Προγραμματισμός Ι (ΗΥ120)

Διάλεξη 2:
Μνήμη – Εισαγωγή στη C

Χρήστος Δ. Αντωνόπουλος
25/9/2018

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μνήμη

- Μια διάταξη από «δωμάτια» κάθε ένα από τα οποία μπορεί να χρησιμοποιηθεί για την αποθήκευση 1 bit (0 / 1).
- 8 τέτοια δωμάτια οργανώνονται σε μια μονάδα (Byte)
  - Ένα «σπίτι» με διεύθυνση κλπ!
- Τη μνήμη ο προγραμματιστής τη «βλέπει» σα μια ακολουθία από bytes.
  - Η διεύθυνση κάθε byte είναι η σειρά του στην ακολουθία

Πόσο Μεγάλη Μνήμη;

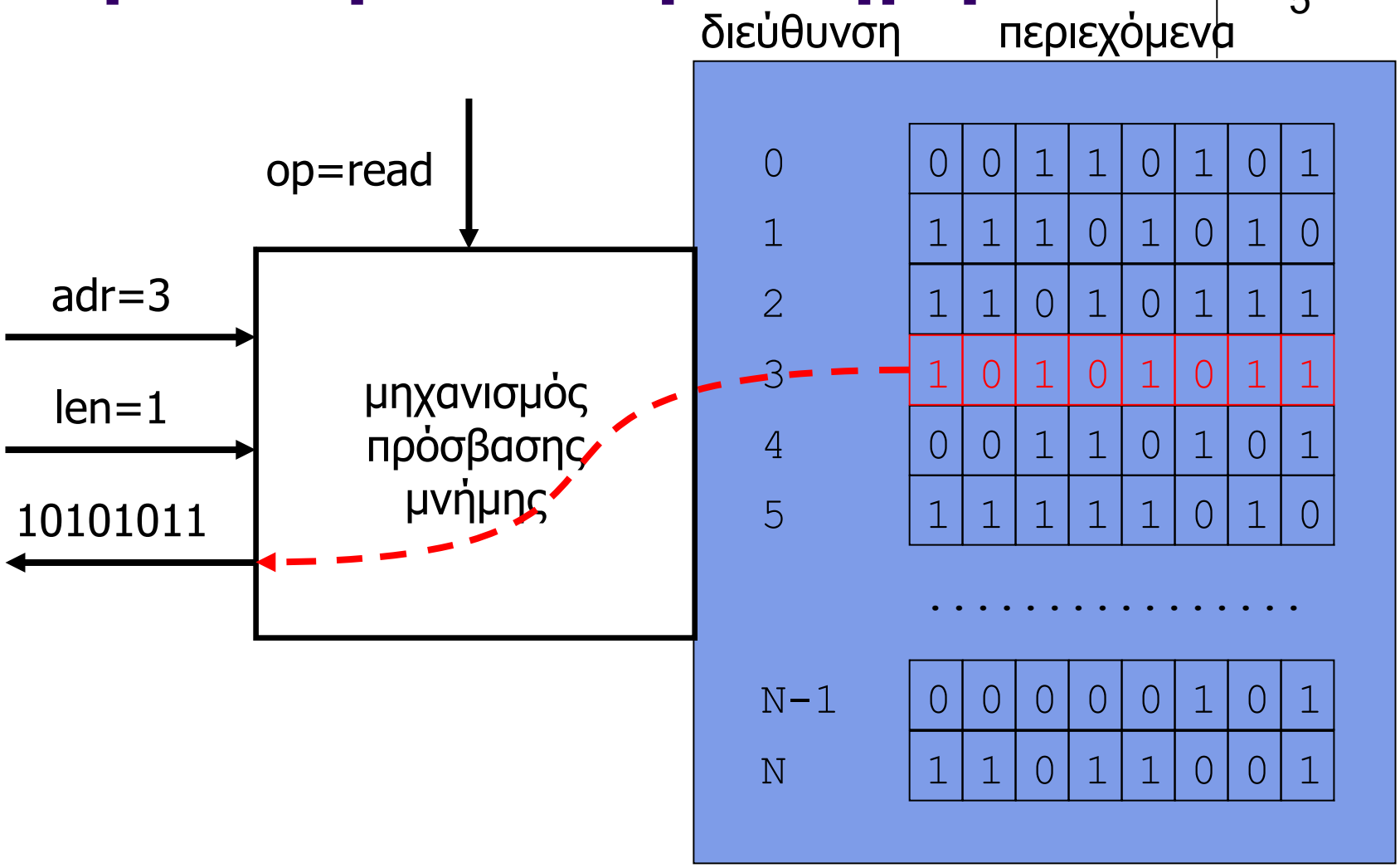
- Τυπικός Η/Υ:
  - Κύρια μνήμη (RAM) μερικών Gigabyte
  - Σκληρός δίσκο λίγων Terabyte.
    - 1 byte (B) = 8 bits
    - 1 KB = 1024 B = 2^{10} B
    - 1 MB = 1024 KB = 2^{20} B
    - 1 GB = 1024 MB = 2^{30} B
    - 1 TB = 1024 GB = 2^{40} B
    - …

