Αρχές & Γλώσσες Προγραμματισμού 2006-2007

Ένατη Διάλεξη

Π. Σταθοπούλου
pstath@ece.upatras.gr ή
pstath@upatras.gr

Ομάδα Α΄ (Φοιτητές με μονό αριθμό Μητρώου)

- Διδασκαλία : Παρασκευή 11πμ-13μμ ΗΛ7
- Φροντιστήριο : Δευτέρα 11πμ-12πμ ΗΛ4

Προηγούμενη Διάλεξη

Χρήση αριθμητικών Πινάκων

- ***Τι είναι πίνακας ?***
- ***Μονοδιάστατοι & πολυδιάστατοι πίνακες***
- ***Ονομασία & δήλωση πινάκων***
- ***Απόδοση αρχικών τιμών σε πίνακες***
- ***Μέγιστο μέγεθος πίνακα***

Σήμερα

Χρήση και Κατανόηση των Δεικτών

- ***Τι είναι δείκτης ?***
- ***Δημιουργία δείκτη***
- ***Δείκτες & απλές μεταβλητές***
- ***Απόδοση αρχικών τιμών σε δείκτες***
- ***Δείκτες & τύποι μεταβλητών***

Χρήση & Κατανόηση των Δεικτών

Τι είναι δείκτης ?

Δείκτης είναι μία μεταβλητή η οποία περιέχει την θέση μνήμης στον υπολογιστή μιας άλλης μεταβλητής

1000	1001	1002	1003	1004	1005
				100	

Έστω μεταβλητή **rate** με αρχική τιμή **100**

ο μεταγλωττιστής δεσμεύει χώρο στη μνήμη του υπολογιστή και την αποθηκεύει στην διεύθυνση **1004**

Η μεταβλητή **p_rate** που έχει τιμή **1004** είναι δείκτης της **rate**

Χρήση & Κατανόηση των Δεικτών

Δημιουργία μεταβλητή δείκτη σημαίνει **δήλωση** μιας νέας μεταβλητής στην οποία αποθηκεύεται **διεύθυνση μνήμης.**

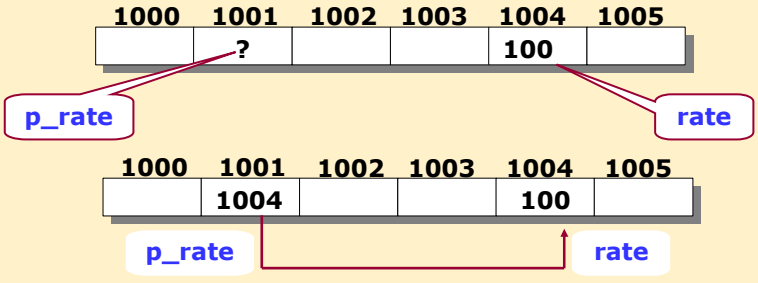
Πως ?

- **rate** απλή μεταβλητή
- **p_rate** μεταβλητή δείκτης

Χρήση & Κατανόηση των Δεικτών

Γραφική απεικόνιση

απλής μεταβλητής & μεταβλητής δείκτη



Χρήση & Κατανόηση των Δεικτών

Πως δηλώνουμε μεταβλητή δείκτη ?

typename *ptrname;

όπου
- **typename** τύπος μεταβλητής
- **ptrname** όνομα μεταβλητής
- * τελεστής εμμεσότητας (indirection)

Χρήση & Κατανόηση των Δεικτών

Παραδείγματα δηλώσεων μεταβλητών δεικτών

char *ch1, *ch2;

***ch1** και ***ch2** είναι δείκτες των μεταβλητών **ch1**, **ch2** τύπου χαρακτήρα(**char**).

float *value, percent;

