ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΛΙΣΤΕΣ
4.4 Δείκτες και Δυναμική Δέσμευση/ Αποδέσμευση Μνήμης στη C

Εισαγωγή

- Από τον ορισμό της λίστας ως αφηρημένος τύπος δεδομένων προκύπτει ότι θεωρητικά μπορεί να εισαχθεί απεριόριστο πλήθος στοιχείων σ' αυτήν.
- Κατά συνέπεια, οποιαδήποτε υλοποίηση λίστας που χρησιμοποιεί πίνακα για την αποθήκευση των στοιχείων της δεν θα είναι πιστή αναπαράσταση λίστας, γιατί ένας πίνακας έχει σταθερό μέγεθος το οποίο δεν μπορεί να αλλάξει από τη στιγμή που θα ορισθεί και μετά.
- Μια πιο πιστή υλοποίηση λίστας θα πρέπει να έχει τη δυνατότητα δέσμευσης και αποδέσμευσης θέσεων μνήμης για τους κόμβους δυναμικά κατά τη διάρκεια εκτέλεσης του προγράμματος, χωρίς να χρειάζεται να είναι προκαθορισμένο το όριο μεγέθους της δεξαμενής κόμβων πριν την εκτέλεση του προγράμματος ή γενικώς πριν την εισαγωγή στοιχείου.
- Η δυνατότητα αυτή υπάρχει στην C και παρέχεται από τις προκαθορισμένες **συναρτήσεις malloc** και **free** (ορίζονται στην *stdilb.h*) που χρησιμοποιούνται σε συνδυασμό με τους τύπους δεδομένων.

Η συνάρτηση malloc

Η συνάρτηση **malloc** χρησιμοποιείται για να δεσμεύει θέσεις μνήμης κατά τη διάρκεια εκτέλεσης του προγράμματος.

Όταν κληθεί, επιστρέφει τη διεύθυνση μιας θέσης μνήμης στην οποία μπορεί να αποθηκευτεί μια τιμή.

Για να αναφερόμαστε σ' αυτήν τη θέση μνήμης και να μπορούμε να αποθηκεύουμε δεδομένα και να τα ανακτούμε από αυτήν, χρησιμοποιούμε ένα ειδικό είδος μεταβλητής, που ονομάζεται **μεταβλητή δείκτης** ή απλά **δείκτης** και η τιμή της είναι η διεύθυνση μιας θέσης μνήμης.

Δήλωση μεταβλητής δείκτη

Ο τύπος μιας μεταβλητής δείκτη που χρησιμοποιείται για αναφορά στη θέση μνήμης όπου είναι αποθηκευμένη κάποια τιμή πρέπει να οριστεί ως:

type-identifier* identifier ή **type-identifier *identifier**

όπου type-identifier είναι ο τύπος δεδομένων της τιμής που αποθηκεύεται.

Πχ int* p; ή int *p;

Ο δείκτης είναι δεσμευμένος σε αυτόν τον τύπο δεδομένων και δεν γίνεται να αποθηκευτούν δεδομένα άλλων τύπων στη θέση μνήμης στην οποία αναφέρεται.

Παράδειγμα μεταβλητής δείκτη

Για παράδειγμα, αν τα δεδομένα είναι ακέραιοι, τότε ένας δείκτης σε μια θέση μνήμης, που μπορεί να χρησιμοποιηθεί για να αποθηκευτεί ένας ακέραιος, μπορεί να δηλωθεί ως εξής:

```
int *p1;
```

Αυτή η μεταβλητή δείκτη p1 είναι δεσμευμένη στον τύπο int και μπορεί να χρησιμοποιηθεί μόνο για να αναφέρεται σε θέσεις μνήμης στις οποίες μπορούν να αποθηκευτούν τιμές αυτού του τύπου.

Η p1 αναφέρεται ως *μεταβλητή δείκτης προς int (ακέραιο)* ή *δείκτης προς int*.

Παράδειγμα μεταβλητής δείκτη

Η συνάρτηση malloc μπορεί να χρησιμοποιηθεί για την απόκτηση μιας τέτοιας θέσης μνήμη κατά την εκτέλεση του προγράμματος. Η κλήση της συνάρτησης malloc είναι της μορφής:

```
p1 = malloc(sizeof(int));
```

και καταχωρεί τη διεύθυνση μιας θέσης μνήμης στον δείκτη p1.

