

Εργασία 3 – Adventure game

Συνοπτική περιγραφή

Σε αυτή την εργασία θα αναπτύξετε ένα παιχνίδι τύπου adventure.

Ο κόσμος του παιχνιδιού είναι ένα σύστημα σπηλαίων που αναπαρίσταται ως γράφος: Κάθε κόμβος του γράφου είναι ένας θάλαμος και κάθε ακμή είναι μια σήραγγα που συνδέει δύο θαλάμους και προς τις δύο κατευθύνσεις.

Ο παίκτης είναι οπλισμένος με τόξο και βέλη και μπορεί να μεταβαίνει από ένα θάλαμο σε έναν άλλο μέσω της σήραγγας που τους ενώνει. Το παιχνίδι τερματίζει με νίκη αν ο παίκτης σκοτώσει με ένα βέλος το δράκο που κρύβεται σε έναν θάλαμο (χωρίς να μετακινείται) ή με ήττα αν επικρατήσει ο δράκος.

Κάποιοι θάλαμοι περιέχουν νάρκες. Είσοδος σε τέτοιο θάλαμο έχει ως αποτέλεσμα (α) να καταστραφεί ο θάλαμος και οι σήραγγες που τον συνδέουν με γειτονικούς θαλάμους (καταστροφή κόμβου και ακμών από/προς γειτονικούς κόμβους) και (β) ο παίκτης να τηλεμεταφερθεί σε ένα άλλο τυχαίο θάλαμο.

Κάποιοι θάλαμοι περιέχουν μαγικές πύλες. Είσοδος σε τέτοιο θάλαμο έχει ως αποτέλεσμα (α) να δημιουργηθεί ένας νέος θάλαμος που συνδέεται με δύο τυχαίους ήδη υπάρχοντες θαλάμους (δημιουργία νέου κόμβου και προσθήκη ακμών ανάμεσα σε αυτόν και δύο υπάρχοντες κόμβους), και (β) να τηλεμεταφερθεί ο παίκτης στον νέο θάλαμο.

Κάποιοι θάλαμοι περιέχουν βέλη (ο παίκτης μπορεί να τα βάλει στη φαρέτρα του), ενώ οι περισσότεροι είναι απλά άδειοι.

Επιπλέον, δίνεται η δυνατότητα αποθήκευσης της κατάστασης ενός παιχνιδιού σε αρχείο καθώς και φόρτωσης μιας σωσμένης κατάστασης από αρχείο.

Ο κώδικας του παιχνιδιού είναι οργανωμένος σε αρκετά αρχεία .c και .h. Μελετήστε τον έτοιμο κώδικα που σας δίνουμε και διαβάστε προσεκτικά τα σχόλια που τον συνοδεύουν.

Στάδια υλοποίησης

Για το στάδιο 1 υλοποιήστε όλες τις λειτουργίες του παιχνιδιού εκτός από τις λειτουργίες αρχείου.

Έτοιμη βιβλιοθήκη [αρχεία libhw3.h, libhw3.a]

Στο αρχείο libhw3.h βρίσκονται:

- Η επικεφαλίδα της συνάρτησης *create_game_graph* η οποία παίρνει ως παράμετρο έναν ακέραιο συντελεστή δυσκολίας, δημιουργεί δυναμικά ένα γράφο το μέγεθος του οποίου προκύπτει από αυτόν τον συντελεστή και επιστρέφει δείκτη προς το γράφο. Η δημιουργία του γράφου γίνεται με τυχαίο τρόπο αλλά με τις παρακάτω εγγυήσεις: (α) δεν περιέχει αποκομμένους κόμβους, δηλαδή υπάρχει πάντα τρόπος άμεσης ή έμμεσης μετάβασης από ένα κόμβο σε οποιοδήποτε άλλο κόμβο, (β) ακριβώς ένας κόμβος περιέχει τον δράκο, (γ) ο κόμβος από όπου ξεκινάει ο παίκτης (που βρίσκεται πάντα στην πρώτη θέση του πίνακα κόμβων) δεν περιέχει ποτέ δράκο.
- Η επικεφαλίδα της συνάρτησης *print_graph* η οποία παίρνει ως παράμετρο δείκτη σε γράφο κι εκτυπώνει τα περιεχόμενα του.

Μη μεταβάλετε το αρχείο libhw3.h.

