**Preliminary Project**

**Keyword: Olympiad**

Name: Kolenda Ekaterina

ID: 323811

**Introduction**

This is a program for the Olympics (Olympic Games). Manager enter data about Competitions, Athletes and Countries. Program depending on the result and type of competition, distributes medals and maintains a team (Country) standing, create and show leaderboards.

**Class introduction**

**Class Olympiad**

This class contain Countries, Competitions, Athletes and data about Olympiad.

**Method function:**

AddCountry: checks for uniqueness using FindCountry and call constructor Country with data from parameters.

AddCompetition: checks for uniqueness using FindCopetition and call constructor Competition with data from parameters.

AddAthlete: checks for uniqueness using FindAthlete and call constructor Athlete with data from parameters.

PrintCountriesLeaderboard: Print leaderboard until number from parameter using Country::PrintCountry, sorted by Country::operator<.

PrintCountres: Print all Countries that fit the filter using Countries::PrintCountry.

PrintListOfCompetitions: Print all Competitions that fit the filter using Competition::PrintCompetition.

PrintListOfAthletes: Print all Athletes that fit the filter using Athlete::PrintAthlete.

FindCountry: Search Country by name and return iterator on it or iterator on the end.

FindCompetition: Search Competition by name and return iterator on it or iterator on the end.

FindAthlete: Search Athlete by name\_surname and return iterator on it or iterator on the end.

SaveToFile: Save All data about Olympics using Country::SaveToFile, Athlete::SaveToFile and Competition::SaveToFile.

LoadFromFile: Read data about Olympics using Country::LoadFromFile, Athlete::LoadFromFile and Competition::LoadFromFile.

class Olympiad

{

private:

    string name;

    string description;

public:

    list<Country> countries;

    list<Competition> competitions;

    list<Athlete> athletes;

    void AddCountry(const string \*name);

    void AddCompetition(const string \*name, const string \*description, const int \*type\_of\_result, const bool first\_place\_is\_smaller);

    void AddAthlete(const string name, const string surname, const Country \*country, const int age, const int weight, const bool gender, const bool disqualification);

    void PrintCountriesLeaderboard(const int number = 10);

    void PrintCountries(const string filter = "");

    void PrintListOfCompetitions(const string filter = "");

    void PrintListOfAthletes(const string filter = "");

    vector<Country>::iterator FindCountry(const string \*name);

    vector<Competition>::iterator FindCompetition(const string \*name);

    vector<Athlete>::iterator FindAthlete(const string \*name\_surname);

    Olympiad();

    Olympiad(const string \*name, const string \*description);

    void SaveToFile(const string fileName = "olympiad.txt");

    void LoadFromFile(const string fileName = "olympiad.txt");

};

**Class Country**

This class contain name, number of gold, silver and bronze and pointer to Athletes which stood for this country.

**Method function:**

AddAthlete: Add Athlete to vector.

AddMedal: Depends on const in medal (1- gold, 2-silver, 3-bronze) add one medal.

PrintAthletes: Print all Athletes that fit the filter from vector using Athlete::PrintAthlete.

PrintCountry: Print data about country.

Operator<: Compare number of medals and return True or False.

SaveToFile: Write data about Country in file.

LoadFromFile: Read from line from file data about Country.

class Country

{

private:

    string name;

    vector<Athlete \*> athletes;

    int gold\_medal;

    int silver\_medal;

    int bronze\_medal;

    list<Athlete> \*all\_athletes;

public:

    Country();

    Country(const string \*name);

    void AddAthlete(const Athlete \*athlete);

    void AddMedal(const int \*medal);

    void PrintAthletes(const string filter = "");

    void PrintCountry();

    bool operator<(const Country &other);

    void SaveToFile(ofstream &file);

    void LoadFromFile(istringstream &line);

};

**Class Competition**

This class contain name, description Athletes in set, type\_of\_result ( for example, 1- time, 2 – distance, 3 – points and etc.), first\_place\_is\_smaller (True sort Athletes to increase, False – decrease), finished.

