содержание

ВВЕДЕНИЕ ………………………………………………………………………...2

1. ПОСТАНОВКА ЗАДАЧИ ………………………………………………………3

2. РАЗАРАБОТКА АЛГОРИТМОВ……………………………………………….4

3. РАЗРАБОТКА ПРОГРАММЫ………………………………………………….7

3.1. Выбор средств программирования …………………………………………..7

3.2. Разработка функций программы……………………………………………..7

4. ТЕСТИРОВАНИЕ ……………………………………………………………...10

4.1. Описание входных и выходных данных………………………………….....10

4.2. Результаты тестирования……………………………………………………..11

ЗАКЛЮЧЕНИЕ…………………………………………………………………….21

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ………………………………22

ПРИЛОЖЕНИЕ А ТЕКСТ ПРОГРАММЫ

ПРИЛОЖЕНИЕ Б ГРАФИЧЕСКИЙ МАТЕРИАЛ

# ВВЕДЕНИЕ

Целью данной курсовой работы является помощь в ведении учета над учащимися щкол, колледжей и других учреждений образования. Предметом работы является написание программы, содержащей в себе базу данных школьников. Программа должна предоставлять доступ к информации о классах и их составах, позволять добавление новых и удаление ранее созданных элементов, производить изменения введённых данных, осуществлять поиск среди добавленных элементов, сортировку, выполнять подсчёт количества элементов в базе данных приложения, реализовывать вычисление минимальных, средних и максимальных значений как отдельных элементов, так и их совокупностей

Для создания данной программы необходимо пройти ряд этапов: анализ постановки задачи, разработка структуры ПО, разработка алгоритмов функционирования, тестирование и анализ результата проектирования, построение и тестирование программы.

Проектирование программы является одной из наиболее сложных и ответственных задач, связанных с созданием базы данных. В результате ее решения должны быть определены содержание, эффективный для всех ее будущих пользователей способ организации данных и инструментальные средства управления данными.

1. ПОСТАНОВКА ЗАДАЧИ

Необходимо разработать программный комплекс по обработке заданного множества данных. Множество данных представляет собой массив записей. Программа должна быть реализована в виде нескольких модулей. Требуется обеспечить реализацию следующих запросов к заданному множеству структурированной информации:

1. Ввод информации из текстового файла в массив указателей на записи.

2. Добавление новых элементов в конец массива.

3. Просмотр всех элементов массива.

4. Вывод информации из массива в файл.

5. Сортировка записей по выбранному полю с созданием индексных файлов.

6. Корректировка полей выбранного элемента.

7. удаление выбранного элемента.

8. Поиск записи по значению ключевого поля.

9. Вставка новой записи перед выбранной.

10. Выборка записей и вычисление среднего на множестве тех элементов, которые попадают в заданный диапазон по заданному полю (поле числового типа).

11. Фильтр записей по условию (значение выбранного поля больше или меньше заданного).

2. РАЗРАБОТКА АЛГОРИТМОВ

Этот этап один из самых важных. После того, как мы осуществили постановку задачи, для дальнейшей работы нам будет необходимо разработать алгоритмы. Разработка алгоритма – метод для создания математического способа решения проблем, т.е. на основе выбранного метода записывается последовательность действий, приводящих к решению задачи. Успешная разработка алгоритма позволяет избежать многих ошибок, поскольку именно на этом этапе определяется логика будущей программы.

Алгоритм ввода информации из файла:

1. Открываем файл для чтения.

2. Читаем строки.

3. Каждые пять строк из файла заносим в массив.

4. Преобразовываем прочитанные строки в поля структуры.

5. Добавляем элемент в конец массива.

6. Увеличиваем счетчик элементов массива.

7. Когда будет достигнут конец файла, файл закрывается, и работа завершается.

Алгоритм добавления нового элемента:

1. Запрашивается информация о школьнике.
2. Открывается файл.
3. Структура записывается в файл.
4. Файл закрывается.

Алгоритм вывода элементов массива:

1. Запускаем цикл для всех элементов массива.

2. Производим вывод полей структуры на экран в одну строку.

Алгоритм корректировки полей элементов массива:

1. Выбираем номер элемента, для корректировки полей.

2. Выбираем поле для корректировки.

3. На основании выбора вносим новое значение выбранного поля.

Алгоритм поиска записи по значению ключевого поля:

1. Выбираем поле для поиска.

2. Вводим значение для поиска.

3. Запускаем цикл для всех элементов массива.

4. Функция возвращает номер структуры.

Алгоритм вывода информации в текстовый файл:

1. Открываем файл для записи.

2. Запускаем цикл для всех элементов массива.

3. Каждое поле элемента массива преобразуется в символьное значение и записывается в строку файла.

