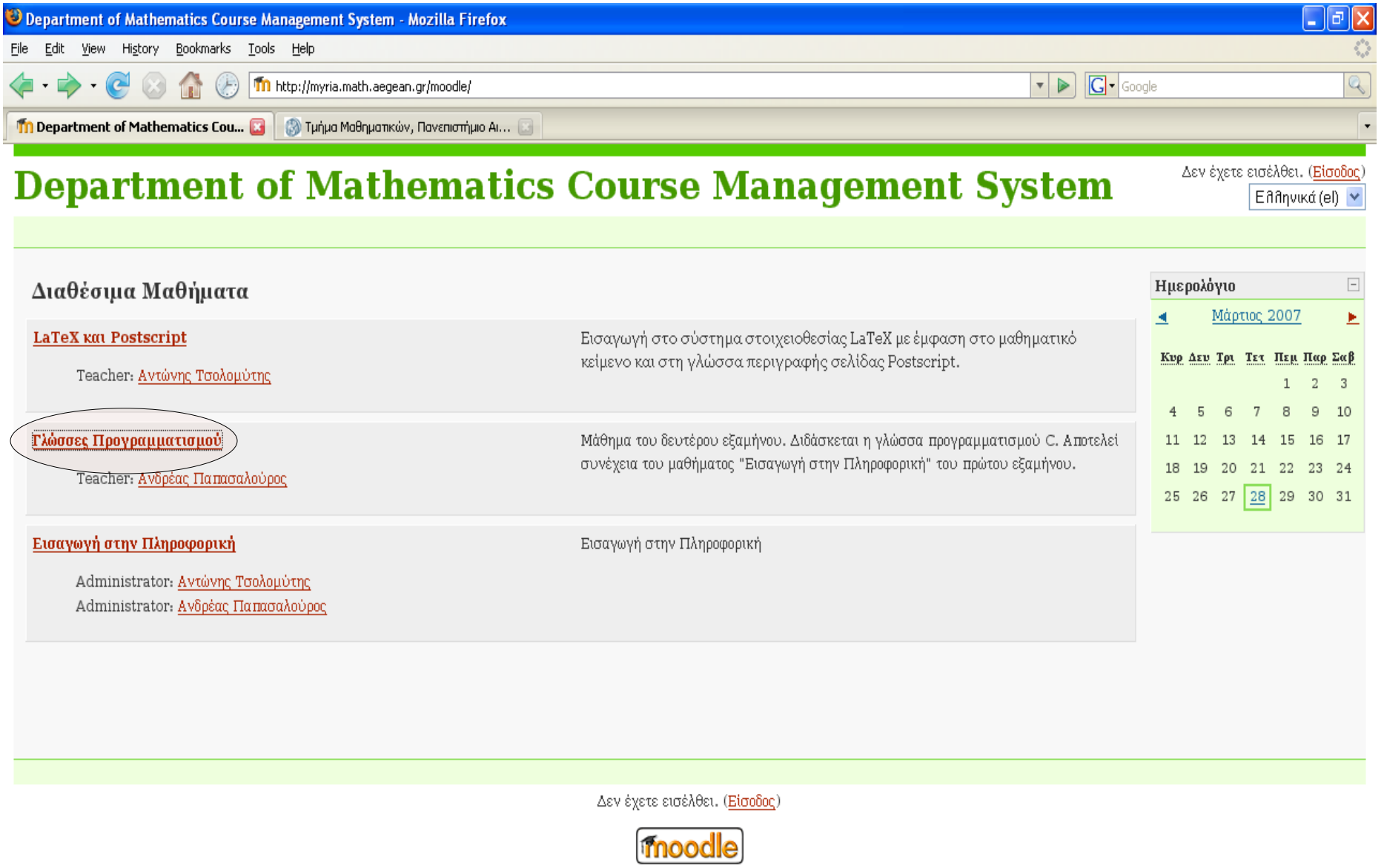
Γλώσσες Προγραμματισμού

- Διδάσκων: Ανδρέας Παπασαλούρος
- Ιστοσελίδα:
  http://www.samos.aegean.gr/math/andpapas/courses/pl/default.htm
- Πλατφόρμα ηλεκτρονικής μάθησης:
  http://myria.math.aegean.gr/moodle/

