ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΗΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Master Chef 2018
Μέρος Β΄
Τετάρτη 25/03/2018

Μιχαήλ Μηναδάκης
ΑΕΜ: 8858
Email: thejokergr@hotmail.gr

Άρης-Ελευθέριος Παπαγγέλης
ΑΕΜ: 8883
Email: aris.papagelis@gmail.com

Στέφανος Παπαδάμ
ΑΕΜ: 8885
Email: stefanospapadam@gmail.com

Illustration 1: Ετοιμοπόλεμοι οι παίκτες μας!

Πρόβλημα προς επίλυση

Καλούμαστε να αναπτύξουμε ένα πρόγραμμα, το οποίο να μοντελοποιεί με όσο το δυνατόν πιο ρεαλιστικό τρόπο το τηλεπαιχνίδι MasterChef.

Περιγραφή Υλοποίησης

Κλάση Player

Η συγκεκριμένη κλάση αναλύεται σε δύο αρχεία*: Player.h* και *Player.cpp*

Player.h:

Στο αρχείο αυτό προδιαγράφουμε τις εξής μεθόδους (οι getters και setters υλοποιούνται κιόλας σε αυτό το αρχείο, ως inline συναρτήσεις) :

- *Player()*: συνάρτηση αρχικών συνθηκών χωρίς ορίσματα.
- *Player(string n, string g, string job, int a, int wins, bool candidate)*: συνάρτηση αρχικών συνθηκών με παραμέτρους.
- *~Player()*: συνάρτηση τελικών συνθηκών.
- *void setName(string n)*: συνάρτηση που θέτει τιμή στη μεταβλητή name
- *string getName()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής name.
- *void setGender(string g)*: συνάρτηση που θέτει τιμή στη μεταβλητή gender.
- *string getGender()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής gender.
- *void setProfession(string job)*: συνάρτηση που θέτει τιμή στη μεταβλητή profession.
- *string getProfession ()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής profession.
- *void setAge(int a)*: συνάρτηση που θέτει τιμή στη μεταβλητή age.
- *int getAge ()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής age.
- *void setNumberOfWins(int wins)*: συνάρτηση που θέτει τιμή στη μεταβλητή numberOfWins.
- *int getNumberOfWins()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής numberOfWins.
- *void setSkill (float s)*: συνάρτηση που θέτει τιμή στη μεταβλητή skill.
- *float getSkill ()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής skill.
- *void setFatigue (float f)*: συνάρτηση που θέτει τιμή στη μεταβλητή fatigue.
- *float getFatigue ()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής fatigue.

- *void setPopularity (float pop)*: συνάρτηση που θέτει τιμή στη μεταβλητή popularity.
- *float getPopularity ()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής popularity.
- *void setEliminationCandidate (bool cand)*: συνάρτηση που θέτει τιμή στη μεταβλητή eliminationCandidate.
- *bool getEliminationCandidate ()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής eliminationCandidate.
- *void works()*: υλοποιεί τη δουλειά του παίκτη στο παιχνίδι.
- *void socializes()*: υλοποιεί την κοινωνικοποίηση του παίκτη στο παιχνίδι.
- *void sleeps()*: υλοποιεί την διαδικασία στην οποία ο παίκτης κοιμάται για να ξεκουραστεί.
- *void practices()*: υλοποιεί την εξάσκηση του παίκτη.
- *void endOfWeek()*: βοηθητική συνάρτηση η οποία επιλέγει τυχαία αν ο παίκτης στο τέλος της εβδομάδας θα κοιμηθεί ή θα εξασκηθεί.
- *void participates()*: υλοποιεί την διαδικασία κατά την οποία ο παίκτης λαμβάνει μέρος στους διαγωνισμούς.

Player.cpp:

Σε αυτό το αρχείο γίνεται η υλοποίηση των παρακάτω συναρτήσεων:

- *Player()*: συνάρτηση αρχικών συνθηκών χωρίς ορίσματα η οποία μηδενίζει τις μεταβλητές τύπου int και float (age,numberOfWins,skill,fatigue,popularity), θέτει false τη μεταβλητή τύπου bool (eliminationCandidate) και θέτει τον κενό χαρακτήρα στις μεταβλητές τύπου string (name,gender,profession).

