Διαδικαστικός Προγραμματισμός

Βασίλης Παλιουράς

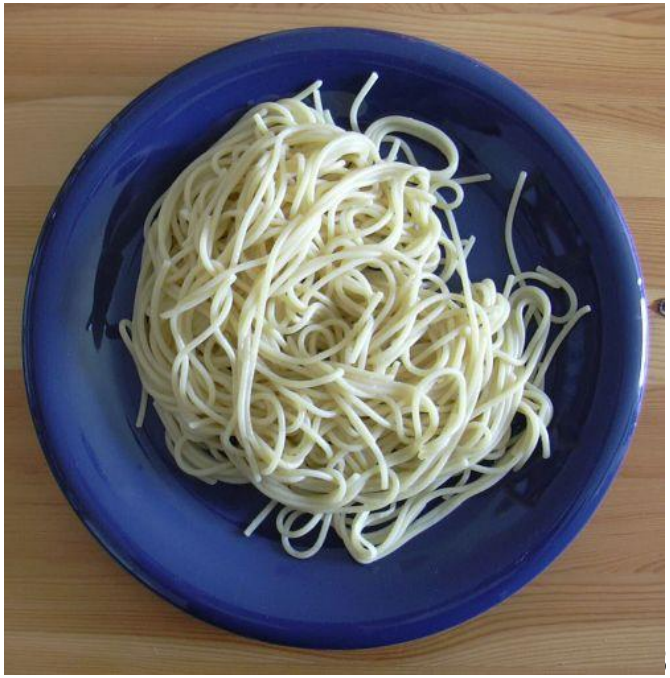
Μέχρι τώρα

- 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}`

Δομημένος προγραμματισμός

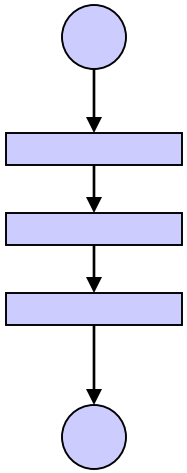
```c
#include <stdio.h>

int main() {
    int i ;
    for (i=1; i<=10; i++)
        printf ("%d %d\n", i, i *i);

    return 0;
}
```

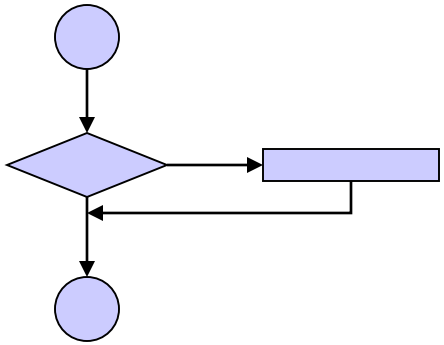
```c
#include <stdio.h>

int main() {

    int i = 0;
    next:
    i = i + 1;
    printf ("%d %d\n", i, i *i);
    if (i>=10) goto end;
    goto next;
    end:
    return 0;
}
```

