ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

4η Εβδομάδα: Δείκτες

Αναφορές

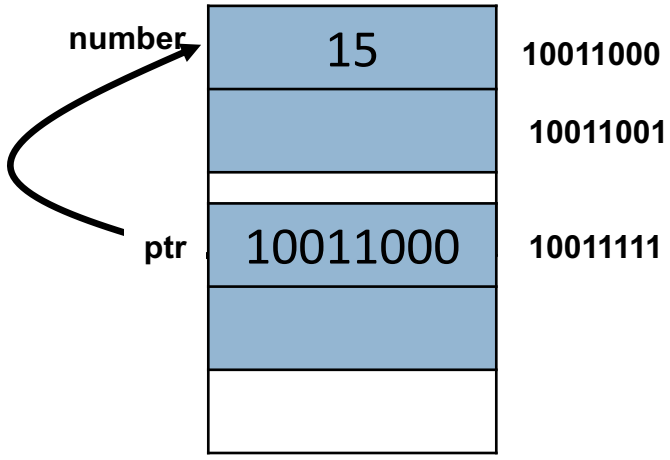
Οι διαφάνειες της διάλεξης στηρίζονται, εν μέρει, σε υλικό παραδόσεων παλαιότερων ετών του **Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογία Υπολογιστών του Πανεπιστημίου Πατρών** καθώς και του **Τμήματος Πληροφορικής του Πανεπιστήμιο Κύπρου.**

Θέματα Διάλεξης

- Να ορίζετε και να αρχικοποιείτε **μεταβλητές διεύθυνσης** (δείκτες/pointers)
- Τελεστές για δείκτες/Pointers
- Να ορίζετε εκφράσεις με δείκτες και αριθμητική με δείκτες

Ορισμός Μεταβλητών Διεύθυνσης (Δείκτες)

- Μια μεταβλητή τύπου δείκτη (Pointer)
  - Είναι μια μεταβλητή που σαν τιμή μπορεί να πάρει μόνο **διεύθυνση μνήμης (στη RAM)**
  - Χρειάζεται ΠΑΝΤΑ να ξέρει ποιος είναι ο **τύπος της τιμής** που υπάρχει σε εκείνη τη διεύθυνση
  - Εάν μια μεταβλητή (π.χ. Τύπου ακέραιου; int number = 15) είναι ένα κουτάκι στη RAM όπου φυλάγεται ένας ακέρραιος, τότε μια **μεταβλητή τύπου δείκτη** (π.χ., int *ptr = &number) είναι ένα **άλλο κουτάκι στη** RAM όπου μέσα υπάρχει η **διεύθυνση προς το κουτάκι όπου** φυλάγεται η μεταβλητή (π.χ., η **number**.



Ορισμός Μεταβλητών Διεύθυνσης (Δείκτες)

- Μια μεταβλητή τύπου δείκτη ορίζεται με το *
- Ταυτόχρονα δηλώνεται ο **τύπος της τιμής (int, float, double, char, κτλ)** που επιτρέπεται να αποθηκευτεί στη διεύθυνση μνήμης (στη RAM) που δείχνει η μεταβλητή τύπου δείκτη
  - Π.χ., το παρακάτω ορίζει μια μεταβλητή τύπου δείκτη (**ptrNumber**) η οποία θα **λάβει μια διεύθυνση** στην RAM στην οποία θα αποθηκευτεί μια μεταβλητή τύπου int

```
int * x;
char *otherpointer;
```

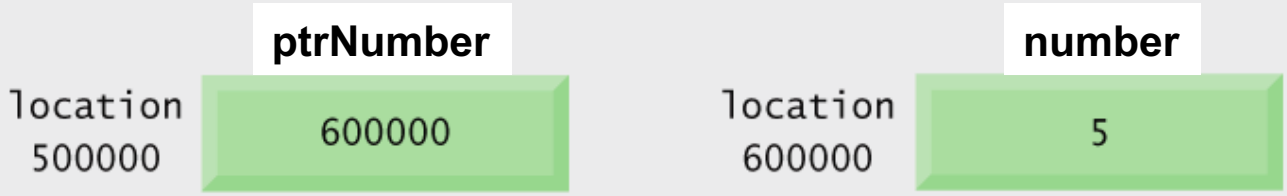
- Είναι δυνατό να ορίσετε **δείκτες σε οποιοδήποτε τύπο δεδομένων**.
- Οι δυνατές **αρχικές τιμές** ενός δείκτη να είναι είτε **NULL είτε μια διεύθυνση** στην μνήμη (RAM)

