Α' Εξάμηνο
ΕΙΣΑΓΩΓΗ ΣΤΟ ΔΟΜΗΜΕΝΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

Εργαστήριο – 6η σειρά ασκήσεων.
Κοζάνη, 16 Νοεμβρίου 2007.

Ας υποθέσουμε ότι θέλουμε να αναπτύξουμε μία εφαρμογή για προβλήματα στα οποία υπεισέρχονται έννοιες του διανυσματικού λογισμού. Θέλουμε να ξεκινήσουμε από ένα είδος "βιβλιοθήκης" πάνω στην οποία θα βασιστούμε ώστε να αναπτύξουμε και μελλοντικά να αναβαθμίζουμε εύκολα την εφαρμογή μας. Για να είναι εύχρηστη αυτή η βιβλιοθήκη, θα πρέπει οι ευκολίες που μας παρέχει να μη μας απασχολούν και πολύ με τις τεχνικές εσωτερικές τους λεπτομέρειες ώστε να μπορούμε να εστιάσουμε την προσοχή μας στην ουσία των προβλημάτων που θα λύσουμε.
Μια βασική έννοια του διανυσματικού λογισμού είναι προφανώς το ίδιο το διάνυσμα. Στη συνέχεια θα δούμε τι είδους βήματα θα μπορούσαμε να κάνουμε προς την υλοποίηση μιας τέτοιας βιβλιοθήκης παίρνοντας ως παράδειγμα το διάνυσμα στις δύο διαστάσεις.

Άσκηση p6-1 (μεταγλώττιση και εκτέλεση)
Δημιουργήστε ένα project, ονομάστε το p6 και δημιουργήστε τα παρακάτω αρχεία που θα ανήκουν σε αυτό:
α) Ένα αρχείο με το όνομα p6main.c και τα ακόλουθα περιεχόμενα (με γαλάζιο οι γραμμές που δίνονται ήδη από το περιβάλλον ανάπτυξης και με κόκκινο αυτές που θα προσθέσουμε):

```
#include <stdio.h>
#include "p6lib.h"

int main(char argc, char *argv[])
{
    point zero, *a;
    zero.x = 0.0;
    zero.y = 0.0;
    printf("Δώσε συντεταγμένες ενός σημείου. \n");
    printf("Πρώτα, το x: \n");
    scanf("%lf", &a->x);
    printf("Μετά, το y: \n");
    scanf("%lf", &a->y);
    printf("Απόσταση από την αρχή: %lf \n", distance(&zero,a));
    system("PAUSE");
    return 0;
}
```

Στο πρόγραμμα αυτό βλέπουμε ότι υπάρχει ένας καινούριος τύπος με το όνομα point για την αποθήκευση δεδομένων που αφορούν σημεία στο επίπεδο. Προφανώς αυτόν τον τύπο τον έχουμε δημιουργήσει εμείς και θα δούμε σε λίγο τον ορισμό του στο αρχείο επικεφαλίδας p6lib.h που έχουμε κάνει include. Επίσης, βλέπουμε μία συνάρτηση με το όνομα distance που παίρνει ως ορίσματα δείκτες σε μεταβλητές τύπου point

(ή ισοδύναμα, διευθύνσεις τέτοιων μεταβλητών). Από αυτή την άποψη και οι δύο μεταβλητές zero και point θα μπορούσαν από την αρχή να δηλωθούν ως δείκτες. Απλώς δηλώσαμε τη μία μόνο ως δείκτη ώστε να δείξουμε τη χρήση και των δύο τελεστών μέλους, της τελείας και του βέλους.
Τι κάνει το πρόγραμμα; Ορίζει ένα σημείο με το όνομα zero και συντεταγμένες 0 (δηλαδή αυτό που λέμε "αρχή των αξόνων") και διαβάζει συντεταγμένες για ένα άλλο σημείο που του δίνουμε. Μετά, τυπώνει την απόσταση του άλλου σημείου από την αρχή των αξόνων.
Προσέξτε στη scanf, ότι αν και το a είναι δείκτης, αυτό δε σημαίνει ότι πρέπει να παραλείψουμε το & διότι, όπως θα δούμε σε λίγο, το μέλος x ή y που παίρνουμε με το βέλος (->) **δεν** είναι δείκτης, άρα χρειάζεται το & (η προτεραιότητα του βέλους είναι πιο υψηλή από του & και έτσι είναι σα να γράψαμε &(a->x)).