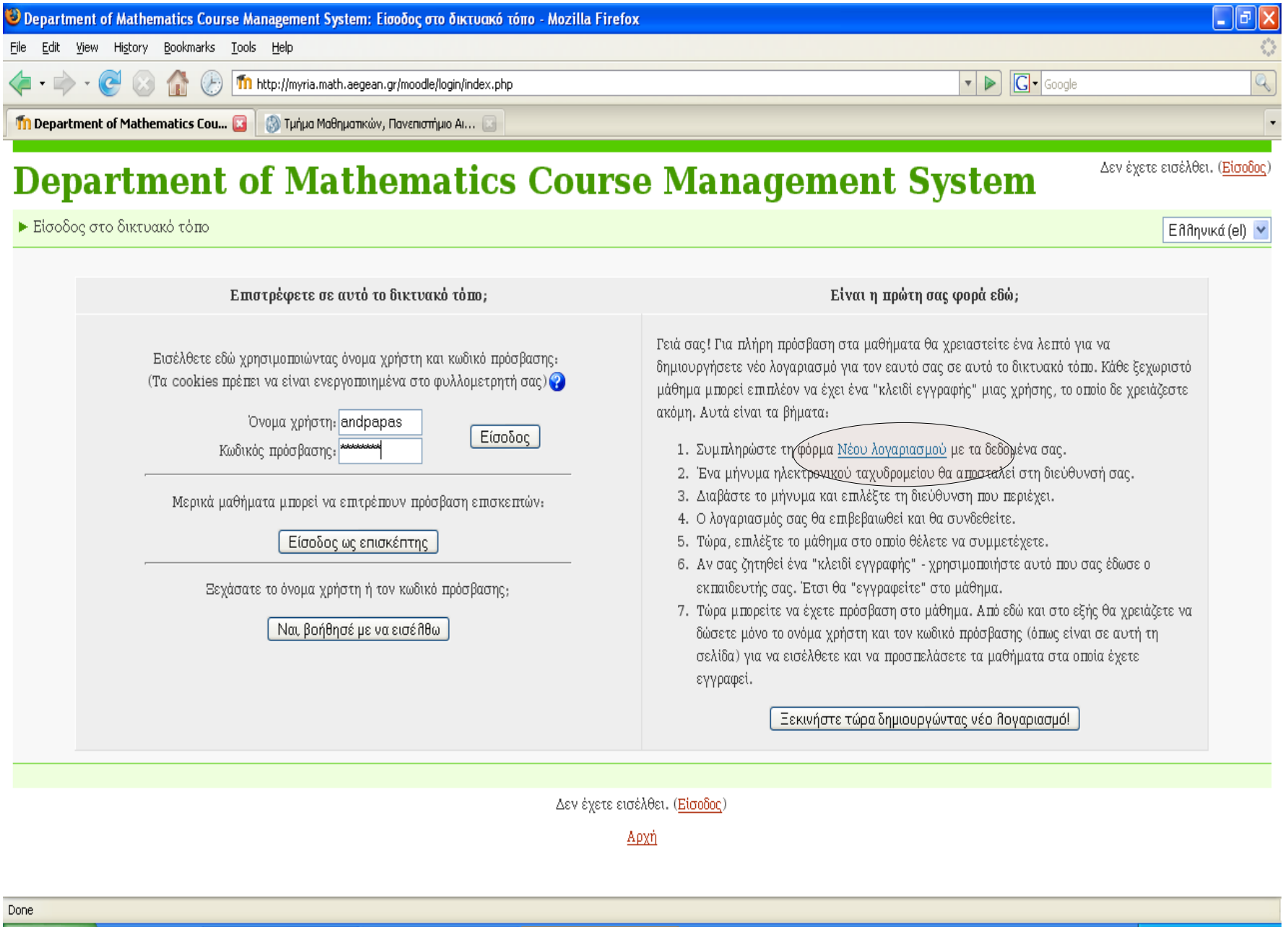
Πλατφόρμα ηλεκτρονικής μάθησης

- Υλικό του μαθήματος
- Υποβολή εργασιών
- Επικοινωνία
- Συζητήσεις για το μάθημα

