

ΕΡΓΑΣΤΗΡΙΟ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΕΡΓΑΣΙΑ ΣΚΑΚΙ



Ονοματεπώνυμο: Βασίλης Τιμούδας

Αριθμός Μητρώου: 171066

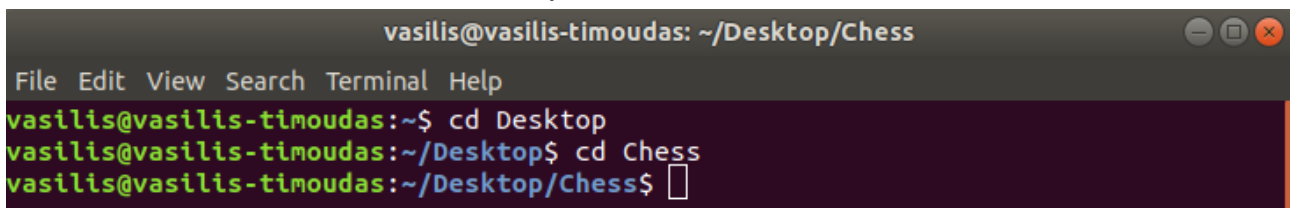
Email: cs171066@uniwa.gr

Περιεχόμενα

1. Οδηγίες μεταγλώττισης	3
2. Χρήσιμες πληροφορίες	4
3. Τι δεν υλοποιήθηκε – λάθη κώδικα	5
4. Τι υλοποιήθηκε και πως	5
5 Επεξηγήσεις σε ορισμένα σημεία του κώδικα	10
6. Ενδεικτικά τρεξίματα κώδικα	15

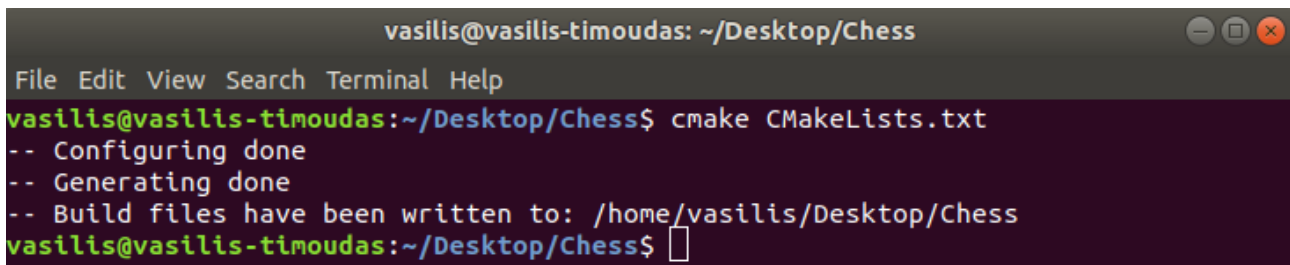
1. Οδηγίες μεταγλώττισης

1) Αρχικά κάνουμε `cd` στον φάκελο Chess.

A terminal window titled 'vasilis@vasilis-timoudas: ~/Desktop/Chess'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the following commands and output:

```
vasilis@vasilis-timoudas:~$ cd Desktop
vasilis@vasilis-timoudas:~/Desktop$ cd Chess
vasilis@vasilis-timoudas:~/Desktop/Chess$
```

2) Στην συνέχεια γράφουμε `cmake CMakeLists.txt`

A terminal window titled 'vasilis@vasilis-timoudas: ~/Desktop/Chess'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows the following commands and output:

```
vasilis@vasilis-timoudas:~/Desktop/Chess$ cmake CMakeLists.txt
-- Configuring done
-- Generating done
-- Build files have been written to: /home/vasilis/Desktop/Chess
vasilis@vasilis-timoudas:~/Desktop/Chess$
```

(Αν υπάρχει πρόβλημα με το version του cmake ανοίγουμε το CMakeLists.txt και αλλάζουμε το version στην πρώτη γραμμή)

