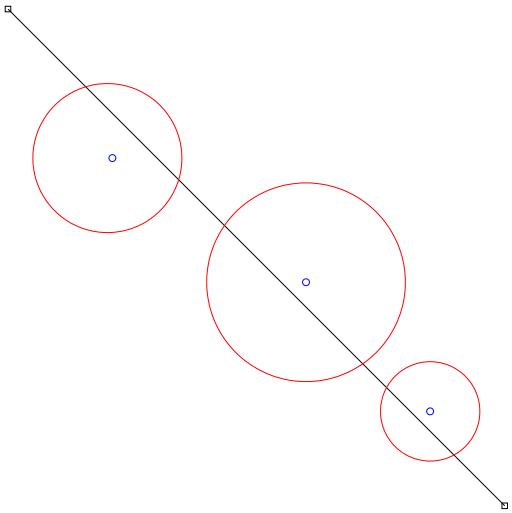
ΤΕΜ-102 Γλώσσα Προγραμματισμού Ηλεκτρονικών Υπολογιστών
3η Εργαστηριακή Άσκηση

Ημερομηνία Παράδοσης **10 Μαΐου 2011, 21:00**

Σκοπός αυτής της εργαστηριακής άσκησης είναι η συγγραφή ενός προγράμματος το οποίο βοηθάει ένα οδηγό να πάει από το σημείο εκκίνησης στον προορισμό του στον ελάχιστο δυνατό χρόνο. Υποθέτουμε ότι ο οδηγός κινείται στο επίπεδο, ξεκινάει από το σημείο με συντεταγμένες (x_s, y_s) και θέλει να φτάσει στο σημείο με συντεταγμένες (x_t, y_t). Η κίνησή του όμως, κατά μήκος της ευθείας L που ενώνει τα δύο σημεία, εμποδίζεται από κυκλικά εμπόδια. Όταν ο οδηγός φτάσει σε ένα από αυτά θα πρέπει να οδηγήσει κατά μήκος του μικρότερου από τα δύο τόξα στα οποία χωρίζει η ευθεία L τον κύκλο και μετά να συνεχίσει την πορεία του πάνω στην ίδια ευθεία (δείτε το παρακάτω σχήμα).



Ονομάστε το πρόγραμμά σας `xxxxea3.c`, όπου `xxxx` είναι ο αριθμός μητρώου σας (σε περίπτωση που αριθμός μητρώου έχει λιγότερα από τέσσερα ψηφία προσθέστε πριν τον αριθμό μητρώου τον κατάλληλο αριθμό μηδενικών), και στείλτε το (ως συνημμένο) με ηλεκτρονικό ταχυδρομείο στη διεύθυνση `tem102lab@gmail.com` το αργότερο μέχρι την Τρίτη 10 Μαΐου 2011, 21:00. Είναι απαραίτητο η πρώτη γραμμή του προγράμματος να περιέχει το όνομά σας και τον αριθμό μητρώου σε σχόλιο, όπως για παράδειγμα,

```
/* Michael Plexousakis, 9999 */
```

Η εξέταση της άσκησης θα γίνει σε ημέρα και ώρα που θα ανακοινωθεί αργότερα. Δείξτε τη δικιά σας δουλειά. Εργασίες που είναι προϊόντα αντιγραφής θα μηδενιστούν. Για τυχόν ερωτήσεις ή διευκρινήσεις για την άσκηση μην διστάσετε να με ρωτήσετε.

Το πρόγραμμα που θα γράψετε θα πρέπει να λειτουργεί ως εξής:

1. Τα δεδομένα του προγράμματος διαβάζονται από το αρχείο `route.dat`. Η πρώτη γραμμή του αρχείου περιέχει τις συντεταγμένες του αρχικού σημείου (δύο `double`'s) και η δεύτερη γραμμή τις συντεταγμένες του τελικού σημείου (δύο `double`'s). Η επόμενη γραμμή περιέχει τον αριθμό των κυκλικών εμποδίων `nobst` (ένας `int`). Οι επόμενες `nobst` γραμμές περιέχουν τις συντεταγμένες του κέντρου κάθε κύκλου και την ακτίνα του (τρείς `double` ανά γραμμή). Υποθέτουμε ότι υπάρχουν το πολύ 20 εμπόδια και ότι:

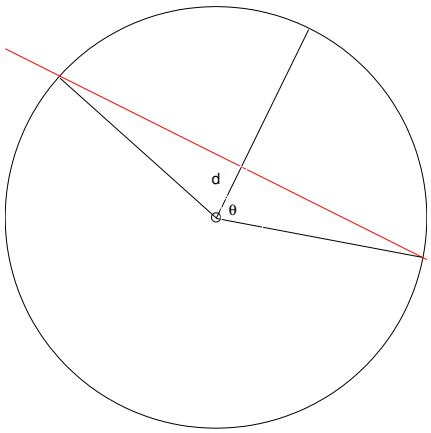
   - $x_s < x_t$, δηλαδή η ευθεία που ενώνει το σημείο εκκίνησης και το σημείο τερματισμού δεν είναι κατακόρυφη και το σημείο εκκίνησης βρίσκεται στ' αριστερά του σημείου τερματισμού
   - τα κυκλικά εμπόδια δεν τέμνονται
   - η x-συντεταγμένη c_x του κέντρου κάθε κύκλου ικανοποιεί $x_s < c_x < x_t$
   - η ευθεία που ενώνει το σημείο εκκίνησης και το σημείο τερματισμού δεν τέμνει αναγκαστικά όλα τα κυκλικά εμπόδια

2. Η έξοδος του προγράμματος αποτελείται από μία και μόνο γραμμή που εκτυπώνεται με την εντολή

   ```
   printf("%f\n", dist);
   ```

   όπου `dist` είναι η απόσταση από το σημείο εκκίνησης στο σημείο τερματισμού που υπολογίζει το πρόγραμμα.

3. Στο πρόγραμμά σας θα χρειαστεί να υπολογίσετε μήκη τόξων και μήκη χορδών. Αν d είναι η απόσταση του κέντρου ενός κυκλικού εμποδίου από τη γραμμή που ενώνει το σημείο εκκίνησης με το σημείο τερματισμού, τότε $\cos\theta = \frac{d}{r}$, όπου r είναι η ακτίνα του κυκλικού εμποδίου. Επομένως, $\theta = \cos^{-1}\frac{d}{r}$. Το μήκος του τόξου είναι τότε $s = r\theta$. Υπενθυμίζουμε



ακόμα ότι μια ευθεία τέμνει ένα κύκλο αν η απόσταση του κέντρου του κύκλου από την ευθεία είναι μικρότερη της ακτίνας του κύκλου.

4. Ίσως σας φανεί χρήσιμο να χρησιμοποιήσετε τις δομές

```
struct point {
        double x;
        double y;
};

struct line {
        double a;   /* Slope */
        double b;   /* y-intercept */
};

struct circle {
        double cx, cy;  /* Center coordinates */
        double r;       /* Circle radius */
};
```

και να υλοποιήσετε, μεταξύ άλλων, τις συναρτήσεις

```
/* Distance between two points */
double PointToPoint(sturct point *p, struct point *q);

 /* Distance between a point and a line */
double PointToLine(struct point *p, struct line *l);
```

5. Μπορείτε να δοκιμάσετε το πρόγραμμά σας με το αρχείο εισόδου που φαίνεται παρακάτω:

```
0.0 10.0
15.0 0.0
4
2.0 7.5 1.75
3.0 3.0 1.0
7.0 5.0 2.0
12.0 1.7 1.5
```