**Αθήνα 16/7/2019**

**ΣΚΑΚΙ**

**Βασίλειος Τσακίρης**

**151082**

**ΠΣ:ΤΕΙ ΑΘΗΝΑΣ Ν2**

[cs151082@uniwa.gr](mailto:cs151082@uniwa.gr)

**1.Εισαγωγή**

Το σκάκι είναι ένα επιτραπέζιο παιχνίδι που θεωρείται ότι ανακαλύφθηκε στην Ινδία τον 6ο αιώνα. Πρόκειται για ένα στρατηγικό επιτραπέζιο παιχνίδι όπου δύο παίκτες ελέγχουν το δικό τους σύνολο δεκαέξι τεμαχίων. Το πιο σημαντικό κομμάτι που κάθε παίκτης κατέχει είναι ο Βασιλιάς.Χρησιμοποιώντας τη λογική οι παίκτες το παίρνουν με τη σειρά τους για να μετακινήσουν ένα κομμάτι καθώς προσπαθούν να ξεπεράσουν τον εαυτό τους και να ελέγξουν τον βασιλιά του αντιπάλου.

Σκοπός αυτής της εφαρμογής είναι να δημιουργηθεί ένα λειτουργικό στοιχειώδες παιχνίδι σκακιού δύο παικτών που χρησιμοποιώντας τις λειτουργίες που είναι διαθέσιμες στην C ++. Αυτή η αναφορά περιγράφει εν συντομία το το σχεδιασμό και την εφαρμογή των αντικειμένων που χρησιμοποιήθηκαν για την υλοποιήση του παιχνιδιού.

**2. Σχεδίαση κώδικα και υλοποίηση**

Η προσέγγιση που ακολουθήθηκε για την ανάπτυξη του σκακίου χωρίζεται στα παρακάτω στάδια:

* Δημιουργία του παιχνιδιού μέσω της συνάρτηση main()
* Δημιουργία του ταμπλό και εμφάνισή του στην κονσόλα
* Έλεγχος ορθότητας των συντεταγμένων που δίνει ο χρήστης
* Υποβολή κίνησης και έλεγχος ορθότητας κίνησης ανάλογα με το πιόνι
* Εκκίνηση της λογικής του παιχνιδιού
* Ενημέρωση του ταμπλό άναλογα με την νομιμότα των κινήσεων ανά πιόνι
* Υλοποίηση τερματικής κατάστασης του παιχνιδιού (checkmate,stalemate)

**2.1 Δημιουργία του παιχνιδιού μέσω της συνάρτηση main()**

To παιχνίδι ξεκινάει από την main συνάρτηση η οποία δημιουργεί ένα singleton instance της κλασης Game.Αφού δημιουργηθεί το προαναφερθέν instance της κλάσης Game καλείται η μέθοδος run η οποία εμφανίζει το banner της εφαρμογής και το menu με χρήσιμες οδήγιες σχετικά με την διεξαγωγή του παιχνιδιού καθώς και διάφορα χαρακτηριστικά του όπως εμφάνιση των κινήσεων (εάν υπάρχουν).

|  |  |
| --- | --- |
| Μέθοδος | Περιγραφή |
| void Game::run() | Δημιουργεί ένα instance της κλάσης Board και στην συνέχεια εμφανίζει το ταμπλό στην κονσόλα καλώντας την printBoard().Ελέγχει την νομιμότητα της εισόδου από τον χρήστη και παρέχει την δυνατότητα επανεμφάνισης των κινήσεων του παιχνιδιού αν υπάρχουν.Η ορθότητα της εισόδου δηλαδή των συντεταγμένων που δίνει ο χρήστης πραγματοποιείται με regular expression το οποίο υλοποιήθηκε με την βοήθεια της βιβλιοθήκης <regex>. |

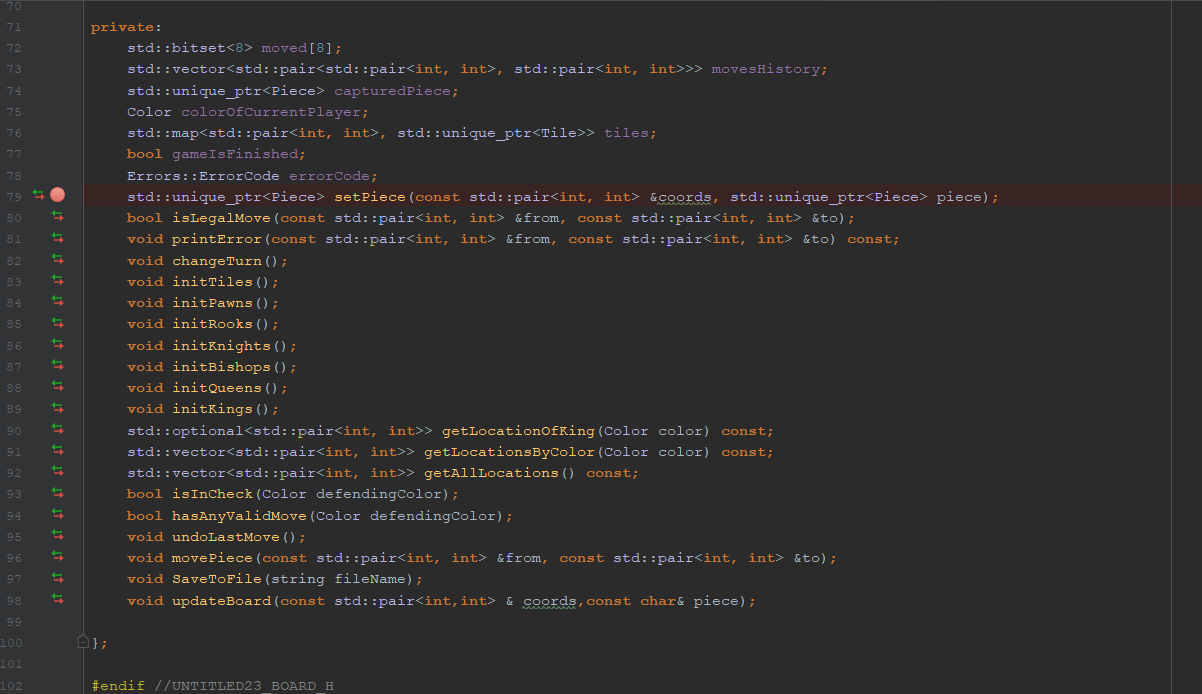
**2.2 Δημιουργία του ταμπλό και εμφάνισή του στην κονσόλα**

Μια σκακιέρα (board) αποτελείται από ένα πλέγμα 8x8 τετραγώνων (tiles or squares) που περιέχει 16 μαύρα κομμάτια τοποθετημένα στη μία πλευρά του πλέγματος και 16 λευκά κομμάτια στην άλλη. Τα θεμελιώδη συστατικά του συγκεκριμένου παιχνιδιού είναι οι κλάσεις Piece και Tiles καθώς και οι συντεταγμένες στις οποιές βρίσκεται το κάθε πιόνι.Η κάθε συντεταγμένη αποθηκέυεται σε μία δομή δυάδων (std::pair<int,int>) τύπου integer.O λόγος που αποθηκεύεται ως integer είναι διότι η είσοδος του χρήστη από algebraic notation μετατρέπεται σε συντενταγμένη τύπου integer.