3) Γράφουμε `make`

```
vasilis@vasilis-timoudas: ~/Desktop/Chess
File Edit View Search Terminal Help
vasilis@vasilis-timoudas:~/Desktop/Chess$ cmake CMakeLists.txt
-- Configuring done
-- Generating done
-- Build files have been written to: /home/vasilis/Desktop/Chess
vasilis@vasilis-timoudas:~/Desktop/Chess$ make
Scanning dependencies of target Chess
[ 8%] Building CXX object CMakeFiles/Chess.dir/main.cpp.o
[ 16%] Building CXX object CMakeFiles/Chess.dir/piece.cpp.o
[ 25%] Building CXX object CMakeFiles/Chess.dir/pawn.cpp.o
[ 33%] Building CXX object CMakeFiles/Chess.dir/rook.cpp.o
[ 41%] Building CXX object CMakeFiles/Chess.dir/knight.cpp.o
[ 50%] Building CXX object CMakeFiles/Chess.dir/bishop.cpp.o
[ 58%] Building CXX object CMakeFiles/Chess.dir/queen.cpp.o
[ 66%] Building CXX object CMakeFiles/Chess.dir/king.cpp.o
[ 75%] Building CXX object CMakeFiles/Chess.dir/empty.cpp.o
[ 83%] Building CXX object CMakeFiles/Chess.dir/board.cpp.o
[ 91%] Building CXX object CMakeFiles/Chess.dir/game.cpp.o
[100%] Linking CXX executable Chess
[100%] Built target Chess
vasilis@vasilis-timoudas:~/Desktop/Chess$
```

4) Τέλος γράφουμε ./Chess

```
Activities Terminal Περ 17:59
vasilis@vasilis-timoudas: ~/Desktop/Chess
File Edit View Search Terminal Help
vasilis@vasilis-timoudas:~/Desktop/Chess$ ./Chess
Εργασία Εξαμήνου ΣΚΑΚΙ
ΑΜ: 171066
Ονοματεπώνυμο: Βασίλης Τιμούδας

  a | b | c | d | e | f | g | h |
  --+---+---+---+---+---+---+---+
8 | r | n | b | k | q | b | n | r |
  --+---+---+---+---+---+---+---+
7 | p | p | p | p | p | p | p | p |
  --+---+---+---+---+---+---+---+
6 |   |   |   |   |   |   |   |   |
  --+---+---+---+---+---+---+---+
5 |   |   |   |   |   |   |   |   |
  --+---+---+---+---+---+---+---+
4 |   |   |   |   |   |   |   |   |
  --+---+---+---+---+---+---+---+
3 |   |   |   |   |   |   |   |   |
  --+---+---+---+---+---+---+---
```

2. Χρήσιμες πληροφορίες

Για την υλοποίηση της εργασίας χρησιμοποίησα το JetBrains Clion.

Το project τρέχει και σε windows και σε linux. Στα windows μόνο μπορεί να έχει πρόβλημα με τα ελληνικά. Στα linux τα ελληνικά λειτουργούν κανονικά.

Το Binary αρχείο είναι το **boardState.bin** και βρίσκεται μέσα στον φάκελο **cmake-build-debug**.

3. Τι δεν υλοποιήθηκε – λάθη κώδικα

Αρχικά δεν έχω υλοποιήσει σωστά την συνάρτηση που τερματίζει το παιχνίδι για να εμφανίζει νικητή ή ισοπαλία (η συνάρτηση είναι η `isGameOver` στην κλάση `Game`).

Επίσης δεν υλοποιήθηκε η κίνηση του πιονιού 2 θέσεις πάνω (αν είναι η πρώτη του κίνηση μόνο).

Λάθη κώδικα υπάρχουν όταν δίνουμε συντεταγμένες για να επιλέξουμε πιόνι.

πχ. αν δώσουμε συντεταγμένες τύπου a123 θα δεχτεί μόνο το a1 ενώ είναι λάθος.

