Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κεντρικής Μακεδονίας - Σέρρες
Τμήμα Μηχανικών Πληροφορικής

Προγραμματισμός ΙΙ (Θ)

Δρ. Δημήτρης Βαρσάμης
Επίκουρος Καθηγητής

Μάρτιος 2017

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΙΙ (Θ)

1. Δείκτες
   - Εισαγωγή
   - Δείκτες και Μνήμη
   - Δήλωση μεταβλητής δείκτης
   - Αρχικοποίηση δεικτών
   - Δείκτες και πίνακες

2. Παραδείγματα
   - Παράδειγμα 1: Δείκτες και συναρτήσεις
   - Παράδειγμα 2: Δείκτες και πίνακες
   - Παράδειγμα 3: Δείκτες και πίνακες

Δείκτες - Εισαγωγή Ι

- Σε αυτήν την ενότητα θα μελετήσουμε ένα από τα ισχυρότερα χαρακτηριστικά του προγραμματισμού, τους δείκτες (pointers).
- Οι δείκτες ανήκουν στα πιο πολύπλοκα κεφάλαια της C, δίνουν όμως την δυνατότητα στα προγράμματα να δημιουργούν και να χειρίζονται δυναμικές δομές δεδομένων, καθώς και να προσομοιώνουν την κλήση στοιχείων μέσω αναφοράς.
- Η κλήση μέσω αναφοράς και όχι μέσω αξίας (τιμής) μας δίνει τη δυνατότητα να διαχειριστούμε μεγάλα δεδομένα χωρίς να χρειάζεται να δημιουργούμε αντίγραφα δεδομένων για κάθε κλήση μέσω αξίας (τιμής). Έτσι έχουμε εξοικονόμηση μνήμης.
- Προσοχή, η κλήση μέσω αναφοράς δίνει τη δυνατότητα να τροποποιηθούν τα περιεχόμενα της μεταβλητής που δείχνει ο δείκτης.

Δείκτες και Μνήμη I

- Οι **δείκτες (pointers)** είναι απλές μεταβλητές.
- Έχουν όμως μία ιδιαιτερότητα, δέχονται για τιμές την **διεύθυνση (address) μνήμης** ενός στοιχείου και με αυτόν τον τρόπο έμμεσα έχουν πρόσβαση στην τιμή του.
- Η αναφορά σε μια τιμή μέσω ενός δείκτη ονομάζεται **έμμεση διευθυνσιοδότηση(indirection)**.

Δείκτες και Μνήμη I

Ας δούμε αναλυτικά πώς χρησιμοποιούμε τους δείκτες, ας θυμηθούμε κάποια βασικά πράγματα για την μνήμη τα οποία θα μας βοηθήσουν στην κατανόηση τους.

- Η **μνήμη** είναι διατεταγμένη με τέτοιο τρόπο ώστε κάθε στοιχείο της να έχει την δυνατότητα να οριστεί μονοσήμαντα από έναν αριθμό.
- Αυτός ο αριθμός είναι η **τιμή** που παίρνει σε κάθε περίπτωση ο **δείκτης**.
- Ας θυμηθούμε όμως και πόσα **bytes** μνήμης καταλαμβάνουν οι διάφοροι τύποι δεδομένων που χρησιμοποιούμε.

Δείκτες και Μνήμη I

Τύποι Δεδομένων

Τύπος	Bytes
`char`	1
`int`	4
`short`	2
`long`	4
`float`	4
`double`	8
`long double`	16

Δείκτες και Μνήμη I

Μνήμη

Διευθυνση	**Περιεχόμενο**
900:	x, 10
904:	y, 20
908:	z, 3.14
916:	m, 4
920:	

Δήλωση μεταβλητής δείκτης I

Για να **δηλώσουμε** έναν δείκτη χρησιμοποιούμε τον τελεστή (*) ο οποίος καλείται ως **τελεστής έμμεσης διευθυνσιοδότησης** ή **τελεστής προσπέλασης της θέσης αναφοράς**:

```
data_type *pointer_name;
```

όπου

- `data_type` ο τύπος της μεταβλητής αποθηκεύεται στη θέση που δείχνει ο δείκτης
- * προσδιορίζει ότι δηλώνεται μία μεταβλητή δείκτη. Ο τελεστής * χρησιμοποιείται στο όνομα της μεταβλητής και όχι στον τύπο της μεταβλητής.
- `pointer_name` το όνομα της μεταβλητής δείκτη. Καλή προγραμματιστική πρακτική: το πρώτο γράμμα του ονόματος να είναι πάντοτε p

Δήλωση μεταβλητής δείκτης I

Γιατί πρέπει να δηλωθεί ο τύπος της κανονικής μεταβλητής;

- Γιατί όταν δηλώνεται μία (κανονική) μεταβλητή, δεσμεύεται συγκεκριμένη μνήμη, π.χ. 8 bytes για double, 4 bytes για int.
- Ο δείκτης αναφέρεται σε μία διεύθυνση, στην οποία αποθηκεύεται η τιμή μίας κανονικής μεταβλητής.
- Ο δείκτης χρησιμοποιείται για να γίνεται έμμεση αναφορά σ' αυτήν την τιμή. Έτσι, πρέπει να γνωρίζουμε πόση ακριβώς μνήμη καταλαμβάνει αυτή η τιμή.

