СОДЕРЖАНИЕ

ВВЕДЕНИЕ

База данных «Школьники» предназначена для ведения учета информации об успеваемости. Автоматизация хранения и обработки такого рода информации является актуальным и востребованным, так как позволяет регулярно мониторить состояние учитываемых объектов, оперативно получать необходимую информацию, своевременно отвечать на поставленные запросы, безошибочно и корректно формировать требуемые отчеты.

Ведение базы данных подразумевает стандартный набор действий над записями, а именно: добавление новых записей, корректировка конкретной информации, обзорный просмотр и сортировка всех записей, выборка информации по заданным критериям, статистическая обработка информации и многое другое.

Цель курсовой работы – систематизация, развитие и применение теоретических и практических знаний и умений по алгоритмизации и программированию на примере разработки программного комплекса для автоматизированной обработки массива записей базы данных «Школьники», представленной в отдельном текстовом файле.

Необходимо разработать приложение, которое будет обрабатывать массив структурированных данных «Школьники» и которое, в том числе, будет вести учёт успеваемости учеников.

1. ПОСТАНОВКА ЗАДАЧИ

Сперва определим предметную область, чтобы поставить корректные цели и задачи. Предметная область, подлежащая изучению - «школьники». В сферу этой предметной области попадают пол школьников, имена, фамилии, отчества, успеваемость (оценки по определённым предметам и общий средний балл), класс, номер телефона.

Необходимо разработать приложение, которое будет обрабатывать массив структурированных данных «Школьники» и которое, в том числе, будет вести учёт успеваемости учеников. Бинарный файл исходных данных должен содержать следующие поля: ФИО, пол, класс, номер телефона, оценки по физике, математике, русскому языку и литературе. Структура «Школьники» также будет содержать дополнительное рассчитываемое поле со средним баллом каждого школьника. Это поле будет рассчитываться в самой программе и по ходу её выполнения записываться в рабочий бинарный файл. Бинарный файл был выбран основным для работы из-за своей лёгкости (запись и чтение из него реализуется сразу структурой, в то время как в текстовый файл необходимо записывать отдельно каждое поле), для ещё большего удобства в самое начало бинарного файла записывается размерность массива (количество записей в файле).

Приложение необходимо разработать в консольном виде, а также создать консольное меню. Как отмечалось ранее, данные должны быть организованны с помощью структур, а записи должны храниться в отдельном бинарном файле и в текстовом для того, чтобы всегда можно было проверить правильность работы с данными.

В программе должно быть реализовано:

– дополнительное поле с рассчитываемым значением – «Средний балл»;

– ввод информации из текстового файла в массив указателей на записи;

– добавление новых элементов в структуру в конец массива, после выбранной записи;

– просмотр всех элементов массива, вывод информации из массива в файл;

– изменение записи, удаление информации по любому полю структуры;

– сортировка по всем полям структуры с созданием индексных бинарных файлов;

– поиск информации в массиве структур по любому полю;

– вычисление количества записей со средним баллом из задаваемого интервала.

Алгоритмы функциональных процедур проектируемого приложения представлены в виде блок-схем, выполненных в соответствии с ГОСТ ЕСПД 19.701-90. Для создания программного комплекса была использована интегрированная среда разработки Microsoft Visual Studio 2019. Код организован с помощью языка программирования C++.

Автоматизированная информационная система (АИС) - это система, в которой информационный процесс управления автоматизирован за счет применения специальных методов обработки данных, использующих комплекс вычислительных, коммуникационных и других технических средств, в целях получения и доставки результатной информации пользователю-специалисту для выполнения возложенных на него функций управления.

Решения в системе управления принимаются людьми на основе информации, являющейся продуктом ИС. На ее входе находится первичная информация обо всех изменениях, происходящих в объекте управления. Она фиксируется в результате выполнения функций оперативного учета. В ИС первичная информация преобразуется в результатную, пригодную для принятия решений. В автоматизированных ИС часть процедур формального преобразования первичной информации в результатную автоматически выполняются техническими средствами по заранее заданным алгоритмам, без непосредственного вмешательства человека.

Это не означает, что ИС может полностью функционировать в автоматическом режиме. Персонал системы управления определяет состав и структуру первичной и результатной информации, порядок сбора и регистрации первичной информации, контролирует ее полноту и достоверность, определяет порядок выполнения преобразований первичной информации в результатную, контролирует ход выполнения процесса преобразований. К тому же до сих пор слабо автоматизирована процедура сбора первичной информации, поэтому ее ввод в технические средства также осуществляется персоналом ИС.

Важнейшей частью технических средств преобразования информации являются компьютеры, осуществляющие автоматический процесс обработки данных на основе заранее заданных программ. В современных АИС процедуры информационного процесса децентрализованы и выполняются в диалоговом режиме работы пользователя с компьютером, что позволяет ему контролировать процесс преобразования данных, оперативно направляя его в нужное ему русло. Этим они отличаются от АИС, базирующихся на больших ЭВМ, в которых процесс обработки информации выполнялся централизованно и был отделен от управленческого персонала. Последний получал лишь конечные результаты обработки данных и, если они его по тем или иным причинам (например, вследствие поздно выявленных ошибок в исходных данных) не

устраивали, вынужден был делать запрос соответствующим службам на повторение процесса решения интересующей его задачи.

Таким образом, в современных АИС автоматически выполняемые процедуры информационного процесса интегрированы с функциями управления. Наряду со своими основными функциями, их непосредственно выполняет управленческий персонал. Более того, используя инструментальные программные средства, ориентированные на пользователя, не имеющего профессиональной компьютерной подготовки, специалист-управленец часто сам может автоматизировать выполнение необходимых ему процедур обработки данных, выступая и в роли постановщика задачи и программиста.

Отметим, что в современном понятии термин «информационные системы» подразумевает автоматизацию информационных процессов. Поэтому оба термина используются как равноправные. Но следует помнить о том, что информационные системы могут использовать и неавтоматизированную технологию обработки информации.

Одно из важнейших мест в информационных системах предприятий занимает функция учёта. Для выполнения в полном объеме функций в управлении учреждением образования и для составления отчетности, предоставляемой внешним пользователям, необходимо осуществлять сбор, регистрацию, передачу, накопление, хранение и обработку учетных данных. Для реализации этого информационного процесса требуются соответствующие формы организации работы, технические средства, методы и способы преобразования данных, а также персонал определенной квалификации.

На сегодняшний день подобные базы данных (электронные дневники), чрезвычайно полезны в любой школе мира. Они позволяют надёжно хранить много информации, ежедневно использовать, обрабатывать и обновлять большие массивы данных.

Автоматизированная система позволит приводить в порядок информацию по базам данных школьников и их успеваемости, возможность обновления в связи с изменением некоторых факторов (изменение учеников в классе, изменение оценок, номеров телефона и так далее). В частности, изменения оценок происходят постоянно, поэтому для удобного слежения за успеваемостью учеников данная база данных очень полезна. А также использование баз данных — возможность надёжного хранения информации.

2. РАЗРАБОТКА АЛГОРИТМОВ

Этот этап один из самых важных. После постановки задачи, для дальнейшей работы будет необходимо разработать алгоритмы. Разработка алгоритма – это запись последовательность действия, приводящих к решению задачи, на основе выбранного метода. Успешная разработка алгоритма позволяет избежать многих ошибок, поскольку именно на этом этапе определяется логика будущей программы. Для решения поставленной задачи потребуется создать простой и удобный алгоритм взаимодействия пользователя с базой данных «Школьники». Для этого в основном модуле программы нужно реализовать меню. Также мы будем использовать бинарный файл для хранения основной информации, текстовый файл также будет, но основная работа будет происходить в бинарном, в текстовый же будет только происходить запись уже изменённой информации. Ещё для сортировки будут использованы индексные бинарные файлы, если сортировка уже происходила. После изменения информации же сортировка будет происходить заново, с записывание обновлённых индексов в соответствующий бинарный файл (каждому полю в структуре будет создан соответствующий индексный бинарный файл). Для построения блок-схем будем использовать programforyou.ru/block-diagram-redactor, из-за удобства и возможности редактировать их в дальнейшем.

Алгоритм работы main будет заключаться в следующем: в начале работы программы будет объявляться переменная, хранящая имя основного рабочего бинарного файла, хранящего всю исходную информацию. Далее создаётся переменная, которая хранит количество элементов, присваиваем ей значение ноль, после чего создаём динамический массив структур с 0 количеством элемента (это необходимо для корректной работы функции чтения данных из файла), далее мы вызываем функция чтения из файла, потом функцию меню, после окончания работы функции меню, удаляем выделенную в начале динамическую память и после этого программа завершает свою работу.