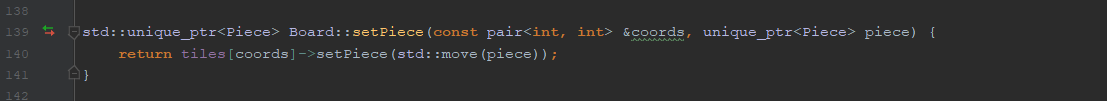
Η κλάση Tile ορίζει έναν δείκτη σε αντικείμενο Piece.Συγκεκριμένα η αποθήκευση των πιονιών γινεται σε map container όπου κλειδί είναι οι συντεταγμένες για κάθε πιόνι και και τιμή ένας δείκτης (smart pointer ) στην κλάση Tile.

Η κλάση Piece είναι μία αφηρημένη κλάση από την οποία προκύπτουν όλα τα κομμάτια της παρτίδας. Υπάρχουν έξι κομμάτια κομμάτια στο σκάκι που είναι pawn, rook, knight, bishop, queen and king. Το μόνο πράγμα που διαφοροποιεί έναν τύπο Piece από τον άλλο είναι ο τρόπος με τον οποίο κινούνται.

Ακολουθεί στιγμιότυπο με τα ιδιωτικά δεδομένα της κλάσης Board:

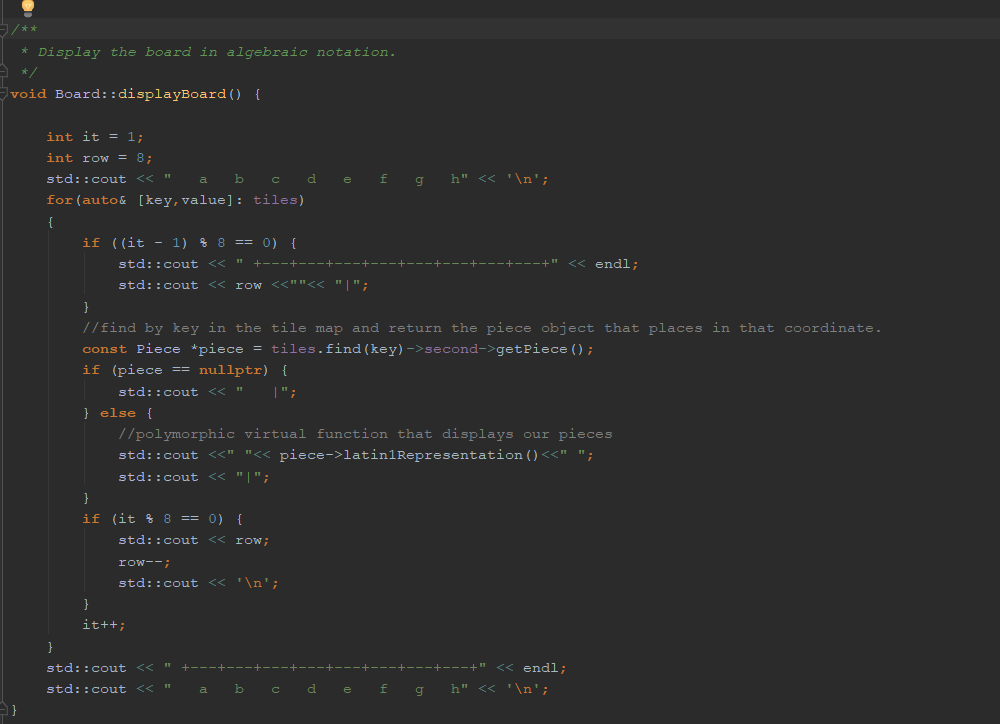


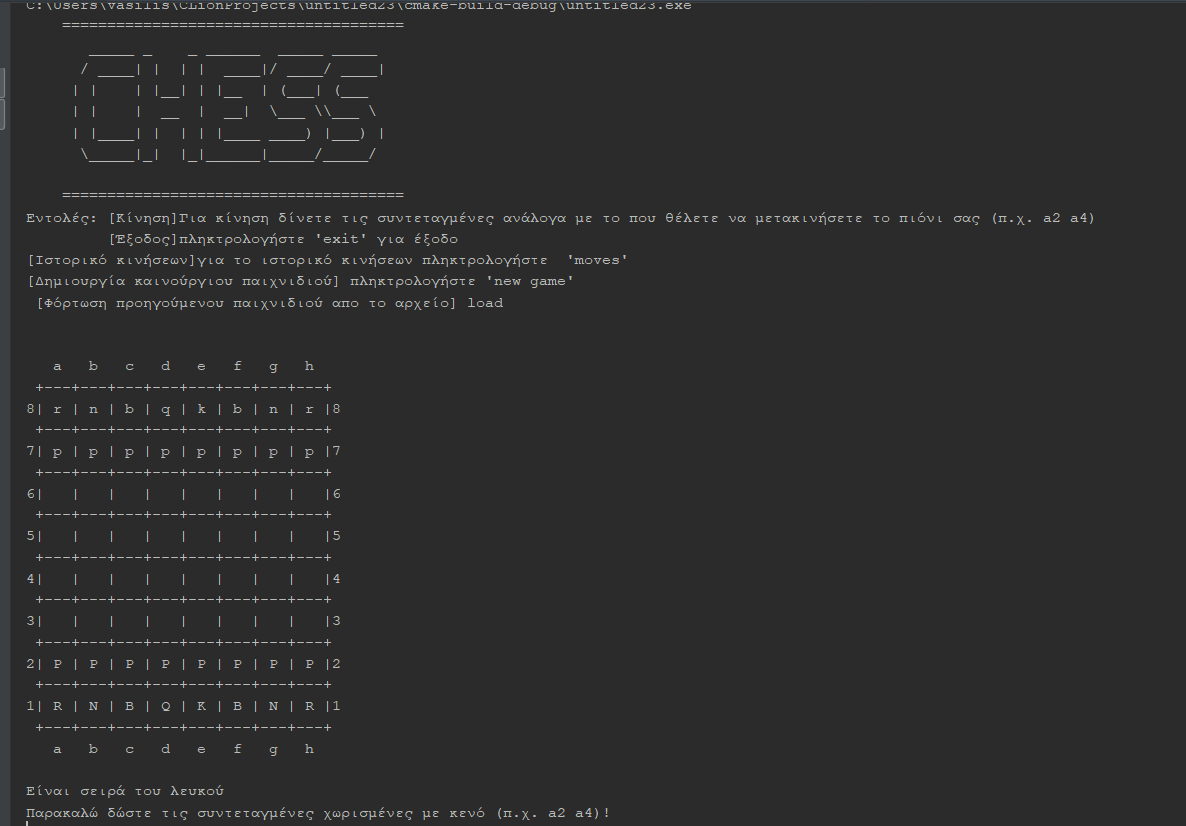
Όπως διαφαίνεται παράπανω η μεταβλητή tiles αναπαριστά έναν mapper (assosiative container) από συντεταγμενες και από δείκτη στην κλάση Tile.Εκεί αποθηκευόνται και επεξεργάζονται όλα τα κομμάτια της παρτίδας.Ο default constructor καλεί την κάθε μέθοδο που είναι υπεύθυνη για την δημιουργία των κομματιών της παρτίδας.Για τον σκοπό αυτό έχει δημιουργηθεί μία μέθοδος setPiece.



