Διαδικαστικός Προγραμματισμός

Βασίλης Παλιουράς

Αληθείς και Ψευδείς Εκφράσεις

- `τιμή έκφρασης != 0` $\Rightarrow$ η έκφραση είναι **αληθής**

- `τιμή έκφρασης == 0` $\Rightarrow$ η έκφραση είναι **ψευδής**

Λογικές μεταβλητές στη C

- ISO C90
  - Δεν υπάρχει τύπος λογικής μεταβλητής
    - έκφραση !=0 είναι αληθής
- ISO C99
  - Επεκτείνει το C90
  - Ορίζει _**Bool**
  - Τυποποιεί το header file <stdbool.h>
    - **bool, true, false**

Ομαδοποίηση Τελεστών

Κατηγορία	Ενδεικτικά C
Αριθμητικοί	* / % + -
Λογικοί	&& || !
Συσχετιστικοί	> >= == !=
Διαχείρισης δυαδικών ψηφίων μιας λέξης	>> & | ^
Τελεστές διαχείρισης μνήμης	& [ ] . -> *

iso646.h

- Τροποποίηση C95 του C90

Macro	Ορίζεται ως
`and`	`&&`
`and_eq`	`&=`
`bitand`	`&`
`bitor`	`|`
`compl`	`~`
`not`	`!`
`not_eq`	`!=`
`or`	`||`
`or_eq`	`|=`
`xor`	`^`
`xor_eq`	`^=`

Παράδειγμα ISO C99

```c
#include <stdio.h>
#include <stdbool.h>
#include <iso646.h>
// C99 style for comments and boolean logic

int main() {
          bool bmorethana;
          bool blessthanc;
          bool between;
          int a = 1, b =2, c =3;

          bmorethana = b>a ;
          blessthanc = b<c ;

          if(bmorethana and blessthanc) {
              printf("b is between 'em\n");
              between = true;
          }
          else {
              printf("b is out of limits\n");
              between = false ;
          }

          if (between)
             printf("between is true");

          return 0;
}
```

Στα πλαίσια του μαθήματος γράφουμε κυρίως ISO C90

Portability

Έλεγχος Ισότητας

- `a == 5;` /* αληθές αν το a είναι 5 */
- `a != 5;` /* αληθές αν το a δεν είναι 5 */
- άλλος ο ρόλος του = άλλος του ==

Έλεγχος Ισότητας ==

```
int main () {
int a = 1, b =1;

if ( a == b) {
  printf ("equal");
}
else {
  printf ("unequal");
}
return 0;
}
```

```
int main () {
int a = 1, b =1;

if ( a != b) {
  printf ("unequal");
}
else {
  printf ("equal");
}
return 0;
}
```

Ανάθεση και Ισότητα

```
int main () {
  int a = 1, b =1;
  int condition;

  condition = (a==b);

  if (condition)
      printf("equal");
  else
      printf("unequal");

return 0;
}
```

```
int main () {
  int a = 1, b =1;
  int condition;

  condition = (a==b);

  if (!condition)
      printf("unequal");
  else
      printf("equal");

return 0;
}
```

Λογικοί τελεστές

```
#include <stdio.h>

void main () {

int a=1, b=2, c=3;

if (b>a && b<c)
   printf("b is between 'em");
else
   printf("b is out of limits");

return ;
}
```

Συμβολισμός Προθέματος-Επιθέματος Prefix – Postfix notation

- ```
  a++;     /* postfix */
  ```
- ```
  ++a;     /* prefix */
  ```
- ```
  f(a++); /*  function called, then a
              updated */
  ```
- ```
  f(++a); /* a updated, then function
             called */
  ```

Τι τυπώνεται;

```c
#include <stdio.h>

int main() {

        int a = 4;

        printf("%d\n", a);
        a++;
        printf("%d\n", a);
        ++a;
        printf("%d\n", a++);
        printf("%d\n", a);
        printf("%d\n", ++a);
        printf("%d\n", a);

        return 0;
}
```

Σύνδεση εκφράσεων με λογικούς τελεστές

```c
#include <stdio.h>

int main()
{
  int a = 1, b = 0;

  if (a==1 || ++b ==1)
          printf("hello\n");

  printf("value of b after if: %d\n", b);

  return 0;
}
```

Τι αλλάζει στη συμπεριφορά, αν το a αρχικοποιηθεί στο 0;

Short-circuit evaluation

Ισοδύναμος κώδικας για || στη λογική συνθήκη

```c
#include <stdio.h>

int main()
{
  int a = 0, b = 0;

  if (a==1)
       printf("hello\n");
  else if (++b ==1)
       printf("hello\n");

  printf("value of b after if: %d\n", b);

  return 0;
}
```

Στην περίπτωση συνθήκης με &&, σε τι θα διέφερε;

Παράδειγμα 2.1: Αριθμός επαναλήψεων εξαρτώμενος από τα δεδομένα

- Να γραφεί ένα πρόγραμμα που διαβάζει **ακεραίους**, έναν κάθε φορά και τυπώνει το μερικό άθροισμα, **όσο** ο χρήστης δίνει ως είσοδο **αριθμούς > 0**.
- Αριθμοί ≤ 0 δεν λαμβάνονται υπόψη στους υπολογισμούς.
- Στο **τέλος** τυπώνεται το συνολικό άθροισμα και το γινόμενό τους.

έκδοση 5

```c
#include <stdio.h>

int main( )
{

  int input=1, sum = 0 , prod = 1;

  while ( input > 0) {
     scanf("%d", &input) ;

     if (input <=0 )
        break;

     sum = sum + input ;
     printf("partial sum: %d\n", sum);
     prod = prod * input ;
     }

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

  return 0;
}
```

Παράδειγμα 3.1

- Να γραφεί ένα πρόγραμμα που διαβάζει **ακεραίους** έναν κάθε φορά και τυπώνει το μερικό άθροισμα **των άρτιων**, **όσο** ο χρήστης δίνει ως είσοδο **αριθμούς > 0**.
- Αριθμοί ≤0 δεν λαμβάνονται υπόψη στους υπολογισμούς
- Στο τέλος τυπώνεται το συνολικό άθροισμα και το γινόμενό **των άρτιων**.

Παράδειγμα 3.1 έκδοση 1

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

int main( )
{

  int input=1, sum = 0 , prod = 1;

  while ( input > 0) {
     scanf("%d", &input) ;
     if (input > 0 ) {
               if ( input % 2 == 0 ) {
                     sum = sum + input ;
                     printf("input even, partial sum: %d\n", sum);
                     prod = prod * input ;
                  }
        }
 }

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

 return 0;
}
```

Παράδειγμα 3.1 έκδοση 1.1

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

int main( )
{

  int input=1, sum = 0 , prod = 1;

  while ( input > 0) {
     scanf("%d", &input) ;

     if (input > 0 &&  input % 2 == 0 ) {
                    sum = sum + input ;
                    printf("input even, partial sum: %d\n", sum);
                    prod = prod * input ;
                  }

   }

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

 return 0;
}
```

