Αρχές Γλωσσών Προγραμματισμου

Χρήστος Νομικός

Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Πανεπιστήμιο Ιωαννίνων

2015

Περιεχόμενα

Περιεχόμενα

Εντολή ανάθεσης

Εντολή ανάθεσης: σκοπός της είναι να θέσει ως τιμή της μεταβλητής που βρίσκεται αριστερά του τελεστή ανάθεσης, την τιμή που προκύπτει από την αποτίμηση της παράστασης που βρίσκεται δεξιά του τελεστή ανάθεσης.

Συνήθως χρησιμοποιείται το $=$ ή το $:=$ για να συμβολίσει τον τελεστή ανάθεσης.

Το πώς ακριβώς υλοποιείται η εντολή ανάθεσης εξαρτάται από το αν εφαρμόζεται το μοντέλο τιμών ή το μοντέλο αναφορών για τη μεταβλητή.

Εντολή ανάθεσης

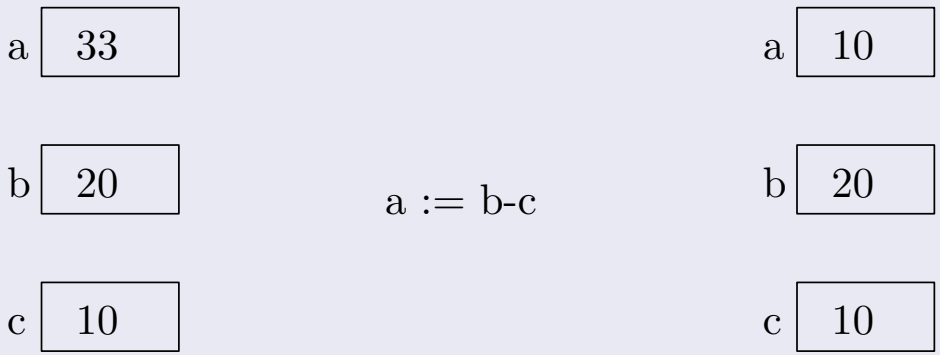
Όταν εφαρμόζεται το μοντέλο τιμών, τότε η τιμή που προκύπτει από την αποτίμηση της παράστασης αποθηκεύται στη θέση μνήμης που αντιστοιχεί στη μεταβλητή.

Όταν εφαρμόζεται το μοντέλο αναφορών, τότε από την αποτίμηση της παράστασης προκύπτει ένα αντικείμενο, το οποίο συνδέεται με τη μεταβλητή (η αναφορά του αντικειμένου αποθηκεύται στη θέση μνήμης που αντιστοιχεί στη μεταβλητή). Το αντικείμενο αυτό μπορεί να υπήρχε πριν από την εκτέλεση της εντολής ανάθεσης ή να κατασκευάστηκε ως αποτέλεσμα της αποτίμησης της παράστασης. Η τιμή της μεταβλητής είναι η τιμή του αντικειμένου με το οποίο συνδέεται.

Εντολή ανάθεσης

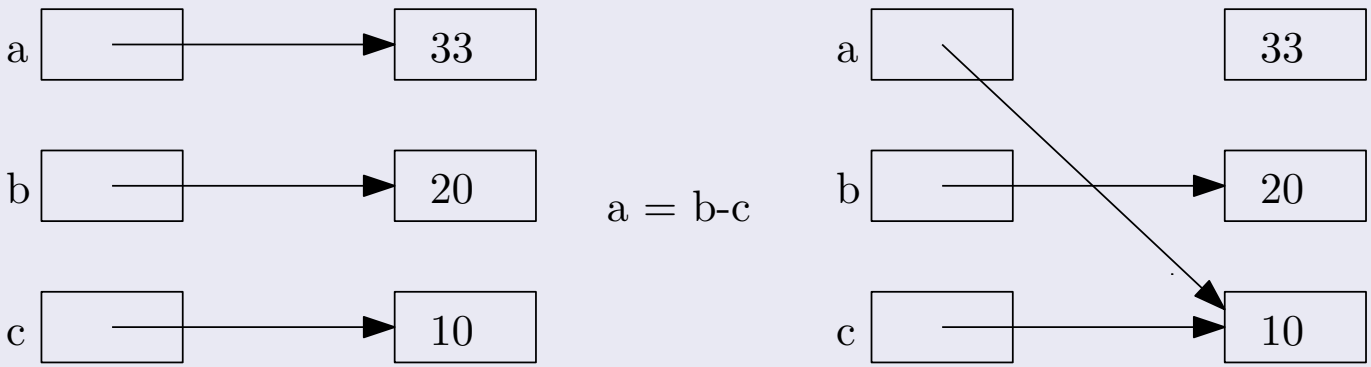
Παράδειγμα

Στο παρακάτω σχήμα φαίνεται το αποτέλεσμα της εκτέλεσης της εντολής a:=b-c στην Pascal (η οποία χρησιμοποιεί μοντέλο τιμών για τις μεταβλητές). Οι τιμές των μεταβλητών a,b,c πριν από την εκτέλεση της εντολής είναι αντίστοιχα 33, 20,10.



Εντολή ανάθεσης

Στο παρακάτω σχήμα φαίνεται το αποτέλεσμα της εκτέλεσης της αντίστοιχης εντολής a=b-c στην Python (η οποία χρησιμοποιεί μοντέλο αναφορών για τις μεταβλητές).



Εντολή ανάθεσης

Παρατηρούμε ότι σε μία εντολή ανάθεσης τα ονόματα των μεταβλητών έχουν διαφορετική σημασία ανάλογα με το αν χρησιμοποιούνται αριστερά ή δεξιά του τελεστή ανάθεσης:

- Στο μοντέλο τιμών, το όνομα μίας μεταβλητής αριστερά του τελεστή ανάθεσης προσδιορίζει τη θέση μνήμης στην οποία αποθηκεύεται η τιμή της μεταβλητής (στην οποία θα αποθηκευτεί η τιμή που θα προκύψει από την αποτίμηση της παράστασης που βρίσκεται δεξιά του τελεστή ανάθεσης), ενώ δεξια του τελεστή ανάθεσης συμβολίζει την τιμή της μεταβλητής.
- Στο μοντέλο αναφορών, το όνομα μίας μεταβλητής αριστερά του τελεστή ανάθεσης προσδιορίζει τη θέση μνήμης στην οποία αποθηκεύεται η αναφορά του αντικειμένου με το οποίο είναι συνδεδεμένη η μεταβλητή, ενώ δεξια του τελεστή ανάθεσης συμβολίζει το αντικείμενο με το οποίο είναι συνδεδεμένη η μεταβλητή.

Εντολή ανάθεσης

Γενικότερα, αριστερά του τελεστή ανάθεσης μπορεί να βρίσκεται σύνθετη έκφραση η οποία προσδιορίζει μία θέση μνήμης στην οποία αποθηκεύται μία τιμή (αν εφαρμόζεται το μοντέλο τιμών) ή η αναφορά ενός αντικειμένου (αν εφαρμόζεται το μοντέλο αναφορών).

