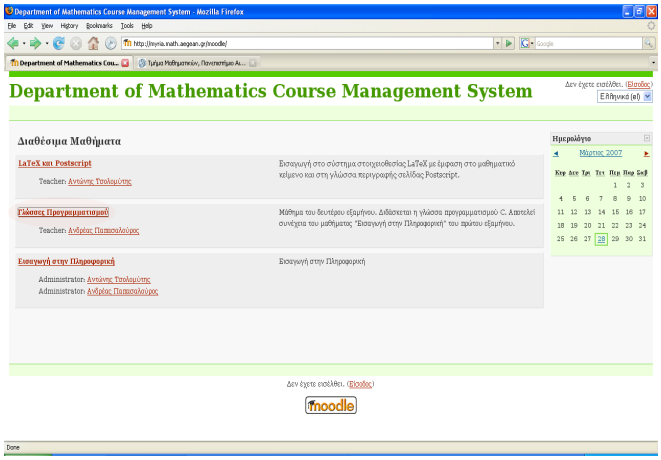
Γλώσσες Προγραμματισμού

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

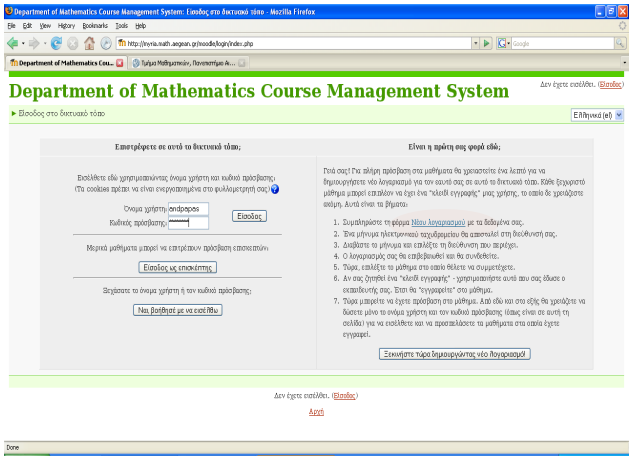
Πλατφόρμα ηλεκτρονικής μάθησης

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

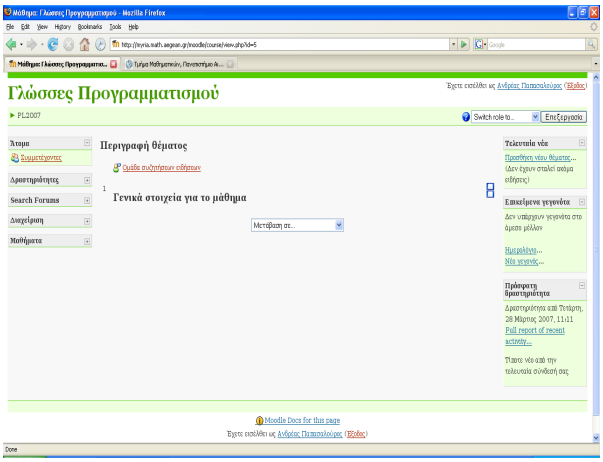
Η πλατφόρμα ηλεκτρονικής μάθησης moodle



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

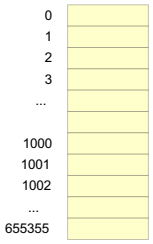
```
int i;
for (i = 0; i < SIZE; i++) {
   printf("size[%d] = %d", i,values[i]);
}
```

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.

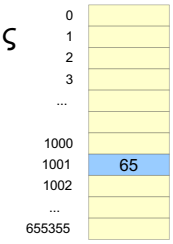


Διευθύνσεις μνήμης (2)

- Η δήλωση

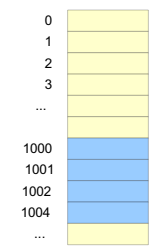
```
char ch = 'A';
```

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 */
}
```

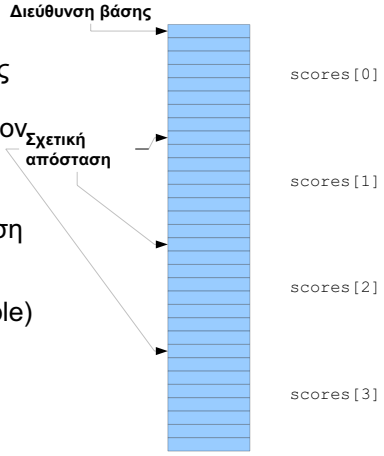
Κατανομή μνήμης σε πίνακες

- 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.

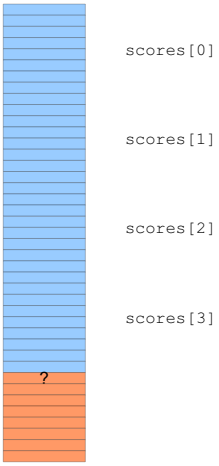


Αναφορά σε στοιχεία έξω από τα όρια πίνακα

- 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**)

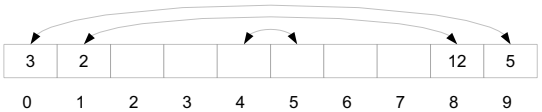
- Τα παραπάνω ισχύουν και για την συνάρτηση ReverseIntegerArray, της οποίας το πρωτότυπο γράφεται

```
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);
    }
}
```

Γλώσσες Προγραμματισμού

Πολυδιάστατοι πίνακες
Στατικές μεταβλητές
Στατική ανάθεση τιμών σε πίνακες

19/4/2007

Πολυδιάστατοι πίνακες

				5	2		4	8
		8		7	4	9		
	3	2			8			5
8	9				3		5	1
2	7	5		1		8	3	
	2				1	5		
1				4	5			3
5		4		3			2	6

- Παράδειγμα: Sudoku
- Πώς αναπαριστάται η διπλανή εικόνα από έναν πίνακα;
- Η λύση είναι η χρήση ενός **δισδιάστατου πίνακα**

Πολυδιάστατοι πίνακες

- Είναι *πίνακες πινάκων*.
- Συνηθέστερη περίπτωση αποτελούν οι δισδιάστατοι πίνακες
  - Ονομάζονται μήτρες (matrices, εν. matrix).
- Στη C επιτρέπονται πίνακες με 3 ή περισσότερες διαστάσεις.

Τρίλιζα (συνέχ).

X	O	O
	X	
		X

- Ο πίνακας της τρίλιζας μπορεί να αναπαρασταθεί από έναν δισδιάστατο πίνακα
- Δήλωση του δισδιάστατου πίνακα

```
char board[3][3];
int matrix[5][10];
```

Παράδειγμα

- Τα στοιχεία του πίνακα board είναι χαρακτήρες
- Η κατανομή στοιχείων στις “θέσεις” του πίνακα είναι η ακόλουθη

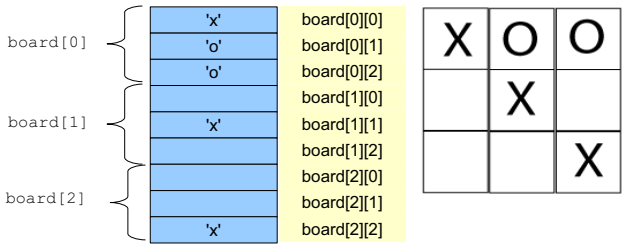
board[0][0] board[0][1] board[0][2]
board[1][0] board[1][1] board[1][2]
board[2][0] board[2][1] board[2][2]

- Αρχικοποίηση πίνακα κατά τη δήλωσή του

```
char board[][3] = {
   {'x','o','o'},
   {' ','o',' '},
   {' ',' ','x'}
};
```

X	O	O
	X	
		X

Εσωτερική αναπαράσταση πολυδιάστατων πινάκων



- Οι πολυδιάστατοι πίνακες αναπαριστώνται εσωτερικά ως μονοδιάστατοι πίνακες
- Στο παραπάνω παράδειγμα: Ο αριστερός αριθμοδείκτης παριστάνει αριθμό γραμμής ενώ ο δεξιός παριστάνει αριθμό στήλης. *Αυτό αποτελεί σύμβαση.*
- Η τιμή του πρώτου αριθμοδείκτη μεταβάλλεται ποιο γρήγορα από αυτή του δεύτερου.

Μεταβίβαση πολυδιάστατων πινάκων ως παραμέτρων σε συναρτήσεις

- Παράδειγμα συνάρτησης με όρισμα πολυδιάστατο πίνακα