Η συγκεκριμένη μέθοδος δημιουργεί ένα πιόνι, το οποίο κληρονομεί την αφηρημένη κλάση Piece , και εν συνεχεία ανάλογα με τις συντεγμένες που δίνονται προσπελαύνει την μέθοδο setPiece της κλάσης Tile η οποία αρχικοποιεί το εκάστοτε Piece σε έναν δείκτη που τύπου Piece.Έτσι με αυτόν τον τρόπο μπορούμε να προσπελάσουμε οποιόδηποτε πιόνι χρησιμοποιώντας την κλάση Tile και όχι την Piece μειώνοντας έτσι το κόστος εφόσον είναι και based κλάση.

Aφου γίνουν construct τα αντικείμενα για κάθε πιόνι η κλάση Board διαθέτει μία μέθοδο η οποία προοσομοιάζει το την σκακιέρα και προσδίδει ένα φιλικό γραφικό περιβάλλον προς τον χρήστη.Παρακάτω διαφαίνεται η μέθοδος αυτή καθώς και τα αποτελέσματά της:





Όπως φαίνετι παραπάνω μαζί με το ταμπλό εμφανίζεται και το banner του παιχνιδιού καθως και οι επιλογές που μπορεί να κάνει ο χρήστης. Αν ο χρήστης επιθυμεί να δημιουργήσει καινούργιο παιχνίδι μπορεί να πληκτρολογήσει ‘new game’ και να ξεκινήσει η νέα παρτίδα.

**2.2.1 Φόρτωση ταμπλό από δυαδικό αρχείο**

Η φόρτωση του δυαδικού αρχείου στον mapper δηλαδή στο ταμπλό γίνεται αφου ο λευκός πραγματοποιήσει την δεύτερή του κίνηση στο παιχνίδι.Αυτό ισχύει μόνο για τον λευκό παίκτη.Ωστόσο η εγγραφή στο αρχείο γίνεται μετά από κάθε κίνηση που πραγματοποιείται στο παιχνίδι άσχετα από το χρώμα του παίκτη.Επίσης , μπορεί να φορτωθεί η προηγούμενη παρτίδα πληκτρολογώντας την λέξη load.Προυπόθεση για το προηγούμενο να μην έχει ξεκινήσει νέο παιχνίδι.

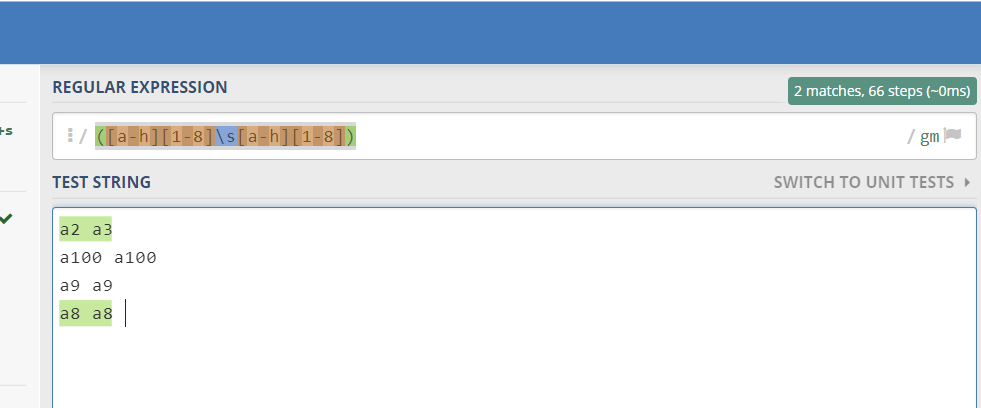
**2.3 Έλεγχος ορθότητας των συντεταγμένων που δίνει ο χρήστης**

Η μέθοδος run της κλάσης Game μας προσφέρει επίσης κάποια πολύ χρήσιμα χαρακτηριστικά για την σωστή λειτουργία του παιχνιδιού.Ας υποθέσουμε ότι ο χρήστης δίνει λανθασμένες συντεταγμένες και το πρόγραμμα κρασάρει.Για τον λόγο αυτό και για την προλάβουμε την προσπέλαση της Κλάσης Board, και βεβαίως με την βοήθεια της μοντέρνας c++, δημιουργήθηκε regular expression για την εισαγωγή συγκεκριμένων συντεταγμένων τύπου algebraic notation κατά την είσοδο.

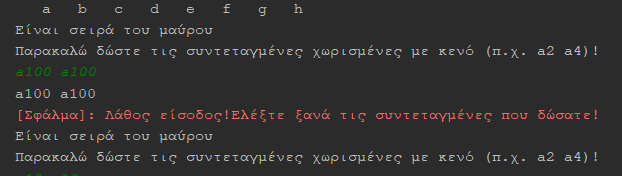


Παραπάνω διακρίνεται η κανονική έκφραση για την αποδοχή δύο διαφορετικών χωρισμένων με κενό συντεταγμένων από το standard input.