β) Ένα αρχείο με το όνομα p6lib.c και τα ακόλουθα περιεχόμενα:

```
#include <math.h>
#include "p6lib.h"

double distance( point *a, point *b) {
    double dx, dy;
    dx = b->x - a->x;
    dy = b->y - a->y;
    return sqrt( dx*dx + dy*dy );
}
```

Σε αυτό το αρχείο ορίζεται η συνάρτηση distance που έχει παραμέτρους δύο δείκτες σε σημεία του επιπέδου (μεταβλητές τύπου point) και επιστρέφει την απόσταση των σημείων. Όπως ξέρουμε, η απόσταση μεταξύ δύο σημείων (x_1, y_1) και (x_2, y_2) είναι $d=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$ Προσέξτε την οδηγία για να συμπεριληφθεί το αρχείο math.h ώστε να μπορεί να κληθεί η συνάρτηση της τετραγωνικής ρίζας (sqrt).

γ) Ένα αρχείο "επικεφαλίδας" με το όνομα p6lib.h και τα ακόλουθα περιεχόμενα:

```
#ifndef P6LIB_H
#define P6LIB_H

struct simio {
    double x, y;
};

typedef struct simio point;

double distance(point * , point * );

#endif
```

Να και ο ορισμός του τύπου point! Στην πραγματικότητα μάλιστα, έχει δηλωθεί μία δομή δεδομένων με το όνομα simio και μετά, για να μην επαναλαμβάνουμε συνέχεια το "struct simio" της δώσαμε το ψευδώνυμο point με την εντολή typedef. Η δομή αυτή έχει δύο μέλη που είναι οι συντεταγμένες του σημείου. Υπάρχουν δύο συντεταγμένες, άρα πρόκειται για σημείο στο επίπεδο.
Εδώ δηλώνεται επίσης και η συνάρτηση distance που ορίζεται στο προηγούμενο αρχείο.

Μεταγλωττίστε και εκτελέστε.

Άσκηση p6-2 (μεταγλώττιση και εκτέλεση)
Στο αρχείο επικεφαλίδας p6lib.h που δημιουργήσατε, εισάγετε άλλη μία δήλωση δομής δεδομένων

και ένα μνημονικό όνομα, όπως φαίνεται παρακάτω (με κόκκινο οι νέες γραμμές που θα προστεθούν):

```
#ifndef P6LIB_H
#define P6LIB_H

struct simio {
    double x, y;
};

typedef struct simio point;

struct dianysma {
    point a, b;
    double length;
    double (*calcLength)();
};

typedef struct dianysma vector;

double distance(point * , point * );

#endif
```

Εδώ βλέπουμε να ορίζεται άλλη μία δομή δεδομένων με όνομα dianysma και μνημονικό όνομα τύπου το vector, η οποία έχει ως μέλη δύο αντίτυπα της προηγούμενης δομής που ορίσαμε, δηλαδή δύο σημεία του επιπέδου και μία double μεταβλητή με το όνομα length. Προφανώς εδώ ορίζουμε έναν τύπο για να περιγράφουμε διανύσματα στο επίπεδο και τα μέλη του είναι τα σημεία που ορίζουν τα άκρα του, καθώς και το μήκος του.
Αλλά υπάρχει και κάτι ακόμη ενδιαφέρον εδώ, και συγκεκριμένα το μέλος calcLength που το όνομά του δείχνει ότι προορίζεται να έχει σχέση με τον υπολογισμό του μήκους και όπως φαίνεται από τη δήλωσή του είναι δείκτης σε συνάρτηση που επιστρέφει double. Δηλαδή η νέα δομή δεδομένων που δημιουργήσαμε (και μαζί και ο νέος τύπος vector) "εξοπλίζεται" μέσω της δυνατότητας να "δείξει" σε μια συνάρτηση, με μία "μέθοδο" υπολογισμού των δεδομένων της.
Προσοχή! Είναι απαραίτητο ο ορισμός της δομής dianysma να μπει μετά από την typdef όπου ορίζεται ο τύπος point αλλιώς ο μεταγλωττιστής δε θα καταλάβει τι είναι αυτά τα μέλη τύπου point που υπάρχουν στη νέα μας δομή και θα βγάλει μήνυμα λάθους.