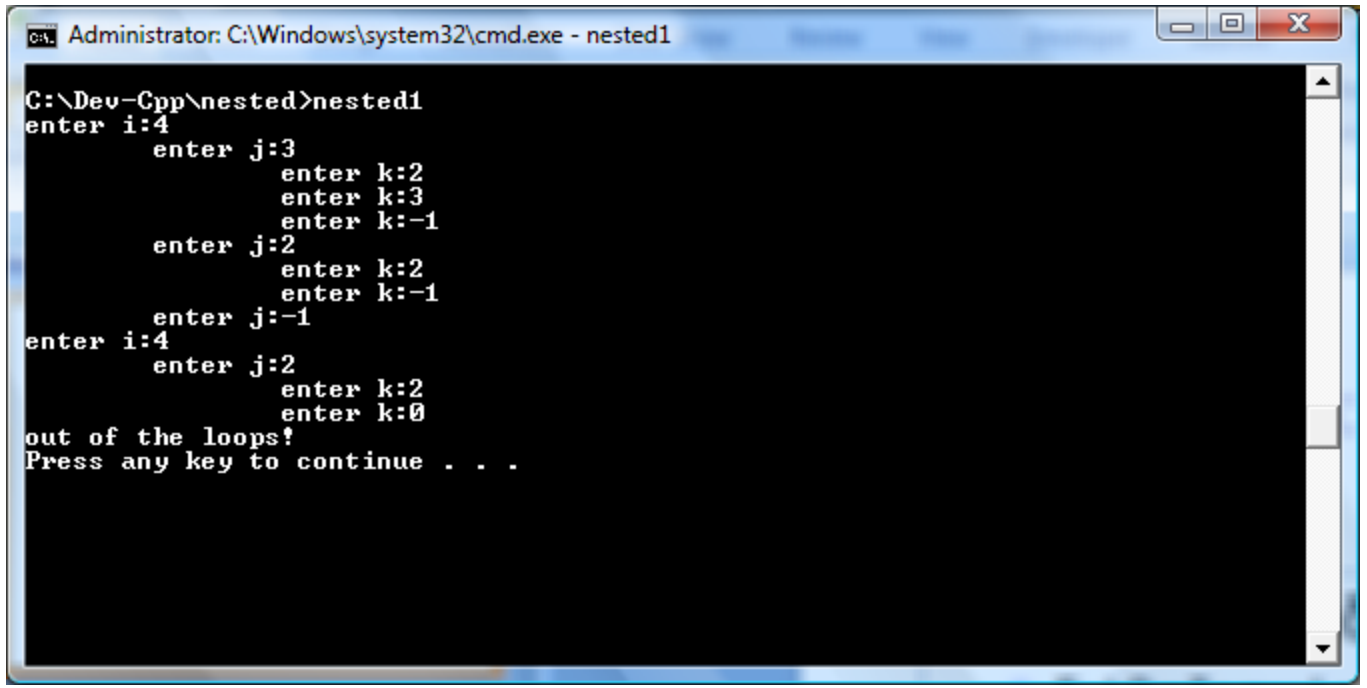
λογική σύζευξη (ΚΑΙ, AND)

Παράδειγμα 3.1 έκδοση 2

```c
#include <stdio.h>

int main( )
{
    int input=1, sum = 0 , prod = 1;

  while ( input > 0) {
     scanf("%d", &input) ;

     if (input <=0 )
                 break ;

     if (input % 2 != 0)
                 continue;

     sum = sum + input ;
     printf("input even, partial sum: %d\n", sum);
     prod = prod * input ;
   }

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

return 0;
}
```

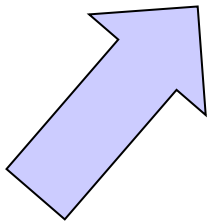
Παράδειγμα 4: Ένθετοι βρόχοι επανάληψης

- Διάβασε τριάδες ακεραίων i, j, k ως εξής
  - διάβαζε τιμές i, **όσο** i>0. Για κάθε i:
    - **αν** i <=0, σταμάτα **αλλιώς**
    - διάβαζε τιμές του j, **όσο** j>0. Για κάθε j
      - **αν** j <=0, διάβασε νέα τιμή του i αλλιώς
      - διάβαζε τιμές του k, **όσο** k>0. Για κάθε k
        - αν k ==0 σταμάτα το διάβασμα **όλων**
        - αν k <=0 διάβασε νέα τιμή του j


Administrator: C:\Windows\system32\cmd.exe - nested1
C:\Dev-Cpp\nested>nested1
enter i:4
enter j:3
enter k:2
enter k:3
enter k:-1
enter j:2
enter k:2
enter k:-1
enter j:-1
enter i:4
enter j:2
enter k:2
enter k:0
out of the loops!
Press any key to continue . . .

Έκδοση 0 – Λεκτική

```
Αρχικοποίηση i
Όσο (i>0) {
    Διάβασε i
    Αν (i > 0) {
        Αρχικοποίηση j
        Όσο (j > 0) {
            Διάβασε j
            Αν (j > 0) {
                Αρχικοποίηση k
                Όσο (k>0) {
                    Διάβασε k
                    Αν (k == 0) {
                        βγες εκτός των βρόχων
                        }
                }
            }

    }
    }
    }
```



Πώς θα γίνει αυτό;

```c
#include <stdio.h>
int main() {
  int i ,j, k, sum;
  int exitall = 0 ;
  i = 1;
  while(i>0 && !exitall) {
            printf("enter i:");
            scanf("%d", &i);
            if ( i > 0) {
               j = 1;
               while (j>0 && !exitall) {
                  printf("\tenter j:");
                  scanf("%d", &j);
                  if ( j > 0) {
                     k = 1;
                     while (k > 0 ) {
                        printf("\t\tenter k:");
                        scanf("%d", &k);
                        if ( k == 0)
                               exitall = 1;
                        else {
                            if (k > 0 ) {
                               sum = sum + k;
                               }
                            }
                     }
                  }
               }
            }
   }
printf("out of the loops!\n");
return 0;
}
```

έκδοση 1

Δομημένο στυλ

`exitall`

Ελέγχει την έξοδο από

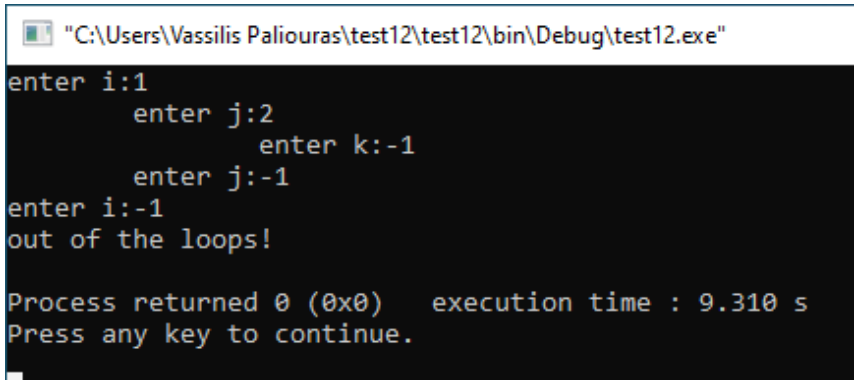
Βρόχους επανάληψης.

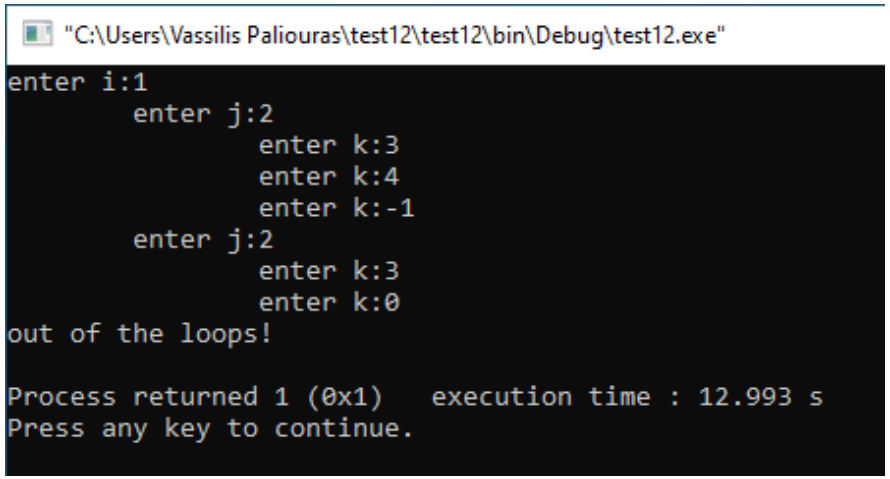
έκδοση 2

```c
#include <stdio.h>
int main( )  {
  int i ,j, k, sum;
  i = 1;
  while (i>0) {
            printf("enter i:");
            scanf("%d", &i);
            if ( i > 0) {
               j = 1;
               while (j>0 ) {
                  printf("\tenter j:");
                  scanf("%d", &j);
                  if ( j > 0) {
                     k = 1;
                     while (k > 0 ) {
                        printf("\t\tenter k:");
                        scanf("%d", &k);
                        if ( k == 0)
                               goto EXITLOOPS;
                        else {
                            if (k > 0 ) {
                               sum = sum + k;
                               }
                            }
                     }
                  }
               }
            }
   }
  EXITLOOPS:
  printf("out of the loops!\n");
  return 0;
}
```

```c
#include <stdio.h>
int main( )  {
  int i ,j, k, sum;
  i = 1;
  while (i>0) {
            printf("enter i:");
            scanf("%d", &i);
            if ( i > 0) {
               j = 1;
               while (j>0 ) {
                  printf("\tenter j:");
                  scanf("%d", &j);
                  if ( j > 0) {
                     k = 1;
                     while (k > 0 ) {
                        printf("\t\tenter k:");
                        scanf("%d", &k);
                        if ( k == 0) {
                               printf("out of the loops!\n");
                               return 1;
                            }
                        else {
                            if (k > 0 ) {
                               sum = sum + k;
                               }
                            }
                     }
                  }
               }
            }
   }
  printf("out of the loops!\n");
  return 0;
}
```