```
static void DisplayBoard(char board[3][3])
{
   int row, column;
   for (row = 0; row < 3; row++) {
      if (row != 0) printf("---+---+---\n");
      for (column = 0; column < 3; column++) {
         if (column != 0 ) printf("|");
         printf(" %c ", board[row][column]);
      }
      printf("\n");
   }
}
```

- Η εκτέλεση της συνάρτησης εκτυπώνει τον πίνακα στην παρακάτω μορφή:

```
 x | o | x
---+---+---
   | x | o
---+---+---
 x |   | o
```

- Η επικεφαλίδα της συνάρτησης DisplayBoard μπορεί να γραφεί ως εξής:

```
static void DisplayBoard(char
 board[][3]);
```

Ανάθεση αρχικών τιμών σε πολυδιάστατους πίνακες

- Η δήλωση

```
static double identityMatrix[3][3] = {
   {1.0, 0.0, 0.0},
   {0.0, 1.0, 0.0},
   {0.0, 0.0, 1.0},
};
```

δηλώνει έναν πίνακα (μήτρα) αριθμών κινητής υποδιαστολής 3x3 (μοναδιαίος πίνακας).

Παράδειγμα

- Να κατασκευαστεί ένα πρόγραμμα το οποίο να γεμίζει με ακέραιες τιμές μεταξύ 0 και M έναν τετραγωνικό (δισδιάστατο) πίνακα NxN. Θεωρείστε ότι τα M και N είναι σταθερές του προγράμματος. Στη συνέχεια το πρόγραμμα να υπολογίζει και να εκτυπώνει τη συχνότητα εμφάνισης κάθε τιμής στον πίνακα r. Θα χρειαστεί ένας δεύτερος μονοδιάστατος πίνακας freq με M +1 στοιχεία.
- Λύση

Παράδειγμα 2

- Να γραφεί μια **κατηγορηματική** συνάρτηση η οποία δέχεται ως παράμετρο έναν δισδιάστατο πίνακα `a` ακέραιων αριθμών μεγέθους NxN και αποφασίζει αν ο πίνακας είναι συμμετρικός, δηλ. αν ισχύει `a[i][j]==a[j][i]` για κάθε τιμή των `i` και `j`.

```
#define N  3
bool isSymmetric(array[][N]);
```

Λύση

Στατικές μεταβλητές (Επαν.)

- Στατικές και δυναμικές μεταβλητές
- Δυναμικές μεταβλητές: Η ανάθεση τιμής γίνεται κατά την **εκτέλεση** του προγράμματος

```
int x = 1;
```

- Στατικές μεταβλητές: Η ανάθεση τιμής γίνεται **πριν την εκτέλεση** του προγράμματος

```
static int x = 1;
```

Καθολικές μεταβλητές (Εν. 10.2)

- Οι μεταβλητές που δηλώνονται στο σώμα μιας συνάρτησης ονομάζονται **τοπικές μεταβλητές**.
- Μεταβλητές που δηλώνονται έξω από τον ορισμό οποιασδήποτε συνάρτησης ονομάζονται **καθολικές μεταβλητές**.
- Οι καθολικές μεταβλητές είναι προσπελάσιμες από όλα τα σημεία ενός προγράμματος.
- Το τμήμα προγράμματος στο οποίο μπορεί να χρησιμοποιηθεί μια μεταβλητή ονομάζεται **εμβέλεια** της μεταβλητής.

Στατική ανάθεση αρχικών τιμών σε πίνακες

- Αν ένας πίνακας δηλωθεί ως **στατική καθολική μεταβλητή** είναι δυνατή η ανάθεση αρχικών τιμών στα στοιχεία του πριν ξεκινήσει η εκτέλεση του προγράμματος.

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

- Η παραπάνω δήλωση μπορεί να γραφεί

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

- Προσδιορισμός μεγέθους πίνακα στον οποίον έχουν έχουν ανατεθεί αρχικές τιμές.

```
static string cities[] =
  {"Athens", "Salonica", "Patras",
  "Volos"};
```

- Το μέγεθος (αριθμός στοιχείων) του παραπάνω πίνακα δίνεται από την παράσταση
- `sizeof cities / sizeof cities[0];`

Παράδειγμα

```
/*... εντολές include */
static string cities[ ] =  {"Athens", "Salonica",
  "Patras", "Volos"};
void printCities();
main()
{
   printCities();
}
void printCities()
{
   int i;
   for (i=0;i<4;i++)
      printf("%s",cities[i]);
}
```

Αναζήτηση και ταξινόμηση

Περιεχόμενα

- Αναζήτηση (searching): εύρεση ενός στοιχείου σε έναν πίνακα
- Ταξινόμηση (sorting): αναδιάταξη των στοιχείων ενός πίνακα ώστε να είναι τοποθετημένα με μια καθορισμένη σειρά
- Η αναζήτηση και η ταξινόμηση έχουν ιδιαίτερη σημασία στον προγραμματισμό
  - αποτελούν συχνά χρησιμοποιούμενες προγραμματιστικές τεχνικές
  - αποτελούν καλά παραδείγματα προβλημάτων για τα οποία υπάρχουν αλγόριθμοι με διαφορετική απόδοση

Παράδειγμα αναζήτησης

- Αναζήτηση του πρώτου φωνήεντος σε ένα αλφαριθμητικό

```
int FindFirstVowel(string word)
{
    int i;

    for (i = 0; i < StringLength(word); i++) {
        if (IsVowel(IthChar(word, i))) return (i);
    }
    return (-1);
}
```

- Για τις επισημασμένες συναρτήσεις δείτε το κεφ. 9.

Αναζήτηση σε πίνακα ακεραίων

- Η συνάρτηση αναζήτησης

```
int FindIntegerInArray(int key, int array[], int n)
{
   int i;

   for (i = 0; i < n; i++) {
      if (key == array[i]) return (i);
   }
   return (-1);
}
```

Παράλληλοι πίνακες

coinValues

0	1
1	2
2	5
3	10
4	20

coinNames

0	lepto
1	dilepto
2	pentalepto
3	dekalepto
4	eikosaletpo

- Οι πίνακες στους οποίους χρησιμοποιούνται αντίστοιχοι αριθμοδείκτες θέσης για την αποθήκευση συσχετιζόμενων τιμών ονομάζονται **παράλληλοι πίνακες**.

Παράδειγμα με χρήση παράλληλων πινάκων

- findcoin.c
- Το πρόγραμμα δέχεται ως είσοδο την αξία ενός νομίσματος και επιστρέφει το όνομά του.

Ένα δεύτερο παράδειγμα

- Ένα πρόγραμμα το οποίο
  - διαβάζει τα ονόματα δύο πόλεων
  - επιστρέφει την (απευθείας) απόσταση μεταξύ τους
- Το πρόγραμμα χρησιμοποιεί έναν δισδιάστατο πίνακα με τις αποστάσεις μεταξύ των πόλεων
- Το πρόγραμμα χρησιμοποιεί έναν μονοδιάστατο πίνακα με τα ονόματα των πόλεων

Ο κώδικας του προγράμματος

- Το πρόγραμμα υπολογισμού αποστάσεων πόλεων

Γραμμική αναζήτηση

- Αλγόριθμος γραμμικής αναζήτησης
  - Η αναζήτηση ξεκινά από την αρχή του πίνακα και διατρέχονται όλα του τα στοιχεία με τη σειρά, μέχρι να διαπιστωθεί ταύτιση ή να φτάσουμε στο τέλος του πίνακα.
- Ο αριθμός των βημάτων εκτέλεσης του αλγορίθμου (συγκρίσεων) είναι **ανάλογος** του μεγέθους του πίνακα.

Δυαδική αναζήτηση

- Θεωρούμε ότι ο πίνακας είναι ταξινομημένος

0	1
1	12
2	32
3	41
4	42
5	63
6	84
7	96
8	123
9	221
10	228
11	246

Αναζήτηση του 221

Συνάρτηση που υλοποιεί τη δυαδική αναζήτηση