Πως παίρνουμε τη διεύθυνση μιας μεταβλητής;

- Ο τελεστής διεύθυνσης **&**
  - **Επιστρέφει την διεύθυνση** μιας μεταβλητής, π.χ.

```
int number = 5;
int *ptrNumber;
ptrNumber = &number;     // η ptrNumber παίρνει σαν τιμή
                         // τη διεύθυνση του number
```

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

void main(void) {

    int number = 15;
    int *ptrNumber = NULL;
    ptrNumber = &number;

    printf("%d\n", number);
    printf("%x\n", ptrNumber);
    printf("%d\n", *ptrNumber);
}
```

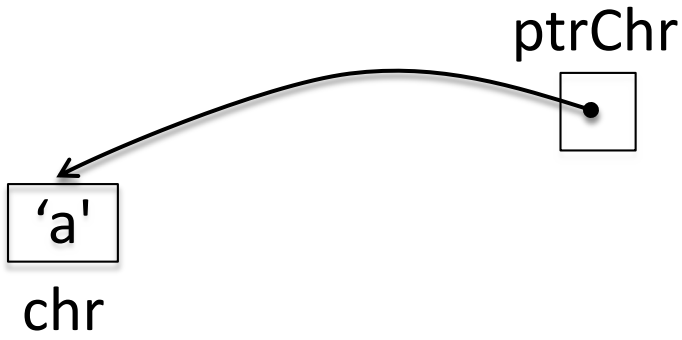


Λέμε ότι η μεταβλητή **ptrNumber** έχει αποθηκευμένη τη διεύθυνση που "δείχνει στη" RAM όπου είναι αποθηκευμένη η μεταβλητή number η οποία είναι τύπου int

Πως παίρνουμε τη διεύθυνση μιας μεταβλητής;

```
int number = 5;
int *ptrNumber;
ptrNumber = &number;
```


```
char chr = 'a';
char *ptrChr;
ptrChr = &chr;
```



Πως χρησιμοποιούμε τους δείκτες;

- Ο Τελεστής *
  - **Μας πηγαίνει στη διεύθυνση που του δίνουμε**, πχ.

```
int number = 5;
int *ptrNumber;
ptrNumber = &number;

printf("%d", *ptrNumber);      // τυπώνει 5

*ptrNumber = 7; // αλλάζει την τιμή της number σε 7 */
```

Άρα, όποτε γράφω ***ptrNumber** θα είναι σαν να γράφω **number**!

Χαρακτήρας *

- Τελεστής γινομένου
  - Number1 * Number2;
- Δήλωση Δείκτη
  - int *ptrNumber;
- Τελεστής Έμμεσης Αναφοράς *
  - int number = 5;
  - int ptrNumber = &number;
  - number = *ptrNumber + 1;

Παράδειγμα 1


```
int number, *ptr1, *ptr2;

number = 6;             // number παίρνει την τιμή 6
ptr1 = &number;         // ptr1 παίρνει την διεύθυνση που βρίσκεται η number στη RAM
ptr2 = ptr1;            // ptr2 παίρνει την διεύθυνση που αποθηκεύσαμε στη ptr1

*ptr1 = *ptr1 + 1;      // number = number + 1
*ptr2 = *ptr2 + 1;

printf("%d %d %d", number, *ptr1, *ptr2);
```

8 8 8 (Γιατί;)

Παράδειγμα με δείκτες

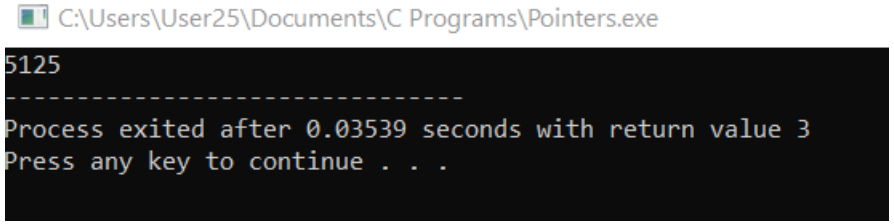
```
int     x = 1,   y = 2,   z[10];
int     *ip,  *iq;

