ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Τεχνολογία και Προγραμματισμός Υπολογιστών

Ενότητα 6: Τύπος δείκτη, Δείκτες-πίνακες, δείκτες-αλφαριθμητικά

Διδάσκων:

Χρήστος Μακρής

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Πατρών

Τύπος δείκτη, Δείκτες-πίνακες, δείκτες-αλφαριθμητικά

(οι διαφάνειες είναι βασισμένες jκυρίως στο βιβλίο Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C KERNIGHAN W. BRIAN, RITCHIE M. DENNIS, εκδόσεις Κλειδάριθμος δεύτερη έκδοση, 2011)

Τύπος Δείκτη (1)

- Ορισμός: Μεταβλητή δείκτη (pointer variable) ή δείκτης (pointer) = μεταβλητή με τιμή μία διεύθυνση της κύριας μνήμης.
- Δεν υπάρχει τέτοια δυνατότητα σε όλες τις γλώσσες.
- Μεταβλητή με δείκτη (Indexed variable): είναι κάτι διαφορετικό (πίνακες)

Τύπος Δείκτη (2)

Δήλωση

<τύπος> * <όνομα – δείκτη>;

π.χ. int * num_ptr;

Αρχικοποίηση

<τύπος> * <όνομα-δείκτη>=>διεύθυνση>;

π.χ. int *num_ptr = 1000; (άμεσος)

int *num_ptr= # (έμμεσος)

Τύπος Δείκτη (3)

Ανάθεση / Καταχώρηση

<όνομα-δείκτη> = <διεύθυνση>;

π.χ. num_ptr = 1000; (άμεσος)

num_ptr = # (έμμεσος)

Δείκτης σε δείκτη

<τύπος> * * <μετ-δείκτη>;

π.χ. int ** ptr ;

Ανάθεση: int *num_ptr; ptr = & num_ptr

Επιτρεπόμενοι Τελεστές (1)

=	ανάθεση τιμής σε δείκτη
*	περιεχόμενο διεύθυνσης που δείχνει ο δείκτης
&	διεύθυνση μεταβλητής

- Ο τελεστής & εφαρμόζεται μόνο σε αντικείμενα που είναι στη μνήμη δηλαδή σε μεταβλητές και στοιχεία πινάκων. Δεν μπορεί να εφαρμοστεί σε παραστάσεις, σταθερές ή μεταβλητές register.

Επιτρεπόμενοι Τελεστές (2)

‘Εγκυρες πράξεις με δείκτες είναι:

- απόδοση τιμής με δείκτη ίδιου τύπου
- η πρόσθεση ή αφαίρεση δείκτη και ακεραίου
- η αφαίρεση ή σύγκριση δύο δεικτών για μέλη ίδιου πίνακα
- η αντικατάσταση ή σύγκριση με NULL

Δεν είναι έγκυρη πράξη η:

- πρόσθεση δύο δεικτών
- πολλαπλασιασμός, διαίρεση, ολίσθηση, η πρόσθεση float, double σε δείκτες.

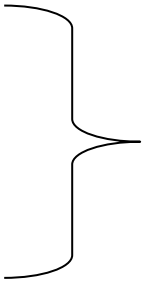
Παράδειγμα (1)

```
int x=1;
int y=2;
int *p;
P=&x;  /*  η p δείχνει τη διεύθυνση της x      */
y=*p;  /*  η y γίνεται 1, δηλαδή η τιμή της x   */
*p=0;  /*  η x γίνεται 0  */
*p=*p+10;  /*  x=x+10 */
```

Παράδειγμα (2)

```
y=*p+10; /* y=x+10 */
```

```
*p+=1;
++*p;
(*p)++;
```



x=x+1

(προτεραιότητα ο ++)

Παράδειγμα (3)

```
#include <stdio.h>
main()
{   int x=5; int y=10; int *ptr; int **ptr2;
    ptr=&x;
    ptr2=&ptr;
    *ptr2=&y;
    printf("%d", *ptr);
}
```

Παρατηρήσεις

- Ο κενός δείκτης: NULL
  - Δεν αποδεικτοδοτείται.
  - Αντιστοιχεί σε ψευδή λογική τιμή, όταν χρησιμοποιείται.
- Ο δείκτης σε κενό: void *
  - Γενική μορφή δείκτη
  - Δεν επιτρέπεται αριθμητική δεικτών.