Επομένως, η εντολή

```
p1 = malloc(sizeof(int));
```

καταχωρεί μια διεύθυνση μνήμης, π.χ. 946, στη μεταβλητή p1.

Δηλαδή ο δείκτης p1 κρατά την τιμή της θέσης μνήμης στην οποία είναι αποθηκευμένος ένας ακέραιος, και όχι ο ίδιος ο ακέραιος.

Μάλιστα μπορεί αυτή η θέση μνήμης να είναι η θέση της πρώτης λέξης σε ένα μπλοκ διαδοχικών θέσεων μνήμης, όπως όταν ο δείκτης δείχνει σε εγγραφή, πίνακα κτλ

Ανώνυμες/δυναμικές μεταβλητές

Εφόσον η προαναφερθείσα περιοχή μνήμης μπορεί να χρησιμοποιηθεί για να αποθηκεύει τιμές τύπου int, είναι μια μεταβλητή, η οποία όμως δεν έχει όνομα.

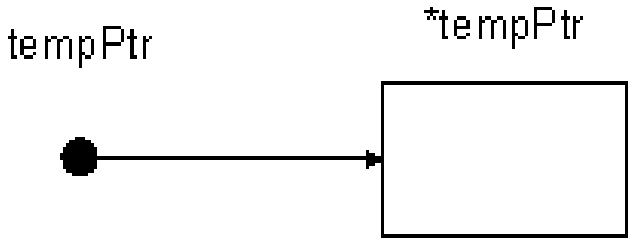
Τέτοιες μεταβλητές ονομάζονται **ανώνυμες μεταβλητές** (**anonymous variables**) και οι δείκτες μπορούν να δείχνουν σε ανώνυμες μεταβλητές.

Επειδή αυτές οι μεταβλητές αρχίζουν να υπάρχουν κατά τη διάρκεια εκτέλεσης του προγράμματος και μπορεί να πάψουν να υπάρχουν αργότερα, ονομάζονται και **δυναμικές μεταβλητές** (**dynamic variables**).

Η σταθερά δείκτη NULL

Κάθε κλήση της malloc αποκτά μια νέα θέση μνήμης και καταχωρεί τη διεύθυνσή της στον συγκεκριμένο δείκτη.

Έτσι, λοιπόν, αν η TempPtr είναι επίσης τύπου int, τότε η εντολή




```
tempPtr = malloc(sizeof(int));
```

αποκτά μια νέα θέση μνήμης στην οποία δείχνει ο TempPtr:

Η C παρέχει την **ειδική σταθερά δείκτη NULL** για την περίπτωση που θέλουμε να καταχωρήσουμε μια τιμή σε μια μεταβλητή δείκτη που δεν δείχνει σε καμιά θέση μνήμης.

Αυτή η τιμή μπορεί να καταχωρηθεί σε δείκτη οποιουδήποτε τύπου με μια εντολή της μορφής: **pointer = NULL;**

Όπως είναι αναμενόμενο ένας τέτοιος δείκτης θα συμβολίζεται απλά με μια τελεία: **pointer •**

Λειτουργίες που εκτελούνται με τιμές δεικτών

Οι τιμές των δεικτών είναι διευθύνσεις μνήμης, οι λειτουργίες που μπορούν να εκτελεστούν σ' αυτές :

- ανάθεση τιμής
- σύγκριση με τους σχεσιακούς τελεστές == και !=.

Ανάθεση τιμής

Αν οι δείκτες ptr1 και ptr2 και είναι δεσμευμένοι στον ίδιο τύπο, τότε μια εντολή ανάθεσης τιμής της μορφής:

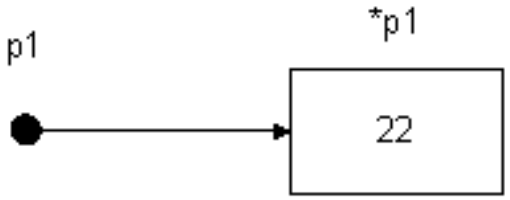
ptr1 = ptr2;

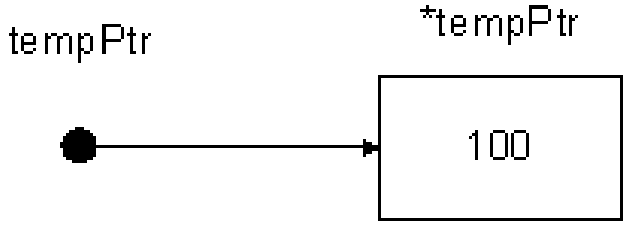
καταχωρεί την τιμή του ptr2 στον ptr1, οπότε και οι δυο δείχνουν στην ίδια θέση μνήμης.