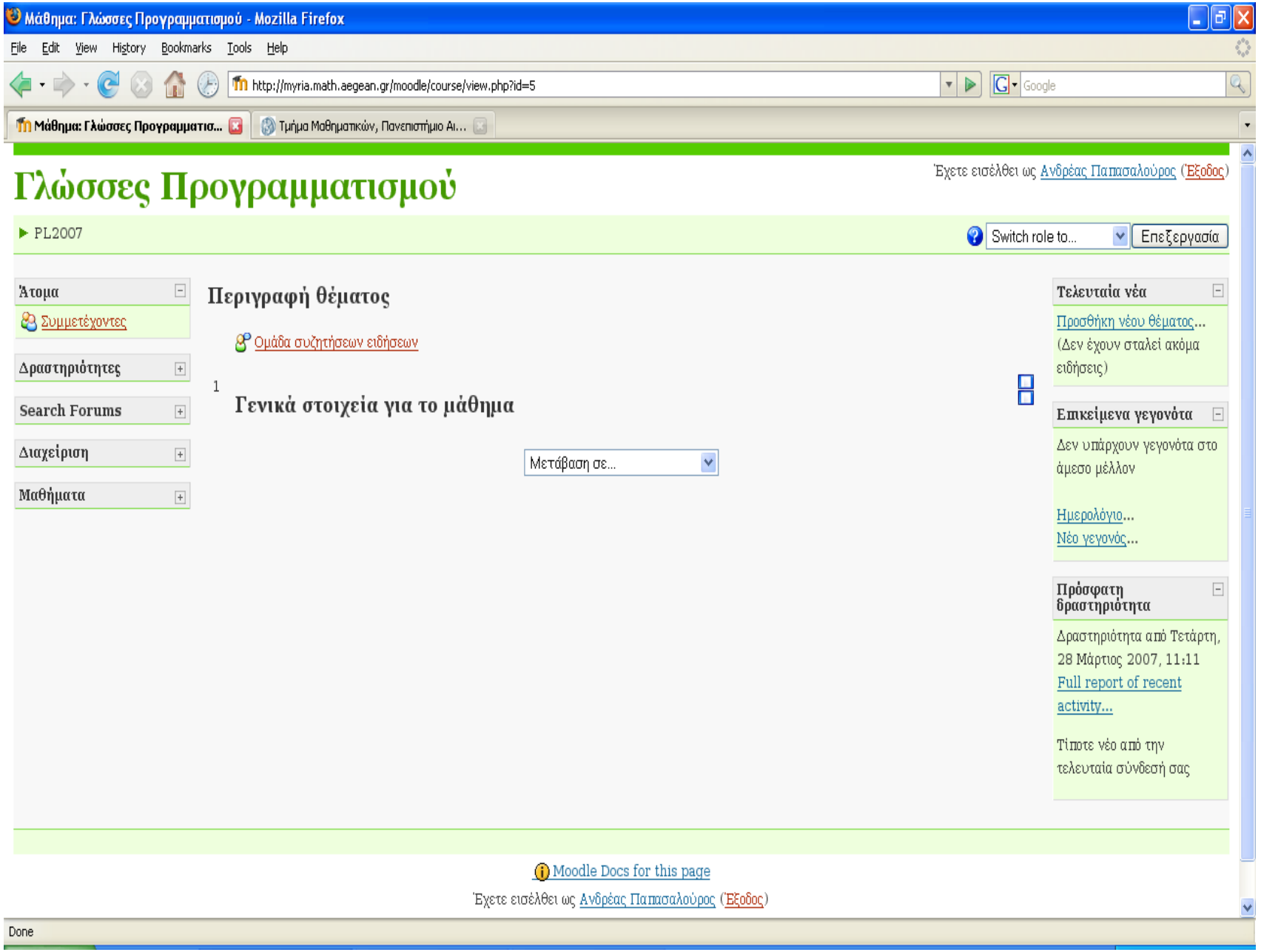
Η πλατφόρμα ηλεκτρονικής μάθησης moodle



Εγγραφή στην πλατφόρμα



Η κεντρική σελίδα του μαθήματος



Αναλυτικός σχεδιασμός προγραμμάτων

Αρθρωτή (δομημένη) ανάπτυξη προγραμμάτων

- Ενότητα 5.6 του βιβλίου του Roberts
- Ένα πρόγραμμα υλοποιεί τη λύση ενός (συνήθως σύνθετου) **προβλήματος**.
- Παράδειγμα: Ένα ταξίδι στο Λονδίνο
  - Έκδοση φθηνού αεροπορικού εισιτηρίου
    - Συμβουλή πράκτορα
    - Έκδοση του εισιτηρίου
  - Κλείσιμο ξενοδοχείου
  - Ταξίδι
  - Διαμονή
  - Επιστροφή

Αναλυτικός σχεδιασμός προγραμμάτων

- Μια μέθοδος για την επίλυση σύνθετων προβλημάτων είναι η ανάλυσή (analysis) ή αποσύνθεση (decomposition) τους σε επιμέρους (“μικρότερα”) υποπροβλήματα.
- Κάθε υποπρόβλημα αναλύεται περαιτέρω μέχρι να φτάσουμε σε απλά ή ήδη λυμένα προβλήματα τα οποία επιλύονται άμεσα.
- Η σχεδίαση ενός προγράμματος με την ανάλυση ενός προβλήματος από το πιο γενικό στο πιο ειδικό ονομάζεται αναλυτικός σχεδιασμός ή βηματική εκλέπτυνση.