Πέρασμα Μεταβλητών (1)

```
#include <stdio.h>
void swap(int x, int y);

int main()
{
        int x=1, y=2;
        printf("x:%d, y:%d \n", x,y);
        swap(x,y);
        printf("x:%d, y:%d \n", x,y);
}

void swap(int x, int y)   /* λάθος */
{
        int temp;
        temp=x;
        x=y;
        y=temp;
}}
```

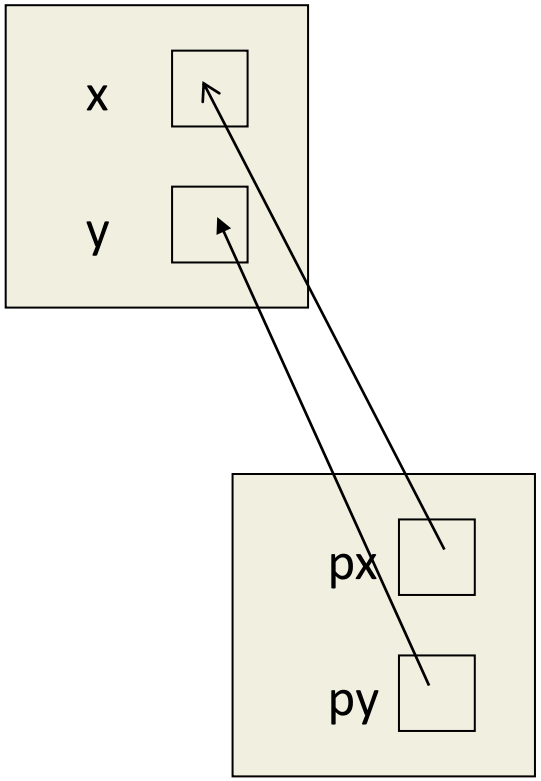
Πέρασμα Μεταβλητών (2)

```
#include <stdio.h>
void swap(int x, int y);

int main()
{
        int x=1, y=2;
        printf("x:%d, y:%d \n", x,y);
        swap(&x,&y);
        printf("x:%d, y:%d \n", x,y);
}

void swap(int *px, int *py)   /*σωστή υλοποίηση */
{
        int temp;
        temp=*px;
        *px=*py;
        *py=temp;
}}
```

Πέρασμα Μεταβλητών (3)



από βιβλίο Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C KERNIGHAN W. BRIAN, RITCHIE M. DENNIS, εκδόσεις Κλειδάριθμος δεύτερη έκδοση, 2011

Δυναμική Διαχείριση Μνήμης

- ***Calloc, Malloc***,

  για δέσμευση μνήμης
- ***Realloc***

  για αλλαγή μεγέθους δεσμευμένης μνήμης
- ***Free***

  για απελευθέρωση μνήμης

Malloc()

```
#include <stdlib.h>

void *malloc(size_t size);
```

Η συνάρτηση malloc δεσμεύει χώρο μεγέθους τουλάχιστον size bytes.

Επιστρέφει ένα δείκτη στη δεσμευθείσα μνήμη ή NULL, αν δεν υπάρχει διαθέσιμη μνήμη. Τα bytes του χώρου μνήμης δεν παίρνουν αρχική τιμή

(int *)malloc(n*sizeof(int));

x=(int **)mallloc(n*sizeof(int *));

for (i=0; i<n; i++) *(x+i)=(int *)malloc(m*sizeof(int));

&x[i][j] ---- *(x+i)+j

x[i][j] ---- *(*(x+i)+j)

Calloc()

```
#include <stdlib.h>

void *calloc(size_t n,size_t size);
```

Η συνάρτηση calloc δεσμεύει χώρο για ένα πίνακα n στοιχείων, μεγέθους size το καθένα. Διασφαλίζει την αρχικοποίησή της με 0.

Επιστρέφει ένα δείκτη στη δεσμευθείσα μνήμη ή NULL, αν δεν υπάρχει διαθέσιμη μνήμη.

p=(int *)calloc(n, sizeof(int));

Realloc()

```
#include <stdlib.h>

void *realloc(void *buffer, size_t size);
```

Η συνάρτηση realloc μεταβάλλει το μέγεθος ενός τμήματος μνήμης που είχε προηγούμενα δεσμευτεί. Το νέο μέγεθος ορίζεται από τη size. Διασφαλίζει τα υπάρχοντα περιεχόμενα στη μνήμη. Επιστρέφει ένα δείκτη στο νέο τμήμα μνήμης που μπορεί να είναι ίδιος με τον buffer ή διαφορετικός αν το τμήμα μετακινηθεί. Αν ο buffer είναι NULL, η realloc λειτουργεί σαν τη malloc.

p=(int *) realloc(p, n*sizeof(int));

Free ()

```
#include <stdlib.h>

void free(void * buffer);
```

Η συνάρτηση free αποδεσμεύει τη μνήμη η οποία σηματοδοτείται από το όρισμα buffer και η οποία είχε προηγούμενα δεσμευτεί με κλήση μίας εκ των calloc, malloc, realloc.