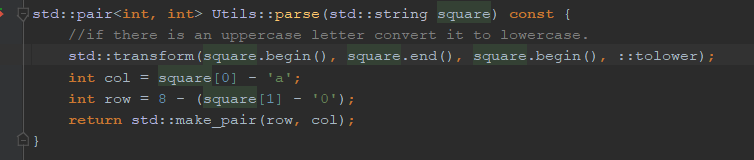
Ας την επαληθεύσουμε χρησιμοποιώντας το εργαλείο <https://regex101.com/>:



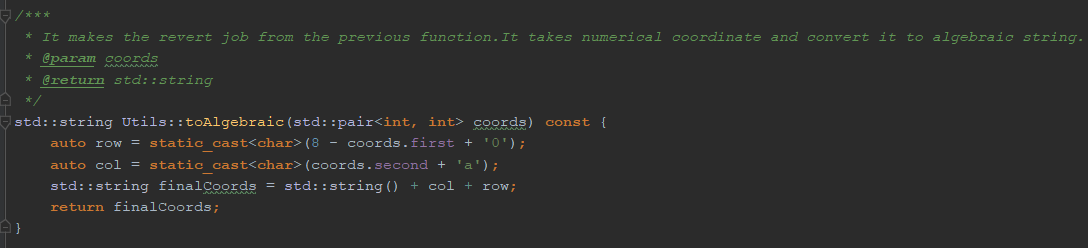
Ακολουθεί ενδεικτική μη αποδεκτή είσοδο σύμφωνα με όσα προαναφέρθηκαν:



Από την άλλη μεριά αν ο χρήστης δώσει αν ο χρήστης δώσει τις σωστές συντεταγμένες τύπου algebraic notation τότε διασπώνται με διαχωριστή το κενό και αποθηκεύνται σε έναν vector (dynamic array) έτσι ώστε να γίνει το parsing σε numeric value.Αυτή την μετατροπή από algebraic notation σε συντεταγμένες τύπου std::pair<int,int> την αναλαμβάνει η μέθοδος parse της κλάσης Util.



Την αντίστροφη διαδικασία δηλαδή από std::pair<int, int> σε string και συγκεκριμένα σε algebraic notation την αναλαμβάνει η μέθοδος toalgebraic της ίδιας κλάσης.

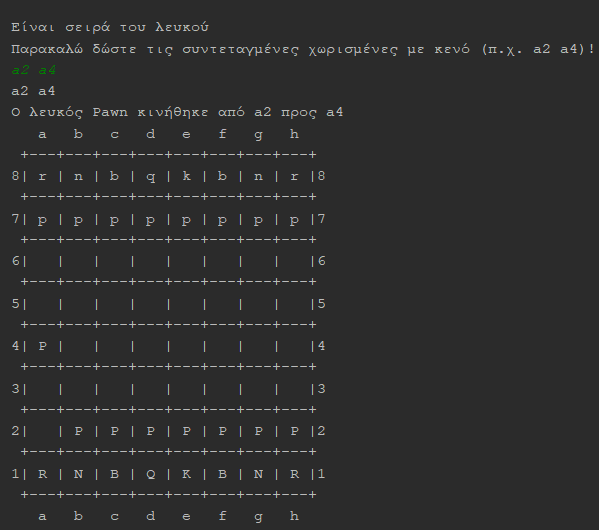


**2.4 Υποβολή κίνησης και έλεγχος ορθότητας κίνησης ανάλογα με το πιόνι**

Εφόσον λοιπον έχουμε πραγματοποιήσει τον βασικό έλεγχο που αφορα την ορθότητα των συντεταγμένων περνάμε στον λογικό έλεγχο της εισόδου του χρήστη.Στόχος της παρούσας φάσης είναι να αποφευχθει η χρήση εισόδων που παραβιάζουν την ορθή λειτουργία του παιχνιδιού.Για τον λόγο αυτό εχει δημιουργηθεί μία Enum Class (thread safe ,προτειμάται έναντι του παραδοσιακού Enum) με όνομα ErrorCodes στην οποία βρίσκονται όλα τα σφάλματα αλλά και καταστάσεις του προγράμματος που προκειται να το τερματίσουν ή να το εκκινήσουν.Παρακάτω παρατίθενται οι κωδικοί προγράμματος ανά κατηγορία:

|  |  |
| --- | --- |
| *Κωδικοί Σφαλμάτων* | Περιγραφή |
| ERROR\_INVALID\_INPUT\_SOURCE | Λανθασμένη εισαγωγή συντεταγμένων πηγής |
| ERROR\_INVALID\_INPUT\_AND\_SOURCE | Λανθασμένη εισαγωγή συντεταγμένων πηγής και  Προορισμού |
| ERROR\_INVALID\_INPUT\_DESTINATION | Λανθασμένη εισαγωγή συντεταγμένων προορισμού |
| ERROR\_NO\_PIECE\_AT\_SOURCE | Επιλογή κενής θέσης ως πιόνι |
| ERROR\_WRONG\_PLAYER | Λανθασμένη επιλογή πιονιών ανάλογα με το χρώμα του κάθε παίκτη |
| ΕRROR\_SOURCE\_AND\_DESTINATION\_ARE\_SAME | Οι συντεταγμένες πηγής και προορισμού είναι όμοιες |
| ERROR\_FRIENDLY\_PIECE\_AT\_DESTINATION | Στις συντεταγμένες προορισμού υπάρχει φιλικό πιόνι |
| ΕRROR\_INVALID\_PIECE\_MOVE | Λανθασμένη κίνηση για το εκάστοτε πιόνι |
| ERROR\_OWN\_KING\_IN\_CHECK | Θέτει τον βασιλιά του παίκτη που έχει σειρά σε check |

Aφού γίνουν οι έλεγχοι για τα παραπάνω σφάλματα υποβάλλεται η κίνηση.Έπειτα ελέγχονται κάποιες ιδιαιτερότητες που αφορούν την ορθή εκτέλεση του παιχνιδιού όπως για παράδειγμα εάν καποιο πιόνι έχει αιχμαλωτιστεί ή αν η κίνηση που πραγματοποιήθηκε ήταν από το pawn και αν ήταν η πρώτη κίνηση του πιονιού στο παιχνίδι.Eν συνεχεία γίνονται έλεγχει για καταστάσεις που τερματίζουν το παιχνίδι ( checkmate,stalemate) ή θέτουν σε κίνδυνο τον βασιλιά του παίκτη.Ο χρήστης ενημερώνεται κατάλληλα με μηνύματα που διευκρινίζουν την κατάσταση του παιχνιδιού.