Αναλυτικός σχεδιασμός

- Αναλύουμε ένα πρόβλημα σε υποπροβλήματα
  - Ένα υποπρόβλημα είναι δυνατόν να αναλύεται σε περαιτέρω υποπροβλήματα.
- Ορίζουμε μια κύρια συνάρτηση για την επίλυση του προβλήματος.
- Ορίζουμε μια συνάρτηση ή διαδικασία για κάθε υποπρόβλημα.
- Τα υποπροβλήματα στα οποία αναλύεται ένα πρόβλημα είναι συναρτήσεις που καλούνται από τη συνάρτηση που επιλύει το πρόβλημα

Παράδειγμα

```
main()
{
   findTickets();
   findHotel();
   travel();
   comeBack();
}
void findTickets() {
      consultAgent();
      bookTickets();
   }
...
```

Πίνακες στην C

Πίνακες στην C

- Δήλωση πίνακα
  - #define SIZE 20
  - int values[SIZE];
- Προσπέλαση στοιχείου πίνακα
- int v;
- v = values[10];

Πίνακες στην C (συνέχ.)

- Προσπέλαση όλων των στοιχείων ενός πίνακα

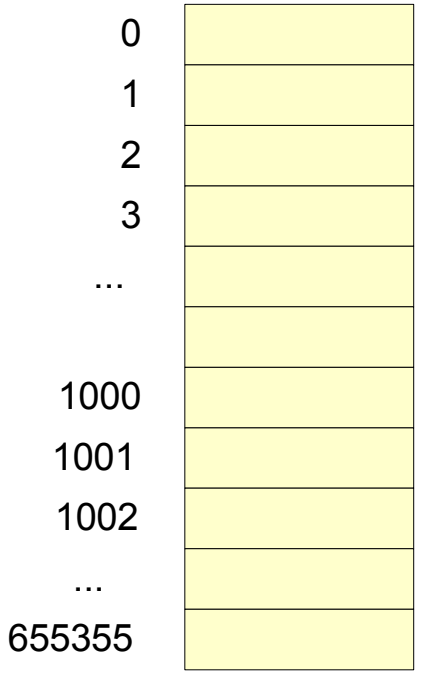
```
int i;
for (i = 0; i < SIZE; i++) {
   printf("size[%d] = %d", i,values[i]);
}
```

Εσωτερική αναπαράσταση δεδομένων

- Κάθε τιμή δεδομένων αποθηκεύεται εσωτερικά με έναν αριθμό bits.
- Η μικρότερη μονάδα αποθήκευσης είναι το byte
  - 1 byte = 8 bits
  - 1 word = 2 bytes
- Η χωρητικότητα της μνήμης ενός υπολογιστή μετριέται σε bytes
  - 1 KB = 2^{10} bytes = 1024 bytes
  - 1 MB = 2^{20} bytes = 1.048.576 bytes

Διευθύνσεις μνήμης

- Κάθε byte μνήμης προσδιορίζεται από μια **αριθμητική διεύθυνση**
- Κάθε byte μνήμης μπορεί να αποθηκεύσει ένα χαρακτήρα πληροφορίας
- Κατά τη δήλωση μιας μεταβλητής δεσμεύεται κατάλληλος χώρος στη μνήμη.

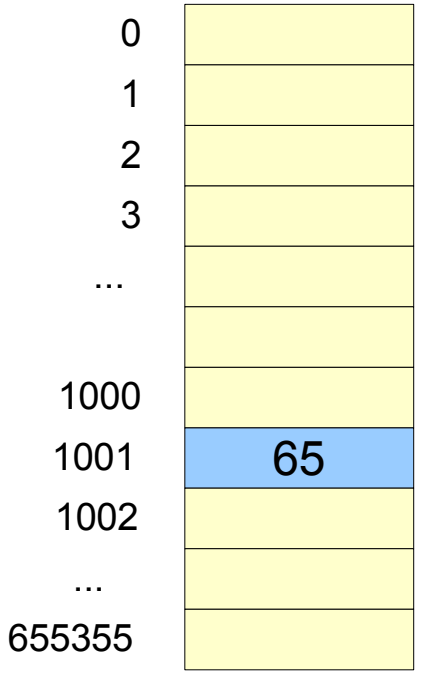


Διευθύνσεις μνήμης (2)