Για παράδειγμα αριστερά του τελεστή ανάθεσης μπορεί να υπάρχει κάποιο στοιχείο ενός πίνακα (το οποίο καθορίζεται από το όνομα του πίνακα και τη θέση του στοιχείου που μπορεί να προκύπτει από την αποτίμηση μίας παράστασης), ένα πεδιό μίας εγγραφής, η θέση μνήμης στην οποία δείχνει ένας δείκτης κλπ.

Εντολή ανάθεσης

Παραδείγματα εντολών ανάθεσης με σύνθετες εκφράσεις αριστερά του τελεστή ανάθεσης (χρησιμοποιείται η σύνταξη της Pascal):

```
a[i] := 0;

a[2*i+j] := a[i]+f(n);

r.height := 1.84;

p^.next := nil;
```

Εντολή ανάθεσης

Λόγω της ψευδωνυμίας, είναι δυνατόν η τιμή μίας μεταβλητής να αλλάζει από την εκτέλεση μίας εντολής ανάθεσης, παρότι το όνομα της μεταβλητής δεν εμπλέκεται στην εντολή αυτή.

Εντολή ανάθεσης

Παράδειγμα

Μετά την εκτέλεση των παρακάτω εντολών σε C η τιμή της μεταβλητής n είναι 1.

```
n = 0;
p = &n;
*p = 1;
```

Η δεύτερη εντολή έχει ως αποτέλεσμα ο δείκτης p να δείχνει στη θέση που είναι αποθηκευμένη η n. Η τρίτη εντολή αποθηκεύει την τιμή 1 στη θέση μνήμης στην οποία δείχνει ο p, η οποία είναι η θέση μνήμης που είναι αποθηκευμένη η n. Συνεπώς η τρίτη εντολή αλλάζει την τιμή της n, παρότι η n δεν αναφέρεται σε αυτή.

Εντολή ανάθεσης

Παράδειγμα

Μετά την εκτέλεση των παρακάτω εντολών Python η τιμή της μεταβλητής b είναι [0,2,3].

```
a = [1,2,3]
b = a
a[0] = 0
```

Η πρώτη εντολή έχει ως αποτέλεσμα της δημιουργία ενός αντικειμένου (της λίστας [1,2,3]). Η δεύτερη εντολή έχει ως αποτέλεσμα η μεταβλητή b να συνδεθεί με την ίδια λίστα με την οποία συνδέεται και η a. Η τρίτη εντολή τροποποιεί ένα στοιχείο της λίστας.
Η τιμή της b αλλάζει ύστερα από την εκτέλεση της τρίτη εντολής: παρότι η b εξακολουθεί να συνδέεται με το ίδιο αντικείμενο, η εντολή έχει ως αποτέλεσμα την τροποποίηση του αντικειμένου αυτού (αντικείμενα τα οποία μπορούν να αλλάζουν τιμές όπως οι λίστες ονομάζονται μεταλλάξιμα (mutable)).

Εντολή ανάθεσης

Για να είναι καλά ορισμένη η σημασία της εντολής ανάθεσης θα πρέπει να καθορίζεται αν η διεύθυνση μνήμης αριστερά του τελεστή ανάθεσης προσδιορίζεται πριν ή μετά τον υπολογισμό της παράστασης δεξιά του τελεστή ανάθεσης.

Σε αντίθετη περίπτωση ενδέχεται το αποτέλεσμα της εντολής ανάθεσης να μην ορίζεται μονοσήμαντα.

Εντολή ανάθεσης

Παράδειγμα

Έστω το παρακάτω πρόγραμμα Python:

```
def f(n):
        global i
        i = i-1
        return n*n

a = [1,2,3]
i = 2
a[i] = f(i)
```

Εντολή ανάθεσης

- αν κατά την εκτέλεση της εντολής a[i] = f(i) ο καθορισμός του στοιχείου της λίστας που θα αλλάξει, γίνει πριν από τον υπολογισμό του f(i), τότε το i θα έχει τιμή 2 και μετά την εκτέλεση των παραπάνω εντολών η a θα έχει τιμή [1,2,4].
- αν αντιθέτως ο καθορισμός του στοιχείου της λίστας που θα αλλάξει, γίνει μετά από τον υπολογισμό του f(i), τότε το i θα έχει τιμή 1 και μετά την εκτέλεση των παραπάνω εντολών η a θα έχει τιμή [1,4,3].

Εντολή ανάθεσης

Υπάρχουν πιο σύνθετες μορφές εντολής ανάθεσης

- Πολλαπλή ανάθεση. Python:

  ```
  x,y = y,x
  ```

  Επιτρέπει την εναλλαγή των τιμών δύο μεταβλητών χωρις χρήση τρίτης μεταβλητής.
- Συνδυασμένη ανάθεση. C, C++, Java:

  ```
  student[code].grade[index] *= 1.2;
  ```

  Επιτρέπει την αποφυγή της επανάληψης της ίδιας έκφρασης αριστερα και δεξιά του τελεστή ανάθεσης.
- Εντολές αύξησης ή μειωσης κατά 1. C, C++, Java:

  ```
  x++; ++x; x--; --x;
  ```

Εντολή ανάθεσης

Σε ορισμένες γλώσσες (π.χ.C, C++, Java) η εντολή ανάθεσης θεωρείται ως παράσταση που έχει ως τιμή την ανατιθέμενη τιμή.

Ο τελεστής = προσεταιρίζεται από δεξιά προς τα αριστερά.

Εντολή ανάθεσης

Αυτό επιτρέπει να γράφουμε αλυσιδωτές αναθέσεις:

```
c = b = 1;
```

Το b = 1, εκτός του ότι αναθέτει την τιμή 1 στη b, λαμβάνεται ως παράσταση που έχει ως τιμή 1. Η τιμή 1 ανατίθεται και στη c.

Επιπλέον μπορούμε να έχουμε περισσότερες από μία αναθέσεις διαφορετικών τιμών σε μεταβλητές σε μία μόνο εντολή. Η παρακάτω εντολή έχει ως αποτέλεσμα να ανατεθούν οι τιμές 2,3 και 5 αντίστοιχα στις a,b,c:

```
c = (a=2) + (b=3);
```

Εντολή ανάθεσης

Η τιμή μίας μεταβλητής μπορεί να αρχικοποιηθεί με μία εντολή ανάθεσης στην αρχή του μπλοκ στο οποίο δηλώνεται, ή σε γενικότερα σε οποιοδήποτε σημείο πριν από την χρησιμοποίηση της τιμής της μεταβλητής σε κάποια παράσταση.

Ορισμένες γλώσσες παρέχουν τη δυνατότητα αρχικοποίησης της μεταβλητής κατά τη δήλωσή της, όπως για παράδειγμα η C:

```
int n=0, a[3]={4,5,7};
```

Επίσης ορισμένες γλώσσες αναθέτουν προκαθορισμένες τιμές σε μεταβλητές που δεν έχουν αρχικοποιηθεί. Για παράδειγμα η C αναθέτει μηδενικές τιμές σε όλες τις στατικές μεταβλητές που δεν αρχικοποιούνται.