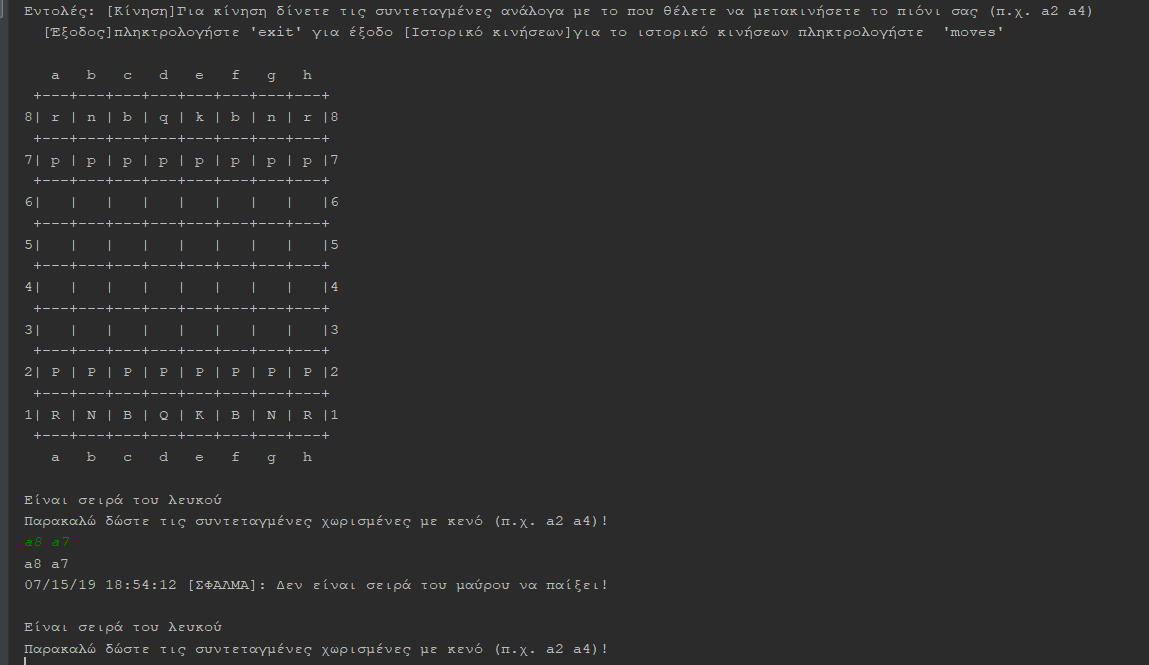
* 1. **Εκκίνηση της λογικής του παιχνιδιού**

Η συγκεκριμένη φάση είναι η καρδιά του παιχνιδιού καθώ περιέχει όλη την λογική του.Βέβαια το σκάκι είναι ένα διαχρονικό παιχνίδι με ανεξίτιλους κανόνες.Στο συγκεκριμένο πρόγραμμα έχουν υλοποιηθεί οι κανόνες ανά πιόνι οι οποίοι αναπαρίστανται μέσω μίας pure virtual συνάρτησης εν ονόματι ***isValidMove***:

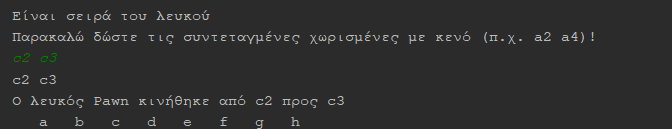


Αρχικά η κλάση Piece είναι η κλάση πατέρας και το κάθε πιόνι θεωρείται παιδί του.Εδώ με την δύναμη του runtime πολυμορφισμού έχουμε την δυνατότητα να χρησιμοποιήσουμε pure virtual function για κάθε πιόνι και να το προσπελάσουμε σχετικά έυκολα με την τεχνική του dynamic\_casting <> δηλαδη του downcasting εν προκειμένω από πατέρα σε παιδί.

Όπως αναφέρθηκε παραπάνω έχουν δημιουργηθεί error codes για την επιστροφή σαφών μηνυμάτων λάθους στον χρήστη.Επομένως αν χρήστης προσπαθήσει να μετακινήσει πιόνι του αντιπάλου το αποτέλεσμα θα είναι να εμφανιστεί ένα μήνυμα λάθους το οποίο αποσαφηνίζει την παράνομη κίνηση.Ακολουθεί ενδεικτικό παράδειγμα( για περισσότερα error cases ανατρέξτε στον πίνακα που περιγράφονται οι κωδικοί λάθους):



Αφού ελεγχθούν όλες οι συνθήκες λάθους η κίνηση υποβάλλεται και ένα συνεχεία ελέγχεται αν ο παίκτης που υπέβαλλεται την κίνηση είναι σε check αν είναι όντως σε check .Σε περίπτωση που ισχύουν τα προηγούμενα η κίνηση που υπέβαλλε αναιρείται η κίνηση αυτή και εμφανίζεται το ανάλογο μήνυμα λάθους.Αντιθέτως αν τα προηγούμενα δεν είναι αληθή ένα μήνυμα με την εκάστοτε κίνηση του εκάστοτε παίκτη επιστρέφεται στον client:

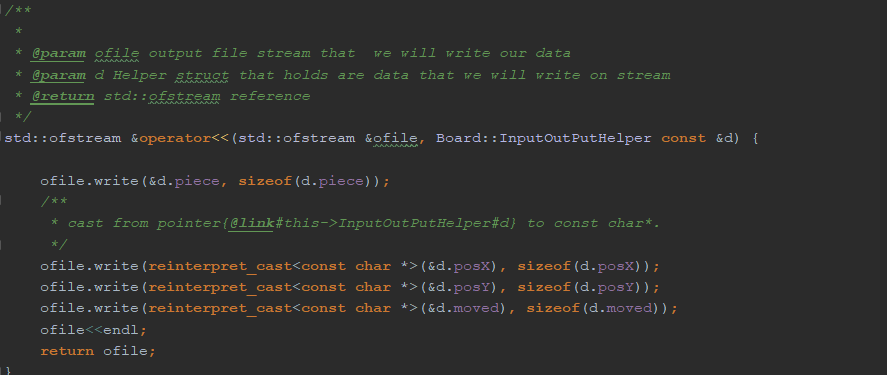


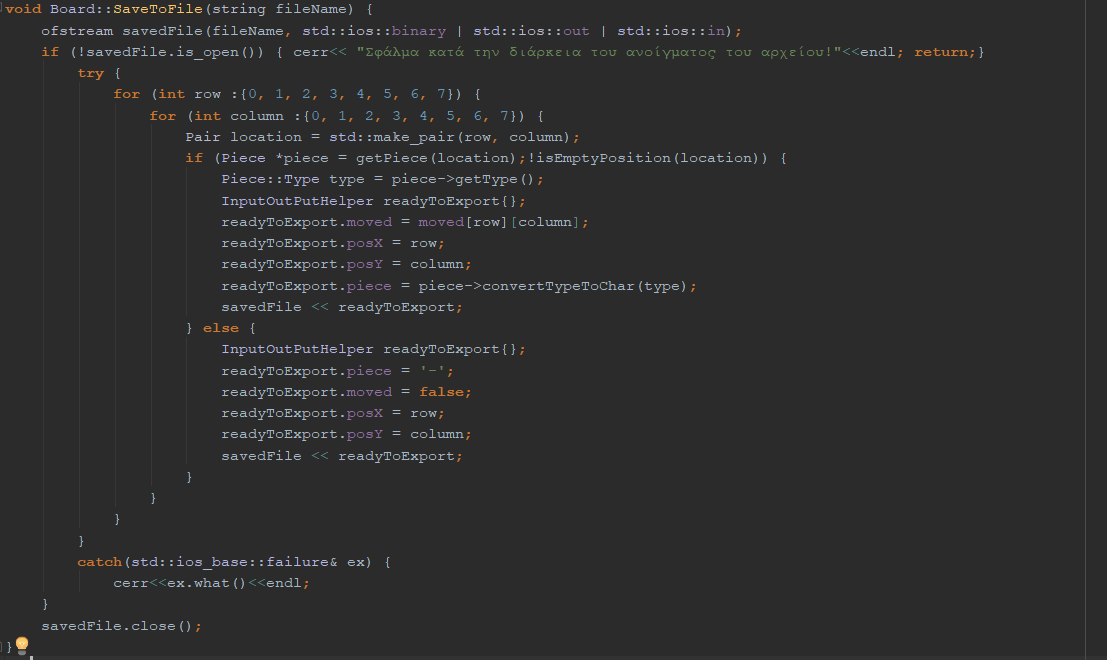
Επιπλέον ένας προσωρινός smart pointer τύπου Piece με όνομα capturedPiece χρησιμοποιείται για την αποθήκευση ενός αιχμαλωτισμένου πιονιού από τον αντίπαλο.Αν έχει πραγματοποιηθεί αιχμαλώτιση από κάποιο παίκτη ανάλογο μήνυμα εμφανίζεται στην οθόνη.