Πρόσβαση στη Μνήμη

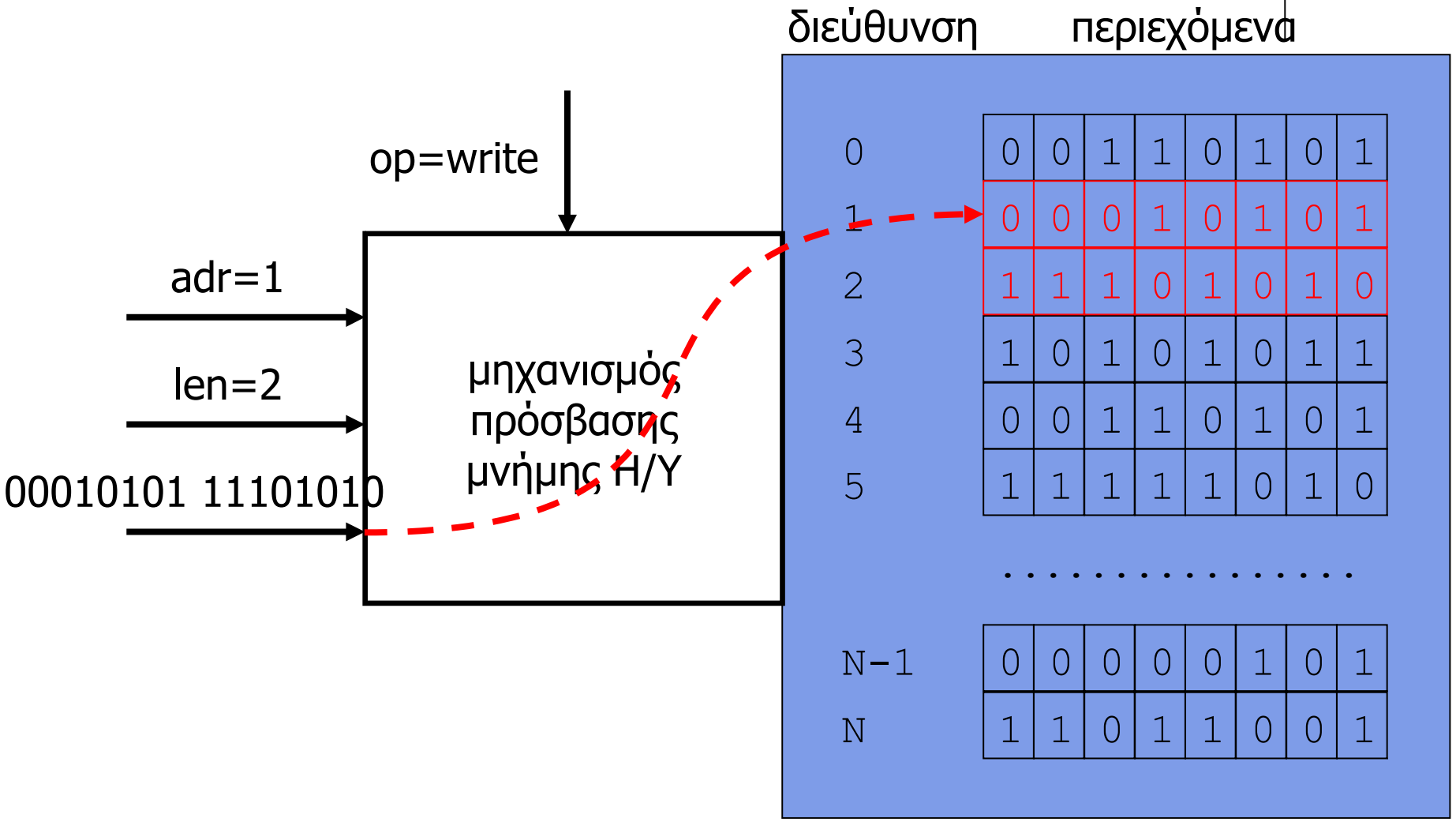
- Τι πρέπει να καθορίσω;
  - Τη διεύθυνση
  - Τι θέλω να κάνω (ανάγνωση / εγγραφή)
  - Σε πόσα bytes
    - Πόσο μεγάλη γειτονιά…

- «Ανάγνωση»: **επιστρέφει** τις τιμές των bytes που έχουν αποθηκευτεί στις αντίστοιχες θέσεις.
- «Εγγραφή»: **προσδιορίζει** τις τιμές των bytes που θα αποθηκευτούν στις αντίστοιχες θέσεις.