- *Player(string n, string g, string job, int a, int wins, bool candidate)*: συνάρτηση αρχικών συνθηκών με ορίσματα τα οποία αποδίδονται στις αντίστοιχες μεταβλητές. Τις μεταβλητές fatigue και popularity τις αρχικοποιούμε με τις τιμές 0 και 50 αντίστοιχα διότι με αυτές τις τιμές ξεκινάει το παιχνίδι κάθε παίκτης και η μεταβλητή skill αρχικοποιείται μέσω της έκφρασης (float) (rand()%81) η οποία δίνει στη μεταβλητή μία τυχαία τιμή από 0 έως 80.

- *~Player()*: συνάρτηση τελικών συνθηκών η οποία εμφανίζει κατάλληλο μήνυμα όταν το αντικείμενο καταστρέφεται. Το μήνυμα που εκτυπώνει βέβαια είναι κενό για λόγους αισθητικής.

- *works()*: υλοποιεί τη δουλειά του παίκτη στο παιχνίδι, δηλαδή η κούραση του παίκτη αυξάνεται σε ένα τυχαίο ποσοστό από 20% έως 40% (fatigue+=rand() %21 + 20) και η τεχνική του κατάρτιση αυξάνεται κατά 5% ποσοστιαία (skill+=5*skill/100). Στη συνέχεια με δύο εντολές if ελέγχουμε τις μεταβλητές

fatigue και skill και σε περίπτωση που έχουν ξεπεράσει το 100 τις θέτουμε ίσες με 100.

- *socializes()*: υλοποιεί την κοινωνικοποίηση του παίκτη στο παιχνίδι, δηλαδή μεταβάλλει σε ένα τυχαίο ποσοστό μεταξύ -10% και +10% ποσοστιαία την μεταβλητή popularity (popularity += ((rand()%21)-10)*popularity/100) και ελέγχει στη συνέχεια τη μεταβλητή αν έχει έγκυρες τιμές. Σε περίπτωση που η μεταβλητή πάρει τιμή μεγαλύτερη από 100 ή μικρότερη από 0 τότε τη θέτει 100 και 0 αντίστοιχα.

- *sleeps()*: υλοποιεί τη διαδικασία κατά την οποία ο παίκτης κοιμάται για να ξεκουραστεί, όπου θέτουμε τη μεταβλητή fatigue ίση με 0.

- *practices()*: υλοποιεί την εξάσκηση του παίκτη στο παιχνίδι, δηλαδή αυξάνουμε τη μεταβλητή skill κατά 5% απόλυτα (skill+=5) και ελέγχουμε αν έχει πάρει τιμή μεγαλύτερη από 100 οπότε και τη θέτουμε ίση με 100.

- *endOfWeek()*: υλοποιεί μία βοηθητική συνάρτηση η οποία επιλέγει τυχαία στο τέλος της εβδομάδας αν ο παίκτης θα ξεκουραστεί ή θα εξασκηθεί. Ορίζουμε μια μεταβλητή choice η οποία παίρνει μία τυχαία τιμή 0 ή 1 (int choice=rand()%2) και σε περίπτωση που η τιμή αυτή είναι 0, τότε καλείται η συνάρτηση sleeps(), δηλαδή ο παίκτης κοιμάται για να ξεκουραστεί, αλλιώς καλείται η συνάρτηση practices(), δηλαδη ο παίκτης εξασκείται.

- *participates()*: υλοποιεί τη συμμετοχή του παίκτη στους διαγωνισμούς όπου η κούραση του αυξάνεται κατά 10% έως 20% ποσοστιαία (fatigue+=((rand()%11)+10)*fatigue/100).

Κλάση Team

Η συγκεκριμένη κλάση αναλύεται σε δύο αρχεία*: Team.h* και Team.*cpp*

Team.h:

Στο αρχείο αυτό προδιαγράφουμε τις εξής μεθόδους (οι getters και setters υλοποιούνται κιόλας σε αυτό το αρχείο, ως inline συναρτήσεις, με εξαίρεση αυτους του playerList[11]) :

- *Team()*: συνάρτηση αρχικών συνθηκών χωρίς ορίσματα.
- *Team(string c, int wins, int members, int supplies)*: συνάρτηση αρχικών συνθηκών με παραμέτρους.

- *~Team()*: συνάρτηση τελικών συνθηκών.
- *void setColour(string c)*: συνάρτηση που θέτει τιμή στη μεταβλητή colour.
- *string getColour()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής colour.
- *void setNumberOfWins(int wins)*: συνάρτηση που θέτει τιμή στη μεταβλητή numberOfWins.
- *int getNumberOfWins()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής numberOfWins.
- *void setNumberOfMembers(int members)*: συνάρτηση που θέτει τιμή στη μεταβλητή numberOfMembers.
- *int getNumberOfMembers()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής numberOfMembers.
- *void setNumberOfSupplies(int supplies)*: συνάρτηση που θέτει τιμή στη μεταβλητή numberOfSupplies.
- *int getNumberOfSupplies()*: συνάρτηση που επιστρέφει την τιμή της μεταβλητής numberOfSupplies.
- *void setPlayerList(int i, string n, string g, string job, int a, int wins ,bool candidate)*: συνάρτηση που θέτει τιμές στον πίνακα playerList[11].
- *Player getPlayerList(int j)*: συνάρτηση που επιστρέφει το αντικείμενο τύπου Player που βρίσκεται στη θέση j του πίνακα playerList[11].

