ΕΡΓΑΣΙΑ ΣΤΟΝ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

ΠΕΜΠΤΟ ΠΑΡΑΔΟΤΕΟ (ΠΡΟΑΙΡΕΤΙΚΟ)

Υλοποίηση έξυπνου παίκτη για το παιχνίδι σε C++

Το πέμπτο παραδοτέο της εργασίας αφορά τη δημιουργία ενός έξυπνου παίκτη για το παιχνίδι Blokus. Σημειώστε ότι **το παραδοτέο είναι προαιρετικό και δε προσμετράται στις μονάδες της εργασίας.** Αντί αυτού, οι υλοποιήσεις σας θα συμμετάσχουν σε ένα τουρνουά! Οι παίκτες της νικήτριας ομάδας θα πάρουν **bonus 1.5 μονάδας**, οι παίκτες της δεύτερης καλύτερης ομάδας θα πάρουν **bonus 1.0 μονάδας** και οι παίκτες της τρίτης καλύτερης ομάδας (που θα προκύψει από μικρό τελικό) θα πάρουν **bonus 0.5 μονάδας**.

Σημειώστε ότι **για να φτιάξετε έναν έξυπνο παίκτη θα πρέπει να έχετε υλοποιήσει πρώτα το τέταρτο παραδοτέο της εργασίας**. Πριν λοιπόν ξεκινήσετε, θα πρέπει να **αντικαταστήσετε τα αρχεία computerplayer.cpp και algorithms.h** με τα υλοποιημένα αρχεία σας από το τέταρτο παραδοτέο.

Στη συνέχεια, θα πρέπει να υλοποιήσετε τον παίκτη σας στην κλάση SmartPlayer στα αρχεία **smartplayer.h** και **smartplayer.cpp**. Στα συγκεκριμένα αρχεία θα πρέπει αρχικά να θέσετε τη μεταβλητή team του παίκτη στον constructor ανάλογα με τον αριθμό της ομάδας σας χρησιμοποιώντας τρία ψηφία (π.χ. για την ομάδα 4 θα πρέπει να βάλετε όνομα "Team 004"). Στη συνέχεια, θα πρέπει να υλοποιήσετε την εξής συνάρτηση απόφασης για τον έξυπνο παίκτη:

Συνάρτηση makeMove
Η συνάρτηση αυτή καλείται για να κάνει μια κίνηση ο παίκτης στο ταμπλό. Λαμβάνει ως είσοδο το ταμπλό του παιχνιδιού και επιστρέφει δείκτη σε **μια κίνηση (αντικείμενο τύπου Move)** που περιλαμβάνει το κομμάτι, τη θέση του στο ταμπλό (γραμμή και στήλη όπου μπαίνει το άνω αριστερά άκρο του κομματιού), την περιστροφή του κομματιού (UP, RIGHT, DOWN, LEFT) και την αναστροφή του (NO, YES).

Για να κατασκευάσετε τη στρατηγική σας, προτείνεται να χρησιμοποιήσετε τα στοιχεία που δίνονται στον παίκτη. Με το board μπορείτε να γνωρίζετε σε ποια σημεία έχει κομμάτια ο παίκτης και ο αντίπαλός του, με τη μεταβλητή-δείκτη opponent μπορείτε να έχετε πρόσβαση στα κομμάτια του αντιπάλου.

Σημειώστε ότι παίκτης μπορεί να δει όλα τα αντικείμενα του παιχνιδιού (όπως π.χ. τα κομμάτια του αντιπάλου). Ωστόσο, **απαγορεύεται να πειράξει τα αντικείμενα που δίνονται (π.χ. απαγορεύεται η χρήση της μεθόδου placePiece στο αντικείμενο board που δίνεται)**. Ακόμα, ο παίκτης **δε θα πρέπει να κάνει αλλαγές στα κομμάτια του από τον πίνακα pieces ή να τα τοποθετεί στο αντικείμενο board που δίνεται**. Για να δοκιμάσετε τις καταστάσεις του ταμπλό τοποθετώντας κομμάτια μπορείτε να χρησιμοποιήσετε τις **συναρτήσεις deepCopy** που βρίσκονται στο ταμπλό, στα κομμάτια και στον παίκτη (αντίπαλο). Ως ένα παράδειγμα τέτοιας χρήσης μπορείτε να δείτε τη στρατηγική του ComputerPlayer που αντιγράφει αντικείμενα στην evaluateMove ώστε να δει πώς θα είναι το ταμπλό εφόσον παιχτεί κάποια κίνηση στην evaluateBoard.