Από κει και πέρα το παιχνίδι συνεχίζεται και σειρά έχει ο επόμενος παίκτης.Πρωτού όμως πάρει τον έλεγχο του παιχνιδιού το πρόγραμμα ελέγχει αν είναι σε check.Αν ο παίκτης είναι σε check ελέγχεται αν μπορεί να εκτελέσει κάποια κίνηση (μέσω μία μεθόδου hasAnyValidMove η οποία θα αναλυθεί παρακάτω) .Αν δεν μπορει να εκτελέσει καμία κίνηση τότε έιναι σε check αλλιώς σε checkmate και το παιχνίδι τελειώνει εκεί καθώς ο παίκτης αυτός έχει ηττηθεί.Ένας τελευταίος έλεγχος που πραγματοποιείται πρωτού ο παίκτης αγωνιστεί σχετίζεται με το αν μπορεί ο εν λόγω παίκτης να εκτέλεσει οποιαδήποτε κίνησει.Αν αυτό δεν είναι εφικτό τότε έχουμε ισοπαλία και το παιχνίδι τελειώνει εκεί.

Σημειώνεται ότι στην αλλάγη της σειράς τα δεδομένα τα οποία είναι αποθηκευμένα στον mapper αποθηκεύονται σε ένα δυαδικό αρχείο μέσω της μεθόδου saveΤοFile.Στην προκείμενη περίπτωση έχει υπερφορτωθεί ο τελεστής << και ένα struct InputOutHelper βοηθά στην διαδικασία αυτή κάνοντας τον κώδικα πιο ευανάγνωστο.





* 1. **Ενημέρωση του ταμπλό άναλογα με την νομιμότα των κινήσεων ανά πιόνι**

Αφου πραγματοποιηθούν όλοι οι έλεγχοι που παραβιάζουν τους κανόνες και την νομιμότηττα του παιχνιδιού ενημερώνεται το ταμπλό .Υπεύθυνη μέθοδος γι’ αυτό είναι η movePiece η οποία δεν ενημερώνει μόνο το ταμπλό αλλά εκτέλει και τα παρακάτω :

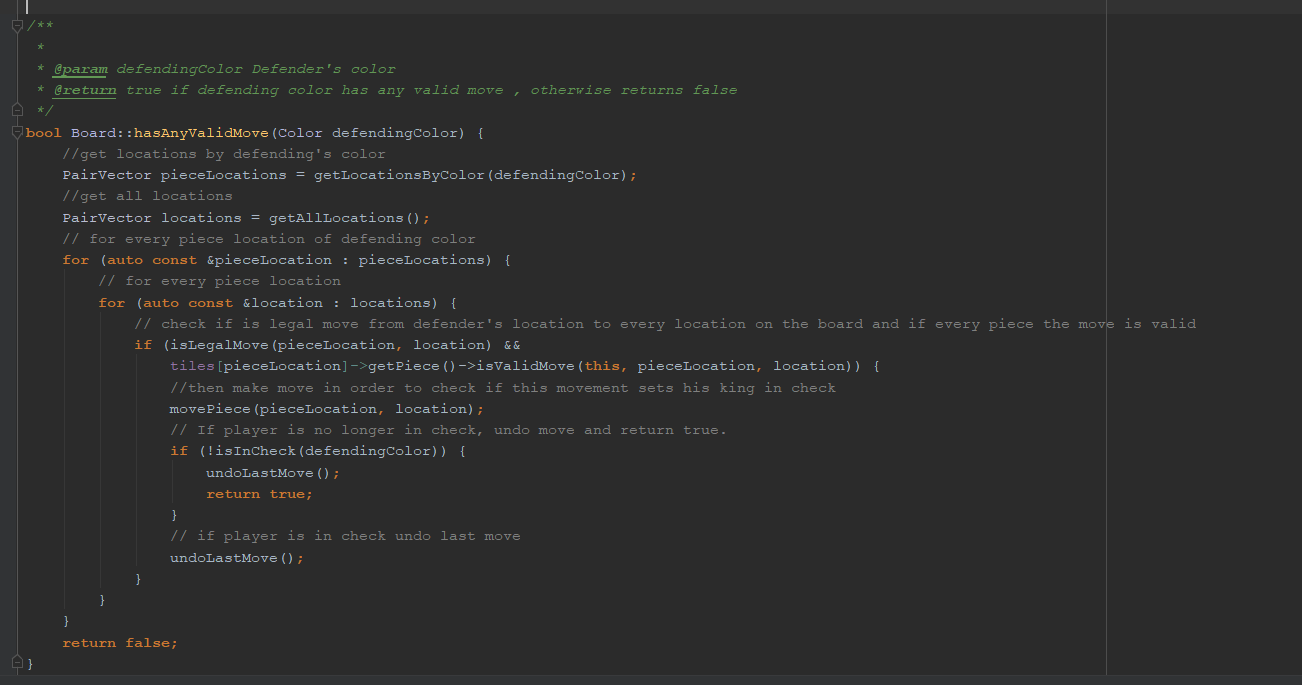
* Ενημέρωνει μία λίστα από pairs στην οποία αποθηκεύονται οι τρέχουσες κινήσεις του κάθε παίκτη.
* Ενημέρωνει ένα bitset πίνακα στον οποίο αποθηκεύονται οι δυαδικές τιμές 0 ή 1 .Στην προκείμενη περίπτωση αποθηκεύεται 1 στην τοποθεσία όπου μετακινήθηκε το πιόνι.
* Θέτει τον smart pointer capturedPiece τυπου Piece,ο οποίος χρησιμοποιείται προσωρινά για την αποθηκεύση και επαλήθευση των αιχμαλωτιζόμενων πιόνιων , ίση με το με το πιόνι που βρισκόταν στην τοποθεσία στόχο.Με αυτόν τον τρόπο αν υπήρχε κάποιο πιόνι στην τοποθεσία αυτή αποθηκεύεται στο capturedPiece διότι έχουμε αιχμαλώτιση πιονιού και ακολούθως εμφανίζεται στην οθόνη μαζί με άλλες χρήσιμες πληροφορίες του αγώνος.