4. Τι υλοποιήθηκε και πως

Αρχικά το binary αρχείο είναι το **boardState.bin** και βρίσκεται μέσα στον φάκελο cmake-build-debug.

1. Αποθήκευση της τρέχουσας κατάστασης του ταμπλό σε αρχείο.

game.h

```
57 void save(ofstream &of);  
58 void writeBinaryFile(string strFile);
```

```
30 struct Data  
31 {  
32     char y, x, type, moved;  
33 };  
34  
35 Data *data = new Data[64];
```

Αρχικά το struct Data περιέχει αυτά που ζητάτε για το Binary αρχείο. Η συνάρτηση save τα διαβάζει από το struct Data τις τιμές του και τις περνάει στο binary αρχείο boardState.bin. Η συνάρτηση writeBinaryFile απλώς χρησιμοποιεί την συνάρτηση save.

2. Διάβασμα κατάστασης ταμπλό από αρχείο.

game.h

```
59 void readBinaryFile(string strFile);
```

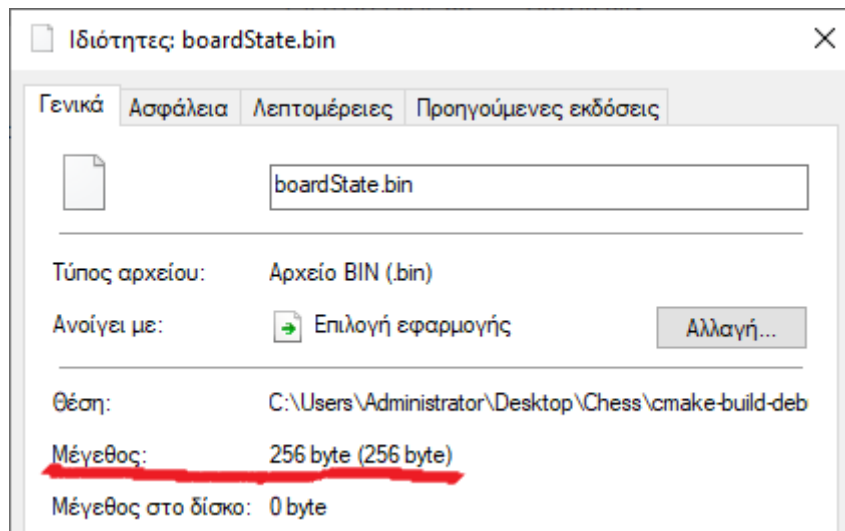
Η συνάρτηση readBinaryFile διαβάζει το binary αρχείο boardState.bin και τα αποθηκεύει στο struct Data.

3. Η αποθήκευση του ταμπλό θα γίνεται σε binary αρχείο. Για κάθε κομμάτι που βρίσκεται στο ταμπλό θα γράφεται μία εγγραφή. Η εγγραφή θα αποτελείται από τέσσερα πεδία του ενός byte:

- Την γραμμή του ταμπλό στην οποία βρίσκεται το κομμάτι
- Την στήλη του ταμπλό στην οποία βρίσκεται το κομμάτι
- Το γράμμα το οποίο συμβολίζει το κομμάτι (κεφαλαίο για λευκό κομμάτι, πεζό για μαύρο κομμάτι)
- Την τιμή 1 αν το κομμάτι έχει κινηθεί ή την τιμή 0 αν το κομμάτι δεν έχει κινηθεί.

Αρχικά είναι 4 byte όπως βλέπουμε και στο struct Data γιατί υπάρχουν 4 char του ενός byte.