Ανάγνωση από τη Μνήμη



Εγγραφή στη Μνήμη



Κωδικοποίηση Δεδομένων

- Τα πάντα είναι 0 και 1!
- Κωδικοποίηση: η μετατροπή δεδομένων από ένα σύστημα σε ένα άλλο (π.χ. Δεκαδικό -> δυαδικό, χαρακτήρες -> δυαδικό).
- Πόσα bits για κάθε είδος δεδομένων;
  - Περισσότερα bits/bytes:
    - Όσο μεγαλύτερο το πεδίο τιμών
    - Όσο μεγαλύτερη η επιθυμητή ακρίβεια
  - Με N bits κωδικοποιούμε 2^N διαφορετικές τιμές:
    - Π.χ. καλύπτουμε το πεδίο φυσικών $[0...2^N-1]$
    - Ή το πεδίο ακεραίων $[-2^{N-1}...2^{N-1}-1]$.

Δεδομένα Προγράμματος και Μνήμη

- Για κάθε δεδομένο πρέπει να καθοριστεί (από τον προγραμματιστή):
  - Η θέση μνήμης για την αποθήκευση του
  - Ο αριθμός των bytes που χρειάζονται για την αποθήκευση των τιμών που μπορεί να λάβει
  - Η κωδικοποίηση που χρησιμοποιείται για κάθε τιμή στο δυαδικό σύστημα
- Απάνθρωπο;
  - Χρειάζεται κατάλληλη υποστήριξη από τις γλώσσες προγραμματισμού ...

Τύποι Δεδομένων

- Κάθε γλώσσα ορίζει ένα σύνολο από βασικούς **τύπους δεδομένων**.
- Κάθε βασικός τύπος έχει:
  - Προκαθορισμένο μέγεθος (σε bytes)
  - Συγκεκριμένη δυαδική κωδικοποίηση και αντίστοιχο πεδίο τιμών.
- Ο προγραμματιστής ορίζει κάθε δεδομένο του προγράμματος ως αντικείμενο ενός τύπου
  - Καθορίζει έμμεσα το μέγεθος και την κωδικοποίηση του (την **σημασία** των bits).
  - Η μνήμη αποθηκεύει τις τιμές των bits/bytes **χωρίς** να γνωρίζει το πως αυτές ερμηνεύονται από το πρόγραμμα.

Ονομασία Θέσεων Δεδομένων

- Σε κάθε αντικείμενο δεδομένων δίνεται και ένα **μνημονικό** όνομα.
  - Το όνομα μπορεί να χρησιμοποιείται, αντί της διεύθυνσης, στις πράξεις ανάγνωσης και αλλαγής των τιμών του.
  - Η πρόσβαση στην μνήμη εξακολουθεί να γίνεται με βάση την θέση μνήμης και τον αριθμό bytes προς ανάγνωση/αποθήκευση
    - Η αντιστοίχιση γίνεται (αυτόματα) από την εκάστοτε γλώσσα προγραμματισμού.

Μεταβλητές Προγράμματος

- **Μεταβλητή:** αντικείμενο δεδομένων του προγράμματος με
  - Όνομα και
  - Τύπο
- «Διαβάζουμε» μεταβλητή: Διαβάζουμε την τιμή που έχει αποθηκευτεί στην αντίστοιχη θέση της μνήμης.
- «Αναθέτουμε» τιμή σε μεταβλητή: Γράφουμε αυτή την τιμή στην αντίστοιχη θέση της μνήμης.
- «Δέσμευση» μεταβλητής: Δέσμευση αντίστοιχου χώρου στην μνήμη
- «Ζωή» μεταβλητής: Διάστημα κατά το οποίο αυτός ο χώρος παραμένει δεσμευμένος.