***value** είναι δείκτης της μεταβλητής **value** τύπου **float**

percent είναι απλή μεταβλητή τύπου **float**

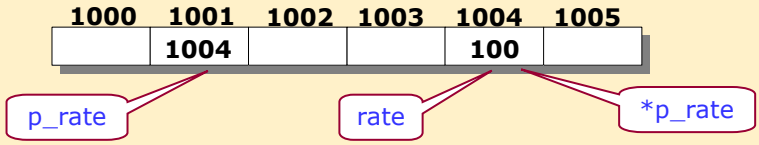
Χρήση & Κατανόηση των Δεικτών

Απόδοση αρχικών τιμών σε δείκτες **(Χρήση δεικτών)**

pointer = &namevariable; π.χ. ***p_rate = &rate;***

printf ("%d", rate); άμεση πρόσβαση

printf ("%d", *p_rate); έμμεση πρόσβαση



p_rate rate *p_rate

Χρήση & Κατανόηση των Δεικτών

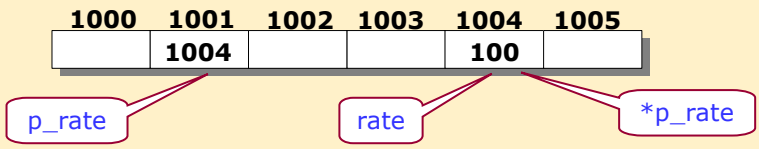
Έστω δείκτης με όνομα **ptr** και δείχνει μεταβλητή **var**

- ***ptr** και **var**

αναφέρονται στα περιεχόμενα της μεταβλητής **var**

- **ptr** και **&var**

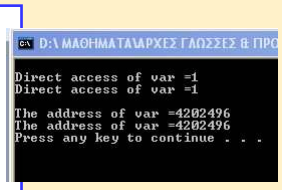
αναφέρονται στην διεύθυνση της μεταβλητής **var**



p_rate rate *p_rate

Χρήση & Κατανόηση των Δεικτών

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

int var=1;
int *ptr;

int main(int argc, char *argv[]){
  ptr=&var;

  printf("\nDirect access of var =%d",var);
  printf("\nDirect access of var =%d",*ptr);

  printf("\n\nThe address of var =%d",&var);
  printf("\nThe address of var =%d\n",ptr);

  system("PAUSE");
  return 0;
}
```



Χρήση & Κατανόηση των Δεικτών

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

int *p

int main(int argc, char *argv[])
{
  *p=10;


  system("PAUSE");
  return 0;
}
```

- Λάθος **p** δείκτης ακεραίου που δεν δείχνει μια μεταβλητή

Χρήση & Κατανόηση των Δεικτών

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

int *p;
double q,temp;

int main(int argc, char *argv[])
{
  temp=1234.34;
  p=&temp;
  q=*p;
  printf("%f",q);

  system("PAUSE");
  return 0;
}
```

- Λάθος **p** δείκτης ακεραίου
- **temp** float

  Δεν γίνεται compilation

- Λάθος **q** double μεταβλητή
- **p** δείκτης ακεραίου

  Δεν τρέχει το πρόγραμμα

Χρήση & Κατανόηση των Δεικτών

Οι **δείκτες** είναι ιδιαίτερης σημασίας στην **C** και χρησιμοποιούνται για την υποστήριξη **σύνθετων δομών δεδομένων**

- Διασυνδεδεμένες λίστες (linked lists)
- Δυαδικά δένδρα (binary trees)

- Η **C** υποστηρίζει δύο τελεστές για τους δείκτες:

  ***** και **&**
- Σε μεταβλητές δεικτών εφαρμόζονται επίσης και οι αριθμητικοί τελεστές

  +,++,- και **--**

Χρήση & Κατανόηση των Δεικτών

Στους **δείκτες** μπορούν να προστεθούν ή να αφαιρεθούν **μόνο** ακέραιες ποσότητες από δείκτες.

ΠΡΟΣΟΧΗ: Δεν μπορούμε να προσθέσουμε **ποτέ** αριθμό κινητής υποδιαστολής σε δείκτη!!!

Η **αριθμητική των δεικτών** διαφέρει από την **κανονική αριθμητική** με σημαντικό τρόπο.

Εκτελείται σε σχέση με τον βασικό τύπο του δείκτη

Αρχές & Γλώσσες Προγραμματισμού 2006-2007

Χρήση & Κατανόηση των Δεικτών

Στους **δείκτες** μπορούν να προστεθούν ή να αφαιρεθούν **μόνο** ακέραιες ποσότητες από δείκτες.

ΠΡΟΣΟΧΗ: Δεν μπορούμε να προσθέσουμε **ποτέ** αριθμό κινητής υποδιαστολής σε δείκτη!!!

Η **αριθμητική των δεικτών** διαφέρει από την **κανονική αριθμητική** με σημαντικό τρόπο.

Εκτελείται σε σχέση με τον βασικό τύπο του δείκτη

Χρήση & Κατανόηση των Δεικτών

Παράδειγμα

έστω ένας δείκτης τύπου ακεραίου με όνομα **p** που περιέχει την διεύθυνση μνήμης **200** τότε

η έκφραση **p++** έχει την τιμή **202** γιατί **int** 2 **bytes**

if p δείκτης **float** τότε **p++ =204**

Στους δείκτες μπορούμε να εφαρμόσουμε:

```
int *p;
.....
p=p+200;
```

Τελεστές αύξησης & μείωσης

***p++ ;**	αυξάνει ο δείκτης μετά τιμή
(*p)++;	αυξάνει η τιμή του δείκτη

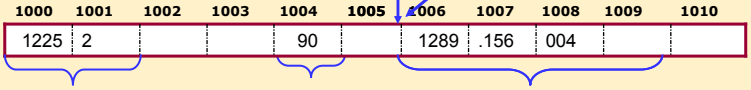
Δείκτες & τύποι μεταβλητών

Διαφορετικοί τύποι μεταβλητών

διαφορετικό χώρο στην μνήμη

short 2 bytes **float 4 bytes**

Δηλώσεις Μεταβλητών	Δηλώσεις Δεικτών	Σχέση δεικτών/μεταβλητών
int vshort=12252; **char** vchar= 90; **float** vfloat=1289.156004	**int** *p_vshort ; **char** *p_vchar ; **float** *p_vfloat ;	p_vshort=&vshort p_vchar = &vchar p_vfloat =&vfloat



Χρήση & Κατανόηση των Δεικτών

Δείκτες & Πίνακες

Στη **C** υπάρχει **ειδική σχέση** ανάμεσα στους **δείκτες** & στους **πίνακες**.

Χρήση αριθμητικών Πινάκων **=** χρήση Δεικτών

Έστω πίνακας **data[]**

data = διεύθυνση του 1ου στοιχείου του πίνακα

data = & data

Χρήση & Κατανόηση των Δεικτών

Δείκτες & Πίνακες

δείκτης =διεύθυνση

```
int array[100], *p_array;
p_array=array;
```

```
p_array=array[0];
p_array=array[i];
```

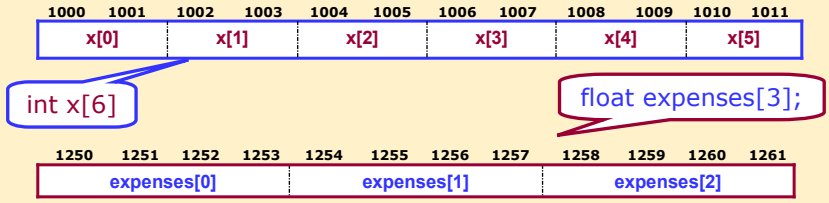
διεύθυνση 1ου στοιχείου

1250

Χρήση & Κατανόηση των Δεικτών

Δείκτες & Πίνακες

X==1000	expenses==1250
&x[0]==1000	&expenses[0]==1250
&x[1]=1002	&expences[1]==1254



Χρήση & Κατανόηση των Δεικτών

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

int ctr;
short array_s[10];
float array_f[10];
double array_d[10];

int main(int argc, char *argv[])
{
 printf("\n\tThe size of array SHORT type= %d bytes\n",sizeof(array_s));
 printf("\tThe size of array FLOAT type= %d bytes\n",sizeof(array_f));
 printf("\tThe size of array DOUBLE type= %d bytes\n\n",sizeof(array_d));

  system("PAUSE");
  return 0;
}
```

Χρήση & Κατανόηση των Δεικτών

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

int ctr;
short array_s[10];
float array_f[10];
double array_d[10];
int main(int argc, char *argv[])
{
printf("\t\tShort array\tFloat array\tDouble array");
printf("\n================================");
printf("========================");
   for (ctr = 0; ctr < 10; ctr++)
        printf("\nElement %d:\t%ld\t\t%ld\t\t%ld", ctr, &array_s[ctr],
&array_f[ctr], &array_d[ctr]);
printf("\n================================");
printf("========================\n\n");
system("PAUSE");
  return 0;
}
```

```
                Short array     Float array     Double array
================================================
Element 0:      4210928         4210784         4210848
Element 1:      4210930         4210788         4210856
Element 2:      4210932         4210792         4210864
Element 3:      4210934         4210796         4210872
Element 4:      4210936         4210800         4210880
Element 5:      4210938         4210804         4210888
Element 6:      4210940         4210808         4210896
Element 7:      4210942         4210812         4210904
Element 8:      4210944         4210816         4210912
Element 9:      4210946         4210820         4210920
================================================

Press any key to continue . . .
```


Χρήση & Κατανόηση των Δεικτών

Αριθμητική των δεικτών στους Πίνακες

Αύξηση & Μείωση των δεικτών

*ptrToShort δείκτης για Πίνακα τύπου short

*ptrToShort++; αύξηση κατά 2 bytes
επόμενο στοιχείο του πίνακα

*ptrToFloat δείκτης για Πίνακα τύπου float

*ptrToFloat++; αύξηση κατά 4 bytes
επόμενο στοιχείο του πίνακα

Χρήση & Κατανόηση των Δεικτών

Αριθμητική των δεικτών στους Πίνακες

Αύξηση & Μείωση των δεικτών

*ptrToShort +=4; αύξηση κατά 2*4=8 bytes
4ο επόμενο στοιχείο του πίνακα

*ptrToFloat +=10; αύξηση κατά 4*10=40 bytes
10ο επόμενο στοιχείο του πίνακα

Χρήση & Κατανόηση των Δεικτών

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

#define MAX 10
int i_array[MAX] = { 0,1,2,3,4,5,6,7,8,9 };
float f_array[MAX] = { .0, .1, .2, .3, .4, .5, .6, .7, .8, .9 };
int *i_ptr, count;
float *f_ptr;
int main(int argc, char *argv[])
{
  i_ptr = i_array;
  f_ptr = f_array;

  for (count = 0; count < MAX; count++)
      printf("%d\t%f\n", *i_ptr++, *f_ptr++);

 system("PAUSE");
 return 0;
}
```

```
D:\ ΜΑΘΗΜΑΤΑ\ΑΡΧΕΣ ΓΛΩΣΣΕΣ & ΠΡΟ
0       0.000000
1       0.100000
2       0.200000
3       0.300000
4       0.400000
5       0.500000
6       0.600000
7       0.700000
8       0.800000
9       0.900000
Press any key to continue . . .
```

Χρήση & Κατανόηση των Δεικτών

Πράξεις δεικτών

- **Διαφοροποίηση = αφαίρεση δεικτών**
  *ptr1-*ptr2 πράξη έγκυρη
- **Συγκρίσεις δεικτών** *ptr1 <*ptr2 πράξη έγκυρη

Πράξη	Περιγραφή
Εκχώρηση	**&** **η τιμή είναι διεύθυνση της μεταβλητής**
Εμμεσότητα	* **η τιμή της μεταβλητής**
Διεύθυνση	** **η τιμή είναι διεύθυνση της διεύθυνσης**
Αύξηση	++, +
Μείωση	--
Διαφοροποίηση	-
Σύγκριση	<, ==, <, !=, >=, <=

Χρήση & Κατανόηση των Δεικτών

Σημειογραφία Πινάκων & Δεικτών

```
*(array)== array[0]
*(array+1)== array[1]
*(array+2)== array[2]
 ..................................
 ..................................
*(array+n)== array[n]
```

Χρήση & Κατανόηση των Δεικτών

Πίνακες σε Συναρτήσεις

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10

int array[MAX+1], count;
int largest(int num_array[]);

t main(int argc, char *argv[])
{
.....
printf("\n\nLargest value = %d\n", largest(array));
}
```

```
int largest(int num_array[])
{
.....
return biggest;
}
```


Χρήση & Κατανόηση των Δεικτών

Πίνακες σε Συναρτήσεις

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10

int array[MAX+1], count;
int largest(int *num_array);

t main(int argc, char *argv[])
{
.....
printf("\n\nLargest value = %d\n", largest(array));
}
```

```
int largest(int *num_array)
{
.....
return biggest;
}
```