(επιβεβαιώνεται με την εντολή `cout << sizeof(Data);`)



64 εγγραφές x 4 byte = 256 byte

Για το ύψος, το πλάτος και γράμμα δεν έχω να προσθέσω κάτι.

board.h

```
31      int piecesHasMoved[BOARD_SIZE][BOARD_SIZE]; // 1 όποτε κινήτε κάθε πιόνι αλλιώς 0

61  void initPieceHasMoved(); // αρχικοποιήτε με 0
62  void setPieceHasMoved(int posY, int posX, int value); // όποτε κινήτε ένα πιόνι η θέση του γίνεται 1
63  int getPieceHasMoved(int posY, int posX);
64  void getBoardPieceHasMoved(); // εμφάνιση όλου του 8x8 board για να δούμε ποιες θέσεις κομματιών έχουν κινήσει
```


Για το αν έχει κινηθεί χρησιμοποιώ δισδιάστατο πίνακα 8x8. Στην θέση που έχει κινηθεί το συγκεκριμένο piece μπαίνει η τιμή 1. Στην αρχή αρχικοποιούνται όλα με 0.

4. Καταγραφή κινήσεων παικτών. Το πρόγραμμα καταγράφει τις κινήσεις των παικτών και αν ζητηθεί μπορεί να επαναλάβει τις κινήσεις της παρτίδας.

game.h

```
28     vector<string> moves; // για καταγραφή όλων των κινήσεων στο παιχνίδι

49     void addMove(string m); // προσθέτουμε κίνηση
50     void showAllMoves(); // εμφάνιση όλων των κινήσεων
51     void repeatGame(); // για να επαναλάβει την παρτίδα
```

Χρησιμοποίησα vector για την καταγραφή των κινήσεων όπου περιέχει μέσα strings. Οι παραπάνω συναρτήσεις υλοποιούν τα ζητούμενα.

5. Το πρόγραμμα δέχεται από την standard είσοδο την κίνηση του κάθε παίκτη, ελέγχει την νομιμότητά της και αν είναι νόμιμη την εκτελεί.

game.h

```
40 void startNewGame();  
41  
42 void selectPiece(BOARD board, int id, int from[]); // για να διαλέξουμε κομμάτι  
43 void movePiece(BOARD board, int id, int from[], int to[]); // για να δώσουμε θέση κίνησης του κομματιού  
  
55 bool checkDistance(BOARD board, int fromY, int fromX, int toY, int toX);
```

board.h

```
41 bool isPickedRight(int playerId, int posY, int posX); // ο παίχτης 1(Λευκά) μπορεί να διαλέξει μόνο κεφαλαία  
42  
43 void doMove(int fromY, int fromX, int toY, int toX); // κίνηση
```

piece.h

```
31 // κάθε κομμάτι έχει διαφορετικό checkMove  
32 virtual int checkMove(int fromY, int fromX, int toY, int toX) = 0;
```

Η συνάρτηση selectPiece διαλέγει κομμάτι.

Η συνάρτηση movePiece διαλέγει θέση κίνησης κομματιού.

Η νομιμότητά της ελέγχεται από τις συναρτήσεις checkDistance και checkMove.

Η συνάρτηση doMove κάνει την κίνηση.

Τέλος όλα αυτά υλοποιούνται στην συνάρτηση startNewGame.

5. Επεξηγήσεις σε ορισμένα σημεία του κώδικα

το κενό στο ταμπλό το θεωρώ και αυτό piece



Τα κενά κληρονομούν την κλάση PIECE οπότε και αυτά έχουν τις ιδιότητές της όμως δεν θα τις χρησιμοποιούν καθώς ο παίχτης δεν μπορεί να διαλέξει το κενό.

```
void BOARD::initBoard()
```

```

40      // Αρχικοποίηση και δημιουργία κενών στο ταμπλό
41      for(int i=2; i<6; i++)
42          for(int j=0; j<8; j++)
43              board[i][j] = new Empty(false, i, j);

```

Αρχικοποιούνται και αυτά στο ταμπλό. Το χρώμα τους είναι αδιάφορο είτε είναι άσπρο είτε είναι μαύρο.

τι γίνεται όταν θέλω να μετακινήσω ένα piece σε ένα κενό?