Εντολή ανάθεσης

Σε ορισμένες γλώσσες χρησιμοποιείται το σύμβολο = για ανάθεση αλλά και για σύγκριση. Αυτό μειώνει την αναγνωσιμότητα του προγράμματος. PL/1:.

```
A = B = C
```

Περιεχόμενα

Εντολές επιλογής

Εντολές επιλογής: σκοπός αυτής της κατηγορίας εντολών είναι η εκτέλεση μίας ή περισσοτέρων εντολών μόνο στην περίπτωση που ικανοποιούνται κάποιες προϋποθέσεις.

Υπάρχουν δύο κατηγορίες εντολών επιλογής, ανάλογα με το πώς καθορίζεται η εντολή που θα εκτελεστεί:

- από μία λογική συνθήκη
- από την τιμή μίας παράστασης

Εντολές επιλογής

Μία εντολή επιλογής με λογική συνθήκη περιλαμβάνει μία λογική συνθήκη και μία ή περισσότερες εντολές που εκτελούνται αν αληθεύει η συνθήκη.

Στις περισσότερες γλώσσες αυτή η εντολή ονομάζεται if.

Εντολές επιλογής

Σε κάποιες γλώσσες η σύνταξη της εντολής if επιτρέπει την εκτέλεση μίας μόνο εντολής στην περίπτωση που αληθεύει η συνθήκη. Ωστόσο κατά κανόνα αυτές οι γλώσσες επιτρέπουν το σχηματισμό σύνθετων εντολών (με begin - end ή { }). Συνεπώς για την εκτέλεση περισσότερων από μία εντολές όταν αληθεύει η συνθήκη, σχηματίζεται μία σύνθετη εντολή που αποτελείται από τις εντολές αυτές.

Σε άλλες γλώσσες υπάρχει κάποιος τρόπος δήλωσης του τέλους τις εντολής if (π.χ. με end ή end if ή με βάση κάποιους κανόνες στοίχισης). Σε αυτή την περίπτωση η συνθήκη της εντολής if μπορεί να ακολουθείται από οποιοδήποτε πλήθος εντολών.

Εντολές επιλογής

Παράδειγμα

Η παρακάτω εντολή if σε Pascal αντιμεταθέτει τις τιμές των στοιχείων A[i] και A[i+1], αν το A[i] έχει μεγαλύτερη τιμή από το A[i+1]:

```
if A[i] > A[i+1] then
   begin
      tmp := A[i];
      A[i] := A[i+1];
      A[i+1] := tmp
   end;
```

Εντολές επιλογής

Η αντίστοιχη εντολή σε C ή σε Java είναι:

```
if (A[i] > A[i+1]) {
      tmp = A[i];
      A[i] = A[i+1];
      A[i+1] = tmp;
   }
```

Η αντίστοιχη εντολή σε Modula 2 είναι:

```
IF A[i] > A[i+1] THEN
      tmp := A[i];
      A[i] := A[i+1];
      A[i+1] := tmp;
END;
```

Εντολές επιλογής

Τέλος σε Python η αντίστοιχη εντολή είναι:

```
if A[i] > A[i+1]:
    tmp = A[i]
    A[i] = A[i+1]
    A[i+1] = tmp
```

(η επόμενη εντολή, αν υπάρχει, θα πρέπει να ξεκινήσει στην ίδια στήλη με τη λέξη if).

Εντολές επιλογής

Στις περισσότερες περιπτώσεις δίνεται η δυνατότητα καθορισμού μιας εντολής που θα εκτελεστεί αν δεν αληθεύει η συνθήκη. Αυτή συνήθως ακολουθεί τη λέξη else.

Εντολές επιλογής

Παράδειγμα

Η παρακάτω εντολή if σε Pascal βρίσκει το μεγαλύτερο ανάμεσα σε δύο δύο αριθμούς:

```
if a >= b then
   max := a
else
   max := b
```

Η αντίστοιχη εντολή σε C ή σε Java είναι:

```
if (a >= b)
   max = a;
else
   max = b;
```

Εντολές επιλογής

Η αντίστοιχη εντολή σε Modula 2 είναι:

```
IF a >= b THEN
   max := a;
ELSE
   max := b;
END;
```

Τέλος σε Python η αντίστοιχη εντολή είναι:

```
if a >= b:
   max = a
else:
   max = b
```

(η επόμενη εντολή, αν υπάρχει, θα πρέπει να ξεκινάει στην ίδια στήλη με τις λέξεις if και else).

Εντολές επιλογής

Σε γλώσσες στις οποίες δεν υπάρχει κάποιος τρόπος δήλωσης της ολοκλήρωσης της εντολής if, σε περίπτωση που υπάρχουν στο πρόγραμμα φωλιασμένες εντολές if ακολουθούμενες από ένα else, θα πρέπει η γλώσσα να καθορίζει σε ποιό από όλα τα if αναφέρεται το συγκεκριμένο else:

```
if ... then
if ... then ...
else ...
```

Εντολές επιλογής

Η πιο συνήθης προσέγγιση είναι το else να ταιριάζει με το πιο πρόσφατο (δηλαδή το πιο εσωτερικό) if το οποίο δεν έχει ταιριάξει με κάποιο προηγούμενο else. Η προσέγγιση αυτή χρησιμοποιείται μεταξύ άλλων από την Pascal, τη C και τη Java.

Η ALGOL 60 ακολουθεί μία διαφορετική προσσέγιση, μην επιτρέποντας να υπάρχουν άμεσα φωλιασμένα if. Αν θέλουμε η εντολή που ακολουθεί το if ή το else να είναι επίσης if, την κλείνουμε μέσα σε begin - end.

Εντολές επιλογής

Παράδειγμα

Η παρακάτω εντολή if σε Pascal δίνει στη μεταβλητή z την τιμή x mod y όταν οι τιμές των x και y είναι θετικές και την τιμή -1 όταν η τιμή της x είναι θετική και της y αρνητική ή 0. Αν η x δεν έχει θετική τιμή η τιμή της z δεν αλλάζει.

```
if x > 0 then
if y > 0 then z := x mod y
else z := -1
```

Εντολές επιλογής

Ορισμένες γλώσσες (Modula 2, ALGOL68, FORTRAN 77, Ada) δίνουν τη δυνατότητα ελέγχου εναλλακτικών συνθηκών αν δεν ικανοποιείται η πρώτη συνθήκη της if, οι οποίες ακολουθούν μία λέξη κλειδί (π.χ. elsif ή elif).

Σε αυτή τη μορφή της if, εξετάζεται ποια είναι η πρώτη κατά σειρά συνθήκη (μετά από if ή elsif) που αληθεύει και εκτελούνται οι εντολές που ακολουθούν την αντίστοιχη συνθήκη. Αν καμια συνθήκη δεν αληθεύει εκτελείται η εντολή που ακολουθεί το else.