Οι υλοποιήσεις των συναρτήσεων βρίσκονται στη στατική βιβλιοθήκη libhw3.a.

Μηνύματα εξόδου [αρχείο msg.h]

Στο αρχείο msg.h έχουν οριστεί συμβολικά ονόματα για όλα τα μηνύματα εξόδου της εφαρμογής. Χρησιμοποιήστε τα συμβολικά ονόματα όπου χρειάζεται και όπως περιγράφεται στις ενότητες που ακολουθούν. Μην αλλάξετε αυτά τα μηνύματα γιατί δεν θα λειτουργούν σωστά οι έλεγχοι του autolab.

Δομή και υλοποίηση γράφου [αρχεία graph.h, graph.c]

Στο αρχείο graph.h ορίζονται οι παρακάτω τύποι τους οποίους δεν επιτρέπεται να μεταβάλετε:

- *contents_t*: enum που αναπαριστά τα περιεχόμενα ενός θαλάμου.
- *vertex_t*: struct που αναπαριστά κόμβο του γράφου (θάλαμος). Περιέχει (α) το μοναδικό αναγνωριστικό του κόμβου, (β) ένα πεδίο τύπου *contents_t*, και (γ) δείκτη προς μία λίστα από ακμές (σήραγγες προς άλλους θαλάμους). Η λίστα είναι απλά διασυνδεδεμένη, μη-κυκλική, χωρίς τερματικό.
- *edge_t*: struct που αναπαριστά ακμή του γράφου (σήραγγα). Περιέχει (α) δείκτη προς τον κόμβο που οδηγεί η σήραγγα, και (β) δείκτη προς την επόμενη ακμή της λίστας.
- *graph_t*: struct που αναπαριστά ένα γράφο. Περιέχει (α) δείκτη προς δυναμικά δεσμευμένο πίνακα από δείκτες προς *vertex_t*, (β) το πλήθος κόμβων του γράφου (μέγεθος του πίνακα), και (γ) το μέγιστο αναγνωριστικό κόμβου που έχει δοθεί κατά τη διάρκεια του παιχνιδιού. Καθώς οι σήραγγες είναι διπλής κατεύθυνσης, αν δύο κόμβοι κ1 και κ2 συνδέονται, τότε στη λίστα ακμών του κ1 υπάρχει μία ακμή με δείκτη προς τον κ2 και στη λίστα ακμών του κ2 υπάρχει μία ακμή με δείκτη προς τον κ1.

Ορίστε συναρτήσεις που υλοποιούν τις βασικές λειτουργίες του γράφου. Κατά το ελάχιστο, πρέπει να έχετε:

- Μια συνάρτηση αρχικοποίησης ενός άδειου γράφου (με μηδέν κόμβους).
- Μία συνάρτηση που προσθέτει ένα νέο κόμβο στο γράφο επεκτείνοντας τον πίνακα κατά ένα στοιχείο. Ο νέος κόμβος δε συνδέεται με άλλους κόμβους. Το αναγνωριστικό τίθεται ίσο με 1 για τον πρώτο κόμβο που προστίθεται ή ίσο με το μέγιστο αναγνωριστικό που έχει δοθεί μέχρι εκείνη τη στιγμή συν 1 για επόμενους κόμβους.
- Μία συνάρτηση που συνδέει δύο υπάρχοντες κόμβους μέσω μιας σήραγγας. Αυτό γίνεται προσθέτοντας στη λίστα ακμών του πρώτου μία νέα ακμή η οποία περιέχει δείκτη προς τον δεύτερο κόμβο και, αντίστοιχα, προσθέτοντας στη λίστα ακμών του δεύτερου κόμβου μία νέα ακμή η οποία περιέχει δείκτη προς τον πρώτο κόμβο.
- Μία συνάρτηση που αφαιρεί τη σήραγγα ανάμεσα σε δύο υπάρχοντες κόμβους. Αυτό γίνεται αφαιρώντας από τη λίστα ακμών του πρώτου κόμβου την ακμή που περιέχει δείκτη προς τον δεύτερο κόμβο και, αντίστοιχα, αφαιρώντας από τη λίστα ακμών του δεύτερου κόμβου την ακμή που περιέχει δείκτη προς τον πρώτο.
- Μία συνάρτηση που αφαιρεί ένα κόμβο του γράφου. Αυτό γίνεται διαγράφοντας όλες τις ακμές που συνδέουν αυτόν τον κόμβο με άλλους (και τις ακμές που συνδέουν τους άλλους κόμβους με αυτόν). Ακολούθως, μετακινείται στη θέση του ο κόμβος που βρίσκεται στην τελευταία θέση του πίνακα κόμβων και το μέγεθος του πίνακα μειώνεται κατά ένα στοιχείο. Η τιμή του μέγιστου αναγνωριστικού δε μεταβάλλεται κατά τη διαγραφή.
- Μια συνάρτηση αποδέσμευσης όλης της δυναμικής μνήμης του γράφου.

