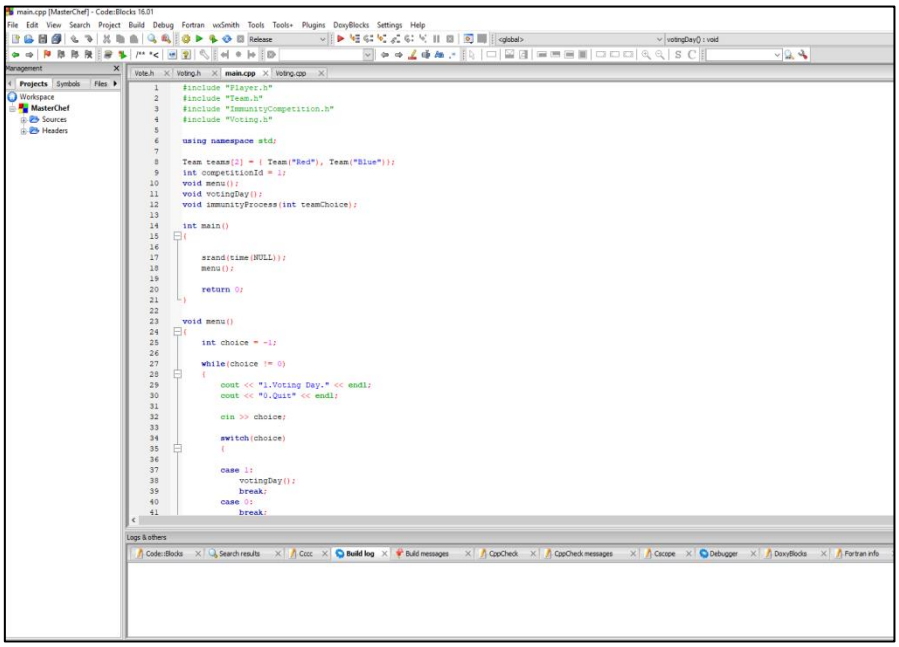
4η Εργασία OOP:
MasterChef Part IV: A New Chef

Βέλλιος Γεώργιος-Σεραφείμ
AEM: 9471
velliosg@ece.auth.gr

Μανούσος Διαγόρας
AEM: 9554
dmanouso@ece.auth.gr



<u>Περιγραφή του προβλήματος:</u>

Καλούμαστε να υλοποιήσουμε την ημέρα αποχώρησης στο διαγωνισμό του MasterChef. Αυτή περιλαμβάνει την αποχώρηση ενός παίκτη κατόπιν ψηφοφορίας από όλους τους παίκτες της ομάδας που έχασε στον ομαδικό διαγωνισμό της προηγούμενης ημέρας. Για την επίτευξη αυτού του στόχου δημιουργούμε τις ***κλάσεις Vote και Voting.***

<u>Επίλυση του προβλήματος:</u>

Αρχικά δημιουργήθηκε *η κλάση Vote*, η οποία θα χρησιμοποιηθεί *από τη μέθοδο votingProcess* και από *τον vector votes της κλάσης Voting. Κ*ατόπιν δημιουργήθηκε η *κλάση Voting*. Περισσότερες πληροφορίες για τις δύο κλάσεις υπάρχουν με τη μορφή σχολίων στα αντίστοιχα **αρχεία .h** και **.cpp.** Επιλεκτικά αναφέρεται ότι:

- Η *κλάση Vote* έχει δύο μεταβλητές τύπου *string,*την *voted* (ποιος ψηφίστηκε) και την *reason* (για ποιον λόγο ψηφίστηκε)
- Οι εντολές όλων των μεθόδων είναι inlineκαι έχουν γραφτεί στο αρχείο Vote.h
- Οι μεταβλητές της *κλάσης Voting*,όπως και η μέθοδός της , είναι static. Γι'αυτό το λόγο οι μεταβλητές της(*vector votes, map results*) ορίζονται στο αρχείο Voting.cpp.
- Οι μεταβλητές της *κλάσης Voting*,όπως και η μέθοδός της , είναι public,οπότε οι μεταβλητές δεν χρειάζονται *getters και setters*.