Алгоритм ввода информации из файла в массив записей:

1. Открываем бинарный файл в корневой (там, где исполняемый файл) папке программы для чтения.
2. Считываем сначала количество записей в файле (эта информация хранится в самом начале бинарного файла)
3. Удаляем выделенную ранее память под динамический массив структур.

4. Присваиваем указателю на удалённый ранее массив адрес нового динамического массива с уже новым количеством элементов в массиве.
5. С помощью цикла for от 0 до количества структур считываем все записи с файла в массив структур.
6. Закрываем файл.

Алгоритм меню:
1. Выводятся возможные действия меню
2. Вводится номер нужного действия
3. Согласно действию вызывается необходимая функция
4. Если это действие не выход из меню, тогда возврат к пункту 2, если же выход - функция прекращает свою работу.

Алгоритм вывода информации на экран:
1. Проверка количества выводов (если их нет, выводит шапку)
2. Выводим все записи на экран

Алгоритм добавления новой записи в файл:
1. Выводим текущий список
2. Выбираем после какой записи добавить новую
3. Создание расширенного буферного динамического массива
4. Ввод новой записи
5. Удаление основного динамического массива
6. Копирование адреса буферного массива в указатель, ранее указывавший на удалённый основной динамический массив.
7. Перезапись всего массива в файл (текстовый и бинарный)

Алгоритм перезаписи в файл:
1. Открываем текстовый файл
2. Записываем поочерёдно каждую структуру в файл (перед добавлением поля со средним баллом, рассчитать его)
3. Закрытие текстового файла
4. Открытие бинарного файла
5. Записать поочерёдно каждую структуру в файл
6. Закрытие бинарного файла

Алгоритм ввода определённой информации согласно интервалу:
1. Ввод необходимого интервала

2. Перебор всех записей массива
3. Если поле со средним баллом принадлежит интервалу, вывести эту структуру на экран и прибавить единицу к количеству выводов
4. После окончания цикла вывести на экран количество выводов (результат)

Алгоритм изменения бинарного файла:
1. Открываем соответствующий индексный бинарный файл
2. Уменьшаем переменную с количеством записей на 1
3. Записываем в файл эту переменную
4. Закрываем файл

Алгоритм поиска:
1. Выбор поля, по которому будет происходить поиск
2. Ввод информации для поиска
3. Перебор всех элементов массива структур
4. Сравнения поля текущей структуры с введённой информацией для поиска
5. Если информация идентична, тогда вывести всю текущую структуру на экран

Алгоритм удаления:
1. Вывод текущего списка на экран
2. Создание буферного динамического массива
3. Копирование элементов из основного массива в буферный
4. Удаление основного динамического массива
5. Выбор варианта удаления: по полю или по целой записи
6. Если по целой записи, тогда вводится номер нужной для удаления информации.

6.1. Перебор буферного массива структур

6.2. Если номер элемента + 1 совпадает с номером для удаления, происходит удаления всей записи (перемещение нужной записи в конец массива)

6.3. Переменная количество элементов в структуре уменьшается на единицу

6.4. Создаётся новый основной динамический массив, с уже уменьшенным числом элементов

6.5. Копирование элементов из буферного массива в основной

7. Если по конкретному полю, то вызывается функция поиска с соответствующим режимом для удаления, и ей передаётся буферный массив (см. предыдущий алгоритм, меняться здесь будет только 5 пункт (если информация идентична, то произойдёт удаление))

7.1. Идентичен 6.4, соответственно следующий шаг идентичен 6.5

8. Удаление буферного массива
9. Перезапись информации в файл

Алгоритм изменения информации:

1. Выбор способа изменения: сразу всю запись или только конкретное поле

2.1. Если сразу всю запись, то выводится весь текущий список

2.2. Вводится номер записи, которую необходимо изменить

2.3. Перебор всех элементов массива

2.4. Если номер элемента + 1 равен номеру записи для изменения, то вводится новая информацию в эту запись посредством последовательного ввода информации в каждое из полей

2.5. Изменение индексных бинарных файлов каждого из полей (см. Алгоритм изменения бинарного файла)

3. Если изменение только конкретного поля, то сначала нужно выбрать поле, которое нужно менять

3.1. Следующие несколько действий идентичны с 2.2, 2.3

3.2. Если номер элемента + 1 равен номеру записи для изменения, то вводится новая информация в соответствующее поле текущей структуры

3.3. Изменяется индексный бинарный файл соответствующего поля (см. Алгоритм изменения бинарного файла)

4. Перезапись информации в файл

Алгоритм сортировки:

1. Создаётся поток для чтения и записи данных одновременно (fstream)
2. Создаётся динамический массив индексов (его размерность равна размерности основного динамического массива)
3. Массив индексов заполняется индексами посредством перебора основного массива и записи его индексов в массив индексов
4. Выводится вся текущая информация
5. Выбор поля для сортировки
6. Согласно выбранному полю открывается соответствующий индексный бинарный файл
7. С него считывается количество индексов
8. Если количество индексов меньше, чем количество элементов в структуре, то с помощью сортировки выбором происходит сортировка элементов основного массива и массива индексов

8.1. Вывод текущей информации

8.2. Запись в соответствующий бинарный файл количество индексов в массиве индексов

8.3. Запись элементов отсортированного массива индексов в соответствующий индексный бинарный файл

9. Если количество индексов равно количеству элементов массива, то с индексного бинарного файла считываются индексы

9.1. После считывания одного индекса выводится соответствующая структура из основного массива (благодаря массиву индексов, ведь индексы из файла записываются именно туда), и так пока индексы в файле не завершатся и весь основной массив отсортированный будет выведен на экран

10. Закрытие файла

11. Удаление динамического массива индексов

3. РАЗРАБОТКА ПРОГРАММЫ

3.1. Выбор средств программирования

Для разработки программы была выбрана среда разработки VisualStudio, так как она является одной из самых современных, удобных и регулярно обновляемых сред программирования, имеет в своем распоряжении большое количество инструментов для разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред для разработки, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки. Операционная система, установленная на ПК, Windows 10, регулярно обновляется и поддерживается разработчиками, что также является большим плюсом для создания и тестирования программного обеспечения. Язык программирования – C++.

Используемые библиотеки: iostream - заголовочный файл с классами, функциями и переменными для организации ввода-вывода в языке программирования C++. fstream - заголовочный файл из стандартной библиотеки C++, включающий набор классов, методов и функций, которые предоставляют интерфейс для чтения/записи данных из/в файл. iomanip - реализует инструменты для работы с форматированием вывода. conio.h – для функции _getch(), которая обеспечивает мгновенный ввод символа и вывода его в консоль. Это обеспечивает быстрое и комфортное перемещение по программе.

Перейдём к переменным, которые встречаются в программе. Начнём с source[] типа char, этот массив символов хранит имя основного рабочего бинарного файла, num типа int хранит количество элементов в основном динамическом массиве, а List[num] типа School является основным динамическим массивом структур School. Сама структура School имеет следующие поля: char FIO[50] – хранит ФИО школьника, char pol[10] -–хранит информацию о поле школьника, char class_name[5] – хранит информацию о классе, в котором учиться школьник, char telephone[20] – содержит номер телефона школьника, double fisic – хранит балл школьника по физике, double math – по математике, double rus – русскому языку, double lit – русской литературе, double rat – средний балл школьника, который рассчитывается как сумма всех баллов по предыдущим предметам, поделённая на 4. Создавать переменные для оценок типа double было решено для того, чтобы избежать предупреждений со стороны Visual Studio 2019 относительно вычисления рассчитываемого поля, которое должно быть в double, так как редко бывает целым. Также это способствует удобству, ведь теперь в базу данных можно вводить балл школьника более точно.