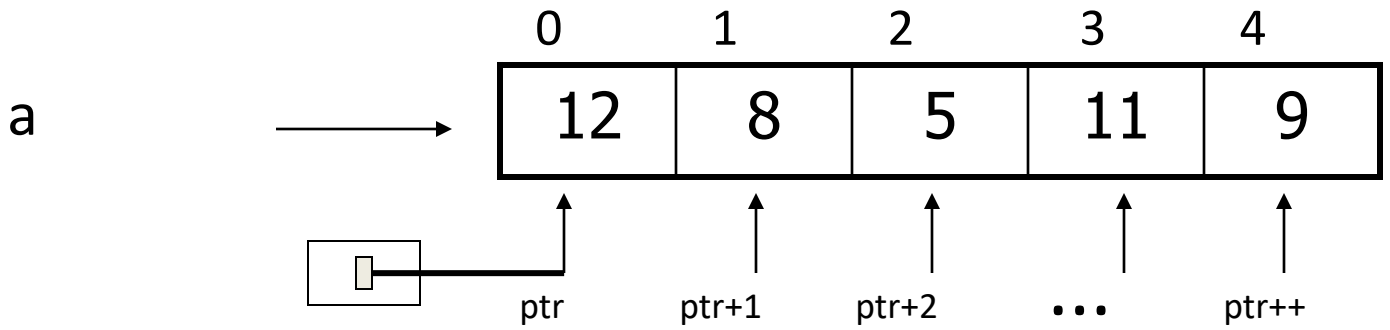
Δείκτες – Πίνακες (1)

- Το όνομα ενός πίνακα είναι μία «σταθερά τύπου δείκτη» με τιμή τη διεύθυνση του πρώτου στοιχείου του πίνακα.
- Λόγω αυτού του γεγονότος μπορούμε να ορίσουμε ένα δείκτη στον πίνακα και να χειριστούμε τον πίνακα μέσω αυτού του δείκτη.
- Δεν είναι εφικτή η ανάθεση τιμής και η αριθμητική με δείκτες

Δείκτες – Πίνακες (2)

Έστω
```
int a[5] = { 12, 8, 5, 11, 9};
int *ptr;

ptr = a; ή  ptr = &a[0];
```



Προσοχή: a++; a-- (το όνομα του πίνακα δεν είναι δείκτης (δηλ. μεταβλητή δείκτη))

Δείκτες – Πίνακες (3)

Ισοδύναμες εκφράσεις

*(a+i) ⟷ a[i]

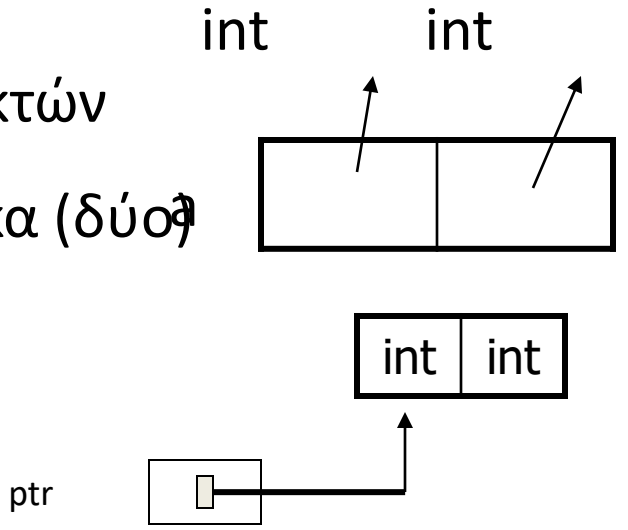
a+i ⟷ &a[i]

Δείκτης σε δείκτη

Προτεραιότητα [ ] μεγαλύτερη από *

π.χ. int *a[2] πίνακας (δύο) δεικτών

int (*ptr)[2] δείκτης σε πίνακα (δύο) ακεραίων



Ασκήσεις

1. Δημιουργείστε κώδικα ο οποίος χειρίζεται δύο μονοδιάστατους πίνακες ακεραίων αριθμών ίδιου μεγέθους, και στη συνέχεια προσθέτει τα περιεχόμενα τους ανά δύο και τα τοποθετεί στον πρώτο πίνακα. Τό μέγεθος των πινάκων θα πρέπει να δίνεται από τον χρήστη ενώ η ανάγνωση δεδομένων και η πρόσθεση των στοιχείων των πινάκων θα πρέπει να γίνεται με κλήση δύο κατάλληλα ορισμένων συναρτήσεων.