ip    = &x;                  /* ο ip δείχνει τώρα την x */
*ip   = *ip + 1;             /* η μεταβλητή που δείχει ο ip
                          (η x) αυξάνεται κατά 1 */
y     = *ip + 3;             /* η y παίρνει την τιμή 5 */
*ip   = 0;                   /* η x παίρνει την τιμή 0 */
ip    = &z[6];               /* ο ip δείχνει τώρα το z[6] */
iq    = ip;                  /*ο iq δείχνει εκεί που
                             δείχνει και ο ip */
```

Συναρτήσεις: Παράμετροι Εισόδου/Εξόδου --

- Υπάρχουν δύο τρόποι να περάσουμε παραμέτρους εισόδου σε μια συνάρτηση:
  - **Πέρασμα διά τιμής (call by value):** Οι τιμές των μεταβλητών αντιγράφονται στις παραμέτρους της συνάρτησης οι οποίες χρησιμοποιούνται για υπολογισμό/επεξεργασία. Οι τιμές των μεταβλητών όμως στο πρόγραμμα μας από το οποίο γίνεται η κλήση της συνάρτησης δεν επηρεάζονται. → **Αυτό τον τρόπο τον είδαμε σε προηγούμενες διαλέξεις.**
  - **Πέρασμα δια αναφοράς (call by reference):** Με τον τρόπο αυτό **μεταβιβάζονται** στις παραμέτρους της συνάρτησης **οι διευθύνσεις μνήμης που είναι αποθηκευμένες οι μεταβλητές στην RAM**. Έτσι η συνάρτηση μπορεί να επηρεάσει τις τιμές των μεταβλητών στο πρόγραμμα από το οποίο γίνεται κλήση της συνάρτησης γιατί έχει πρόσβαση στις διευθύνσεις των μεταβλητών.
    - Με αυτό τον τρόπο μπορεί να γίνει επιστροφή πολλών τιμών μέσω διευθύνσεων

Παράδειγμα χρήσης δεικτών


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

void cubeGivenValue(int *num);

void main(void) {
    // Dilwsi metavlitis kai arxikopoiisi
    int number = 5;

    // Typwse arxiki timi tis number
    printf("%d", number);

    // Kalese tin synartisi me orisma TIN DIEYTHINSI tis number
    cubeGivenValue(&number);

    // Typwse tin timi tis number
    printf("%d", number);
}

void cubeGivenValue(int *num){
    *num = *num * *num * *num;
}
```



Εδώ περνάμε σαν παράμετρο **μια ΔΙΕΥΘΥΝΣΗ προς ακέραια μεταβλητή**

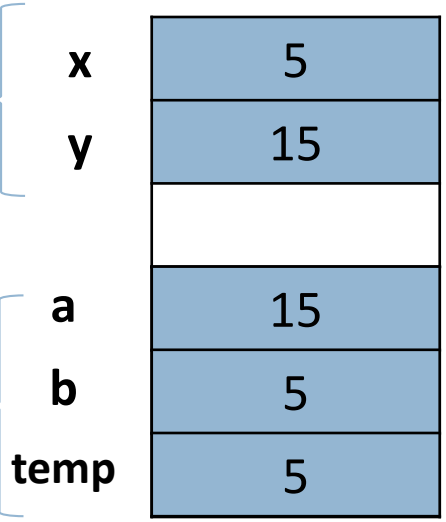
Κλήση της διαδικασίας **By Reference**.

