Διαδικαστικός Προγραμματισμός

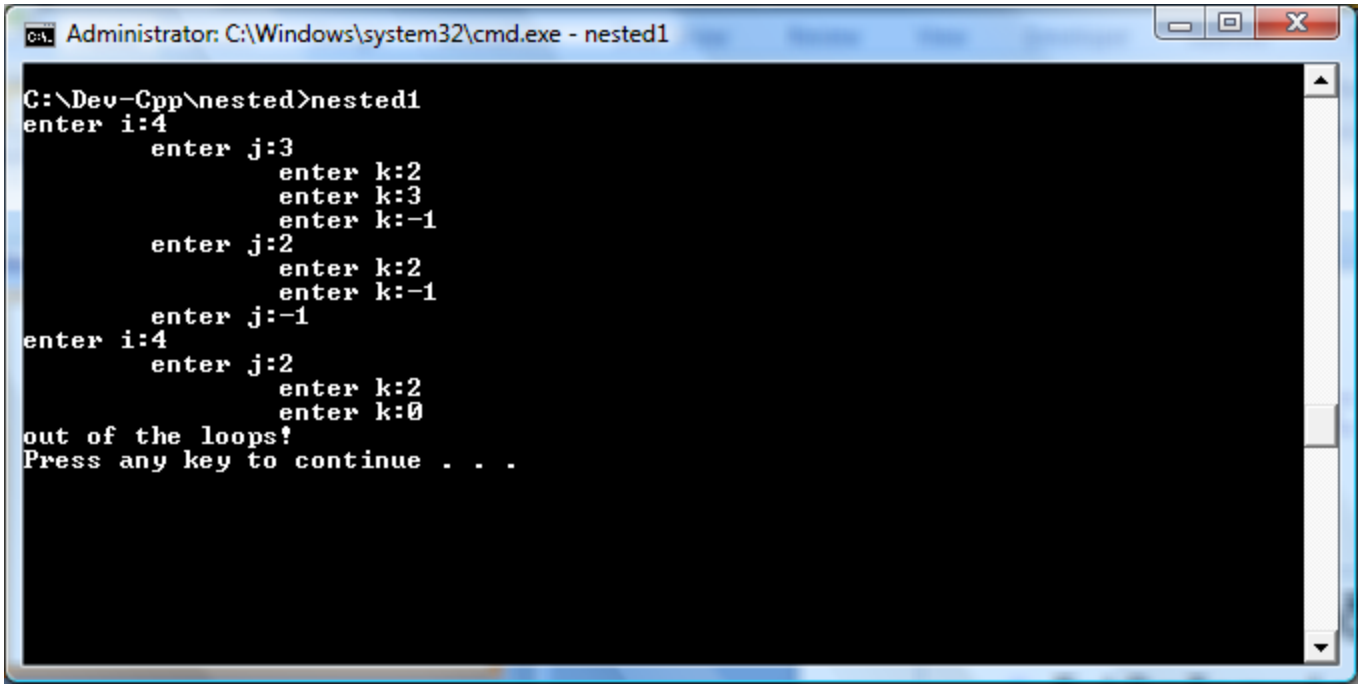
Βασίλης Παλιουράς

Μέχρι τώρα

- Οργάνωση προγράμματος στη C
  - συναρτήσεις
- Μεθοδολογίες σχεδιασμού προγραμμάτων
  - Top-down
  - Λεκτική περιγραφή (προστακτικός τρόπος)
  - Αυξητική ανάπτυξη προγράμματος
- Αναγνωριστές, Τελεστές, Εκφράσεις, Προτάσεις
- Βρόχοι επανάληψης στη C
  - `do { σύνθετη εντολή; } while (έκφραση) ;`
  - `while (έκφραση) { σύνθετη εντολή;}`
  - `for (έκφραση1; έκφραση2;έκφραση3)`
    `{σύνθετη εντολή}`

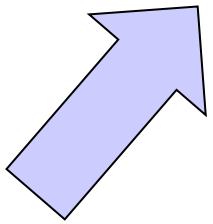
Παράδειγμα 4: Ένθετοι βρόχοι επανάληψης

- 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:\Users\Vassilis Paliouras\test12\test12\bin\Debug\test12.exe"
enter i:1
        enter j:2
                enter k:-1
        enter j:-1
enter i:-1
out of the loops!

Process returned 0 (0x0)   execution time : 9.310 s
Press any key to continue.
```

```
"C:\Users\Vassilis Paliouras\test12\test12\bin\Debug\test12.exe"
enter i:1
        enter j:2
                enter k:3
                enter k:4
                enter k:-1
        enter j:2
                enter k:3
                enter k:0
out of the loops!

Process returned 1 (0x1)   execution time : 12.993 s
Press any key to continue.
```

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

int main( )  {
  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 το χρήστη **να επιλέξει μεταξύ τριών επιλογών**:
  - να ξεκινήσει μια συγκεκριμένη διεργασία,
  - να σταματήσει η διεργασία,
  - να λήξει η εκτέλεση του προγράμματος.
- Θα ζητείται είσοδος από το χρήστη **έως ότου επιλεγεί η λήξη** του προγράμματος.

Παράδειγμα – Λεκτική περιγραφή λύσης

- Ζήτησε από το χρήστη **να επιλέξει μεταξύ τριών επιλογών**: `int userchoice;`
  - 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.

Παράδειγμα – Λεκτική περιγραφή λύσης (2)

- 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.

Λεκτική περιγραφή λύσης

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

Οργάνωση βασικού βρόχου (1)

```
int main ( ) {
    int userchoice ;

    userchoice = getchoice ( ) ;

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

  return 0;
}
```

```
int main ( ){
int userchoice ;

userchoice = getchoice();

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

   userchoice=getchoice();

   }

return 0;
}
```

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

κλήση (⇒ χρήση)

(πχ στην υλοποίηση της main())

userchoice = getchoice( );

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

int main (){

    int userchoice ;

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

   return 0;
}
```

```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...");
}

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

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

int main (){

    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]

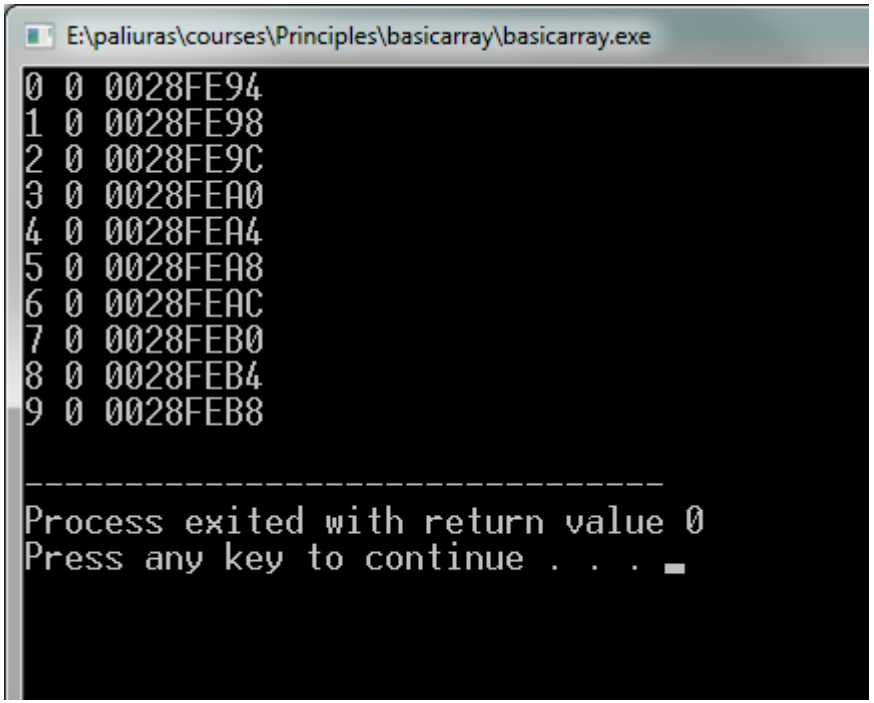
Παράδειγμα

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

   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;
}
```



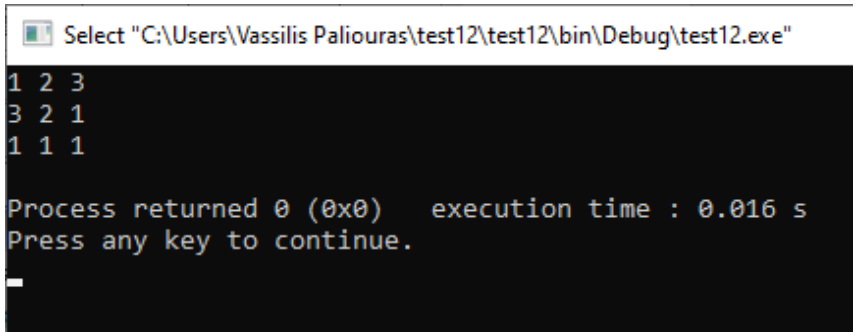
0	1	2	3	4	5	6	7	8	9

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

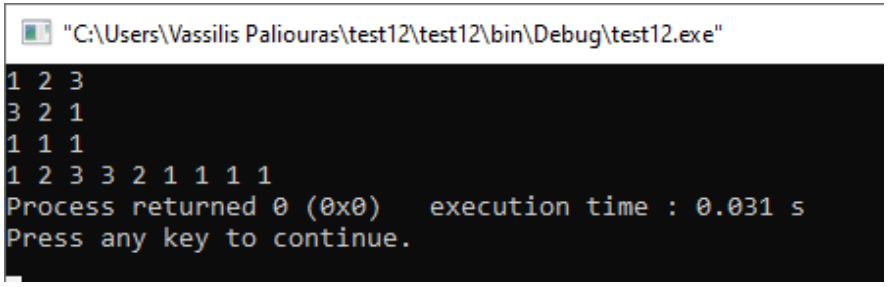
1	2	3
3	2	1
1	1	1

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 ( ) {
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);
```

τελεστής διεύθυνσης (&):

Επιστρέφει τη διεύθυνση της θέσης μνήμης η οποία αντιστοιχεί στη μεταβλητή που ακολουθεί

Παράδειγμα

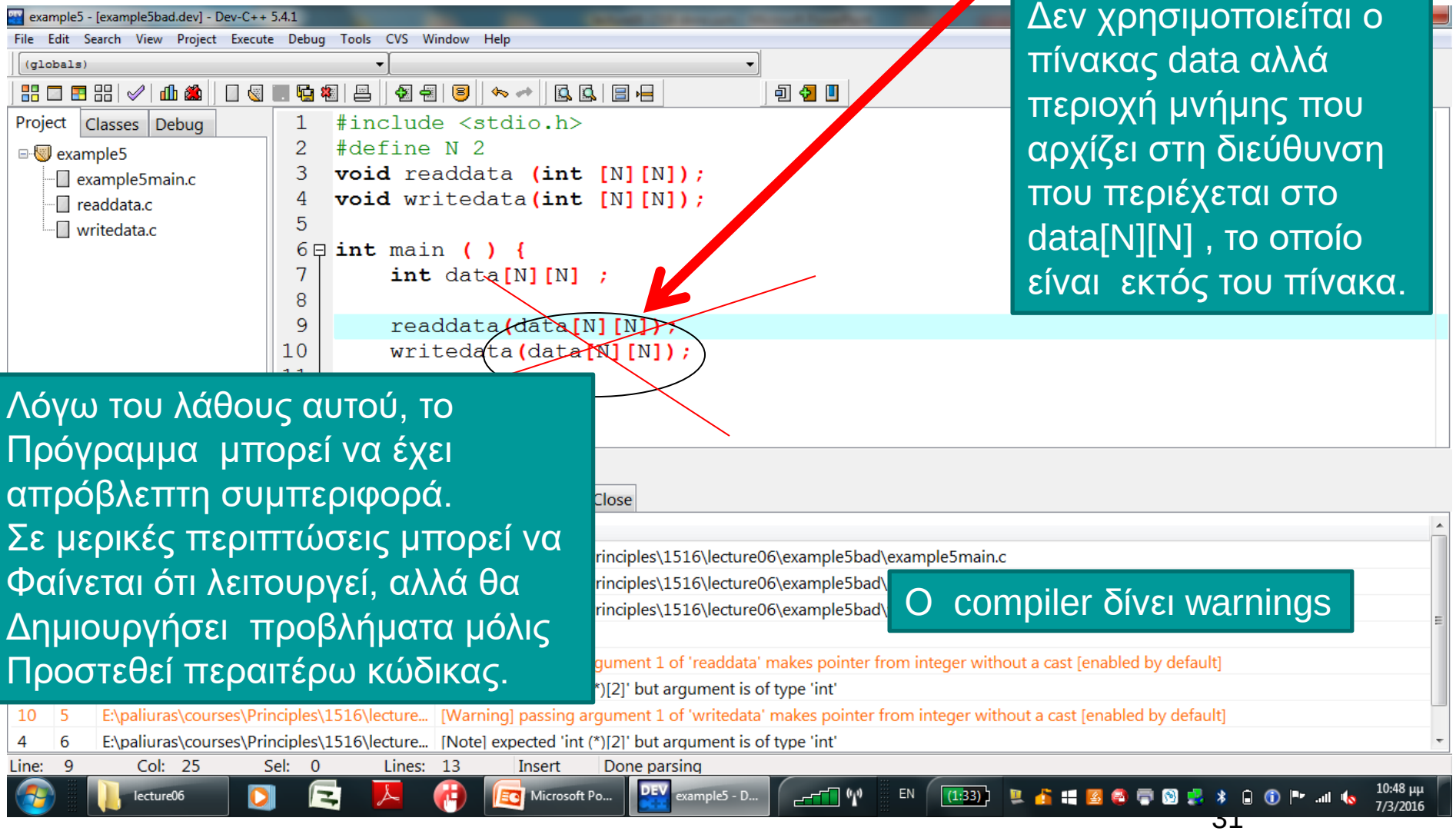
```
#define N 2
#include <stdio.h>
int main ( ) {
   int data[N][N] ;
   int i, j ;

   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!!!



Δεν χρησιμοποιείται ο πίνακας data αλλά περιοχή μνήμης που αρχίζει στη διεύθυνση που περιέχεται στο data[N][N] , το οποίο είναι εκτός του πίνακα.

Λόγω του λάθους αυτού, το Πρόγραμμα μπορεί να έχει απρόβλεπτη συμπεριφορά.
Σε μερικές περιπτώσεις μπορεί να Φαίνεται ότι λειτουργεί, αλλά θα Δημιουργήσει προβλήματα μόλις Προστεθεί περαιτέρω κώδικας.

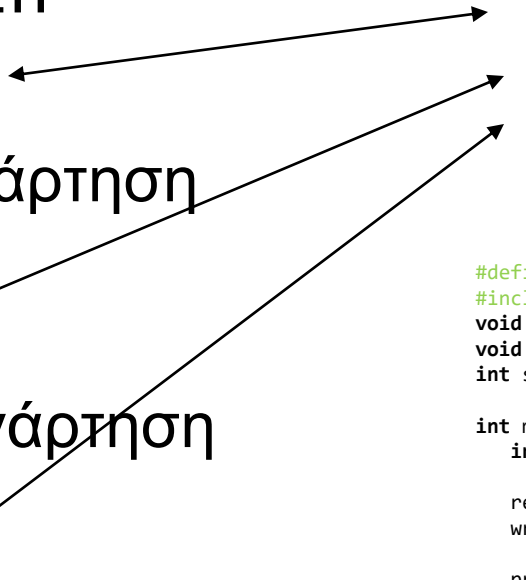
Ο compiler δίνει warnings

```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`



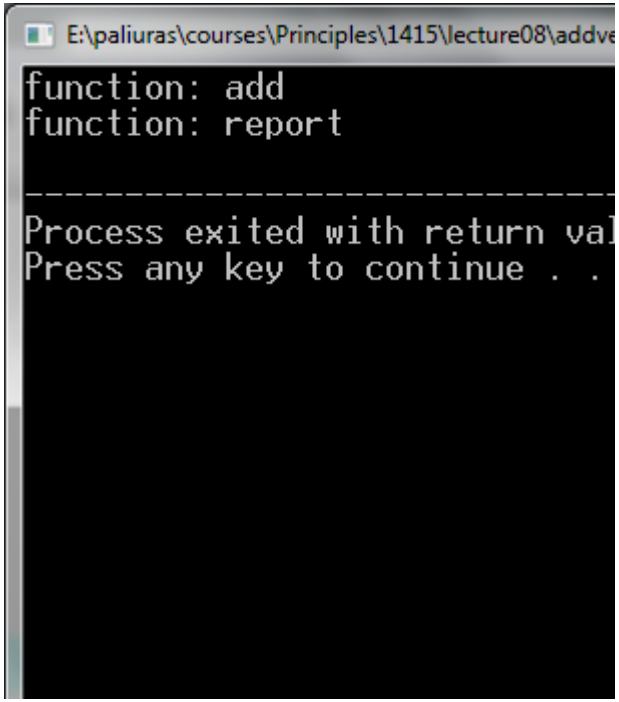
Πίνακας δεδομένων data

```
#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

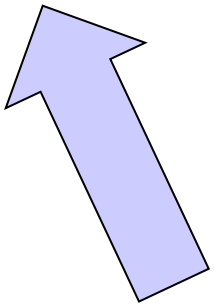
Μια προγραμματιστική τεχνική

- ***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

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;
}
```

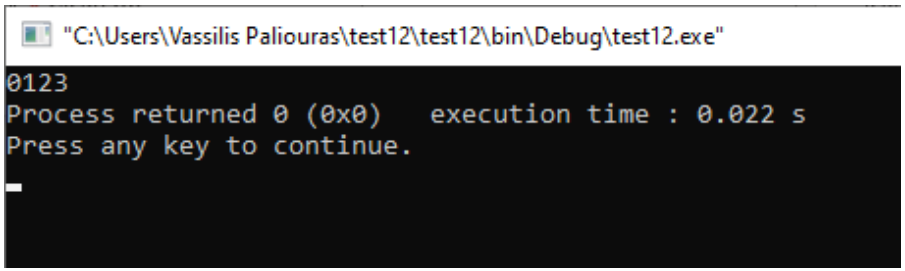
```
c = add(a, b); /* Λάθος: Το c δεν μπορεί να αλλάξει, είναι η
                  διεύθυνση του πρώτου στοιχείου του πίνακα! */