έκδοση 2.1





```c
#include <stdio.h>
int nestedloops(void);

int main(void )  {
  int status ;

  status  = nestedloops();
  printf("out of the loops! (%d)\n", status);
  return 0;
}
```

return σε συνάρτηση

```c
int nestedloops(void) {
  int i ,j, k, sum;
  i = 1;

  while (i > 0) {
             printf("enter i:");
             scanf("%d", &i);
             if ( i > 0) {
                j = 1;
                while (j > 0) {
                   printf("\tenter j:");
                   scanf("%d", &j);
                   if (j > 0) {
                      k = 1;
                      while (k > 0) {
                         printf("\t\tenter k:");
                         scanf("%d", &k);
                         if (k == 0) {
                                return 1;
                                }
                         else {
                             if (k > 0) {
                                sum = sum + k;
                                }
                             }
                      }
                }
             }
          }
   }
   return 0;
}
```

Πολλαπλές επιλογές: switch

```
switch (έκφραση)  {
   case τιμή1: εντολές ; break;
   case τιμή2: εντολές ; break;
   case τιμή3: εντολές ; break;
   /* … */
   default: εντολές ; break;
   }
```

case τιμή1: λειτουργεί ως label

break: μεταφέρει τον έλεγχο εκτός του switch () {}

τι γίνεται χωρίς **break**

Παράδειγμα

- Να γραφεί πρόγραμμα τέτοιο ώστε το σύστημα να ζητά από τον χρήστη **να επιλέξει μία μεταξύ τριών διαθέσιμων επιλογών**:
  - να ξεκινήσει μια συγκεκριμένη διεργασία,
  - να σταματήσει η διεργασία,
  - να λήξει η εκτέλεση του προγράμματος.
- Θα ζητείται είσοδος από τον χρήστη **έως ότου επιλεγεί η λήξη** του προγράμματος.

Παράδειγμα – Λεκτική περιγραφή λύσης σε προστακτική μορφή

- Ζήτησε από τον χρήστη **να επιλέξει μία μεταξύ τριών επιλογών**:
  - να ξεκινήσει μια συγκεκριμένη διεργασία,
  - να σταματήσει η διεργασία,
  - να λήξει η εκτέλεση του προγράμματος.
- Συνέχισε να ζητάς επιλογή από χρήστη **έως ότου επιλεγεί η λήξη** του προγράμματος.

```
int userchoice;
```

Παράδειγμα – Λεκτική περιγραφή - ρήματα

- Ζήτησε από το χρήστη **να επιλέξει μεταξύ τριών επιλογών**: `getchoice( )`
- Ανάλογα με την userchoice
  - αν είναι 1, ξεκίνησε τη διεργασία, `start( )`
  - αν είναι 2, σταμάτα τη διεργασία, `stop( )`
  - αν είναι 3, να λήξει η εκτέλεση του προγράμματος.
- Συνέχισε να ζητάς επιλογή από χρήστη **έως ότου επιλεγεί η λήξη** του προγράμματος.

Από λεκτική περιγραφή λύσης σε κατασκευές

```
userchoice = getchoice();
while (δεν επιλέχθηκε η λήξη) {
    Ανάλογα με την (τιμή της) userchoice
        αν είναι 1, start();
        αν είναι 2, stop();
        αν είναι 3, να λήξει η εκτέλεση του προγράμματος.
    userchoice = getchoice();
    }
```

Οργάνωση βασικού βρόχου επανάληψης (1)

```
int main (void ) {
    int userchoice ;

    userchoice = getchoice ( ) ;

    while (userchoice != 3 ) {
        switch (userchoice) {
           case 1: start( ); break;
           case 2: stop( ); break;
           default: break;
         }
        userchoice =  getchoice() ;
  }

  return 0;
}
```

```c
int main ( void){
int userchoice ;

userchoice = getchoice();

while (userchoice != 3) {
   switch (userchoice) {
        case 1: start( );
                break;
        case 2: stop( );
                break;
        default:break;
        }

   userchoice=getchoice();

   }

return 0;
}
```

```c
int main (void ) {
int userchoice ;

while((userchoice=getchoice())!= 3)
   {
       switch (userchoice)
        {
        case 1: start(); break;
        case 2: stop(); break;
        default: break;
        }

   }
return 0;
}
```

Υλοποίηση συνάρτησης getchoice( )

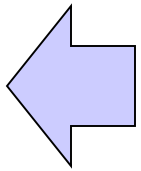
```
int getchoice (void) {
    int choice ;

    printf("1: start\n2: stop\n3: quit\n");
    printf("enter choice:\n");
    scanf("%d", &choice);

    return choice;
}
```

Η κλήση (⇒ χρήση)

(για παράδειγμα στην υλοποίηση της `main()`)

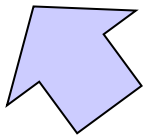
```
userchoice = getchoice( );
```

```c
#include <stdio.h>
int getchoice (void) ;
void start (void) ;
void stop (void);

int main (void){

    int userchoice ;

    while ((userchoice = getchoice()) != 3){
     switch (userchoice) {
        case 1: start() ;
                break;
        case 2: stop();
                break;
        default: break;
     }
   }

   return 0;
}
```

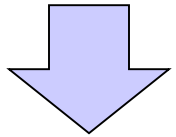


Πρότυπο της συνάρτησης

Δήλωση για type checking



Ορισμός συνάρτησης



```c
int getchoice (void ) {
 int choice ;

 printf("1: start\n2: stop\n3:quit\n");
 printf("enter choice:\n");
 scanf("%d", &choice);

 return choice;
}

void start (void) {
    printf("Start...");
     return;
}

void stop (void) {
    printf("Stop...");
    return ;
}
```

```c
#include <stdio.h>
int getchoice (void) ;
void start (void) ;
void stop (void);

int main (void){

    int userchoice ;

    while ((userchoice = getchoice()) != 3){
        switch (userchoice) {
            case 1: start();
            case 2: stop();
            default: printf("default\n");
        }
    }

    return 0;
}
```

Τι θα συμβεί αν παραλείψουμε τα break;

Πίνακες

- Συλλογή μεταβλητών **ίδιου τύπου**, οι οποίες αποθηκεύονται σε διαδοχικές θέσεις μνήμης.
- **float** temperature[31];
  - *δήλωση* πίνακα μεταβλητών **float,** 31 στοιχείων
  - temperature[**0**] είναι το **πρώτο** στοιχείο,
  - temperature[**1**] είναι το **δεύτερο** στοιχείο,
  - ...
  - temperature[**30**] είναι το **τριακοστό πρώτο** στοιχείο,
  - temperature είναι **η διεύθυνση του πρώτου στοιχείου**
    - temperature είναι το ίδιο με &temperature[0]

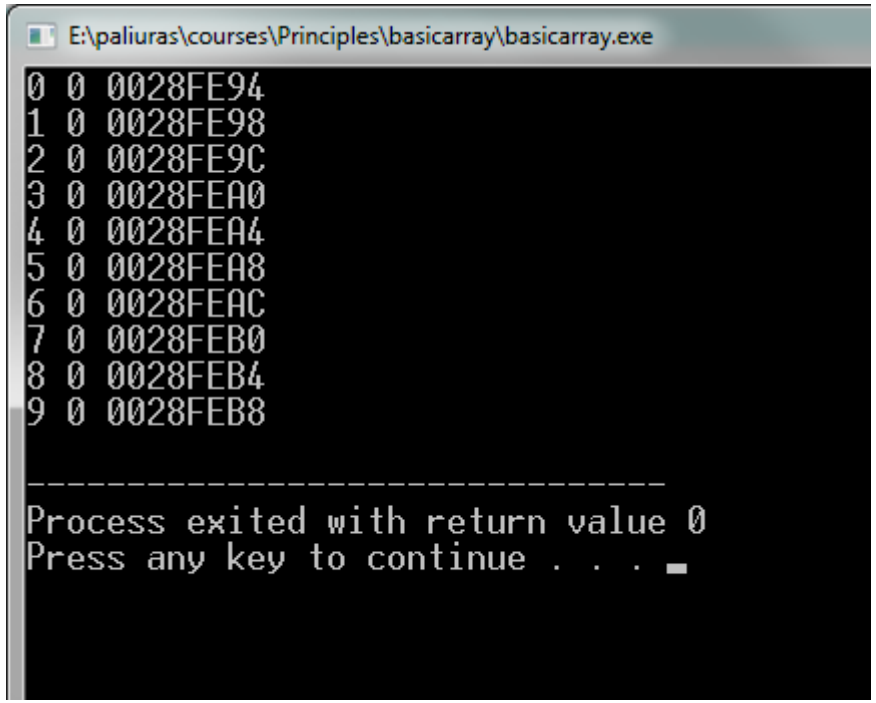
Παράδειγμα

```
#include <stdio.h>
int main (void ) {
  int i;
  float temp[3]  = { -2.1f, 5.5f, 10.1f};

  for (i=0; i< 3; i = i + 1)
     printf("\ttemp[%d]: %f\n", i, temp[i]);

  return 0;
}
```

```
printf("\ttemp[%d]: %+6.2f\n", i, temp[i]);
```

cygwin

```c
#include <stdio.h>
#define N 10

int main() {

        int i ;
        int data[N] = {0};

        for (i=0; i< 10; i++)
                printf("%d %d %p\n", i, data[i], &data[i]);

        return 0;
}
```