Двигаясь дальше по программе, заметим в функции Menu переменную char act, она нужна для того, чтобы пользователь смог выбрать нужное действие в меню. Переменная bool is_working является проверкой для меню, ведь если is_working примет значение false (а примет она его только тогда, когда пользователь выберет вариант 0 в меню – выход из программы), то функция Menu прекратит своё выполнение. Переменная int CoutNum считает количество выводов, она нужна для того, чтобы сказать пользователю о том, что в функции поиска по его запросу ничего не найдено, или, например, для того чтобы в функции вывода всей информации (Print_List) шапка не выводилась постоянно, после каждого вызова. В функции AddNew есть переменная int kuda, исходя из своего названия, она хранит в себе номер записи, после которой нужно будет добавить новую запись. Также широко распространённым в программе является динамический массив Buffer[num], он служит главным образом для того, чтобы основной массив List изменил своё количество элементов (будь то уменьшение или увеличение, в том числе изменение). В функции Interval переменные double FirstPoint и double SecondPoint отвечают за точку начала необходимого для пользователя интервала и точку его конца соответственно. В функции Search массив символов Searching[50] и int SearchRate нужны для того, чтобы хранить введённую пользователем информацию, в случае, если поле искомой структуры типа char, используется Searching[50] , иначе SearchRate, а bool Similarity проверяет схожесть введённых пользователем данных с нужным полем структуры. При получении false, программа сразу переходит к следующей структуре в массиве. SearchAct типа char хранит выбранное пользователем поле для поиска. Нужно отметить, что последующие переменные типа char ActDelete, Change, WhatChange, SortAct используются для таких же целей (хранение выбора пользователя, для работы в switch). DeleteID типа int хранит номер записи для удаления, а переменная ChangeInfo такого же типа хранит номер записи, которую необходимо изменить. В функции Sort есть важная переменная int ind, которая будет считывать количество индексов из нужного индексного бинарного файла и сравнивать своё значение с num, в результате чего будет определяться переменная int mode – отвечает за режим функции. Эта переменная встречается в нескольких функциях программы и прежде всего помогает сделать программу гораздо более оптимизированной, нежели чем писать несколько функций сортировок и поисков, которые во многом будут схожи, за исключением нескольких моментов. Целочисленный динамический массив Index[num] хранит индексы и в зависимости от режима функции сортировки либо хранит индексы из индексного бинарного файла, либо же индексы, сперва в обычном порядке, сортируются вместе с List и в конце программы нужный порядок индексов записывается в соответствующий индексный бинарный файл.

Ещё нужно отметить, что функция меню, как и многие другие внутренние меню (например, в функциях сортировки, поиска, удаления и т.д.) реализованы через switch,

а не через массив указателей на функцию, главным образом из-за различного количества параметров у функций.

Ввод выбора действия в различных меню (как в главном, так и во внутренних), реализован с помощью _getch(). Это сделано главным образом для удобства и приятности пользованием программой, ведь вместо двух нажатий, становится одно, при этом на экран не выводится ничего лишнего. Здесь есть некоторый минус, ведь если количество возможных действий было 10 и больше, то ввести двухзначные действия было бы невозможно, однако из-за того, что во всех меню и подменю программы менее десяти действия, то использовании _getch() гораздо более предпочтительно.

Выбор бинарного файла как основного рабочего при редактировании и обновлении информации в базе данных благодаря удобству и быстроте, ведь запись и чтение с бинарного файла осуществляется сразу всей структурой, в то время как в текстовый приходилось бы записывать информацию отдельно по полям.

Для форматированного вывода использовалась функция setw(), её использование объясняется желанием сделать качественный, приятный и понятный вывод информации на экран.

Для сортировки использовалась сортировка выбором. Плюс: занимает мало места и крайне проста в реализации. Минус: при огромной количестве данных, долго работает. Но т.к. в базе данных «Школьники», информации вряд ли будет слишком много, учитывая, что в классе обычно до 30 человек, то выбор пал именно на сортировку выбором.

3.2. Разработка модулей

Программа разбита на 4 основных модуля: Kursovaya.cpp, AddTakeOut.h и AddTakeOut.cpp, WorkWithStruct.h и WorkWithStruct.cpp, Sort.h и Sort.cpp и заголовочный файл Shcolniki.h. Разбитие на модули было необходимо для более простого редактирования, контроля и обновления программы.

Модуль Kursovaya.cpp

Является основным модулем программы, который содержит функцию main и Menu. В main объявляется динамический массив структур и вызываются функции WorkingWithFile и Menu.

Функция void Menu (int num, School* &List, char* source)

Входные параметры: int num – размерность массива, School* &List – передача не только динамического массива List, но и ссылка на его указатель, чтобы передать в

другие функции, которые будут вызваны в меню, char* source – массив символов, хранящий имя файла, с которым нужно работать.

Назначение: показать пользователю функционал программы и вызвать желаемую пользователем функцию.

Возвращаемое значение: значений не возвращает.

Модуль Shcolniki.h

Модуль содержит в себе саму структуру. Этот модуль необходим и подключается во всех остальных модулях для того, чтобы данные структуры могли читаться и, соответственно, программа могла работать с этим данными в любых модулях.

Модуль AddTakeOut.h

Содержит в себе прототипы функций, описанных в AddTakeOut.cpp.

Модуль AddTakeOut.cpp

Содержит в себе реализацию всех функций, объявленных в модуле AddTakeOut.h:

1. **Функция void Print_List(int num, School* List, int i, int CoutNum)**

Входные параметры: int num – размерность массива, School* List – передача динамического массива структур (т.к. имя массива есть указатель на его первый элемент, то для этого в параметрах и нужно ставить *), int i – переменная в цикле вывода, нужна для некоторых функций, которым нужно в конкретной итерации вывести только одну структуру (по умолчанию i = 0), int CoutNum – переменная, подсчитывающая количество выводов, нужна для случаев, если нужно по отдельности выводить каждую структуру (например, при сортировке или поиске), поэтому для того, чтобы не выводить каждый раз шапку, и сделана эта переменная (по умолчанию она равна 0)

Назначение: вывод всех структур на экран

Возвращаемое значение: значений не возвращает.

2. **Функция void WorkingWithFile(int &num, School* &List, char* source)**

Входные параметры: int &num – ссылка на размерность массива (для того, чтобы изменения переменной в функции синхронизировались с самой переменной), School* &List – ссылка на указатель на первый элемент динамического массива структур (ссылка нужна для того, чтобы изменить адрес, на который будет указывать именно это ссылка, подобное может пригодиться, если динамический массив нужно уменьшить, увеличить), char*

source – указатель на первый элемент массива символов, хранящий имя основного рабочего бинарного файла.
В дальнейшем при подобных записях будем сокращать, например, вместо указателя на первый элемент, будем использовать просто массив, вместо ссылка на указатель (…) будем использовать просто ссылка на указатель.
Назначение: чтение информации с бинарного файла и запись её в динамический массив структур
Возвращаемое значение: значений не возвращает.

3. Функция void AddNew(School* &List, int &num, char* source)

Входные данные: School* &List – ссылка на указатель, int &num – ссылка на размерность, char* source – массив с именем файла
Назначение: добавление новой записи
Возвращаемое значение: значений не возвращает.

4. Функция void Rewrite(School* List, int num, char* source)

Входные данные: School* List – основной массив структур, int num – его размерность, char* source – массив с именем файла
Назначение: перезапись информации в файл (текстовые и бинарные)
Возвращаемое значение: значений не возвращает.

5. Функция void Interval(School* List, int num)

Входные данные: School* List – основной массив структур, int num – его размерность
Назначение: ввод записей согласно заданному интервалу
Возвращаемое значение: значений не возвращает.

6. Функция void ForSortAfterChange(School* List, int num, const char IndexFile[])

Входные параметры: School* List – основной массив структур, int num – его размерность, const char IndexFile[] – хранит название того индексного бинарного файла, который нужно открыть функции (записи [] и * передают массив в функцию, нет разницы, что именно использовать, [] использовалось здесь прежде всего, чтобы было наглядно отличие, ведь здесь есть const, которая в свою очередь говорит о том, что вызов этой функции не изменит состояние объекта)
Назначение: изменение соответствующего индексного бинарного файла после изменения информации, чтобы функция сортировки не использовала индексы

из него (ведь они могут быть неверными после изменения полей), а отсортировала массив заново.
Возвращаемое значение: значений не возвращает.

Модуль WorkWithStruct.h
Содержит в себе прототипы функций, описанных в модуле WorkWithFile.cpp.

Модуль WorkWithStruct.cpp
Содержит в себе реализацию всех функций, объявленных в модуле WorkWithFile.h:

1. **Функция void Search(Schoool* List, int &num, int mode)**
Входные значения: School* List – основной массив структур, int &num – ссылка на размерность, int mode – режим функции (либо после нахождения соответствий вывести результат на экран, либо удалить из массива, по умолчанию mode = 1 – вывод на экран)
Назначение: поиск и вывод на экран или удаление по любому полю (в зависимости от режима)
Возвращаемое значение: значений не возвращает.

2. **Функция void Delete(School* List, int &num, char* source)**
Входные значения: School* List – основной массив структур, int &num – ссылка на размерность, char* source – массив с именем файла
Назначение: удаление по номеру записи или по любому полю
Возвращаемое значение: значений не возвращает.