2. Δημιουργείστε κώδικα που χειρίζεται ένα πίνακα δύο διαστάσεων οριζόμενο με τη μορφή δείκτη σε δείκτη, διαβάζει τα περιεχόμενα του πίνακα και στη συνέχεια τα εκτυπώνει. Οι διαστάσεις του πίνακα θα πρέπει να δίνονται από τον χρήστη, η ανάγνωση των περιεχομένων και η εκτύπωση θα πρέπει να γίνεται με κλήση συναρτήσεων.

3. Δημιουργείστε κώδικα ο οποίος χειρίζεται δύο δισδιάστατους πίνακες ακεραίων αριθμών Α και Β μεγέθους n x k και k x m αντίστοιχα (τα n, m, k δίνονται από τον χρήστη). Το πρόγραμμα θα πρέπει να διαβάζει τα περιεχόμενα των δύο πινάκων και στη συνέχεια να αποθηκεύει το γινόμενό τους σε ένα πίνακα C διαστάσεων nxm. Η ανάγνωση των περιεχομένων και ο πολλαπλασιασμός θα πρέπει να γίνεται με την κλήση κατάλληλων συναρτήσεων.

Δείκτες - Αλφαριθμητικά (1)

Δήλωση δείκτη αλφαριθμητικού

char * <όνομα – δείκτη>;

π.χ. char * pmsg;

Ανάθεση σε δείκτη αλφαριθμητικού

<όνομα-δείκτη> = < αλφαριθμητικό >;

π.χ. pmsg = “Today is Thursday”;

Δείκτες - Αλφαριθμητικά (2)

Προσοχή στις διαφορές

char msg[ ] = “Today is Thursday”; (πίνακας χαρακτήρων)

char * pmsg = “Today is Thursday”; (δείκτης σε πίνακα, μπορεί να πάρει άλλη τιμή)

char * msg[18]; (πίνακας δεικτών χαρακτήρα)

msg[1] -> 2ος δείκτης

* (msg[1]) -> ο 1ος χαρακτήρας του δεύτερου δείκτη

Παράδειγμα (2)

```
void strcpy(char *s, char *t)
{
int  i=0;
while ((s[i]=t[i])!='\0')
i++;
}
```

```
void strcpy(char *s, char *t)
{
while ((*s=*t) != '\0') { s++; t++;}
}
```

```
void strcpy(char *s, char *t)
{
while ((* s++=*t++) != '\0') ;
}
```

στο βιβλίο Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C KERNIGHAN W. BRIAN, RITCHIE M. DENNIS, εκδόσεις Κλειδάριθμος δεύτερη έκδοση, 2011

Παράδειγμα (3)

```
void strcmp(char *s, char *t)
{
int  i=0;
for (ι=0; s[i]==t[i]; i++)
if (s[i]==‘\0’) return 0;
return s[i]-t[i];
}
```

```
int strcmp(char *s, char *t)
{
for (; *s==*t; s++, t++)
If (*s==‘\0’)  return 0;
return  *s-*t;
}
```

στο βιβλίο Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C KERNIGHAN W. BRIAN, RITCHIE M. DENNIS, εκδόσεις Κλειδάριθμος δεύτερη έκδοση, 2011

Δομές

- Στην C μπορούμε να προσθέσουμε τους δικούς μας τύπους (π.χ. FILE)
- Βασικά εργαλεία: *struct*, *typedef*.

  struct: ορίζει ένα νέο τύπο δεδομένων.

  typedef: συσχετίζει ένα όνομα με αυτόν.

- Οι δομές μπορούν να αποδίδονται σαν τιμές, να μεταβιβάζονται σε συναρτήσεις και να επιστρέφονται από συναρτήσεις. Επίσης, μπορεί να εφαρμόζεται σε αυτές ο τελεστής *sizeof*().
- Αρχικοποιούνται με λίστα τιμών όπως οι πίνακες.
- ΔΕΝ συγκρίνονται με χρήση τελεστή == ή !=
- Η αναφορά σε ένα μέλος γίνεται: *όνομα-δομής.μέλος*
- Ειδικός συμβολισμός για προσπέλαση μέλους μεταβλητής δείκτη σε δομή.