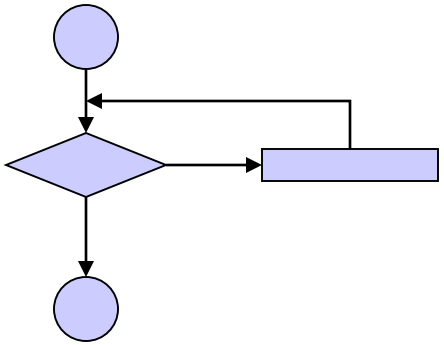


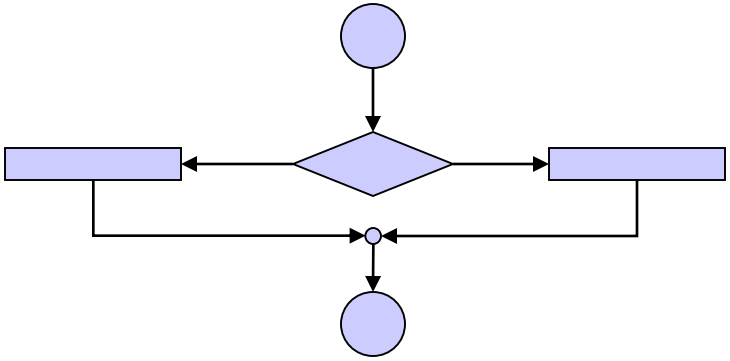
Spaghetti code

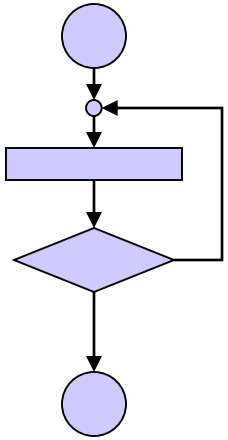
Σύνοψη δομών ελέγχου











Δομημένος προγραμματισμός – Στοιχεία ελέγχου ροής προγράμματος

- Προγραμματισμός χωρίς χρήση **goto**
  - '**goto**-less' programming
  - γνωστό ότι είναι εφικτό από τη δεκαετία του 60
  - στη βιομηχανία, αποδειγμένος ως αποδοτικός
- Αρκεί η υποστήριξη επιλογών ροής
  - απλή, διπλή, πολλαπλή,
  - βρόχοι υπό συνθήκη (do..while, while)

C. Bohm and C. Jacopini, "Flow diagrams, Turing machines, and languages with only two formation rules," *Comms. of ACM*, vol. 9, no. 5, May 1966, pp. 336 – 371.

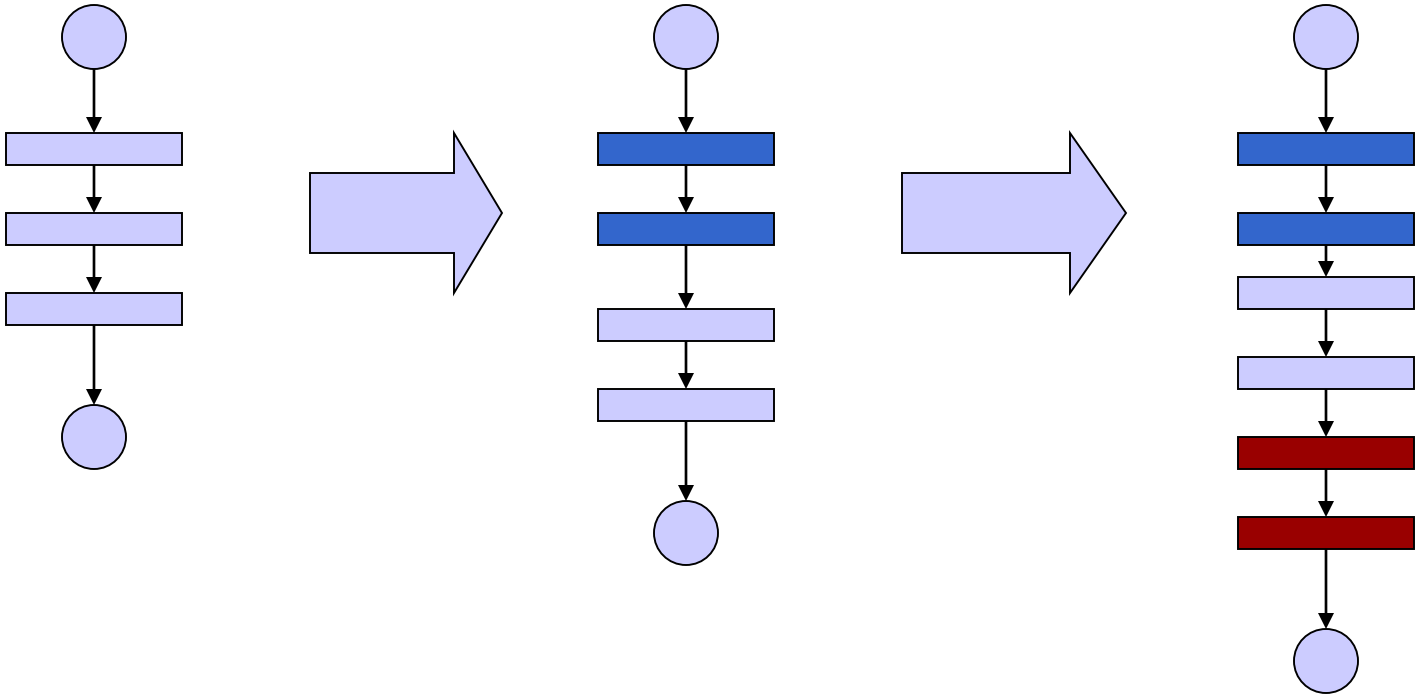
Κανόνες κατασκευής δομημένου προγράμματος