Προσθέστε επιπλέον συναρτήσεις ανάλογα με τις ανάγκες σας. Για παράδειγμα, συνάρτηση αναζήτησης κόμβου με βάση το αναγνωριστικό, συνάρτηση ελέγχου για το αν δύο κόμβοι συνδέονται με μια ακμή, κτλ.

Παραγωγή ψευδοτυχαίων αριθμών [αρχεία `random.h`, `random.c`]

Κάποιες ενέργειες του παιχνιδιού (π.χ. η επιλογή θαλάμου για τον δράκο) αποφασίζονται με ψευδοτυχαίο τρόπο. Στο αρχείο `random.h` βρίσκονται οι επικεφαλίδες των παρακάτω συναρτήσεων:

- `set_seed`: Αρχικοποιεί τη γεννήτρια ψευδοτυχαίων αριθμών. Η κλήση της συνάρτησης βρίσκεται ήδη στη συνάρτηση `main`. Μην την αλλάξετε και μην την ξανακαλέσετε.
- `get_rand_range`: Παίρνει ως παραμέτρους δύο απρόσημους ακεραίους `from`, `to` και παράγει κι επιστρέφει έναν ψευδοτυχαίο απρόσημο ακέραιο στο διάστημα `[from, to)`.

Οι υλοποιήσεις των παραπάνω συναρτήσεων βρίσκονται στο αρχείο `random.c`. Μην αλλάξετε αυτά τα αρχεία.

Gameplay [αρχεία `game.h`, `game.c`]

Το αρχείο `game.h` περιέχει:

- Τον ορισμό του τύπου `player_t` που αναπαριστά ένα παίκτη. Περιέχει (α) το όνομα του παίκτη, (β) το πλήθος βελών που φέρει πάνω του ο παίκτης, και (γ) δείκτη προς τον κόμβο (θάλαμο) στον οποίο βρίσκεται ο παίκτης.
- Τον ορισμό του τύπου `game_status_t` που αναπαριστά αν το παιχνίδι είναι σε εξέλιξη ή όχι
- Τον ορισμό της σταθεράς `INIT_ARROWS` που αναπαριστά το αρχικό πλήθος βελών που έχει ο παίκτης.
- Τις επικεφαλίδες των συναρτήσεων `main_menu` και `in_game_menu` που εκτυπώνουν τις επιλογές του χρήστη κατά την εκκίνηση και κατά τη διάρκεια του παιχνιδιού αντίστοιχα, διαβάζουν την απάντηση και την επιστρέφουν. Οι υλοποιήσεις αυτών των συναρτήσεων βρίσκονται στο `game.c` και δεν επιτρέπεται να αλλαχθούν.

Ορίστε τις παρακάτω συναρτήσεις που υλοποιούν τη βασική λειτουργία του παιχνιδιού:

- Συνάρτηση αρχικοποίησης στοιχείων παίκτη: Εκτυπώνει το μήνυμα `NAME_REQUEST_MSG`, διαβάζει το όνομα του παίκτη (μπορείτε να υποθέσετε ότι δεν περιέχει κενά και χωράει σε πίνακα μεγέθους `NAME_SIZE`), καλεί εντολή `printf` με πρώτη παράμετρο το μήνυμα `GREETING_MSG` και δεύτερη παράμετρο το όνομα του παίκτη, και αρχικοποιεί το πλήθος βελών που έχει ο παίκτης.
- Συνάρτηση δημιουργίας νέου παιχνιδιού: Καλεί την έτοιμη συνάρτηση `create_game_graph` με παράμετρο τον συντελεστή δυσκολίας και ακολούθως θέτει το θάλαμο που βρίσκεται ο παίκτης να είναι ο πρώτος κόμβος του πίνακα κόμβων (θέση `[0]`).
- Συνάρτηση που παίζει το παιχνίδι, η οποία παίρνει ως παραμέτρους δείκτες προς το γράφο του παιχνιδιού και τον παίκτη κι επιστρέφει `game_status_t`. Η συνάρτηση σε επανάληψη διαχειρίζεται τα περιεχόμενα του θαλάμου στον οποίο βρίσκεται ο παίκτης, δίνει πληροφορίες για τους γειτονικούς θαλάμους και ζητάει από τον παίκτη να επιλέξει αν θέλει να μετακινηθεί ή να ρίξει ένα βέλος σε έναν από τους γειτονικούς θαλάμους. Για την ακρίβεια, σε επανάληψη:
 - Εάν έχει οριστεί η συμβολική σταθερά `DEBUG` καλεί τη συνάρτηση εκτύπωσης του γράφου στην οθόνη.
 - Εκτυπώνει το μήνυμα `ENTER_ROOM_MSG`.
 - Εξετάζει τα περιεχόμενα του θαλάμου στον οποίο βρίσκεται ο παίκτης.
 - Αν είναι `MINE`, εκτυπώνει το μήνυμα `TRIGGERED_MINE_MSG`, διαχειρίζεται την ενεργοποίηση της νάρκης και συνεχίζει στην επόμενη επανάληψη. Όταν ενεργοποιείται μια νάρκη, αφαιρείται από το γράφο ο κόμβος στον οποίο βρίσκεται ο παίκτης (και οι ακμές που τον συνδέουν με γειτονικούς κόμβους), χρησιμοποιείται η συνάρτηση `get_rand_range` για να βρει μια τυχαία θέση στον πίνακα κόμβων και ο παίκτης μεταβαίνει στον κόμβο που βρίσκεται σε αυτή τη θέση.

- Αν είναι *PORTAL*, εκτυπώνει το μήνυμα *TRIGGERED_PORTAL_MSG*, διαχειρίζεται την ενεργοποίηση της πύλης και συνεχίζει στην επόμενη επανάληψη. Όταν ενεργοποιείται μια πύλη, προστίθεται ένας νέος κόμβος στο γράφο (με περιεχόμενα *EMPTY*), χρησιμοποιείται η συνάρτηση *get_rand_range* για να βρεθούν δύο τυχαίες θέσεις στον πίνακα κόμβων (εκτός αυτής του νέου κόμβου και με επανάληψη στην εύρεση της δεύτερης θέσης ώστε να εξασφαλιστεί ότι δε θα είναι ίδια με την πρώτη), προσθέτει ακμές ανάμεσα στο νέο κόμβο και στους κόμβους που βρίσκονται στις θέσεις που βρέθηκαν και τέλος ο παίκτης μεταβαίνει στο νέο κόμβο που δημιουργήθηκε.
- Αν είναι *DRAGON*, εκτυπώνει το μήνυμα *ENCOUNTERED_DRAGON_MSG* και το παιχνίδι τερματίζει με ήττα για τον παίκτη (η συνάρτηση επιστρέφει *GAME_OVER*).
- Αν είναι *ARROW*, τότε υπάρχει ένα βέλος στον θάλαμο. Ο παίκτης το παίρνει (αυξάνεται το πλήθος από τα βέλη που έχει στην φαρέτρα του), εκτυπώνεται το μήνυμα *FOUND_ARROW_MSG* και ο θάλαμος αδειάζει (δεν έχει πλέον άλλα βέλη).
- Αν είναι *EMPTY* δε γίνεται κάτι.
- Εφόσον το παιχνίδι δεν έχει τερματίσει και ο παίκτης δεν έχει πέσει σε *MINE* ή *PORTAL*, εκτυπώνεται το μήνυμα *PASSAGE_INFO_MSG* με δεύτερη παράμετρο το πλήθος γειτονικών θαλάμων και ακολούθως τα αναγνωριστικά των γειτονικών θαλάμων, ταξινομημένα σε αύξουσα σειρά, με ένα κενό πριν κάθε αναγνωριστικό και ένα '\n' μετά το τελευταίο.
- Το πρόγραμμα ελέγχει αν κάποιος γειτονικός θάλαμος περιέχει τον δράκο, κι αν ναι, εκτυπώνει το μήνυμα *DETECTED_DRAGON_MSG*.
- Μετά ελέγχει αν κάποιος γειτονικός θάλαμος περιέχει *PORTAL* κι αν ναι, εκτυπώνει το μήνυμα *DETECTED_PORTAL_MSG*.
- Διαχειρίζεται την επιλογή του παίκτη για την επόμενη κίνηση του. Για την ακρίβεια,
 - Καλεί τη συνάρτηση *in_game_menu*.
 - Αν ο παίκτης επέλεξε *EXIT*, το παιχνίδι διακόπτεται και η συνάρτηση επιστρέφει *IN_PROGRESS*.
 - Αν ο παίκτης επέλεξε *MOVE*, διαβάζει το ακέραιο αναγνωριστικό του θαλάμου προς τον οποίο θέλει να μετακινηθεί. Αν δεν υπάρχει γειτονικός θάλαμος με αυτό το αναγνωριστικό, εκτυπώνει το μήνυμα *NO_PASSAGE_MSG* και το βήμα διαχείρισης της επιλογής του παίκτη επαναλαμβάνεται. Αν υπάρχει, τότε ο παίκτης μετακινείται σε αυτόν τον θάλαμο.
 - Αν ο παίκτης επέλεξε *SHOOT*, διαβάζει το ακέραιο αναγνωριστικό του θαλάμου προς τον οποίο θέλει να ρίξει το βέλος. Αν δεν υπάρχει γειτονικός θάλαμος με αυτό το αναγνωριστικό, εκτυπώνει το μήνυμα *NO_PASSAGE_MSG* και το βήμα διαχείρισης της επιλογής του παίκτη επαναλαμβάνεται. Αν ο παίκτης δεν έχει διαθέσιμα βέλη, εκτυπώνει το μήνυμα *NO_ARROWS_MSG* και το βήμα διαχείρισης της επιλογής του παίκτη επαναλαμβάνεται. Διαφορετικά, αν ο θάλαμος περιέχει τον δράκο, εκτυπώνεται το μήνυμα *SHOT_DRAGON_MSG* και η συνάρτηση επιστρέφει *GAME_OVER*, ενώ αν δεν τον περιέχει, τα βέλη μειώνονται κατά ένα, εκτυπώνεται το μήνυμα *ARROW_MISSED_MSG* με δεύτερη παράμετρο το πλήθος βελών που απομένουν και ο παίκτης παραμένει στον ίδιο θάλαμο.