3. **Функция void ChangeInfo(School* List, int num, char* source)**
Входные значения: School* List – основной массив структур, int num - размерность, char* source – массив с именем файла
Назначение: изменение всей записи или конкретного поля
Возвращаемое значение: значений не возвращает.

Модуль Sort.h
Содержит в себе прототип функции сортировки, описанной в Sort.cpp.

Модуль Sort.cpp
Содержит в себе реализацию функции сортировки, объявленной в модуле Sort.h

Функция void Sort(School* List, int num, int mode)

Входные значения: School* List – основной массив структур, int num - размерность, int mode – режим функции (первый режим сортирует массив структур и индексов, а второй считывает индексы с индексного бинарного файла и выводит структуры согласно индексам, по умолчанию = 0)
Назначение: сортировка
Возвращаемое значение: значений не возвращает.

4. ТЕСТИРОВАНИЕ

4.1. Описание входных и выходных данных

При запуске программы выполняется чтение данных из бинарного файла и запись их в динамический массив структур. Затем на экране появляется меню для работы с базой данных. В ходе выполнения программы происходит изменение информации, хранимой в массиве и в файле. Существует, однако, несколько ограничений по вводу, которые следуют из 3.1: поле с ФИО школьника может содержать только 50 символов, пол – 10, класс – 5, номер телефона – 20. Оценки объявлены как double, поэтому вводить символы сюда будет некорректным.

Хоть чтение и происходит из бинарного файла, и основная работа введётся в нём, но программа также после любого изменения информации записывает всё не только в бинарный, но и в текстовый, поэтому содержимое и в бинраном, и в текстовом идентично. Всё это для наглядности, поэтому воспользуемся этим и ниже будет приведён скриншот начальных данных из текстового файла (рис.4.1):

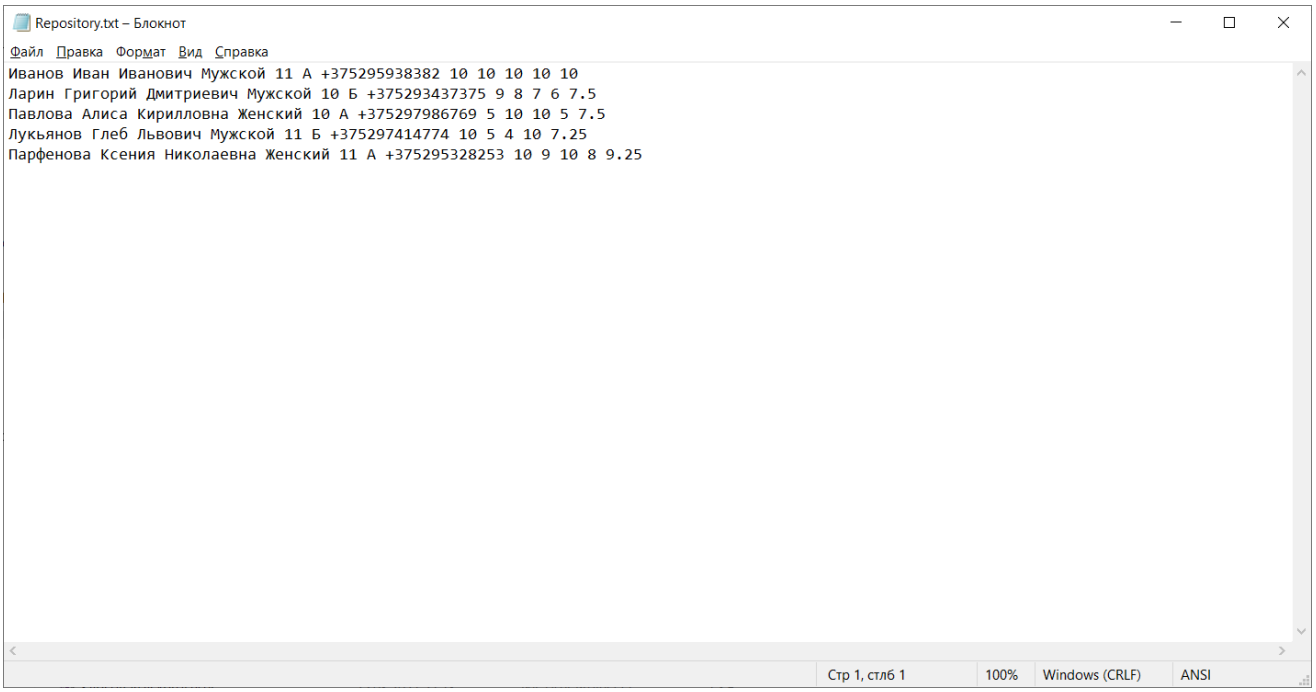

Рисунок 4.1 - Файл с информацией

Как видно из скриншота, информация на русском языке. Она может быть и на латинице, здесь никаких ограничений нет, и программа будет корректно работать как с русским, так и с латинским алфавитами.

4.2. Результаты тестирования

При запуске программы на 3 секунды (с помощью Sleep()), выводится приветственное сообщение:

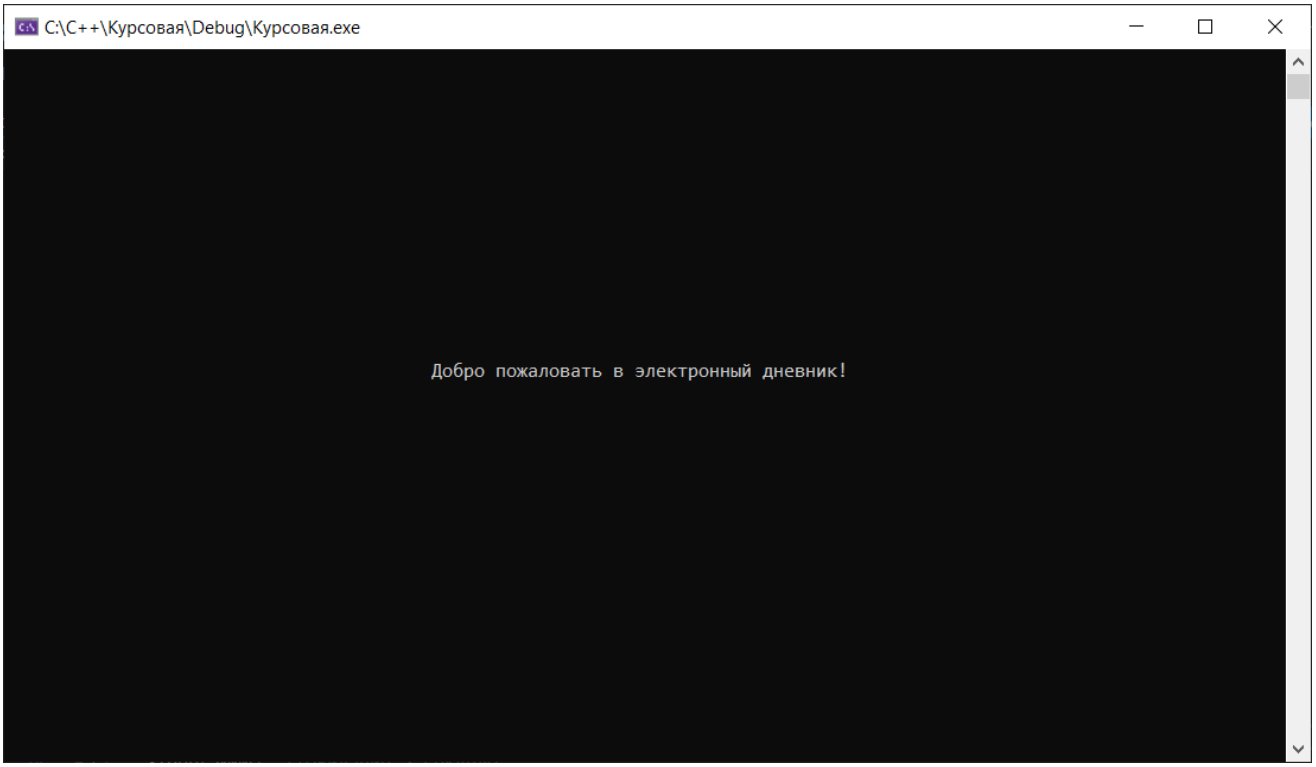

Рис. 4.2 – Приветственное сообщение

Далее программа автоматически начнёт считывание данных из основного бинарного файла и создаст для этого динамический массив соответствующего размера.

Пользователь ли увидит лишь следующее действие – вызов меню.

Там пользователю будет предложено множество вариантов действий, из которых он сможет выбрать подходящее.