Στην παρούσα φάση αξίζει να σημειώθει ότι το container std::map της stl παρέχει πολλές χρήσιμες μεθόδους και εύκολο iteration και γι΄αυτό το λόγο προτιμήθηκε έναντι άλλων λύσεων (π.χ. vector<vector<>>) .Τα keys μας όπου είναι οι τοποθεσίας του ταμπλό είναι μοναδικά και είναι ταξινομημένα στον mapper tiles πράγμα που σημάινει ότι μπορούμε να κάνουμε έυκολο lookup στο container.

**2.6 Υλοποίηση τερματικής κατάστασης του παιχνιδιού (checkmate,stalemate)**

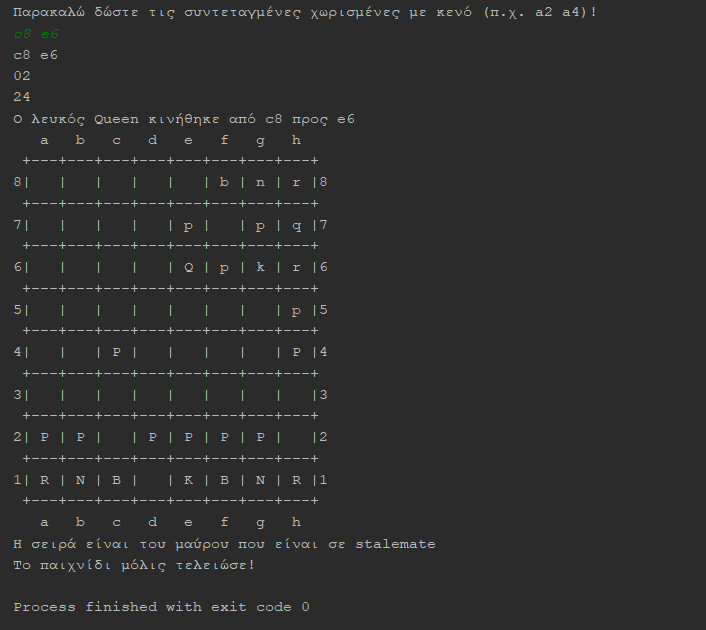
Όπως περιγράφηκε παραπάνω οι τερματικές καταστάσεις του παιχνιδιού είναι το checkmate και το stalemate.Η υλοποίηση των παράπανω πραγματοποιήθηκες με τις παρακάτω μεθόδους :

* hasAnyValidMove



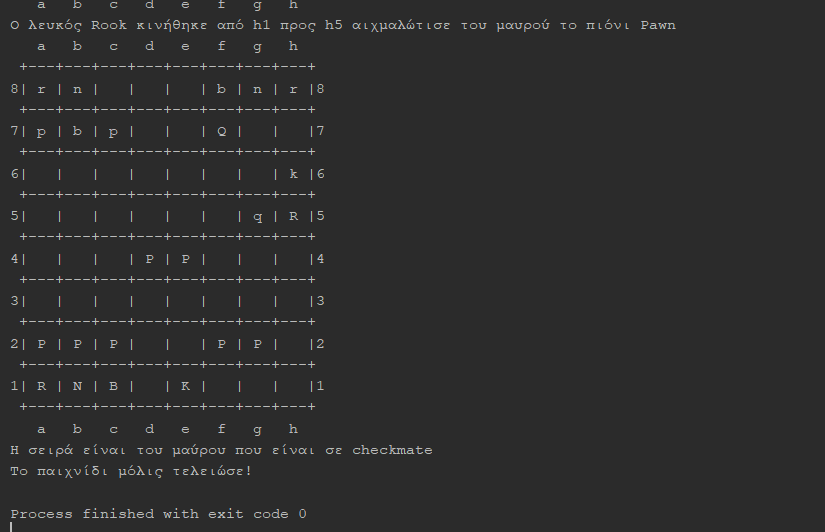
Αυτή η μέθοδος χρησιμοποιείται για να ελέγξουμε αν ο αμυνόμενος παίκτης μπορεί να πραγματοποιήσει οποιαδήποτε κίνηση σε αντίπαλο πιόνι.Άν αυτό είναι εφικτό πραγματοποιείται κίνηση για να ελέγξουμε αν είναι είναι σε check .Αν δεν είναι σε check τότε αναιρείται η κίνηση και επιστρεφέται true στον καλόντα.Διαφορέτικα αναιρείται η κίνηση και η διαδικασία επαναλαμβάνεται.

Αν δεν ικανοποιήθουν τα παράπανω δεν υπάρχει καμία ελπίδα ο παίκτης δεν μπορεί αν πραγματοποιήσει καμία κίνηση άρα έχουμε ισοπαλία!



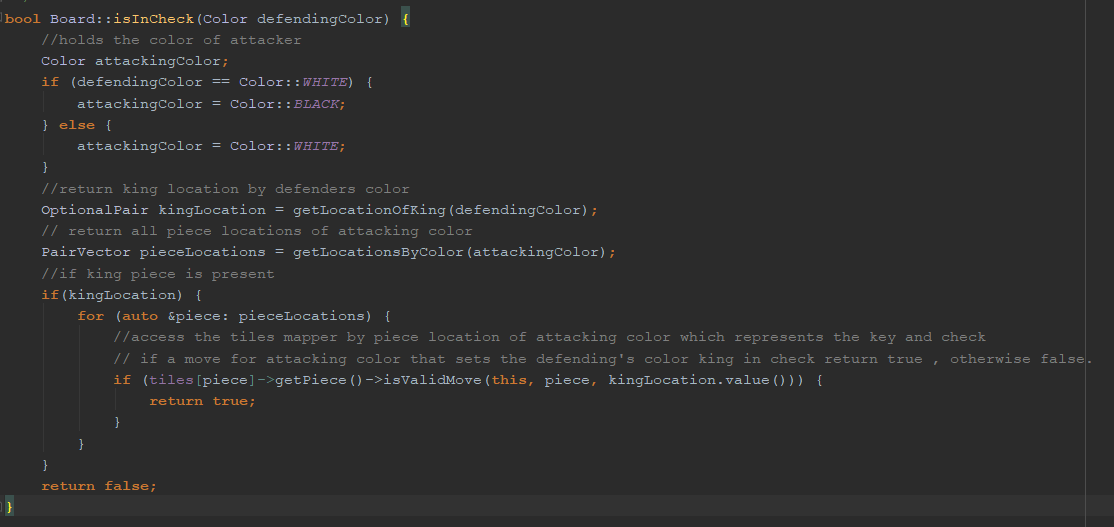
Παράπανω φαίνεται η περίτπτωση που έχουμε ισοπαλία και το παιχνίδι τερματίζει!