Δήλωση μεταβλητής δείκτης I

Πόση μνήμη καταλαμβάνει η ίδια η μεταβλητή δείκτη;

- Γιατί όταν δηλώνεται μία (κανονική) μεταβλητή, δεσμεύεται συγκεκριμένη μνήμη, π.χ. 8 bytes για double, 4 bytes για int.

Γιατί είναι απαραίτητος ο αστερίσκος;

- Διότι προσδιορίζει ότι δηλώνεται μία μεταβλητή με δείκτη και όχι μία κανονική μεταβλητή.
- ΠΡΟΣΟΧΗ : Αν και η μεταβλητή με δείκτη περιέχει μία διεύθυνση (ακέραιος αριθμός), ΔΕΝ ΕΙΝΑΙ ΙΔΙΑ με μία κανονική ακέραια μεταβλητή. Ο μεταγλωττιστής γνωρίζει ότι η τιμή της μεταβλητής με δείκτη είναι μία συγκεκριμένη διεύθυνση μνήμης, σε αντιδιαστολή με την "κανονική" ακέραια τιμή.

Δήλωση μεταβλητής δείκτης I

Ο αστερίσκος συνδέεται με το όνομα κι όχι με τον τύπο:

- `int *pcount;`
  δείκτης σε ακεραίους, με ονομασία pcount
- `int *pcount, *pnum;`
  δείκτες σε ακεραίους, με ονομασίες pcount και pnum
- `int *pcount, number;`
  ένας δείκτης σε ακέραιο, με ονομασία pcount και ένας ακέραιος με ονομασία number

Δήλωση μεταβλητής δείκτης I

Πώς επιλέγεται το όνομα ενός δείκτη;

- Οι ίδιες συμβάσεις που ισχύουν στις κανονικές μεταβλητές.
- Ωστόσο, συνήθως ο αρχικός χαρακτήρας του ονόματος δείκτη είναι το `p`, έτσι ώστε το πρόγραμμα να καθίσταται περισσότερο ευανάγνωστο, καθώς με τον πρώτο χαρακτήρα φαίνεται εάν μία μεταβλητή είναι δείκτης ή όχι. Εναλλακτικά, μπορούμε να προσθέτουμε την κατάληξη `_ptr` ή `Ptr`.

Παράδειγμα:

- `int *pcount, *count_ptr;` δείκτες σε ακεραίους
- `char *pword, *wordPtr;` δείκτες σε χαρακτήρες

Αρχικοποίηση δεικτών I

Οι δείκτες πρέπει πάντα να αποκτούν αρχικές τιμές. Αυτό μπορεί να συμβεί είτε κατά την δήλωση τους είτε μέσω μίας πρότασης εκχώρησης τιμής. Ας δούμε λοιπόν πως τους αποδίδουμε τιμές.
Για να **εκχωρήσουμε** κάποια **τιμή** χρησιμοποιούμε τον τελεστή '**&**' γνωστός και ως **τελεστής διεύθυνσης**:

```
pointer_name = &variable_name;
```

Αρχικοποίηση δεικτών I

Τι συμβαίνει όμως στην μνήμη όταν δηλώνουμε μια μεταβλητή τύπου δείκτη σε κάποιο ακέραιο αριθμό;
Ας κατασκευάσουμε λοιπόν ένα πρόγραμμα και ας δούμε τι δεδομένα θα περάσει στην μνήμη.
Το πρόγραμμα θα πρέπει να έχει:

- έναν ακέραιο `x` με την τιμή `23`,
- έναν δείκτη στον ακέραιο `x`

Και θα εμφανίζει:

- τις τιμές του ακέραιου και του δείκτη
- και τις τιμές των θέσεων μνήμης.