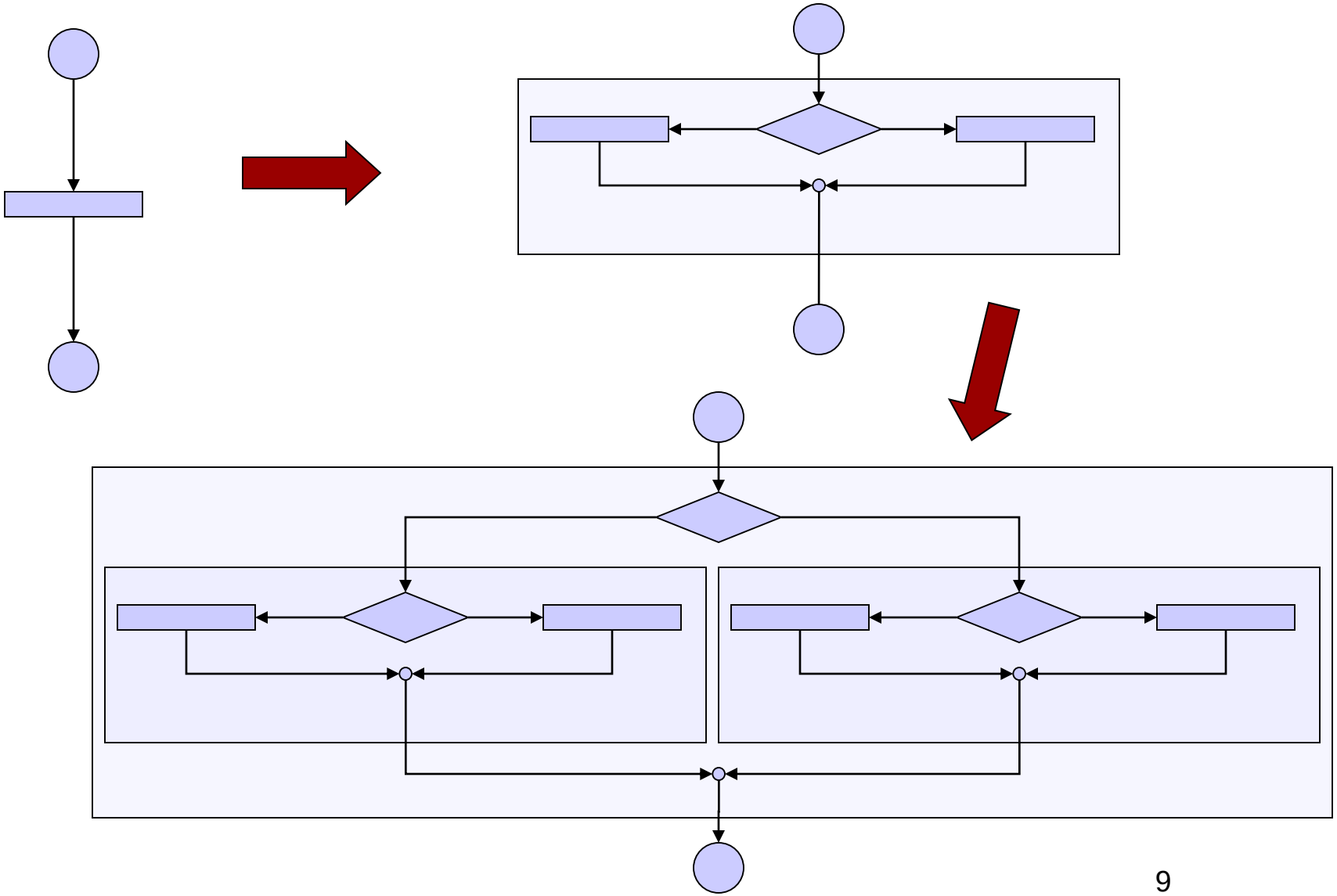
1. Αρχή: Το **απλούστερο** διάγραμμα ροής
2. Κάθε διεργασία μπορεί να αντικατασταθεί από δύο διεργασίες σε σειρά
3. Κάθε διεργασία μπορεί να αντικατασταθεί από οποιαδήποτε από τις δομές ελέγχου
4. Οι κανόνες 2 και 3 εφαρμόζονται με όποια σειρά και όσες φορές απαιτείται.