Λειτουργίες αρχείου [αρχεία fileio.h, fileio.c]

Το αρχείο fileio.h περιέχει τον ορισμό του magic number που θα λειτουργήσει ως μεταπληροφορία με βάση την οποία μπορεί να γίνεται έλεγχος για το κατά πόσο αυτό έχει δημιουργηθεί από το πρόγραμμα σας και έχει σωστή δομή (περισσότερα γι' αυτό στο φροντιστήριο).

Ορίστε συναρτήσεις που υλοποιούν τις βασικές λειτουργίες αποθήκευσης/φόρτωσης:

- Μία συνάρτηση αποθήκευσης που αποθηκεύει σε αρχείο τα στοιχεία ενός παιχνιδιού (στοιχεία γράφου, θέση παίκτη, και διαθέσιμα βέλη).
- Μία συνάρτηση φόρτωσης που διαβάζει από αρχείο τα στοιχεία ενός παιχνιδιού και αρχικοποιεί κατάλληλα τον γράφο και την κατάσταση του παίκτη.
- Μία βοηθητική συνάρτηση (για τη λειτουργία της φόρτωσης) που ελέγχει αν το αρχείο υπάρχει κι έχει σωστή δομή.
- Μια βοηθητική συνάρτηση (για τη λειτουργία της αποθήκευσης) που ελέγχει αν υπάρχει ήδη αρχείο με το ίδιο όνομα (σκεφτείτε κατάλληλο συνδυασμό των options στην open).

Το όνομα του αρχείου στο οποίο αποθηκεύονται τα στοιχεία του παιχνιδιού προκύπτει από το όνομα του παίκτη με κατάληξη ".sav".

Η πρόσβαση στο αρχείο πρέπει να γίνεται χρησιμοποιώντας τις λειτουργίες του συστήματος (όχι της βιβλιοθήκης stdio). Η ανάγνωση και η εγγραφή πρέπει να γίνονται ξεχωριστά για κάθε βασικό αντικείμενο δεδομένων (π.χ. πλήθος κόμβων, αναγνωριστικό κόμβου, κτλ.). Δεν επιτρέπεται (ούτε έχει νόημα) να γράφετε σε αρχείο ή να διαβάζετε από αρχείο τιμές δεικτών στην μνήμη (σκεφτείτε το γιατί).