Εντολές επιλογής

Παράδειγμα

Η παρακάτω εντολή if σε Modula 2 αναθέτει στην μεταβλητή days το πλήθος των ημερών του μήνα που προσδιορίζεται από την τιμή της month

```
IF month = 2 THEN
   days := 28;
ELSIF (month = 4) OR (month = 6) OR (month = 9) OR (month = 11) THEN
   days := 30;
ELSE
   days := 31;
END
```

Εντολές επιλογής

Μία παραλλαγή της παραπάνω εντολής σε Python είναι:

```
if month == 2:
   days = 28
elif month in {4,6,9,11}:
   days = 30
else:
   days = 31
```

Εντολές επιλογής

Σε μια εντολή επιλογής με παράσταση, οι εντολές που θα εκτελεστούν καθορίζονται από την τιμή μίας παράστασης.

Η χρήση μία εντολής επιλογής με παράσταση είναι προτιμότερη από τη χρήση μίας ισοδύναμης εντολής if, τόσο για λόγους αναγνωσιμότητας, αλλά επειδή μπορεί να οδηγήσει σε παραγωγή αποδοτικότερου κώδικα.

Παραδείγματα εντολών που εμπίπτουν στην παραπάνω κατηγορία:

- το αριθμητικό IF της FORTRAN
- οι εντολες case της Pascal (προτοεμφανίστηκε στην Algol-W) και switch της C.

Εντολές επιλογής

Στην εντολή case της Pascal δίνεται μία παράσταση και ένα σύνολο περιπτώσεων. Κάθε περίπτωση περιέχει ένα σύνολο τιμών και μία εντολή (η οποία ενδέχεται να είναι σύνθετη). Αν η αποτίμηση της παράστασης δώσει τιμή που περιλαμβάνεται σε κάποια περίπτωση τότε εκτελείται η αντίστοιχη εντολή (και μόνο αυτή).

Αν η τιμή της παράστασης δεν περιλαμβάνεται σε κάποια περίπτωση τότε το αποτέλεσμα είναι απροσδιόριστο. Ορισμένες διάλεκτοι της Pascal επιτρέπουν να δοθεί μία εντολή που θα εκτελεστεί στην περίπτωση αυτή μετά από τη λέξη otherwise ή else.

Εντολές επιλογής

Η εντολή switch της C λειτουργεί παρόμοια, όμως για μία δεδομένη τιμή δεν εκτελεί μόνο μία εντολή, αλλά όλες τις εντολές που ακολουθούν την εμφάνιση της τιμής στην switch μέχρι το τέλος της switch. Αυτό μπορεί να αποφευχθεί με χρήση της break.

Η εντολή switch επιτρέπεται να έχει στο τέλος της μία ετικέτα default, η οποία ταιριάζει με οποιαδήποτε τιμή. Οι εντολές που ακολουθούν την default εκτελούνται για όλες τις τιμές, εκτός από αυτές για τις οποίες έχει εκτελεστεί προηγουμένως η εντολή break.

Εντολές επιλογής

Παράδειγμα

Η παρακάτω εντολή case μπορεί να χρησιμοποιηθεί για την αναθέτει στην μεταβλητή days του πλήθους των ημερών του μήνα που προσδιορίζεται από την τιμή της month:

```
case month of
  2: days := 28;
  4,6,9,11: days := 30;
  1,3,5,7,8,10,12: days := 31
end
```

Εντολές επιλογής

Η αντίστοιχη εντολή switch είναι:

```
switch(month) {
  case 2: days = 28; break;
  case 4:
  case 6:
  case 9:
  case 11: days = 30; break;
  default: days = 31;
}
```

Περιεχόμενα

Εντολές επανάληψης

Εντολές επανάληψης: σκοπός αυτής της κατηγορίας εντολών είναι η επαναληπτική εκτέλεση μίας ή περισσοτέρων εντολών.

Υπάρχουν δύο κατηγορίες εντολών επανάληψης, ανάλογα με το πώς καθορίζεται το αν θα υπάρχει άλλη επανάληψη:

- Με λογική συνθήκη
- Με μεταβλητή ελέγχου

Εντολές επανάληψης

Μία εντολή επανάληψης με λογική συνθήκη περιλαμβάνει μία λογική συνθήκη και μία ή περισσότερες εντολές που εκτελούνται ενώσο αληθεύει η συνθήκη.

Διακρίνουμε δύο περιπτώσεις, ανάλογα με το πότε γίνεται ο έλεγχος της συνθήκης:

- Η συνθήκη ελέγχεται πριν την εκτέλεση των εντολών (εντολή while, εντολή for της C και της Java). Αν αρχικά η συνθήκη δεν αληθεύει τότε οι εντολές δεν εκτελούνται ποτέ.
- Η συνθήκη ελέγχεται μετά την εκτέλεση των εντολής (εντολή repeat-until της Pascal, εντολή do-while της C). Η εντολές εκτελούνται τουλάχιστον μία φορά.

Εντολές επανάληψης

Παράδειγμα

Η παρακάτω εντολή while σε Pascal βρίσκει τη θέση του στοιχείου a σε έναν ταξινομημένο πίνακα S. Υποθέτουμε ότι το στοιχείο υπάρχει στον πίνακα και ότι οι αρχικές τιμές των i και j είναι τα όρια του πίνακα S. Η τελική τιμή των i και j δίνει τη θέση του στοιχείου a:

```
while i<j do
   begin
     k := (i+j) div 2;
     if a > S[k] then i := k+1
     else j := k
   end
```

Εντολές επανάληψης

Η αντίστοιχη εντολή σε C ή σε Java είναι:

```
while (i<j) {
     k = (i+j) / 2;
     if (a > S[k]) i = k+1;
     else j = k;
}
```

Η αντίστοιχη εντολή σε Python είναι:

```
while i<j:
     k = (i+j) // 2
     if a > S[k]:
        i = k+1
     else:
        j = k
```

Εντολές επανάληψης

Παράδειγμα

Η παρακάτω εντολή repeat-until σε Pascal βρίσκει (υπό προϋποθέσεις) ένα σταθερό σημείο της συνάρτησης f.

```
repeat
   y := x;
   x := f(x)
until y=x
```

Εντολές επανάληψης

Η αντίστοιχη εντολή σε C ή σε Java είναι:

```
do {
      y = x;
      x = f(x);
} while (x != y);
```

Εντολές επανάληψης

Σε μία εντολή επανάληψης με μεταβλητή ελέγχου, το πλήθος των επαναλήψεων καθορίζεται από έναν είδος μετρητή, τη μεταβλητή ελέγχου.

Ο καθορισμός της αρχικής τιμής, της τελικής τιμής και του βήματος αποτελούν τμήματα της εντολης επανάληψης. Συνήθως εντολή επανάληψης με μεταβλητή ελέγχου ονομάζεται for.