Η θέση στην οποία έδειχνε πρωτύτερα ο ptr1 δεν είναι πλέον προσπελάσιμη εκτός και αν κάποιος άλλος δείκτης δείχνει σ' αυτήν.

Ανάθεση τιμής

Παράδειγμα: έστω ότι δυο δείκτες, οι p1 και tempPtr, είναι δηλωμένοι τύπου int και δείχνουν σε δυο θέσεις μνήμης που περιέχουν τις τιμές 22 και 100, αντίστοιχα.

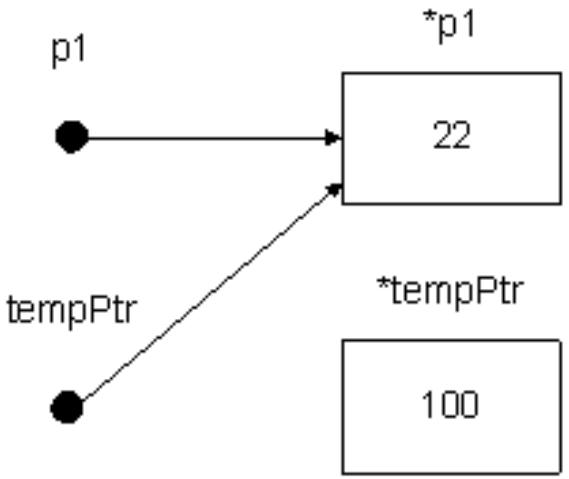




Μια εντολή ανάθεσης

tempPtr = p1 ;

αναθέτει την τιμή της p1 στην tempPtr οπότε η tempPtr δείχνει στην θέση μνήμης που δείχνει και η p1.



Τώρα δεν έχουμε πλέον πρόσβαση στη μεταβλητή *tempPtr, δηλαδή στο 100, εκτός και αν κάποιος άλλος δείκτης δείχνει εκεί.

Σχεσιακοί τελεστές

Οι σχεσιακοί τελεστές **==** και **!=** μπορούν να χρησιμοποιηθούν για να συγκριθούν δυο δείκτες δεσμευμένοι στον ίδιο τύπο δεδομένων για να καθοριστεί αν και οι δύο δείχνουν στην ίδια θέση μνήμης ή αν είναι και οι δύο μηδενικοί.

Επομένως, η boolean έκφραση

p1 == tempPtr

είναι έγκυρη και είναι αληθής αν και μόνο αν οι p1 και tempPtr δείχνουν στην ίδια θέση μνήμης ή έχουν τιμή NULL.

Επίσης, η έκφραση

p1 != NULL

είναι κι αυτή μια έγκυρη boolean έκφραση.

Δείκτες ως παράμετροι/τιμές συνάρτησης

Οι δείκτες μπορούν να χρησιμοποιηθούν και ως παράμετροι σε συναρτήσεις και διαδικασίες.

Οι παράμετροι μπορεί να είναι είτε τιμές είτε μεταβλητές, όμως οι αντίστοιχοι δείκτες τυπικοί παράμετροι πρέπει να είναι δεσμευμένοι στον ίδιο τύπο. Ακόμα και η τιμή μιας συνάρτησης μπορεί να είναι ένας δείκτης.

Για να αναφερθούμε στην τιμή που είναι αποθηκευμένη στη θέση μνήμης όπου δείχνει ένας δείκτης, χρησιμοποιούμε το σύμβολα * πριν από το όνομα του δείκτη:

*pointer

Ανάθεση τιμής

Έστω, για παράδειγμα, ότι έχουμε δηλώσει ένα δείκτη p1 τύπου int, ο οποίος δείχνει στη θέση μνήμης 2645 και θέλουμε να εκχωρήσουμε τη τιμή 22.

Τότε:

***p1 = 22**

Ένας μηδενικός ή μη ορισμένος δείκτης, όμως, δεν αναφέρεται σε καμιά θέση μνήμης, οπότε οποιαδήποτε προσπάθεια χρησιμοποίησής του είναι σφάλμα.