Παράδειγμα


```
int var;
```

θεση/διεύθυνση: 1
τύπος: int

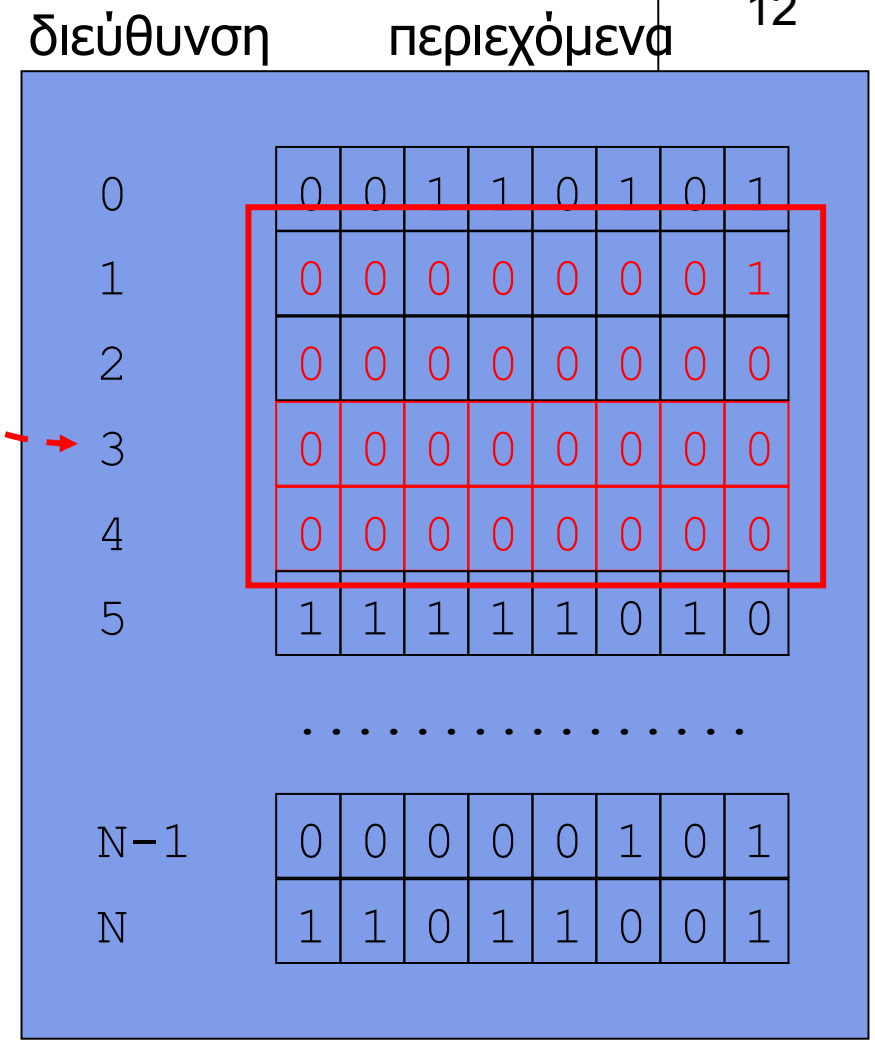
τύπος int

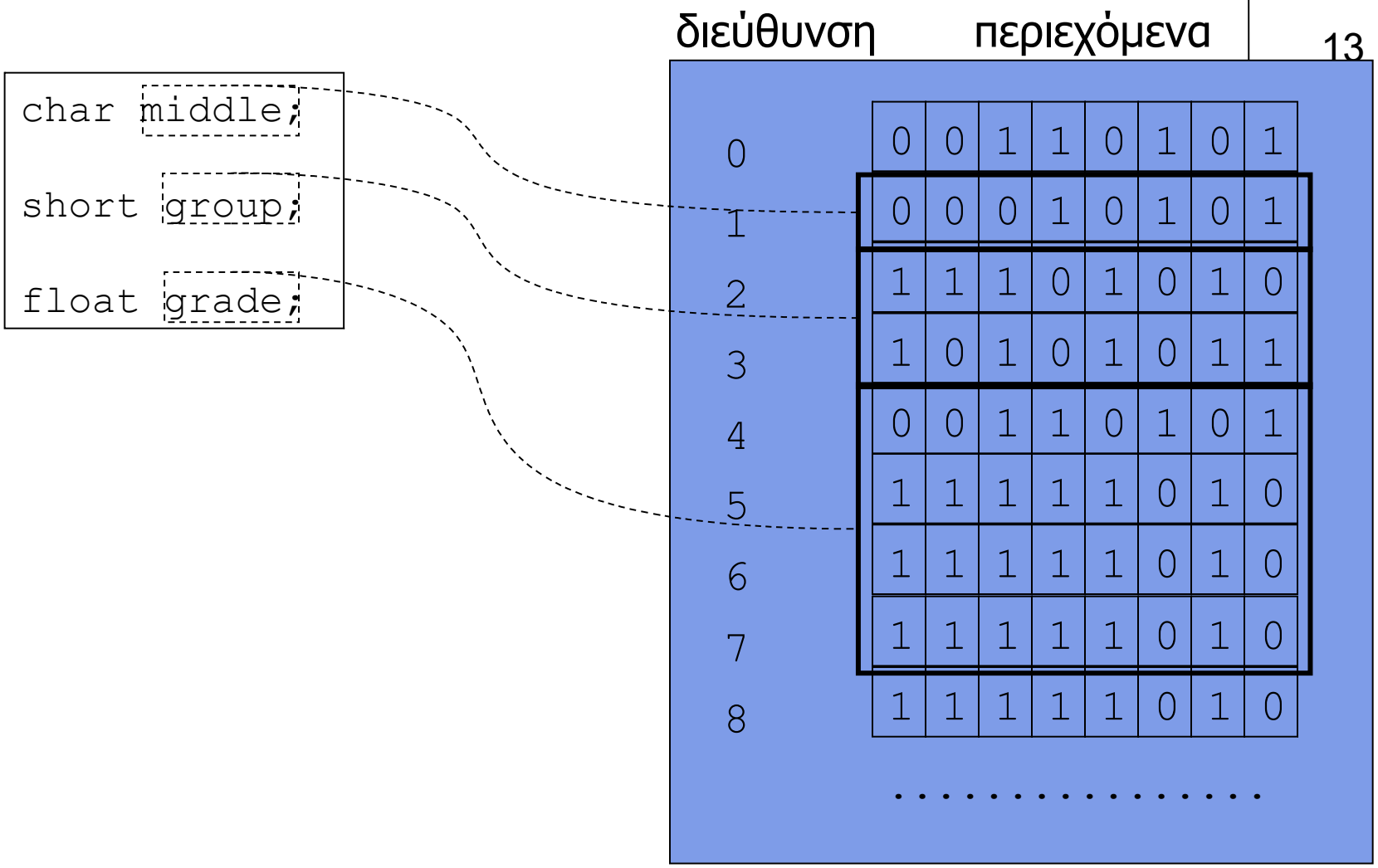
μέγεθος: 4 bytes
ερμηνεία: ακέραιος
κωδικοποίηση: 2's complement