Εντολές επανάληψης

Υπάρχουν πολλές εναλλακτικές μορφές που μπορεί να πάρει η for, που καθορίζονται από τις απαντήσεις στα παρακάτω ερωτήματα:

- Ποιοί είναι οι επιτρεπτοί τύποι για τη μεταβλητή ελέγχου;
- Πόσο σύνθετες παραστάσεις μπορεί να είναι η αρχική τιμή, η τελική τιμή και το βήμα;
- Πόσο συχνά υπολογίζεται το βήμα και η τελική τιμή;
- Πότε ελέγχεται η τιμή της μεταβλητής ελέγχου;
- Μπορεί να γίνει ανάθεση τιμής στη μεταβλητή ελέγχου;
- Ποιά είναι η τιμή της μεταβλητής ελέγχου μετά την ολοκλήρωση της for;

Εντολές επανάληψης

Στην PASCAL:

- Η μεταβλητή ελέγχου μπορεί να πάρει οποιονδήποτε βαθμωτό τύπο εκτός από real (π.χ. ακέραιο, χαρακτήρα, απαριθμητό τύπο).
- Η αρχική και τελική τιμή μπορεί να είναι οποιαδήποτε παράσταση του ίδιου τύπου με τη μεταβλητή ελέγχου, ενώ το βήμα είναι +1 ή -1 (σε μη αριθμητικούς τύπους είναι η προηγούμενη ή η επόμενη τιμή).
- Η αρχική και η τελική τιμή υπολογίζονται μόνο μία φορά, στην αρχή.
- Η τιμή της μεταβλητής ελέγχου ελέγχεται στην αρχή κάθε επανάληψης.
- Δεν μπορεί να γίνει ανάθεση τιμής στη μεταβλητή ελέγχου.
- Η τιμή της μεταβλητής ελέγχου μετά την εκτέλεση της for είναι αόριστη.

Εντολές επανάληψης

Παράδειγμα

Η παρακάτω φωλιασμένες εντολές for σε Pascal υπολογίζουν το γινομενο δύο πινάκων A και B διάστασης $n \times n$:

```
for i:=1 to n do
    for j:=1 to n do
        begin
           sum := 0;
           for k:=1 to n do
               sum := sum + A[i,k]*B[k,j];
           C[i,j] := sum
        end
```

Εντολές επανάληψης

Παράδειγμα

Η παρακάτω εντολή for σε Pascal ολισθαίνει τα στοιχεία του πίνακα A κατά μία θέση. Το downto δηλωνει ότι το βήμα είναι -1:

```
for i:=n downto 2 do
    A[i] := A[i-1]
```

Εντολές επανάληψης

Στις εκδοχή της εντολής for που περιγράψαμε μέχρι τώρα, οι τιμές της μεταβλητής ελέγχου διαφέρουν κατά μία σταθερή τιμή που καθορίζεται από το βήμα. Η γλώσσα Clu εισήγαγε έναν πιο γενικό και ευέλικτο μηχανισμό για τον καθορισμό των τιμών της μεταβλητής ελέγχου, τους επαναλήπτες.

Ένας επαναλήπτης είναι παρόμοιος μέ ένα υποπρόγραμμα, με την παρακάτω διαφορά: Ο επαναλήπτης επιστρέφει τιμές με χρήση μίας εντολής yield (αντί της return). Μετά την εκτέλεση της yield η λειτουργία του επαναλήπτη δεν ολοκληρώνεται αλλά αναστέλλεται. Όταν η μεταβλητή ελέγχου χρειαστεί την επόμενη τιμή, η λειτουργία του επαναλήπτη συνεχίζεται ακριβώς μετά από το σημείο που βρίσκεται η τελευταία εντολή yield που εκτελέστηκε.

Εντολές επανάληψης

Χρησιμοποιώντας επαναλήπτες είναι δυνατόν η μεταβλητή ελέγχου να πάρει διαδοχικά τις τιμές οποιουδήποτε συνόλου το οποίο μπορεί να κατασκευαστεί από το πρόγραμμα και με οποιαδήποτε σειρά.

Επίσης οι επαναλήπτες δίνουν τη δυνατότητα διαχωρισμού της παραγωγής των τιμών της μεταβλητής ελέγχου από την ίδια την εντολή for που χρησιμοποιεί τις τιμές. Με αυτόν τον τρόπο διαφορετικές εντολές for μπορούν να χρησιμοποιούν τον ίδιο επαναλήπτη.

Επαναλήπτες υποστηρίζονται από σύγχρονες γλώσσες όπως η Python, η Ruby και η C#.

Εντολές επανάληψης

Παράδειγμα

Ο παρακάτω επαναλήπτης σε Python κατασκευάζει όλα τα υποσύνολα με m στοιχεία ενός συνόλου S:

Εντολές επανάληψης

```
def combinations(S,m):
    L = list(S)
    n = len(L)
    if m>n: return
    k = n-m+1
    c = L[:m]
    yield(set(c))
    if m == 0: return
    while c[0]!=L[-m]:
        i = m-1
        while i>=0 and c[i]==i+k:
            i = i-1
        c[i] = c[i]+1
        i = i+1
        while i<m:
            c[i] = c[i-1]+1
            i = i+1
        yield(set(c))
    return
```

Εντολές επανάληψης

Μπορούμε επαναληπτικά νε επεξεργαστούμε (π.χ. να τυπώσουμε) τα υποσύνολα πληθικότητας m ενός συνόλου S, χρησιμοποιώντας τον επαναλήπτη combinations σε μία εντολή for:

```
>>> for i in combinations({1,2,3,4,5},2):
    print(i)

{1, 2}
{1, 3}
{1, 4}
{1, 5}
{2, 3}
{2, 4}
{2, 5}
{3, 4}
{3, 5}
{4, 5}
```

Εντολές επανάληψης

Σε αντικειμενοστρεφείς γλώσσες που δεν υποστηρίζουν την εντολή yield δεν μπορούν να οριστούν αληθινοί επαναλήπτες. Σε αυτές τις γλώσσες οι εντολή for μπορεί να χρησιμοποιεί αντικείμενα επαναλήπτες, τα οποία παρέχουν ακολουθιακά όλα τα στοιχεία ενός συνόλου.

Για την αναπαράσταση συνόλων στις αντικειμενοστρεφείς γλώσσες χρησιμοποιούνται αντικείμενα τα οποία ονομάζονται αποθήκες (containers). Ένα αντικείμενο-επαναλήπτης συνδέεται με ένα αντικείμενο-αποθήκη και να παρέχει τα στοιχεία του, ένα κάθε φορά, με χρήση κατάλληλων μεθόδων.

Σημειώνεται ότι ο επάναλήπτης δεν ταυτίζεται με την ίδια την αποθήκη, καθώς μπορούμε να ορίσουμε περισσότερους από έναν επαναλήπτες για το ίδιο σύνολο στοιχείων (καθένας από τους οποίους επιστρέφει τα στοιχεία με διαφορετική σειρά).