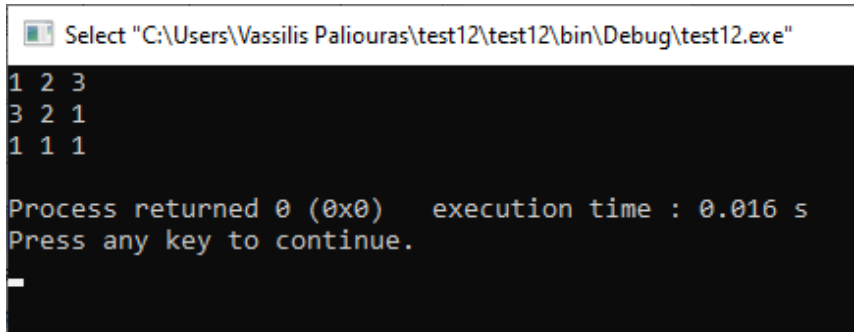
0	1	2	3	4	5	6	7	8	9

Πίνακες δύο (ή περισσότερων) διαστάσεων

- `int a[3][3] ;`
- `int a[3][3] = {{1,2,3}, {3,2,1}, {1,1,1}};`

1	2	3
3	2	1
1	1	1

```c
#include <stdio.h>
#define N 3

int main ( ) {
int i, j;

int a[N][N] = {{1,2,3}, {3,2,1}, {1,1,1}};

for (i=0; i<N; i++) {
    for (j=0; j<N; j++)
            printf("%d ",a[i][j]);
    printf("\n");
}

return 0;
}
```



Αποθήκευση στη μνήμη - πίνακας μιας διάστασης

```
int a[3] = {10,20,30};
```

Διεύθυνση	Στοιχείο	Περιεχόμενα
1000	a[0]	10
1004	a[1]	20
1008	a[2]	30

Αποθήκευση στη μνήμη πίνακας δύο διαστάσεων

a[1] σημαίνει δεύτερη γραμμή
a[1][2] σημαίνει τρίτο στοιχείο δεύτερης γραμμής

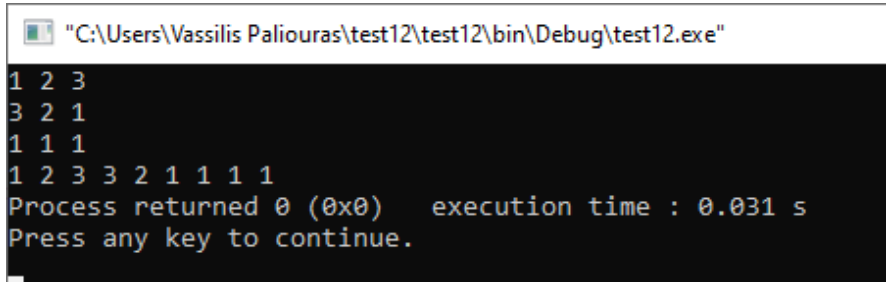
1	2	3
3	2	1
1	1	1

Διεύθυνση	Στοιχείο
1000	1
1004	2
1008	3
100C	3
1010	2
1014	1
1018	1
101C	1
1020	1

```c
#include <stdio.h>
#define N 3

int main ( void) {
int i, j;

int a[N][N] = {{1,2,3}, {3,2,1}, {1,1,1}};
int *b = &a[0][0];  /* b holds the address of first
                     * element */

for (i=0; i< N; i++) {
    for (j=0; j< N; j++)
            printf("%d ",a[i][j]);
    printf("\n");
    }

for (i=0; i< N*N; i++)
    printf("%d ", *(b+i)); /* value of ith integer
                              starting from address b*/

return 0;
}
```

Στην περίπτωση αυτή
Αριθμητική τιμή του b, 1000

Διεύθυνση	Στοιχείο
1000	1
1004	2
1008	3
100C	3
1010	2
1014	1
1018	1
101C	1
1020	1



Συνάρτηση της βασικής βιβλιοθήκης scanf ( )

```
int number;
char ch;
```

%d θα διαβάσει ακέραιο

```
scanf("%d", &number);
```

%c θα διαβάσει χαρακτήρα

```
scanf("%c", &ch);

scanf("\n%c", &ch);
```

τελεστής διεύθυνσης (&): Επιστρέφει τη διεύθυνση της θέσης μνήμης η οποία αντιστοιχεί στη μεταβλητή που ακολουθεί

Παράδειγμα

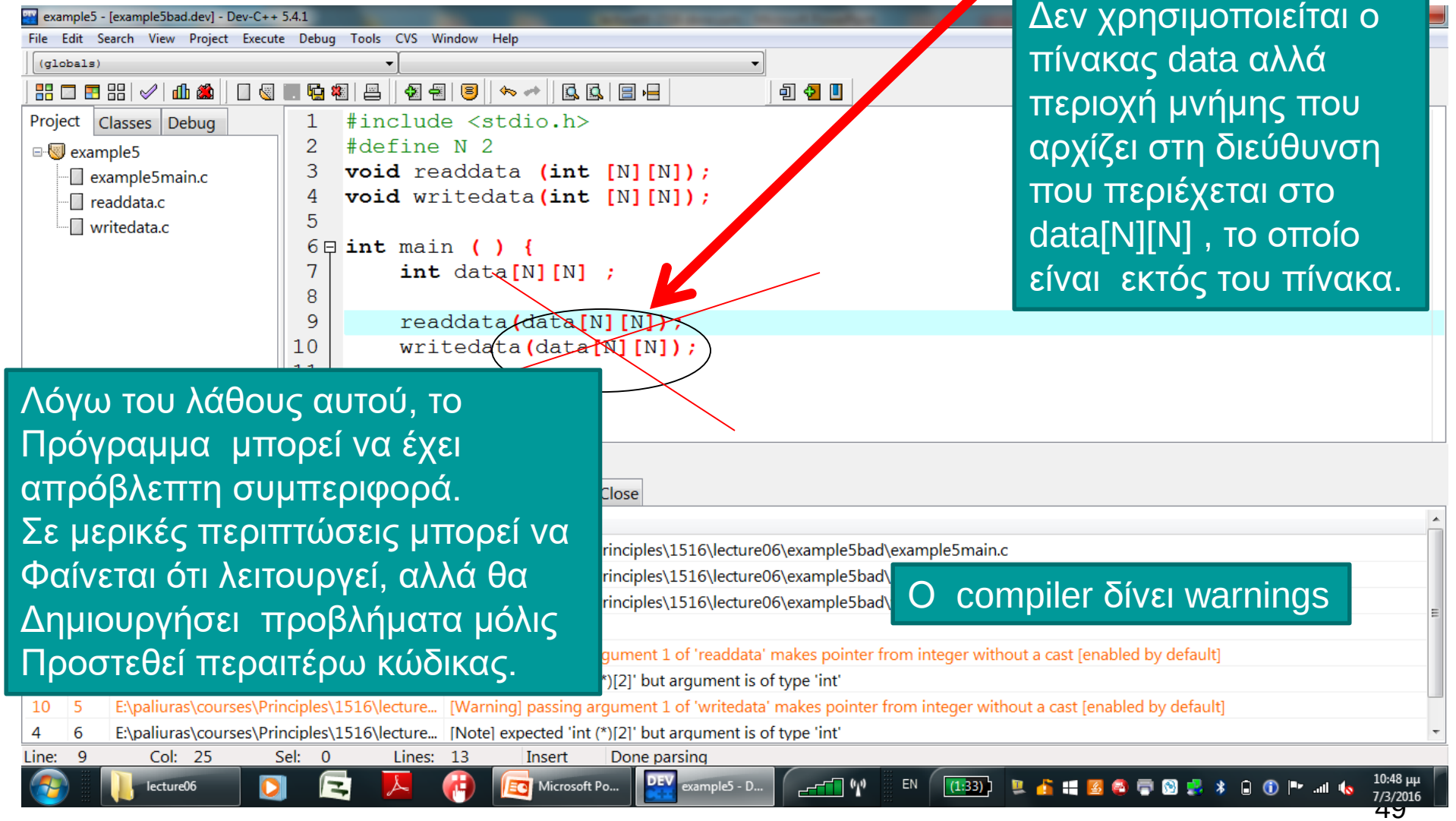
```
#define N 2
#include <stdio.h>
int main ( ) {
   int data[N][N] ;
   int i, j ;

   for (i =0 ; i < N ; i++)
    for ( j = 0 ; j < N ; j ++) {
      printf ("element (%d,%d)?\t", i, j);
      scanf("%d", &data[i][j]);
      }

   for (i =0 ; i < N ; i++) {
    for ( j = 0 ; j < N ; j ++)
       printf ("%d\t", data[i][j]);
     printf("\n");
   }
 return 0;
}
```

```c
#define N 2
#include <stdio.h>
void readdata(int [N][N]);
void writedata(int [N][N]);

int main ( ) {
   int data[N][N] ;

   readdata(data) ;
   writedata(data);

   return 0;
}
```

```c
void readdata(int a[N][N]) {
int i,j;
for (i =0 ; i < N ; i++)
   for ( j = 0 ; j < N ; j ++) {
     printf ("element (%d,%d)?\t", i, j);
     scanf("%d", &a[i][j]);
     }
}

void writedata(int b[N][N]) {
int i,j;
for (i =0 ; i < N ; i++) {
   for ( j = 0 ; j < N ; j ++)
        printf ("%d\t", b[i][j]);
   printf("\n");
 }
}
```