τι τιμή έχει η μεταβλητή var;

διάβασε από τη διεύθυνση της var όσα bytes ορίζει ο τύπος της T (π.χ. int), και ερμήνευσέ τα σύμφωνα με τη σύμβαση κωδικοποίησης του T

αποτέλεσμα




διεύθυνση
περιεχόμενα
char middle;
short group;
float grade;
0
1
2
3
4
5
6
7
8
0 0 1 1 0 1 0 1
0 0 0 1 0 1 0 1
1 1 1 0 1 0 1 0
1 0 1 0 1 0 1 1
0 0 1 1 0 1 0 1
1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0
1 1 1 1 1 0 1 0
. . . . . .

Τι είναι η C;

- Είναι μια γλώσσα «προστακτικού» (imperative) προγραμματισμού με σχετικά λίγες εντολές.
- Είναι ιδιαίτερα διαδεδομένη, σε επίπεδο υλοποίησης λειτουργικών συστημάτων αλλά και εφαρμογών.
- Η ελάχιστη μονάδα υποπρογράμματος (δόμησης κώδικα) είναι η συνάρτηση (function).
- Ένα πρόγραμμα C μεταφράζεται και μετά εκτελείται.
  - Υποστηρίζεται ξεχωριστή μετάφραση διαφορετικών υποπρογραμμάτων με στατική ή δυναμική σύνδεση.
- Οι γλώσσες C++ και Java είναι «απόγονοι» της C.

Μορφοποίηση Κώδικα στη C

- Σχεδόν κάθε εντολή τερματίζεται με '`;`'.
  - Μπορούμε να γράφουμε πολλές διαδοχικές εντολές στην ίδια γραμμή του κειμένου.
- Χρησιμοποιούμε τον όρο «σώμα» ή «μπλοκ» για να αναφερθούμε σε μια ή περισσότερες εντολές που δίνονται ανάμεσα σε '`{`' και '`}`' και συμπεριφέρονται σαν «ομάδα».
- Σχόλια (κείμενο που δεν λαμβάνει υπ' όψη του ο μεταφραστής) δίνονται ανάμεσα σε '`/*`' και '`*/`'.
- Κενοί χαρακτήρες και γραμμές που βρίσκονται ανάμεσα σε εντολές δεν λαμβάνονται υπ' όψη από τον μεταφραστή
  - Χρησιμεύουν, όπως και τα σχόλια, για την αναγνωσιμότητα του κώδικα.