Εντολές επανάληψης

Η εκτέλεση μιας οποιασδήποτε εντολής επανάληψης είναι δυνατόν να διακόπτεται με μη ομαλό τρόπο, αν χρησιμοποιηθούν εντολές όπως οι goto, break και continue, οι οποίες περιγράφονται παρακάτω.

Η μη ομαλή έξοδος από εντολές επανάληψης μειώνει την αναγνωσιμότητα του προγράμματος και είναι καλό χρήση της να γίνεται με μέτρο.

Εντολές επανάληψης

Μπορούμε να επαναλάβουμε εντολές με έμμεσους τρόπους, χρησιμοποιώντας

- αναδρομή (στις συναρτησιακές και λογικές γλώσσες είναι ο μοναδικός τρόπος για να γίνει επανάληψη)
- την εντολή goto (είναι ο μοναδικός τρόπος για επανάληψη στη γλώσσα μηχανής)

Περιεχόμενα

Εντολές μετάβασης

Εντολές μετάβασης: σκοπός αυτής της κατηγορίας εντολών είναι το να καθορίζουν την εντολή με την οποία θα συνεχιστεί η εκτέλεση του προγράμματος.

Στη συνέχεια περιγράφονται οι βασικότερες εντολές αυτής της κατηγορίας.

Εντολές μετάβασης

Η εντολή άλματος, η οποία συνήθως ονομάζεται goto, καθορίζει με άμεσο τρόπο την επόμενη εντολή που θα εκτέλεστεί: η εντολή που θα εκτελεστεί είναι αυτή που βρίσκεται αμέσως μετά από την ετικέτα η οποία προσδιορίζεται στην εντολή goto.

Η εντολή goto έχει αρνητική επίδραση στην αναγνωσιμότητα του προγράμματος, ενώ μπορεί να έχει απροσδιόριστα αποτελέσματα σε μερικές περιπτώσεις (π.χ. μετάβαση στο εσωτερικό ενός βρόχου for, παρακάμπτοντας την αρχικοποίηση της μεταβλητής ελέγχου ή στο σώμα ενός υποπρογράμματος, παρακάμπτοντας την ενημέρωση της στοίβας και το πέρασμα παραμέτρων).

Εντολές μετάβασης

Στις ορισμένες από τις πρώτες γλώσσες υψηλού επιπέδου που αναπτύχθηκαν (π.χ. Cobol, PL1), η χρήση της εντολής goto ήταν ο μοναδικός τρόπος για την υπλοποίηση της επανάληψης.

Στις γλώσσες που παρέχουν δομημένες εντολές επανάληψης, η goto μπορεί να φανεί χρήσιμη μόνο σε περιορισμένες περιπτώσεις, όπως οι παρακάτω:

- άμεσος τερματισμός της εκτέλεσης μίας εντολής επανάληψης (με άλμα στην πρώτη εντολή μετά την εντολή επανάληψης).
- άμεσος τερματισμός της τρέχουσας επανάληψης σε μία εντολή επανάληψης (με άλμα στο τέλος του μπλοκ των εντολών που εκτελούνται επαναληπτικά).
- άμεσος τερματισμός της εκτέλεσης ενός υποπρογραμματος (με άλμα στο τέλος του σώματος του υποπρογράμματος).

Εντολές μετάβασης

Οι πιο σύγχρονες γλώσσες, όπως η Clu, η Java και η Python, δεν έχουν εντολή goto και παρέχουν ειδικές εντολές οι οποίες μπορούν να εκτελεστούν στις παραπάνω περιπτώσεις.

Στην Pascal υπάρχει εντολή goto, με διάφορους περιορισμούς, ωστόσο η χρήση της δεν συνίσταται. Στις παραπάνω περιπτώσεις αντί για χρήση της εντολής goto σε κάποιο σημείο του σώματος μίας εντολής επανάληψης ή ενός υποπρογράμματος, μπορεί να χρησιμοποιηθούν κατάλληλες λογικές μεταβλητές και το υπόλοιπο του σώματος να μπεί μέσα σε μία εντολή if.

Εντολές μετάβασης

Παράδειγμα

Το παρακάτω τμήμα προγράμματος σε C βρίσκει την πρώτη θέση ενός διδιάστατου πίνακα στην οποία υπάρχει το στοιχείο 0. Όταν εντοπιστεί το στοιχείο αυτό χρησιμοποιείται η εντολή goto για άμεσο τερματισμό των φωλιασμένων εντολών for:

```
for (i = 0; i < MAX; i++) {
    for (j = 0; j < MAX; j++) {
        if (A[i][j] == 0) goto Label1;
    }
  }
Label1: ...
```

Εντολές μετάβασης

Η εντολή άμεσου τερματισμού εντολής επανάληψης, προκαλεί διακοπή της εκτέλεσης της εντολής επανάληψης στην οποία περιέχεται και τη μετάβαση της ροής εκτέλεσης του προγράμματος αμέσως μετά την εντολή επανάληψης.

Τέτοιες εντολές είναι για παράδειγμα η break της C, της Java, και της Python και η EXIT της FORTRAN 90.

Εντολές μετάβασης

Στη C και στην Python η break επιδρά στην εσωτερικότερη εντολή επανάληψης που την περιέχει.

Στη Java μπορεί να επιλεγεί η διακοπή οποιασδήποτε εντολής επανάληψης απο αυτές που περιέχουν τη break, χρησιμοποιώντας κατάλληλη ετικέτα.

Στη FORTRAN η εντολή EXIT προκαλεί άμεση έξοδο από την εξωτερικότερη εντολή επανάληψης που την περιέχει.

Η Pascal που υποστηρίζει το δομημένο προγραμματισμό, δεν έχει κάποια εντολή αυτής της κατηγορίας.

Εντολές μετάβασης

Παράδειγμα

Το παρακάτω τμήμα προγράμματος σε Java εκτελεί την ίδια λειτουργία με το τμήμα προγράμματος του προηγούμενου παραδείγματος. Αντί της εντολής goto (η οποία δεν υπάρχει στη Java) χρησιμοποιείται η εντολή break για την έξοδο από τον εξωτερικότερο βρόχο for (όπως προσδιορίζει η ετικέτα που ακολουθέι τη λέξη break):

```
OuterLoop: for (i = 0; i < MAX; i++) {
               for (j = 0; j < MAX; j++) {
                   if (A[i][j] == 0) break OuterLoop;
               }
           }
```

Εντολές μετάβασης

Στη C για να επιτευχθεί το ίδιο αποτέλεσμα, χρειάζονται δύο εντολές break (μία για έξοδο από το εσωτερικό for και άλλη μία για έξοδο από το εξωτερικό for):

```
for (i=0; i<MAX; i++) {
    for (j=0; j<MAX; j++) {
        if (A[i][j] == 0) break;
    }
    if (j<MAX) break;
  }
```

Εντολές μετάβασης

Η εντολή άμεσου τερματισμού της τρέχουσας επανάληψης, έχει ως αποτέλεσμα την άμεση ολοκληρωση της τρέχουσας επανάληψης ενός επαναληπτικού βρόχου και την μετάβαση της ροής εκτέλεσης του προγράμματος στον έλεγχο της συνθήκης της εντολής επανάληψης (ώστε να εξεταστεί αν θα υπάρχει επόμενη επανάληψη ή θα ολοκληρωθεί η εκτέλεση του βρόχου).