Στο αρχείο με το κύριο πρόγραμμα πραγματοποιήστε τις παρακάτω αλλαγές (με γαλάζιο ο,τι έμεινε από πριν και με κόκκινο οι νέες γραμμές):

```
#include <stdio.h>
#include "p6lib.h"

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

    v.calcLength = distance;
```

```
    printf("Δώσε συντεταγμένες ενός σημείου. \n");
    printf("Πρώτα, το x: \n");
    scanf("%lf", &v.a.x);
    printf("Μετά, το y: \n");
    scanf("%lf", &v.a.y);

    printf("Δώσε συντεταγμένες ενός σημείου. \n");
    printf("Πρώτα, το x: \n");
    scanf("%lf", &v.b.x);
    printf("Μετά, το y: \n");
    scanf("%lf", &v.b.y);

    printf("Μήκος διανύσματος: %lf \n", v.calcLength(&v.a, &v.b));
}
```

Εδώ ορίζουμε μία μεταβλητή τύπου vector, δίνουμε τις συντεταγμένες των σημείων που το αποτελούν και βάσει αυτών υπολογίζουμε και εμφανίζουμε το μήκος του. Αξίζει να προσέξουμε τα εξής:

- Μετά από τον ορισμό του διανύσματος v ορίζουμε ότι το μέλος του calcLength θα "δείχνει" στη συνάρτηση distance. Έτσι, θα μπορεί να χρησιμεύσει στον υπολογισμό του μήκους του ανύσματος.
- Όταν διαβάζουμε τις συντεταγμένες, ουσιαστικά καλούμε το μέλος του μέλους: η παράσταση v.a.x σημαίνει "η τιμή του x που είναι μέλος του σημείου a που είναι μέλος του διανύσματος v".
- Στο τέλος, κάνουμε χρήση του μέλους calcLength που δείχνει στη distance και περνώντας του ως ορίσματα τις διευθύνσεις από τα σημεία-μέλη του διανύσματος παίρνουμε το μήκος του.

Μεταγλωττίστε και εκτελέστε.

Άσκηση p6-3 (μεταγλώττιση και εκτέλεση)
Με τη βοήθεια του τύπου vector ορίσαμε μία κατηγορία μαθηματικών οντοτήτων (διανύσματα) που χαρακτηρίζεται από μια σειρά από δεδομένα (συντεταγμένες κλπ) τα οποία υπόκεινται σε συγκεκριμένους χειρισμούς. Θα θέλαμε να κάνουμε αυτή την κατηγορία οντοτήτων πιο "συμπαγή". Δηλαδή θα θέλαμε να τυποποιήσουμε τους χειρισμούς των δεδομένων της και συγχρόνως να αποκρύψουμε τις λεπτομέρειες των σχετικών υπολογισμών από το κύριο πρόγραμμα και γενικά την καλούσα συνάρτηση, ώστε να μην απασχολούν τους μελλοντικούς χρήστες όταν χρησιμοποιούν αυτή την κατηγορία.

Πρώτα τροποποιούμε το αρχείο επικεφαλίδας. Εδώ θα κάνουμε μόνο μία προσθήκη προς το τέλος, γι' αυτό και δεν το γράφουμε ολόκληρο (και πάλι, με κόκκινο τα νέα στοιχεία ή αλλαγές):

```
#ifndef P6LIB_H
#define P6LIB_H
......
double distance(point * , point * );
vector newVector( void );

#endif
```

Δηλαδή προσθέσαμε μία ακόμη συνάρτηση που επιστρέφει μία μεταβλητή τύπου vector. Το τι κάνει αυτή θα δούμε στη συνέχεια.

