ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΑΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Master Chef

3η Εργασία

Υλοποίηση Κλάσεων Διαγωνισμών

Καρακώστας Κωνσταντίνος 7892 karakosk@ece.auth.gr

Κοσέογλου Σωκράτης 8837 sokrkose@ece.auth.gr

Με μια γρήγορη ματιά…

Η παρούσα εργασία έχει να κάνει με την υλοποίηση των κλάσεων για τους διαγωνισμούς του παιχνιδιού. Οι διαγωνισμοί αυτοί είναι οι εξής:

- Ομαδικός Διαγωνισμός (TeamCompetition)
- Διαγωνισμός Ασυλίας (ImmunityCompetition)
- Διαγωνισμός Δημιουργικότητας (CreativityCompetition)

Τέλος, το πρόγραμμα χωρίζεται σε ημέρες και κάθε ημέρα έχει και ένα διαγωνισμό, εκτός από την **κανονική ημέρα**.

1. Κανονική Ημέρα

Η κανονική ημέρα (όπως και κάθε ημέρα) ξεκινάει με τους παίκτες να **τρώνε** και να κάνουνε **δουλειές**. Γι' αυτό τον λόγο έχουμε δημιουργήσει την συνάρτηση **MorningRoutine()**. Τέλος η ημέρα (όπως και κάθε μέρα) τελειώνει με τους παίκτες να **τρώνε**, να **συναναστρέφονται με άλλους παίκτες** και να **κοιμούνται**. Γι' αυτό τον λόγο έχουμε δημιουργήσει την **NightRoutine()**. Οι υλοποιήσεις αυτών το συναρτήσεων φαίνονται παρακάτω:

```
1.  void MorningRoutine() { //Morning Work
2.      for (int j = 0; j < 2; j++) {
3.          teams[j].teamEats();
4.          teams[j].teamWorks();
5.      }
6.  }
1.  void NightRoutine() { //Night eat and sleep
2.      for (int j = 0; j < 2; j++) {
3.          teams[j].teamEats();
4.          teams[j].teamSleeps();
5.      } //End of a normal day
6.      for (int j = 0; j < 2; j++) teams[j].teamSocializes();
7.  }
```

2.Ημέρα Ομαδικού Διαγωνισμού

Αρχικά, το αρχείο **TeamCompetition.h** δημιουργεί την κλάση **Class TeamCompetition** και αρχικοποιεί όλες τις μεταβλητές-μέλη αλλά και συναρτήσεις-μέλη με βάση την εκφώνηση της εργασίας. Η πιο σημαντική συνάρτηση της κλάσης αυτής είναι η συνάρτηση **void compete(Team &r, Team &b)** η οποία δέχεται ως ορίσματα αναφορές τις δύο ομάδες του

παιχνιδιού (δηλ. **teams[0]** για την κόκκινη, **teams[1]** για την μπλε). Ακόμη, επειδή ο διαγωνισμός αυτός αποτελείτε από 3 γύρους, έχουμε δημιουργήσει στο header file ένα αντικείμενο **Round rounds[3]**. Τώρα, μέσα στην **compete()** έχουμε τις εξής ενέργειες: Η συνάρτηση **string RoundWinner(Team &r, Team &b, int round)** επιστρέφει την ομάδα-νικητή του κάθε γύρου. Για τον λόγο αυτό με την βοήθεια μιας **while** τρέχουμε την **RoundWinner** 3 φορές και οι μεταβλητές **wins1** και **wins2** δείχνουν τελικός ποια είναι η νικήτρια ομάδα του ομαδικού διαγωνισμού. Ακόμη, σε κάθε γύρο διαγωνίζονται 5 τυχαίοι παίκτες από κάθε ομάδα και ο κώδικας που υλοποιεί αυτή την διαδικασία φαίνεται παρακάτω.

```
int chosenPlayersIndex = 0;
int table[5]; // table[5] will have 5 DIFFERENT numbers from 0 to 11
while (chosenPlayersIndex < 5) {
    int
    var = (int) rand() % 11; // var will have values from 0 to 11
    switch (chosenPlayersIndex) {
        case 0:
            table[0] =
                var;
            chosenPlayersIndex++;
        case 1:
            if (var != table[0]) {
                table[1] =
                    var;
                chosenPlayersIndex++;
            }
        case 2:
            if (var != table[0] &&
                var != table[1]) {
                table[2] =
                    var;
                chosenPlayersIndex++;
            }
        case 3:
            if (var != table[0] &&
                var != table[1] &&
                var != table[2]) {
                table[3] =
                    var;
                chosenPlayersIndex++;
            }
        case 4:
            if (var != table[0] &&
                var != table[1] &&
                var != table[2] &&
                var != table[3]) {
                table[4] =
                    var;
                chosenPlayersIndex++;
            }
        default:
            break;
    }
}
```