Παίζοντας με τη Μορφοποίηση ;-)

```
/*                                  ,*/
          #include                                  <time.h>
           #include/*                          _  ,o*/  <stdlib.h>
           #define  c(C)/*     -       . */return       ( C); /*    2004*/
            #include   <stdio.h>/*.   Moekan            "'   `\b-'     */
              typedef/*  */char   p;p* u                      ,w       [9
                ][128] ,*v;typedef  int _;_   R,i,N,I,A              ,m,o,e
             [9],  a[256],k    [9], n[              256];FILE*f       ;_ x   (_ K,_ r
      ,_ q){;   for(;                                        r<     q    ; K       =((
   0xffffff)   &(K>>8))^                                       n[255     &         ( K
  ^u[0        +                                      r  ++      ]     )]);c          (K
 )}        _ E                              (p*r,    p*q  ){     c(          f         =
         fopen                         (r  ,q))}_   B(_ q){c(    fseek        (f,      0
       ,q))}_ D(){c(  fclose(f ))}_  C(  p    *q){c(  0-    puts(q    )   )}_/*   /
     */main(_ t,p**z){if(t<4)c(   C("<in"       "file>"    "\40<l"   "a"  "yout> "
    /*b9213272*/"<outfile>"    ) )u=0;i=I=(E(z[1],"rb")) ?B(2)?0 :   (((o   =ftell
   (f))>=8)?(u     =(p*)malloc(o))?B(0)?0:!fread(u,o,1,f):0:0)?0:  D():0      ;if(
  !u)c(C("      bad\40input  "));if(E(z[2],"rb" )){for(N=-1;256> i;n[i++] =-1   )a[
  i]=0;       for(i=I=0;   i<o&&(R  =fgetc(   f))>-1;i++)++a[R] ?(R==N)?( ++I>7)?(n[
 N]+1       )?0:(n [N   ]=i-7):0:   (N=R)    |(I=1):0;A =-1;N=o+1;for(i=33;i<127;i++
 )(        n[i   ]+  1&&N>a[i])?    N= a     [A=i]     :0;B(i=I=0);if(A+1)for(N=n[A];
I<       8&&   (R  =fgetc(f ))>    -1&& i   <o         ;i++)(i<N||i>N+7)?(R==A)?((*w[I
]             =u [i])?1:(*w[I]=   46))?(a             [I++]=i):0:0:0;D();}if(I<1)c(C(
              "  bad\40la" "yout  "))for(i            =0;256>(R=  i);n[i++]=R)for(A=8;
            A  >0;A --)   R  = ( (R&1)==0)          ?(unsigned int)R>>(01):((unsigned
            /*kero  Q'        ,KSS  */)R>>         1)^        0xedb88320;m=a[I-1];a[I
            ]=(m           <N)?(m=   N+8):        ++        m;for(i=00;i<I;e[i++]=0){
           v=w          [i]+1;for(R                     =33;127  >R;R++)if(R-47&&R-92
           &&       R-(_)* w[i])*(                  v++)=    (p)R;*v=0;}for(sprintf
                  /*'_  G*/  (*w+1,              "%0"     "8x",x(R=time(i=0),m,o)^~
               0)   ;i<       8;++           i)u      [N+ i]=*(*w+i+1);for(*k=x(~
                 0,i=0     ,*a);i>-        1;      ){for (A=i;A<I;A++){u[+a [ A]
               ]=w[A       ][e[A]]  ;     k     [A+1]=x (k[A],a[A],a[A+1]
             );}if     (R==k[I])      c(      (E(z[3  ],"wb+"))?fwrite(
            /* */   u,o,1,f)?D          ()|C("  \n    OK."):0    :C(
           "  \n  WriteError"            ))  for  (i  =+I-
           1  ;i >-1?!w[i][++               e[+ i]]:0;
             ) for( A=+i--;                A<I;e[A++]
             =0); (i <I-4                 )?putchar
             ((_   )  46)                  | fflush
            /*'      ,*/                  ( stdout
            ):       0&                    0;}c(C
           ("       \n                   fail")
           )      /*                     dP' /
                 dP                     pd  '
                '                       zc
                                       */
                                      }
```

Από το Obfuscated C Contest
http://www.ioccc.org/

Η συνάρτηση στη C

- Συνάρτηση: Η κύρια μονάδα υποπρογράμματος
- Μπορεί να βασίζεται σε άλλες συναρτήσεις, που με τη σειρά τους μπορεί να βασίζονται σε άλλες συναρτήσεις κλπ.
- Η συνάρτηση **`main`** (κυρίως συνάρτηση) καλείται από το περιβάλλον εκτέλεσης (λειτουργικό) για να αρχίσει η εκτέλεση του προγράμματος
  - Ανεξάρτητα από το αν το πρόγραμμα έχει και άλλες συναρτήσεις.
  - Αρχικά θα εστιάσουμε σε προγράμματα που αποτελούνται μόνο από την `main`.