```
static int FindStringInSortedArray(string key,
                                   string array[],
                                   int n)
{
    int lh, rh, mid, cmp;

    lh = 0;
    rh = n - 1;
    while (lh <= rh) {
        mid = (lh + rh) / 2;
        cmp = StringCompare(key, array[mid]);
        if (cmp == 0) return (mid);
        if (cmp < 0) {
            rh = mid - 1;
        } else {
            lh = mid + 1;
        }
    }
    return (-1);
}
```

Αποδοτικότητα του αλγορίθμου αναζήτησης

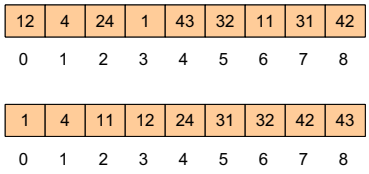
- Έστω πίνακας με N στοιχεία
- Μετά την πρώτη σύγκριση η αναζήτηση θα συνεχιστεί σε N/2 στοιχεία
- Αν k είναι ο αριθμός των βημάτων μέχρι να τελειώσει η αναζήτηση
  - $N = 2^k$
- Άρα $k = \log_2 N$

Σύγκριση γραμμικής και δυαδικής αναζήτησης

N	**$\log_2 N$**
10	3
100	7
1000	10
1.000.000	20
1.000.000.000	30

Ταξινόμηση

- Η διάταξη μιας λίστας τιμών (συνήθως σε μορφή πίνακα) σε μια καθορισμένη σειρά.
- Παράδειγμα: Ταξινόμηση ακεραίων



Μια συνάρτηση ταξινόμησης ακεραίων

```
void SortIntegerArray(int array[],int n);
```

- Μετά την εκτέλεση της παραπάνω συνάρτησης τα στοιχεία του πίνακα **`array`** μεγέθους **`n`** θα είναι ταξινομημένα

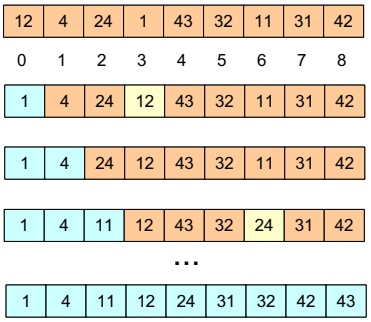
Αλγόριθμος ταξινόμησης με επιλογή

- Υπάρχει μια ποικιλία αλγορίθμων ταξινόμησης
- Θα παρουσιάσουμε έναν από τους απλούστερους: τον αλγόριθμο ταξινόμησης με επιλογή (selection sort).

Ταξινόμηση με επιλογή: περιγραφή του αλγορίθμου

1. Βρίσκουμε το μικρότερο στοιχείο του πίνακα
2. μετακινούμε το στοιχείο αυτό στην αριστερότερη θέση του πίνακα
3. επαναλαμβάνουμε τα βήματα 1 και 2 για τα **υπόλοιπα** στοιχεία του πίνακα

Ταξινόμηση με επιλογή: Παράδειγμα εκτέλεσης



Η παραπάνω διεργασία περιγράφεται από τον *ψευδοκώδικα*

for (*κάθε αριθμοδείκτη θέσης* `lh` *του πίνακα*) {

Αποθήκευσε στην `rh` *τον αριθμοδείκτη της μικρότερης τιμής μεταξύ* `lh` *και του τέλους της λίστας*

Αντιμετάθεσε τα στοιχεία με αριθμοδείκτες θέσης `lh` *και* `rh`

}

Υποπροβλήματα

- Η παραπάνω διαδικασία εισάγει δύο υποπροβλήματα:
  - Την εύρεση της θέσης του μικρότερου στοιχείου ενός πίνακα μεταξύ δυο προκαθορισμένων θέσεων

```
int FindSmallestInteger(int array[],
  int low, int high);
```

  - Την αντιμετάθεση δύο στοιχείων του πίνακα

```
void SwapIntegerElements(int array[],
  int low, int high);
```

(Γνωστή από το κεφ. 11).

Η συνάρτηση **`FindSmallestInteger`**

```
static int FindSmallestInteger(int array[], int low, int high)
{
    int i,spos;

    spos = low;
    for (i = low; i <= high; i++) {
        if (array[i] < array[spos]) spos = i;
    }
    return (spos);
}
```

Η συνάρτηση **`SwapIntegerElements`**

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

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

Η συνάρτηση ταξινόμησης

```
void SortIntegerArray(int array[], int n)
{
    int lh, rh;

    for (lh = 0; lh < n; lh++) {
        rh = FindSmallestInteger(array, lh, n-1);
        SwapIntegerElements(array, lh, rh);
    }
}
```

Ανάλυση του αλγορίθμου ταξινόμησης με επιλογή

- Έστω ταξινόμηση πίνακα μεγέθους N
- Κατά την ταξινόμηση με επιλογή απαιτούνται
- N + (N-1) +...+ 1 βήματα
- Ισχύει

$$N+N-1+N-2+..+3+2+1=\sum_{i=0}^{N-1}(N-i)=\frac{N^2+N}{2}$$

- Αλγόριθμοι που εκτελούνται με τέτοιο αριθμό βημάτων ονομάζονται **τετραγωνικοί** και δεν είναι αποδοτικοί για μεγάλες τιμές του N

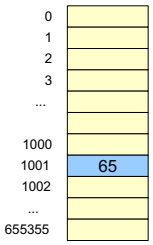
Δείκτες

Μεταβλητές

- Έστω η μεταβλητή

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

- Η παραπάνω δήλωση δεσμεύει χώρο στη μνήμη για τη μεταβλητή **ch**.
- Η τιμή της ch αντιστοιχεί στο περιεχόμενο μιας θέσης μνήμης, π.χ. της 1001.



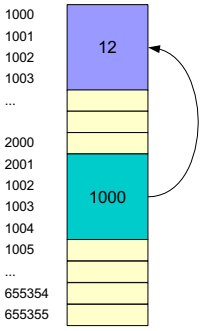
Δείκτες

- Στη C επιτρέπονται μεταβλητές που περιέχουν τη διεύθυνση μιας άλλης μεταβλητής.

```
int i = 12;
```

- Η δήλωση ενός δείκτη γίνεται ως εξής στη θέση μνήμης της μεταβλητής i γίνεται ως εξής:

```
int *p;
p = &i;
```



Δήλωση δεικτών

τύπος_βάσης * *μεταβλητή_δείκτη*;

όπου

τύπος_βάσης ο τύπος της μεταβλητής που δείχνει ο δείκτης

μεταβλητή_δείκτη είναι το όνομα της μεταβλητής που δηλώνεται

```
int *p;
char *cptr;
```

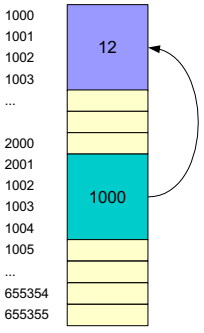
Τελεστές για το χειρισμό δεικτών

& διεύθυνση

* τιμή στην οποία δείχνει

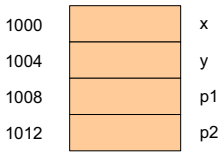
Παράδειγμα

```
int *p;
p = &i;

Η p έχει την τιμή 1000
Η *p "    "   "   12
```



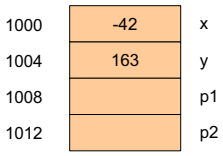
Ένα ακόμη παράδειγμα

int x,y;
int *p1, *p2;



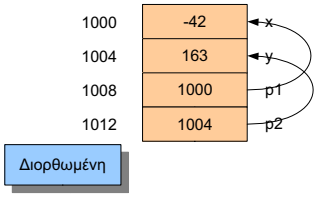
Ένα ακόμη παράδειγμα

x= - 42;
y = 163;



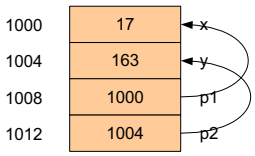
Ένα ακόμη παράδειγμα

p1 = &x;
p2 = &y;



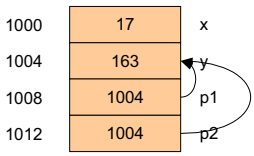
Ένα ακόμη παράδειγμα

*p1 = 17;



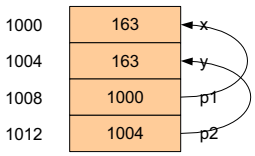
Ένα ακόμη παράδειγμα

p1 = p2;



Εναλλακτικά

*p1 = *p2;



Ο ειδικός δείκτης NULL

Ειδική τιμή που δηλώνει ότι ένας δείκτης “δείχνει” σε μια μη έγκυρη διεύθυνση.

Δεν επιτρέπεται η πρόσβαση στην τιμή ενός δείκτη που έχει τιμή NULL. Μια τέτοια κλήση έχει ως αποτέλεσμα, συνήθως, την κατάρρευση του προγράμματος.

```
p = NULL;
*p = 1; /* Το πρόγραμμα θα καταρρεύσει */
```

Μεταβίβαση παραμέτρων με αναφορά

- Η κλήση μιας συνάρτησης στη C δεν αλλάζει τις τιμές των παραμέτρων της συνάρτησης (κλήση με τιμή)
- Παράδειγμα

```
void SetToZero(int var)
{
   var = 0;
}
```

- Η κλήση της παραπάνω συνάρτησης δεν έχει κανένα αποτέλεσμα πάνω στο όρισμά της

- Η κλήση

```
int x=10;
SetToZero(x);
/* το x εξακολουθεί να έχει την τιμή
  10... */
```

- δεν έχει κανένα αποτέλεσμα

Κλήση με αναφορά

- Για την επίλυση του παραπάνω προβλήματος περνάμε ως παράμετρο όχι τη μεταβλητή αλλά τη διεύθυνσή της.

```
void SetToZero(int *ip)
{
   *ip=0;
}
```

Στην περίπτωση αυτή η κλήση της συνάρτησης αλλάζει την τιμή της παραμέτρου

- ΠΡΟΣΟΧΗ στον τρόπο περάσματος της διεύθυνσης της μεταβλητής.

Κλήση με αναφορά (2)

```
int x = 10;
SetToZero(&x);
/* Τώρα είναι x = 0 */
```

Νέα συνάρτηση ανταλλαγής ακεραίων

```
void SwapInteger(int *p1, int *p2)
{
   int tmp;
   tmp = *p1;
   *p1 = *p2;
   *p2 = tmp;
}
```

Μια συνάρτηση που επιστρέφει πολλές τιμές

- Είσοδος: Ένα χρονικό διάστημα σε λεπτά της ώρας
- Έξοδος: το χρονικό διάστημα σε ώρες και λεπτά
- Η επικεφαλίδα της αντίστοιχης συνάρτησης:

```
void ConvertTimeToHM(int time, int
  *pHours, int *pMinutes);
```

Παράδειγμα

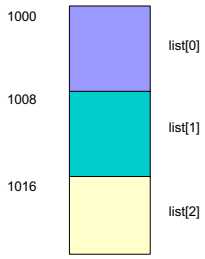
```
#define MinutesPerHour 60

/* Function prototypes */

static void ConvertTimeToHM(int time, int *pHours, int *pMinutes);

main()
{
    int time, hours, minutes;

    printf("Test program to convert time values\n");
    printf("Enter a time duration in minutes: ");
    time = GetInteger();
    ConvertTimeToHM(time, &hours, &minutes);
    printf("HH:MM format: %d:%02d\n", hours, minutes);
}

static void ConvertTimeToHM(int time, int *pHours, int *pMinutes)
{
    *pHours = time / MinutesPerHour;
    *pMinutes = time % MinutesPerHour;
}
```

Δείκτες και πίνακες

- Η παράσταση

double list[3];

ορίζει έναν πίνακα 3 στοιχείων



- Το **`&list[0]`** είναι η διεύθυνση του πρώτου στοιχείου του πίνακα
- Η έκφραση &list[0] είναι ισοδύναμη με την **`list`**.

Πίνακες ως ορίσματα συναρτήσεων

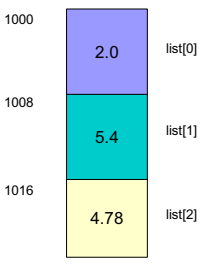
- Η δήλωση

```
void f(int a[]);
```

- είναι ισοδύναμη με την

```
void f(int *a);
```

Πίνακες ως δείκτες



- Η διεύθυνση του στοιχείου list[1] είναι η list + 1 *sizeof (double)
- Η παράσταση list + 1 αντιστοιχεί στην παραπάνω διεύθυνση (ο υπολογισμός γίνεται αυτόματα)

*(list + 1) = 5.4;

Αριθμητική δεικτών

- Η έκφραση

`p + n`

- όπου p δείκτης και n ένας ακέραιος
- εννοεί τη διεύθυνση του n-οστού αντικειμένου μετά από αυτό που δείχνει ο p.
- Έχουν νόημα οι εξής παραστάσεις
- **`p - k`**
- **`p1 - p2`**

Οι τελεστές ++ και --

- Μεταθεματική μορφή

  i++

  Αποτιμάται η έκφραση και και μετά αυξάνει κατά 1

  i--
- Προθεματική μορφή

  ++i

  Πρώτα αυξάνει κατά 1 και μετά αποτιμάται η έκφραση

  --i

Παραδείγματα

```
int x,y;
x = 5;
y = ++x;
```

Αλλά

```
int x,y;
x = 5;
y = x++;
```

Αύξηση και μείωση δεικτών

- Η έκφραση
- *p++
- ισοδυναμεί με την
- *(p++)
-
- Η έκφραση
- p++;
- για ένα δείκτη σε ακαίρεο αυξάνει τον δείκτη ώστε να δείχνει στον επόμενο ακέραιο.

Δυναμική κατανομή μνήμης: malloc και free

- Η συνάρτηση malloc

int *p = (int *) malloc(sizeof int);

- Δυναμικοί πίνακες

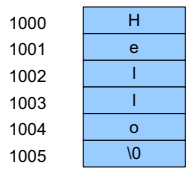
char * cp = (char *) malloc(10 * sizeof (char));

- Η συνάρτηση
  - free(void * p);
- αποδεσμεύει την περιοχή μνήμης στην οποία δείχνει ο p και η οποία είχε προηγουμένως δεσμευτεί.

Αλφαριθμητικά, πίνακες και δείκτες

Τα αλφαριθμητικά ως πίνακες

- Ένα αλφαριθμητικό (string) αναπαρίσταται εσωτερικά ως ένας πίνακας χαρακτήρων που τερματίζεται από τον ειδικό χαρακτήρα NULL ('\0')



"Hello"

Τα αλφαριθμητικά ως πίνακες (συνέχ).

- Το παραπάνω δημιουργείται με τις εξής εντολές:

```
char carray[6];

carray[0] = 'H';
carray[1] = 'e';
carray[2] = 'l';
carray[3] = 'l';
carray[4] = 'o';
carray[5] = '\0';
```

Τα αλφαριθμητικά ως πίνακες (συνέχ.)

- Το ίδιο αποτέλεσμα έχουν και οι εντολές

```
char carray[] = "Hello";

char carray[6] = "Hello";

char carray[] = {'H','e','l','l','o','\0'};

char carray[6] = {'H','e','l','l','o','\0'};
```

Τα αλφαριθμητικά ως πίνακες (συνέχ.)

- Για την προσπέλαση κάθε χαρακτήρα ενός αλφαριθμητικού:

```
int i;
char s[] = "Hello, John";
for (i=0; s[i] != '\0';i++)
{
   κάνε κάτι με το στοιχείο s[i]
}
```

Παράδειγμα: Η συνάρτηση FindFirstVowel

- Με χρήση αφηρημένων αλφαριθμητικών (βιβλιοθήκη strlib)

```
int FindFirstVowel(string word)
{
   int i;
   for (i=0; i<StringLength(word); i++) {
      if (IsVowel(IthChar(word,i)) return (i);
   }
   return (-1);
}
```

Παράδειγμα: Η συνάρτηση FindFirstVowel

- Με χρήση πινάκων

```
int FindFirstVowel(char word[])
{
   int i;
   for (i=0; word[i] != '\0'; i++) {
      if (IsVowel(word[i])) return (i);
   }
   return (-1);
}
```

Παράδειγμα: Η συνάρτηση FindFirstVowel

- Με χρήση δεικτών

```
int FindFirstVowel(char * word)
{
   char *cp;
   for (cp = word; *cp != '\0' ; cp++) {
      if (IsVowel(*cp) return (cp - word);
   }
   return (-1);
}
```

Τα αλφαριθμητικά ως αφηρημένος τύπος

Η δήλωση

```
char *s;
```

είναι ισοδύναμη με την

```
string s;
```

Στη βιβλιοθήκη genlib.h υπάρχει ο ακόλουθος ορισμός

```
typedef char *string;
```

Δείκτες και μεταβλητές πίνακα για αλφαριθμητικά

- Το παρακάτω τμήμα κώδικα είναι έγκυρο

```
char *carray;
carray = “A fool on the hill”;
```

- Το παρακάτω τμήμα κώδικα δεν θα μεταγλωττιστεί

```
char carray[32];
carray = “Another Green World”;
```

- Θα προκύψει ένα μήνυμα σφάλματος στη μεταγλώττιση της μορφής “Lvalue required...”

Η βιβλιοθήκη strlib.h (ANSI C)

- strcpy(dst, src)
- strncpy(dst, src, n)
- strcat(dist, src)
- strncat(dist, src,n)
- strlen(s)
- strcmp(s1,s2)
- strchr(s,ch)
- strrchr(s, ch)
- strstr(strpattern,string)

Η συνάρτηση strcpy

```
strcpy(char dst[], char src[])
```

Αντιγράφει τη συμβολοσειρά src στην συμβολοσειρά dst.

Παράδειγμα

```
char buffer[128];
strcpy(buffer,”Elearnor Rigby”);
```

Μια υλοποίηση της strcpy

```
void strcpy(char dst[], char src[])
{
  int i;
  for (i = 0; src[i] != '\0'; i++) {
    dst[i] = src[i];
  }
  dst[i] = '\0';
}
```

Μια “παραδοσιακή” υλοποίηση της strcpy

```
void strcpy(char *dst, char *src)
{
    while (*dst++ = *src++);
}
```

Υπερχείλιση

- Με τη χρήση της strcpy είναι δυνατόν να συμβεί υπερχείλιση περιοχής προσωρινής αποθήκευσης (buffer overflow)

char carray[6];

strcpy(carray, “This is a long string”);

Η συνάρτηση strcat

strcat (char dest[], char src[])

Αντιγράφει τη συμβολοσειρά dest στο τέλος της συμβολοσειράς src

Παράδειγμα

char name[128];

strcpy(name, “John”);

strcat(name, “ ”);

strcat(name, “Lennon”);

Η συνάρτηση strncat

strncat (char dest[], char src[], int n)

Αντιγράφει n το πολύ χαρακτήρες της συμβολοσειράς dest στο τέλος της συμβολοσειράς src

Παράδειγμα

char name[128];

strcpy(name, “John”);

strcat(name, “ ”);

strncat(name, “Lennon”, 3); /* “John Len” */

Συναρτήσεις αναζήτησης

char * strchr(char s[], char ch)

Επιστρέφει έναν δείκτη προς τη θέση σε ένα αλφαριθμητικό όπου βρέθηκε ο χαρακτήρας ch

char * strrchr(char s[], char ch)

Ίδια λειτουργία με την strchr ξεκινώντας από το τέλος του αλφαριθμητικού

char * strstr(char s1, char s2[])

Αναζητά την πρώτη εμφάνιση της συμβολοσειράς s2 στην s1.

Εφαρμογή 1

- Να φτιαχτεί συνάρτηση η οποία δέχεται ως είσοδο μια συμβολοσειρά και επιστρέφει τη **θέση του μεγαλύτερου στη σειρά χαρακτήρα** της συμβολοσειράς, σύμφωνα με τη διάταξη των χαρακτήρων

'a', 'b', ...,'x','y',z', 'A', 'B', ..., 'X', 'Y', 'Z',...

Παράδειγμα

για τη συμβολοσειρά “another” επιστρέφεται το 3, η θέση του 't'.

Εφαρμογή 2

- (Άσκηση 1 σελ 600). Να φτιαχτεί μια συνάρτηση η οποία δέχεται ως όρισμα ένα αλφαριθμητικό ως πίνακα χαρακτήρων και επιστρέφει το μέγεθος του αλφαριθμητικού. Πρόκειται για εναλλακτική υλοποίηση της **`strlen`**.
-
- Να γίνει με for και με while

Η βιβλιοθήκη ctype (επιλογή)

- **`isalnum,isalpha`** Ελέγχει αν ένας χαρακτήρας είναι αλφαριθμητικός (A-Z, a-z, 0-9). *Επιστρέφει 1 σε περίπτωση αν ο χαρακτήρας είναι αλφαριθμητικός και 0 αλλιώς*
- **`isdigit`** Ελέγχει αν ένας χαρακτήρας είναι ψηφίο (0-9)
- **`islower`** Ελέγχει αν ένας χαρακτήρας είναι μικρό (λατινικό) γράμμα (a-z).
- **`isupper`** Ελέγχει αν ένας χαρακτήρας είναι κεφαλαίο (λατινικό) γράμμα (A-Z).

Ένα σύνθετο παράδειγμα

- Πρόγραμμα invert.c σελ. 594.
- Πρόγραμμα αντιστροφής ενός ονόματος.
- Παράδείγματα:
  - Δίνοντας το όνομα “Andreas F. Papasalouros” παίρνουμε το “Papasalouros, Andreas F.”
  - Δίνοντας “Steve Miller” παίρνουμε “Miller, Steve”

Η συνάρτηση scanf

Περιγραφή

- Συνάρτηση για **είσοδο** δεδομένων
- Αντίστοιχη με την `printf`
- Μορφή της scanf

scanf(*αλφαριθμητικό ελέγχου, όρισμα1*, *όρισμα2*, …);

Το αλφαριθμητικό ελέγχου περιέχει περιέχει έναν ή περισσότερους *κωδικούς μορφοποίησης:* %d, %f, %c, %s*, κ.λπ.*

Ανάγνωση συμβολοσειρών με την scanf

```
#define MaxWord 32

char word[MaxWord];
scanf("%s", word);
```

Ανάγνωση ακεραίων με τη scanf

```
int x;
printf("Δώστε έναν ακέραιο: ");
scanf("%d", &x);
```

Ανάγνωση πραγματικών με τη scanf

```
double x;
printf("Δώστε έναν πραγματικό
  αριθμό: ");
scanf("%f", &x);
```

Ανάγνωση χαρακτήρων με τη scanf

```
char ch;
printf("Δώστε έναν πραγματικό
  αριθμό: ");
scanf("%c", &ch);
```

Μορφοποιημένη είσοδος

- Κάθε ένα από τα ορίσματα της scanf μετά τη *συμβολοσειρά ελέγχου* είναι μια *διεύθυνση* της μεταβλητής στην οποία θα αποθηκευτεί η αντίστοιχη τιμή.
- Η scanf χρησιμοποιείται για τη "μορφοποιημένη είσοδο" περισσότερων από μιας τιμών σε μια κλήση της

Παράδειγμα 1

```
#include <stdio.h>

main()
{
  char name[32];
  int age = 0;

  while (1) /* Η while (TRUE) */
  {
     scanf("%s %d", name, &age);
     if (age < 0) break;
     printf("Name: %s age: %d\n", name, age);
  }
}
```

Ανάγνωση των στοιχείων ενός πίνακα με τη `scanf`

```
#include <stdio.h>
#define N 5
main()
{
  int i;
  int array[N];

  for (i = 0; i < N; i++) {
     printf("array[%d] = ? ", i);
     scanf("%d", &array[i]);
  }

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

Άσκηση 1

Να γραφεί μια συνάρτηση

`int bin2dec(char s[])`

η οποία δέχεται ως όρισμα ένα αλφαριθμητικό που παριστάνει έναν δυαδικό αριθμό και επιστρέφει την αριθμητική τιμή που αντιστοιχεί στο αλφαριθμητικό. συνάρτηση επιστρέφει -1 αν το όρισμα s δεν είναι έγκυρο.

Λύση

```
int bin2dec(char s[])
{
    int i;
    int value = 0;
    for (i = 0; s[i] != '\0'; i++) {
          value = value * 2;
          if (s[i]=='1')
                value ++;
          else if (s[i] == '0')
                  continue;
          else return -1;
    }
    return value;
}
```

Άσκηση 2

Να γραφεί μια συνάρτηση

`int atoi2(char s[])`

η οποία μετατρέπει μια συμβολοσειρά s που παριστάνει έναν ακέραιο στην αντίστοιχη ακέραια τιμή. Η συνάρτηση επιστρέφει -1 αν το όρισμα s δεν είναι έγκυρο.

Λύση

```
int atoi2(char s[])
{
   int i;
   int value = 0;

   for (i=0; s[i] != '\0'; i++)
   {
       value = value * 10;
       if (s[i] > '9' || s[i] < '0') return -1;
       value = value + (s[i] - '0');
   }
   return value;
}
```

Αναδρομή – Ανάλυση Αλγορίθμων

Παράδειγμα: Υπολογισμός του παραγοντικού

- Ορισμός του n!
  n! = n x (n - 1) x … x 2 x 1
- Ο παραπάνω ορισμός μπορεί να γραφεί ως

$$n! = \begin{cases} 1 & \text{αν } n = 0 \\ n \times (n-1)! & \text{αλλιώς} \end{cases}$$

Παράδειγμα (συνέχ).

- Στον εναλλακτικό ορισμό, ο υπολογισμός του n! (σύνθετο πρόβλημα) ανάγεται στον υπολογισμό του (n-1)! (απλούστερο πρόβλημα).

Αναδρομή

- Στρατηγική επίλυσης προβλημάτων
- Τεχνική προγραμματισμού σύμφωνα με την οποία ένα σύνθετο πρόβλημα ανάγεται σε ένα απλούστερο **της ίδιας μορφής**.
- Είναι μια *παράξενη*, μη *διαισθητική τεχνική* η οποία στην αρχή δυσκολεύει τους προγραμματιστές
- Σε κάθε περίπτωση όπου χρησιμοποιείται αναδρομή, είναι δυνατόν, εναλλακτικά, να χρησιμοποιηθεί επανάληψη
- Η χρήση της αναδρομής προσφέρει, σε κάποιες περιπτώσεις, απλούστερες λύσεις.

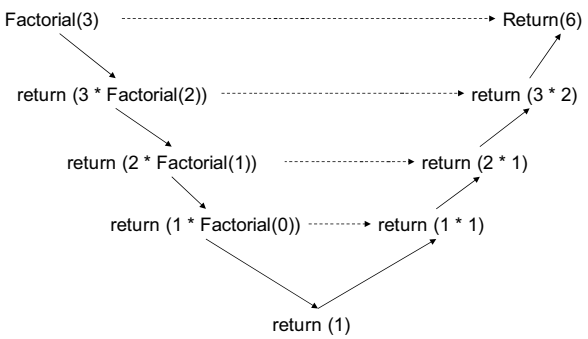
Η αναδρομή στη C

- Η αναδρομή στη C πραγματοποιείται ορίζοντας συναρτήσεις που καλούν **τον εαυτό τους** (άμεση αναδρομή).
- Υποστηρίζεται και η **έμμεση** αναδρομή

Παράδειγμα: Η συνάρτηση Factorial

```
int Factorial(int n)
{
  if (n == 0)
    return 1;
  return (n * Factorial(n-1));
}
```

Παράδειγμα κλήσης της Factorial



Επαναληπτική υλοποίηση

```
int Factorial(int n)
{
  int i, result;
  result = 1;
  for (i = 2; i<= n; i++)
  {
    result = result * i;
  }
  return result;
}
```

Βασική δομή αναδρομικής συνάρτησης

```
συνάρτηση()
{

   if (έλεγχος απλής περίπτωσης) {
     return (απλή λύση χωρίς τη χρήση αναδρομής);
   } else {
     return (αναδρομική κλήση που απαιτεί την κλήση της
     ίδιας συνάρτησης);
}
```

Ένα παράδειγμα: Συνάρτηση ύψωσης στη δύναμη

```
static int RaiseIntToPower(int n, int k)
{
   /* Ελεγχος απλής περίπτωσης */
   if (k ==0) {
      return (1);
   } else {
   /* Αναδρομική κλήση της ίδιας συνάρτησης */
      return (n * RaiseIntToPower(n, k-1));
   }
}
```

Κατά τη χρήση της αναδρομής

- Βρίσκουμε τη λύση για την απλή περίπτωση.
- «Φανταζόμαστε» ότι έχουμε βρει τη λύση για μια (ή περισσότερες) απλούστερες περιπτώσεις
- Με βάση τις παραπάνω απλούστερες περιπτώσεις, συνθέτουμε τη λύση για τη γενικότερη περίπτωση

Τι θα συμβεί αν λείπει ο έλεγχος της απλής περίπτωσης;

- Τι κάνει το παρακάτω πρόγραμμα;

```
main() { inc(1);}

int inc(int k)
{
   k++;
   inc(k);
}
```

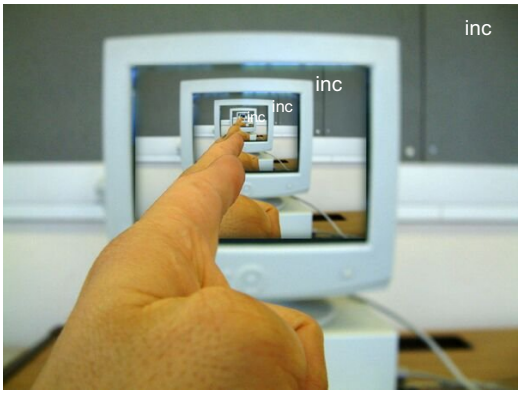
Περίπτωση «ατέρμονος» αναδρομής

- Θεωρητικά, η προηγούμενη συνάρτηση, inc, όταν κληθεί, καλεί επ' άπειρον τον εαυτό της
  - Το πρόγραμμα μπαίνει σε μια ατέρμονα επανάληψη («κολλάει»)
- Πρακτικά, μετά από κάποιο χρόνο το πρόγραμμα τερματίζει εμφανίζοντας ένα μήνυμα σφάλματος.
  - π.χ. `Stack overflow` ή
  - `Segmentation Fault`

Γιατί συμβαίνει αυτό;

- Σε κάθε κλήση συνάρτησης (αναδρομική ή όχι) δεδομένα εγγράφονται σε μια περιοχή της μνήμης που ονομάζεται Στοίβα (Stack).
- Τέτοια δεδομένα είναι:
  - Οι τιμές των πραγματικών παραμέτρων
  - Η διεύθυνση επιστροφής
  - Τοπικές μεταβλητές
- Μετά από ένα μεγάλο αριθμό αναδρομικών κλήσεων η στοίβα υπερχειλίζει (overflow) και το πρόγραμμα τερματίζει με την εμφάνιση κατάλληλου μηνύματος.

Περίπτωση άπειρων αναφορών μιας οντότητας στον εαυτό της



Μαθηματικά

- Η **μαθηματική επαγωγή**, ως μέθοδος απόδειξης, είναι μια περίπτωση αναδρομής

Η αναδρομή στην Τέχνη

M.C. Escher, Circle limit IV

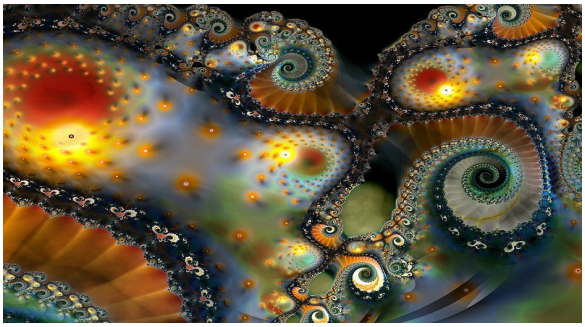
Regular Division of the Plane VI

Η αναδρομή στην Τέχνη

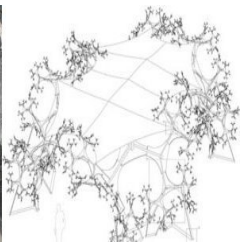
M. C. Escher, Square Limit

Circle Limit III

Fractals

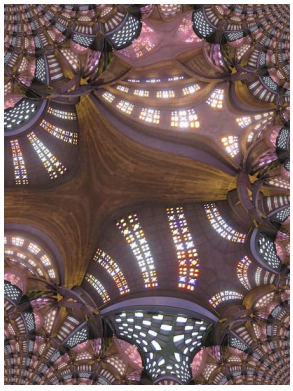


Αρχιτεκτονική/Γλυπτική



http://www.theverymany.net/2006_05_01_archive.html

Αρχιτεκτονική

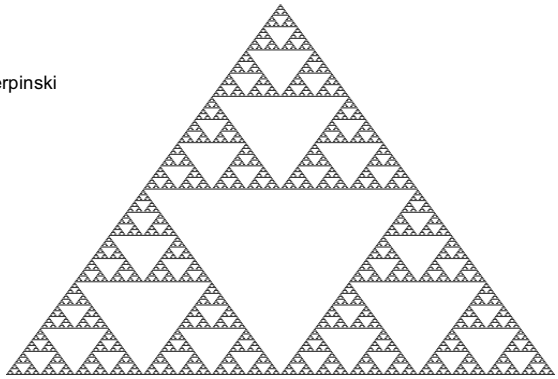


St. Joseph's Church, Havre

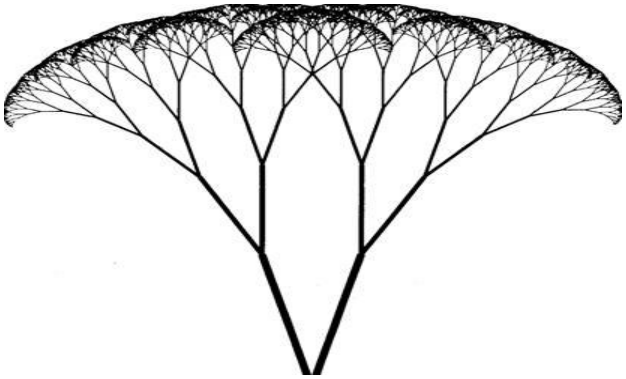
Auguste Perret (and R. Audigier) 1953-57

Fractals (2)

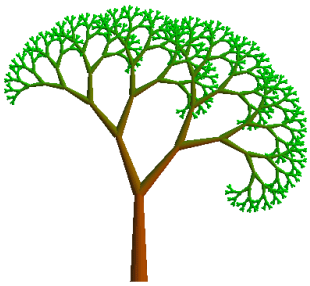
Τρίγωνο Sierpinski



Δένδρο Fractal (1)



Δένδρο Fractal (2)



Αναδρομή … στη φύση (1)

Αναδρομή … στη φύση (2)

Κούκλες «Μπαμπούσκα»

- Μια κούκλα μέσα σε μια ίδια κούκλα
- Μια συνάρτηση καλείται από μια «ίδια» συνάρτηση

Άλλα πεδία όπου είναι δυνατόν να προκύψει αναδρομή

- Λογοτεχνία
- Γλώσσα
- Μουσική
- …

Παράδειγμα: Αριθμοί Fibonacci

```
int fib(int n)
{
  if (n==0)
     return 0;
  if (n==1)
     return 1;
  return (fib(n-1) + fib(n-2));
}
```

Πλήρες πρόγραμμα

```
#include <stdio.h>

int fib(int n)
{
   if (n==0)
        return 0;
   if (n==1)
        return 1;
   return (fib(n-1) + fib(n-2));
}

main()
{
   int x;
   while(1){
        scanf("%d",&x);
        if (x<0)
                return;
        printf("%d\n", fib(x));
   }
}
```

Ανάλυση αλγορίθμων

Αξιολόγηση της αποδοτικότητας αλγορίθμων

- Πόσο γρήγορα “εκτελείται” ένας αλγόριθμος;
- Η αξιολόγηση της **σχετικής** αποδοτικότητας αλγορίθμων είναι γνωστή ως **ανάλυση αλγορίθμων**
- Σχετική αποδοτικότητα
  - Πόσο αποδοτικότερος είναι ένας αλγόριθμος σε σχέση με έναν άλλον

Σχετική αποδοτικότητα

- Δεν ενδιαφέρει η απόλυτη ταχύτητα εκτέλεσης ενός αλγορίθμου καθώς εξαρτάται από
  - την ταχύτητα του εκάστοτε υπολογιστή στον οποίο εκτελείται
  - τη γλώσσα προγραμματισμού στον οποίο είναι υλοποιημένος
  - άλλους παράγοντες

Υπολογιστική πολυπλοκότητα

- Πόσο ποιο αργή είναι η ταξινόμηση ενός πίνακα 1000 στοιχείων σε σχέση με την ταξινόμηση ενός πίνακα 10000 στοιχείων;
- Το μέγεθος ενός προβλήματος (π.χ. το μέγεθος του πίνακα) παριστάνεται με το N
- "Η σχέση του N και του χρόνου εκτέλεσης ενός αλγορίθμου, όταν το N είναι μεγάλο, ονομάζεται **υπολογιστική πολυπλοκότητα** του συγκεκριμένου αλγορίθμου".

Μέτρο πολυπλοκότητας

- Στην επιστήμη των υπολογιστών χρησιμοποιείται ένας ειδικός συμβολισμός που παριστάνει το **μέτρο της πολυπλοκότητας ενός αλγορίθμου**.
- Συμβολισμός του μεγάλου όμικρον O (από τη λέξη Order (τάξη).
- Παραδείγματα O(1), $O(N^2)$, O(logN), κ.λπ.

Παραδείγματα: O(1)

- Ένας αλγόριθμος για την εύρεση του **πρώτου** στοιχείου ενός πίνακα **δεν εξαρτάται** από το μέγεθος του πίνακα.
  - Λέμε ότι ο αλγόριθμος εκτελείται σε σταθερό χρόνο
  - Ο σταθερός χρόνος συμβολίζεται ως **O(1)**

Παραδείγματα: O(N)

- Αλγόριθμος γραμμικής αναζήτησης ενός στοιχείου σε έναν πίνακα:
- Για έναν πίνακα με N στοιχεία χρειάζονται N βήματα (συγκρίσεις)
- Ο αλγόριθμος εκτελείται σε **γραμμικό χρόνο**.
- Ο γραμμικός χρόνος εκτέλεσης συμβολίζεται ως O(N).

Παραδείγματα: O(logN)

- Αλγόριθμος δυαδικής αναζήτησης ενός στοιχείου σε έναν πίνακα:
- Για έναν πίνακα με N στοιχεία χρειάζονται (περίπτου) $\log_2 N$ βήματα (συγκρίσεις)
- Ο αλγόριθμος εκτελείται σε **λογαριθμικό χρόνο**.
- Ο γραμμικός χρόνος εκτέλεσης συμβολίζεται ως O(logN).

Παράδειγμα: $O(N^2)$

- Ο αλγόριθμος της ταξινόμησης με επιλογή εκτελείται σε

$$N+N-1+N-2+..+3+2+1=\sum_{i=0}^{N-1}(N-i)=\frac{N^2+N}{2}$$

- Στον παραπάνω τύπο:
  - Απαλοίφονται οι όροι που δεν είναι σημαντικοί για μεγάλες τιμές του N
  - Απαλοίφονται σταθεροί συντελεστές
- Η επίδοση του αλγορίθμου είναι $O(N^2)$. Λέμε ότι ο αλγόριθμος εκτελείται σε **τετραγωνικό χρόνο**.

Παράδειγμα: O(NlogN)

- Ο αλγόριθμος ταξινόμησης με συγχώνευση
  - Ο πίνακας διαιρείται σε δύο υπο-πίνακες μισού μεγέθους από τον αρχικό
  - Αν ο πίνακας δεν έχει κανένα ή έχει μόνο ένα στοιχείο τότε είναι ταξινομημένος
  - Οι δύο υποπίνακες ταξινομούνται (αναδρομικά)
  - Οι δύο ταξινομημένοι υποπίνακες συγχωνεύονται
- Αλγόριθμος “διαίρει και βασίλευε”.
- Ο χρόνος εκτέλεσης του αλγορίθμου είναι $N\log_2 N$

Ανάλυση του αλγορίθμου

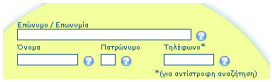
- Απαιτούνται
  - N βήματα στο πρώτο επίπεδο
  - 2 N/2 βήματα στο δεύτερο επίπεδο
  - 4 N/4 βήματα στο τρίτο επίπεδο
  - 8 N/8 βήματα στο τέταρτο επίπεδο
  - ...
- Ο χρόνος εκτέλεσης του αλγορίθμου είναι $N\log_2 N$

```
 right)
{
  int i, last;
  if (left>= right) return;
  swap(v,left, left+right)/2);
  last = left;
  for (i = left+1; i <= right; i++)
    if (v[i] < v[left]) {
      last++;
      swap(v, last, i);
    }
    qsort(v,left, last-1);
    qsort(v, last+1, right);
  }
```

Μέτρο πολυπλοκότητας (συνέχ.)

- Με τον παραπάνω συμβολισμό ορίζονται κατηγορίες (κλάσεις) αλγορίθμων ανάλογα με τον χρόνο εκτέλεσής τους.
- Έτσι, δύο αλγόριθμοι που εκτελούνται σε γραμμικό χρόνο λέμε ανήκουν στην κλάση O(N)
- Στην περίπτωση αυτή λέμε ότι οι αλγόριθμοι “είναι O(N)”

Εγγραφές δεδομένων



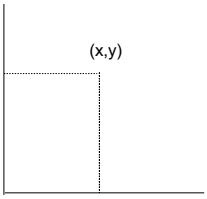
- Τα δεδομένα των προγραμμάτων δεν ανήκουν πάντα σε κάποιον **απλό** τύπο (ακέραιος, πραγματικός, κ.λπ.)
- Αποτελούν ποιο σύνθετες οντότητες που συντίθενται από απλούστερες
- Χρειαζόμαστε τύπους για την αναπαράσταση δεδομένων του πραγματικού κόσμου

- Μια εγγραφή συνίσταται από συστατικά στοιχεία που περιέχουν επιμέρους πληροφορία
- Πεδία ή μέλη
- Παράδειγμα
  - Επώνυμο
  - Όνομα
  - Πατρώνυμο
  - Τηλέφωνο

- Κάθε πεδίο σχετίζεται με έναν τύπο δεδομένων
- Παράδειγμα:
  - Επώνυμο: string
  - Πατρώνυμο: char

Παραδείγματα εγγραφών

- Ένα σημείο στο επίπεδο καθορίζεται από τις συντεταγμένες του, x και y.



Εγγραφές στη C

- Στη C οι εγγραφές αναπαρίστανται ως δομές (structs)
- Για τον χειρισμό δεδομένων εγγραφών
  - *Ορίζουμε* έναν τύπο *δομής*.
  - *Δηλώνουμε* μεταβλητές του τύπου της δομής που ορίσαμε.

Ορισμός τύπου δομής

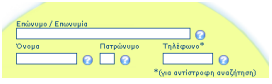
- Με χρήση της typedef

typedef struct {

δηλώσεις_πεδίων

} *όνομα_νέου_τύπου*;

Παράδειγμα




```
typedef struct {
   string lastname;
   string firstname;
   char fathername;
   string phone;
} CatalogEntryT;
```

Δήλωση μεταβλητών δομών και προσπέλαση μελών

```
CatalogEntryT entry;

entry.lastname = "Παπασαλούρος";
entry.firstname = "Ανδρέας";
entry.fathername = 'Φ';
/*Αλφαριθμητικό*/
entry.phone = "2273082136";
```

Δήλωση και αρχικοποίηση μιας μεταβλητής τύπου δομής

CatalogEntryT entry = {“Παπασαλούρος”, “Ανδρέας”, 'Φ', “2273082136”};

Χειρισμός δομών από συναρτήσεις

```
typedef struct {
    double x, y;
} pointT;
```

- Τιμές τύπου δομής επιστρέφονται από συναρτήσεις

```
pointT CreatePoint(double x, double y)
{
   pointT p;
   p.x = x;
   p.y = y;
   return (p);
}
```

Χειρισμός μεταβλητών τύπου δομής

- Οι μεταβλητές τύπου δομής είναι “αριστερές τιμές” δηλαδή επιτρέπεται η ανάθεση τιμών σε αυτές.

```
pointT origin;
origin = CreatePoint(0,0);
```

Δομές ως παράμετροι συναρτήσεων

```
pointT AddPoint(pointT p1, pointT p2)
{
   pointT p;

   p.x = p1.x + p2.x;
   p.y = p1.y + p2.y;

   return (p);
}
```

Δομές ως παράμετροι συναρτήσεων (2)

- Κατά την κλήση μιας συνάρτησης που δέχεται μια εγγραφή (δομή) ως όρισμα, η τιμή της εγγραφής αντιγράφεται στην αντίστοιχη *τυπική* παράμετρο της συνάρτησης.
- Μετά την επιστροφή της συνάρτησης η τιμή της παραμέτρου δεν αλλάζει: Οι εγγραφές περνιούνται ως παράμετροι *με τιμή (by value)*.

Πίνακες και δομές

- Είναι δυνατή η δήλωση ενός πίνακα με στοιχεία τύπου εγγραφής
- Παραδείγματα
- Τηλεφωνικός κατάλογος με 1000 στοιχεία

```
CatalogEntryT catalog[1000];
```

- Παράδειγμα: Μια πολυγωνική γραμμή είναι δυνατόν να αναπαρασταθεί ως ένας πίνακας σημείων
- pointT polygon[10];

Πρόσβαση στα στοιχεία ενός πίνακα εγγραφών

- Το πρώτο στοιχείο ενός πίνακα εγγραφών, π.χ. του πίνακα catalog, γίνεται ως εξής:
- CatalogEntryT entry = catalog[0];
- Το όνομα (lastname) του παραπάνω στοιχείου προσπελάζεται ως
  - entry.lastname
  - ή
  - catalog[0].lastname

Προσπέλαση στα μέλη πολύπλοκων δομών

•Το μέλος (πεδίο) lastname ενός μιας δομής CatalogEntryT είναι τύπου αλφαριθμητικού (πίνακα χαρακτήρων).

•Το στοιχείο catalog[10].lastname[3] είναι ο τέταρτος χαρακτήρας του επωνύμου της ενδέκατης εγγραφής.

Παράδειγμα

- Να γραφεί ένα πρόγραμμα το οποίο δέχεται ως είσοδο ένα ποσό (σε ευρώ) και τυπώνει μια λίστα με τα νομίσματα (χαρτονομίσματα και κέρματα) που αντιστοιχούν σε αυτό το ποσό ώστε ο αριθμός των νομισμάτων να είναι ελάχιστος.
- Λύση.

Άσκηση

Να γίνει ένα πρόγραμμα διαχείρισης μιας λίστας τηλεφώνων. Σε κάθε εγγραφή της λίστας θα καταχωρούνται το επώνυμο, το όνομα και το τηλέφωνο. Θα είναι δυνατά τα παρακάτω:

- Εισαγωγή μιας καταχώρησης
- Αναζήτηση ενός τηλεφώνου με βάση το επώνυμο

Λύση

- http://www.samos.aegean.gr/math/andpapas/courses/pl/presentations/???

Ασκήσεις για επανάληψη

Άσκηση 1

- Άσκηση 4 Κεφ. 11, σελ. 490 του βιβλίου του Roberts
- Η τυπική απόκλιση ενός συνόλου δεδομένων μιας κατανομής, x_1, x_2, ..., x_n δίνεται από τον τύπο

$$\sigma = \sqrt{\frac{\sum_{1}^{n} (\bar{x} - x_i)}{n}}$$

Άσκηση 1 (συνέχ.)

- Να γραφεί μια συνάρτηση `StandardDeviation(array, n)` η οποία να δέχεται έναν πίνακα διμών κινητής υποδιαστολής και το τρέχον μέγεθος αυτού του πίνακα και να επιστρέφει την τυπική απόκλιση της κατανομής των δεδομένων τα οποία περιέχει ο πίνακας.
- Λύση

Άσκηση 2

- Μαγικά τετράγωνα
- Ένα μαγικό τετράγωνο περιέχει αριθμούς έτσι ώστε το άθροισμα κάθε γραμμής και κάθε στήλης να είναι σταθερό.

Παραδείγματα

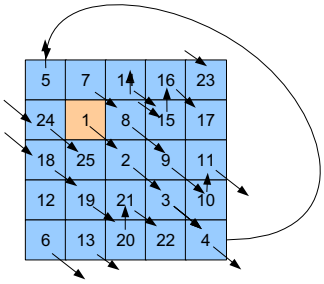
4	9	2
3	5	7
8	1	6

Τετράγωνο τάξης 3

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Τετράγωνο τάξης 4

Δημιουργία μαγικών τετραγώνων περιττής τάξης



Σιαμέζικη μέθοδος ή μέθοδος DeLouber

http://mathworld.wolfram.com/MagicSquare.html

Άσκηση 2

- Να γραφεί πρόγραμμα που δέχεται έναν περιττό ακέραιο από το πληκτρολόγιο και σχηματίζει ένα μαγικό τετράγωνο της τάξης του αριθμού που εισήχθηκε.
- Λύση