Παράδειγμα μετακινούμε ένα πιόνι 2 θέσεις πάνω



Απάντηση

```
void BOARD::doMove()
```

```

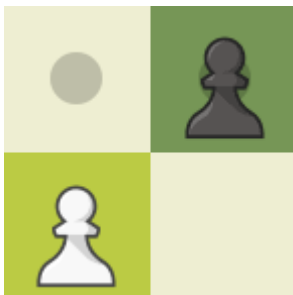
116     PIECE *temp[1][1];
117
118     if(this->board[toY][toX]->getType() == ' ') // κάνει κίνηση σε άδεια θέση
119     {
120         // swap
121         temp[0][0] = board[fromY][fromX];
122         board[fromY][fromX] = board[toY][toX];
123         board[toY][toX] = temp[0][0];
124     }

```

Κάνει swap την θέση του πιονιού με την θέση του κενού.

τι γίνεται όταν θέλω να επιτεθώ σε ένα αντίπαλο piece?

Παράδειγμα επίθεση άσπρου pawn σε μαύρο pawn



Απάντηση

void BOARD::doMove()

```
125     else // δεν είναι άδεια η θέση (υπάρχει αντίπαλο πιόνι)
126     {
127         // swap
128         temp[0][0] = board[fromY][fromX];
129         board[fromY][fromX] = board[toY][toX];
130         board[toY][toX] = temp[0][0];
131
132         getPieceAtBoard(fromY, fromX)->~PIECE(); // διαγράφεται από την μνήμη εφόσον 'πεθαίνει' στο παιχνίδι
133
134         board[fromY][fromX] = new Empty(true, fromY, fromX); // στην θέση αυτού δημιουργείται κενό
135     }
```

Γίνεται swap ανάμεσα στο άσπρο με το μαύρο πιόνι. Το μαύρο πιόνι διαγράφεται από τον destructor. Στην θέση του μαύρου πιόνιου (αρχική άσπρου) δημιουργείται κενό.

Η συνάρτηση checkMove() κάνει return int

class PIECE

```
31 // κάθε κομμάτι έχει διαφορετικό checkMove
32 virtual int checkMove(int fromY, int fromX, int toY, int toX) = 0;
```

γιατί είναι τύπου int και όχι bool?

Παράδειγμα για άσπρο πιόνι

Γνωρίζουμε ότι το άσπρο πιόνι έχει 4 κινήσεις:

1. κίνηση 1 θέση πάνω
2. επίθεση 1 θέση πάνω δεξιά
3. επίθεση 1 θέση πάνω αριστερά
4. κίνηση 2 θέσεις πάνω αν είναι η πρώτη του κίνηση
(δεν υλοποιήθηκε)

```
13 int Pawn::checkMove(int fromY, int fromX, int toY, int toX) {  
14     if(this->color) // άσπρο πιόνι  
15     {  
16         if(toX == fromX && toY + 1 == fromY) // 1 θέση πάνω  
17             return 1;  
18         if(toX - 1 == fromX && toY + 1 == fromY) // επίθεση 1 θέση πάνω δεξιά  
19             return 2;  
20         if(toX + 1 == fromX && toY + 1 == fromY) // επίθεση 1 θέση πάνω αριστερά  
21             return 3;  
22     }
```

Στην κάθε περίπτωση κάνει το αντίστοιχο return.

Όμως για να μπορέσει το πιόνι να κάνει την αντίστοιχη κίνηση σε κάθε περίπτωση πρέπει να ελεγχθούν και κάποια άλλα πράγματα.

game.cpp

```

203 bool Game::checkDistance(BOARD board, int fromY, int fromX, int toY, int toX) {
204     char ptFrom = board.getPieceType(fromY, fromX);
205     int cMove = board.getPieceAtBoard(fromY, fromX)->checkMove(fromY, fromX, toY, toX);
206
207     if(ptFrom == 'P') // άσπρο πιόνι
208     {
209         if(cMove == 1 && !board.isNotEmpty(toY, toX)) // κίνηση 1 θέση πάνω
210             return true;
211         if(cMove == 2 && isEnemyPieceThere(board, fromY, fromX, toY, toX)) // επίθεση 1 θέση πάνω δεξιά
212             return true;
213         if(cMove == 3 && isEnemyPieceThere(board, fromY, fromX, toY, toX)) // επίθεση 1 θέση πάνω αριστερά
214             return true;