Team.cpp:

Σε αυτό το αρχείο γίνεται η υλοποίηση των παρακάτω συναρτήσεων:

- *Team()*: συνάρτηση αρχικών συνθηκών χωρίς ορίσματα η οποία μηδενίζει τις μεταβλητές τύπου int(numberOfWins, numberOfMembers, numberOfSupplies), θέτει τον κενό χαρακτήρα στη μεταβλητή τύπου string (colour) και με μία εντολή επανάληψης for θέτει σε κάθε θέση του πίνακα playerList ένα παίκτη καλώντας την κενή συνάρτηση αρχικών συνθηκών Player() .

- *Team(string c, int wins, int members, int supplies)*: συνάρτηση αρχικών συνθηκών με ορίσματα τα οποία αποδίδονται στις αντίστοιχες μεταβλητές και ομοίως με μία εντολή επανάληψης for κάθε θέση του πίνακα αρχικοποιείται με ένα παίκτη ο οποίος δημιουργείται από την κενή συνάρτηση αρχικών συνθηκών Player() .

- *~Team()*: συνάρτηση τελικών συνθηκών η οποία εμφανίζει κατάλληλο μήνυμα όταν το αντικείμενο καταστρέφεται.Το μήνυμα που εκτυπώνει βέβαια είναι κενό για λόγους αισθητικής.

- *setPlayerList(int i, string n, string g, string job, int a, int wins, bool candidate)*: Η συνάρτηση αυτή αρχικοποιεί τη θέση i του πίνακα playerList καλώντας τη συνάρτηση αρχικών συνθηκών Player(string n, string g, string job, int a, int wins, bool candidate).Τα ορίσματα της συνάρτησης δίνονται ως είσοδος από το χρήστη στη main, ενώ στη συνέχεια αυξάνουμε και τον αριθμό των μελών της ομάδας κατά 1 (numberOfMembers +=1).

- *getPlayerList(int j)*: Στη συνάρτηση αυτή δίνουμε ως όρισμα τη θέση j της playerList απο τη συνάρτηση main, και μας επιστρέφει το αντικείμενο τύπου Player που βρίσκεται σε αυτή την θέση.

MAIN

main.cpp:

Σε αυτό το αρχείο γίνεται η υλοποίηση των παρακάτω συναρτήσεων:

- *Team playerInsert(Team T)*: Αυτή η συνάρτηση δέχεται όρισμα τύπου Team και διαβάζει από τον χρήστη τα στοιχεία του πάικτη που θέλει να εισάγει, λαμβάνοντας υπόψιν τους περιορισμούς της κάθε μεταβλητής. Ύστερα εισάγει στην λίστα των παικτών της Team, στην θέση που δείχνει η getNumberOfMembers(), αντικείμενο τύπου Player με ορίσματα τα εισαχθέντα από τον χρήστη στοιχεία. Τέλος, επιστρέφει το Team.

- *void teamStatus(Team T)*: Αυτή η συνάρτηση δέχεται όρισμα τύπου Team και εκτυπώνει όλες τις μεταβλητές του Team. Επιπλέον, εκτυπώνει όλα τα ονόματα των παικτών του Team μαζί με τα αντιστοίχα Index, αλλά μόνο σε περίπτωση που υπάρχουν πάικτες στην ομάδα.

- *void playerStatus(Team T)*: Αυτή η συνάρτηση δέχεται όρισμα τύπου Team και, αν η ομάδα έχει παίκτες, ρωτάει τον χρήστη αν θέλει να ψάξει κατά Name ή κατά Index.

  Σε περίπτωση αναζήτησης κατά όνομα ζητέιται από το χρήστη να εισάγει το όνομα που αναζητά και, σε περίπτωση που το όνομα υπάρχει σε αυτήν την ομάδα, εκτυπώνει τα στοιχεία του παίκτη με αυτό το όνομα. Χρησιμοποιούμε γραμμική αναζήτηση για την εύρεση του ονόματος, με πολυπλοκότητα O(n), όπου n=T.getNumberOfMembers().