Καλύτερα!

ΠΡΟΣΟΧΗ!!! ΤΕΡΑΣΤΙΟ ΛΑΘΟΣ!!!

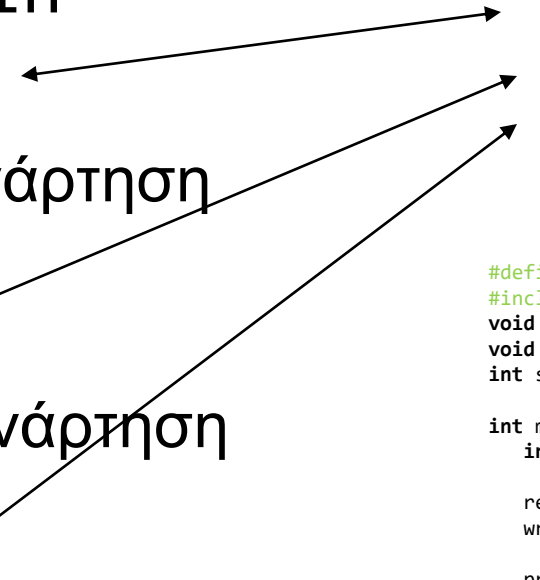


```c
#define N 2
#include <stdio.h>
void readdata(int a[N][N]);
void writedata(int b[N][N]);
int sumdata(int x[N][N]);

int main ( ) {
   int data[N][N] ;

   readdata(data) ;
   writedata(data);

   printf("The sum is: %d\n",
   sumdata(data));

return 0;
}
```

```c
int sumdata(int x[N][N]) {
   int i, j;
   int sum = 0;

   for (i=0; i<N; i++)
       for (j=0; j<N; j++)
              sum += x[i][j];

   return sum;
}
```

Στη μνήμη υπάρχει μόνο ένας πίνακας!

- Συνάρτηση `main`
  - Πίνακας `data`
  - Καλείται η συνάρτηση `readdata`
    - Όρισμα `a`
  - Καλείται η συνάρτηση `writedata`
    - Όρισμα `b`



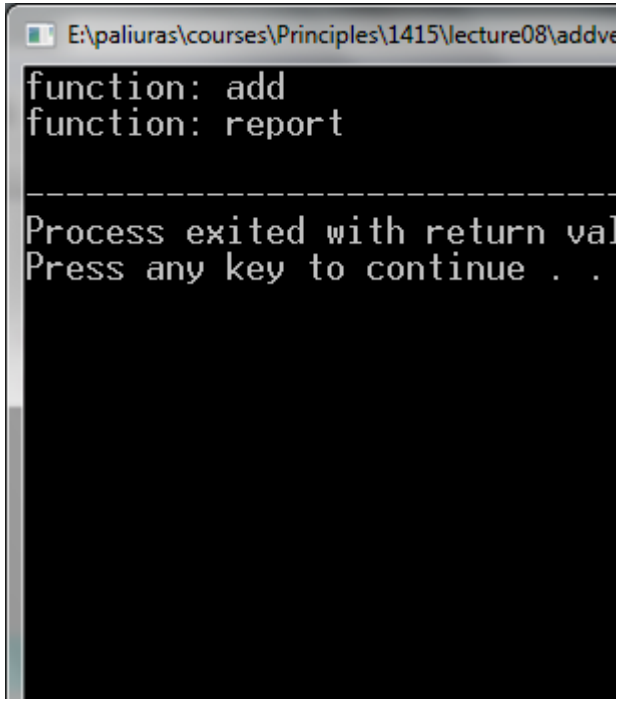
Πίνακας δεδομένων data

```c
#define N 2
#include <stdio.h>
void readdata(int a[N][N]);
void writedata(int b[N][N]);
int sumdata(int x[N][N]);

int main ( ) {
   int data[N][N] ;

   readdata(data) ;
   writedata(data);

   printf("The sum is: %d\n",
                sumdata(data));

return 0;
}
```

Δομημένη λεκτική περιγραφή

- Δήλωσε/αρχικοποίησε πίνακες `a, b, c`
- Άθροισε πίνακες `a, b` με αποτέλεσμα στον `c`
  - συνάρτηση `add`
- Εμφάνισε τον πίνακα `c`
  - συνάρτηση `report`

```c
#include <stdio.h>
#define N 5
void add (const int [], const int [], int []);
void report (const int []);

int main() {
        int a[N] = {1, 2, 3, 4, 5};
        int b[N] = {6, 7, 8, 9, 0};
        int c[N];

        add(a, b, c);
        report (c);

        return 0;
}
```



- Πλήρης main
- Κενές add, report

```c
void add(const int a[N], const int b[N], int c[N] ) {
        printf("add vectors\n");
        return ;
}

void report (const int c[N]) {
        printf("report\n");
        return ;
}
```

```
void add(const int a[N], const int b[N], int c[N] ) {
        int i;

        for (i=0; i<N; i++)
                c[i] = b[i] + a[i];

        return ;
}

void report (const int c[N]) {
        int i;

        for (i=0; i<N; i++)
                printf("%d ", c[i]);

        printf("\n");
        return ;
}
```

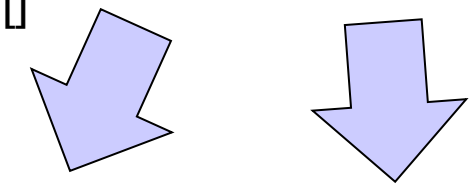
- Πλήρης υλοποίηση συναρτήσεων add, report

Μια προγραμματιστική τεχνική

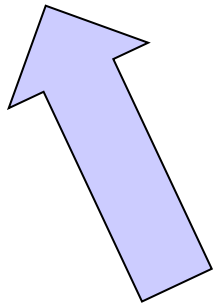
- ***Εξασφαλίζουμε*** ότι μια συνάρτηση μπορεί να αλλάξει τιμές πίνακα ***μόνο αν*** αναλυτικά το επιτρέψουμε.
- Εφαρμογή της αρχής ***ελαχίστου δικαιώματος*** *(principle of least privilege).*
- Χρήση τύπου **`const int [ ]`**

Πίνακες ως είσοδοι και έξοδοι

Είσοδοι: δεν επιτρέπεται στη συνάρτηση να αλλάξει τις τιμές στοιχείων πινάκων **const int** []



```
void add(const int a[N], const int b[N], int c[N] ) {
        int i;

        for (i=0; i<N; i++)
                c[i] = b[i] + a[i];

        return ;
}
```