Αρχικοποίηση δεικτών II

```
int x;
int *xPtr;
x=23;
xPtr = &x;
printf("The address of x is: %d",&x);
printf("\n The value of xPtr is: %d",xPtr);
printf("\n The value of x is: %d",x);
printf("\n The value of *xPtr is: %d", *xPtr);
```

Αρχικοποίηση δεικτών I

Μνήμη

Διεύθυνση	**Περιεχόμενο**
900:	x, junk
904:	
908:	
912:	
916:	

Αρχικοποίηση δεικτών I

Μνήμη

Διευθυνση	**Περιεχόμενο**
900:	x, junk
904:	xPtr, junk
908:	
912:	
916:	

Αρχικοποίηση δεικτών I

Μνήμη

Διευθυνση	**Περιεχόμενο**
900:	x, 23
904:	xPtr, junk
908:	
912:	
916:	

Αρχικοποίηση δεικτών I

Μνήμη

Διευθυνση	**Περιεχόμενο**
900:	x, 23
904:	xPtr, 900
908:	
912:	
916:	

Αρχικοποίηση δεικτών I

Έχοντας λάβει υπόψιν το περιεχόμενο της μνήμης, το πρόγραμμα θα μας δώσει τα παραπάνω αποτελέσματα.

The address of 'x' is	900
The value of 'xPtr' is	900
The value of 'x' is	23
The value of '*xPtr'	23

Παρατηρούμε λοιπόν ότι όταν θέλουμε να αναφερθούμε στην τιμή του δείκτη αναφερόμαστε απλώς με το όνομα της μεταβλητής, όταν **όμως** θέλουμε την τιμή της μεταβλητής που αναφέρετε ο δείκτης χρησιμοποιούμε πάλι τον τελεστή έμμεσης διευθυνσιοδότησης (*).

Ανάθεση τιμής στην μεταβλητή που δείχνει ο δείκτης I

```
*pcount = 20;
```

- Ο αστερίσκος ονομάζεται τελεστής περιεχομένου (dereferencing operator).
- Διαβάζεται "στη διεύθυνση".
- Χρησιμοποιείται για να προσπελαστούν τα περιεχόμενα της θέσης μνήμης στην οποία δείχνει ο δείκτης.
- Δε θα πρέπει να συγχέεται με τον αστερίσκο της δήλωσης δείκτη.

Παράδειγμα

- Να δημιουργηθεί πρόγραμμα το οποίο θα ορίζει δυο ακεραίους και η συνάρτηση θα αντιμεταθέτει τις τιμές των δυο ακεραίων.
- Με δήλωση δεικτών και αρχικοποίηση των τιμών.
- Με δήλωση δεικτών και απευθείας δήλωση στην συνάρτηση.

Παράδειγμα I

```
#include <stdio.h>
#include <stdlib.h>

void swap (int *pa, int *pb);

int main(){
    int x=10, y=25;
    int *px, *py;
    px = &x;
    py = &y;
    printf("The value of x is: %d the value of y is: %d"
        , x,y);
    swap(px,py);
    printf("The value of x is: %d the value of y is: %d"
        , x,y);
    return 0;
}
```

Παράδειγμα II

```
void swap (int *pa, int *pb) {
    int temp;
    temp = *pa;
    *pa = *pb;
    *pb = temp;
}
```

Με δήλωση δεικτών και αρχικοποίηση των τιμών.

Παράδειγμα I

```
#include <stdio.h>
#include <stdlib.h>

void swap (int *pa, int *pb);

int main(){
    int x=10, y=25;
    int *px, *py;
    printf("The value of x is: %d the value of y is: %d"
        , x,y);
    swap(&x, &y);
    printf("The value of x is: %d the value of y is: %d"
        , x,y);
    return 0;
}

void swap (int *pa, int *pb) {
```

Παράδειγμα II

```
    int temp;
    temp = *pa;
    *pa = *pb;
    *pb = temp;
}
```

Με δήλωση δεικτών και απευθείας δήλωση στην συνάρτηση.

Δείκτες και πίνακες I

Στην C, οι δείκτες και οι πίνακες μπορούν να συνδυαστούν και να μας προσφέρουν ακόμα περισσότερες λειτουργίες. Αυτές οι λειτουργίες αφορούν την ταχύτητα του κώδικα, την ποιότητα του προγράμματος που κατασκευάσαμε και επίσης μας παρέχουν μεγάλη ευκολία. Εάν για παράδειγμα θα θέλαμε να εκχωρήσουμε σε έναν δείκτη την διεύθυνση ενός στοιχείου του πίνακα, αλλά δεν δηλώσουμε που είναι η θέση του, τότε ο δείκτης θα πάρει αυτόματα την διεύθυνση του πρώτου στοιχείου.
Ας δούμε ένα απλό παράδειγμα χρήσης πινάκων και δεικτών:

```
int table[20];
int *pointer;
pointer = table[2];
```

Δείκτες και πίνακες I

- Σε αυτό το μικρό παράδειγμα κάναμε δήλωση ενός πίνακα και ενός δείκτη και έπειτα εκχωρήσαμε στον δείκτη την διεύθυνση του τρίτου (3) στοιχείου του πίνακα.
- Οι δείκτες όμως δεν έχουν μόνο την δυνατότητα να αποθηκεύουν τις διευθύνσεις των στοιχείων των πινάκων όπως είδαμε παραπάνω.
- Μας παρέχουν επίσης την δυνατότητα να δημιουργήσουμε νέους πίνακες όπου τα στοιχεία τους θα είναι δείκτες.

