ΟΝΤΟΚΕΝΤΡΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΙΙ

ΑΚΑΔΗΜΑΙΚΟ ΕΤΟΣ 2010-2011

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

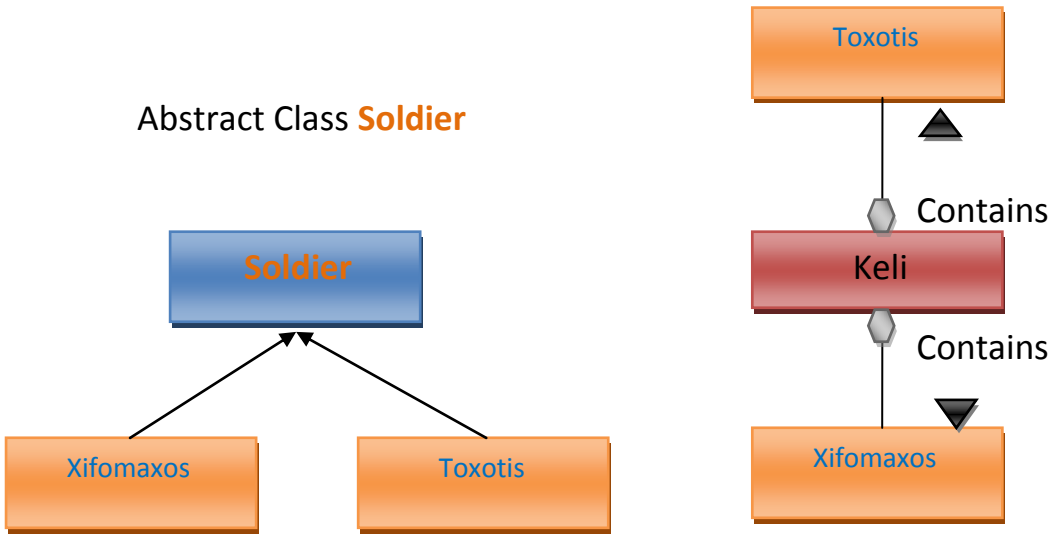
ΟΜΑΔΑ:

ΑΝΤΩΝΙΑΔΗΣ ΠΡΟΚΟΠΗΣ 4935

ΖΙΩΡΗΣ ΡΑΦΑΗΛ 4721

ΜΟΝΟΠΑΤΗΣ ΔΗΜΗΤΡΙΟΣ 4776

Στην παρούσα εργασία υλοποιήσαμε ένα παιχνίδι μονομαχίας. Το διάγραμμα κλάσεων εμφανίζεται παρακάτω ενώ το πρόγραμμα επεξηγείται μετά από αυτό.



Keli

Η κλάση Keli αποτελεί τον χώρο του παιχνιδιού μας, όπου μετακινούνται , μάχονται κτλ. οι στρατιώτες όπως θα δούμε και παρακάτω.
Αποτελείται από τα εξής private χαρακτηριστικά:

- arithmospagidwn : ακέραιος αριθμός που δηλώνει πόσες παγίδες από ένα στρατόπεδο υπάρχουν στο κελί

- pagida : χαρακτήρας που δηλώνει εάν υπάρχει παγίδα τοποθετημένη στο κελί και αν ναι ποιο στρατόπεδο έχει τοποθετήσει παγίδα/παγίδες (A, B, C , D ή e εάν δεν υπάρχει παγίδα)
- kastro : χαρακτήρας που δηλώνει εάν στο κελί είναι τοποθετημένο κάστρο και αν ναι ποιου στρατοπέδου είναι (A, B, C, D ή e εάν δεν υπάρχει κάστρο)
- plhthos[4] : πίνακας ακεραίων τεσσάρων θέσεων που δηλώνει τον αριθμό στρατιωτών που βρίσκονται μέσα σε ένα κελί από κάθε στρατόπεδο. Η θέση 0 αντιστοιχεί στο πλήθος στρατιωτών του στρατοπέδου Α, η 1 στο πλήθος στρατιωτών του στρατοπέδου Β κ.ο.κ.