4. Закрываем файл.

Алгоритм удаления выбранного элемента:

1. Запускаем цикл для нахождения элемента в массиве.

2. Удаляем элемент из массива.

3. Сдвигаем все элементы массива вверх.

4. Уменьшаем счетчик элементов массива.

Алгоритм сортировки записей по выбранному полю:

1. Выбор поля для сортировки

2. Последовательно сравниваем элементы попарно.

3. Если порядок в паре неверный, выполняется обмен элементов.

4. Проходы по массиву повторяются n-1 раз, пока на очередном проходе не окажется, что обмены больше не нужны.

Алгоритм вставки новой записи перед выбранной:

1. Открываем файл.

2. Выбираем элемент массива.

3. Записываем элементы массива до выбранного.

4. Вводим новую структуру.

5. Записываем ее в файл.

6. Записываем оставшиеся элементы массива в файл.

7. Закрываем файл.

Алгоритм выборки записей и вычисления среднего на множестве тех элементов, которые попадают в заданный диапазон по заданному полю (поле числового типа):

1. Вводим границы диапазона.

2. Выбираем поле, по которому будем считать среднее значение.

3. Организовываем цикл для считывания суммы значений выбранного поля всех элементов.

4. Делим полученное значение на количество элементов массива в заданном диапазоне.

5. Выводим полученное значение.

Алгоритм фильтрации записей по условию (значение выбранного поля больше или меньше заданного):

1. Выбираем поле.

2. Выбираем первый параметр для фильтра.

3. Вводим второй параметр для фильтра.

4. Запускаем цикл для всех элементов массива.

3. Выводим элементы массива, удовлетворяющие условию, заданному ранее.

3. РАЗРАБОТКА ПРОГРАММЫ

3.1. Выбор средств программирования

Используемая среда разработки VisualStudio, OC Windows 10, язык программирования – С++.

Используемые библиотеки: iostream — [заголовочный файл](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D0%BE%D0%BB%D0%BE%D0%B2%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%84%D0%B0%D0%B9%D0%BB) с классами, функциями и переменными для организации ввода-вывода в языке программирования [C++](https://ru.wikipedia.org/wiki/C%2B%2B); fstream — заголовочный файл из стандартной библиотеки C++, включающий набор классов, методов и функций, которые предоставляют интерфейс для чтения/записи данных из/в файл; windows.h — этот файл содержит множество определений типов данных, макросов, прототипов функций, констант и т. д.; cstring — класс с методами и переменными для организации работы с символами в языке программирования [C++](https://ru.wikipedia.org/wiki/C%2B%2B); string — класс с методами и переменными для организации работы с символами в языке программирования [C++](https://ru.wikipedia.org/wiki/C%2B%2B).

3.2. Разработка функций программы

**Функция Schooler\* fromFile(string path, int st)**

Входные параметры: объект класса string, количество элементов массива записей.

Назначение: извлечение из файла информации и занесение ее в структуру.

Возвращаемое значение: объект класса Schooler.

**Функция void add(string path)**

Входные параметры: объект класса string.

Назначение: добавление информации в файл.

Возвращаемое значение: -.

**Функция void out(Schooler\* data, int st)**

Входные параметры: массив объектов класса Schooler, номер элемента массива записей.

Назначение: вывод элемента массива по номеру.

Возвращаемое значение: -.

**Функция void toFile(string path, Schooler\* data, int st)**

Входные параметры: объект класса string, массив объектов класса Schooler, количество элементов массива записей.

Назначение: запись в файл массива структур.

Возвращаемое значение: -.

**Функция void sort(Schooler\* data, int st, string pathi)**

Входные параметры: массив объектов класса Schooler, количество элементов массива записей, объект класса string.

Назначение: сортировка элементов массива.

Возвращаемое значение: -.

**Функция void correct(Schooler\* data)**

Входные параметры: массив объектов класса Schooler.

Назначение: изменение элемента массива.

Возвращаемое значение: -.

**Функция void deleting(Schooler\* data, int st)**

Входные параметры: массив объектов класса Schooler, количество элементов массива записей.

Назначение: удаление элемента массива.

Возвращаемое значение: -.

**Функция void insert(Schooler\* data, int st, string path)**

Входные параметры: массив объектов класса Schooler, количество элементов массива записей, объект класса string.

Назначение: запись элемента массива в файл перед выбранным.

Возвращаемое значение: -.

**Функция void filter(Schooler\* data, int st)**

Входные параметры: массив объектов класса Schooler, количество элементов массива записей.

Назначение: вывод на экран элементов массива, удовлетворяющих выбранным параметрам.

Возвращаемое значение: -.