Πρέπει να ελέγχονται πάντα οι τιμές επιστροφής των συναρτήσεων open/read/write/lseek και σε περίπτωση σφάλματος η συνάρτηση αποθήκευσης ή φόρτωσης να κλείνει το αρχείο και να επιστρέφει άμεσα κατάλληλο κωδικό λάθους.

Μετά την αποθήκευση ενός παιχνιδιού, το αρχείο περιέχει:

- Το *MAGIC_NUMBER* (χωρίς τον χαρακτήρα '\0' στο τέλος),
- Το πλήθος βελών, το αναγνωριστικό του θαλάμου στο οποίο βρίσκεται ο παίκτης, το μέγιστο αναγνωριστικό κόμβου και το πλήθος κόμβων του γράφου. Όλες αυτές οι τιμές αποθηκεύονται ως *unsigned int*.
- Το *MAGIC_NUMBER* (χωρίς τον χαρακτήρα '\0' στο τέλος),
- Μετά, για κάθε κόμβο, με τη σειρά που εμφανίζεται στον πίνακα, αποθηκεύεται το αναγνωριστικό του ως *unsigned int* και ακολούθως τα περιεχόμενα του ως *contents_t*.
- Τέλος, για κάθε κόμβο, με τη σειρά που εμφανίζεται στον πίνακα, αποθηκεύεται το πλήθος γειτονικών του κόμβων (μήκος λίστας ακμών) ως *unsigned int* και ακολούθως τα αναγνωριστικά των γειτονικών του κόμβων ως *unsigned int*, ταξινομημένα σε αύξουσα σειρά.
- Τέλος αποθηκεύεται ξανά το *MAGIC_NUMBER* (χωρίς τον χαρακτήρα '\0' στο τέλος).

Όλα τα παραπάνω δεδομένα πρέπει να αποθηκεύονται στο αρχείο σε δυαδική μορφή (binary).

Συνάρτηση main [αρχεία hw3.c, hw3.h]

Το αρχείο hw3.c περιέχει τη συνάρτηση main, το μεγαλύτερο μέρος της οποίας έχει ήδη υλοποιηθεί για εσάς:

Για την ακρίβεια, έχει υλοποιηθεί η ανάγνωση του συντελεστή δυσκολίας, η αρχικοποίηση της γεννήτριας τυχαίων αριθμών καθώς και η βασική επανάληψη που εκτυπώνει το κύριο μενού του παιχνιδιού και διαβάζει την επιλογή του παίκτη.

Καλέστε τη συνάρτηση αρχικοποίησης του παίκτη πριν την επανάληψη.

Στη συνέχεια, προσθέστε κώδικα σε κάθε μία από τις περιπτώσεις της switch:

- Στην επιλογή *NEW*, αν υπάρχει ήδη παιχνίδι σε εξέλιξη, εκτυπώνει το μήνυμα *CONFIRM_NEW_GAME_MSG*, διαβάζει την απάντηση του παίκτη (μπορείτε να υποθέσετε ότι θα δοθεί είτε 'n' είτε 'y'), κι εφόσον αυτή είναι 'y', διαγράφει την υπάρχουσα κατάσταση του παιχνιδιού. Στη συνέχεια, σε κάθε περίπτωση, καλεί τις απαραίτητες συναρτήσεις για να δημιουργήσει και να παίξει ένα παιχνίδι.
- Στην επιλογή *LOAD*, αν υπάρχει ήδη παιχνίδι σε εξέλιξη, εκτυπώνει το μήνυμα *CONFIRM_NEW_GAME_MSG*, διαβάζει την απάντηση του παίκτη (μπορείτε να υποθέσετε ότι θα δοθεί είτε 'n' είτε 'y'), κι εφόσον αυτή είναι 'y', καλεί τη συνάρτηση ελέγχου της δομής του αρχείου. Τα σφάλματα που ανιχνεύει η συνάρτηση ελέγχου της δομής του αρχείου αντιμετωπίζονται ως εξής:
 - Αν δεν ανοίξει επιτυχώς το αρχείο, εκτυπώνεται το μήνυμα *OPEN_ERROR_MSG*
 - Αν αποτύχει κάποια εντολή read, εκτυπώνεται το μήνυμα *READ_ERROR_MSG*
 - Αν η δομή δεν είναι σωστή (δηλαδή το *MAGIC_NUMBER* δεν βρίσκεται ακριβώς στα σημεία που πρέπει), εκτυπώνεται το μήνυμα *CORRUPTED_ERROR_MSG*

