|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |  | | **МИНОБРНАУКИ РОССИИ** | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  **МИРЭА** | | | | | | |
| Институт информационных технологий (ИТ) | | | | |
| Кафедра инструментального и прикладного программного обеспечения (ИППО) | | | | |
|  | | **Утверждаю** | | |
|  | | Заведующий кафедрой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
|  | | «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_2017 г. | | |
| **ЗАДАНИЕ** | | | | |
| **на выполнение курсовой работы** | | | | |
| **по дисциплине «Структуры и алгоритмы ОД»** | | | | |
|  | | | | |
| Студент Киясов Ислам Русланович Группа ИКБО-01-16 | | | | |
| **Тема:** «**Конкурс»** | | | | |
| **Исходные данные: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | | | |
| **Перечень вопросов, подлежащих разработке, и обязательного графического материала:**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Срок представления к защите курсового проекта (работы):**  до «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2017 г. | | | | |
|  | | | | |
| **Срок представления к защите курсовой работы: до** «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. | | | | |
|  | | | | |
| Задание на курсовую работу выдал | «\_\_\_»\_\_\_\_\_\_2017г. | |
| Задание на курсовую работу получил | «\_\_\_»\_\_\_\_\_\_2017г. | |
|  |  | |

Оглавление

[Техническое задание 3](#_Toc498517164)

[1. Введение. 3](#_Toc498517165)

[2. Основание для разработки. 3](#_Toc498517166)

[3. Назначение разработки. 3](#_Toc498517167)

[4. Требования, предъявляемые к программе. 3](#_Toc498517168)

[5. Требования к программной документации. 4](#_Toc498517169)

[6. Этапы разработки. 4](#_Toc498517170)

[Описание программы 4](#_Toc498517171)

[1. Общие сведения. 4](#_Toc498517172)

[2. Функциональное назначение программы: Программа для проведение конкурсов. 4](#_Toc498517173)

[3. Технические средства, которые используются при работе 4](#_Toc498517174)

[программы: 2-ядерный процессор Intel Core m3 с тактовой частотой 1,2 ГГц (ускорение Turbo Boost до 3,0 ГГц) и 4 МБ кэш-памяти третьего уровня. 4](#_Toc498517175)

[4. Вызов программы: запустить файл soft. 4](#_Toc498517176)

[5. Входные данные: данные вводимые с клавиатуры, или из файла (не обязательно) 4](#_Toc498517177)

[6. Выходные данные: значения элементов вектора, выводимые на 4](#_Toc498517178)

[экран. данные выводимые на экран. 4](#_Toc498517179)

[Библиографический список 4](#_Toc498517180)

[Приложение 5](#_Toc498517181)

[Меню: 5](#_Toc498517182)

[Новый конкурс: 5](#_Toc498517183)

[Добавление участника: 5](#_Toc498517184)

[Вывод участников: 5](#_Toc498517185)

[Исходный код: 7](#_Toc498517186)

## **Техническое задание**

### 1. Введение.

1.1. Наименование программы: «Конкурс».

1.2. Краткая характеристика области применения.

Программа разрабатывается для проведения конкурсов.

### 2. Основание для разработки.

Учебный план по направлению 231300 «Прикладная математика»

(бакалавры).

### 3. Назначение разработки.

Программа предназначена для проведения конкурсов.

### 4. Требования, предъявляемые к программе.

4.1. Требования к функциональным характеристикам программы.

В программе должны быть реализованы следующие операции:

Количество участников конкурса и количество призовых мест, на которые

претендуют участники, ограничено. Участник конкурса имеет

идентификационный номер, имя. Во время проведения конкурса его

участникам присваивают баллы.

Операции конкурса:

\* открытие конкурса;

\* регистрация участника;

\* окончание регистрации участников;

\* проведение конкурса: регистрация балла, полученного участником

конкурса;

\* окончание конкурса;

\* подведение итогов конкурса: сортировка списка участников по

убыванию баллов и вывод упорядоченного списка участников на

экран и в текстовый файл.

Регистрация участников может происходить в течение нескольких

сеансов работы программы, поэтому при завершении работы программы

(до окончания регистрации участников) необходимо участников,

зарегистрированных в текущем сеансе, дописывать в конец текстового

файла. Регистрация результатов конкурса и подведение итогов

выполняется в течение одного сеанса работы программы.

4.2. Требования к техническим средствам, используемым при работе

программы.

Работа программы должна осуществляться на macOS.

4.3. Требования к языкам программы и среде разработки программы.

Исходный код программы должен быть записан на языке

программирования С++.

4.4. Требования к информационным структурам на входе и выходе

программы.

На входе: ввод данных с клавиатуры, чтение файла.

На выходе: вывод данных на экран, запись в файл.

### 5. Требования к программной документации.

Состав программной документации должен включать:

\* техническое задание;

\* описание программы;

\* текст программы.

### 6. Этапы разработки.

6.1. Обзор способов организации данных и обоснование выбора структуры

данных для эффективного выполнения операций 01.10.2017-3.10.2017.

6.2. Разработка программы: 4.10.2017-6.10.2011.

6.4. Оформление пояснительной записки 26.10.2017-27.10.2017.

6.5. Защита курсовой работы: 2.11.2017.

## **Описание программы**

### 1. Общие сведения.

1.2. Наименование программы: «Конкурс».

1.3. Программное обеспечение, необходимое для функционирования

программы: macOS.

1.4. Язык программирования, на котором написана программа: С++.

### 2. Функциональное назначение программы: Программа для проведение конкурсов.

### 3. Технические средства, которые используются при работе

### программы: 2-ядерный процессор Intel Core m3 с тактовой частотой 1,2 ГГц (ускорение Turbo Boost до 3,0 ГГц) и 4 МБ кэш-памяти третьего уровня.

### 4. Вызов программы: запустить файл soft.

### 5. Входные данные: данные вводимые с клавиатуры, или из файла (не обязательно)

### 6. Выходные данные: значения элементов вектора, выводимые на

### экран. данные выводимые на экран.

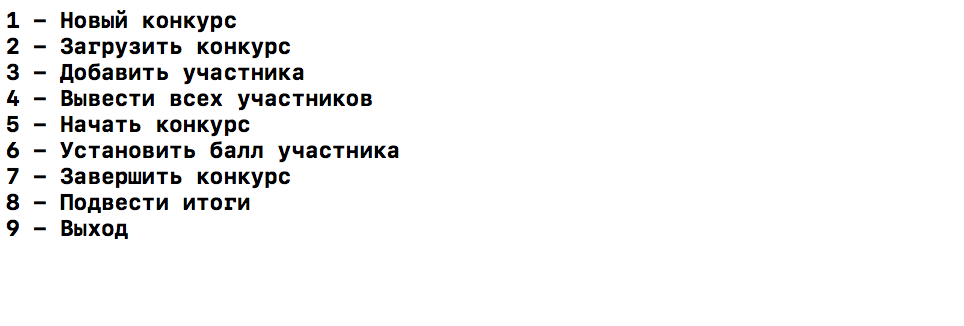
## **Библиографический список**

1. Лафоре Р. Объектно-ориентированное программирование в С++ / Р. Лафоре ; пер. с англ. А. Кузнецова, М. Назарова, В. Шрага. - 4-е изд. - СПб. : Питер, 2014. - 923 с.

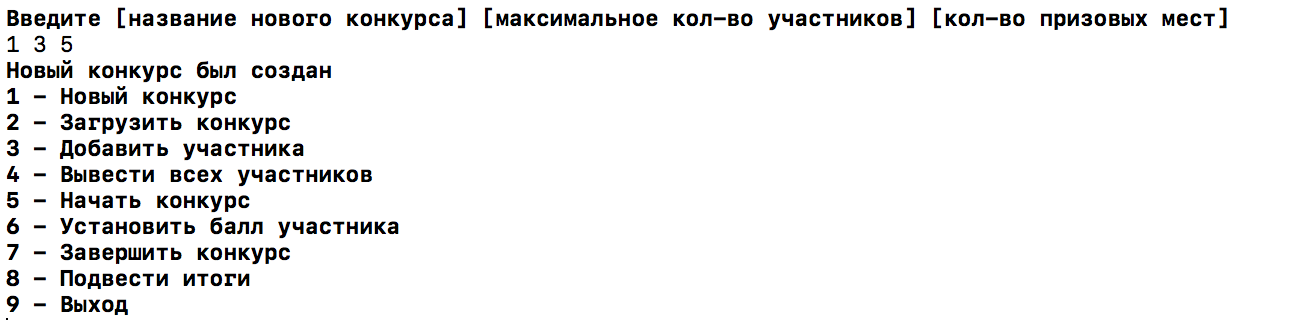
2. C++ Primer, Fourth Edition. Stanley B. Lippman, Josée Lajoie, Barbara E. Moo. ISBN: 5-8459-1121-4 (0-201-72148-1).

## **Приложение**

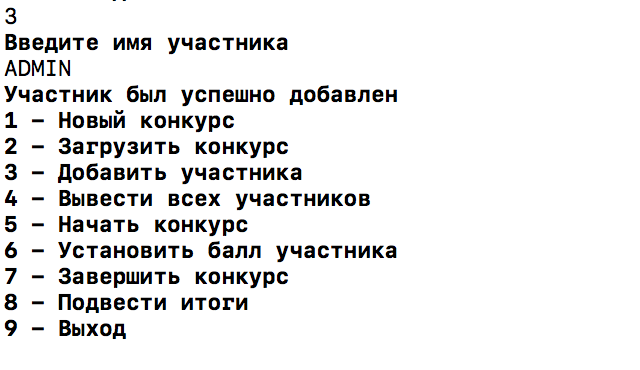
### Меню:



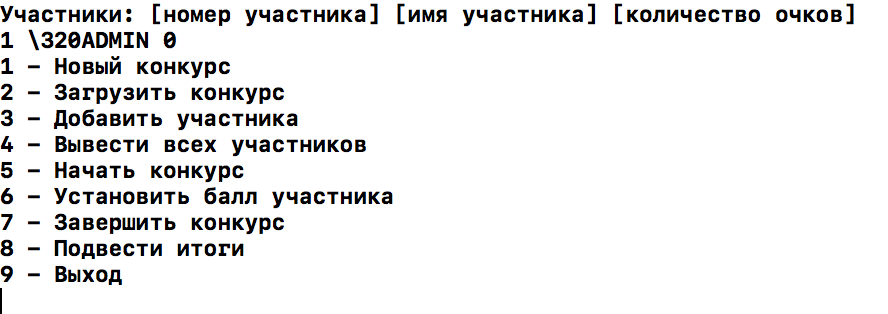
### Новый конкурс:



### Добавление участника:

****

### Вывод участников:

****

### Исходный код:

#include <string>

#include <iostream>

#include <fstream>

#include <vector>

using namespace std;

struct Participant //Участник

{

    string Name;

    int Id;

    double Score;

};

struct Contest //Конкурс

{

    vector<Participant> Participants; //Участники

    string Name;

    int MaxParticipants; //Макc. кол-во участников

    int PrizePlaceCount; //Кол-во призов

    int LoggedParticipantsCount; //Кол-во участников

    int Status;

    Contest(string name, int maxParticipants, int prizePlaceCount) // Новый

    {

        Name = name;

        Participants = vector<Participant>();

        MaxParticipants = maxParticipants;

        PrizePlaceCount = prizePlaceCount;

        LoggedParticipantsCount = 0;

        Status = 0;

    }

    Contest(string name, vector<Participant> participants, int maxParticipants, int prizePlaceCount)

    {

        Name = name;

        Participants = participants;

        MaxParticipants = maxParticipants;

        PrizePlaceCount = prizePlaceCount;

        LoggedParticipantsCount = participants.size();

        Status = 0;

    }

    bool TrySetScore(int id, double score) //Ставим баллы

    {

        if (Participants.size() < id)

        {

            return false;

        }

        else

        {

            Participants[id - 1].Score = score;

            return true;

        }

    }

    bool TryAddParticipant(string name) //Добавляем участника

    {

        if (Participants.size() == MaxParticipants)

            return false;

        else

        {

            Participant temp;

            temp.Id = Participants.size() + 1;

            temp.Name = name;

            temp.Score = 0;

            Participants.push\_back(temp);

            return true;

        }

    }

};

bool compareParticipants(const Participant &first, const Participant &second)

{

    return first.Score > second.Score;

}

Contest ResumeContest(string name)

{

    ifstream file;

    file.open(name + "\_participants.txt");

    int maxParticipants;

    int prizePlaceCount;

    file >> maxParticipants >> prizePlaceCount;

    vector<Participant> result;

    int counter = 1;

    while (!file.eof()) //Конец файла

    {

        Participant temp;

        getline(file, temp.Name);

        if (temp.Name != "")

        {

            temp.Id = counter++;

            temp.Score = 0;

            result.push\_back(temp);

        }

    }

    file.close();

    return Contest(name, result, maxParticipants, prizePlaceCount);

}

bool IsContestExist(string name)

{

    ifstream file(name + "\_participants.txt");

    return file.good();

}

void GetResults(vector<Participant> &participants)

{

    sort(participants.begin(), participants.end(), compareParticipants); //Сортировка по баллам

}

Contest StartNewContest(string name, int maxParticipants, int prizePlaceCount)

{

    return Contest(name, maxParticipants, prizePlaceCount);

}

void PrintParticipants(Contest contest)

{

    cout << "Участники: [номер участника] [имя участника] [количество очков]" << endl;

    for (int i = 0; i < contest.Participants.size(); i++)

    {

        cout << contest.Participants[i].Id << " " << contest.Participants[i].Name << " " << contest.Participants[i].Score << endl;

    }

}

void PrintResult(Contest contest)

{

    cout << "Название сореванования: " << contest.Name << endl;

    cout << "Количество участников " << contest.Participants.size() << endl;

    cout << "Максимальное количество участников в конкурсе " << contest.MaxParticipants << endl;

    cout << "Количество призовых мест " << contest.PrizePlaceCount << endl;

    cout << "Результаты конкурса: [место] [номер участника] [имя участника] [количество очков] " << endl;

    cout << "Победители:" << endl;

    for (int i = 0; i < min((int)contest.Participants.size(), contest.PrizePlaceCount); i++)

    {

        cout << i + 1 << " " << contest.Participants[i].Id << " " << contest.Participants[i].Name << " " << contest.Participants[i].Score << endl;

    }

    cout << "Остальные участники:" << endl;

    for (int i = contest.PrizePlaceCount; i < contest.Participants.size(); i++)

    {

        cout << i + 1 << " " << contest.Participants[i].Id << " " << contest.Participants[i].Name << " " << contest.Participants[i].Score << endl;

    }

}

void PrintResultToFile(Contest contest)

{

    ofstream file;

    file.open(contest.Name + "\_result.txt");

    file << "Название сореванования: " << contest.Name << endl;

    file << "Количество участников " << contest.Participants.size() << endl;

    file << "Максимальное количество участников в конкурсе " << contest.MaxParticipants << endl;

    file << "Количество призовых мест " << contest.PrizePlaceCount << endl;

    file << "Результаты конкурса: [место] [номер участника] [имя участника] [количество очков]"<<endl;

    file << "Победители:" << endl;

    for (int i = 0; i < min((int)contest.Participants.size(), contest.PrizePlaceCount); i++)

    {

        file << contest.Participants[i].Id << " " << contest.Participants[i].Name << " " << contest.Participants[i].Score << endl;

    }

    file << "Остальные участники:" << endl;

    for (int i = contest.PrizePlaceCount; i < contest.Participants.size(); i++)

    {

        file << contest.Participants[i].Id << " " << contest.Participants[i].Name << " " << contest.Participants[i].Score << endl;

    }

    file.close();

}

void PrintParticipantsToFile(Contest contest)

{

    if (!IsContestExist(contest.Name)) // Проверка на налич файла

    {

        ofstream f(contest.Name + "\_participants.txt");

        f << contest.MaxParticipants << " " << contest.PrizePlaceCount << endl;

        f.close();

    }

    ofstream file;

    file.open(contest.Name + "\_participants.txt", std::ios::app); // Добавляем в конец файла

    for (int i = contest.LoggedParticipantsCount; i < contest.Participants.size(); i++)

    {

        file << contest.Participants[i].Name << endl;

    }

    file.close();

}

int main()

{

    setlocale(LC\_ALL, "Russian");

    Contest currentContest("", 0, 0);

    while (true)

    {

        string input;

        cout << "1 - Новый конкурс\n2 - Загрузить конкурс\n3 - Добавить участника\n4 - Вывести всех участников\n";

        cout << "5 - Начать конкурс\n6 - Установить балл участника\n7 - Завершить конкурс\n8 - Подвести итоги\n9 - Выход\n";

        char c;

        cin >> c;

        switch (c)

        {

            case '1':

                if (currentContest.MaxParticipants != 0)

                    PrintParticipantsToFile(currentContest);

                cout << "Введите [название нового конкурса] [максимальное кол-во участников] [кол-во призовых мест]" << endl;

                int maxParticipants, prizePlaces;

                cin >> input >> maxParticipants >> prizePlaces;

                currentContest = StartNewContest(input, maxParticipants, prizePlaces);

                cout << "Новый конкурс был создан" << endl;

                break;

            case '2':

                if (currentContest.MaxParticipants != 0)

                    PrintParticipantsToFile(currentContest);

                cout << "Введите конкурса, который вы хотите загрузить" << endl;

                cin >> input;

                if (!IsContestExist(input))

                {

                    cout << "Такой конкурс не найден" << endl;

                    currentContest = Contest("", 0, 0);

                }

                else

                {

                    currentContest = ResumeContest(input);

                    cout << "Конкурс загружен" << endl;

                }

                break;

            case '3':

                if (currentContest.MaxParticipants == 0)

                {

                    cout << "Нельзя добавить участника пока не выбран конкурс" << endl;

                    break;

                } else if (currentContest.Status != 0)

                {

                    cout << "Нельзя добавить участника в конкурс который уже был начат" << endl;

                    break;

                }

                cout << "Введите имя участника" << endl;

                cin >> input;

                if (currentContest.TryAddParticipant(input))

                    cout << "Участник был успешно добавлен" << endl;

                else

                    cout << "Превышено количество участников конкурса" << endl;

                break;

            case '4':

                if (currentContest.MaxParticipants == 0)

                    cout << "Конкурс не выбран" << endl;

                else

                    PrintParticipants(currentContest);

                break;

            case '5':

                if (currentContest.MaxParticipants == 0)

                    cout << "Конкурс не выбран" << endl;

                else if (currentContest.Status == 1)

                    cout << "Нельзя начать конкурс который уже был начат" << endl;

                else

                {

                    currentContest.Status = 1;

                    cout << "Конкурс начат" << endl;

                }

                break;

            case '6':

                if (currentContest.MaxParticipants == 0)

                    cout << "Конкурс не выбран" << endl;

                else if (currentContest.Status != 1)

                    cout << "Конкурс недоступен для проставки баллов участников" << endl;

                else

                {

                    PrintParticipants(currentContest);

                    cout << "Введите [номер участника] [балл]";

                    int n;

                    double score;

                    cin >> n >> score;

                    if (!currentContest.TrySetScore(n, score))

                        cout << "Участник с таким номером не найден" << endl;

                    else

                        cout << "Балл участника успешно установлен" << endl;

                }

                break;

            case '7':

                if (currentContest.MaxParticipants == 0)

                    cout << "Конкурс не выбран" << endl;

                else if (currentContest.Status != 1)

                    cout << "Невозможно завершить этот конкурс" << endl;

                else

                {

                    cout << "Конкурс завершён, можете подвести итоги" << endl;

                    currentContest.Status = 2;

                }

                break;

            case '8':

                if (currentContest.MaxParticipants == 0)

                    cout << "Конкурс не выбран" << endl;

                else if (currentContest.Status != 2)

                    cout << "Невозможно подвести итоги незавершенного конкурса" << endl;

                else

                {

                    GetResults(currentContest.Participants);

                    PrintResult(currentContest);

                    PrintResultToFile(currentContest);

                }

                break;

            case '9':

                if (currentContest.MaxParticipants != 0)

                    PrintParticipantsToFile(currentContest);

                return 0;

            default:

                cout << "Неизвестная команда" << endl;

        }

    }

    return 0;

}