενεργοποιήσης για συναρτήσεις & υποπρογράμματα



Παράμετροι

- Θέσεις μνήμης με δεδομένα από το block που καλεί.

Εγγραφές ενεργοποιήσης για συναρτήσεις & υποπρογράμματα



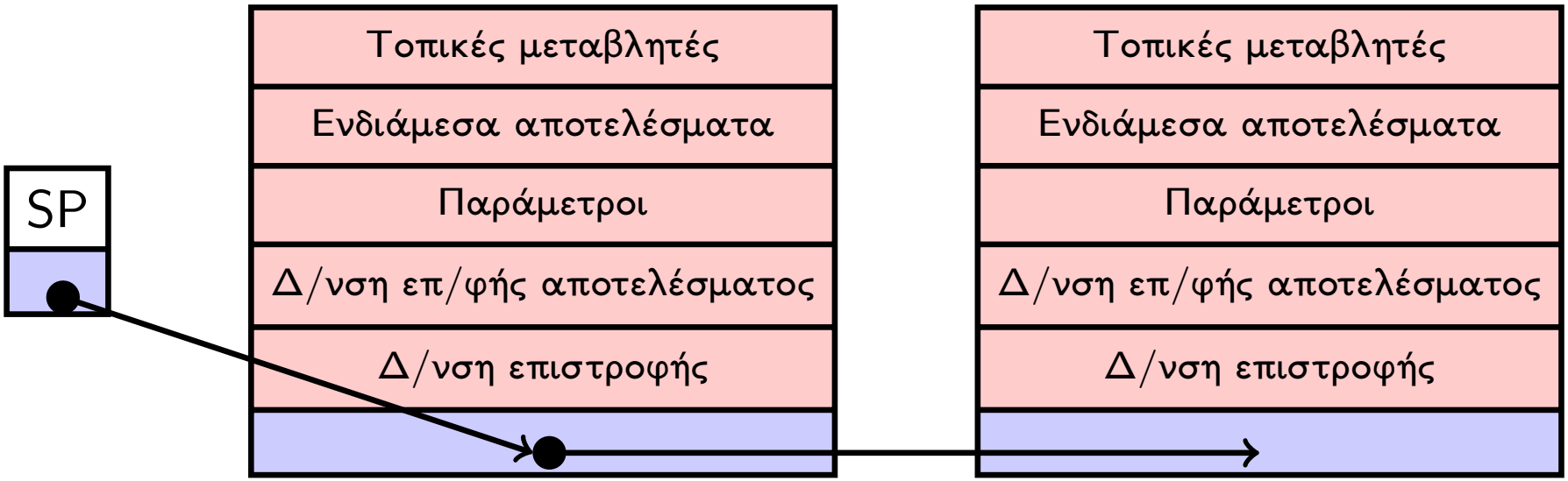
Παράμετροι

- Θέσεις μνήμης με δεδομένα από το block που καλεί.

Δ/νση επιστροφής αποτελέσματος

- Δ/νση του block που καλεί για να υποδεχθεί το αποτέλεσμα.

Εγγραφές ενεργοποιήσης για συναρτήσεις & υποπρογράμματα



Παράμετροι

- Θέσεις μνήμης με δεδομένα από το block που καλεί.

Δ/νση επιστροφής αποτελέσματος

- Δ/νση του block που καλεί για να υποδεχθεί το αποτέλεσμα.

Δ/νση επιστροφής

- Θέση μνήμης του κώδικα που πρόκειται να εκτελεστεί με την επιστροφή.

Πέρασμα παραμέτρων

Γενική ορολογία

Θεωρήστε την εντολή εκχώρησης: `y:=x+3`

- Το όνομα αριστερά αναφέρεται σε θέση μνήμης και ονομάζεται L-value.
- Το όνομα δεξιά αναφέρεται σε περιεχόμενα και ονομάζεται R-value.

Πέρασμα παραμέτρων

Γενική ορολογία

Θεωρήστε την εντολή εκχώρησης: `y:=x+3`

- Το όνομα αριστερά αναφέρεται σε θέση μνήμης και ονομάζεται L-value.
- Το όνομα δεξιά αναφέρεται σε περιεχόμενα και ονομάζεται R-value.

Πέρασμα παραμέτρου με αναφορά

- Τοποθέτηση L-value (διεύθυνση) στην εγγραφή δραστηριοποιήσης.
- Η συνάρτηση μπορεί να εκχωρήσει στη μεταβλητή που περνάει ως παράμετρος.

Πέρασμα παραμέτρων

Γενική ορολογία

Θεωρήστε την εντολή εκχώρησης: `y:=x+3`

- Το όνομα αριστερά αναφέρεται σε θέση μνήμης και ονομάζεται L-value.
- Το όνομα δεξιά αναφέρεται σε περιεχόμενα και ονομάζεται R-value.

Πέρασμα παραμέτρου με αναφορά

- Τοποθέτηση L-value (διεύθυνση) στην εγγραφή δραστηριοποιήσης.
- Η συνάρτηση μπορεί να εκχωρήσει στη μεταβλητή που περνάει ως παράμετρος.

Πέρασμα παραμέτρου με τιμή

- Τοποθέτηση R-value (περιεχόμενο) στην εγγραφή δραστηριοποιήσης.
- Η συνάρτηση δε μπορεί να επηρεάσει την τιμή της μεταβλητής του καλούντος.
- Περιορίζει την ψευδωνυμία (aliasing - δύο η περισσότερα ονόματα που αναφέρονται στην ίδια δ/νση)

Πέρασμα παραμέτρων

Θεωρήστε το παρακάτω πρόγραμμα σε δύο ΓΠ, όπου y μία ακέραια μεταβλητή:

```
function f (x)
{ x := x+1;
  return x };
  . . . .f(y) . . . ;
```

```
function f (x)
{ x := x+1;
  return x };
  . . . .f(y) . . . ;
```

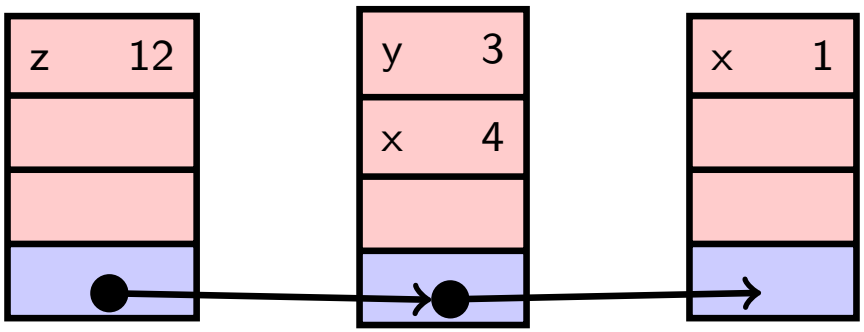
Πέρασμα με αναφορά (σημασία)
Περνά στην f μία αναφορά σε ακέραιο, δηλ. το L-value της y.
Στο σώμα της f επηρεάζεται απευθείας η τιμή της y (actual parameter).
Με την επιστροφή της η f έχει αλλάξει την τιμή της y.

Πέρασμα με τιμή (σημασία)
Περνά στην f μία ακέραια τιμή (actual parameter), το R-value της y.
Στο σώμα της f εκχωρείται μία νέα θέση μνήμης για ακέραιο και αρχικοποιείται με το R-value της y.
Η εκτέλεση της f δεν επηρεάζει την τιμή της y.

Προσπέλαση σε καθολικές μεταβλητές: στατική εμβέλεια

Ποιο x χρησιμοποιείται στην έκφραση x+z ;

```
1  int x = 1;
2  function g(z) = x + z;
3  function f(y) {
4          int x = y + 1;
5          return g(y * x)
6          };
7  f(3);
```

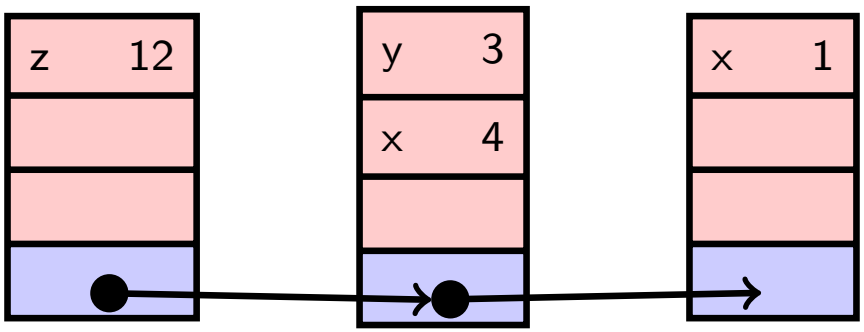


- Στο πλησιέστερο από τα block που στο κείμενο περικλείουν την έκφραση και ορίζουν τη x.

Προσπέλαση σε καθολικές μεταβλητές: στατική εμβέλεια

Ποιο x χρησιμοποιείται στην έκφραση x+z ;

```
1  int x = 1;
2  function g(z) = x + z;
3  function f(y) {
4          int x = y + 1;
5          return g(y * x)
6          };
7  f(3);
```

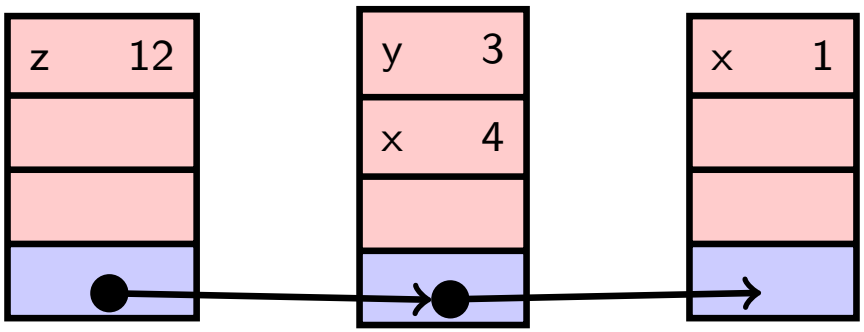


- Στο πλησιέστερο από τα block που στο κείμενο περικλείουν την έκφραση και ορίζουν τη x.
- Στην περίπτωση αυτή έχουμε x=1

Προσπέλαση σε καθολικές μεταβλητές: δυναμική εμβέλεια

Ποιο x χρησιμοποιείται στην έκφραση x+z ;

```
1  int x = 1;
2  function g(z) = x + z;
3  function f(y) {
4          int x = y + 1;
5          return g(y * x)
6          };
7  f(3);
```

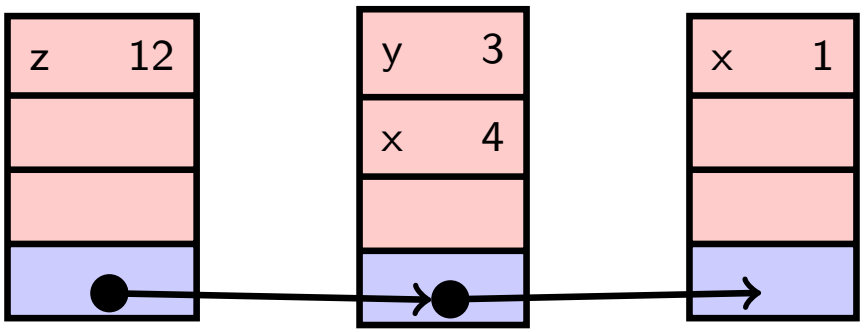


- Το x που βρίσκεται στην πιο πρόσφατα δημιουργηθείσα εγγραφή δραστηριοποιήσης.

Προσπέλαση σε καθολικές μεταβλητές: δυναμική εμβέλεια

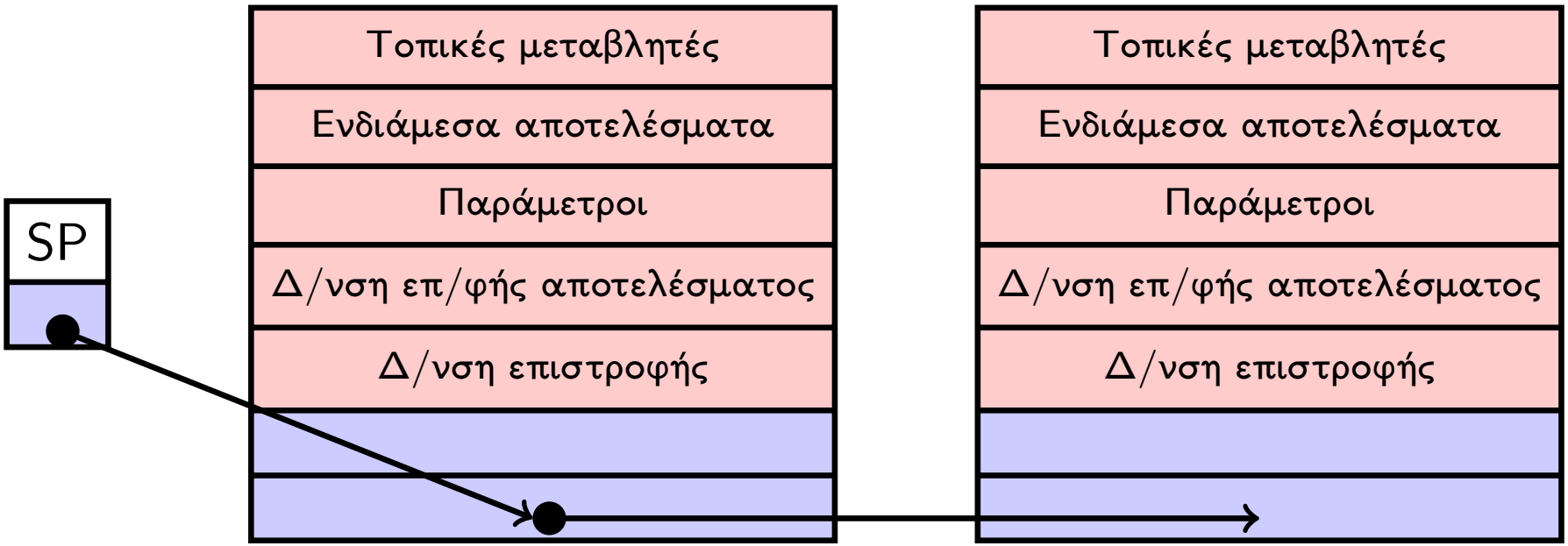
Ποιο x χρησιμοποιείται στην έκφραση x+z ;

```
1  int x = 1;
2  function g(z) = x + z;
3  function f(y) {
4          int x = y + 1;
5          return g(y * x)
6          };
7  f(3);
```



- Το x που βρίσκεται στην πιο πρόσφατα δημιουργηθείσα εγγραφή δραστηριοποιήσης.
- Στην περίπτωση αυτή έχουμε x=4

Εγγραφές ενεργοποιήσης για στατική εμβέλεια



- Υπάρχει ένας επιπλέον σύνδεσμος για τη στατική εμβέλεια.
- Παραπέμπει στην εγγραφή δραστηριοποιήσης του πλησιέστερου block που περικλείει το τρέχον στο κείμενο του προγράμματος.
- Ο ένας σύνδεσμος εξαρτάται από τη δυναμική συμπεριφορά του προγράμματος.
- Ο σύνδεσμος στατικής εμβέλειας εξαρτάται από το κείμενο του προγράμματος.

Κανόνες εμβέλειας στις ΓΠ

- Η **στατική εμβέλεια μιας μεταβλητής** είναι το πλησιέστερο block κώδικα, που περιλαμβάνει ορισμό αυτής από αυτά που περιλκείουν το τρέχον.
- Οι περισσότερες σύγχρονες ΓΠ όπως οι Pascal, C++ και Java υιοθετούν τη στατική εμβέλεια για τις μεταβλητές.

Κανόνες εμβέλειας στις ΓΠ

- Η **στατική εμβέλεια μιας μεταβλητής** είναι το πλησιέστερο block κώδικα, που περιλαμβάνει ορισμό αυτής από αυτά που περιλκείουν το τρέχον.
- Οι περισσότερες σύγχρονες ΓΠ όπως οι Pascal, C++ και Java υιοθετούν τη στατική εμβέλεια για τις μεταβλητές.

- Η **στατική εμβέλεια του ονόματος ενός υποπρογράμματος** καθορίζει όλα τα υποπρογράμματα/συναρτήσεις που μπορούν να το καλέσουν.

Κανόνες εμβέλειας στις ΓΠ

- Η **στατική εμβέλεια μιας μεταβλητής** είναι το πλησιέστερο block κώδικα, που περιλαμβάνει ορισμό αυτής από αυτά που περιλκείουν το τρέχον.
- Οι περισσότερες σύγχρονες ΓΠ όπως οι Pascal, C++ και Java υιοθετούν τη στατική εμβέλεια για τις μεταβλητές.

- Η **στατική εμβέλεια του ονόματος ενός υποπρογράμματος** καθορίζει όλα τα υποπρογράμματα/συναρτήσεις που μπορούν να το καλέσουν.
- Ονόματα υποπρογραμμάτων στην Pascal: η στατική εμβέλεια του ονόματος p() ενός υποπρογράμματος περιλαμβάνει:
  1. Το ίδιο το υποπρόγραμμα p().
  2. Όλα τα υποπρογράμματα που ορίζονται μέσα στο p().
  3. Το υποπρόγραμμα μέσα στο οποίο ορίζεται το p(), έστω f().
  4. Όλα τα υποπρογράμματα που ορίζονται μέσα στο f() μετά από το p(), αλλά και όλα όσα ορίζονται μέσα σ' αυτά άμεσα ή μέσω άλλων υποπρογραμμάτων.

Κανόνες εμβέλειας στις ΓΠ

- Η **δυναμική εμβέλεια μιας μεταβλητής** επεκτείνεται σε όλα τα υποπρογράμματα που κατά τη διάρκεια μιας εκτέλεσης έχουν κληθεί μετά από αυτό στο οποίο βρίσκεται, μέχρι όμως το υποπρόγραμμα το οποίο ορίζει ξανά την ίδια μεταβλητή.
- Δε μπορεί να καθοριστεί μελετώντας τη δομή του προγράμματος.
- Δυναμική εμβέλεια χρησιμοποιούσαν κάποιες πρώιμες ΓΠ όπως οι πρώτες εκδόσεις της Lisp και οι Snobol4, APL, καθώς και ειδικές γλώσσες μακροεντολών και επεξεργασίας κειμένου (TeX /LaTeX).
- Δυναμική εμβέλεια χρησιμοποιείται σε πολλές σύγχρονες ΓΠ κατά τη διαχείριση εξαιρέσεων.

Παράδειγμα Κανόνων Εμβέλειας

Κανόνες Εμβέλειας της C

- Ένα όνομα έχει **τοπική εμβέλεια**, αν ορίζεται μέσα σε ένα block, οπότε μπορεί να χρησιμοποιηθεί στο block αυτό και σε όσα περικλείει αρκεί να έχει προηγουμένως δηλωθεί. Κατά την έξοδο από ένα bock τα ονόματα που δηλώνονται σε αυτό δεν είναι πλέον διαθέσιμα.
- Τα ονόματα των παραμέτρων συνάρτησης έχουν εμβέλεια όλο το block με τον κώδικα της συνάρτησης. Αν η συνάρτηση δηλώνεται αλλά δεν ορίζεται, τότε η εμβέλεια περιορίζεται στο πρωτότυπο της συνάρτησης.
- Αν ένα block περιέχει ένα άλλο block, οι μεταβλητές του είναι ορατές στο ένθετο block. Αν όμως μια μεταβλητή του ένθετου δηλώνεται με το όνομα μεταβλητής του περικλείοντος block, τότε η ένθετη δήλωση αποκρύπτει μέρος της εμβέλειας της δήλωσης του περικλείοντος block.
- Μόνο τα ονόματα ετικετών έχουν **εμβέλεια συνάρτησης**. Η δήλωση ετικέτας υπονοείται από την εμφάνιση της στο πρόγραμμα και είναι ορατή στη συνάρτηση που αυτή εμφανίζεται. Μία ετικέτα μπορεί να χρησιμοποιηθεί σε εντολή goto πριν από την δήλωσή της.
- Ένα όνομα έχει **εμβέλεια αρχείου** αν δεν εμφανίζεται μέσα σε block.