  Αποτελείται επίσης από το public χαρακτηριστικό Pointers το οποίο είναι τύπου ουράς από δείκτες σε τύπο Soldier. Η χρήση του είναι η εξής: Όταν ένας στρατιώτης είναι διαθέσιμος για μάχη (το ποιες είναι αυτές οι συνθήκες αναφέρεται στην ανάλυση της Main και της συνάρτησης battle παρακάτω) εισέρχεται στην ουρά και αναμένει για μάχη, ενώ όταν πλέον δεν είναι διαθέσιμος εξέρχεται από αυτήν.

Αποτελείται από τις εξής public συναρτήσεις:

- Τον constructor που δημιουργεί κελιά τα οποία για τις ανάγκες του παιχνιδιού ξεκινούν άδεια, χωρίς κάστρα και παγίδες (‘e’) με πλήθος στρατιωτών από κάθε στρατόπεδο και αριθμό παγίδων 0, ενώ στη main βλέπουμε πως μετά τη δημιουργία των στιγμιοτύπων της κλάσης Keli, γίνονται οι τοποθετήσεις των κάστρων, των στρατιωτών, κλπ.
- Τις συναρτήσεις set και get για τα private χαρακτηριστικά arithmospagidwn, pagida, kastro, plhthos[4], που ακολουθούν την κλασική σύνταξη συναρτήσεων set και get της c++ με μοναδική διαφορά τις setplhthos και getplhthos οι οποιες δέχονται ένα παραπάνω όρισμα σε σύγκριση με τις άλλες set και get, το οποίο είναι η θέση του πίνακα plhthos για τον οποίο καλούνται οι συναρτήσεις αυτές.
- Τη συνάρτηση valepagida που δέχεται ως ορίσμα από την main ένα δείκτη τύπου Soldier, τον typeVector[j] της main ο οποίος δείχνει προς τον τρέχοντα στρατιώτη την επιθυμία του οποίου να βάλει παγίδα και επεξεργαζόμαστε. Μέσα στη συνάρτηση αυτή ελέγχουμε αρχικά εάν το κελί στο οποίο βρίσκεται ο στρατιώτης είναι άδειο από παγίδες ή εάν υπάρχει παγίδα (μία ή και περισσότερες) και έχει τοποθετηθεί από μέλος του στρατοπέδου του στρατιώτη. Εάν ισχύει κάποια από αυτές τις συνθήκες, τότε καλείται η συνάρτηση setpagida του κελιού που τοποθετεί στο χαρακτηριστικό παγίδα το αναγνωριστικό του στρατοπέδου του στρατιώτη (typecamp), αυξάνει τον αριθμό των παγίδων (μπορεί να υπήρχαν και άλλες παγίδες του ίδιου στρατοπέδου) κατά ένα και εκτυπώνει σε ποιο κελί

τοποθετήθηκε παγίδα (από τις συντεταγμένες x και y του στρατιώτη) και από ποιο στρατιώτη (typecamp, είδος στρατιώτη (ξιφομάχος ή τοξότης) και αναγνωριστικό id).