Θα μπορούσαμε να ταξινομούμε αλφαβητικά τον πίνακα μετά από κάθε εισαγωγή παίκτη και ύστερα να εφαρμόζαμε δυαδική αναζήτηση για την εύρεση του κάθε παίκτη, πολυπλοκότητας O(logn)<O(n), ούτως ώστε να κάνουμε τον αλγόριθμό μας ταχύτερο. Βέβαια, αυτό δεν έχει πολύ νόημα για τόσο μικρό αριθμό παικτών (μέχρι 11), αλλά για μεγάλο πλήθος στοιχείων θα ήταν η ενδεδειγμένη λύση.

Σε περίπτωση αναζήτησης κατά Index, το οποίο θα πρέπει να είναι έγκυρο, εκτυπώνει τα στοιχεία του παίκτη που αντιστοιχεί σε αυτό το Index της λίστας παικτών της ομάδας.

- *int main()*: Στη βασική συνάρτηση του προγράμματός μας αρχικά αρχικοποιούμε τις δύο ομάδες. Ύστερα ζητείται από τον χρήστη να εισάγει μία από τις τέσσερις διαθέσιμες επιλογές.

  Σε περίπτωση που ο χρήστης επιλέξει να εισάγει έναν παίκτη σε μία ομάδα (επιλογή 1), του ζητείται η ομάδα (κόκκινη ή μπλε) και ύστερα καλείται η συνάρτηση playerInsert με όρισμα την ομάδα που επιλέχθηκε.

```
Choose one of the following:
1.Insert a player in a team.
2.Display team status.
3.Display player status.
4.Exit program.
1

In which team do you want to put the player? (Red or Blue):
Red
Enter player's name:
Aris
Is the player Male or Female?:
Male
Enter player's profession:
Engineer
Enter player's age:
21
Enter player's number of wins:
0
Is the player a candidate for elimination?(Yes or No):
No

Choose one of the following:
1.Insert a player in a team.
2.Display team status.
3.Display player status.
4.Exit program.
1

In which team do you want to put the player? (Red or Blue):
Red
Enter player's name:
Mik
Is the player Male or Female?:
Female
Enter player's profession:
Gamer
Enter player's age:
21
Enter player's number of wins:
0
Is the player a candidate for elimination?(Yes or No):
Yes

Choose one of the following:
1.Insert a player in a team.
2.Display team status.
3.Display player status.
4.Exit program.
1

In which team do you want to put the player? (Red or Blue):
Red
Enter player's name:
Stef
Is the player Male or Female?:
Female
Enter player's profession:
Bodkoc
Enter player's age:
21
Enter player's number of wins:
0
Is the player a candidate for elimination?(Yes or No):
Yes

Choose one of the following:
1.Insert a player in a team.
2.Display team status.
3.Display player status.
4.Exit program.
```

Illustration 2: Εισαγωγή στοιχείων.

Σε περίπτωση που ο χρήστης επιλέξει να δει την κατάσταση της ομάδας (επιλογή 2), του ζητείται η ομάδα (κόκκινη ή μπλε) και ύστερα καλείται η συνάρτηση teamStatus με όρισμα την ομάδα που επιλέχθηκε.

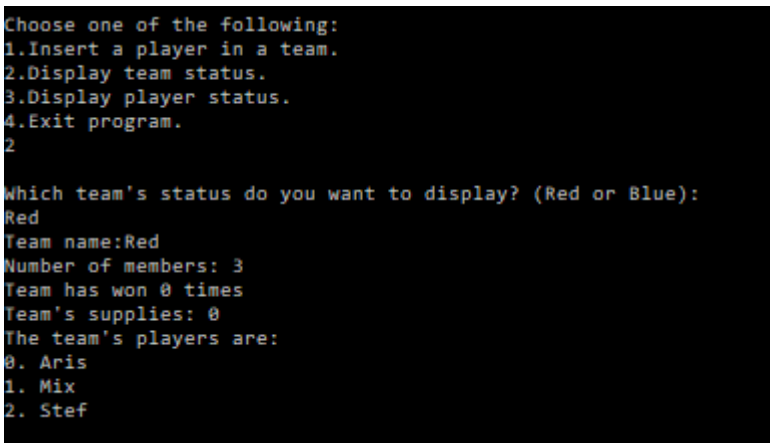

Illustration 3: Προβολή κατάστασης ομάδας.