Μετά, τροποποιούμε και το αρχείο p6lib.c όπου ορίζουμε τη νέα μας συνάρτηση κάτω από τα

προηγούμενα περιεχόμενα. Και πάλι αφού πρόκειται απλώς για μια προσθήκη στο τέλος του αρχείου παραλείπουμε τις προηγούμενες γραμμές.

```
......
vector newVector( void ) {
    vector v;
    v.a.x = v.a.y = v.b.x = v.b.y = 0.0;
    v.calcLength = distance;
    return v;
}
```

Η νέα συνάρτηση είναι πολύ απλή: ορίζει μία τοπική μεταβλητή v τύπου vector, κάνει το μέλος-δείκτη-σε-συνάρτηση να δείχνει τη distance και επιστρέφει τη v. Επίσης, δίνεται μια καλή ευκαιρία να προσθέσουμε και άλλες λεπτομέρειες όπως το να αποδώσουμε μηδενικές αρχικές συντεταγμένες στα σημεία του διανύσματος. Στη συνέχεια θα δούμε γιατί το κάναμε αυτό.

Τέλος, τροποποιούμε το κύριο πρόγραμμα. Και εδώ επίσης η αλλαγή είναι μία: αντικαθιστούμε την εντολή

```
    v.calcLength = distance;
```

με την

```
    vector v = newVector();
```

Το πλεονέκτημα μπορεί να φαίνεται αμελητέο αλλά έχουμε κερδίσει κάτι: κατά την ανάπτυξη του κύριου προγράμματος δεν ασχολούμαστε με τεχνικές λεπτομέρειες όπως ποια συνάρτηση αντιστοιχίζεται σε ποιον δείκτη για τον υπολογισμό του μήκους ή πότε και πώς θα αποδοθούν αρχικές τιμές στο διάνυσμα. Αυτό είναι κάτι που γίνεται "εσωτερικά". Επιπλέον αργότερα μπορεί να εμπλουτίσουμε τη δομή struct dianysma με νέα στοιχεία και πιθανά να τροποποιήσουμε τη newVector αναλόγως χωρίς να χρειαστεί να πειράξουμε το ίδιο το πρόγραμμα που θα παραμείνει αναλλοίωτο.

Μεταγλωττίστε και εκτελέστε.

Άσκηση p6-4 (μεταγλώττιση και εκτέλεση)
Στην ίδια λογική της μεταφοράς των τεχνικών λεπτομερειών μπορούμε να μεταφέρουμε και την ανάγνωση των συντεταγμένων σε μία συνάρτηση που να καλείται από το κύριο πρόγραμμα και να κάνει όλη τη σχετική δουλειά.
Πρώτα θα δηλώσουμε αυτή τη συνάρτηση στο αρχείο επικεφαλίδας. Αφού ασχολείται με συντεταγμένες θα μπορούσαμε να της δώσουμε ένα ταιριαστό όνομα όπως setCoordinates. Ωστόσο, προτιμούμε να της δώσουμε ένα πιο γενικό όνομα όπως setVector. Η γενικότητα είναι μια καλή τακτική που παρέχει ευελιξία κατά την αναβάθμιση ενός προγράμματος – εξ άλλου εδώ έχουμε λόγους για αυτή την επιλογή που θα γίνουν σύντομα φανεροί.
Η δήλωση της setVector μπαίνει στο τέλος του p5lib.h (πριν το #endif βέβαια!) και είναι η εξής:

```
void setVector(vector * );
```