Ανάθεση τιμής

Αν οι p1 και tempPtr είναι μη μηδενικοί δείκτες τύπου int με

*p1=22 και

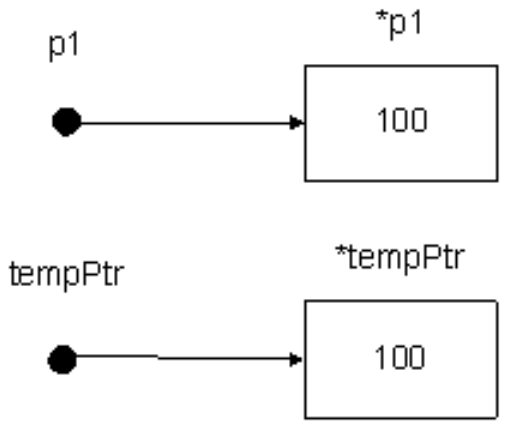
*tempPtr =100,

τότε η εντολή

***p1 = *tempPtr;**

είναι μια έγκυρη εντολή ανάθεσης τιμής, αφού και οι δυο δυναμικές μεταβλητές *p1 και *tempPtr υπάρχουν και είναι του ίδιου τύπου.

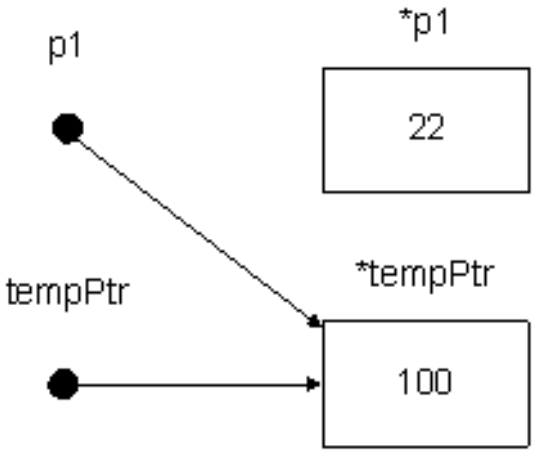
Η εντολή αυτή αντιγράφει το περιεχόμενο της θέσης μνήμης στην οποία δείχνει η p1 στην θέση μνήμης στην οποία δείχνει η tempPtr



Η παραπάνω εντολή διαφέρει πολύ από την

p1 = tempPtr;

η οποία δίνει την τιμή της p1 στην tempPtr με αποτέλεσμα να δείχνουν και οι δυο στην ίδια θέση μνήμης:



Εκφράσεις σύγκρισης

Επίσης, διαφορετική είναι η τιμή των boolean εκφράσεων σύγκρισης

p1 == tempPtr;

και

***p1 == *tempPtr;**

Η

πρώτη έκφραση είναι αληθής αν και μόνο αν **οι δείκτες p1 και tempPtr δείχνουν στην ίδια θέση μνήμης**,

ενώ η

δεύτερη έκφραση **συγκρίνει τον ακέραιο** που είναι αποθηκευμένος στη θέση μνήμης στην οποία δείχνει ο p1 με τον **ακέραιο** που είναι αποθηκευμένος στη θέση μνήμης στην οποία δείχνει ο tempPtr.

Εκφράσεις σύγκρισης

Προφανώς:

- αν η **p1 == tempPtr;** είναι αληθής και
- οι δυο δείκτες **δεν είναι μηδενικοί**,

τότε και η ***p1 == *tempPtr;** είναι αληθής.

Το αντίστροφο όμως δεν ισχύει.

Δηλαδή, αν ***p1 == *tempPtr;** δεν σημαίνει ότι οι p1 και tempPtr δείχνουν στην ίδια θέση μνήμης, αλλά απλά ότι τα περιεχόμενά τους είναι ίδια.

Τέλος, αν κάποιος από τους δύο δείκτες είναι μηδενικός, τότε:

- η έκφραση p1 == tempPtr είναι έγκυρη,
- όχι όμως και η *p1 == *tempPtr η οποία δεν είναι έγκυρη

Αποδέσμευση θέσης μνήμης

Αν η θέση μνήμης στην οποία δείχνει ένας δείκτης δεν χρειάζεται πλέον, μπορεί να ελευθερωθεί και να γίνει διαθέσιμη για να δεσμευτεί αργότερα με κλήση της διαδικασίας free ως εξής:

```
free(pointer);
```

Η διαδικασία αυτή ελευθερώνει τη θέση μνήμης στην οποία δείχνει η pointer και αφήνει τη μεταβλητή pointer μη ορισμένη.