Η συνάρτηση main()


```
/* Αρχείο hello.c. Εμφανίζει στην
   οθόνη το μήνυμα hello world
*/

#include <stdio.h>

int main(int argc, char* argv[]) {

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

Σχόλια

Συμπερίληψη δηλώσεων εξωτερικών συναρτήσεων

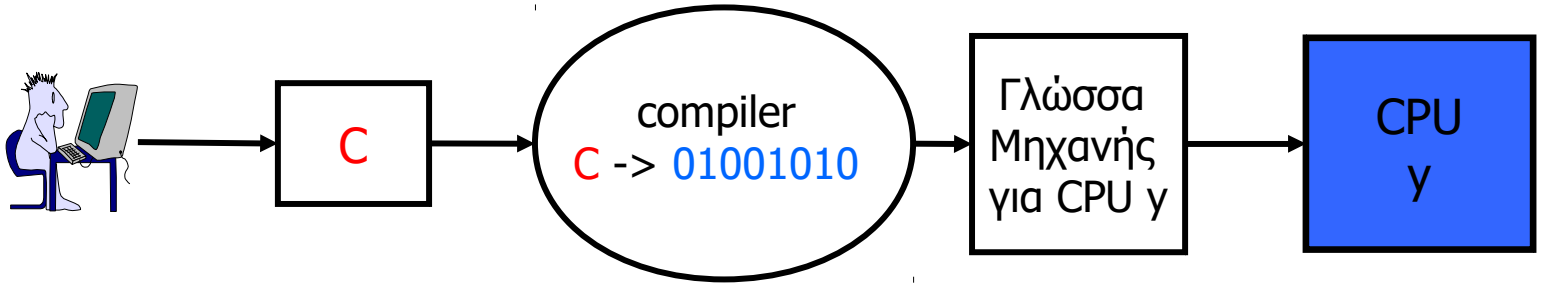
Ορίσματα συνάρτησης

Επικεφαλίδα συνάρτησης

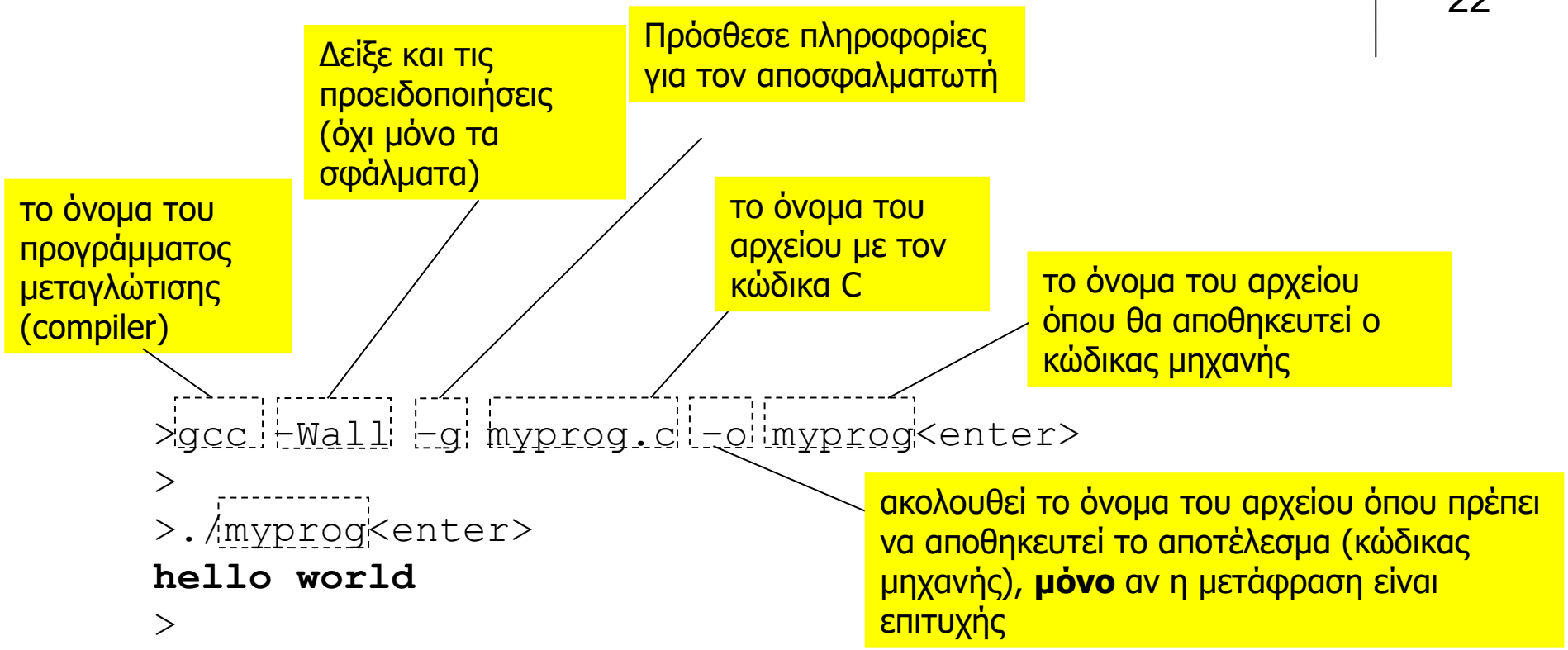
Σώμα (εντολές)

Μετάφραση / μεταγλώτιση (compilation)

- Πώς «καταλαβαίνει» ο υπολογιστής τη γλώσσα υψηλού επιπέδου;
- Πρέπει οι εντολές να **μετατραπούν** σε εντολές γλώσσας μηχανής.
  - Αυτό ονομάζεται **μεταγλώττιση**
  - Γίνεται από ειδικά προγράμματα, τους μεταγλωττιστές (compilers).
  - Συνήθως μια εντολή γλώσσας υψηλού επιπέδου αντιστοιχεί σε μια σειρά εντολών γλώσσας μηχανής
  - Η μεταγλώττιση γίνεται (συνήθως) ως ξεχωριστή διαδικασία, πολύ πριν αρχίσει η εκτέλεση του κώδικα.
    - Υπάρχουν περιπτώσεις όπου η μετάφραση γίνεται ακριβώς πριν την εκτέλεση (just in time compilation).


C
compiler
C -> 01001010
Γλώσσα
Μηχανής
για CPU y
CPU
y

Μεταγλώττιση / Εκτέλεση στο Linux



Ένα Ακόμα Παράδειγμα


