Προγραμματισμός Υπολογιστών

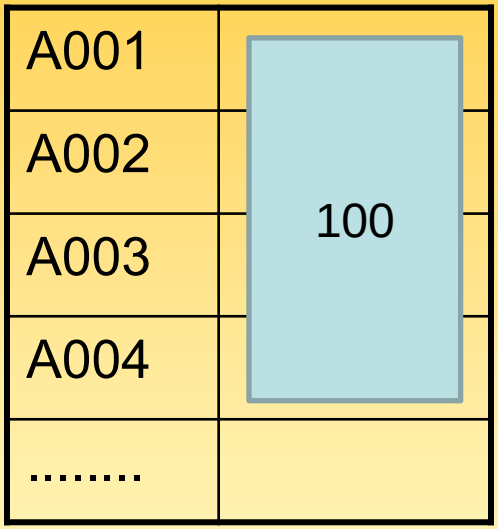
Εισαγωγή στους δείκτες

Νικόλαος Ζ. Ζάχαρης
Καθηγητής

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Δείκτες (pointers)

Η μνήμη του υπολογιστή αποτελείται από διαδοχικές θέσεις αποθήκευσης. Η κάθε θέση προσδιορίζεται από μία μοναδική διεύθυνση. Όταν δηλώνουμε μια μεταβλητή, τότε δεσμεύουμε μια ή περισσότερες θέσεις μνήμης για την αποθήκευση της τιμής της μεταβλητής.



Η εκτέλεση της εντολής int x; θα δεσμεύσει θέσεις μνήμης για να αποθηκεύσει τα περιεχόμενα της μεταβλητής. Ας θεωρήσουμε ότι δεσμευτήκαν 4 bytes τα οποία αρχίζουν στη διεύθυνση A001. Επίσης κατά την εκτέλεση της εφαρμογής διατηρείται η αντιστοιχία, δηλαδή η θέση μνήμης της κάθε μεταβλητής ώστε όταν διαβάζουμε ή αναθέτουμε τιμή στη μεταβλητή π.χ. x = 100 να γνωρίζει σε ποία θέση μνήμης να αποδώσει την τιμή.

Πόσες θέσεις μνήμης θα δεσμεύσει ο μεταγλωττιστής για την μεταβλητή εξαρτάται από το τύπο της. Και αυτό είναι το μεγάλο πλεονέκτημα των μεταβλητών, δηλαδή ότι χρησιμοποιούμε φιλικά ονόματα π.χ. x, foros αντί των διευθύνσεων των θέσεων μνήμης. Στην πραγματικότητα, κατά την εκτέλεση της εφαρμογής, με την δήλωση int x δημιουργούμε ένα “ψευδώνυμο” για την πραγματική διεύθυνση στη μνήμη του υπολογιστή.

Οι δείκτες είναι μεταβλητές των οποίων οι τιμές είναι διευθύνσεις μνήμης και μπορεί να δείχνουν σε θέσεις μνήμης άλλων μεταβλητών. Η δήλωση ενός δείκτη, γίνεται ως εξής :

Τύπος_Δεδομένων * Ονομα_Μεταβλητής

Έτσι λοιπόν η δήλωση ενός δείκτη, ο οποίος θα δείχνει στην διεύθυνση της μεταβλητής x θα γίνει σε δύο στάδια : a) θα δηλώσουμε την μεταβλητή **p** ως εξής **int *p;** δηλαδή ότι η p είναι ένας δείκτης σε μια διεύθυνση μνήμης που το περιεχόμενο της είναι μια ακέραια τιμή. Ότι ισχύει για τις άλλες μεταβλητές ισχύει και για τους δείκτες, δηλαδή θα δεσμευτούν θέσεις μνήμης. Ανεξαρτήτως του τύπου του δείκτη θα δεσμευτούν τόσα bytes όσα χρειάζονται για έναν ακέραιο αριθμό, επειδή οι διευθύνσεις είναι ακέραιοι αριθμοί. Έστω τα bytes που ξεκινούν στη διεύθυνση A006. β) Εν συνεχεία αναθέτουμε σαν τιμή στη μεταβλητή p την διεύθυνση της x με την εντολή **p = &x;**

A001	100
…….	
…….	
A006	A001
……..	

η οποία θα αναθέσει την θέση μνήμης A001, που είναι η διεύθυνση της x, σαν τιμή στην p. Ο τελεστής διεύθυνσης (address operator), το &, έχει σαν λειτουργία να επιστρέψει τη διεύθυνση της μεταβλητής που βρίσκεται στα δεξιά του. Έτσι λοιπόν η τιμή του δείκτη ισούται με την διεύθυνση της μεταβλητής x.

Στο παράδειγμα μας στα δεξιά του τελεστή είναι το x οπότε όλη μαζί η έκφραση &x επιστρέφει A001.