- Τη συνάρτηση killpagidakastro που δέχεται σαν όρισμα ένα δείκτη τύπου Soldier, τον typeVector[j] της Main ο οποίος δείχνει προς τον τρέχοντα στρατιώτη που επεξεργαζόμαστε ώστε να δούμε εάν πρέπει να πεθάνει από παγίδα ή κάστρο. Αρχικά ελέγχουμε εάν ο στρατιώτης έχει πέσει σε παγίδα μέσα από την εντολή [if (getarithmospagidwn() > 0 && getpagida() !=vctr->gettypecamp())], δηλαδή ελέγχουμε εάν στο κελί υπάρχουν παγίδες και εφόσον υπάρχουν εάν έχουν τοποθετηθεί από αντίπαλο στρατόπεδο. Εάν οι δύο συνθήκες αυτές ισχύουν, τότε η παγίδα αφαιρείται από το arithmospagidwn (εάν ο arithmospagidwn γίνει 0, το χαρακτηριστικό pagida γίνεται 'e' και πλέον μπορεί οποιοδήποτε στρατόπεδο να τοποθετήσει παγίδα) , ο στρατιώτης πεθαίνει (το life γίνεται false), εκτυπώνεται σε ποιο κελί παγίδα (από τις συντεταγμένες x και y του στρατιώτη) και ποιος στρατιώτης σκοτώθηκε από την παγίδα (typecamp, είδος στρατιώτη (ξιφομάχος ή τοξότης) και αναγνωριστικό id) και τέλος μειώνεται το πλήθος των στρατιωτών του στρατοπέδου του στρατιώτη που πέθανε κατά ένα. Στην περίπτωση που οι συνθήκες δεν ίσχυαν, ελέγχουμε αν στο κελί υπάρχει κάστρο ελέγχοντας αν το kastro είναι διαφορετικό του 'e' και εάν το κάστρο αυτό είναι διαφορετικού στρατοπέδου από τον στρατιώτη. Στην περίπτωση που ισχύουν και οι 2 συνθήκες τότε ο στρατιώτης πεθαίνει (life=false), εκτυπώνεται το γεγονός και μειώνεται το πλήθος των στρατιωτών του στρατοπέδου του στρατιώτη που πέθανε κατά ένα.
- Την συνάρτηση battle που δέχεται σαν όρισμα ένα δείκτη τύπου Soldier, τον typeVector[j] της Main ο οποίος δείχνει προς τον τρέχοντα στρατιώτη που μάχεται με τους στρατιώτες που περιμένουν στην ουρά για να κάνουν μάχη. Αρχικά δηλώνουμε τις ακέραιες μεταβλητές battlepith και k όπου battlepith είναι η μεταβλητή στην οποία δίνουμε τιμές μέσω της συνάρτησης random ώστε να πάρουμε τις πιθανότητες για την μάχη, ενώ η k δηλώνει τα όρια των πιθανοτήτων όπως ορίζονται από την εκφώνηση της άσκησης. Η συνάρτηση αποτελείται από μια επαναληπτική διαδικασία που συνεχίζει μέχρι να σκοτωθούν από τον typeVector[j] όλοι οι στρατιώτες της ουράς ή να πεθάνει ο ίδιος. Μάλιστα στο τέλος εάν ο typeVector[j] νικήσει όλους τους στρατιώτες της ουράς εισέρχεται ο ίδιος στην ουρά. Μέσα στην επανάληψη γίνονται τα εξής: Παίρνουμε μία τυχαία τιμή από 1 έως 100 για την battlepith. Εάν ο typeVector[j] και ο πρώτος που μπήκε στην ουρά, είναι ίδιου τύπου (τοξότες ή ξιφομάχοι) τότε το όριο πιθανότητας k είναι το 50. Αν προκύψει battlepith<50 τότε πεθαίνει ο στρατιώτης της ουράς , εκτυπώνεται το ποιος σκότωσε ποιον και σε ποιο κελί συνέβη η μάχη ,

μειώνεται το πλήθος των στρατιωτών του στρατοπέδου του στρατιώτη της ουράς κατά ένα και τέλος ο στρατιώτης αυτός αφαιρείται από την ουρά, αλλιώς πεθαίνει ο typeVector[j] και έχουμε αντίστοιχα και τα υπόλοιπα (εκτός από την αφαίρεση από την ουρά). Αντίστοιχα και σύμφωνα με την εκφώνηση της άσκησης γίνονται οι μάχες όταν οι αντίπαλοι στρατιώτες είναι διαφορετικού τύπου.

- Τη συνάρτηση printinfo που παίρνει σαν ορίσματα τη μεταβλητή χαρακτήρα gramma που μας χρησιμεύει σαν μετρητής για την επανάληψη στο εσωτερικό της συνάρτησης και το Sigma (S), το πλήθος των στρατοπέδων ,και εκτυπώνει πόσοι στρατιώτες βρίσκονται στο κελί από κάθε στρατόπεδο καθώς και τον αριθμό παγίδων, από ποιο στρατόπεδο έχουν τοποθετηθεί, και το αν υπάρχει κάστρο.

Soldier

Η κλάση Soldier αποτελεί την κλάση βάση για τους παίκτες του παιχνιδιού μας. Είναι μία αφηρημένη κλάση από την οποία κληρονομούν οι κλάσεις Toxotis και Xifomaxos. Αποτελείται μόνο από virtual συναρτήσεις καθώς και από μία pure virtual συνάρτηση που κάνει την κλάση Soldier αφηρημένη.
Αποτελείται από τα εξής private χαρακτηριστικά:

- life : Boolean μεταβλητή που δηλώνει το εάν ένας στρατιώτης είναι ζωντανός ή όχι
- typecamp : μεταβλητή χαρακτήρα που δηλώνει σε ποιο στρατόπεδο ανήκει ο στρατιώτης (A,B,C,D)
- x,y : ακέραιες μεταβλητές-συντεταγμένες των στρατιωτών πάνω στον κόσμο του παιχνιδιού, μας δείχνουν σε ποιο κελί βρίσκονται
- id : ακέραια μεταβλητή που δίνει τη μοναδική ταυτότητα κάθε στρατιώτη ανά στρατόπεδο και απεικονίζεται με τη μορφη xx δηλαδή 00, 01 ,02. Ένας στρατιώτης μπορεί να έχει ίδιο id με κάποιον από άλλο στρατόπεδο.