Όλη η διαδικασία της ψηφοφορίας γίνεται στην συνάρτηση *votingProcess(Team&team)* της *κλάσης Voting*. Ο τρόπος λειτουργίας της περιγράφεται παρακάτω:

Αρχικά, κρίνεται απαραίτητο να αναφερθεί ο τρόπος με τον οποίο ψηφίζει ο κάθε παίκτης. Έτσι, η επιλογή του άλλου παίκτη που ψηφίζει προς αποχώρηση κάποιος γίνεται με βάση τέσσερα κριτήρια και κάθε φορά επιλέγεται τυχαία να γίνει χρήση ενός από αυτά. Αυτά είναι:

1. Ψήφιση προς αποχώρηση του παίκτη με την μεγαλύτερη τεχνική κατάρτιση
2. Ψήφιση προς αποχώρηση του παίκτη με την μεγαλύτερη δημοφιλία
3. Ψήφιση προς αποχώρηση του παίκτη με την λιγότερη κούραση
4. Ψήφιση προς αποχώρηση του παίκτη με τον μεγαλύτερο συνδυασμό τεχνικής κατάρτισης, δημοφιλίας και μη κούρασης

Τα παραπάνω κριτήρια προσπαθούν να προσομοιώσουν πως θα αποφάσιζε και στην πραγματική ζωή να ψηφίσει κάποιος ένα αντίπαλο για αποχώρηση (θα ψήφιζε αυτόν που αποτελούσε τον μεγαλύτερο κίνδυνο).

Επίσης, έχει ληφθεί μέριμνα και για την ειδική περίπτωση που έχουν μείνει δύο παίκτες στην ομάδα, ώστε να αποχωρεί αυτός που δεν έχει ασυλία. Αυτός ο έλεγχος επιτυγχάνεται πριν αρχίσει η διαδικασία με την εντολή **if(team.getNumberOfPlayers() != 2**) που συνεχίζει την κανονική λειτουργία του

προγράμματος αν δεν είναι δύο οι παίκτες της ομάδας. Αλλιώς, εκτελείται το παρακάτω μπλοκ:

```
else{
    for(int a=0; a<11; a++){
        if(team.getPlayers()[a].getAge() != 0 && team.getPlayers()[a].getImmunity()==false){
            Vote vote(team.getPlayers()[a], "Akraia periptwsi. fevgei opoios den exei Immunity");
            Voting::votes.push_back(vote);
        }
```

Εικόνα 1: μπλοκ εντολών όταν έχουμε δύο εναπομείναντες παίκτες

Που ελέγχει ποιος από τους παίκτες που υπάρχει δεν έχει ασυλία, και τοποθετεί στον *vector votes* μία ψήφο με το όνομα του.

<u>Κανονική λειτουργία της συνάρτησης:</u>

Αρχικά ορίζονται οι μεταβλητές *int random* (τυχαία επιλογή του ενός από τα τέσσερα κριτήρια με το οποίο θα ψηφίσει ο παίκτης*), int ptexniki, pdimofilia, pkourasi, psyndyasmos* (θέση του παίκτη που ικανοποιεί τα κριτήρια που παρουσιάστηκαν παραπάνω **ΚΑΙ** δεν είναι ίδιος με τον παίκτη που ψηφίζει, υπάρχει και δεν έχει ασυλία). Με την παρακάτω επανάληψη και έλεγχο :

```
for(int i=0; i<11; i++){
    if(team.getPlayers()[i].getAge() != 0){
        for(int j=1; j<=team.getPlayers()[i].getVotes(); j++){
```