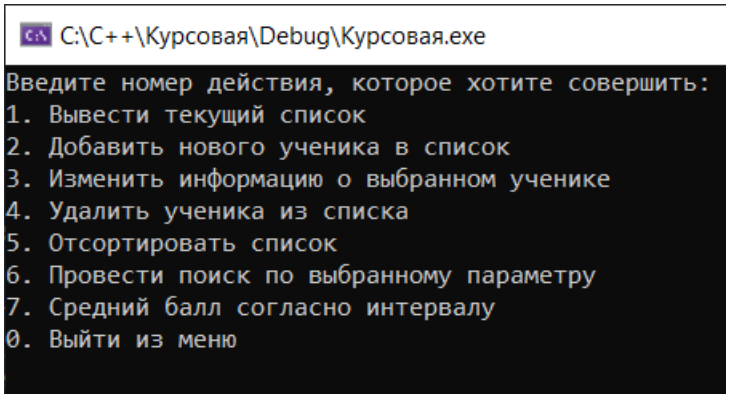

Рис. 4.3 – Основное меню программы

Благодаря функции _getch(), пользователь может нажать на любую кнопку и мгновенно, без вывода ненужной информации на экран, сразу же начать совершать нужное действие.

Если пользователь нажмёт номер действия, которого нет в данном меню, то программа выдаст ошибку:

Рис. 4.4. – Ошибка при неверном номере действия

После этого пользователь может нажать любую клавишу и вернуться в меню.

Проверим функционал программы:

Пусть пользователь сперва захочет вывести весь список на экран и выберет вариант 1 в меню:

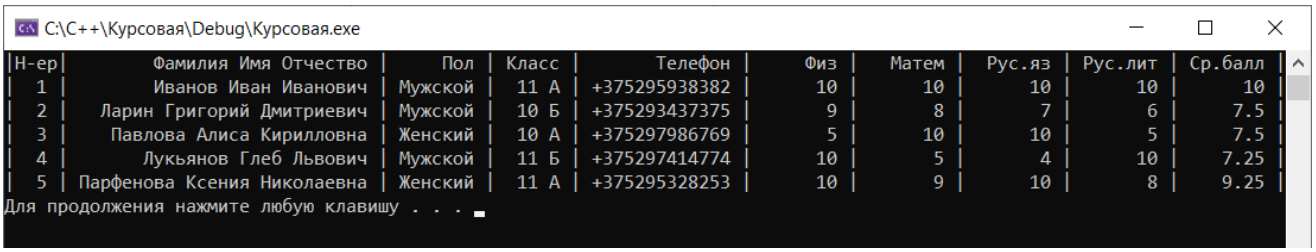

Рис. 4.5 – Вывод всей информации на экран

Снова нажав любую клавишу, пользователь вернётся в меню (такое возвращение будет после каждой функции).

Посмотрим пункт 2 в меню:

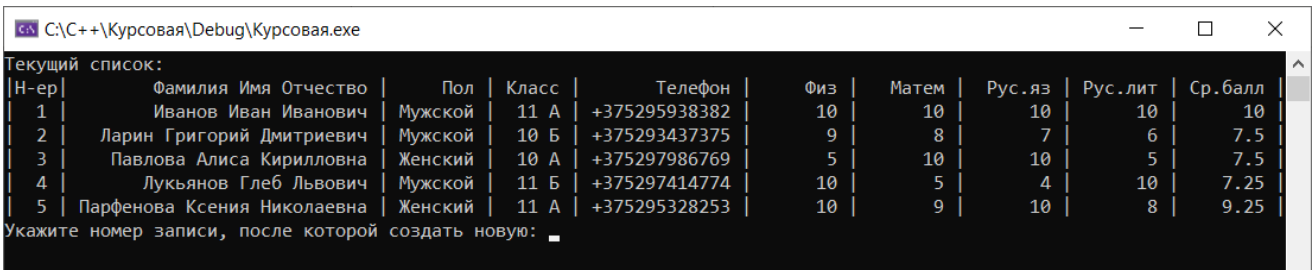

Рис. 4.6 – Вывод информации перед добавлением новой записи

Для удобства выводится весь список, чтобы пользователю было легче выбрать, где ему необходимо создать новую запись. Также это сделано, чтобы если пользователь

захотел сразу добавить новую запись, а не сначала просматривать весь список, то он бы смог сразу сориентироваться в текущей информации.

Если будет введён неверный номер записи (для примера это 6), то будет выдана ошибка:

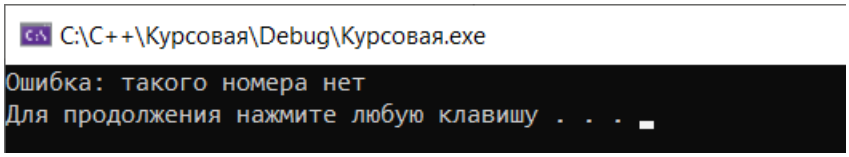

Рис. 4.7 – Ошибка при добавлении

Как только пользователь выберет корректный номер записи, после которой нужно создать новою запись, для примера будет введено 0, чтобы показать, что программа в таком случае создаст запись в начале, начнётся само добавление. Нужно будет поочерёдно заполнить каждое поле структуры (чтобы не затягивать, скриншот ниже будет показывать уже введённую новую информацию):

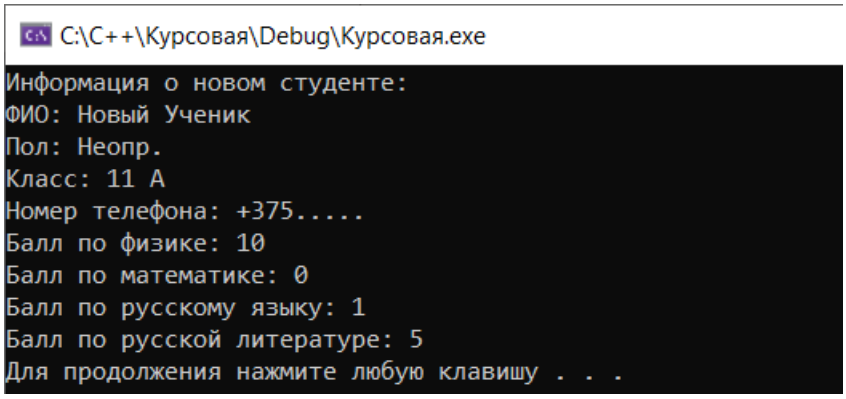

Рис. 4.8 – Введённая новая информация

Как видно из скриншота, программа корректно принимает записи на русскому языке, а также без проблем воспринимает пробелы в записях, если они нужны пользователю.

После того, как была добавлена новая запись, нужно удостовериться, что всё было сделано правильно, для этого вернёмся в меню и снова выведем весь список на экран:

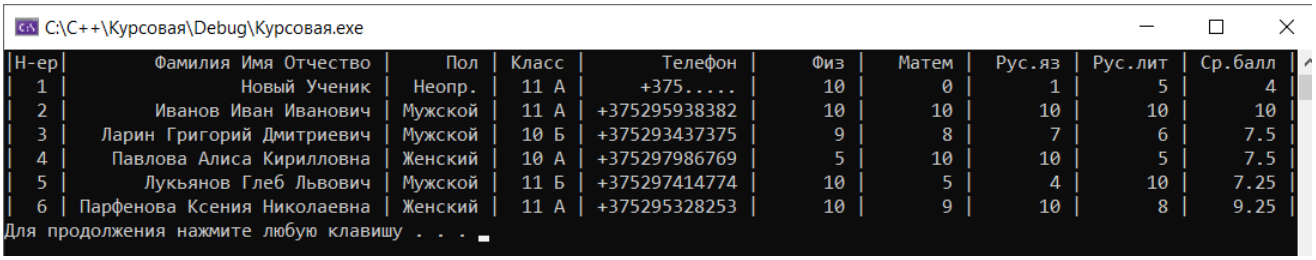

Рис. 4.9 – Проверка добавления новой информации

Новая информация была добавлено корректно в начало списка, номера учеников изменились и показывают их действительное значение в таблице, а средний балл, посчитанный автоматически, также верно отражает действительную успеваемость Нового Ученика.

Теперь рассмотрим возможность изменить информацию. Выберем пункт 3 в меню:

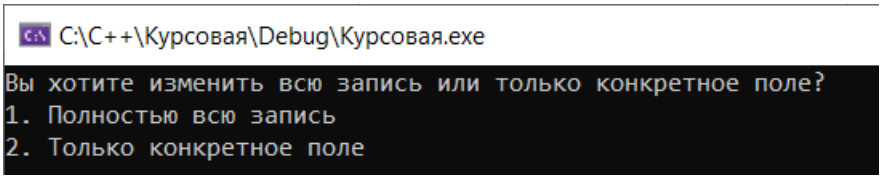

Рис. 4.11. – Меню изменения

Здесь пользователю нужно выбрать предпочтительный вариант изменения информация об ученике. Сначала выберем 1.