Αποτελείται από τις εξής public συναρτήσεις:

- Τις virtual συναρτήσεις set και get για τα private χαρακτηριστικά life, typecamp, x, y, id, που ακολουθούν την κλασική σύνταξη συναρτήσεων set και get της c++.
- Τη virtual συνάρτηση nextmove που δέχεται τρία ακέραια ορίσματα, τις 2 συντεταγμένες του στρατιώτη (a και b) και το N. Σε αυτή τη συνάρτηση ο στρατιώτης εκτελεί κίνηση. Αρχικά γίνεται η εκτύπωση της προηγούμενης θέσης και στη συνέχεια παίρνουμε μία τυχαία τιμή για την μεταβλητή epomenikinisi η οποία θα καθορίσει προς ποια κατεύθυνση θα κινηθεί σύμφωνα με την εκφώνηση της άσκησης. Ταυτόχρονα ελέγχεται αν ο στρατιώτης με την κίνησή του θα υπερβεί τα όρια του NxN κόσμου και σε

περίπτωση που γίνει αυτό αλλάζει την κίνησή του και οδηγείται προς την αντίθετη πλευρά. Τέλος θέτουμε τα a και b ως τις νέες συντεταγμένες του στρατιώτη και εκτυπώνουμε τη νέα θέση του.

- Τη pure virtual συνάρτηση getType που μας επιστρέφει το είδος του στρατιώτη (τοξότης ή ξιφομάχος). Στην περίπτωση του Soldier που είναι κλάση βάση και δεν χαρακτηρίζεται από κάποιο είδος, η συνάρτηση δεν ορίζεται άρα είναι ίση με το 0.
- Δεν έχει constructor καθώς είναι αφηρημένη κλάση και δεν δημιουργούνται στιγμιότυπά της.

Toxotis-Xifomaxos

Η κλάσεις Toxotis και Xifomaxos κληρονομούν όλα τα χαρακτηριστικά και τις συναρτήσεις από την αφηρημένη κλάση βάση Soldier. Αποτελούνται από τον constructor που δημιουργεί στρατιώτες τύπου τοξότη/ξιφομάχου αντίστοιχα και τους αρχικοποιεί τη ζωή-life ως true. Τέλος γίνεται το override της pure virtual συνάρτησης της κλάσης Soldier getType ώστε για τους τοξότες να επιστρέφει τον χαρακτήρα 'T' και για τους ξιφομάχους τον χαρακτήρα 'X'.

Main

Στο Main.cpp υπάρχει η υλοποίηση 2 συναρτήσεων και της main.

- Η πρώτη συνάρτηση είναι η συνάρτηση Termination_Error που δεν παίρνει ορίσματα και επιστρέφει την τιμή EXIT_SUCCESS. Η συνάρτηση αυτή καλείται από την main όταν ο χρήστης έχει τοποθετήσει στο αρχείο τιμές στις μεταβλητές που ξεπερνούν τα επιτρεπτά από την εκφώνηση της άσκησης όρια και είναι υπεύθυνη για να εκτυπώσει στο χρήστη ένα ανάλογο μήνυμα και να τερματίσει το πρόγραμμα ώστε να αλλάξει ο χρήστης τις τιμές του αρχείου.
- Η δεύτερη συνάρτηση, η συνάρτηση printvaluesfiles δεν παίρνει ορίσματα και είναι υπεύθυνη να ανοίξει το αρχείο και να επιστρέψει στη main ένα δείκτη που να δείχνει προς τη πρώτο χαρακτήρα του κειμένου εσωτερικά του αρχείου. Το αρχείο τύπου .txt πρέπει να είναι στον ίδιο φάκελο με το εκτελέσιμο. Το περιεχόμενο του αρχείου, πρέπει να έχει αυστηρά την εξής μορφή:N=xx S=x S1=xx S2=xx S3=xx S4=xx ,όπου N το μέγεθος του δισδιάστατου κόσμου NxN, S το πλήθος των αντίπαλων στρατοπέδων, S1, S2, S3, S4 το αρχικό πλήθος των στρατιωτών. Ορίζουμε δύο string των 31 θέσεων , το ένα για να δώσει ο χρήστης το όνομα του αρχείου (stra) και το άλλο για να αποθηκευτούν σε αυτό τα περιεχόμενα του αρχείου (c). Το νούμερο 31 επιλέχτηκε για το πρώτο string αυθαίρετα, ενώ για το δεύτερο είναι σύμφωνα με τη σύμβασή μας για το περιεχόμενο του αρχείου το μέγεθος του περιεχομένου του. Επίσης ορίζουμε ένα δείκτη χαρακτήρων