Τέτοιες εντολές είναι για παράδειγμα η continue της C, της Java, και της Python και η CYCLE της FORTRAN 90.

Εντολές μετάβασης

Η εντολή continue της C και της Python και η CYCLE της FORTRAN 90 επιδρούν στην εσωτερικότερη εντολή επανάληψης που την περιέχει.

Στη Java μπορεί να επιλεγεί η διακοπή της τρέχουσας επανάληψης οποιασδήποτε εντολής επανάληψης απο αυτές που περιέχουν την continue, χρησιμοποιώντας κατάλληλη ετικέτα.

Η Pascal δεν έχει κάποια εντολή αυτής της κατηγορίας.

Εντολές μετάβασης

Παράδειγμα

Το παρακάτω τμήμα προγράμματος σε Java μετράει το πλήθος των γραμμών ενός πίνακα οι οποίες περιέχουν το στοιχείο 0. Όταν εντοπιστεί σε μία γραμμή το στοιχείο 0, η εντολή continue προκαλεί τερματισμό της τρέχουσας επανάληψης του εξωτερικού for (όπως προσδιορίζει η ετικέτα) ώστε να συνεχιστεί ο έλεγχος στην επόμενη γραμμή του πίνακα (αν υπάρχει):

```
OuterLoop: for (counter=0,i = 0; i < MAX; i++) {
               for (j = 0; j < MAX; j++) {
                   if (A[i][j] == 0) {
                      counter++;
                      continue OuterLoop;
                   }
               }
           }
```

Εντολές μετάβασης

Η κλήση υποπρογράμματος, είναι μία εντολή μετάβασης η εκτέλεση της οποίας έχει ως αποτέλεσμα η ροή της εκτέλεσης του προγράμματος να μεταφερθεί στην πρώτη εντολή του υποπρογράμματος (αφού πρώτα εκτελεστούν οι απαραίτητες εντολές για ενημέρωση της στοίβας, πέρασμα παραμέτρων κλπ).

Στις περισσότερες γλώσσες η εντολή αυτή δεν έχει κάποιο όνομα και για την εκτέλεσή της γράφεται απλά το όνομα του υποπρογράμματος μαζί με τις πραγματικές παραμέτρους.

Εντολές μετάβασης

Η εντολή επιστροφής από υποπρόγραμμα έχει ως αποτέλεσμα τον άμεσο τερματισμό της εκτέλεσης ενός υποπρογράμματος, και τη μετάβαση της ροής εκτέλεσης του προγράμματος αμέσως μετά από το σημείο από το οποίο είχε κληθεί το υποπρόγραμμα.

Τέτοια εντολή είναι η return της C, της Java και της Python.

Όταν χρησιμοποιείται μέσα σε συνάρτηση, συνήθως επιστρέφει και την τιμή της συνάρτησης.

Εντολές μετάβασης

Παρόμοια εντολή είναι και η yield που χρησιμοποιείται από τους επαναλήπτες, η οποία ωστόσο αναστέλλει (και δεν τερματίζει) τη λειτουργία του υποπρογράμματος.

Η Pascal δεν έχει κάποια εντολή αυτής της κατηγορίας (το υποπρόγραμμα ολοκληρώνεται με την εκτέλεση της τελευταία εντολής και υπάρχει διαφορετικός μηχανισμός επιστροφής τιμής από τις συναρτήσεις).

Εντολές μετάβασης

Η εντολή ρίψης εξαίρεσης έχει ως αποτέλεσμα τον άμεσο τερματισμό της εκτέλεσης ενός ή περισσότερων υποπρογραμμάτων τα οποία είναι ενεργά, μέχρι να βρεθεί ένα τμήμα του προγράμματος το οποίο μπορεί να χειριστεί την εξαίρεση.

Εντολές αυτού του είδους υπάρχουν σε γλώσσες που υποστηρίζουν χειρισμό εξαιρέσεων, ο οποίος είναι ένας μηχανισμός ώστε να μπορεί το πρόγραμμα να διαχειρίζεται απρόβλεπτες καταστάσεις (για τις οποίες στο σημείο που εμφανίζονται δεν υπάρχει πάντα η απαραίτητη πληροφορία για το πώς πρέπει να αντιμετωπιστούν).

Τέτοια εντολή είναι η throw της Java.

Περιεχόμενα

Σύνθετη εντολή και άλλες εντολές

Σύνθετη εντολή: σκοπός της είναι να ομαδοποιήσει ένα πλήθος εντολών, έτσι ώστε να μπορούν να χρησιμοποιηθούν συντακτικά ως μία μοναδική εντολή.

Οι εντολές τοποθετούνται μέσα σε begin - end (π.χ. Pascal) ή { } (π.χ. C, Java).

Η Pascal χρησιμοποιεί για διαχωριστικό το ; ενώ η C και η Java δεν χρησιμοποιούν κάποιο διαχωριστικό, καθώς κάθε εντολή τερματίζεται με ;.

Σύνθετη εντολή και άλλες εντολές

Άλλες εντολές: ενδέχεται τέλος μια γλώσσα να περιέχει και άλλες εντολές όπως:

- Εντολές εισόδου - εξόδου (αν και συνήθως χρησιμοποιούνται προκαθορισμένες συναρτήσεις ή διαδικασίες)
- Εντολή with της Pascal: χρησιμοποιείται για ευκολότερη πρόσβαση των πεδίων μίας εγγραφής
- Κενή εντολή της Pascal: χρησιμοποιείται για λόγους ευκολίας στη σύνταξη της γλώσσας
- Εντολή pass της Python: δεν κάνει τίποτα, χρησιμοποιείται επίσης για λόγους ευκολίας στη συγγραφή προγραμμάτων.

Περιεχόμενα

Μονάδες Κώδικα

Η δυνατότητα δήλωσης ονομάτων τοπικά σε ένα υποπρόγραμμα (ή γενικότερα μπλοκ) αποτελεί ένα μηχανισμό απόκρυψης πληροφορίας.

Τα τοπικά ονόματα δεν είναι ορατά έξω από το υποπρόγραμμα στο οποίο δηλώνονται. Αυτό επιτρέπει την απόκρυψη λεπτομερειών, τις οποίες δεν είναι απαραίτητο να γνωρίζει όποιος χρησιμοποιεί το υποπρόγραμμα.

Μονάδες Κώδικα

Με αυτό τον τρόπο, οι λεπτομέρειες της υλοποίησης ενός υποπρογράμματος είναι δυνατόν να αλλάξουν, χωρίς αυτό να επηρεάσει τη λειτουργία του συνολικού προγράμματος.