Η τιμή της παραμέτρου είναι η διεύθυνση της μεταβλητής number στην μνήμη RAM

Το αποτέλεσμα της συνάρτησης αποθηκεύεται απευθείας στη θέση της μνήμης στη RAM όπου δείχνει ο δείκτης p **(Τι ακριβώς γίνεται εδώ;)**

Αλλαγή τιμών μεταβλητών εντός συνάρτησης Χωρίς χρήση μεταβλητών δεικτών


```
#include <stdio.h>
void main(void) {
    int x = 5, y = 15;
    printf("x=%d, y=%d\n", x, y);
    swap( x, y );
    printf("x=%d, y=%d\n", x, y);
}

void swap(int a, int b){

    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

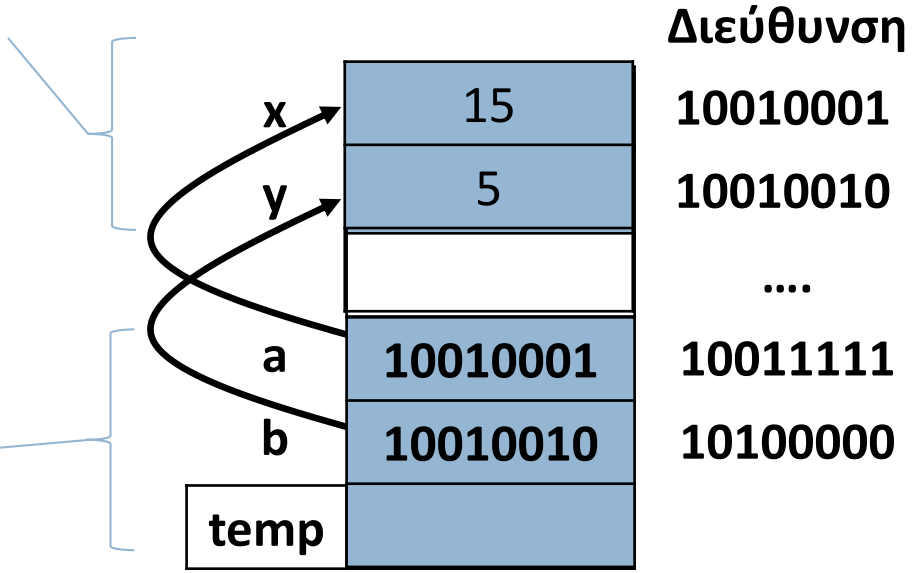



```
x=5, y=15
x=5, y=15
```

Γιατί δεν γίνεται η αλλαγή;

Αλλαγή τιμών μεταβλητών εντός συνάρτησης
Με χρήση μεταβλητών δεικτών


```
#include <stdio.h>
void main(void) {
    int x = 5, y = 15;
    printf("x=%d, y=%d", x, y);
    swap( &x, &y );
    printf("x=%d, y=%d", x, y);
}

void swap(int *a, int *b){

    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```




```
x=5, y=15
x=15, y=5
```

Δείκτες και Πίνακες

Στην C υπάρχει ισχυρός δεσμός ανάμεσα στους **δείκτες** και στους **πίνακες**. Οποιαδήποτε πράξη μπορεί να γίνει με **δείκτες πινάκων** (πχ α[i]) μπορεί να γίνει και με **δείκτες διευθύνσεων *pa**.

Έστω πίνακας int a[10]. Η δήλωση αυτή ορίζει ένα μπλοκ 10 διαδοχικών αντικειμένων με ονόματα α[0], α[1], ..., α[9].

Αν `int *pa;`

τότε η ανάθεση `pa = &a[0];`

κάνει τον `pa` να δείχνει το μηδενικό στοιχείο του πίνακα.

Αριθμητική δεικτών : Αν ο pa είναι δείκτης σε κάποια θέση του πίνακα, τότε οι pa ± i, pa++, pa-- είναι επίσης δείκτες

- `pa + 1, pa++` δείκτης στο επόμενο στοιχείο του πίνακα από αυτό που δείχνει ο pa.
- `pa + i , (pa – i)` δείκτης στο σημείο του πίνακα που βρίσκεται i θέσεις μετά (πριν) από αυτό που δείχνει ο pa.

Δείκτες και Πίνακες (συν.)

Έτσι, αν pa = &a[0], τότε για παράδειγμα,

pa + 2	δείχνει στη τρίτη θέση του πίνακα, δηλ. στο α[2].
*(pa + 2)	έχει τιμή την τιμή της 3ης θέσης του πίνακα, δηλ. το α[2].

Έξ'ορισμού, η τιμή μιας μεταβλητής τύπου πίνακα (int a[]) είναι η διεύθυνση του μηδενικού στοιχείου του πίνακα.

Επομένως αντί pa = &a[0] μπορούμε να γράψουμε pa = a.

Τότε οι pa και a έχουν τις ίδιες τιμές και μπορούν να χρησιμοποιούνται η μια στη θέση της άλλης:

- Το a[i] είναι ταυτόσημο με τα *(pa +i) ή *(a + i) ή pa[i].
- Το &a[i] είναι ταυτόσημο με τα a+i, pa+i.

Δείκτες και Πίνακες


```
int table[20];
int *ptrTable;
ptrTable = table;
```

- Το όνομα του πίνακα (table), είναι η διεύθυνση στοιχείου table[0]
  - table ίδιο με &table[0]
  - *ptrTable ίδιο με table[0]

- **Προσαύξηση Δείκτη** (Γίναται μόνο με μεταβλητές δεικτών)
  - ptrTable++ ο δείκτης μετακινείται στο επόμενο στοιχείο του πινακα table
- **Μειωση Δείκτη** (Γίναται μόνο με μταβλητές δεικτών)
  - ptrTable-- ο δείκτης μετακινείται στο προηγούμενο στοιχείο του πινακα table

Προσοχή!

- Δείκτης μπορεί να πάρει νέα τιμή

```
int table[10]= {0,1,2,3,4,5,6,7,8,9}, *ptr;

ptr = table;                    // ptr = &table[0]
ptr[4] = 3;                     // ανάθεσε στην table[4] την τιμή 3
++ptr;                          // ptr μετακινείται στο table[1]
++ptr;                          // ptr μετακινείται στο table[2]
*ptr = 14;          // ανάθεσε στην table[2] την τιμή 14
++table;                        // ΔΕΝ ΕΠΙΤΡΕΠΕΤΑΙ
```

Το όνομα ενός πίνακα δεν είναι μεταβλητή

Δείκτες και Πίνακες

- **Ποιες από τις πιο κάτω εντολές είναι έγκυρες;**

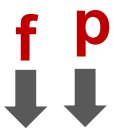
  int f[9] = { }, i = 5, *p;

Index	0	1	2	3	4	5	6	7	8
Value	0	0	0	0	0	0	0	0	0

Δείκτες και Πίνακες

- **Ποιες από τις πιο κάτω εντολές είναι έγκυρες;**

```
int f[9] = { }, i = 5, *p;
p = f;
```

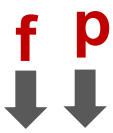


Index	0	1	2	3	4	5	6	7	8
Value	0	0	0	0	0	0	0	0	0

Δείκτες και Πίνακες

- **Ποιες από τις πιο κάτω εντολές είναι έγκυρες;**

```
int f[9] = { }, i = 5; *p;
p = f;
p[2] = 3;
```

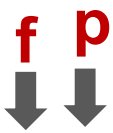


Index	0	1	2	3	4	5	6	7	8
Value	0	0	3	0	0	0	0	0	0

Δείκτες και Πίνακες

- **Ποιες από τις πιο κάτω εντολές είναι έγκυρες;**

```
int f[9] = { }, i = 5, *p;
p = f;
p[2] = 3;
++p;
```



Index	0	1	2	3	4	5	6	7	8
Value	0	0	3	0	0	0	0	0	0

Δείκτες και Πίνακες

- **Ποιες από τις πιο κάτω εντολές είναι έγκυρες;**

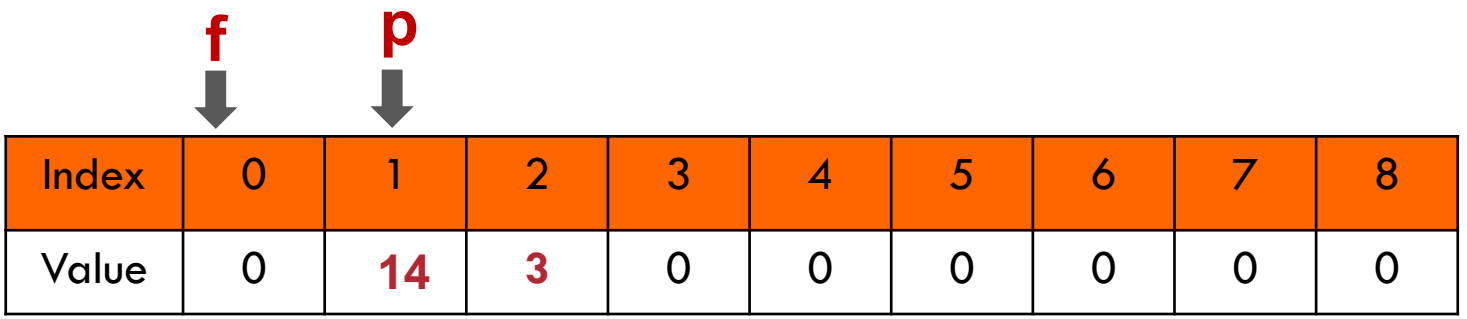
```
int f[9] = { }, i = 5, *p;
p = f;
p[2] = 3;
++p;
```

