**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:**

Interface: Print menu and depending on answer call another method.

AddCountry: Ask manager a name of country, checks for uniqueness using FindCountry and call constructor Country.

AddCompetition: Ask manager a name of competition and description, checks for uniqueness using FindCopetition and call constructor Competition.

AddAthlete: Ask manager a name of competition and description, checks for uniqueness using FindAthlete and call constructor Athlete.

ManageCompetition: Call PrintListOfCompetitions, ask manager which Competitioon manage and call FindCompetition, then call Competition::Interface.

ManageAthlete: Call PrintListOfAthletes, ask manager which Athlete manage and call FindAthelte, then call Athelete::UpdateAthlete.

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;

    list <Country> countries;

    list <Competition> competitions;

    list <Athlete> athletes;

    void AddCountry();

    void AddCompetition();

    void AddAthlete();

    void ManageAthlete();

    void ManageCompetition();

    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);

public:

    Olympiad();

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

    void Interface();

    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 and pointer to list of all Athletes for adding new Athlete.

**Method function:**

AddAthlete: Ask manager a name of competition and description, checks for uniqueness using FindAthlete and call constructor Athlete.

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.

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

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 ManageAthletes();

    void PrintAthletes(const string filter = "");

    void PrintCountry();

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

    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, pointer to vector of Counties and list of Athletes for adding.

**Method function:**

ReadCompetirionData: Ask manager about type\_of\_result and first\_place\_is\_smaller and save, can be call only once.

FindCountry: Search Country 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.

AddAthlete: Ask manager a name of competition and description, checks for uniqueness using FindAthlete and call constructor Athlete.

ManageAthlete: Call PrintListOfAthletes, ask manager which Athlete manage and call FindAthelte, then call Athelete::UpdateAthlete.

PrintCompetition: Print data about Competition.

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

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

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

IsFinished: Retrunt finished.

AddCountry: Ask manager a name of country, checks for uniqueness using FindCountry and call constructor Country.

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

Interface: Print menu and depending on answer call another method.

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;

    vector<Country> \*countries;

    list <Athlete> \*all\_athletes;

public:

    Competition(const string \*name, const string \*description, vector<Country> \*countries);

    void ReadCompetitionData();

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

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

    void AddAthlete();

    void ManageAthletes();

    void PrintCompetition();

    void PrintAthletes(const string filter = "");

    void PrintCountries(const string filter = "");

    bool CheckResult(const float \*result);

    bool IsFinished();

    void AddCountry();

    void FinishCompetition();

    void Interface();

    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.

GetResult: Return result of certain competition.

UpdateAthlete: Ask manager which data need to update, ask manager this data and save.

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;

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 float result, const int type\_of\_result, const bool gender, const bool disqualification);

    void PrintAthlete();

    float GetResult(const Competition \*competition);

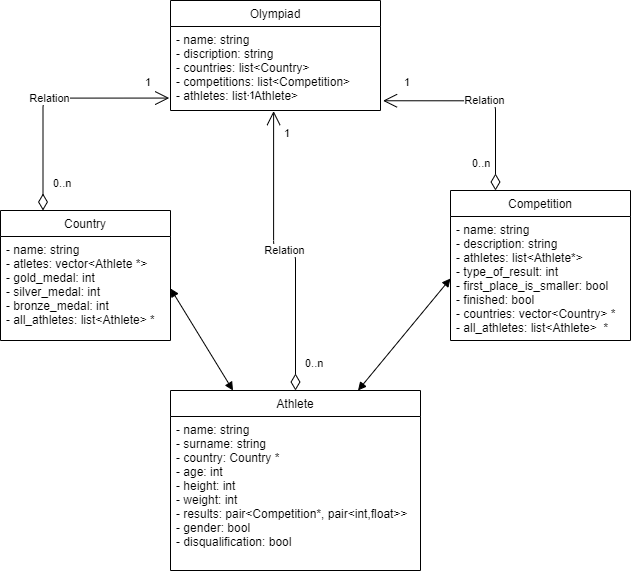
    void UpdateAthlete();

    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.