- Η δήλωση

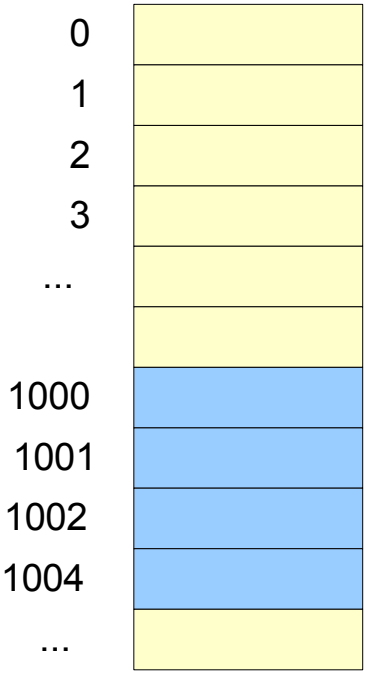
```
char ch = 'A';
```

δεσμεύει ένα byte σε μια συγκεκριμένη διεύθυνση της μνήμης και αποθηκεύει σε αυτή τον κωδικό **ASCII** του χαρακτήρα A (65).



Διευθύνσεις μνήμης (3)

- Άλλες τιμές δεδομένων διαφορετικών τύπων απαιτούν διαφορετικό αριθμό bytes για την αποθήκευσή τους.
- Για παράδειγμα, ένας ακαίρεος (τύπος int) δεσμεύει 2 ή ακόμη και byte της μνήμης, ανάλογα με την υλοποίηση.
- Ένας αριθμός τύπου double δεσμεύει 8 bytes μνήμης.



Ο τελεστής `sizeof`

- Ο τελεστής **`sizeof`** της C επιστρέφει τον αριθμό των bytes που δεσμεύει μια μεταβλητή, σταθερά ή τύπος δεδομένων.
- Παράδειγμα

```
#include <stdio.h>

main()
{
      char c = 'b';

      printf("%d\n", sizeof(c));       /* 1 */
      printf("%d\n", sizeof(12));      /* 2 */
      printf("%d\n", sizeof(double)); /* 8 */
      printf("%d\n", sizeof(12.5));    /* 8 */
}
```