**Method function:**

AddAthlete:Add Athlete to competition and add result using Athlete::AddCompetition.

PrintCompetition: Print data about Competition.

PrintAthletes: Call for all Athletes that fit the filter from set Athlete::PrintAthlete.

CheckResult: Check result on uniqueness and return True or False.

IsFinished: Retrunt finished.

FinishCompetition: Change finished to True and add medals to countries.

SaveToFile: Write data about Competition in file.

LoadFromFile: Read from line from file data about Competition.

class Competition

{

private:

    string name;

    string description;

    list<Athlete \*> athletes;

    int type\_of\_result = 0;

    bool first\_place\_is\_smaller;

    bool finished;

public:

    Competition(const string \*name, const string \*description, const int \*type\_of\_result, const bool first\_place\_is\_smaller, vector<Country> \*countries);

    void AddAthlete(const Athlete \*athlete, const float \*result);

    void PrintCompetition();

    void PrintAthletes(const string filter = "");

    bool CheckResult(const float \*result);

    bool IsFinished();

    void FinishCompetition();

    void SaveToFile(ofstream &file);

    void LoadFromFile(istringstream &line);

};

**Class Athlete**

This class contain data about Athlete.

pair <Competition\*,pair <int,float>> results;

contain pointer to competition and result, which contain type\_of\_result and value.

**Method function:**

PrintAthlete: Print data about Athlete.

AddCompetition: Add competition with result to vector

GetResult: Return result of certain competition.

UpdateAthlete: Updata data using parameters.

Compare: Compare result of certain competition and return 1, 0 or -1.

SaveToFile: Write data about Athlete in file.

LoadFromFile: Read from line from file data about Athlete.

class Athlete

{

private:

    string name;

    string surname;

    Country \*country;

    int age;

    int height;

    int weight;

    list <pair<Competition \*, pair<int, float>>> results;

    bool gender;

    bool disqualification;

public:

    Athlete(const string \*name, const string \*surname, const Country \*country, const int \*age, const int \*weight, const bool \*gender, const bool \*disqualification);

    void PrintAthlete();

    void AddCompetition(const Competition \*competition, const int \*type\_of\_result, const float \*result);

    float GetResult(const Competition \*competition);

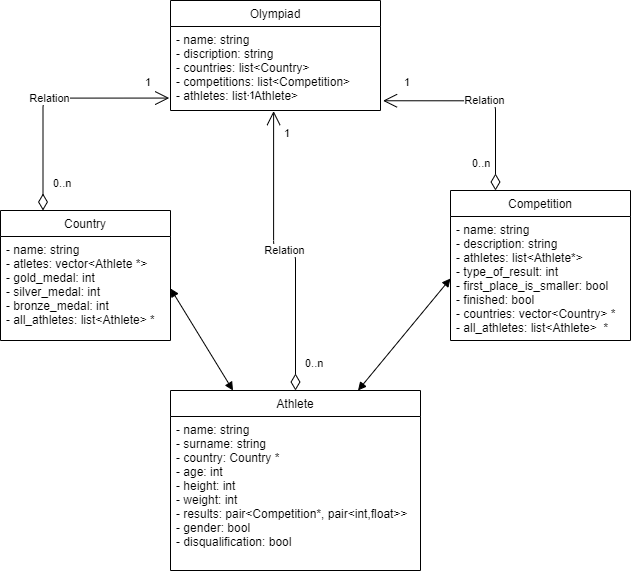
    void UpdateAthlete(const string \*name = nullptr, const string \*surname = nullptr, const Country \*country = nullptr, const int \*age = nullptr, const int \*weight = nullptr, const bool \*gender = nullptr, const bool \*disqualification = nullptr);

    int Compair(const Athlete &other, const Competition \*competition);

    void SaveToFile(ofstream &file);

    void LoadFromFile(istringstream &line);

};

**Mapping**

**Testing**

1.Try to add Country and Competition with same name.

2.Try to call ReadCompetitionData more than one time.

3.Try to add two Athletes with same result.

4.Try to call interface of Competition after end of competition.