Εικόνα 2: Έλεγχος και επανάληψη

Επαναλαμβάνεται η τοποθέτηση ψήφου για όσους παίκτες υπάρχουν (γραμμές 1 και 2). Επίσης, κάθε παίκτες ψηφίζει τόσες φορές όσες οι ψήφοι που έχει (γραμμή 3). Στην συνέχεια αρχικοποιούνται οι μεταβλητές *float maxTexniki, maxDimofilia, minKourasi, maxSyndyasmos* που κρατάνε την μέγιστη κατάρτιση, μέγιστη δημοφιλία, μικρότερη κούραση και μεγαλύτερο συνδυασμό αντίστοιχα, που έχουν βρεθεί μέχρι στιγμής. Έτσι, αρχικά παίρνουν τιμές έξω από τα όποια τους ώστε κάθε υπάρχουσα τιμή να είναι μεγαλύτερη ή μικρότερη (ανάλογα αν είναι max ή min) από την αρχική και οι θέσεις των παικτών που τις ικανοποιούν γίνονται -1. Στην συνέχεια γίνεται έλεγχος για όσους παίκτες ικανοποιούν τα κριτήρια ψήφου με βάσει τις προϋποθέσεις που μας δόθηκαν ,δηλαδή όχι ίδιος παίκτης, να υπάρχει και να μην έχει ασυλία. Αν γίνεται αυτό, τότε τα ποσοστά τους μεταβιβάζονται στις αντίστοιχες min και max μεταβλητές και η θέση τους στις p…μεταβλητές.

Αφού έχουν βρεθεί οι παίκτες, έφτασε η ώρα να ψηφίσει ο παίκτης ποιον θέλει για αποχώρηση. Αυτό γίνεται με την τυχαία επιλογή ενός από τα 4 παραπάνω κριτήρια και μίας *switch* που δημιουργεί μία ψήφο *Vote vote* με την *μεταβλητή voted* να ισούται με το όνομα του κατάλληλου παίκτη και την *μεταβλητή reason* να ισούται με την κατάλληλη αιτιολογία. Τέλος, και εκτός της *switch*, η ψήφος τοποθετείται στον *vector votes της* κλάσης Voting. Η παραπάνω διαδικασία αντιστοιχεί στον εξής κώδικα:

```
        random=rand()%4;
        Vote vote;
        switch(random){
    case 0:
        vote.setVoted(team.getPlayers()[ptexniki].getName());
        vote.setReason("Megaliteri Texniki Katartisi");
        break;
    case 1:
        vote.setVoted(team.getPlayers()[pdimofilia].getName());
        vote.setReason("Megaliteri Dimofilia");
        break;
    case 2:
        vote.setVoted(team.getPlayers()[pkourasi].getName());
        vote.setReason("Ligoteri kourasi");
        break;
    case 3:
        vote.setVoted(team.getPlayers()[psyndyasmos].getName());
        vote.setReason("Megalitero Pososto katartisis, dimofilias kai mi kourasis");
        break;
    default:
        break;
        }//end switch με ποιο κριτήριο θα αποφασίσει ο παίκτης
        Voting::votes.push_back(vote);
```

Εικόνα 3: Δημιουργία ψήφου με κατάλληλο κριτήριο και τοποθέτηση στον vector votes

Μετά εκτυπώνονται όλοι οι ψήφοι που κατατέθηκαν με την εκτύπωση του *vector votes* με τον εξής κώδικα:

```
vector <Vote>::iterator pv=votes.begin();
while(pv != votes.end()){
  pv->status();
  pv++;
}
```

Εικόνα 4: Εκτύπωση του vector votes

Κατόπιν, ακολουθεί η δημιουργία του *map results* που περιέχει τα ονόματα αυτών που ψηφίστηκαν και τον αριθμό των ψήφων τους. Για την δημιουργία του ακολουθήθηκε ο τρόπος που περιγράφεται στον ψευδοκώδικα. Οι εντολές είναι οι εξής:

```
pv= votes.begin();
map<string, int>::iterator pm;
while(pv != votes.end()){
    pm= results.find(pv->getVoted());
    if(pm != results.end()){
    pm->second+=1;
    }
    else{
        results.insert(pair<string, int>(pv->getVoted(), 0));
        pm= results.find(pv->getVoted());
        pm->second+=1;
    }

    pv++;
}//end while, μετρηθηκαν ολοι οι ψήφοι
```

Εικόνα 5:Δημιουργία του map results

Έτσι, ο *iterator* του *vector votes pv* παίρνει τη τιμή *votes.begin()* και γίνεται μία *while* για όσες ψήφους υπάρχουν. Ο *iterator* του *map results pm* δείχνει την θέση του στοιχείου του *map* που έχει κλειδί το όνομα του ψήφου που δείχνει ο *pv*. Αν υπάρχει ήδη στον χάρτη το κλειδί αυτό, αυξάνει την τιμή του κατά 1. Αν δεν υπάρχει, τοποθετεί στον χάρτη ένα ζεύγος με κλειδί το όνομα του *voted* παίκτη που δείχνει ο *pv* και τιμή 0 (ψήφοι) και την αυξάνει μετά κατά 1.