Έπειτα μέσω τον αντικειμένων **a** και **b** βρίσκουμε τον μέσο όρο της τεχνικής αλλά και της κούρασης των 5 παικτών ώστε να αποφανθούμε για το αποτέλεσμα του κάθε γύρου. Τέλος, η νικητήρια ομάδα παίρνει το έπαθλο της, το οποίο είναι τύπου **FoodSupply** και είναι **supplies**.

```
if (wins1 > wins2) {
    r.teamWins();
    setWinner("Red");
    cout << "Team Competition winner is Team " << getWinner() << endl;
    int
    var = r.getSupplies() + foodAward - > getBonusSupplies();
    r.setSupplies(var);
    return 1;
} else {
    b.teamWins();
    setWinner("Blue");
    cout << "Team Competition winner is Team " << getWinner() << endl;
    int
    var = b.getSupplies() + foodAward - > getBonusSupplies();
    b.setSupplies(var);
    return 0;
}
}
```

3.Ημέρα Διαγωνισμού Ασυλίας

Όπως και προηγουμένως η μέρα ξεκινάει και τελειώνει με τις **MorningRoutine()** και **NightRoutine()** και η πιο σημαντική συνάρτηση της κλάσης **ImmunityCompetition** είναι η συνάρτηση **void compete(Team &team)** η οποία παίρνει ένα argument, είτε **teams[0]**, είτε **teams[1]** καθώς στον διαγωνισμό αυτό αγωνίζεται μόνο η ηττημένη από τον προηγούμενο διαγωνισμό ομάδα. Για τον λόγο αυτό στην **main()** υπάρχει η επιλογή ότι σε περίπτωση που δεν έχει γίνει κάποιος ομαδικός διαγωνισμός δεν υπάρχει η δυνατότητα να τρέξει ο Διαγωνισμός Ασυλίας.

```
if (teamCompetitionWinner == -1) {
    cout << "First press '2' to run Team Competition..." << endl;
    return;
}
```

Ακόμη, η **compete()** δημιουργεί ένα πίνακα **float table[11]** ο οποίος περιέχει το κριτήριο επιτυχίας του διαγωνισμού αυτού. Το κριτήριο εκφράζεται με την σχέση στην 5^{η} σειρά του κώδικα:

```
float table[11];
for (int i = 0; i < 11; i++) {
    float t = team.getPlayers()[i].getTechnique();
    float f = team.getPlayers()[i].getFatigue();
    table[i] = t * 0.75 + (100 - f) * 0.25;
}
```

Τέλος, το έπαθλο αυτού του διαγωνισμού είναι ,προς το παρόν, η αύξηση των **wins** του νικητή:

```
int MAX = 0;
for (int i = 1; i < 11; i++)
    if (table[i] > table[MAX]) MAX = i;
cout << endl << "Player " << team.getPlayers()[MAX].getName() << " of " << team.getColor() <<
" team won the Immunity Competition" << endl << endl;
int wins = team.getPlayers()[MAX].getWins() + 1;
team.getPlayers()[MAX].setWins(wins);
```

4.Ημέρα Διαγωνισμού Δημιουργηκότητας

Η συνάρτηση **void compete(Team &r, Team &b)** δημιουργεί αρχικά ένα πίνακα τύπου Player, **Player table[22]** ο οποίος έχει όλους τους παίκτες του παιχνιδιού και πιο συγκεκριμένο από την θέση 0 έως 10 είναι οι παίκτες της κόκκινης ομάδας και από την θέση 11 έως 21 είναι οι παίκτες της μπλε ομάδας. Στην συνέχεια υλοποιείτε ο κώδικας του διαγράμματος ροής με αξιοσημείωτο σημείο το ότι στο τέλος της compete() ο πίνακας **table[22]** διαγράφεται, γεγονός το οποίο δεν μας ενοχλεί αφού έχουμε δώσει πρώτα το έπαθλο στα αντικείμενα **Team r** και **Team b**.

```
float maxTechnique = -1.0;
int maxIndex = -1;
int playerIndex = 0;
while (playerIndex < 22) {
    float technique = table[playerIndex].getTechnique();
    if (technique > maxTechnique) {
        maxTechnique = technique;
        maxIndex = playerIndex;
    }
    playerIndex++;
}
for (int i = 0; i < 22; i++) {
    if (i < 11) {
        if (table[maxIndex].getName() == r.getPlayers()[i].getName()) {
            float fat = r.getPlayers()[i].getTechnique() + (float) excursionAward - > getTechn
iqueBonus();
            float pop = r.getPlayers()[i].getPopularity() + (float) excursionAward - > getPopu
larityPenalty();
            r.getPlayers()[i].setFatigue(fat);
            r.getPlayers()[i].setPopularity(pop);
            r.getPlayers()[i].status();
            status();
        }
    } else {
        if (table[maxIndex].getName() == b.getPlayers()[i - 11].getName()) {
            float fat = b.getPlayers()[i - 11].getTechnique() + (float) excursionAward - > get
TechniqueBonus();
            float pop = b.getPlayers()[i - 11].getPopularity() + (float) excursionAward - > ge
tPopularityPenalty();
            b.getPlayers()[i - 11].setFatigue(fat);
            b.getPlayers()[i - 11].setPopularity(pop);
            b.getPlayers()[i - 11].status();
            status();
        }
    }
}
cout << "'Player table[22]' is gonna be destroyed...but 'Player* players' still exists !!!" <<
 endl;
```