τον οποίο ορίζουμε να δείχνει στην πρώτη θέση του string c. Στη συνέχεια χρησιμοποιούμε την εντολή fflush(stdin) για να ελευθερώσουμε ότι μπορεί να υπάρχει στον buffer και ενδέχεται να μας δημιουργήσει προβλήματα στη συνέχεια και ζητάμε από τον χρήστη να πληκτρολογήσει το όνομα του αρχείου, το οποίο αποθηκεύουμε στο string stra. Στη συνέχεια μέσω της εντολής myfile.open (stra) προσπαθούμε να ανοίξουμε το αρχείο. Άμα το όνομα του αρχείου δόθηκε σωστά τότε το αρχείο ανοίγει και μέσω της εντολής for (i=0; i<32; i++) c[i] = myfile.get() δίνουμε στο string c το περιεχόμενο του αρχείου και στη συνέχεια το εκτυπώνουμε και το κλείνουμε. Σε περίπτωση που δεν βρέθηκε το αρχείο εκτυπώνεται μήνυμα λάθους. Τέλος επιστρέφουμε στη main τον δείκτη προς το πρώτο στοιχείο του string c.

- **<u>main</u>**
  Αρχικά, μετά την εκτύπωση του καλωσορίσματος, αναθέτουμε στο δείκτη χαρακτήρων str την τιμή που επιστρέφει η printvaluesfiles, δηλαδή την διεύθυνση του πρώτου χαρακτήρα του περιεχομένου του αρχείου. Στη συνέχεια επαναλαμβάνουμε να καλούμε την printvaluesfiles όσο ο χρήστης δεν δίνει αρχείο σύμφωνα με τη σύμβαση (πρώτος χαρακτήρας το N). Μετά αναθέτουμε τις τιμές του αρχείου στα N, Sigma (το S που αναφέρουμε παραπάνω), S1,S2,S3,S4 μέσω του δείκτη str προς το περιεχόμενο του αρχείου (αφού σύμφωνα με τη σύμβαση ξέρουμε σε ποιο σημείο του κειμένου βρίσκεται κάθε στοιχείο). Επίσης ορίζουμε ως h το πλήθος όλων των στρατιωτών και βάζουμε το πλήθος κάθε στρατοπέδου στον πίνακα S[4]. Εάν ο χρήστης δήλωσε το Sigma ίσο με 3 τότε αυτόματα μετατρέπουμε το πλήθος των στρατιωτών του 4ου στρατοπέδου σε 0 και αντίστοιχα αν Sigma=2 τα S3 και S4 0. Στην περίπτωση που ο χρήστης έδωσε Sigma >4 ή <2 τότε καλούμε την Termination_Error καθώς το πρόγραμμα είναι φτιαγμένο για Sigma 2,3 και 4. Το ίδιο κάνουμε αν κάποιο από τα S1,S2,S3,S4 είναι αρνητικό ή μεγαλύτερο του 20. Δημιουργούμε τον πίνακα s[5]= { 0, S1, S1+S2, S1+S2+S3, S1+S2+S3+S4} που θα μας χρειαστεί στη συνέχεια. Στη συνέχεια δημιουργούμε τον κόσμο καλώντας τον constructor της κλάσης Keli για τον δισδιάστατο πίνακα στιγμιοτύπων kosmos[N][N], τον constructor της κλάσης Toxotis για τον πίνακα στιγμιοτύπων toxotes[h/2] (οι μισοί θα είναι τοξότες και οι άλλοι μισοί ξιφομάχοι, και στην περίπτωση περιττού αθροίσματος, οι ξιφομάχοι θα είναι περισσότεροι κατά έναν) και τον constructor της κλάσης Xifomaxos για τη δημιουργία του πίνακα στιγμιοτύπων xifomaxoi[h-h/2]. Επίσης δημιουργούμε ένα vector 2 δεικτών σε αντικείμενα τύπου Soldier και αναθέτουμε τον ένα στην αρχή του πίνακα toxotes και τον άλλο στην αρχή του πίνακα xifomaxoi. Χρησιμοποιούμε την εντολή srand (time (0)) για να έχουμε σε κάθε εκτέλεση του προγράμματος