Εκτυπώνονται τα περιεχόμενα του map results με τον κώδικα:

```
pm=results.begin();
cout<<"\t\tResults:\n";
while(pm != results.end()){
    cout<<"Name: "<<pm->first<<endl;
    cout<<"Votes: "<<pm->second<<endl<<endl;
    pm++;
}
```

Εικόνα 6: Εκτύπωση των περιεχομένων του map results

Ύστερα, βρίσκονται στον χάρτη οι δύο παίκτες με τις περισσότερες ψήφους. Επειδή η εύρεση γίνεται με βάση τις τιμές και βρίσκουμε το όνομα των παικτών, πρέπει να βρούμε σε ποια θέση στον πίνακα των παικτών αντιστοιχεί αυτό το όνομα. Για αυτό γίνεται ένας επιπλέον έλεγχος μέσω μιας επανάληψης *for* και του ελέγχου **if(team.getPlayers()[i].getName() == nameMax…)** και τοποθετείται στις μεταβλητές *technique1, technique2, pos1, pos2* η τεχνική και η θέση στον πίνακα του κάθε παίκτη.

Εκτυπώνονται τα ονόματα των δύο παικτών με τις περισσότερες ψήφους και κατόπιν συγκρίνεται η τεχνική τους κατάρτιση. Αποχωρεί αυτός με την μικρότερη τεχνική κατάρτιση, ή, αν έχουν την ίδια, γίνεται τυχαία επιλογή ανάμεσα τους. Εκτυπώνεται το όνομα του παίκτη που αποχωρεί και αυτός διαγράφεται από την ομάδα. Αυτά γίνονται στο εξής τμήμα κώδικα:

Εκτύπωση των δύο υποψηφίων

Σύγκριση της τεχνικής κατάρτισης των δύο παικτών

```
cout<<"Candidates for leaving:\n";
cout<<team.getPlayers()[pos1].getName()<<endl;
cout<<team.getPlayers()[pos2].getName()<<endl<<endl;
if(team.getPlayers()[pos2].getAge()== 0){
   cout<<team.getPlayers()[pos1].getName()<<" is leaving."<<endl<<endl;
    team.removePlayer(team.getPlayers()[pos1].getName());
}
else if(technique1 > technique2){
    cout<<team.getPlayers()[pos2].getName()<<" is leaving."<<endl<<endl;
    team.removePlayer(team.getPlayers()[pos2].getName());
}
else if(technique1 < technique2){
    cout<<team.getPlayers()[pos1].getName()<<" is leaving."<<endl<<endl;
    team.removePlayer(team.getPlayers()[pos1].getName());
}
else{
    int randl= rand()%2;
    switch(randl){
    case 0:
      cout<<team.getPlayers()[pos1].getName()<<" is leaving."<<endl<<endl;
        team.removePlayer(team.getPlayers()[pos1].getName());
        break;
    case 1:
        cout<<team.getPlayers()[pos2].getName()<<" is leaving."<<endl<<endl;
        team.removePlayer(team.getPlayers()[pos2].getName());
        break;
    default:
        break;
    }
}
```

Εικόνα 7: Εκτύπωση των δύο υποψηφίων και σύγκριση της τεχνικής κατάρτισης των δύο παικτών

Τέλος, καθαρίζονται οι δυναμικές δομές *vector votes* και *map results* και τίθεται η *μεταβλητή Immunity* του παίκτη που την είχε κερδίσει πιο πριν από true σε false.

Ο έλεγχος ορθότητας έγινε με εκτέλεση του προγράμματος για επιβεβαίωση πως όλα λειτουργούν ως πρέπει και μερικές *cout* σε ένα σημείο που αρχικά υπήρχε κάποιο πρόβλημα .