Ο τελεστής αποαναφεροποίησης (dereferencing operator), το *, αναφέρεται στα περιεχόμενα της διεύθυνσης στην οποία δείχνει ο δείκτης. Δηλαδή αν εκτυπώσω το *p τότε θα εκτυπωθεί το 100, που είναι η τιμή του x. Από την άλλη πλευρά αν αναθέσω την τιμή 200 στο *p, στην ουσία αναθέτω το 200 στο x.

```
int main(int argc, char *argv[]) {
  int x = 5, *p;

    p = &x;
    printf("x = %d\n",x);
    printf("&x = %p\n", &x);
    printf("p = %p\n");
    printf("*p = %d\n",*p);

    *p = *p + 10;

    printf("x = %d\n",x);
    return 0;
}
```

Παράδειγμα χρήσης Δεικτών

Το %p σημαίνει διεύθυνση δείκτη και αν στη θέση του χρησιμοποιούσαμε το %d τότε θα εμφανιζόταν το 6422184 που είναι η δεκαδική αναπαράσταση του δεκαεξαδικού 0061FEA8.

```
x = 5
&x = 0061FEA8
p = 0061FEA8
*p = 5
x = 15
```

Στο διπλανό πρόγραμμα δηλώνουμε ότι το x είναι ένας ακέραιος αριθμός και το p είναι ένας δείκτης σε ένα ακέραιο αριθμό. Εν συνεχεία αποδίδουμε τη διεύθυνση του x σαν τιμή στη μεταβλητή p.

Η εκτύπωση της διεύθυνσης του x όσο και της τιμής της μεταβλητής p δείχνουν το ίδιο μήνυμα αφού υπάρχει πλήρη ταύτιση.

```
printf("&x = %p\n", &x);
printf("p = %p\n");
```

```
&x = 0061FEA8
p = 0061FEA8
```

```
printf("*p = %d\n",*p);
```

Η εκτύπωση εν συνεχεία της τιμής που δείχνει η μεταβλητή p στην ουσία εκτυπώνει τη τιμή του x

```
*p = 5
```

Η αύξηση κατά 10 της τιμής που δείχνει η μεταβλητή p στην ουσία αυξάνει κατά 10 τη τιμή του x

```
*p = *p + 10;
```

Η εμφάνιση της τιμής του x επιβεβαιώνει την αλλαγή που προηγήθηκε :

```
printf("x = %d\n",x);
```

```
x = 15
```

Η χρήση των δεικτών ενώ αρχικά φαίνεται ότι είναι άνευ ουσίας στην πραγματικότητα όχι μόνο αυξάνει τις δυνατότητες στο προγραμματισμό αλλά πραγματοποιούνται λειτουργίες που διαφορετικά θα ήταν είτε αδύνατες είτε πολύ χρονοβόρες.

Σχέση πινάκων και δεικτών(1-2)

Ένας πίνακας ΔΕΝ είναι δείκτης αλλά συνεχόμενες θέσεις στη μνήμη του ΗΥ. Όμως το όνομα ενός πίνακα κατά την εκτέλεση ενός προγράμματος μετατρέπεται σε ένα δείκτη στο πρώτο του στοιχείο του.

```
int main(int argc, char *argv[]) {
    int i, *p, pin[3] = {17, 20, 14};

    printf("pin = %p\n", pin);
    printf("&pin[0] = %p\n", &pin[0]);
    p = pin;
    printf("p = %p\n", p);
    printf("*p = %d\n", *p);
    p = p + 1;
    printf("*p = %d\n", *p);
    p = p + 1;
    printf("*p = %d\n", *p);
    p = p - 2;
    printf("*p = %d\n", *p);
    for(i = 0; i < 3; i++)  {
        printf("*p = %d  ", *(p+i));
        printf("*p = %d\n", p[i]);
    }
    return 0;
}
```

```
pin = 0061FE9C
&pin[0] = 0061FE9C
p = 0061FE9C
*p = 17
*p = 20
*p = 14
*p = 17
*p = 17   *p = 17
*p = 20   *p = 20
*p = 14   *p = 14
```

Το όνομα του πίνακα, η διεύθυνση του πρώτου στοιχείου και η τιμή του p μετά την εκτέλεση της εντολής p = pin, όλα αναφέρονται στην ίδια διεύθυνση.

Η εντολή p = p + 1 δεν έχει σαν αποτέλεσμα το p να δείχνει στο επόμενο byte αλλά στον επόμενο ακέραιο αριθμό (p = p+1* sizeof(int)) επειδή ο p είναι δείκτης σε ακέραιο αριθμό. Ο τελεστής [] ορίζεται με όρους δεικτών και για αυτό το λόγο επιτρέπεται να χρησιμοποιείται σε πίνακες και σε δείκτες, και σημαίνει η τιμή που υπάρχει στη διεύθυνση p+(i * sizeof(int)) ή σε pin + (i * sizeof(int))

Σχέση πινάκων και δεικτών (2-2)

```
int main(int argc, char *argv[]) {
    int *p1, pin[3] = {17, 20, 4};
    int *p2, i;

    p1 = pin;

    p2 = p1;

    for(i = 0; i < 3; i++) {
        printf("%d\t", *(p2+i));
    }

    return 0;
}
```

```
17      20      4
```

Ένας δείκτης μπορεί να πάρει σαν τιμή έναν άλλο δείκτη ή τη διεύθυνση ενός πίνακα (είτε με το όνομα του είτε με τη διεύθυνση του πρώτου του στοιχείου).

Ένας πίνακας **δεν** μπορεί να πάρει σαν τιμή έναν δείκτη ούτε τη διεύθυνση ενός άλλου πίνακα.

```
int main(int argc, char *argv[]) {
    int *p, pin1[3] = {17, 20, 4};
    int pin2 [4] = {45, 86, 90, 1};
    int i;

    p = pin2;

    for(i = 0; i < 3; i++) {
        printf("%d\t", *(p+i));
    }

    pin1 = pin2;
    pin1 = p;

    return 0;
```

Compile Log | Debug | Find Results | Close

Message

In function 'main':

[Error] assignment to expression with array type

[Error] assignment to expression with array type