Επίσης η συγγραφή του κώδικα μπορεί να μοιραστεί σε πολλούς προγραμματιστές. Το κάθε υποπρόγραμμα μπορεί να γραφτεί ξεχωριστά, χρησιμοποιώντας τα δικά του ονόματα, ανεξάρτητα από το ποιά ονόματα χρησιμοποιούνται σε άλλα υποπρογράμματα που συνυπάρχουν στο ίδιο πρόγραμμα.

Μονάδες Κώδικα

Ωστόσό, όταν χρησιμοποιείται το μοντέλο τιμών για τις μεταβλητές, τα αντικείμενα τα οποία συνδέονται με τοπικές μεταβλητές είναι αντικείμενα στοίβας, δηλαδή έχουν διάρκεια ζωής που ταυτίζεται με το χρόνο εκτέλεσης του υποπρογράμματος. Τα αντικείμενα αυτά καταστρέφονται όταν ολοκληρωθεί η εκτέλεση του υποπρογράμματος και δεν είναι διαθέσιμα σε επόμενες εκτελέσεις του ίδιου υποπρογράμματος.

Επίσης, όταν χρησιμοποιείτε μοντέλο αναφορών για τις μεταβλητές, τα αντικείμενα που συνδέονται με αυτές ενδέχεται εξακολουθούν να υπάρχουν μετά την ολοκλήρωση της εκτέλεσης του υποπρογράμματος, (καθώς είναι αντικείμενα σωρού), ωστόσο οι συνδέσεις τους με τις τοπικές μεταβλητές (οι οποίες συνδέσεις αποθηκεύονται στη στοίβα) καταστρέφονται.

Μονάδες Κώδικα

Οι στατικές μεταβλητές των υποπρογραμμάτων αποτελούν ένα μηχανισμό ώστε τοπικές μεταβλητές να συνδέονται με στατικά αντικείμενα. Με αυτόν τον τρόπο είναι δυνατόν οι τοπικές μεταβλητές των υποπρογραμμάτων να διατηρούν τις τιμές τους ανάμεσα σε διαφορετικές κλήσεις του ίδιου υποπρογράμματος.

Ωστόσο οι στατικές μεταβλητές είναι ορατές μόνο σε ένα υποπρόγραμμα.

Οι μονάδες κώδικα παρέχουν ένα μηχανισμό ώστε κάποια ονόματα να είναι ορατά μόνο σε ένα συγκεκριμένο σύνολο υποπρογραμμάτων και να διατηρούν τις τιμές τους για διάστημα που υπερβαίνει τη διάρκεια εκτέλεσης των υποπρογραμμάτων αυτών.

Μονάδες Κώδικα

Μία μονάδα κώδικα είναι ένα οριοθετημένο τμήμα του προγράμματος μέσα στο οποίο μπορεί να δηλώνονται μεταβλητές, σταθερές, τύποι, υποπρογράμματα, κλπ, έτσι ώστε:

- Τα ονόματα που δηλώνονται μέσα σέ μία μονάδα κώδικα να είναι ορατά μέσα στη μονάδα αυτή.
- Τα ονόματα που δηλώνονται μέσα σέ μία μονάδα κώδικα να μην είναι ορατά στο τμήμα του προγράμματος που βρίσκεται έξω από τη μονάδα, με εξαίρεση αυτά που εξάγονται με ρητό τρόπο.
- Τα ονόματα που δηλώνονται έξω από μία μονάδα κώδικα να μην είναι ορατά εντός της μονάδας, εκτός αν εισάγονται με ρητό τρόπο (αυτός ο περιορισμός ενδέχεται να μη υπάρχει σε κάποιες γλώσσες που χρησιμοποιούν μονάδες κώδικα).

Μονάδες Κώδικα

Οι μονάδες κώδικα για τις οποίες απαιτείται εισαγωγή ονομάτων ονομάζονται κλειστές. Σε αντίθετη περίπτωση ονομάζονται ανοικτές.

Μονάδες Κώδικα

Παράδειγμα

Στο παρακάτω (σε Modula-2) τα δύο υποπρογράμματα push και pop χρησιμοποιούν έναν πίνακα s για την αποθήκευση της στοίβας και μία μεταβλητή top που δείχνει την κορυφή της στοίβας, τα οποία δεν είναι ορατά έξω από την ενότητα stack.

Μονάδες Κώδικα

```
CONST stack_size = ...                  |   PROCEDURE pop() : element;
TYPE element = ...                      |   BEGIN
...                                     |        IF top = 1 THEN
MODULE stack;                           |           error;
IMPORT element, stack_size;             |        ELSE
EXPORT push, pop;                       |           top := top-1;
TYPE                                    |           RETURN s[top];
     stack_index = [1..stack_size];     |        END;
VAR                                     |   END pop;
     s : ARRAY stack_index of element;  |
     top : stack_index;                 |   BEGIN
                                        |      top :=1;
PROCEDURE error; ...                    |   END stack;
                                        |
PROCEDURE push(elem : element);         |
BEGIN                                   |
     IF top = stack_size THEN           |
        error;                          |
     ELSE                               |
         s[top] := elem;                |
         top := top+1;                  |
     END;                               |
END push;                               |
```

Μονάδες Κώδικα

Απο τα ονόματα που δηλώνονται έξω από τη μονάδα κώδικα stack τα μόνα που είναι ορατά μέσα σε αυτή είναι τα element και stack_size που εισάγονται με την εντολή IMPORT.

Απο τα ονόματα που δηλώνονται μέσα στη μονάδα κώδικα stack το μόνα που είναι ορατά έξω από αυτή είναι τα ονόματα των δύο υποπρογραμμάτων push και pop που εξάγονται με την εντολή EXPORT. Αντίθετα δεν είναι ορατά έξω από τη μονάδα κώδικα stack τα ονόματα s, top και error.

Μονάδες Κώδικα

Η διάρκεια ζωής των αντικειμένων τα οποία δημιουργούνται μέσα σε μία μονάδα κώδικα είναι η ίδια με αυτή που θα είχαν αν καταργούσαμε το όριο της μονάδας κώδικα (δηλαδή αν οι δηλώσεις των αντίστοιχων μεταβλητών βρίσκονταν το σημείο του προγράμματος όπου ορίζεται η μονάδα κώδικα χωρίς όμως να περικλείονται σε αυτή).

Μονάδες Κώδικα

Η εμβέλεια των ονομάτων που εισάγονται σε μία μονάδα κώδικα περιέχει ολόκληρη την μονάδα κώδικα (εκτός αν δημιουργείται τρύπα στην εμβέλεια από κάποια εσωτερική δήλωση).

Τα ονόματα που δηλώνονται μέσα σε μία μονάδα κώδικα και εξάγονται από αυτή, έχουν ως εμβέλεια το τμήμα του προγράμματος που θα αποτελούσε την εμβέλεια ενός ονόματος το οποίο θα δηλωνόταν στο σημείο που βρίσκεται η μονάδα κώδικα.