5.Try to enter incorrect choice in Interface in Competition and Olympiad.

6.Test Read From File and Write To File.

7.Test Filters in Print methods.

int main(){

    Olympiad olympiad;

    olympiad.AddCountry(&string("Russia"));

    olympiad.AddCountry(&string("USA"));

    olympiad.AddCountry(&string("Germany"));

    olympiad.AddCountry(&string("France"));

    if(olympiad.FindCountry(&string("Russia")) == olympiad.countries.end()){

        cout << "ERROR:Russia is not in the list" << endl;

    }

    if(olympiad.FindCountry(&string("USA")) == olympiad.countries.end()){

        cout << "ERROR:USA is not in the list" << endl;

    }

    if(olympiad.FindCountry(&string("Germany")) == olympiad.countries.end()){

        cout << "ERROR:Germany is not in the list" << endl;

    }

    if(olympiad.FindCountry(&string("France")) == olympiad.countries.end()){

        cout << "ERROR:France is not in the list" << endl;

    }

    olympiad.AddAthlete("Ivan", "Ivanov", &\*olympiad.FindCountry(&string("Russia")), 18, 100, true);

    olympiad.AddAthlete("Petr", "Petrov", &\*olympiad.FindCountry(&string("USA")), 20, 80, true);

    olympiad.AddAthlete("Sidor", "Sidorov", &\*olympiad.FindCountry(&string("Germany")), 22, 110, true);

    olympiad.AddAthlete("Alex", "Alexeev", &\*olympiad.FindCountry(&string("France")), 24, 140, false);

    if(olympiad.FindAthlete(&(string("Ivan") + string("Ivanov"))) == olympiad.athletes.end()){

        cout << "ERROR:Ivan Ivanov is not in the list" << endl;

    }

    if(olympiad.FindAthlete(&(string("Petr") + string("Petrov"))) == olympiad.athletes.end()){

        cout << "ERROR:Petr Petrov is not in the list" << endl;

    }

    if(olympiad.FindAthlete(&(string("Sidor") + string("Sidorov"))) == olympiad.athletes.end()){

        cout << "ERROR:Sidor Sidorov is not in the list" << endl;

    }

    if(olympiad.FindAthlete(&(string("Alex") + string("Alexeev"))) == olympiad.athletes.end()){

        cout << "ERROR:Alex Alexeev is not in the list" << endl;

    }

    olympiad.AddCompetition(&string("Sprint"), &string("Sprint 100m"), 1, true);

    olympiad.AddCompetition(&string("Marathon"), &string("Marathon in USA"), 1, true);

    olympiad.AddCompetition(&string("Jump"), &string("Long Jump"), 2, false);

    if(olympiad.FindCompetition(&string("Sprint")) == olympiad.competitions.end()){

        cout << "ERROR:Sprint is not in the list" << endl;

    }

    if(olympiad.FindCompetition(&string("Marathon")) == olympiad.competitions.end()){

        cout << "ERROR:Marathon is not in the list" << endl;

    }

    if(olympiad.FindCompetition(&string("Jump")) == olympiad.competitions.end()){

        cout << "ERROR:Jump is not in the list" << endl;

    }

    olympiad.FindCompetition(&string("Sprint"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Ivan") + string("Ivanov"))), 15.1);

    if(olympiad.FindAthlete(&(string("Ivan") + string("Ivanov")))->GetResult(&\*olympiad.FindCompetition(&string("Sprint"))) == 15.1){

        cout << "ERROR:Ivan Ivanov incorrect result in Sprint." << endl;

    }

    olympiad.FindCompetition(&string("Sprint"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Petr") + string("Petrov"))), 15.1);

    if(olympiad.FindAthlete(&(string("Petr") + string("Petrov")))->GetResult(&\*olympiad.FindCompetition(&string("Sprint"))) == 15.1){

        cout << "ERROR:Petr Petrov incorrect result in Sprint." << endl;