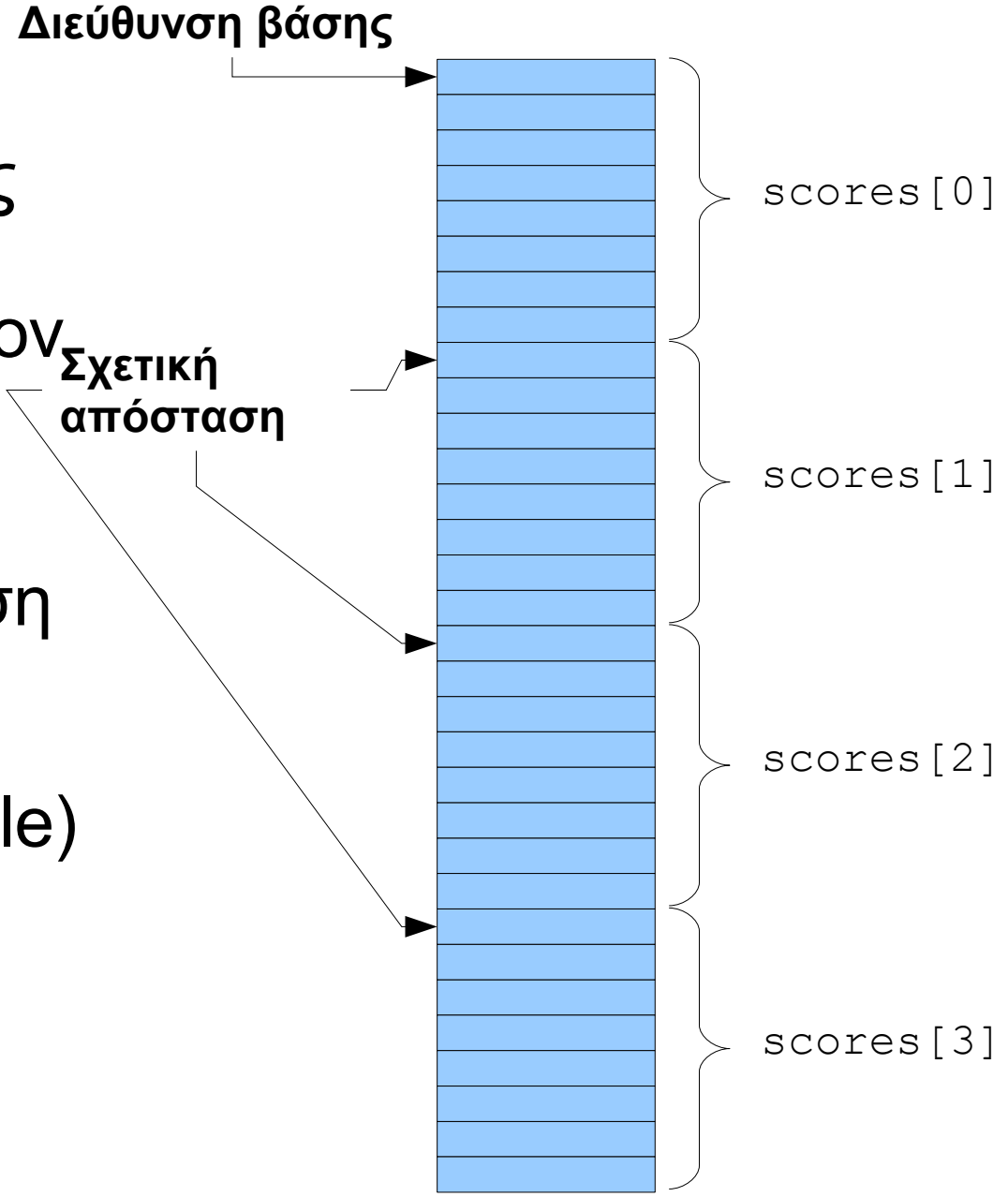
Κατανομή μνήμης σε πίνακες

- Κατά την δήλωση ενός πίνακα δεσμεύεται χώρος στη μνήμη ίσος με το μέγεθος του πίνακα επί τον χώρο που δεσμεύει ο τύπος του πίνακα.
- Για παράδειγμα, η δήλωση

```
double scores[4];
```

δεσμεύει 4 x sizeof(double) = 4 x 8 = 32 bytes

- Το όνομα του πίνακα *περιέχει* τη διεύθυνση βάσης του πίνακα.