Index	0	1	2	3	4	5	6	7	8
Value	0	0	3	0	0	0	0	0	0

Δείκτες και Πίνακες

- **Ποιες από τις πιο κάτω εντολές είναι έγκυρες;**

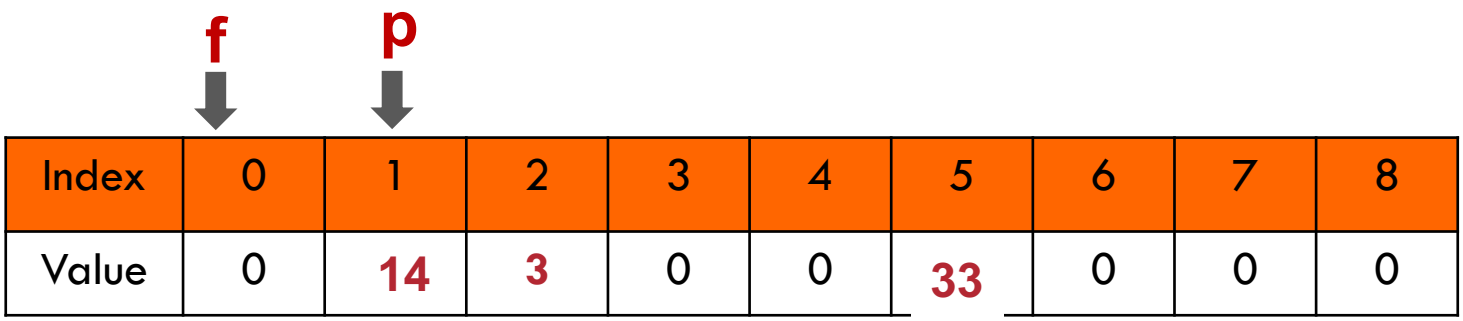
```
int f[9] = { }, i = 5, *p;
p = f;
p[2] = 3;
++p;
*p = 14;
```



Δείκτες και Πίνακες

- **Ποιες από τις πιο κάτω εντολές είναι έγκυρες;**

```
int f[9] = { }, i = 5, *p;
p = f;
p[2] = 3;
++p;
*p = 14;
*(f+i) = 33;
```



Δείκτες και Πίνακες

- Ποιες από τις πιο κάτω εντολές είναι έγκυρες;

```
int f[9] = { }, i = 5, *p;
p = f;
p[2] = 3;
++p;
*p = 14;
*(f+i) = 33;
++f;
```

→ Compile error!

Προσοχή: ο δείκτης είναι μεταβλητή και έτσι p = f και p++ είναι έγκυρες εκφράσεις. Το όνομα ενός πίνακα όμως δεν είναι μεταβλητή, επομένως κατασκευές όπως f = p και f++ δεν είναι έγκυρες!

f p

Index	0	1	2	3	4	5	6	7	8
Value	0	14	3	0	0	33	0	0	0

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/

ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

3η Εβδομάδα: Πίνακες

Πρόβλημα

- Πώς θα γράφαμε πρότυπο και τον ορισμό συνάρτησης η οποία θα δέχεται ως ορίσματα:

  α) το αλφαριθμητικό και

  β) τον χαρακτήρα για τον οποίο γίνεται ο έλεγχος.

- Πρότυπο αυτής:

  int countchar(**char** *, **char**);

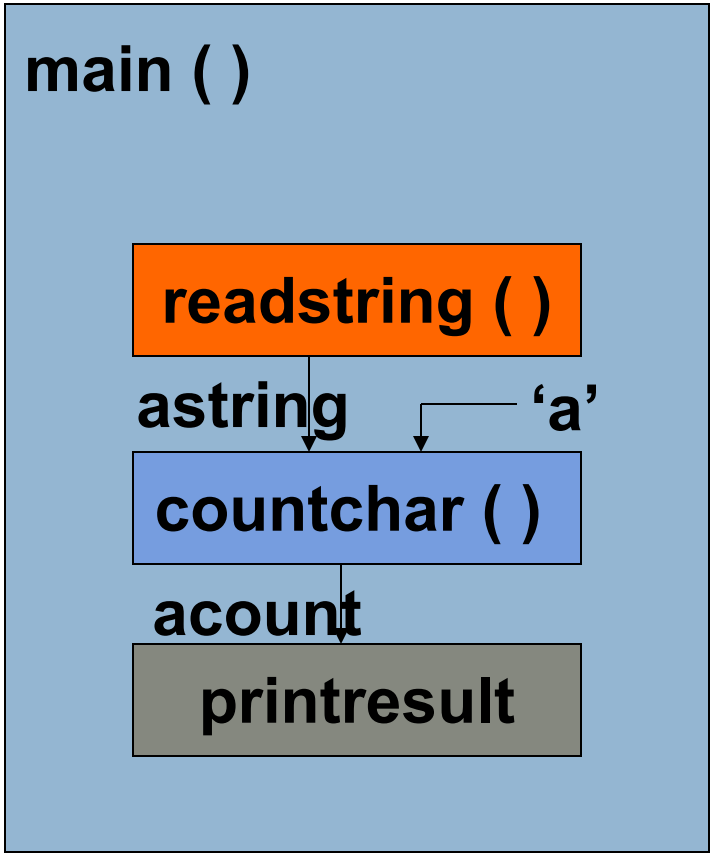
- Παράδειγμα κλήσης

  acount = countchar(astring, 'a');

- Τι πλεονεκτήματα έχει να γράφουμε γενικότερο κώδικα;

Χρήση της countchar(**char *, char**)

```
main ( ) {
 char astring[N];
 int acount ;

 readstring (astring) ;

 acount = countchar(astring, ‘a’);

 printresult(acount) ;
}
```



Πιθανές υλοποιήσεις

```
int countA(char *s, char ch) {
   int count = 0 ;
   int i = 0;
   while ( s[i] != 0 ) {
     if (s[i] == ch)
        count ++;
     i ++ ;
     }
  return count ;
}
```

```
int countA(char *s, char ch) {
   int count = 0 ;
   int i ;
   for (i=0; s[i] != 0; i++)
      if (s[i] == ch)
             count ++;
   return count ;
}
```

int countchar(char *, char);

```
int countchar(char *s, char c) {
   int count = 0 ;
   int i = 0;
   while ( s[i] ) {
              count += (s[i] == c);
              i ++ ;
            }
  return count ;
}
```

```
int countchar(char *s, char c) {
 int count = 0 ;
 int i = 0;
 while ( s[i] )
              count += (s[i++] == c);
 return count ;
}
```

postfix notatio

```
int countchar(char *s, char c) {
 int count = 0 ;
 int i = 0;
 while ( s[i] ) {
              count += (s[i] == c);
              i=i+1;
  }
 return count ;
}
```

int countchar(char *, char);

```
int countchar(char *s, char c) {
int count = 0 ;
int i = 0;
  while ( s[i] ) {
              count += (s[i] == c);
            i ++ ;
            }
  return count ;
}
```

```
int countchar(char *s, char c) {
int count = 0 ;
int i = 0;
  while ( s[i] )
              count += (s[i++] == c);
return count ;
}
```

postfix nota

```
int countchar(char *s, char c) {
int count = 0 ;
int i ;

  for( i = 0; s[i]; count += (s[i++] == c));

return count ;
}
```

```
int countchar(char *s, char c) {
int count = 0 ;
int i = 0;

  for( ; s[i]; count += (s[i++] == c));

return count ;
}
```

mystrcpy(char to[], char from[])

Υλοποιήστε την συνάρτηση:

mystrcpy(char to[], char from[])