```
#include <stdio.h>

int main(int argc, char* argv[]) {

  putchar('h');
  putchar('e');
  putchar('l');
  putchar('l');
  putchar('o');
  putchar(' ');
  putchar('w');
  putchar('o');
  putchar('r');
  putchar('l');
  putchar('d');
  putchar('\n');

  return(0);
}
```

```
>./myprog<enter>
hello world
>
```


```
#include <stdio.h>

int main(int argc, char* argv[]) {

  putchar('5');
  putchar(' ');
  putchar('+');
  putchar(' ');
  putchar('2');
  putchar(' ');
  putchar('=');
  putchar('=');
  putchar(' ');
  putchar('9');
  putchar('\n');

  return(0);
}
```

```
>./myprog<enter>
5 + 2 == 9
>
```

Αποσφαλμάτωση

- Συντακτικό επίπεδο (εύκολο)
  - Ο κώδικας δεν αντιστοιχεί σε επιτρεπτή πρόταση σύμφωνα με τους κανόνες σύνταξης της γλώσσας
- Σημασιολογικό επίπεδο (δύσκολο)
  - Ο κώδικας είναι συντακτικά σωστός αλλά **εμείς** δεν χρησιμοποιούμε σωστά κάποια εντολή
    - Άλλο θέλουμε να κάνουμε και άλλο ζητάμε από τον υπολογιστή να κάνει
    - Παράγεται λάθος αποτέλεσμα
- Λογικό επίπεδο (πιο δύσκολο)
  - Ο κώδικας είναι συντακτικά σωστός και όλες οι εντολές χρησιμοποιούνται σωστά
  - Υπάρχει λάθος σε επίπεδο αλγορίθμου (σκέψης)

Ορθότητα προγραμμάτων

- Αν στη μετάφραση εντοπιστεί συντακτικό λάθος, αυτή τερματίζεται (εκτυπώνεται μήνυμα λάθους), διαφορετικά παράγεται κώδικας μηχανής.
- Το πρόγραμμα μεταφράστηκε επιτυχώς: Είναι σίγουρα σωστό;
  - Κανείς **δεν** εγγυάται ότι η εκτέλεση του θα έχει και το επιθυμητό / αναμενόμενο αποτέλεσμα.
  - Η εκτέλεση ενός (μεταγλωττισμένου) προγράμματος γίνεται πάντα σωστά, δηλαδή όπως ορίζεται από την σημασιολογία της γλώσσας προγραμματισμού.
    - Δε φταίει το «καταραμένο το μηχάνημα»
    - Μπορεί να υπάρχουν **σημασιολογικά** ή/και **λογικά** λάθη.
      - Το πρόγραμμα δεν κάνει αυτό που θέλουμε

Υπολογισμός x=1+2+...+n

Στην καθομιλουμένη γλώσσα:

Θέσε το x ίσο με 0.
Πρόσθεσε στο x την τιμή 1.
Πρόσθεσε στο x την τιμή 2.
Πρόσθεσε στο x την τιμή 3.
...
Πρόσθεσε στο x την τιμή n-1.
Πρόσθεσε στο x την τιμή n.

Στη γλώσσα προγραμματισμού C:

```
x = 0;
x = x+1;
x = x+2
x = x+3;
...
x = x+n-1;
x = x+n;
```

εντοπίζεται από
τον μεταφραστή

δεν έχουμε βάλει ;
εκεί που θα έπρεπε

Υπολογισμός x=1+2+...+n

Στην καθομιλουμένη γλώσσα:

Θέσε το x ίσο με 0.
Πρόσθεσε στο x την τιμή 1.
Πρόσθεσε στο x την τιμή 2.
Πρόσθεσε στο x την τιμή 3.
...
Πρόσθεσε στο x την τιμή n-1.
Πρόσθεσε στο x την τιμή n.

Στη γλώσσα προγραμματισμού C:

```
x = 0;
x = x+1;
x == x+2;
x = x+3;
...
x = x+n-1;
x = x+n;
```

σημασιολογικό λάθος

δεν εντοπίζεται από τον μεταφραστή

δεν έχουμε καταλάβει τι κάνει ο τελεστής ==

Υπολογισμός x=1+2+...+n

Στην καθομιλουμένη γλώσσα:

Θέσε το x ίσο με 0.
Πρόσθεσε στο x την τιμή 1.
Πρόσθεσε στο x την τιμή 3.
Πρόσθεσε στο x την τιμή 3.
...
Πρόσθεσε στο x την τιμή n-1.
Πρόσθεσε στο x την τιμή n.

Στη γλώσσα προγραμματισμού C:

```
x = 0;
x = x+1;
x = x+3;
x = x+3;
...
x = x+n-1;
x = x+n;
```

δεν εντοπίζεται από τον μεταφραστή

έχουμε υλοποιήσει λάθος αλγόριθμο