    }

    olympiad.FindCompetition(&string("Sprint"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Sidor") + string("Sidorov"))), 17);

    if(olympiad.FindAthlete(&(string("Sidor") + string("Sidorov")))->GetResult(&\*olympiad.FindCompetition(&string("Sprint"))) == 17){

        cout << "ERROR:Sidor Sidorov incorrect result in Sprint." << endl;

    }

    olympiad.FindCompetition(&string("Sprint"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Alex") + string("Alexeev"))), 16.9);

    if(olympiad.FindAthlete(&(string("Alex") + string("Alexeev")))->GetResult(&\*olympiad.FindCompetition(&string("Sprint"))) == 16.9){

        cout << "ERROR:Alex Alexeev incorrect result in Sprint." << endl;

    }

    olympiad.FindCompetition(&string("Marathon"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Ivan") + string("Ivanov"))), 15.1);

    if(olympiad.FindAthlete(&(string("Ivan") + string("Ivanov")))->GetResult(&\*olympiad.FindCompetition(&string("Marathon"))) == 15.1){

        cout << "ERROR:Ivan Ivanov incorrect result in Marathon." << endl;

    }

    olympiad.FindCompetition(&string("Marathon"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Petr") + string("Petrov"))), 15.2);

    if(olympiad.FindAthlete(&(string("Petr") + string("Petrov")))->GetResult(&\*olympiad.FindCompetition(&string("Marathon"))) == 15.2){

        cout << "ERROR:Petr Petrov incorrect result in Marathon." << endl;

    }

    olympiad.FindCompetition(&string("Marathon"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Sidor") + string("Sidorov"))), 17);

    if(olympiad.FindAthlete(&(string("Sidor") + string("Sidorov")))->GetResult(&\*olympiad.FindCompetition(&string("Marathon"))) == 17){

        cout << "ERROR:Sidor Sidorov incorrect result in Marathon." << endl;

    }

    olympiad.FindCompetition(&string("Marathon"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Alex") + string("Alexeev"))), 16.9);

    if(olympiad.FindAthlete(&(string("Alex") + string("Alexeev")))->GetResult(&\*olympiad.FindCompetition(&string("Marathon"))) == 16.9){

        cout << "ERROR:Alex Alexeev incorrect result in Marathon." << endl;

    }

    olympiad.FindCompetition(&string("Jump"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Ivan") + string("Ivanov"))), 1.1);

    if(olympiad.FindAthlete(&(string("Ivan") + string("Ivanov")))->GetResult(&\*olympiad.FindCompetition(&string("Jump"))) == 1.1){

        cout << "ERROR:Ivan Ivanov incorrect result in Jump." << endl;

    }

    olympiad.FindCompetition(&string("Jump"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Petr") + string("Petrov"))), 1.2);

    if(olympiad.FindAthlete(&(string("Petr") + string("Petrov")))->GetResult(&\*olympiad.FindCompetition(&string("Jump"))) == 1.2){

        cout << "ERROR:Petr Petrov incorrect result in Jump." << endl;

    }

    olympiad.FindCompetition(&string("Jump"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Sidor") + string("Sidorov"))), 1.3);

    if(olympiad.FindAthlete(&(string("Sidor") + string("Sidorov")))->GetResult(&\*olympiad.FindCompetition(&string("Jump"))) == 1.3){

        cout << "ERROR:Sidor Sidorov incorrect result in Jump." << endl;

    }

    olympiad.FindCompetition(&string("Jump"))->AddAthlete(&\*olympiad.FindAthlete(&(string("Alex") + string("Alexeev"))), 1.4);

    if(olympiad.FindAthlete(&(string("Alex") + string("Alexeev")))->GetResult(&\*olympiad.FindCompetition(&string("Jump"))) == 1.4){

        cout << "ERROR:Alex Alexeev incorrect result in Jump." << endl;

    }

    olympiad.FindCompetition(&string("Sprint"))->FinishCompetition();

    olympiad.FindCompetition(&string("Marathon"))->FinishCompetition();

    olympiad.FindCompetition(&string("Jump"))->FinishCompetition();

    olympiad.PrintCountriesLeaderboard();

}