ΕΡΩΤΗΜΑ 5

Στην Ασκηση 5 πρέπει να φτιάξουμε ένα ρομπότ που θα βρίσκει τη λύση ενός λαβύρινθου. Χρησιμοποιούμε μια στοίβα καταστάσεων δηλαδή ένα stack στο οποίο αποθηκεύουμε κάθε φορά τις συντεταγμένες στις οποίες βρισκόμαστε έτσι ώστε αν βρεθούμε σε αδιέξοδο να μπορουμε να οπισθοχωρήσουμε (backtrack). Σημαντικό να σημειωθεί ότι στο πρόγραμμα που έφτιαξα θεωρώ ότι ο χρήστης θα δώσει λαβύρινθο ο οποίος έχει τουλάχιστον ένα μονοπάτι που οδηγεί στη ζητούμενη θέση.

Διαδικαστικά:

Ο χρήστης πρέπει να κάνει ανακατεύθυνση ενα .txt αρχείο μέσα στο οποίο υπάρχει ο λαβύρινθος θεωρώ ως δεδομένο ότι ακολουθεί σωστό format:

- είναι 6*6
- δεν έχει κενά
- χρησιμοποιούνται ΜΟΝΟ οι χαρακτήρες που επιτρέπονται (‘S’ , ‘.’ , ‘#’ , ‘G’’)
    - S = Start
    - . = free spot
    - # = blocked spot
    - G = Goal
        - αργότερα χρησιμοποιείται και το ‘x’ που σημαίνει λανθασμένο μονοπάτι και το ‘+’ που δείχνει σωστό μονοπάτι.

Το directory μου περιέχει 2 παραδείγματα λαβύρινθου (maze.txt & maze2.txt) τα οποια μπορούν να χρησιμοποιηθούν για τον έλεγχο εγκυρότητας του προγράμματος.

maze.c

Είναι το main αρχείο (περιέχει τη main). Χρησιμοποιώ την getchar() για να διαβάσω χαρ/ρα-χαρ/ρα και να τα αποθηκεύσω σωστά τον λαβύρινθό μου σε έναν πίνακα maze[6][6]. Στη συνέχεια καλώ τη συνάρτηση *mazeSOLVE* στην οποία περνάω σαν όρισμα τον πίνακα maze.

mazeMODS.c

Το αρχείο αυτό περιλαμβάνει τις συναρτήσεις με τις οποίες επεξεργάζομαι το stack μου (push / pop / init / empty) και τη συνάρτηση mazeSOLVE που βρίσκει το path για να φτάσουμε στο ‘G’. Ο τρόπος που δουλεύει είναι ο εξής:

Βρίσκουμε που είναι το ‘S’ και ξεκινάμε από εκεί -> Στην συνέχεια ελέγχουμε αν μπορούμε να κινηθούμε προς κάποια κατεύθυνση (με τη σειρά που ελέγχω: Up - Right - Down - Left).

Για κάθε κατεύθυνση ελέγχουμε τα εξής (με τη σειρά) μέχρι να γίνει κάποια κίνηση:

1. είναι εντός ορίων;
2. είναι το ‘G’;
3. είναι ανοιχτό spot (‘.’);

Αν ισχύει το 1 πάμε στο 2. -> Αν ισχύει το 2 τότε βρήκαμε το Goal και τελειώνουμε. Αν οχι παμε στο 3 -> αν είναι ανοιχτή η θέση κινούμαστε προς τα εκεί και αντικαθιστούμε το '.' με το '+'.
Αν δεν ισχυει το 1 τότε ελέγχουμε την επόμενη κατεύθυνση.
Αν δεν ισχύει καμία από τις 3 συνθήκες για καμία από τις 4 κατευθύνσεις τότε σημαίνει ότι βρεθήκαμε σε αδιέξοδο και πρέπει να πάμε προς τα πίσω μέχρις ότου να βρούμε μια θέση που θα εχει free spot (πηγαίνοντας προς τα πίσω τα σημεία που είχαν '+' τα σημειώνουμε με 'x' για να ξέρουμε ότι είναι λάθος path).
Με αυτή τη μέθοδο θα καταλήξουμε στο 'G' που είναι και το ζητούμενο μας.

Τελικά τυπώνουμε το λαβύρινθο ώστε να δείξουμε το μονοπάτι, καθώς και τις τελικές συντεταγμένες του 'G' (αριθμώ το λαβύρινθο από το 1 ως το 6).

maze.h / item.h
Δηλώσεις των συναρτήσεων που χρησιμοποιούνται και typedef το char σε Item.

Το makefile μου έχει τις εξής λειτουργίες:

- make
  - gcc -c maze.c
  - gcc -c mazeMODS.c
  - gcc -o maze maze.o mazeMODS.o
    - Δημιουργεί τα αντικειμενικά (.o) και στη συνέχεια κάνει τη σύνδεση και μας δίνει το εκτελέσιμο (maze)
- make clean
  - rm *.o maze
    - Εκτελούμε αυτή την εντολή σε περίπτωση που θέλουμε να σβήσουμε τα αντικειμενικά (.o) και το εκτελέσιμο (maze).

Για την εκτέλεση του προγράμματος αρκεί να γράψετε:

- make
- ./maze < file.txt

Όπου το file.txt περιέχει το λαβύρινθο.

Σημείωσεις:

1. *Έχουν χρησιμοποιηθεί οι συναρτήσεις που βρίσκονται στο αρχείο STACKimplementation.c του Sedgewick.*
2. *Το πρόγραμμα θα μπορούσα να το σπάσω σε ακόμα περισσότερα αρχεία (πχ να είχα σε ένα αρχείο τη main, σε ενα την fQueue και σε ενα τις συναρτήσεις επεξεργασίας του stack απλά το θεώρησα περιττό σε αυτή τη φαση.*