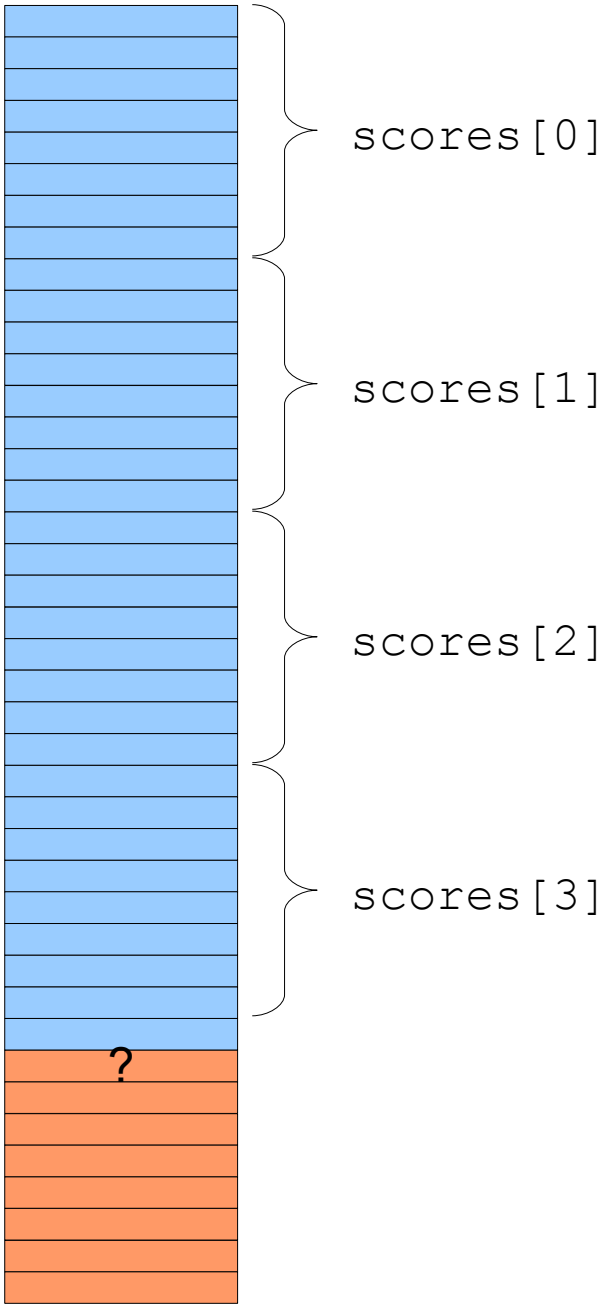


Αναφορά σε στοιχεία έξω από τα όρια πίνακα

- Η αναφορά σε στοιχεία έξω από τα όρια ενός πίνακα αποτελεί σφάλμα που δεν ανιχνεύεται από όλους τους μεταγλωττιστές.
- Για παράδειγμα η αναφορά

  scores[4]

- βρίσκεται έξω από τα όρια του πίνακα scores (μεγέθους 4) και η τιμή της είναι απροσδιόριστη.



Πέρασμα πίνακα ως παραμέτρου σε συνάρτηση

- Ένας πίνακας είναι δυνατόν να αποτελεί όρισμα μιας συνάρτησης.
- Η μορφή του πρωτοτύπου μιας συνάρτησης με όρισμα έναν μονοδιάστατο πίνακα είναι η εξής:

```
int f(int a[], int n);
```

Ενώ η κλήση της συνάρτησης γίνεται ως εξής:

```
int myArray[10];

f(myArray, 10);
```

- Στην πραγματικότητα, περνιέται ως παράμετρος η **διεύθυνση βάσης** του πίνακα.

Πέρασμα πίνακα ως παραμέτρου σε συνάρτηση (2)

- Προσοχή: οι πίνακες περνιούνται ως παράμετροι **με αναφορά**, δηλαδή είναι δυνατή η αλλαγή των στοιχείων του πίνακα που αποτελεί όρισμα στην κλήση μιας συνάρτησης.

Αρχικοποίηση πίνακα κατά τη δήλωσή του

Κατά τη δήλωση του παρακάτω πίνακα digits δίνονται αρχικές τιμές στα στοιχεία του.

```
int digits[10] =
  {1,2,3,4,5,6,7,8,9,10};
```

ή

```
int digits[] =
  {1,2,3,4,5,6,7,8,9,10};
```

Παράδειγμα (Εν. 11.3)

- Αντιστροφή Πίνακα

Ζητείται η κατασκευή ενός προγράμματος το οποίο

1. Διαβάζει μια λίστα ακεραίων μέχρι ο χρήστης να καταχωρήσει την τιμή-φρουρό (sentinel) 0.
2. Αντιστρέφει τα στοιχεία της λίστας
3. Εμφανίζει την αντεστραμμένη λίστα .

Αναλυτική σχεδίαση του προγράμματος

- Το πρόγραμμα αυτό αναφέρεται σε ένα “πρόβλημα” το οποίο αναλύεται στα εξής υποπροβλήματα:

1. Ανάγνωση των στοιχείων της λίστας
2. Αντιστροφή της λίστας
3. Εκτύπωση των στοιχείων της λίστας.

Για κάθε υποπρόβλημα ορίζουμε μια ανίστοιχη διαδικασία ή συνάρτηση και προκύπτει ο ακόλουθος σκελετός προγράμματος:

Ένας σκελετός προγράμματος (μπορεί να αλλάξει)

```
main()
{
  int list[NElements];

  GetIntegerArray(list);
  ReverseIntegerArray(list);
  PrintIntegerArray(list);
}
```

Συναρτήσεις με ορίσματα πίνακες

- Έστω η διαδικασία

```
void PrintIntegerArray(int list[]);
```

- Η παραπάνω διαδικασία δεν είναι δυνατόν να χρησιμοποιηθεί για πίνακα του οποίου ο αριθμός των στοιχείων δεν είναι γνωστός.
- Μια πιο γενική μορφή της είναι η παρακάτω:

```
void PrintIntegerArray(int list[], int n);
```

- Το δεύτερο όρισμα, n, ορίζει τον αριθμό των στοιχείων του πίνακα που θα χρησιμοποιηθούν (**τρέχον μέγεθος** του πίνακα).
- Το τρέχον μέγεθος πρέπει να είναι μικρότερο από το μέγεθος που καθορίζεται στη δήλωσή του (**κατανεμημένο μέγεθος**)