Παρατηρήσεις

- Δεν περιορίζεται σε διαγράμματα ροής
- Λεκτική περιγραφή – ψευδο-κώδικας
- Progressive refinement: Σταδιακά προσθέτουμε λεπτομέρειες υλοποίησης
- Αυξητική ανάπτυξη κώδικα

Προσθέτω λεπτομέρεια στο αρχικό διάγραμμα ροής (κανόνας 2)





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

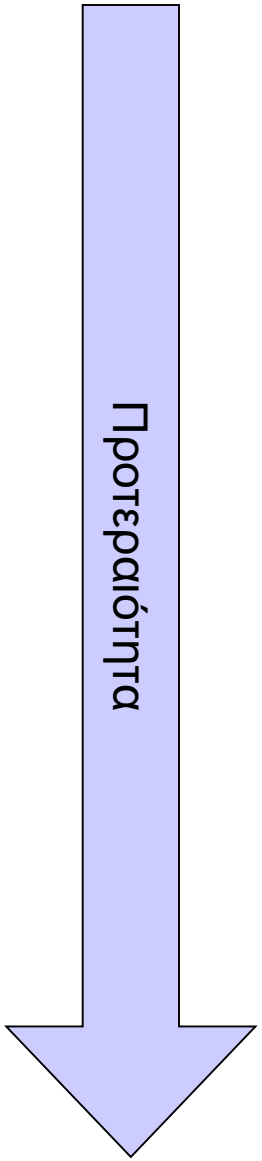
```c
#include <stdio.h>
#include <stdbool.h>
#include <iso646.h>
// C99 style for comments and boolean logic

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

Τελεστές ⇒ Ενέργειες σε δεδομένα

Τελεστές	Προσεταιριστικότητα
`( ) [ ] -> .`	Αριστερά προς δεξιά
`! ~ ++ -- + - * & sizeof`	Δεξιά προς αριστερά
`* / %`	Αριστερά προς δεξιά
`+ -`	Αριστερά προς δεξιά
`<< >>`	Αριστερά προς δεξιά
`< <= > >=`	Αριστερά προς δεξιά
`== !=`	Αριστερά προς δεξιά
`&`	Αριστερά προς δεξιά
`^`	Αριστερά προς δεξιά
`|`	Αριστερά προς δεξιά
`&&`	Αριστερά προς δεξιά
`||`	Αριστερά προς δεξιά
`?:`	Δεξιά προς αριστερά
`= += -= *= /= %= &= ^= |= <<= >>=`	Δεξιά προς αριστερά
`,`	Αριστερά προς δεξιά



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 ;
}
```

Prefix – postfix notation

- a++;
- ++a;
- f(a++);
- f(++a);

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

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

```
#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;
}
```

Short-circuit evaluation

Παραδείγματα χρήσης δομών ελέγχου

1. Σταθερός αριθμός επαναλήψεων
2. Αριθμός επαναλήψεων εξαρτώμενος από τα δεδομένα
3. Ένθετες (nested) δομές ελέγχου

Παράδειγμα 1 – καθορισμένος αριθμός επαναλήψεων

- Να γραφεί ένα πρόγραμμα που διαβάζει **δέκα** **ακεραίους**, έναν κάθε φορά και τυπώνει το **μερικό άθροισμα**.