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;
}
```

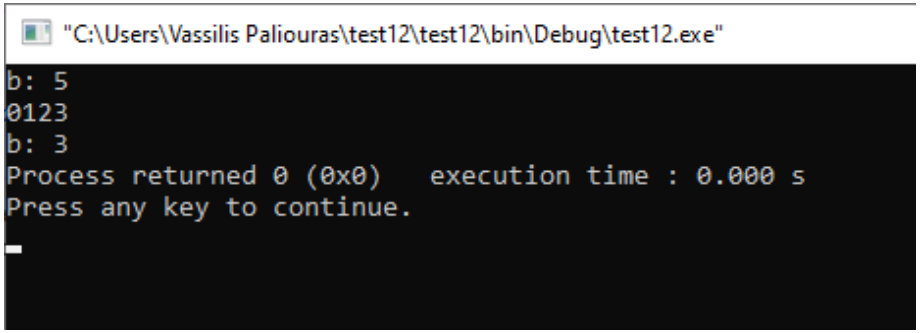
Παραβιάζει το όριο του πίνακα (γράφει σε N+1) στοιχεία

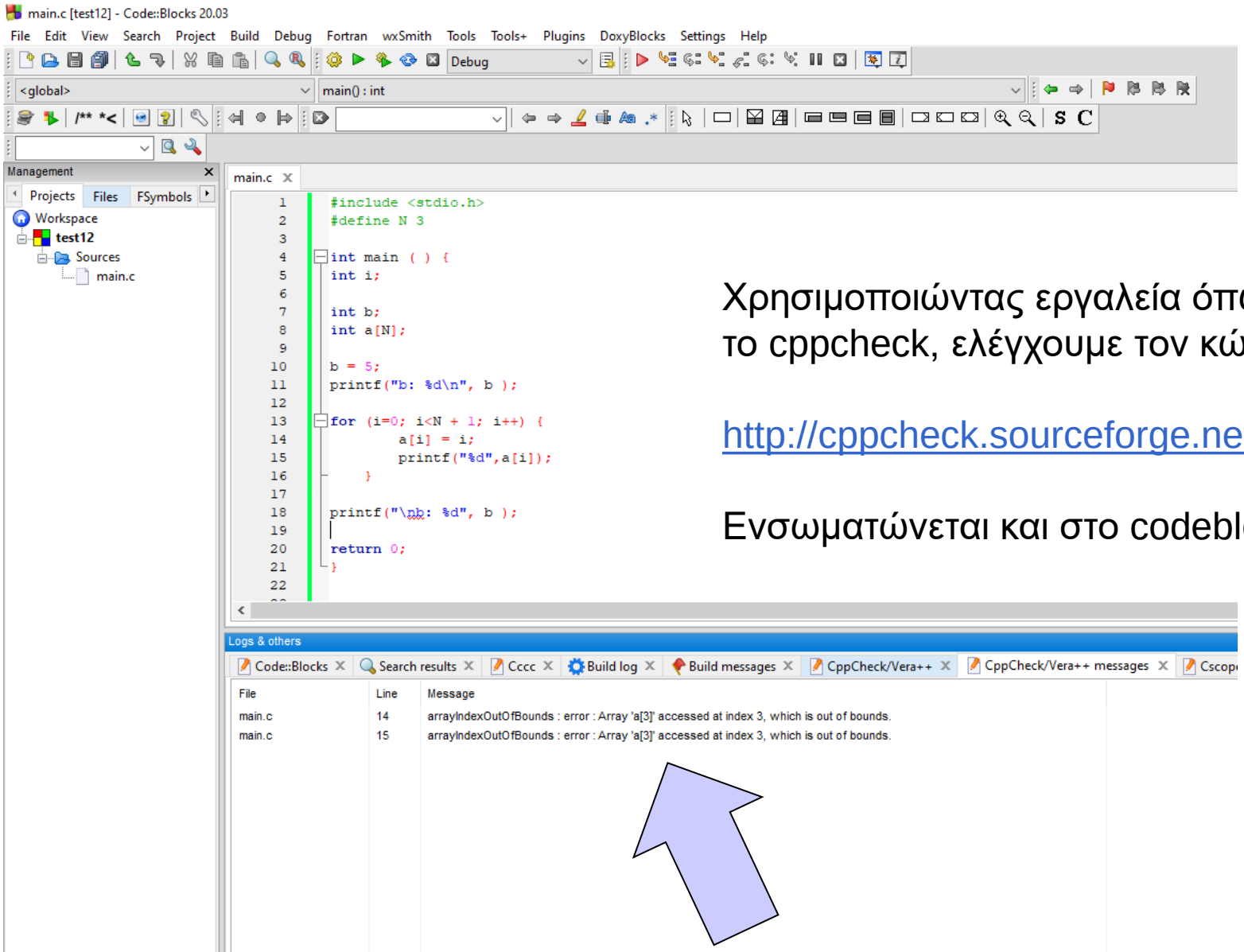
Φαίνεται ότι «δουλεύει»…



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

int main ( ) {
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