Στη συνέχεια τροποποιούμε το αρχείο p6lib.c προσθέτοντας τον ορισμό της νέας συνάρτησης:

```
#include <stdio.h>
#include <math.h>
#include "p6lib.h"

...... (προηγούμενες συναρτήσεις)

void setVector(vector *v) {
```

```
    printf("Δώσε συντεταγμένες ενός σημείου. \n");
    printf("Πρώτα, το x: \n");
    scanf("%lf", &v->a.x);
    printf("Μετά, το y: \n");
    scanf("%lf", &v->a.y);

    printf("Δώσε συντεταγμένες ενός σημείου. \n");
    printf("Πρώτα, το x: \n");
    scanf("%lf", &v->b.x);
    printf("Μετά, το y: \n");
    scanf("%lf", &v->b.y);
}
```

Εδώ παρατηρούμε κατ' αρχήν ότι συμπεριλάβαμε και το αρχείο <stdio.h>. Αυτό είναι λογικό και απαραίτητο αφού σε αυτό το αρχείο μεταφέρουμε της λειτουργίες εκτύπωσης και ανάγνωσης (printf, scanf).
Στη συνέχεια βλέπουμε ότι ο ορισμός της συνάρτησης δεν εξαντλείται σε ένα απλό copy-paste από το κύριο πρόγραμμα. Η παράμετρος της συνάρτησης θα είναι κάποιο διάνυσμα που έχουμε δηλώσει και αρχικοποιήσει με τη newVector. Αυτό θα πρέπει μετά την κλήση της συνάρτησης να έχει αλλαγμένες συντεταγμένες σύμφωνα με την είσοδο που το πρόγραμμα δέχτηκε από το χρήστη. Άρα, το διάνυσμα πρέπει να περνάει *μέσω διεύθυνσης* και όχι μέσω τιμής – αλλιώς οι αλλαγές θα αφορούσαν ένα τοπικό αντίγραφο του διανύσματος και θα χάνονταν χωρίς να περάσουν σε αυτό με το τέλος της συνάρτησης. Γι΄αυτό η παράμετρος εισόδου δηλώνεται ως δείκτης με τη βοήθεια του τελεστή *.
Το πέρασμα μέσω διεύθυνσης έχει συνέπειες και στην εσωτερική δομή της συνάρτησης. Η παράμετρος είναι δείκτης επομένως ο τελεστής μέλους “τελεία” πρέπει να αντικατασταθεί από τον τελεστή μέλους “βέλος”, όπως δείχνουμε με τα έντονα γράμματα, ώστε από το διάνυσμα να πάρουμε τα μέλη-σημεία. Δε χρειάζεται να το κάνουμε και για τα σημεία από τα οποία θα πάρουμε τις συντεταγμένες γιατί αυτά δεν έχουν οριστεί ως μέλη στη δομή struct simio (βλ. και άσκηση 1 στην παρούσα σειρά).

Τέλος, αλλάζουμε το κύριο πρόγραμμα φέρνοντάς το στην παρακάτω μορφή:

```
#include <stdio.h>
#include "p6lib.h"

int main(char argc, char *argv[])
{
    vector v = newVector( );

    setVector( &v );

    printf("Μήκος διανύσματος: %lf \n", v.calcLength(&v.a, &v.b));
}
```

Εξυπηρετώντας τον αρχικό σκοπό μας, της μεταφοράς των τεχνικών λεπτομερειών στο παρασκήνιο, πετύχαμε επιπλέον να φέρουμε το κύριο πρόγραμμα σε μια μορφή πολύ απλούστερη και πιο κατανοητή και ευανάγνωστη από την αρχική.

Μεταγλωττίστε και εκτελέστε.

Άσκηση p6-5 (μεταγλώττιση και εκτέλεση)
Στη μορφή που έχει πάρει το πρόγραμμά μας, η εκτύπωση του μήκους απαιτεί την κλήση της distance (μέσω του calcLength) η οποία υπολογίζει αυτή την ποσότητα. Αυτό δεν είναι καλή

τακτική γιατί αν θέλουμε να χρησιμοποιούμε συχνά το μήκος ενός διανύσματος που παραμένει αμετάβλητο, θα επιβαρύνουμε το πρόγραμμα με τη συχνή επανάληψη των ίδιων υπολογισμών. Θα ήταν καλό να υπολογίσουμε μία φορά το μήκος και να το εμφανίζουμε όποτε χρειαστεί. Για το σκοπό αυτό θα μεταφέρουμε τον υπολογισμό του μήκους μέσα στη setVector (τώρα δικαιώνεται η επιλογή μας να διαλέξουμε ένα γενικό όνομα).
Για την εμφάνιση του μήκους θα μπορούσαμε να χρησιμοποιήσουμε το ίδιο το μέλος v.length του διανύσματος και αυτό θα κάνουμε προς το παρόν.