- **Στο τέλος** τυπώνεται το συνολικό άθροισμα και το γινόμενό τους.

- (Εδώ λύση χωρίς πίνακες).

Λεκτική περιγραφή

- Επανάλαβε για δέκα φορές {
- Διάβασε το num → scanf()
- Υπολόγισε το sum →computeSum() ή
  →sum = sum + num
- Τύπωσε το sum → printf()
- }

```c
#include <stdio.h>

int main() {

  int i, num, sum=0;

  for (i=0; i<10; i++) {
   scanf("%d", &num);
   sum = sum + num;
   printf("partial sum: %d\n", sum);
   }

  printf("total: %d", sum);

return 0;
}
```

```c
#include <stdio.h>
#define N 10
int main() {

  int i, num, sum=0;

  for (i=0; i<N; i++) {
   scanf("%d", &num);
   sum = sum + num;
   printf("partial sum: %d\n", sum);
   }

  printf("total: %d", sum);

return 0;
}
```

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;
}
```

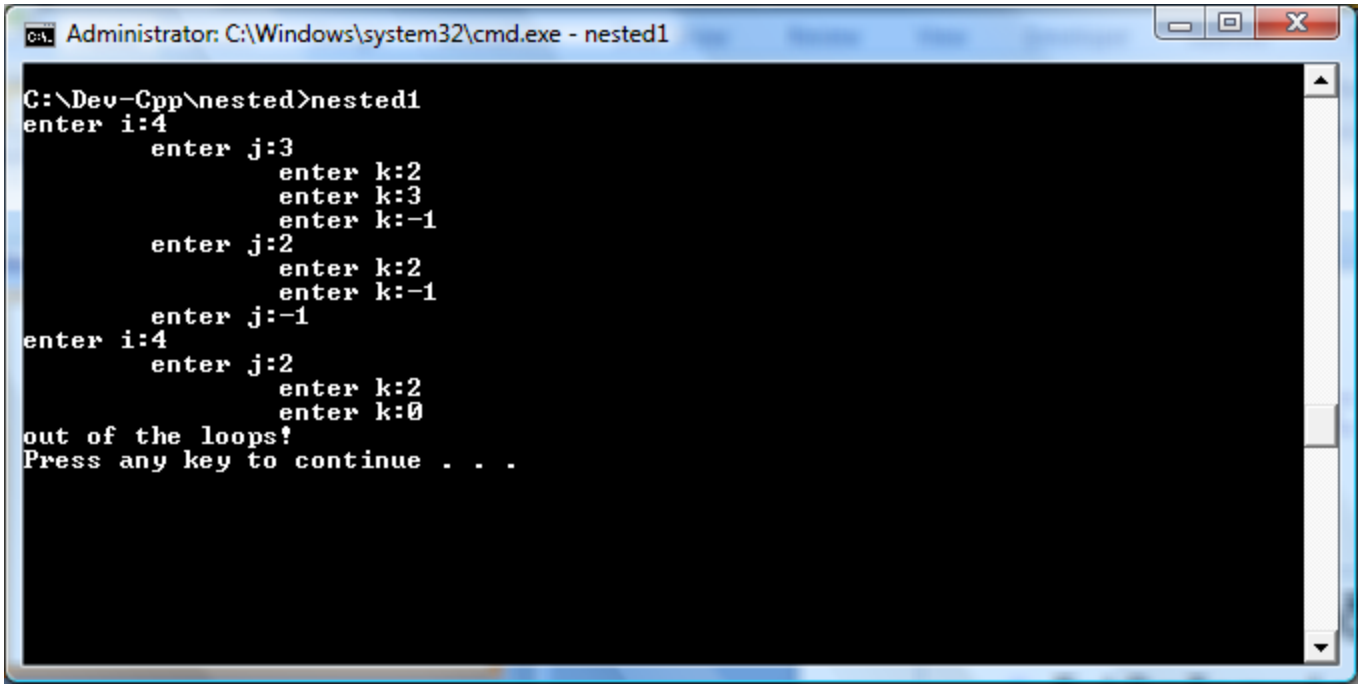
λογική σύζευξη (ΚΑΙ, AND)

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;
}
```