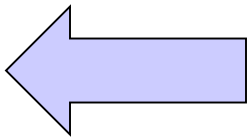
Έξοδος : η συνάρτηση έχει δικαίωμα να Αλλάξει τα στοιχεία του πίνακα **int** []

```
void report (const int c[N]) {
        int i;

        for (i=0; i<N; printf("%d ", c[i++]));

        printf("\n");
        return ;
}
```

- Άλλη υλοποίηση της report:
  - Η τρίτη έκφραση στο (κενό) **for** τυπώνει και αυξάνει το μετρητή με postfix increment

Γραμμή πίνακα 2D ως όρισμα συνάρτησης

```c
#include <stdio.h>
#define N 5

void fillrow(int x, int r[2*N]);
void printarray(int data[N][2*N]);

int main( void) {

        int data[N][2*N];
        int i, j;

        for (i=0; i<N; i++) {
                fillrow(i, data[i]);
        }

        printarray(data);

        return 0;
}
```



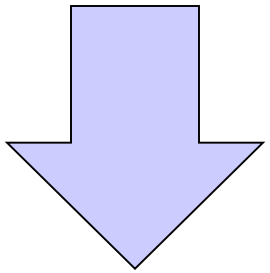
Επειδή data είναι τύπου int [N][2*N]

Το data[i] είναι τύπου int [2*N]

```
void fillrow(int x, int r[2*N]) {
   int i ;

   for (i=0; i<2*N; i++) {
     r[i] = x;
   }

   return ;
}


void printarray(int data[N][2*N]) {
  int i, j ;
  for (i = 0; i<N; i++) {
    for (j = 0; j<2*N; j++) {
      printf("%d ", data[i][j]);
      }
     printf("\n");
  }
  return ;
}
```

const qualifier σε ISO C

```
#include <stdio.h>
#define N 5

void fillrow(int x, int r[2*N]);
void printarray(int data[N][2*N]);

int main( void) {
        int data[N][2*N];

        /* άλλος κώδικας
        */

        printarray(data);

        return 0;
}
```



Χρειάζεται explicit type casting

```
printarray( (const int (*)[2*N]) data);
```

- Σε ISO C (c90,c99,c11,c17)

  error: pointers to arrays with different qualifiers are incompatible in ISO C

- Δεν ισχύει για gcc extensions, c++

```
void printarray(const int data[N][2*N]) {
  int i, j ;
  for (i = 0; i<N; i++) {
    for (j = 0; j<2*N; j++) {
      printf("%d ", data[i][j]);
      }
     printf("\n");
  }
  return ;
}
```

Μπορεί να απλοποιηθεί με typedef

ΣΥΝΗΘΗ ΛΑΘΗ

```
/* Σωστό !!! */
int main() {
        int a[N] = {1, 2, 3, 4, 5};
        int b[N] = {6, 7, 8, 9, 0};
        int c[N];

        add(a, b, c);
        report (c);

        return 0;
}
```

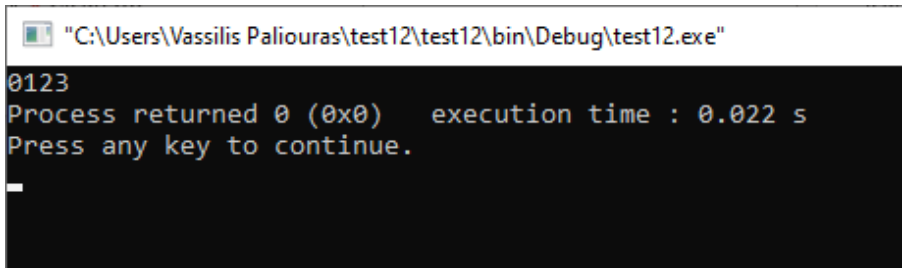
```
c = add(a, b); /* Λάθος: Το c δεν μπορεί να αλλάξει, είναι η
                  διεύθυνση του πρώτου στοιχείου του πίνακα! */

add(a[], b[], c[]); /* Λάθος: Εδώ είναι  syntax error. Μόνο σε
        δήλωση μπορεί να παραληφθεί μια (και μόνο μία) διάσταση (η
        τελευταία). */

add(a[N], b[N], c[N]); /* Λάθος: Η τιμή ενός ακεραίου (έξω από τους
πίνακες) μεταφράζεται σε διεύθυνση!!!  Warning: pointer from integer
without a cast */
```

Buffer overflow

```
#include <stdio.h>
#define N 3

int main ( ) {
int i;

int a[N];

for (i=0; i<N + 1; i++) {
        a[i] = i;
        printf("%d",a[i]);

    }

return 0;
}
```

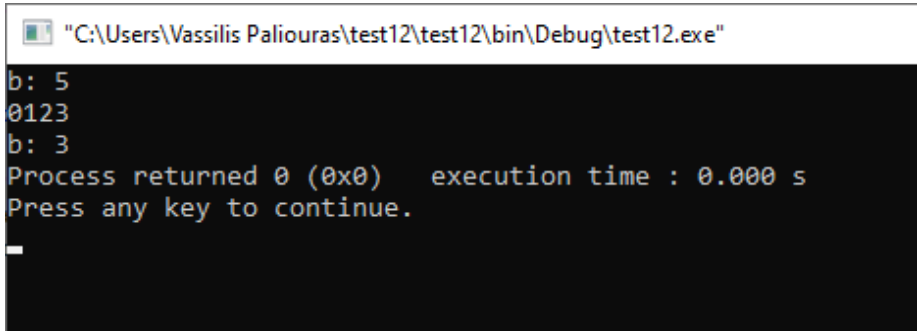
Παραβιάζει το όριο του πίνακα (γράφει σε N+1) στοιχεία

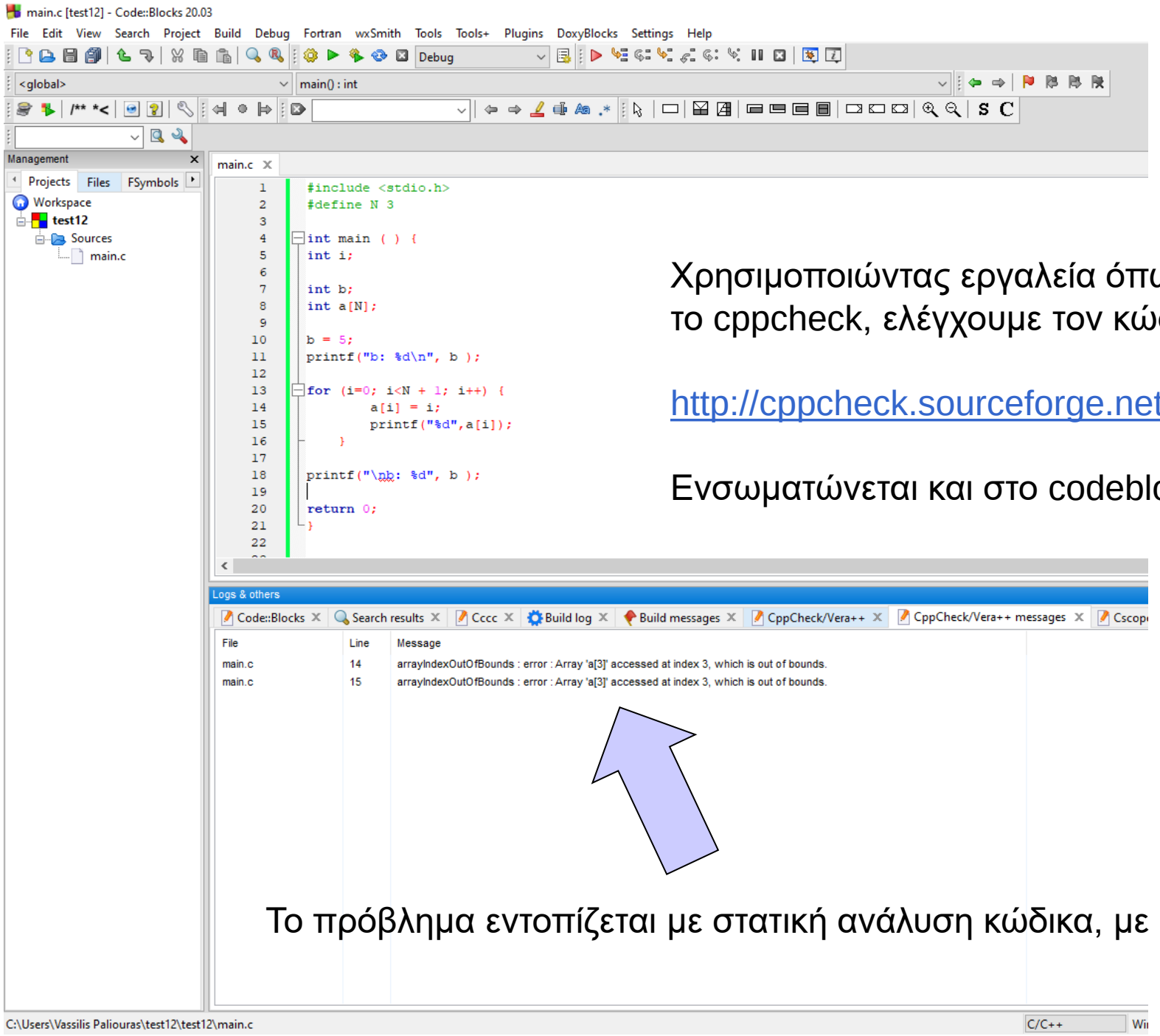
Φαίνεται ότι «δουλεύει»…



```c
#include <stdio.h>
#define N 3

int main (void ) {
        int i;

        int b;
        int a[N];

        b = 5;
        printf("b: %d\n", b );

        for (i=0; i<N + 1; i++) {
                a[i] = i;
                printf("%d",a[i]);
        }

        printf("\nb: %d", b );

return 0;
}
```