Αν η συνάρτηση ελέγχου επιστρέψει ένδειξη σφάλματος, η φόρτωση αποτυγχάνει. Διαφορετικά, το πρόγραμμα καλεί τη συνάρτηση φόρτωσης, κι αν αυτή είναι επιτυχής τη συνάρτηση που παίζει το παιχνίδι.

Τα σφάλματα που ανιχνεύει η συνάρτησης φόρτωσης αντιμετωπίζονται ως εξής:

- Αν δεν ανοίξει επιτυχώς το αρχείο, εκτυπώνεται το μήνυμα *OPEN_ERROR_MSG*.
- Αν έχει αποτύχει κάποια εντολή read, εκτυπώνεται το μήνυμα *READ_ERROR_MSG*.
- Αν το αρχείο έχει τερματίσει πρόωρα ή εμφανίζονται απρόσμενα δεδομένα (π.χ. αρνητικό πλήθος κόμβων, αναφορά σε κόμβο που δεν υπάρχει, κτλ.) εκτυπώνεται το μήνυμα *CORRUPTED_ERROR_MSG*.
- Σε κάθε μία από τις παραπάνω περιπτώσεις, η φόρτωση αποτυγχάνει και τα στοιχεία του παιχνιδιού (γράφος και θέση παίκτη) επανέρχονται στην κατάσταση που ήταν πριν την επιλογή της λειτουργίας φόρτωσης.
- Στην επιλογή *SAVE*, αν δεν υπάρχει ήδη παιχνίδι σε εξέλιξη, εκτυπώνει το μήνυμα *NO_GAME_IN_PROGRESS_MSG*, διαφορετικά καλεί τη συνάρτηση ελέγχου ύπαρξης του αρχείου. Αν το αρχείο υπάρχει, το πρόγραμμα εκτυπώνει το μήνυμα *FILE_EXISTS_MSG*. Στη συνέχεια διαβάζει από τον παίκτη ένα χαρακτήρα 'y' ή 'n' (μπορείτε να υποθέσετε ότι θα δοθεί σωστός χαρακτήρας) και αν είναι 'y' καλεί τη συνάρτηση αποθήκευσης. Τα σφάλματα αντιμετωπίζονται ως εξής:
 - Αν δεν ανοίξει επιτυχώς το αρχείο, εκτυπώνεται το μήνυμα *OPEN_ERROR_MSG*.
 - Αν έχει αποτύχει κάποια εντολή write, εκτυπώνεται το μήνυμα *WRITE_ERROR_MSG*.
 - Σε κάθε μία από τις παραπάνω περιπτώσεις η αποθήκευση αποτυγχάνει. Αν η αποθήκευση πετύχει, εκτυπώνεται το μήνυμα *SAVE_OK_MSG*.
- Στην επιλογή *CONTINUE*, αν δεν υπάρχει ήδη παιχνίδι σε εξέλιξη, εκτυπώνει το μήνυμα *NO_GAME_IN_PROGRESS_MSG*, διαφορετικά καλεί τη συνάρτηση που παίζει το παιχνίδι.
- Η επιλογή *HELP* έχει ήδη υλοποιηθεί.
- Στην επιλογή *QUIT*, αν υπάρχει ήδη παιχνίδι σε εξέλιξη, εκτυπώνει το μήνυμα *CONFIRM_QUIT_MSG*, διαβάζει την απάντηση του παίκτη (μπορείτε να υποθέσετε ότι θα δοθεί είτε 'n' είτε 'y'), κι αν αυτή είναι 'n', συνεχίζει την επανάληψη. Σε κάθε άλλη περίπτωση τερματίζει.

Προσθέστε επιπλέον βοηθητικές συναρτήσεις αν χρειάζεται (με τις επικεφαλίδες τους στο hw3.h).