διαφορετικά αποτελέσματα στην εντολή random. Στη συνέχεια τοποθετούμε τα κάστρα σε τυχαίες συντεταγμένες στα όρια του κόσμου ελέγχοντας κάθε φορά αν οι συντεταγμένες έχουν ήδη κάστρο. Στη συνέχεια στο κέλι προσθέτουμε τους στρατιώτες, και τους αναθέτουμε typecamp, id τις συντεταγμένες τους και τους προσθέτουμε όλους στην ουρά του αντίστοιχου κελιού τους. Στη συνέχεια καλούμε την printinfo για όλα τα κελιά του κόσμου μας. Αμέσως μετά ξεκινάει ουσιαστικά το παιχνίδι. Έχουμε ορίσει μία λογική μεταβλητή z την οποία έχουμε αρχικοποιήσει ως αληθή. Η μεταβλητή αυτή ορίζει το αν έχει τελειώσει το παιχνίδι (αν έχει βγει νικητής) και λεπτομέρειες για το πώς αλλάζει αναφέρονται παρακάτω. Όσο λοιπόν η μεταβλητή αυτή είναι αληθής το παιχνίδι συνεχίζεται και η κάθε κίνηση εκτελείται μόνο αν ο χρήστης πατήσει το πλήκτρο enter. Επαναφέρουμε τα typeVector ώστε να δείχνουν πάλι στις πρώτες θέσεις των πινάκων αντίστοιχα. Ορίζουμε το h ίσο με (S1+S2+S3+S4)/2 ώστε το πρώτο σετ επαναλήψεων να γίνει για όλους τους τοξότες. Η εξωτερική επανάληψη με μετρητή το j μας δίνει 2 σετ επαναλήψεων ένα για κάθε Vector, δηλαδή στην πρώτη επανάληψη έχουμε τις διαδικασίες για τους τοξότες και την δεύτερη για τους ξιφομάχους αφού στο τέλος της πρώτης το h γίνεται ίσο με (S1+S2+S3+S4) - (S1+S2+S3+S4)/2 ώστε να καλύπτει όλους τους ξιφομάχους στην περίπτωση που το πλήθος όλων των στρατιωτών είναι περιττό. Στη συνέχεια για να εκτελέσουμε όλες τις διαδικασίες του παιχνιδιού ελέγχουμε πρώτα αν ο στρατιώτης με τον οποίο ασχολούμαστε είναι ζωντανός και το πλήθος των στρατιωτών του στρατοπέδου του στο κελί που βρίσκεται είναι >0. Μόνο αν ισχύουν και οι δύο συνθήκες εκτελούνται οι διαδικασίες του παιχνιδιού για τον στρατιώτη αυτό. Τοποθετούμε στην ακέραια μεταβλητή movepith μία τυχαία τιμή από 1 έως 100 και αν είναι μικρότερη του 80, ο στρατιώτης κάνει κίνηση, αλλιώς βάζει παγίδα. Στην πρώτη περίπτωση μειώνουμε κατά 1 το πλήθος των στρατιωτών του στρατοπέδου του στο κελί που βρίσκεται και αν το μέγεθος της ουράς του κελιού που βρίσκεται είναι μεγαλύτερο του 0 τότε αφαιρείται από αυτήν και καλείται για αυτόν η συνάρτηση nextmove. Τέλος, αυξάνεται κατά ένα το πλήθος των στρατιωτών του στρατοπέδου του στο νέο κελί που μεταφέρθηκε και καλείται η συνάρτηση killpagidakastro του κελιού για να ελέγξει αν ο στρατιώτης μεταφέρθηκε σε κελί που υπάρχει παγίδα ή κάστρο άλλου στρατοπέδου. Εάν επέζησε μετά το κάλεσμα της συνάρτησης, τότε ελέγχουμε αν πληροί τα κριτήρια για να μπει στην ουρά του κελιού του. Στην ουρά ενός κελιού μπαίνουν στρατιώτες ενός μόνο στρατοπέδου και αναμένουν μέχρι κάποιος αντίπαλου στρατοπέδου εισέλθει στο κελί ώστε να κάνουν μάχη μαζί του. Όταν η ουρά ενός κελιού είναι άδεια, μπορεί να την καλύψει στρατιώτης οποιουδήποτε στρατοπέδου και στη συνέχεια θα