Η δήλωση ενός πίνακα με ακέραιους δείκτες γίνεται ως εξής:

```
int *ptable[50];
```

και για να αναθέσουμε την διεύθυνση κάποιας μεταβλητής στον πίνακα δεικτών κάνουμε το εξής:

Δείκτες και πίνακες II

```
ptable[number] = &variable;
```

Όπου **number** βάζουμε την θέση του πίνακα, στην οποία θα θέλαμε να γίνει η εκχώρηση τιμής και όπου **variable** βάζουμε το όνομα της μεταβλητής που θέλουμε.

Παράδειγμα 1: Δείκτες και συναρτήσεις I

Να δημιουργηθεί πρόγραμμα το οποίο θα:

- ορίζει δυο ακεραίους `a`, `b`
- δημιουργεί μια συνάρτηση η οποία θα εκχωρεί τη μεγαλύτερη τιμή στην `a` και τη μικρότερη τιμή στην `b`

Παράδειγμα 1: Δείκτες και συναρτήσεις I

```
#include <stdio.h>
void maxmin(int *a, int *b);

int main(){
    int a=10, b=15;
    maxmin(&a, &b);
    printf("Max= %d\nMin= %d\n", a, b);
    return 0;
}

void maxmin(int *pa, int *pb) {
    int temp= *pa;
    if (*pa<*pb){
        *pa=*pb;
        *pb=temp;
    }
}
```

Παράδειγμα 1: Δείκτες και συναρτήσεις II

Παράδειγμα 2: Δείκτες και πίνακες I

Να δημιουργηθεί πρόγραμμα το οποίο θα:

- δημιουργεί και θα συμπληρώνει έναν πίνακα ακεραίων
- η διάσταση του θα είναι πέντε (5) στοιχείων
- δηλώνει έναν δείκτη για τον πίνακα
- εμφανίζει τη θέση του στοιχείου του πίνακα και την διεύθυνση μνήμης στην οποία αποθηκεύεται.

Παράδειγμα 2: Δείκτες και πίνακες I

```
#include <stdio.h>

int main(){
    int table[5] = {20,4,67,82,9};
    int *pointer, i;
    pointer = &table[0];
    printf("Show the value and the address of each
        element\n");
    for(i=0; i <5; i=i+1)
    {
        printf("table[%d]: The value is %d and the
            address is %p\n",i, *pointer, pointer);
        pointer=pointer+1;
    }
    return 0;
}
```

Παράδειγμα 3: Δείκτες και πίνακες I

Να δημιουργηθεί πρόγραμμα το οποίο θα:

- δημιουργεί και θα συμπληρώνει έναν πίνακα ακεραίων του οποίου η διάσταση του θα είναι πέντε (5) στοιχείων
- δημιουργεί μια συνάρτηση η οποία θα επιστρέφει το άθροισμα, το μεγαλύτερο και το μικρότερο στοιχείο του πίνακα.
- δημιουργεί συναρτήσεις για το άθροισμα, το μεγαλύτερο και το μικρότερο στοιχείο του πίνακα.

Παράδειγμα 3: Δείκτες και πίνακες I

```
#include <stdio.h>


int *fres(int T[], int size);
int f_sum(int T[], int size);
int f_max(int T[], int size);
int f_min(int T[], int size);

int main(){
    int table[5] = {20,4,67,82,9};
    int *res;
    int size=5, i;
    res=fres(table, size);
    for(i=0; i <= 2; i=i+1) {
        printf("res[%d]: The value is %d\n",i, res[i]);
    }
    return 0;
```

Παράδειγμα 3: Δείκτες και πίνακες II

```
}

int *fres(int T[], int size) {
    static int mat[]={0,0,0};
    mat[0]=f_sum(T, size);
    mat[1]=f_max(T, size);
    mat[2]=f_min(T, size);
    return(mat);
}

int f_sum(int T[], int size) {
    int i, sum=0;
    for(i=0; i<size; i=i+1) {
        sum=sum+T[i];
    }
    return sum;
}
```

Παράδειγμα 3: Δείκτες και πίνακες III

```
int f_max(int T[], int size) {
    int i, max=T[0];
    for(i=1; i<size; i=i+1) {
        if(T[i]>max) {
            max=T[i];
        }
    }
    return max;
}

int f_min(int T[], int size) {
    int i, min=T[0];
    for(i=1; i<size; i=i+1) {
        if(T[i]<min) {
            min=T[i];
        }
```

Παράδειγμα 3: Δείκτες και πίνακες IV

```
    }
    return min;
}
```