Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра инженерной психологии и эргономики

Дисциплина: Основы конструирования программ

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

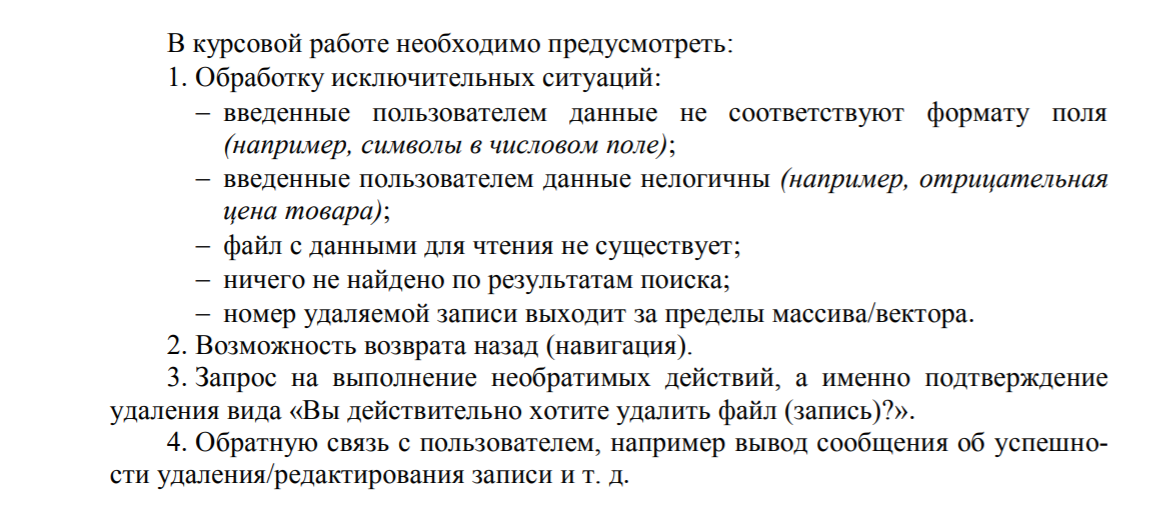
на тему

**РАЗРАБОТКА ПРОГРАММЫ УЧЕТА СВЕДЕНИЙ О МУЗЫКАЛЬНОМ КОНКУРСЕ**

Выполнил: студент группы 910901 Алешко Ксения Игоревна

Проверил: Меженная Марина Михайловна

Минск 2020



Содержание.

1. Требования к программе (см. пояснения ниже).

2. Конструирование программы.

2.1 Разработка модульной структуры программы (см. пояснения ниже).

2.2 Выбор способа организации данных (см. пояснения ниже).

2.3 Разработка перечня пользовательских функций программы (см. пояснения ниже).

3. Разработка алгоритмов работы программы (см. пояснения ниже).

3.1. Алгоритм функции main.

3.2. Алгоритм функции registration.

3.3. Алгоритм функции deleteAccount.

4. Описание работы программы (см. пояснения ниже).

4.1 Авторизация.

4.2 Модуль администратора.

4.3 Модуль пользователя.

4.4 Исключительные ситуации.

*Приложение (обязательное): листинг кода с комментариями (приводится ВЕСЬ код с авторским форматированием и комментариями; допускается применение в листинге кода шрифта 12 пунктов и выше).*

1. **Требования к программе.**
   1. **Разработка программы учета сведений о музыкальном конкурсе.**

Информация о конкурсе включает: ФИО участника, год рождения, название страны, наименование музыкального инструмента (гитара, фортепиано, скрипка, виолончель и др.), занятое место по результатам конкурса.

Индивидуальное задание: по каждому классу музыкальных инструментов вывести список первых трех мест с указанием возраста победителей. Вывести список самых молодых (до 12 лет) победителей конкурса в порядке увеличения возраста.

Общее для всех вариантов задание: реализовать авторизацию для входа в систему, функционал администратора и функционал пользователя.

* 1. **Исходные данные.**

1. Разработка программы учета сведений о музыкальном конкурсе.

2. Язык программирования С++.

3. Среда разработки Microsoft Visual Studio.

4. Вид приложения – консольное.

5. Парадигма программирования – процедурная.

6. Способ организации данных – структуры (struct).

7. Способ хранения данных – файлы (по согласованию с преподавателем допускается подключение баз данных).

8. Каждая логически завершенная подзадача программы должна быть реализована в виде отдельной функции.

9. Построение программного кода должно соответствовать соглашению о коде «С++ Code Convention».

10. К защите курсовой работы представляются: консольное приложение и пояснительная записка.

11. Текст пояснительной записки оформляется в соответствии со стандартом предприятия СТП 01–2017.

**1.3.** **Функциональные требования.**

**Первым этапом работы программы** является авторизация – предоставление прав доступа.

В рамках данного этапа считываются данные из файла с учетными записями пользователей следующего вида:

- login;

- pass;

- role (данное поле служит для разделения в правах администраторов и пользователей)

- access (данное поле служит для предоставления доступа пользователя к аккаунту).

После ввода пользователем своих персональных данных (логина и пароля) и сверки со считанной из файла информацией предусмотрена возможность входа:

- в качестве администратора (в этом случае, например, role = 1),

- в качестве пользователя (в этом случае, например, role = 0).

При вводе пароля осуществляется его маскирование символом \*.

**Регистрация новых пользователей** осуществляется администратором в режиме работы с учетными записями пользователей (т.е. администратор сам создает для пользователей аккаунты), также пользователь может сам зарегистрироваться на начальном этапе программы.

**Вторым этапом работы программы** является собственно работа с данными, которая становится доступной только после прохождения авторизации. Данные хранятся в отдельном файле.

Для работы с данными предусмотрены два функциональных модуля: модуль администратора и модуль пользователя.

**Модуль администратора** включает следующие подмодули (с указанием функциональных возможностей):

1. Управление учетными записями пользователей:

- просмотр всех учетных записей;

- добавление новой учетной записи;

- редактирование учетной записи;

- удаление учетной записи.

Допускается создавать учетные записи для нескольких администраторов.

2. Работа с данными:

а) режим редактирования:

- просмотр всех данных;

- добавление новой записи;

- удаление записи;

- редактирование записи;

б) режим обработки данных:

- выполнение индивидуального задания (вывод списка первых трех мест по каждому классу музыкальных инструментов с указанием возраста победителей. Вывод списка самых молодых (до 12 лет) победителей конкурса в порядке увеличения возраста);

- поиск данных (по ФИО, году рождения и занятому месту);

- сортировка (по ФИО, году рождения и занятому месту).

**Модуль пользователя** включает подмодуль работы с данными со следующими функциональными возможностями:

- просмотр всех данных;

- выполнение индивидуального задания (вывод списка первых трех мест по каждому классу музыкальных инструментов с указанием возраста победителей. Вывод списка самых молодых (до 12 лет) победителей конкурса в порядке увеличения возраста);

- поиск данных (по ФИО, году рождения и занятому месту);

- сортировка (по ФИО, году рождения и занятому месту).

Для реализации перечисленных модулей/подмодулей создано меню с соответствующими пунктами.

**В курсовой работе предусмотрены:**

1. Обработка исключительных ситуаций:

- введенные пользователем данные не соответствуют формату поля (символы в числовом поле);

- введенные пользователем данные нелогичны (отрицательный год рождения или занятое место);

- файл с данными для чтения не существует;

- ничего не найдено по результатам поиска;

- номер удаляемой записи выходит за пределы вектора.

2. Возможность возврата назад (навигация).

3. Обратная связь с пользователем.

**1.4. Требования к программной реализации.**

1. Все переменные и константы имеют осмысленные имена в рамках тематики варианта курсовой работы. Для формирования составного имени используется нижнее подчеркивание (например, number\_of\_students) или «верблюжья нотация» (например, flagExit). Константам присвоены имена, состоящие из букв верхнего регистра (например, SIZE\_ARR\_OF ACCOUNTS, FILE\_OF\_ACCOUNTS)

2. Имена функций являются осмысленными, строится по принципу глагол+существительное (например, addAccount, findStudentBySurname).

3. Не использовать оператор прерывания goto.

4. Код не содержит неименованных числовых констант («магических» чисел), неименованных строковых констант (например, имен файлов и др.). Подобного рода информацию следует представлена, как глобальные константы.

5. Код не дублируется.

6. Одна функция решает только одну задачу. При этом внутри функции возможен вызов других функций.

7. Выполнение операций чтения/записи в файл сведено к минимуму (т.е. после однократной выгрузки данных из файла в вектор дальнейшая работа ведется с этим вектором, а не происходит многократное считывание данных из файла в каждой функции).

Код логически независимых модулей вынесен в отдельные .cpp файлы и они подключены с помощью заголовочных .h файлов.

**2. Конструирование программы.**

**2.1. Разработка модульной структуры программы.**



**Рис. 1 Модульная структура программы.**

**2.2. Выбор способа организации данных.**

В качестве выбора способа описания входных данных были использованы структуры (struct) со следующими полями:

- для учетных записей пользователей: login (логин), password (пароль), role (роль), access(доступ).

- для данных: name (ФИО), country (страна), year (год рождения), instrument (инструмент), place (занятое место).

В качестве способа объединения входных данных были использованы вектора. Их выбранная область видимости локальная.

**2.3 Разработка перечня пользовательских функций программы.**

Курсовая работа состоит из одного главного cpp-файла, 7 связок (h-файл + cpp-файл):

- Coursework.cpp

- authorization.h + authorization.cpp

- administrator.h + administrator.cpp

- user.h + user.cpp

- validation.h + validation.cpp

- accounts\_operations.h + accounts\_operations.cpp

- data\_operations.h + data\_operations.cpp

- file\_operations.h + file\_operations.cpp

**authorize.h:**

void showAccountsMenu();

//отображение меню авторизации

void authorize(vector<Account>& accounts, vector<Participant>& records);

//авторизация пользователя

void login(vector<Account> &accounts, vector<Participant>& records);

//вход пользователя в систему

void registration(vector<Account> &accounts);

//регистрация пользователя

string transformPassword();

//преобразование пароля в \*

void hashPassword(Account account);

//хеширование пароля

**administrator.h:**

void showAdminMenu(vector<Account>& accounts, Account current\_account, vector<Participant>& records);

//отображение меню для администратора

void showAdminAccountMenu(vector<Account>& accounts, Account current\_account);

//отображение меню работы с аккаунтами для администратора

void showAdminDataMenu(vector<Participant>& records);

//отображение меню работы с данными для администратора

**user.h:**

void showUserMenu(vector<Participant> records);

//отображение меню для пользователя

**validation.h:**

bool isNumberNumeric();

//проверка введенных данных на числовой тип

double getNumber();

//проверка на ввод числовых данных

**accounts\_operations.h:**

Account\* getAccount(vector<Account> &accounts, string login);

//нахождение аккаунта по логину

Account\* createFirstAdmin();

//создание первого администратора на случай, если файл с аккаунтами пуст

void showAccount(vector<Account> accounts);

//просмотр аккаунтов

void addAccount(vector<Account> &accounts);

//добавление аккаунта

void editAccount(vector<Account>& accounts);

//редактирование аккаунта

void editionMenu(Account& account);

//отображение меню для редактирования

void deleteAccount(vector<Account>& accounts, Account current\_account);

//удаление выбранного аккаунта

**data\_operations.h:**

void showData(vector<Participant> records);

//просмотр всех данных

void showRecord(Participant record);

//просмотр записи

void showHeader();

//отображение заголовочной строки в таблице

void addRecord(vector<Participant>& records);

//добавление участника

void editData(vector<Participant>& records);

//редактирование участника

void editionMenu(Participant& record);

//отображение меню для редактирования

void deleteRecord(vector<Participant>& records);

//удаление выбранного участника

void showFindingMenu(vector<Participant>& records);

//отображение меню выбора поиска

void findRecordByName(vector<Participant>& records);

//поиск участника по имени

void findRecordByYear(vector<Participant>& records);

//поиск участника по году рождения

void findRecordByPlace(vector<Participant>& records);

//поиск участника по занятому месту

void showSortingMenu(vector<Participant>& records);

// отображение меню выбора сортировки

void sortRecordsByName(vector<Participant>& records);

//вызов функции SortByName и отображение отсортированных участников

bool SortByName(Participant participant\_a, Participant participant\_b);

//сортировка участников по имени

void sortRecordsByYear(vector<Participant>& records);

// вызов функции SortByYear и отображение отсортированных участников

bool SortByYear(Participant participant\_a, Participant participant\_b);

//сортировка участников по году рождения

void sortRecordsByPlace(vector<Participant>& records);

// вызов функции SortByPlace и отображение отсортированных участников

bool SortByPlace(Participant participant\_a, Participant participant\_b);

//сортировка участников по занятому месту

void Task(vector<Participant>& records);

//выполнение индивидуального задания (вывод списка первых трех мест по каждому классу музыкальных инструментов с указанием возраста победителей.

Вывод списка самых молодых (до 12 лет) победителей конкурса в порядке увеличения возраста)

void deleteData(vector<Participant>& records);

//удаление всех участников

**file\_operations.h:**

vector<Account> readAccountsFile();

//заполнение вектора аккаунтов записями из файла

void writeAccountFile(vector<Account> accounts);

// перезаписывание аккаунта в файле

void writeToEndAccountFile(Account account);

//запись аккаунта в конец файла

vector<Participant> readRecordsFile();

//заполнение вектора участников записями из файла

void writeRecordFile(vector<Participant> records);

//перезаписывание участника в файле

void writeToEndRecordsFile(Participant record);

//запись участника в конец файла

**3. Разработка алгоритмов работы программы.**

**3.1. Алгоритм функции main.**

****

**Рис. 2 Алгоритм функции main**

**3.2. Алгоритм функции registration.**

****

**Рис.3. Алгоритм функции registration**

**3.3. Алгоритм функции deleteAccount.**

****

**Рис.4. Алгоритм функции deleteAccount**

**4. Описание работы программы.**

**4.1. Авторизация.**

**4.2. Модуль администратора.**

**4.3. Модуль пользователя.**

**4.4. Исключительные ситуации.**