Αλλά μπορεί να γράψει πάνω σε άλλες μεταβλητές!





Χρησιμοποιώντας εργαλεία όπως το cppcheck, ελέγχουμε τον κώδικα

http://cppcheck.sourceforge.net/

Ενσωματώνεται και στο codeblocks

Το πρόβλημα εντοπίζεται με στατική ανάλυση κώδικα, με cppcheck

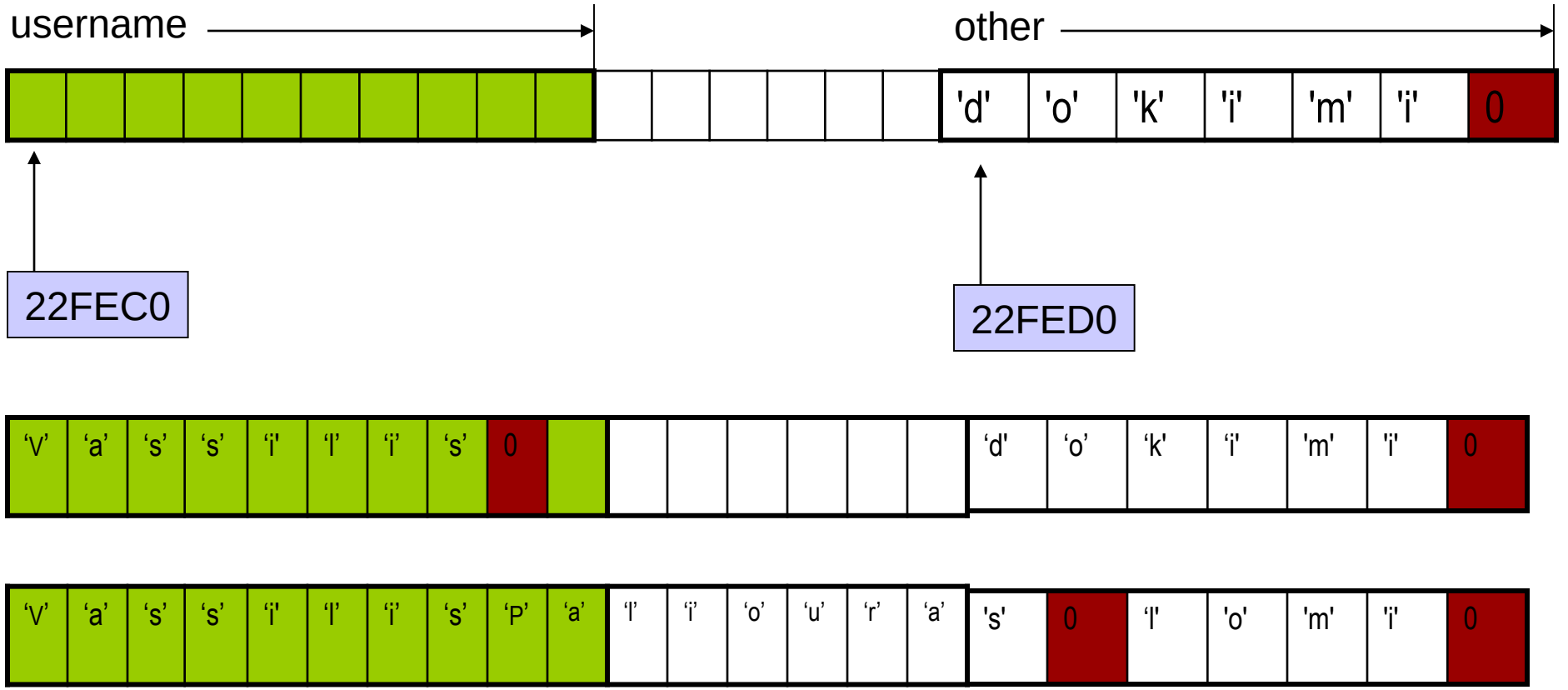
Βασικός τύπος **char**

```c
#include <stdio.h>

int main() {

        char a;

        a = 'g';

        printf("this is a char: %c\n", a);

        return 0;
}
```

- Διαφορετική σημασία απλών / διπλών εισαγωγικών στη C
  - Απλά εισαγωγικά => ένας χαρακτήρας
  - Διπλά εισαγωγικά => ακολουθία χαρακτήρων που τερματίζεται με την αξία 0

- Χαρακτήρας και κώδικας ascii

Σχηματικά η μνήμη – buffer overflow



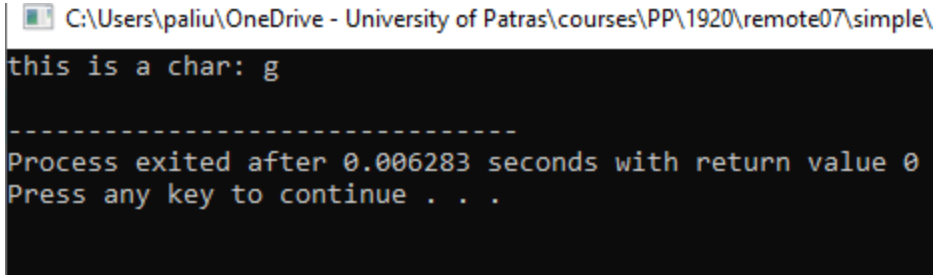
Καταστρέφονται τα περιεχόμενα της other!

Οι ακριβείς θέσεις των πινάκων μπορεί να είναι διαφορετικές σε διαφορετικά περιβάλλοντα, το πρόβλημα είναι το ίδιο.

```
Please enter user name: VassilisPaliouras
Hello, VassilisPaliouras
Your name is 17 letters long stored at 61FE0F
Value of other: liouras at 61FE19
Process returned 0 (0x0)   execution time : 7.772 s
Press any key to continue.
```

Βασικός τύπος **char**

```c
#include <stdio.h>

int main() {

        char a;

        a = 'g';

        printf("this is a char: %c\n", a);

        return 0;
}
```

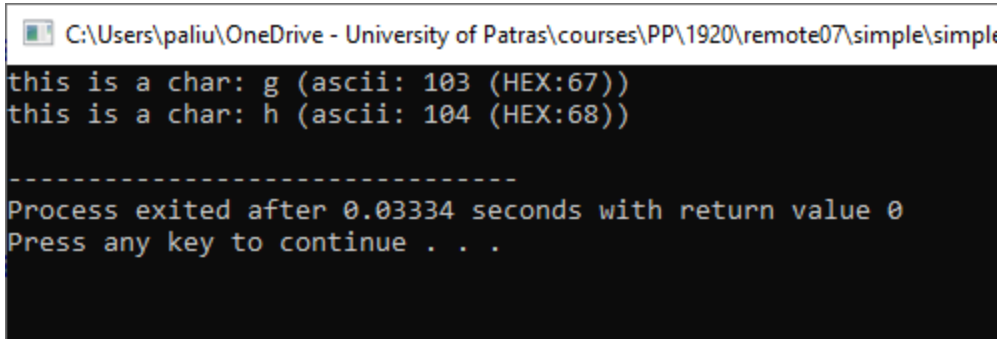


- Διαφορετική σημασία απλών / διπλών εισαγωγικών στη C
  - Απλά εισαγωγικά $\Rightarrow$ ένας χαρακτήρας
  - Διπλά εισαγωγικά $\Rightarrow$ ακολουθία χαρακτήρων που τερματίζεται με την αξία 0