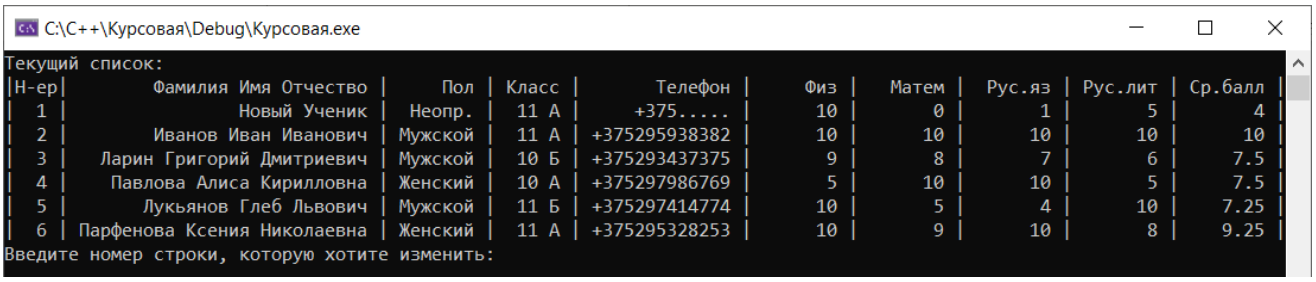

Рис. 4.12 – Перед изменением всей записи

Снова программа выводит всю информацию на экран и просит пользователя выбрать номер записи, которую нужно изменить.

И снова, в случае если пользователь введёт неверный номер записи, будет выведена ошибка:

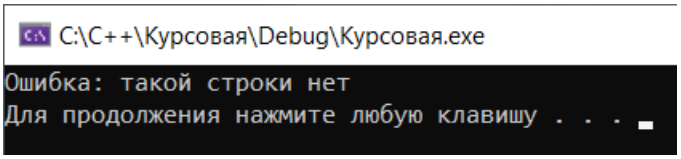

Рис. 4.13 – Ошибка при изменении

Вернёмся в меню, снова выберем изменение информации и выберем пункт 1, так мы вернёмся к рис.4.12. Теперь выберем пункт 1 (чтобы изменить информацию о нашем добавленном недавно ученике). Снова нужно будет поочерёдно заполнить каждое поле, скриншот ниже покажет уже заполненные поля:

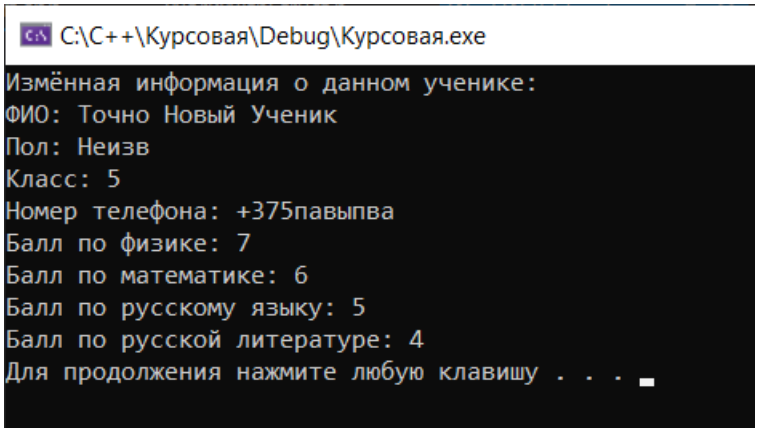

Рис. 4.14 – Изменённая запись

Проверим изменение записи, для этого вернёмся в меню и выберем пункт 1.

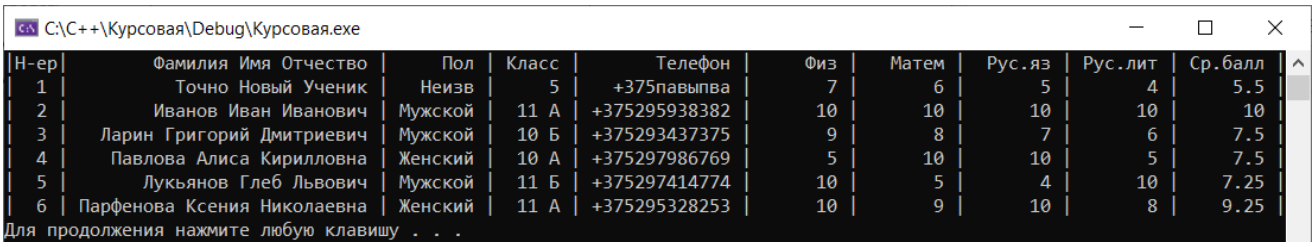

Рис. 4.15 – Проверка изменения всей записи

Теперь рассмотрим изменение по полю. Для этого выберем пункт 3 в главном меню и пункт 2 в меню изменения. То, что будет после этого, идентично Рис. 4.12. Ошибка при неверном номере такая же, как и рис. 4.13. Предположим мы хотим изменить пол нашего добавленного ученика, для этого выберем его номер.

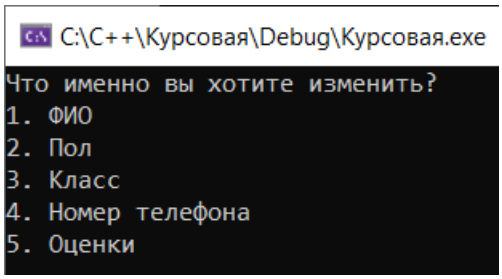

Рис. 4.16 – Меню изменения по полю

Здесь необходимо выбрать поле, которое пользователь захочет изменить. При неверном номере действия выводится ошибка и пользователя возвращают в меню. Нам же нужно поменять пол, поэтому нажмём на 2. После этого нас попросят ввести выбранное поле.

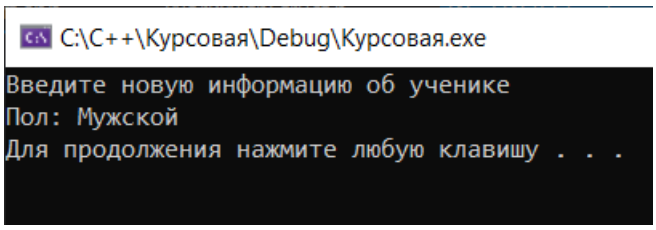

Рис. 4.17 – Изменённое поле

Проверим правильность и выведем весь список на экран:

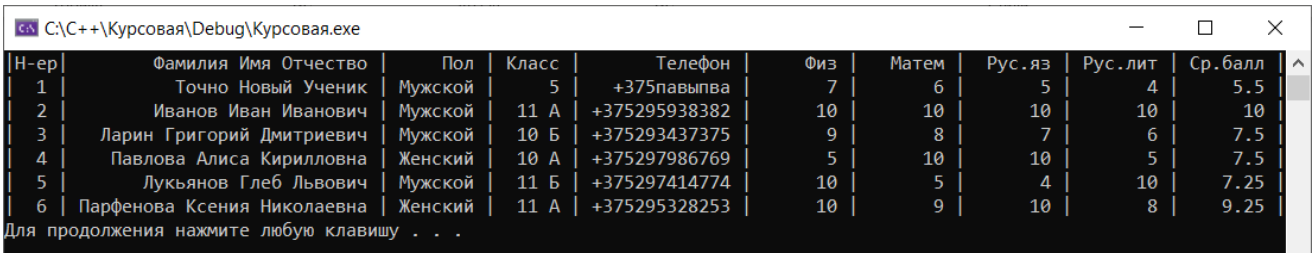

Рис. 4.18 – Проверка изменения по полю

Как видно из скриншота, изменение по полю и по записи работает корректно.

Теперь рассмотрим удаление. Выберем пункт 4. Перед нами откроется меню удаления, схожее с меню, что было в изменении:

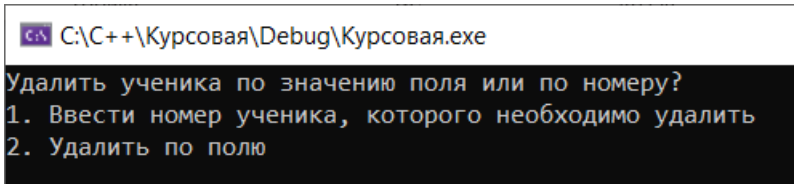

Рис. 4.19 – Меню удаления

Если будет выбрано неверное действие, будет выведена ошибка и пользователь будет возвращён в меню. Чтобы не загромождать тестирование скриншотами об ошибках, об этом и дальше будет просто уточняться.

Выберем удаление по записи:

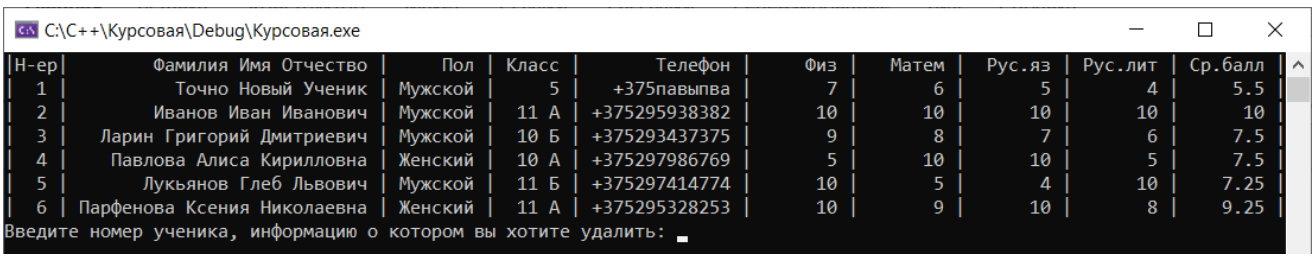

Рис. 4.20 – Выбор номер записи для удаления

Выберем номер 1, чтобы удалить добавленного нами ранее ученика. Проверим удаление, выбрав в главном меню пункт 1:

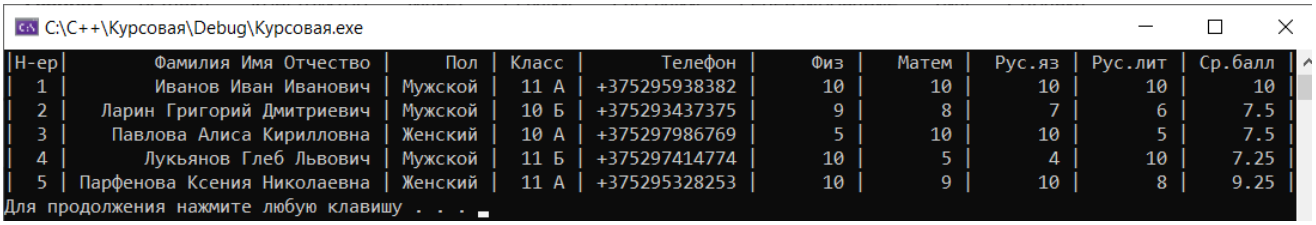

Рис. 4.21 – Проверка удаления по записи

Как видно на скриншоте рис. 4.21, удаление произошло успешно. Теперь проверим удаление по полю. Там снова выведется список, как и в рис. 4.20, но также будет представлено меню выбора поля для удаления, как и в рис. рис. 4.16, только вместо одного поля оценок, там несколько полей, отвечающих за каждую оценку отдельно. Выберем поле с оценками по математике. Нас попросят ввести информацию, а мы введём 5:

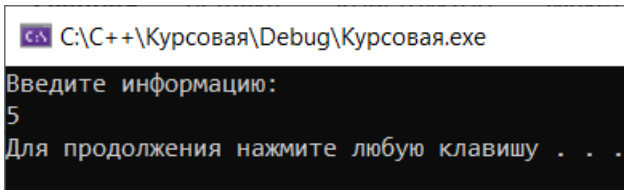

Рис. 4.22 – Удаление по полю

Таким образом, должны удалиться все записи, у которым в поле оценки по математике есть значение 5. Проверим это с помощью вывода:

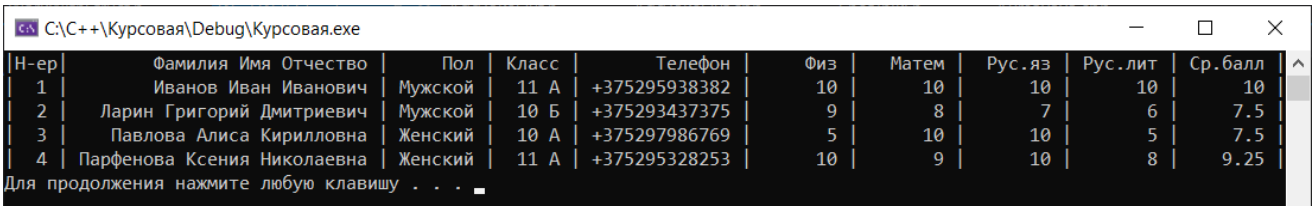

Рис. 4.23 – Проверка удаления по полю

Как видим, удаление по полю прошло успешно.

Следующим действием в главном меню будет сортировка. Выберем пункт 5 в главном меню. У нас снова выведется весь список и появится меню сортировки, где пользователя спросят, по какому полю он хочет сортировать. Меню сортировки такое же, как и в рис. 4.16. Если пользователь выберет номер действия, которого нет, то программа выдаст ошибку и вернёт его в меню. Выберем сортировку по среднему

баллу. Для этого нажмём в меню на 5. (Оценки), и нам откроется дополнительное меню оценок:

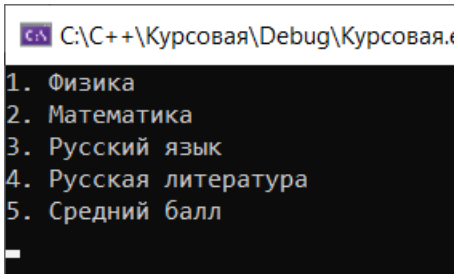

Рис. 4.24 – Меню оценок

Выберем и здесь номер 5. После чего будет выведен отсортированный список:

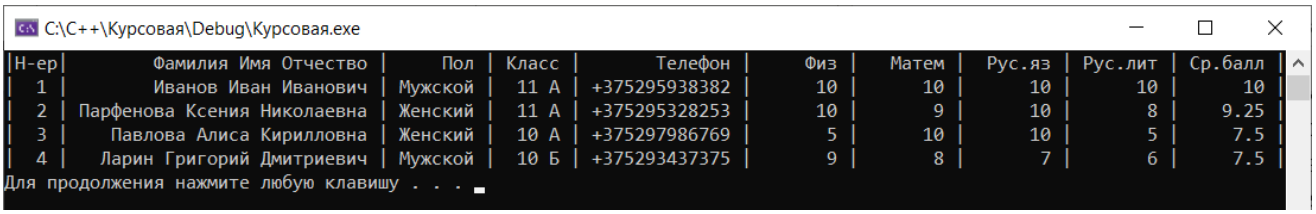

Рис. 4.25 – Сортировка по среднему баллу

Однако нам не нужно, чтобы программа дальше работала с отсортированным массивом данных, поэтому нужно, чтобы программа вернулась к прежнему порядку. Это реализовано с помощью повторного чтения информации из файла после сортировки, ведь сортируя массив, мы не записываем его ни в текстовый, ни в бинарный файл. Только в отдельный индексный бинарный файл для каждого поля, но это лишь для того, чтобы ускорить сортировку в следующий раз. Чтобы убедиться в том, что программа вернулась к прежнему порядку, вернёмся в меню и выведем всю информацию на экран:

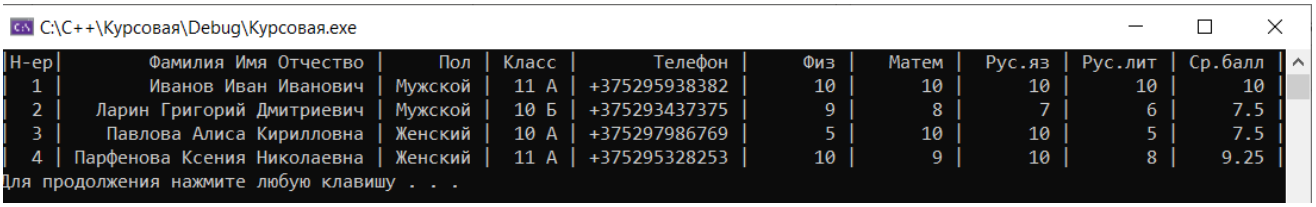

Рис. 4.26 – Проверка данных после сортировки

Как видно из скриншота, программа вернулась к прежнему порядку структур, а значит, можно продолжать тестирование.

Следующим на очереди будет поиск. Для этого выберем пункт 6 в главном меню. Программа выведет меню поиска и попросит выбрать то поле, по которому его

нужно будет провести. При нажатии на неверный номер действия, программа выведет ошибку и вернёт пользователя в меню. Для примера мы возьмём поиск по классу. Выбрав его, нас попросят ввести информацию, как это было на рис. 4.22. Введём туда «11 А». После этого будет выведен результат поиска:

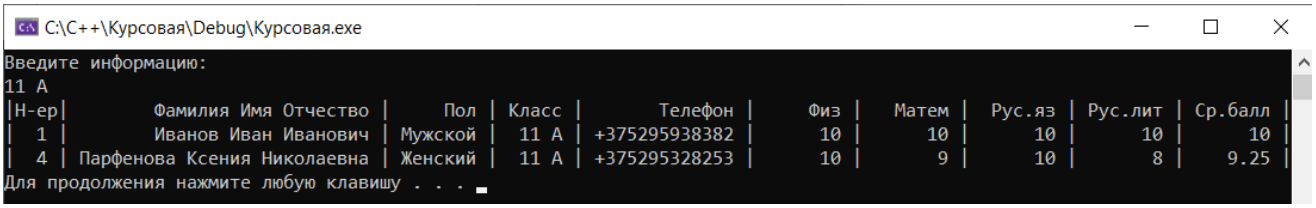

Рис. 4.27 – Результат поиска

Как видно из скриншота, все ученики в 11 А классе были выведены на экран. Значит поиск работает верно. В случае, если информация будет введена некорректно или же совпадений по введённым пользователем данных обнаружено не будет, программа выдаст сообщение о том, что по запросу пользователя ничего найдено не было и вернёт его в главное меню.

Из всего функционала программы осталось лишь проверить вывод согласно интервалу. Для этого выберем пункт 7 в главном меню. Пользователя попросят ввести начальное и конечное значения нужного ему интервала среднего балла. Для примера введём интервал от 5 до 10:

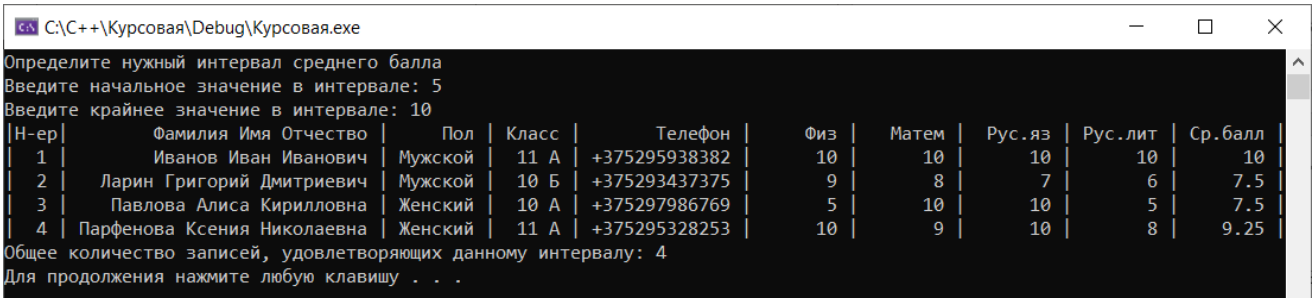

Рис. 4.28 – Результаты вывода по интервалу

Исходя из скриншота, можно заявить, что вывод по интервалу также работает корректно. Также стоит отметить, что если пользователь введёт некорректные данные, или же записей, лежащих в заданном интервале, обнаружено не будет, то программа выдаст сообщение, что никаких записей найдено не было:

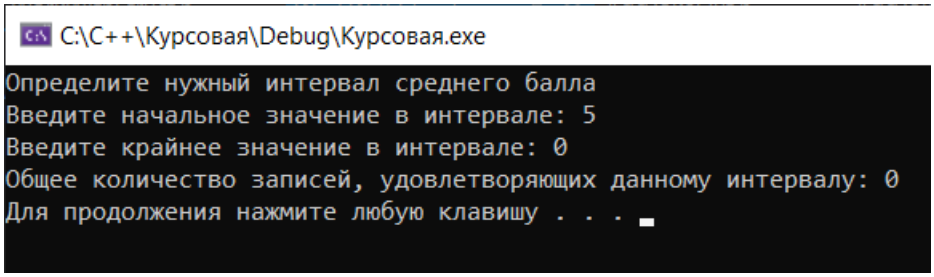

Рис. 4.29 – Ошибка при неверном интервале

Весь функционал программы был рассмотрен. Теперь нам нужно из неё выйти. Для этого в главном меню нажмём на 0 – Выход из программы.

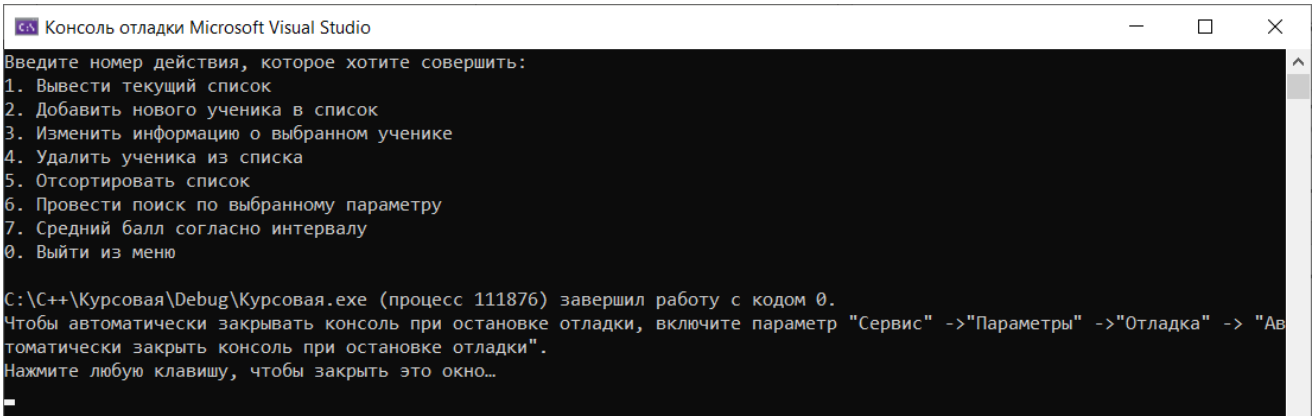

Рис. 4.30 – Выход из программы

Программа завершила свою работу с кодом 0. Никаких ошибок при компиляции или при работе программы не было. Тестирование успешно.

ЗАКЛЮЧЕНИЕ

Результатом является работающая программа, содержащая в себе базу данных «Школьники». Удобный интерфейс, возможность записи данных в файл упрощает работу с большим количеством записей, что является основной целью данной курсовой работы.

Все условия и задачи курсовой работы были успешно решены, программа функционирует исправно. Весь требуемый функционал присутствует в программе, она в полной мере обеспечивает требования пользователя. Возникновение ошибок и исключений сведено к минимуму. Программа простая и удобная в эксплуатации и готова для использования.

В ходе выполнения работы были практически закреплены знания о работе с массивами структурированных данных, инструментами файлового ввода и вывода, основными языковыми средствами языка C++. Опыт в разработке программы, обрабатывающей структуры данных, важен для каждого программиста и поможет в разработке более сложных программ в будущем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. C/C++ Программирование на языке высокого уровня [Электронный ресурс]. – Режим доступа:https://classroom.google.com/u/1/c/MjA1NTIyMjczNjc0/m/MzE0ODY0NjQwMjMz/details - Дата доступа: 04.05.2021.

2. Объектно-ориентированное программирование в C++ [Электронный ресурс]. – Режим доступа: https://vk.com/doc175602161_570916906?hash=283d1eb04062211b2f&dl=95eda26bfa5d45e1c8 – Дата доступа: 05.05.2021.

3. C/C++ Программирование на языке высокого уровня. Структурное программирование. Практикум [Электронный ресурс]. – Режим доступа: https://classroom.google.com/u/1/c/MjA1NTIyMjczNjc0/m/MzE0ODY0NjQwMjMz/details - Дата доступа: 04.05.2021.

4. Основы C++. Программирование для начинающих [Электронный ресурс]. – Режим доступа: https://www.youtube.com/playlist?list=PLQOaTSbfxUtCrKs0nicOg2npJQYSPGO9r – Дата доступа: 06.06.2021

5. ГОСТ 19.504 – 79. ЕСПД. Руководство программиста. Требования к содержанию и оформлению.

6. ГОСТ 2.105-95. ЕСКД. Общие требования к текстовым документам