Σε περίπτωση που ο χρήστης επιλέξει να δει την κατάσταση ενός παίκτη (επιλογή 3), του ζητείται η ομάδα (κόκκινη ή μπλε) στην οποία ανήκει ο παίκτης και ύστερα καλείται η συνάρτηση playerStatus με όρισμα την ομάδα που επιλέχθηκε.

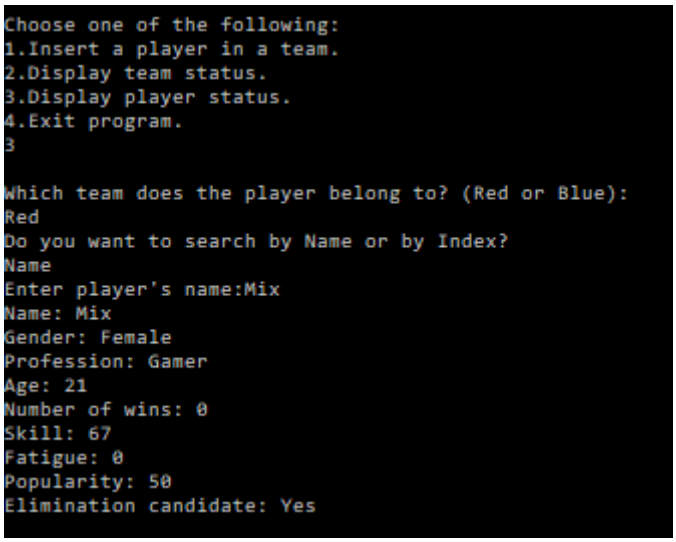

Illustration 4: Αναζήτηση κατά όνομα.

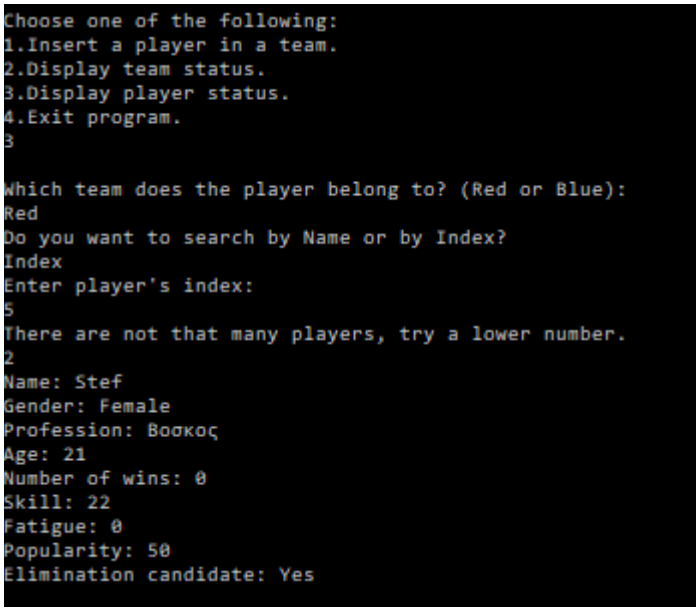

Illustration 5: Αναζήτηση κατά δείκτη.

Τέλος, αν ο χρήστης επιλέξει την έξοδο από το πρόγραμμα (επιλογή 4), το πρόγραμμα τερματίζει.

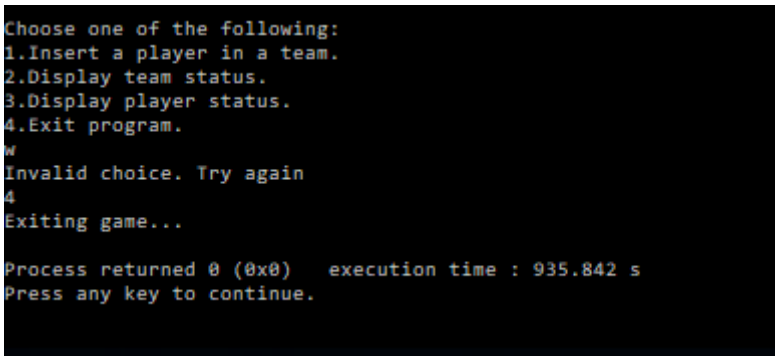

Illustration 6: Έξοδος από το πρόγραμμα.

Υ.Γ. Δεν αποτυπώσαμε κάθε δυνατή λειτουργία στην αναφορά καθαρά για λόγους οικονομίας χώρου. Πάντως, το πρόγραμμα μας λειτουργεί σωστά σε ΚΑΘΕ περίπτωση, όσο λανθασμένη είσοδο και να δώσει ο χρήστης. Εξάλλου:

Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots. So far, the Universe is winning. - *Rick Cook*