```

Περιπτώσεις

1. Όταν η συνάρτηση checkmove() κάνει return 1 σημαίνει πως ο παίχτης θέλει να κινήσει το πιόνι μια θέση πάνω οπότε ελέγχει άμα η πάνω θέση είναι άδεια.
2. Όταν η checkmove() κάνει return 2 σημαίνει πως κάνουμε επίθεση σε αντίθετο κομμάτι, συνεπώς ελέγχει αν στην πάνω δεξιά θέση υπάρχει αντίπαλο κομμάτι.
3. Ακριβώς το ίδιο με το 2 αλλά για αριστερά.

Συναρτήσεις που ξεκινάνε με translate

game.h

```

40 int translateY(char posY); // "μεταφραστής" πχ. δίνει ο χρήστης 'a' και επιστρέφει 0
41 int translateX(char posX); // "μεταφραστής" πχ. δίνει ο χρήστης '5' και επιστρέφει 4

```

board.h


```
55 char translateYFile(int posY); // 'μεταφραστής' μεταφράζει το ύψος του αρχείου  
56 char translateXFile(int posX); // 'μεταφραστής' μεταφράζει το πλάτος του αρχείου
```

Στην πρώτη περίπτωση ουσιαστικά 'μεταφράζει' τις συντεταγμένες που μας δίνει ο χρήστης για να διαλέξει η να δώσει θέση για να κινήσει κάποιο κομμάτι.

π.χ. δίνει a2 οπότε οι συναρτήσεις μεταφράζουν 6 0

Στην δεύτερη περίπτωση αντίστοιχα το ίδιο αλλά για το αρχείο.

Βέβαια θα μπορούσα να το κάνω με κάποια πρόσθεση η αφαίρεση ανάλογα τον κάθε ascii χαρακτήρα αλλά θεώρησα πιο σωστό και πιο βολικό να κάνω αυτές τις συναρτήσεις.

6. Ενδεικτικά τρεξίματα κώδικα

```
vasilis@vasilis-tinouzas:~/Desktop/Chess$ ./Chess
Εργασία Εξαμήνου ΣΚΑΚΙ
ΑΜ: 171066
Όνοματεπώνυμο: Βασίλης Τιμούδας

      a      b      c      d      e      f      g      h
-----
8      r      n      b      k      q      b      n      r
-----
7      p      p      p      p      p      p      p      p
-----
6
-----
5
-----
4
-----
3
-----
2      p      p      p      p      p      p      p      p
-----
1      R      N      B      K      Q      B      N      R
-----

Παίζει ο παίκτης: 1(Λευρά)
Διαλέξετε κομμάτι: a2
Διαλέξατε P
Δώστε θέση: a3
```

```
Παίζει ο παίκτης: 1(Λευρά)
Διαλέξετε κομμάτι: a2
Διαλέξατε P
Δώστε θέση: a3

      a      b      c      d      e      f      g      h
-----
8      r      n      b      k      q      b      n      r
-----
7      p      p      p      p      p      p      p      p
-----
6
-----
5
-----
4
-----
3      p
-----
2      p      p      p      p      p      p      p      p
-----
1      R      N      B      K      Q      B      N      R
-----

Παίζει ο παίκτης: 2(Μαύρα)
Διαλέξετε κομμάτι: b7
Διαλέξατε p
Δώστε θέση: b6
```

Παίζει ο παίκτης: 2(Μαύρα)
 Διαλέξτε κομμάτι: b7
 Διαλέξτε p
 Δώστε θέση: b6

	a	b	c	d	e	f	g	h
8	r	n	b	k	q	b	n	r
7	p		p	p	p	p	p	p
6		p						
5								
4								
3	p							
2		p	p	p	p	p	p	p
1	R	N	B	K	Q	B	N	R

Παίζει ο παίκτης: 1(Λευκά)
 Διαλέξτε κομμάτι: