**Preliminary Project**

**Keyword: Olympiad**

Name: Kolenda Ekaterina

ID: 323811

**Introduction**

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

**Class introduction**

**Class Olympiad**

This class contain Countries, Competitions 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.

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

PrintCountriesLeaderboard: Print leaderboard using Country::PrintCountry, sorted by Country::operator<.

PrintListOfCompetitions: Print all Competitions using Competition::PrintCompetition.

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.

class Olympiad

{

private:

    string name;

    string description;

    vector<Country> countries;

    vector<Competition> competitions;

    void AddCountry();

    void AddCompetition();

    void ManageCompetition();

    void PrintCountriesLeaderboard();

    void PrintListOfCompetitions();

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

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

public:

    Olympiad();

    Olympiad(const string name, const string description);

    void Interface();

};

**Class Country**

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

**Method function:**

AddAthlete: Checks for uniqueness and push to vector.

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

PrintAthletes: Print all Athletes from vector using Athletes::PrintAthlete.

PrintCountry: Print data about country.

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

class Country

{

private:

    string name;

    vector<Athlete \*> athletes;

    int gold\_medal;

    int silver\_medal;

    int bronze\_medal;

public:

    Country();

    Country(const string name);

    void AddAthlete(const Athlete \*athlete);

    void AddMedal(const int medal);

    void PrintAthletes();

    void PrintCountry();

    bool operator<(const Country &other);

};

**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:**

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.

ReadAthleteData: Ask manager about data of Athlete, check result on uniqueness and call constructor Athlete.

PrintCompetition: Print data about Competition.

PrintAthletes: Call for all Athletes from set PrintAthlete.

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.

class Competition

{

private:

    string name;

    string description;

    set<Athlete> athletes;

    int type\_of\_result = 0;

    bool first\_place\_is\_smaller;

    bool finished;

    vector<Country> \*countries;

public:

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

    void ReadCompetitionData();

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

    void ReadAthleteData();

    void PrintCompetition();

    void PrintAthletes();

    bool CheckResult(const float result);

    bool IsFinished();

    void AddCountry();

    void FinishCompetition();

    void Interface();

};

**Class Athlete**

This class contain data about Athlete.

**Method function:**

PrintAthlete: Print data about Athlete.

GetResult: Return result.

Operator<: Compare result and return True or False.

class Athlete

{

private:

    string name;

    string surname;

    Country \*country;

    int age;

    int height;

    int weight;

    float result;

    int type\_of\_result;

    bool gender;

    bool disqualification;

public:

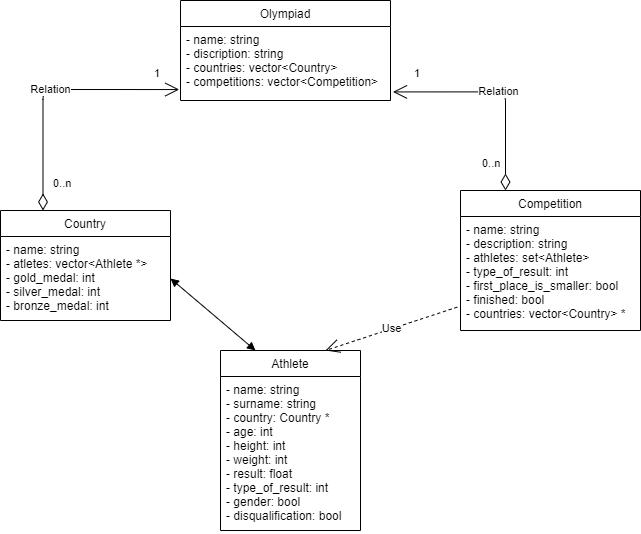
    Athlete(const string name, const string surname, const Country \*country, const int age, const int weight, const float result, const int type\_of\_result, const bool gender, const bool disqualification);

    void PrintAthlete();

    float GetResult();

    bool operator<(const Athlete &other);

};

**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.