Η εντολή *struct (1)*

```
#include <stdio.h>
struct line {
    int x1, y1;  /* co-ords of 1 end of line*/
    int x2, y2;  /* co-ords of other end */
    };
main()
{
struct line line1={1,2,3,4};
.
.
}
```

Typedef

- Η *typedef* επιτρέπει τη συσχέτιση ενός ονόματος με μία δομή (ή άλλο τύπο δεδομένων)
- Τοποθέτησε την *typedef* στην αρχή του προγράμματος.

```
typedef struct line {
    int x1, y1;
    int x2, y2;
    } LINE;

int main()
{
LINE line1;
}
```

Αναφορά στα Μέλη Δομής

Χρησιμοποιείται ο τελεστής . και έχουμε:

<όνομα μεταβλητής>.<όνομα μέλους>

```
LINE line1;
line1.x1= 3;
line1.y1= 5;
line1.x2= 7;

if (line1.y2 == 3) {
    printf ("Y co-ord of end is 3\n");
    }
```

Παράδειγμα

```
typedef struct point{
    int x, y;
    } POINT;
typedef struct line {
    POINT left;
    POINT right;
    } LINE;

int main()
{
LINE line1;
line1.left.x=2;
}
```

Παρατηρήσεις

Μπορούμε να περνάμε και να επιστρέφουμε δομές σε συναρτήσεις (το πρωτότυπο όμως θα πρέπει να έχει οριστεί μετά τη δήλωση δομής)

```
typedef struct line {
    int x1,y1;
    int x2,y2;
} LINE;

LINE find_perp_line (LINE x);
/* Function takes a line and returns a
perpendicular line */
```

Παράδειγμα

```
typedef struct complex {
   float imag;
   float real;
} CPLX;

CPLX mult_complex (CPLX x, CPLX y);

int main()
{
/* Main goes here */
}

CPLX mult_complex (CPLX no1, CPLX no2)
{
    CPLX answer;
    answer.real= no1.real*no2.real - no1.imag*no2.imag;
    answer.imag= no1.imag*no2.real + no1.real*no2.imag;
    return answer;}
```

Χρησιμότητα

- Διευκολύνουν τον προγραμματισμό.
- FILE * είναι μία typedef δομή αλλά μπορούμε να τη χρησιμοποιήσουμε χωρίς να ξέρουμε τι εμπεριέχει.
- Μπορούμε να επεκτείνουμε τη γλώσσα, για παράδειγμα αν χρησιμοποιούμε μιγαδικούς αριθμούς μπορούμε να επεκτείνουμε τη γλώσσα με τον τύπο αυτό.

Παράδειγμα

Να υλοποιήσετε ένα πρόγραμμα που να διαχειρίζεται τις καρτέλες με τα στοιχεία μέχρι 20 μαθητών (επώνυμο, όνομα, βαθμός).

Το πρόγραμμα θα πρέπει να έχει τις δυνατότητες:

1.Εισαγωγής στοιχείων μαθητή

2.Παρουσίαση στην οθόνη όλων των καρτελών

3.Εντοπισμού και παρουσίασης της καρτέλας ενός συγκεκριμένου μαθητή, βασιζόμενο στο επώνυμο

4.Παρουσίαση στην οθόνη του μέσου όρου του βαθμού του συνόλου των μαθητών καθώς και τις καρτέλες με τη μικρότερη και μεγαλύτερη βαθμολογία

5.Εξόδου από το πρόγραμμα μετά από επιλογή του χρήστη.

Ενώσεις (Unions) -1

Μία ένωση (union) μπορεί να περιέχει αντικείμενα διαφορετικών τύπων και μεγεθών

π.χ.
```
union u_tag   {
        int ival;
        float fval;
        char *sval;
        } u;
```

Ενώσεις (Unions) -2

```
if (utype == INT)
    printf("%d\n", u.ival);
else if (utype==FLOAT)
  printf("%f\n", u.fval);
else if (utype==STRING)
  printf("%s\n", u.sval);
```

Στις ενώσεις επιτρέπεται απόδοση τιμής ή αντιγραφή σαν ενότητα, λήψη διεύθυνσης, προσπέλαση μέλους.

Η αρχική τιμή πρέπει να έχει τύπο ίδιο με αυτό του πρώτου της μέλους.

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών**» 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.

Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.0.

Σημείωμα Αδειοδότησης