Αν ο παίκτης είναι σε check και μπορεί να πραγματοποιήσει κινήσεις τότε είναι σε check διαφορετικα είναι σε checkmate και εχει ηττηθεί.

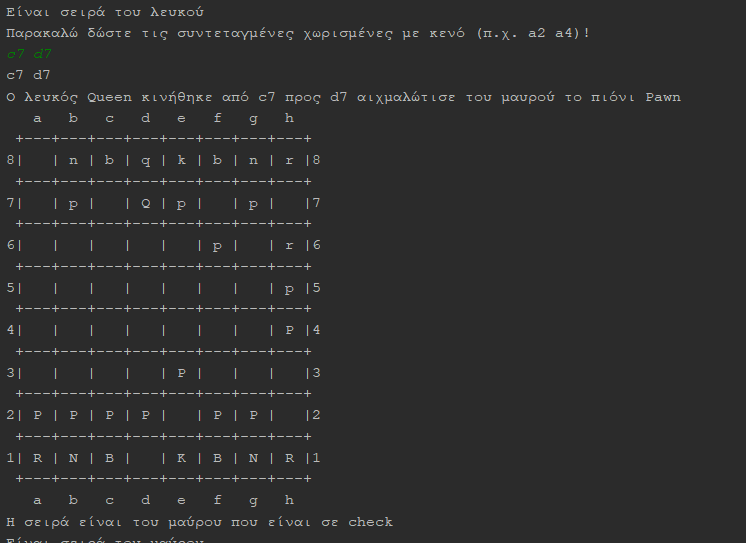


Παραπάνω φαίνεται η περίπτωση που ο μαύρος παίκτης είναι σε checkmate , επομένως το παιχνίδι τελειώνει και ο μαύρος έχει ηττηθεί.

Παρακάτω διαφαίνεται η μέθοδος isInCheck. Η οποία ελέγχει αν ο επιτιθέμενος παίκτης μπορεί να απειλήσει τον βασιλιά του αμυνόμενου παίκτη για κάθε στην οποία βρίσκεται.



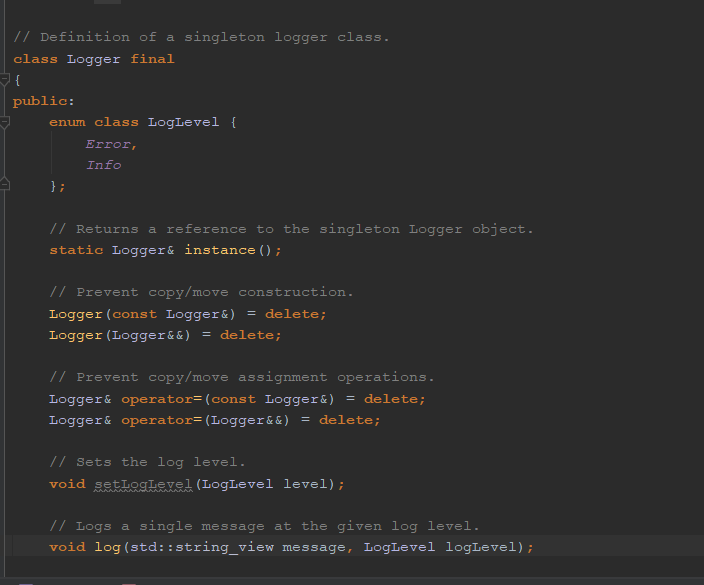
Παρακάτω διαφαίνεται το ενδεικτικό μήνυμα που εμφανίζεται στην κονσόλα του επόμενου παίκτη σε περίπτωση που είναι σε check.



Επίσης δίνεται η δυνατότητα να ελεγχούν τερματικές καταστάσεις πληκτρολογώντας “test staleMate” για τον έλεγχο του stalemate και “test checkMate” για τον έλεγχο του checkmate.Επίπλεον μπορούν να ελεγχθούν και διάφορες εσφαλμένοι είσοδοι πληκτρολογώντας “test errorCases” .

**3.Βοηθητικές Κλάσεις**

Για την εμφάνιση των μηνυμάτων λάθους έχει υλοποιηθεί μία singleton κλάση logger η οποία εμφανίζει το μήνυμα και την τοπική ώρα χρησιμοποιώντας την βιβλιοθήκη της cpp <chrono>.



**4. Οδηγίες μεταγλώττισης**

Για την μεταγλώττιση του προγράμματος χρησιμοποιήθηκε το εργαλείο cmake(<https://cmake.org/cmake/help/v3.10/manual/cmake.1.html>).

***Tι είναι το cmake;***

Το CMake αντιπροσωπεύει το Cross-platform του Make. Το CMake αναγνωρίζει ποιοι μεταγλωττιστές θα χρησιμοποιηθούν για ένα δεδομένο είδος πηγής. Το CMake επικαλείται τη σωστή ακολουθία εντολών για κάθε τύπο στόχο(target file Επομένως, δεν υπάρχει ρητή προδιαγραφή εντολών όπως $ (CC) που χρησιμοποιείται το make.

***Γιατί cmake;***

* Ανιχνεύει ευκολότερα τα cross-platforms των βιβλιοθηκών του συστήματος.
* Ανακαλύπτει και διαμορφώνει την αλυσίδα των εργαλείων.
* Είναι πιο εύκολο να μεταγλωττιστούν τα αρχεία σε μια κοινόχρηστη βιβλιοθήκη και γενικότερα είναι εύκολο στην χρήση.

1. $ sudo apt-get -y install cmake
2. $ which cmake
3. $ usr/bin/cmake
4. $ cmake --version

cmake version 3.10.0

To version πρέπει να είναι >= 3.10.0.Αν δεν κατεβάσει το επιθυμητό version παρακαλώ όπως χρησιμοποιήσατε τον παρακάτω τρόπο:

1. $ sudo apt-get purge cmake
2. $ mkdir ~/temp
3. $ cd ~/temp
4. $ wget https://cmake.org/files/v3.10/cmake-3.10.0.tar.gz
5. $ tar -xzvf cmake-3.10.0.tar.gz
6. $ cd cmake-3.10.0/
7. $ ./bootstrap
8. $ make -j4
9. $ sudo make install
10. $ cmake –version

Πρέπει να είναι 3.10.0 >=

cmake version 3.10.X

Για διαγραφή του εργαλείου παρακαλώ πληκτρολογήστε την παρακάτω εντολή:

$ sudo apt-get purge cmake

**Μεταγλώττιση και εκτέλεση προγράμματος**

$ cd chess-game

$ cmake CMakelists.txt

$ make

$ ./chess-game

**C++ version**

C++17