η οποία αντιγράφει το String b στο String a

Αλγόριθμος

Για κάθε στοιχείο from[i] αντίγραψε το from[i] στην θέση to[i], μέχρι να φτάσεις στο \0.

mystrcpy(char to[], char from[])

```
#include <stdio.h>
void mystrcpy(char to[ ], char from[ ])
{
   int i=0;
  while (from[i] != '\0') {
                    to[i] = from[i];
                    i++;
        }
    to[i]='\0';
}
```

```
void main() {
     char ma[10];
     char mb[10]="HELLO";
mystrcpy(ma,mb);
     printf("ma=%s and mb=%s", ma, mb);
     return;
}
```

Πρίν

	0	1	2	3	4	5	6	7	8	9
to	?	?	?	?	?	?	?	?	?	?
from	H	E	L	L	O	\0	?	?	?	?

Μετά

	0	1	2	3	4	5	6	7	8	9
to	H	E	L	L	O	\0	?	?	?	?
from	H	E	L	L	O	\0	?	?	?	?

mystrcpy(char *str1, char *str2)

- Εκδοχή με δείκτες:

```
void mystrcpy(char *str1, char *str2){

    while (*str1 != '\0'){
        *str2 = *str1;
        str2++;
        str1++;
    }
}
```

```
void main() {
    char ma[10];
    char mb[10]="HELLO";
mystrcpy(mb,ma);
    printf("ma=%s and mb=%s", ma, mb);
    return;
}
```

ΣΥΜΒΟΛΟΣΕΙΡΕΣ ► ΠΙΝΑΚΕΣ

- Πίνακες από συμβολοσειρές:
  - **char names[NUM_STUDENTS][NAME_LEN];**
  - **char weekdays[7][10]={“Monday”, “Tuesday”, “Wednesday”, “Thursday”, “Friday”, “Saturday”, “Sunday”};**

Είναι βέλτιστη η διαχείριση της μνήμης;

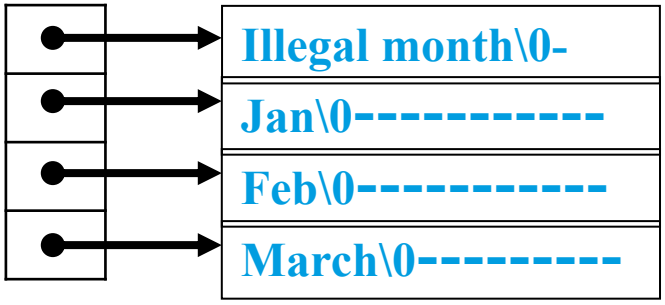
0	1	2	3	4	5	6	7	8	9
M	O	N	D	A	Y	\0			
T	U	E	S	D	A	Y	\0		
W	E	D	N	E	S	D	A	Y	\0
…	…	…	…	…	…	…	…	…	…
S	U	N	D	A	Y	\0			

ΠΙΝΑΚΑΣ ΔΕΙΚΤΩΝ VS ΠΟΛΥΔΙΑΣΤΑΤΟΙ ΠΙΝΑΚΕΣ

- ΠΑΡΑΔΕΙΓΜΑ:

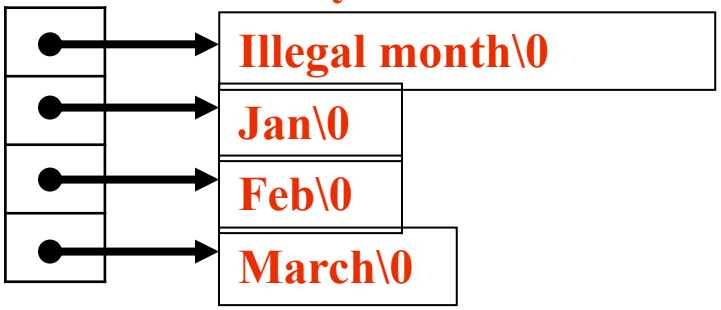
```
#include <stdio.h>
int main() {
    char  fixed[] [15] =  {“Illegal month”, “Jan”, “Feb”, “March”};
    char  *variable[]   =  {“Illegal month”, “Jan”, “Feb”, “March”};
          }
```

Fixed memory allocation:



- συμβολίζει ένα αχρησιμοποίητο byte

Variable memory allocation:



Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/