- Χαρακτήρας και κώδικας ASCII

Βασικός τύπος **char** και ASCII

```c
#include <stdio.h>

int main() {

        char a;

        a = 'g';

        printf("this is a char: %c (ascii: %d (HEX:%X))\n", a, a, a);

        a = '\x68';

        printf("this is a char: %c (ascii: %d (HEX:%X))\n", a, a, a);

        return 0;
}
```



Αλφαριθμητικά (strings)

πρόκειται για **πίνακες χαρακτήρων**:

- `char name[30];`

- αρχικοποίηση με
  `char name[30] = "abcd";`
  το οποίο ισοδυναμεί με

```
name[0] = 'a';
name[1] = 'b';
name[2] = 'c';
name[3] = 'd';
name[4] = 0 ; /* δηλώνει το τέλος ενός αλφαριθμητικού */
```

- `'\0' == (char) 0`

Ανάγνωση και εκτύπωση αλφαριθμητικού

- `char str[N_MAX];`
- `scanf ("%s", str );`
- `printf ("%s\n", str );`

- `%s` → αντιστοιχεί σε αλφαριθμητικό
- `str[0]` είναι ο πρώτος χαρακτήρας
- `str` είναι η **διεύθυνση του πρώτου στοιχείου**
  - str είναι το **ίδιο** με &str[0]
  - ισχύει για κάθε τύπο πίνακα

Παράδειγμα

- Το σύστημα ζητά από το χρήστη το όνομά του και τυπώνει "hello" ακολουθούμενο από το όνομα του χρήστη.

Υλοποίηση σε C

```
#include <stdio.h>
#include <string.h>
#define N 10

int main (  ) {

 char username[N];

 printf("Please enter user name: ");
 scanf("%s", username);

 printf("Hello, %s\n", username);
 printf("Your name is %d letters long", strlen(username));


return  0;
}
```

το όνομα του πίνακα είναι η διεύθυνση του πρώτου στοιχείου (⇒ δεν βάζουμε &)

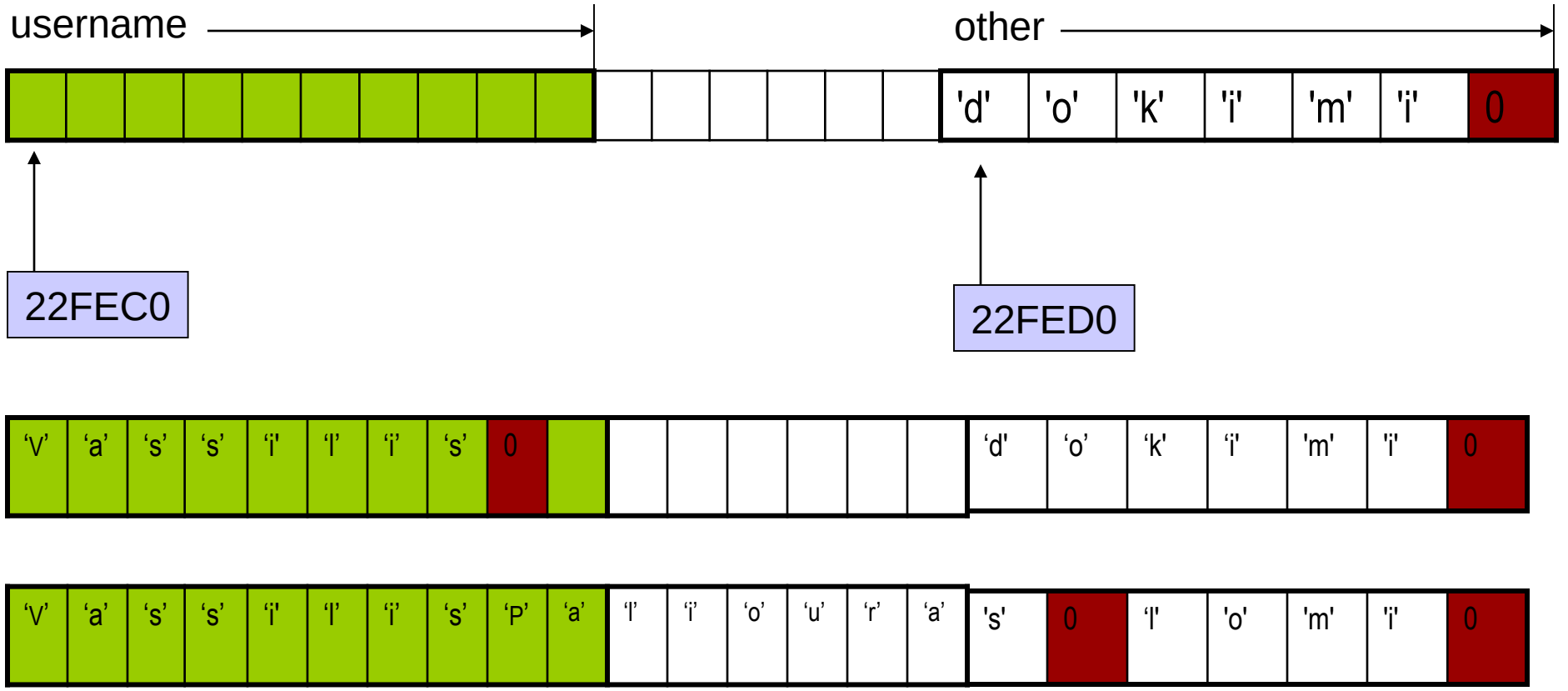
συνάρτηση βιβλιοθήκης strlen()

Υπάρχουν όρια;

```
#include <stdio.h>
#include <string.h>
#define N 10

int main (  ) {

 char other[ ] = "dokimi";
 char username[N];

 printf("Please enter user name: ");
 scanf("%s", username);

 printf("Hello, %s\n", username);
 printf("Your name is %d letters long stored at %X\n", strlen(username), username);

printf("Value of other: %s at %X", other, other);

return 0;
}
```

Ναι, αλλά δεν γίνεται έλεγχος...

Σχηματικά η μνήμη – buffer overflow



Καταστρέφονται τα περιεχόμενα της other!

Επεξεργασία ανά χαρακτήρα

```c
#include <stdio.h>
int main()
{
   char string1[ 20 ], string2[] = "string literal";
   int i;

   printf(" Enter a string: ");
   scanf( "%s", string1 );
   printf( "string1 is: %s\nstring2: is %s\n"
           "string1 with spaces between characters is:\n",
           string1, string2 );

   for ( i = 0; string1[ i ] != '\0'; i++ ) {
      printf( "%c ", string1[ i ] );
      }

   printf( "\n" );
   return 0;
}
```

```
Hello
hello

Process returned 0 (0x0)   execution time : 0.011 s
Press any key to continue.
```

```c
#include <stdio.h>
void DisplayName(char []);

int main ( ) {
  char astring[10] = "Hello";

  DisplayName(astring);
  DisplayName(astring);

  return 0;
}
void DisplayName(char x[]) {
  int i;
  for (i=0; x[i]!=0 ; i++) {
    printf("%c", x[i]); }
 x[0]='h';
 printf("\n");
}
```

```c
#include <stdio.h>
void DisplayName(const char []);

int main ( ) {
  char astring[10] = "Hello";

  DisplayName(astring);
  DisplayName(astring);

  return 0;
}
void DisplayName(const char x[]) {
 int i;
 for (i=0; x[i]!=0 ; i++) {
    printf("%c", x[i]); }
x[0]='h';
printf("\n");
}
```

Error: assignment of read-only location

char και κώδικας ascii

```c
#include <stdio.h>

int main(void) {
   char alphabet[27]; /* 26 letters plus trailing zero */
   char c;

   for (c='A'; c<='Z'; c++) {
        alphabet[c-'A'] = c;
        }

   alphabet[c-'A'] = 0;

   printf("%s", alphabet);

   return 0;
}
```

- Σε μεταβλητές τύπου **char** αποθηκεύεται ο ascii κώδικας πάντα ως ακέραιος του ενός byte.
- Μπορούμε να κάνουμε αριθμητικές με τους ακέραιους αυτούς
- Εδώ βάζουμε αναλυτικά το τελικό 0 στο αλφαριθμητικό alphabet

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```