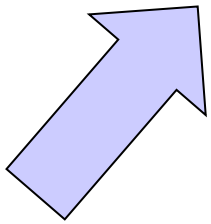
Παράδειγμα 4: Ένθετοι βρόχοι επανάληψης

- 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;
}
```

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...");
}
```

Πίνακες στη C

- Δεσμεύουν ***συνεχή χώρο*** στη μνήμη

- Οι ακόλουθες δηλώσεις οδηγούν τον compiler να δημιουργήσει διαφορετική assembly.

```
int a[32];

int s = 32;
int a[s];
```

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;
}
```

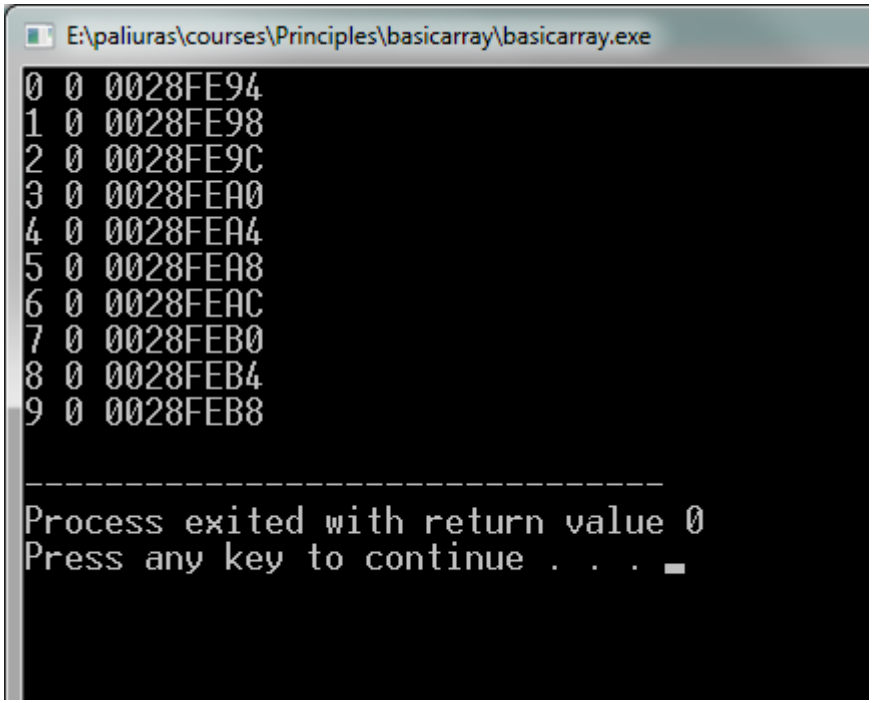
```
printf("\ttemp[%d]: %+6.2f\n", i, temp[i]);
```

cygwin

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]

```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

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];

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));

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

Συνάρτηση της βασικής βιβλιοθήκης scanf ( )

```
int number;
char ch;
scanf("%d", &number);
scanf("%c", &ch);
```

%d θα διαβάσει ακέραιο

%c θα διαβάσει χαρακτήρα

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

Παράδειγμα

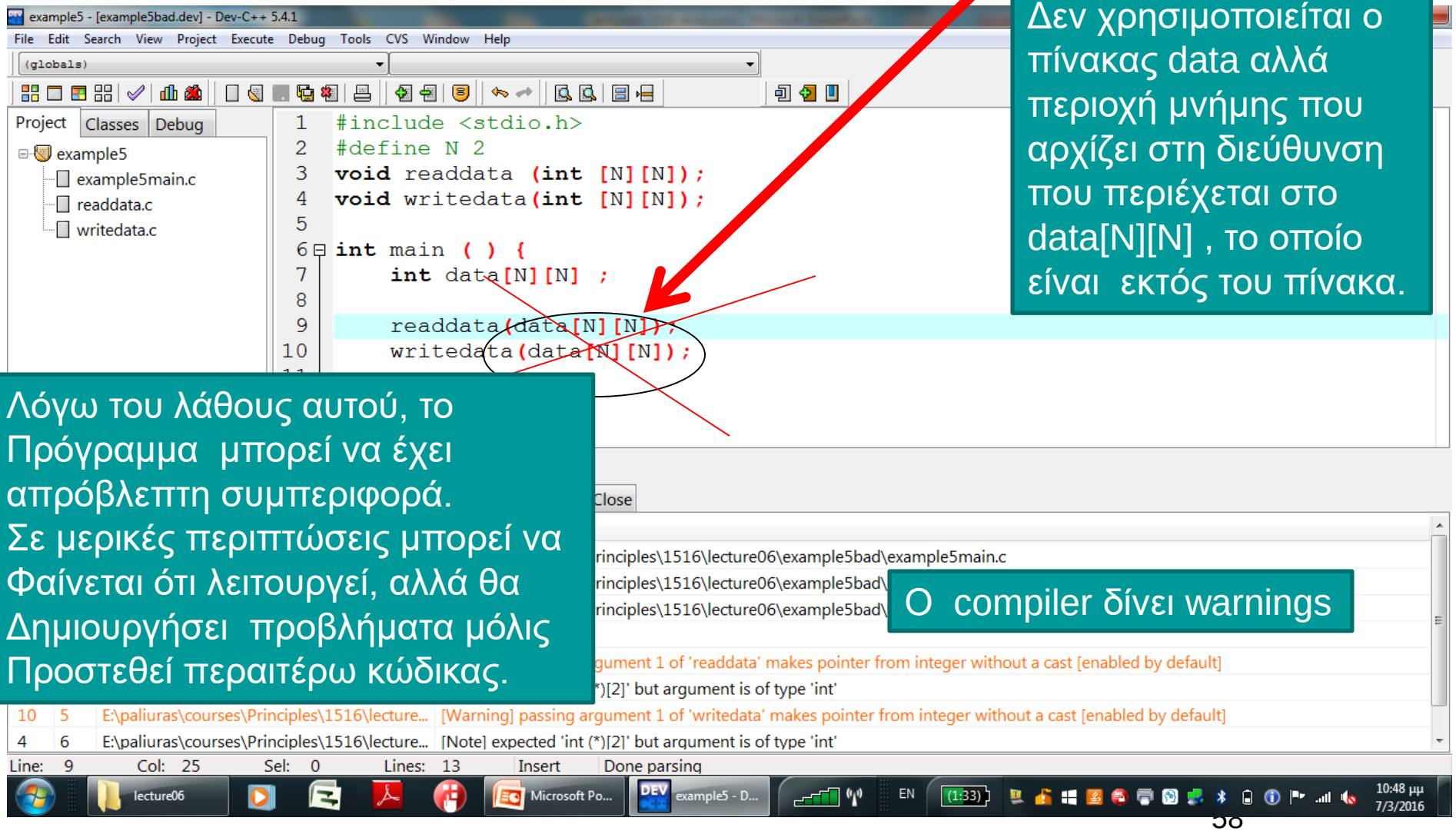
```
#define N 2
#include <stdio.h>
int main ( ) {
   int data[N][N] ;
   int i, j ;

   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!!!



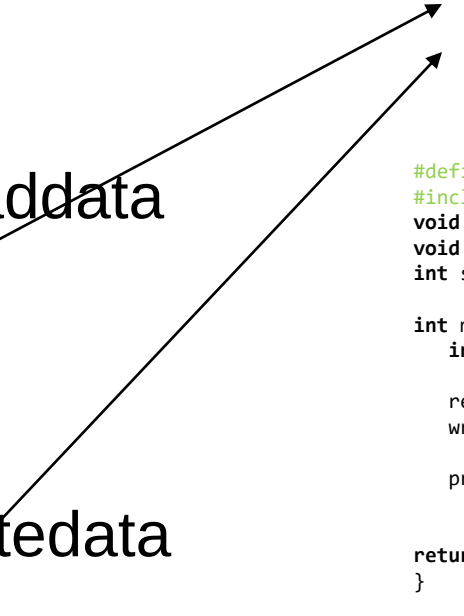
```c
#define N 2
#include <stdio.h>
void readdata(int a[N][N]);
void writedata(int b[N][N]);
int sumdata(int x[N][N]);

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;
}
```

- Συνάρτηση main
  - Πίνακας data
  - Καλεί την συνάρτηση readdata
    - Όρισμα a
  - Καλεί την συνάρτηση writedata
    - Όρισμα b

Υπάρχει ένας πίνακας!

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;
}
```

Παράδειγμα

- Ζητήθηκε από 40 φοιτητές να βαθμολογήσουν το φαγητό στο κυλικείο από 1 (απαίσιο) έως και 10 (αστέρι michelin).
- Να συνοψίσουμε τα αποτελέσματα.

- Από Deitel & Deitel.

- Λεκτική περιγραφή – Προστακτικός προγραμματισμός: (οργάνωση προγράμματος – συναρτήσεις)

  `Διάβασε τις απαντήσεις`

  `Υπολόγισε το ιστόγραμμα` ⇒ `generatehist()`

  `Παρουσίασε το αποτέλεσμα` ⇒ `printhist()`

- Αναπαράσταση δεδομένων
  - Πίνακας για απαντήσεις ⇒ `responses[ ]`
  - Πίνακας για ιστόγραμμα ⇒ `frequency[ ]`

Πρότυπα συναρτήσεων

Υλοποίηση συναρτήσεων

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

#define RESPONSE_SIZE 40
#define FREQUENCY_SIZE 11

void generatehist(int [], int []);
void printhist(int []);
void printstars(int );

int main()
{
   int frequency[ FREQUENCY_SIZE ] = { 0 };
   int responses[ RESPONSE_SIZE ] =
      { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
        1, 6, 3, 8, 6, 10, 3, 8, 2, 7,
        6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
        5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };

   generatehist(frequency, responses);

   printhist(frequency);

   return 0;
}
```

```c
void generatehist(int frequency[], int responses[]){
 int answer;

 for ( answer = 0; answer <= RESPONSE_SIZE - 1; answer++ )
            ++frequency[ responses [ answer ] ];
}

void printhist(int frequency[]) {
 int rating;

 printf( "%s%17s Bar\n", "Rating", "Frequency" );
 for ( rating = 1; rating <= FREQUENCY_SIZE - 1; rating++ ) {
    printf( "%6d%17d ", rating, frequency[ rating ] );
    printstars(frequency[rating]);
    printf("\n");
 }
}

void printstars(int len){
 int i;

 for (i=0;i<len;i++)
   putchar('*');
}
```

Η λεκτική περιγραφή αντιστοιχίζεται σε κλήση συναρτήσεων.

```c
#include <stdio.h>
#define RESPONSE_SIZE 40
#define FREQUENCY_SIZE 11

int main()
{
   int answer, rating, frequency[ FREQUENCY_SIZE ] = { 0 };
   int responses[ RESPONSE_SIZE ] =
      { 1, 2, 6, 4, 8, 5, 9, 7, 8, 10,
        1, 6, 3, 8, 6, 10, 3, 8, 2, 7,
        6, 5, 7, 6, 8, 6, 7, 5, 6, 6,
        5, 6, 7, 5, 6, 4, 8, 6, 8, 10 };

   for ( answer = 0; answer <= RESPONSE_SIZE - 1; answer++ )
      ++frequency[ responses [ answer ] ];

   printf( "%s%17s\n", "Rating", "Frequency" );

   for ( rating = 1; rating <= FREQUENCY_SIZE - 1; rating++ )
      printf( "%6d%17d\n", rating, frequency[ rating ] );

   return 0;
}
```