Διαχείριση μνήμης στις ΓΠ

Στατική εκχώρηση μνήμης.

- Κατάλληλη όταν οι **απαιτήσεις μνήμης είναι γνωστές τη στιγμή της μετάφρασης**.
- Αν η ΓΠ περιλαμβάνει μεταγλώττιση, τότε ο compiler αναθέτει διεύθυνση για τη μεταβλητή/σταθερά που μεταφράζει με ένα offset, που κατά τη διασύνδεση (linking) οδηγεί σε συγκεκριμένη θέση μνήμης.
- Παραδείγματα: όλες οι μεταβλητές στη FORTRAN, οι καθολικές μεταβλητές και οι σταθερές στις C, Ada και Algol.

Διαχείριση μνήμης στις ΓΠ

Στατική εκχώρηση μνήμης.

- Κατάλληλη όταν οι **απαιτήσεις μνήμης είναι γνωστές τη στιγμή της μετάφρασης**.
- Αν η ΓΠ περιλαμβάνει μεταγλώττιση, τότε ο compiler αναθέτει διεύθυνση για τη μεταβλητή/σταθερά που μεταφράζει με ένα offset, που κατά τη διασύνδεση (linking) οδηγεί σε συγκεκριμένη θέση μνήμης.
- Παραδείγματα: όλες οι μεταβλητές στη FORTRAN, οι καθολικές μεταβλητές και οι σταθερές στις C, Ada και Algol.

Εκχώρηση μνήμης στοίβας.

- Κατάλληλη όταν οι **απαιτήσεις μνήμης δεν είναι γνωστές τη στιγμή της μετάφρασης, αλλά ακολουθούν πολιτική Last-In-First-Out**.
- Παραδείγματα: τοπικές μεταβλητές σε υποπρογράμματα στις C/C++, Ada, Algol, Pascal και πληροφορίες κλήσης υποπρογραμμάτων (π.χ. δ/νση επιστροφής).

Διαχείριση μνήμης στις ΓΠ

Εκχώρηση μνήμης στοίβας.

- Επιτρέπει την αναδρομή στα υποπρογράμματα και εκχωρεί μνήμη μόνο όταν αυτά καλούνται.
- Αποδοτική διαχείριση τοπικών μεταβλητών βαθμωτού τύπου, αφού συχνά αυτές περνάνε απαυθείας από τους καταχωρητές.
- Για δυναμικά μεταβαλλόμενα αντικείμενα χρησιμοποιείται **ανακατεύθυνση** (indirection): π.χ. για arrays εκχωρείται χώρος για μία σταθερού μεγέθους περιγραφή (descriptor) με αναφορά στη δ/νση μνήμης που θα βρίσκεται το array και πληροφορίες για τα όρια του.
- Για αναφορές σε μη-τοπικές μεταβλητές υπάρχουν διάφορες τεχνικές, αλλά η πιο διαδεδομένη είναι οι **στατικοί σύνδεσμοι** (static links). Αναφορές σε καθολικές μεταβλητές προγράμματος εκτελούνται πιο αποδοτικά από τις αναφορές σε άλλες μη-τοπικές μεταβλητές.

Διαχείριση μνήμης στις ΓΠ

Εκχώρηση μνήμης στοίβας.

Περιορισμοί:

- Ο μοναδικός τρόπος να επιστραφούν δεδομένα μέσα από υποπρόγραμμα είναι να αντιγραφούν (δε γίνεται με αναφορά).
- Η επιστρεφόμενη τιμή δε μπορεί να είναι ένα υποπρόγραμμα/συνάρτηση και επίσης δε μπορεί να εκχωρηθεί υποπρόγραμμα/συνάρτηση ως τιμή σε καθολική μεταβλητή προγράμματος.

Διαχείριση μνήμης στις ΓΠ

Εκχώρηση μνήμης σωρού.