Για τη στρατηγική σας μην ξεχνάτε ότι πρέπει να μαζέψετε περισσότερους πόντους από τον αντίπαλο παίκτη. Μπορείτε να μαζέψετε πόντους τοποθετώντας τα κομμάτια σας, ενώ κερδίζετε επιπλέον πόντους αν τοποθετηθούν όλα τα κομμάτια, και αν το τελευταίο κομμάτι που θα τοποθετηθεί είναι το μικρότερο (δηλαδή το πρώτο).

Σημειώστε, τέλος, ότι ο παίκτης πρέπει να παίζει κάθε κίνηση σε λογικό χρόνο, **3-4 δευτερόλεπτα το μέγιστο**, οπότε προτείνεται να προσαρμόσετε κατάλληλα τη στρατηγική σας (π.χ. μπορεί να μην είναι δυνατό να ελέγξετε όλες τις πιθανές κινήσεις, αλλά μπορεί να είναι δυνατό να ελέγξετε τις κινήσεις των μεγάλων κομματιών).

Για να δοκιμάσετε τον παίκτη σας, **θα πρέπει να αλλάξετε κατάλληλα τις εντολές στις γραμμές 20 και 21 του αρχείου main.cpp**. Στις συγκεκριμένες γραμμές μπορείτε π.χ. να δοκιμάσετε τον παίκτη σας έναντι του απλού ComputerPlayer ως εξής:

```
players[0] = new ComputerPlayer(0);
players[1] = new SmartPlayer(1);
```

Επίσης, μπορείτε π.χ. να δοκιμάσετε να παίξετε εσείς οι ίδιοι αντίπαλοι με τον παίκτη σας ως εξής:

```
players[0] = new HumanPlayer(0);
players[1] = new SmartPlayer(1);
```

Μπορείτε φυσικά να κάνετε αλλαγές και στον ComputerPlayer, ώστε να δείτε πώς ο παίκτης αντιμετωπίζει άλλες στρατηγικές (ή ακόμα και να αρχικοποιήσετε και όλους τους παίκτες με new SmartPlayer για να δείτε πώς ο παίκτης τα καταφέρνει ενάντια στον εαυτό του).

Παρατηρήσεις

Η υλοποίηση θα πρέπει να γίνει στη C++ και να μπορεί να ανοίξει με το CodeBlocks, **με τις εκδόσεις που χρησιμοποιούμε** στο πλαίσιο του μαθήματος. Ο κώδικάς σας θα πρέπει να είναι καλά τεκμηριωμένος, ώστε να είναι παντού σαφείς οι λεπτομέρειες υλοποίησης.
Για την υλοποίηση, σας δίνονται τα αρχεία κεφαλίδων .h των κλάσεων/συναρτήσεων που πρέπει να υλοποιήσετε καθώς και κάποιες βοηθητικές κλάσεις/συναρτήσεις. Επιπλέον, σας δίνεται ο κώδικας της συνάρτησης main (main.cpp). **Σε καμία περίπτωση δεν επιτρέπεται να επέμβετε στον κώδικα των κλάσεων και των συναρτήσεων αυτών. Σε περίπτωση που το κάνετε, η εργασία σας αυτομάτως θεωρείται λανθασμένη και μηδενίζεται. Θα πρέπει μόνο να αντιγράψετε τα αρχεία computerplayer.cpp και algorithms.h από το τέταρτο παραδοτέο σας και να γράψετε κώδικα μόνο στα αρχεία smartplayer.h και smartplayer.cpp.**

Παραδοτέο

Το παραδοτέο θα είναι **ένα αρχείο zip με όνομα Blokus.zip** που θα περιλαμβάνει **όλο το project (τα αρχεία που θα υλοποιήσετε αλλά και αυτά που σας έχουν δοθεί)**, δηλαδή **ακριβώς ίδιο με το αρχείο Blokus.zip που δίνεται**, φυσικά **με τον κώδικα υλοποιημένο**. Επιπλέον, προτείνεται πριν δημιουργήσετε το αρχείο zip, να κάνετε Clean το project (που γίνεται από το Codeblocks επιλέγοντας στο μενού Build → Clean).

Προθεσμία υποβολής

Το παραδοτέο πρέπει να παραδοθεί μέχρι τις **23:59 της Τετάρτης 8 Ιουνίου**. Καμία παρέκκλιση δε θα γίνει από την παραπάνω προθεσμία.