Τροποποιήστε το αρχείο p6lib.c προσθέτοντας στο τέλος της setVector μία κλήση της calcLength:

```
    v->length = v->calcLength(&v->a, &v->b);
```

Αυτή η προσθήκη δικαιώνει την αρχική μας επιλογή για το όνομα που δώσαμε στη συνάρτηση ανάγνωσης συντεταγμένων. Τώρα, πρόκειται για μια συνάρτηση που "θέτει" όλα τα μέλη του διανύσματος και έτσι της αξίζει ένας πιο γενικός προσδιορισμός.
Και πάλι όπως πριν, το πέρασμα μέσω διεύθυνσης επιβάλλει τη χρήση του τελεστή μέλους "βέλος".

Μεταγλωττίστε και εκτελέστε.

Άσκηση p6-6 (μεταγλώττιση και εκτέλεση)
Το τελευταίο βήμα για την τελειοποίηση είναι να ξεφορτωθούμε και την εμφάνιση του μέλους length στην εκτύπωση του μήκους. Θέλουμε ο μελλοντικός χρήστης ή προγραμματιστής να μην ασχολείται καθόλου με την εσωτερική δομή του αντικειμένου μας (εν προκειμένω διανύσματος) και να "επικοινωνεί" με αυτό μέσω κατάλληλων συναρτήσεων διασύνδεσης. Αυτές επειδή θα επιτελούν τη γενική λειτουργία ανταλλαγής δεδομένων, θα μπορούν να τυποποιηθούν ώστε να έχουν λίγο πολύ πανομοιότυπη δομή και για όλες τις άλλες κατηγορίες δεδομένων στην εφαρμογή μας. Αυτό είναι καλό για το χρήστη και τον προγραμματιστή γιατί θα μπορεί να μάθει πολύ πιο εύκολα και γρήγορα τη χρήση της βιβλιοθήκης που αναπτύσσουμε χωρίς να ασχολείται με τις εσωτερικές λεπτομέρειες που μπορεί να αλλάζουν δραματικά σε κάθε κατηγορία δεδομένων.
Εδώ θα υλοποιήσουμε μια πολύ απλή συνάρτηση που την ονομάζουμε getLength και μόνο σκοπό έχει να επιστρέφει το μήκος του διανύσματος. Για να το κάνει αυτό θα πρέπει να πάρει από κάπου το μήκος του. Υπάρχουν δύο τρόποι για να το πετύχουμε:

- Ορίζουμε την getLength ως αυτοτελή συνάρτηση τύπου double που παίρνει ως όρισμα ένα διάνυσμα ή δείκτη σε αυτό και επιστρέφει το μήκος του. Αυτή είναι η πιο απλή περίπτωση και θα την υλοποιήσουμε εδώ.
- Προσθέτουμε στο διάνυσμα (στη δήλωση της δομής struct dianysma, δηλαδή) ένα δείκτη σε συνάρτηση που τον ονομάζουμε getLength και με δεδομένο ένα διάνυσμα v μπορούμε να γράψουμε v.getLength( ) (ή v->getLength( ), αν πρόκειται για δείκτη σε διάνυσμα).

Για να υλοποιήσουμε την πρώτη εκδοχή:
Στο τέλος του αρχείου p6lib.h δηλώνουμε τη νέα συνάρτηση ως εξής:

```
double getLength(vector * );
```

Στο τέλος του αρχείου p6lib.c ορίζουμε τη συνάρτηση ως εξής:

```
double getLength(vector *v) {
    return(v->length);
}
```

Στο κύριο πρόγραμμα τροποποιούμε την εντολή εκτύπωσης του μήκους ως εξής:

```
  printf("Μήκος διανύσματος : %lf \n", getLength(&v));
```

Αυτή είναι μια απλή και αποτελεσματική λύση, αλλά κάπως υπολείπεται της τελειότητας γιατί μελλοντικά μπορεί να χρειαστεί να αλλάξουμε κάτι σε σχέση με αυτή (το όνομά της, τη λίστα ορισμάτων της) και αν έχουμε να συντηρήσουμε ένα μεγάλο πρόγραμμα θα πρέπει να ανατρέξουμε σε όλα τα σημεία όπου

εμφανίζεται αυτή για να τη διορθώσουμε. Η λύση με το μέλος-δείκτη σε συνάρτηση είναι πιο πολύπλοκη στην υλοποίησή της αλλά υπερτερεί στο θέμα της συντήρησης του λογισμικού επειδή σε περίπτωση τροποποιήσεων δεν έχουμε παρά να αλλάξουμε το πού δείχνει ο δείκτης, κάτι που λογικά θα γίνεται μια φορά στην αρχή του προγράμματος.

Μεταγλωττίστε και εκτελέστε.

Άσκηση p6-7 (ανάπτυξη κώδικα – για το σπίτι!)
Υλοποιήστε τη δεύτερη λύση στο πρόβλημα ανάκτησης του μήκους. Θα πρέπει να μπορούμε να γράψουμε στο κύριο πρόγραμμα μια εντολή της μορφής

```
printf("Μήκος διανύσματος : %lf \n", v.getLength());
```

Η getLength θα μπορούσε να υλοποιηθεί και ως αυτοτελής δείκτης σε συνάρτηση, χωρίς να είναι μέλος του διανύσματος. Είναι όμως προτιμώτερο να υλοποιηθεί ως μέλος του επειδή μπορεί αντικείμενα της εφαρμογής μας άλλης κατηγορίας να έχουν και αυτά την ιδιότητα "μήκος" που θα θέλαμε να ανακτήσουμε (π.χ. ένα "ευθύγραμμο τμήμα" ή ένα "στερεό" με ιδιότητες όπως "μήκος", "πλάτος" και "ύψος"). Τότε, θα έπρεπε να γράψουμε διαφορετικές συναρτήσεις με άλλο όνομα η κάθε μία, ανάλογα με το είδος των αντικειμένου που θα ήταν όρισμα. Αν όμως επιλέξουμε να υλοποιούμε συναρτήσεις-μέλη για το σκοπό αυτό, έχουμε και πάλι βέβαια να αναπτύξουμε μία συνάρτηση για κάθε είδος αντικειμένου, αλλά κερδίζουμε σε ομοιομορφία και άρα ταχύτητα εξοικείωσης τρίτων με το λογισμικό. Πράγματι, με τον τρόπο αυτό, σε κάθε αντικείμενο που έχει "μήκος" μπορούμε να αντιστοιχίσουμε και μία "μέθοδο" εμφάνισής του, δηλαδή μία συνάρτηση που θα έχει πάντα το ίδιο όνομα, π.χ. getLength, χωρίς να υπάρχει πρόβλημα εφόσον είναι μέλος άλλης δομής κάθε φορά.

Άσκηση p6-8 (ανάπτυξη κώδικα – για το σπίτι!)
Γράψτε παρόμοιες συναρτήσεις και μέλη της δομής dianysma για να ορίζεται η τιμή των συνιστωσών του διανύματος και να μπορείτε να τις εμφανίζεται. Όπως ξέρουμε, για ένα διάνυσμα που ορίζεται από τα σημεία (x_1, y_1) και (x_2, y_2) οι συνιστώσες του είναι $x = x_2 - x_1$ και $y = y_2 - y_1$, αντίστοιχα.

Προτεινόμενες απαντήσεις – λύσεις.

coming soon...