Makefile

Μέρος της εργασίας είναι να κατασκευάστε ένα Makefile για την αυτοματοποιημένη διαδικασία μεταγλώττισης της εφαρμογής. Ο πρώτος στόχος (goal) πρέπει να λέγεται play και να έχει ως αποτέλεσμα τη δημιουργία του τελικού εκτελέσιμου (επίσης play) μετά από σύνδεση των επιμέρους object files και χωρίς να είναι ορισμένη η συμβολική σταθερά *DEBUG* κατά τη μεταγλώττιση. Πρέπει να παράγεται ξεχωριστό object file για κάθε .c αρχείο και να υπάρχουν πάντα σωστές εξαρτήσεις. Πρέπει να χρησιμοποιείται τουλάχιστον μία μεταβλητή.

Επιπλέον, το Makefile πρέπει να έχει δύο στόχους test.1 και test.2 που ο καθένας έχει ως αποτέλεσμα την εκτέλεση ενός τεστ ως εξής: Το πρόγραμμα μεταγλωττίζεται (εφόσον χρειάζεται) με ορισμένη τη συμβολική σταθερά *DEBUG* και κατά την εκτέλεση του τεστ X (όπου X ίσο με 1 ή 2) δέχεται το όρισμα X, ανακατευθύνει την είσοδο σε αρχείο με όνομα in.X, ανακατευθύνει την έξοδο σε αρχείο με όνομα myout.X και τέλος εκτελεί την εντολή diff ανάμεσα στα αρχεία myout.X και out.X ανακατευθύνοντας την έξοδο της εντολής diff σε αρχείο με όνομα diff.X. Ένας επιπλέον στόχος tests έχει ως αποτέλεσμα τη δημιουργία των test.1 και test.2.

Απαιτήσεις υλοποίησης

- Απαγορεύεται η χρήση global ή static μεταβλητών¹.
- Απαγορεύεται η χρήση variable length arrays, goto, gets ή fflush.
- Η δυναμικά δεσμευμένη μνήμη πρέπει να αποδεσμεύεται όταν δε χρειάζεται.
- Το πρόγραμμα πρέπει να ακολουθεί τους κανόνες μορφοποίησης, σχολιασμού, κτλ. που έχουν αναρτηθεί στο eclass, καθ' όλη τη διάρκεια της ανάπτυξης του.

Υποβολή κώδικα

Κατασκευάστε ένα αρχείο zip με όνομα hw3.zip που να περιέχει το Makefile και τα αρχεία c και h της εφαρμογής σας. Υποβάλετε το hw3.zip στο autolab του μαθήματος. Ο ελάχιστος βαθμός που πρέπει να πετύχετε είναι 70.

Στάδιο 1: 24/4/2026, 21:00

Ολοκληρωμένη εργασία: Κυριακή 3/5/2026, 21:00

Δεν υπάρχει δυνατότητα καθυστερημένης υποβολής. Το μέγιστο πλήθος υποβολών χωρίς βαθμολογική ποινή είναι 15. Κάθε υποβολή πλέον των 15 έχει ποινή ίση με 5% του βαθμού.

For fun

Αν έχετε ολοκληρώσει την εργασία χωρίς προβλήματα, και εξακολουθείτε να έχετε όρεξη για περισσότερα (κάποια στιγμή που θα έχετε ελεύθερο χρόνο), ρωτήστε μας για το πώς μπορείτε:

1. Να ενσωματώσετε στοιχειώδη κρυπτογράφηση/αποκρυπτογράφηση αρχείου κατά το save/load.
2. Να αποθηκεύσετε στο αρχείο πολλαπλά σωσμένα παιχνίδια με πληροφoρία χρόνου αποθήκευσης ώστε να δίνεται στον παίκτη η δυνατότητα να επιλέξει ποιο παιχνίδι θέλει να φορτώσει.
3. Να υλοποιήσετε έξυπνη σειριοποίηση/αποσειριοποίηση δυαδικών δεδομένων κατά το save/load έτσι ώστε ο κώδικας σας είναι «φορητός» ανάμεσα σε διαφορετικά συστήματα τόσο ως προς το endianness (εξαρτάται από την αρχιτεκτονική) όσο και ως προς την διευθέτηση των περιεχομένων ενός struct στην μνήμη (εξαρτάται από τον compiler).

¹ Εξαιρούνται οι μεταβλητές που χρησιμοποιούνται στα πλαίσια της υλοποίησης της γεννήτριας τυχαίων αριθμών, τον κώδικα της οποίας σε κάθε περίπτωση δεν επιτρέπεται να μεταβάλετε.