- Δυνατότητα εκχώρησης και αποδέσμευσης μνήμης δυναμικά, σε οποιοδήποτε χρόνο **κατά τη διάρκεια της εκτέλεσης**. Υπολογιστικά πιο ακριβή από τη στατική εκχώρηση και τη μνήμη στοίβας.
- Εκχώρηση μνήμης σωρού σε διάφορες ΓΠ

C	συνάρτηση `malloc()`
Lisp/Scheme	συναρτήσεις `cons` και `make-array`
Java	εκχώρηση σωρού με τη δημιουργία νέου αντικείμενου

- Αποδέσμευση μνήμης σωρού σε διάφορες ΓΠ
  1. Ελεγχόμενη από προγραμματιστή
     Αποδοτική στην εκτέλεση, με υψηλό κίνδυνο λαθών διαρροής μνήμης ή αιωρούμενων δεικτών (dangling pointers). Στη C γίνεται με κλήση της `free()`.
  2. Αυτόματη
     Οι ΓΠ που την υλοποιούν (Java, Scheme κ.α.) παίρνουν πίσω μετά από ένα χρονικό διάστημα τη μνήμη, που δεν είναι πλέον προσπελάσιμη.

Διαχείριση μνήμης στις ΓΠ

Εκχώρηση μνήμης σωρού.

- Προσεγγίσεις αυτόματης αποδέσμευσης μνήμης σωρού (μερικές ΓΠ υλοποιούνε υβριδικές λύσεις)
  1. Σημάδεμα - σκούπισμα (mark-sweep): όταν εξαντλείται η μνήμη ιχνηλατεί & σημαδεύει όσες θέσεις είναι ακόμη προσεγγίσιμες και στη συνέχεια σκουπίζει όσες δεν έχουν σημαδευτεί.
  2. Αντιγραφή: διαιρείται η μνήμη σε παλιά & νέα περιοχή. Όταν εξαντλείται ο χώρος στην παλιά περιοχή, τότε ιχνηλατούνται όλες οι προσεγγίσιμες θέσεις και αντιγράφονται στη νέα περιοχή. Στη συνέχεια γίνεται ανταλλαγή του περιεχομένου των δύο περιοχών.
  3. Παραγωγική (generational) Κάποιες θέσεις μνήμης χρησιμοποιούνται λίγο και κάποιες άλλες επί μακρόν. Διαιρείται η μνήμη σε περιοχές: νέα, πιο νέα, παλιά κ.λ.π. Συλλέγονται θέσεις συχνά από τις νεώτερες περιοχές. Αν ένα αντικείμενο επιβιώσει αρκτές φορές από την αναδιοργάνωση, τότε προωθείται σε μία παλαιότερη περιοχή.

Διαχείριση μνήμης στις ΓΠ

- Προσεγγίσεις αυτόματης αποδέσμευσης μνήμης σωρού (μερικές ΓΠ υλοποιούνε υβριδικές λύσεις)
  4. Αυξητική: εκτέλεση από ανεξάρτητο νήμα στο περιθώριο για να μη διακόπτεται η λειτουργία τουπρογράμματος.
  5. Μέτρηση αναφορών: σε κάθε αντικείμενο μνήμης υπάρχει ένας μετρητής. Όταν αυτό δημιουργείται παίρνει την τιμή 1 και κάθε φορά που ένας δείκτης αναφέρεται σ΄ αυτό αυξάνεται κατά 1. Όταν ένας δείκτης παύει να αναφέρεται ο μετρητής μειώνεται κατά 1 και όταν γίνει 0, τότε το περιβάλλον εκτέλεσης μπορεί να διεκδικήσει το χώρο για χρήση.

Προτεινόμενες πηγές για περαιτέρω μελέτη

- Σχεδίαση ΓΠ με δομή blocks: στο κεφάλαιο 7 του βιβλίου *Concepts in Programming Languages* του John Mitchell.
- Υλοποιήση περιβάλλοντος εκτέλεσης ΓΠ με δομή blocks: στο κεφάλαιο 8 του βιβλίου *Μεταγλωττιστές Γλώσσών Προγραμματισμού: Θεωρία & Πράξη* των Κ. Λάζου, Π. Κατσαρού και Ζ. Καραϊσκου.

Τέλος ενότητας

Επεξεργασία: Εμμανουέλα Στάχτιαρη
Θεσσαλονίκη, 23/07/2014