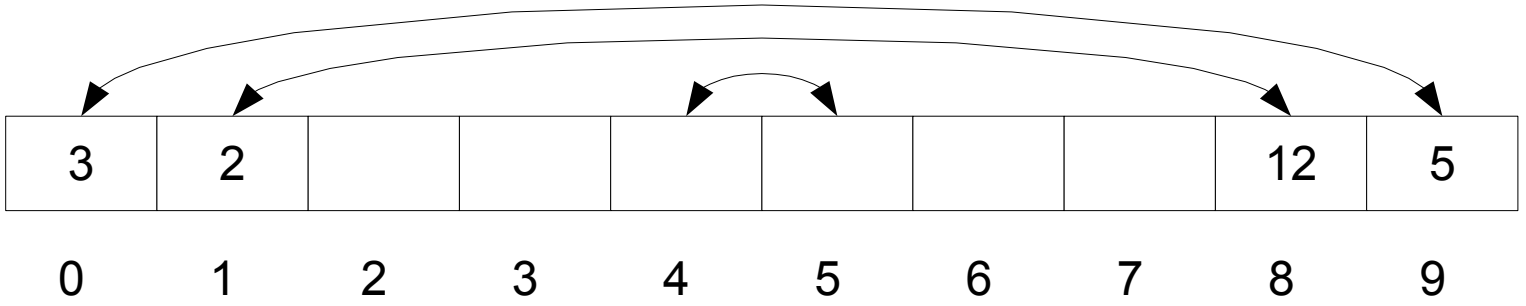
- Τα παραπάνω ισχύουν και για την συνάρτηση ReverseIntegerArray, της οποίας το πρωτότυπο γράφεται

```
void ReverseIntegerArray(int list[],
  int n);
```

- Αντίθετα, η συνάρτηση GetIntegerArray δεν ακολουθεί τον παραπάνω “κανόνα”, γιατί όταν καλείται δεν είναι γνωστό το τρέχον μέγεθος του πίνακα. Έτσι, γράφεται

```
int GetIntegerArray(int array[], int
  max, int sentinel);
```

Η συνάρτηση αντιστροφής




```
static void ReverseIntegerArray(int array[],
   int n)
{
    int i;

    for (i = 0; i < n / 2; i++) {
       SwapIntegerElements(array, i, n - i - 1);
    }
}
```

Το πλήρες πρόγραμμα αντιστροφής

- Η συνάρτηση SwapIntegerElements

```
static void SwapIntegerElements(int
  array[], int p1, int p2)
{
    int tmp;

    tmp = array[p1];
    array[p1] = array[p2];
    array[p2] = tmp;
}
```

- Το πλήρες πρόγραμμα αντιστροφής πίνακα.

Παράδειγμα μέτρησης γραμμάτων

- Το πρόγραμμα countlet.c.
- Για το χειρισμό χαρακτήρων και συμβολοσειρών βλ. κεφ. 9.

Λύση του προβλήματος

- Μέτρημα των γραμμάτων σε γραμμές.
  - Ανάγνωση γραμμής
  - Καταγραφή των γραμμάτων της γραμμής
- Εμφάνιση της συχνότητας εμφάνισης των γραμμάτων

```
main()
{
    int letterCounts[NLetters];


    printf("This program counts letter frequencies.\n");
    printf("Enter a blank line to signal end of
  input.\n");
    ClearIntegerArray(letterCounts, NLetters);
    CountLetters(letterCounts);
    DisplayLetterCounts(letterCounts);
}
```

Η συνάρτηση `ClearIntegerArray`

```
void ClearIntegerArray(int array[], int n)
{
    int i;

    for (i = 0; i < n; i++) {
        array[i] = 0;
    }
}
```

Η συνάρτηση **CountLetters**

```
static void CountLetters(int letterCounts[])
{
    string line;

    while (TRUE) {
        line = GetLine();
        if (StringLength(line) == 0) break;
        CountLettersInString(line,
  letterCounts);
    }
}
```

Η συνάρτηση `CountLettersInString`

```
static void CountLettersInString(string str,
  int letterCounts[])
{
    int i;

    for (i = 0; i < StringLength(str); i++) {
        RecordLetter(IthChar(str, i),
  letterCounts);
    }
}
```

Η συνάρτηση **RecordLetter**

```
void RecordLetter(char ch, int
  letterCounts[])
{
    int index;

    index = LetterIndex(ch);
    if (index != -1) letterCounts[index]++;
}
```

Η συνάρτηση **LetterIndex**

```
int LetterIndex(char ch)
{
    if (isalpha(ch)) {
        return (toupper(ch) - 'A');
    } else {
        return (-1);
    }
}
```