μπουν στρατιώτες του ίδιου στρατοπέδου με αυτόν που εισήχθη πρώτος. Άρα εάν το μέγεθος της ουράς του κελιού στο οποίο βρίσκεται ο στρατιώτης είναι μεγαλύτερο του μηδενός, δηλαδή υπάρχουν άτομα μέσα στην ουρά και αν αυτά είναι διαφορετικού στρατοπέδου από αυτόν, αυτό συνεπάγεται μάχη του στρατιώτη με τον πρώτο της ουράς (καλείται η battle για τον typeVector[j]). Σε περίπτωση που κάποια από αυτές τις συνθήκες δεν πληρείται τότε ο typeVector[j] τοποθετείται στην ουρά (Pointers.push(typeVector[j])). Αυτά γίνονται εάν ο στρατιώτης κάνει κίνηση. Εάν θέλει να τοποθετήσει παγίδα, καλείται η valepagida της κλάσης Keli με όρισμα το typeVector[j]. Μετά ο στρατιώτης αυτός αφαιρείται από την ουρά και ξαναμπαίνει στο πίσω μέρος της γιατί παραμένει στο κελί και υποψήφιος κάθε φορά για να βγει από τη ουρά είναι ο πρώτος. Εάν λοιπόν δεν τον ξανατοποθετήσουμε στο πίσω μέρος της ουράς θα είναι ο επόμενος που θα βγει. Στη συνέχεια για να υπολογίσουμε πόσοι έχουν πεθάνει σε αυτό το γύρο και να υπολογίσουμε αν υπάρχει νικητής ξανααρχικοποιούμε το h σε (S1+S2+S3+S4)/2 και επαναφέρουμε τα typeVector ώστε να δείχνουν στις πρώτες θέσεις των πινάκων που τους αντιστοιχούν. Για να υπάρξει νικητής πρέπει, μετά την κίνηση όλων, να έχουν μείνει στρατιώτες μόνο από ένα στρατόπεδο. Έτσι χρησιμοποιούμε τον πίνακα ακεραίων S[4] ως μετρητή για το πλήθος των ζωντανών στρατιωτών του κάθε στρατοπέδου, με S[0] του A , S[1] του B κ.ο.κ.. Μία επαναληπτική διαδικασία ελέγχει αν οι στρατιώτες είναι πεθαμένοι και αν είναι, μειώνει τον αντίστοιχο μετρητή S. Στην συνέχεια ελέγχοντας το άθροισμα S[0]+S[1]+S[2]+S[3] (εφόσον είναι πάντα μη αρνητικά τα S) συμπεραίνουμε αν νίκησε κάποιο στρατόπεδο. Συγκεκριμένα, αν το άθροισμα είναι ίσο με 0 υπάρχει ισοπαλία διότι πέθαναν όλοι. Ενώ, αν το S[i] είναι ίσο με το άθροισμα σημαίνει ότι ο 'A'+i νικάει αντίστοιχα διότι τα άλλα αθροίσματα είναι 0. Τέλος γίνεται η εκτύπωση των στοιχείων κάθε κελιού από την printinfo. Όλη η διαδικασία που συμπεριλαμβάνεται μέσα στην while εκτελείται επαναληπτικά και αφού η εκτέλεση της while έχει τελειώσει, εφόσον υπάρχει νικητής ή ισοπαλία, γίνεται η εκτύπωση του μηνύματος για νικητή ή ισοπαλία αντίστοιχα και το πρόγραμμα τελειώνει.