**Функция void average(Schooler\* data, int st)**

Входные параметры: массив объектов класса Schooler, количество элементов массива записей.

Назначение: вывод на экран среднего значения выбранного поля элементов массива в выбранном диапазоне.

Возвращаемое значение: -.

**Функция void find(Schooler\* data, int st)**

Входные параметры: массив объектов класса Schooler, количество элементов массива записей.

Назначение: поиск элемента массива по выбранному полю.

Возвращаемое значение: номер элемента массива.

**Функция int f(char\* s)**

Входные параметры: массив типа char

Назначение: перевод информации из символьного типа в целочисленный.

Возвращаемое значение: число типа int.

4. ТЕСТИРОВАНИЕ

4.1. Описание входных и выходных данных

При запуске программы выполняется чтение данных из файла и запись их в массив. Затем на экране появляется меню для работы с нашим массивом. В ходе выполнения программы происходит изменение информации, хранимой в массиве и в файле.

Файл с информацией forKurs.txt (см.рисунок 4.1):

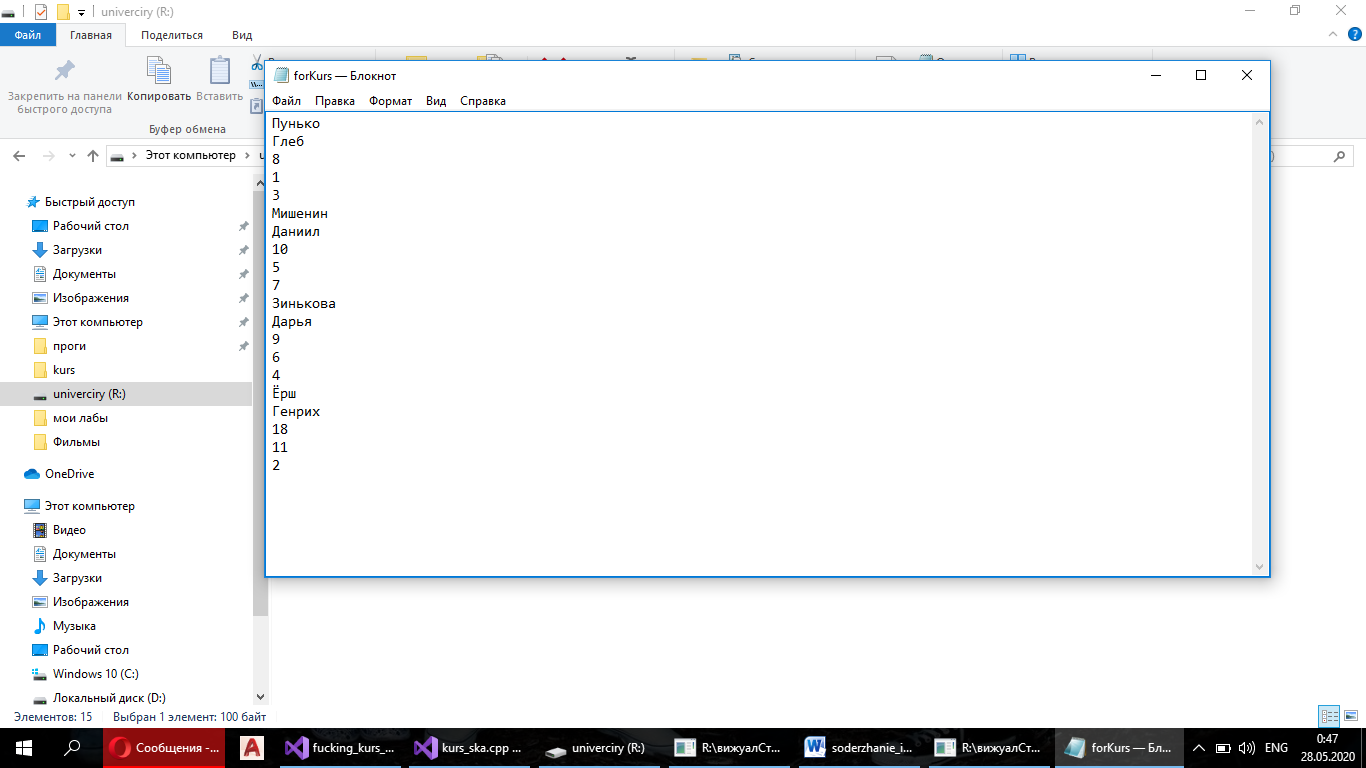


Рисунок 4.1 - Файл с информацией

4.2. Результаты тестирования

При запуске программа должна считать количество структур в файле “forKurs.txt”, считывать данные из файла и заносить их в массив структур. Затем нам предоставляется возможность выбора действия над программой (см. рисунок 4.2)

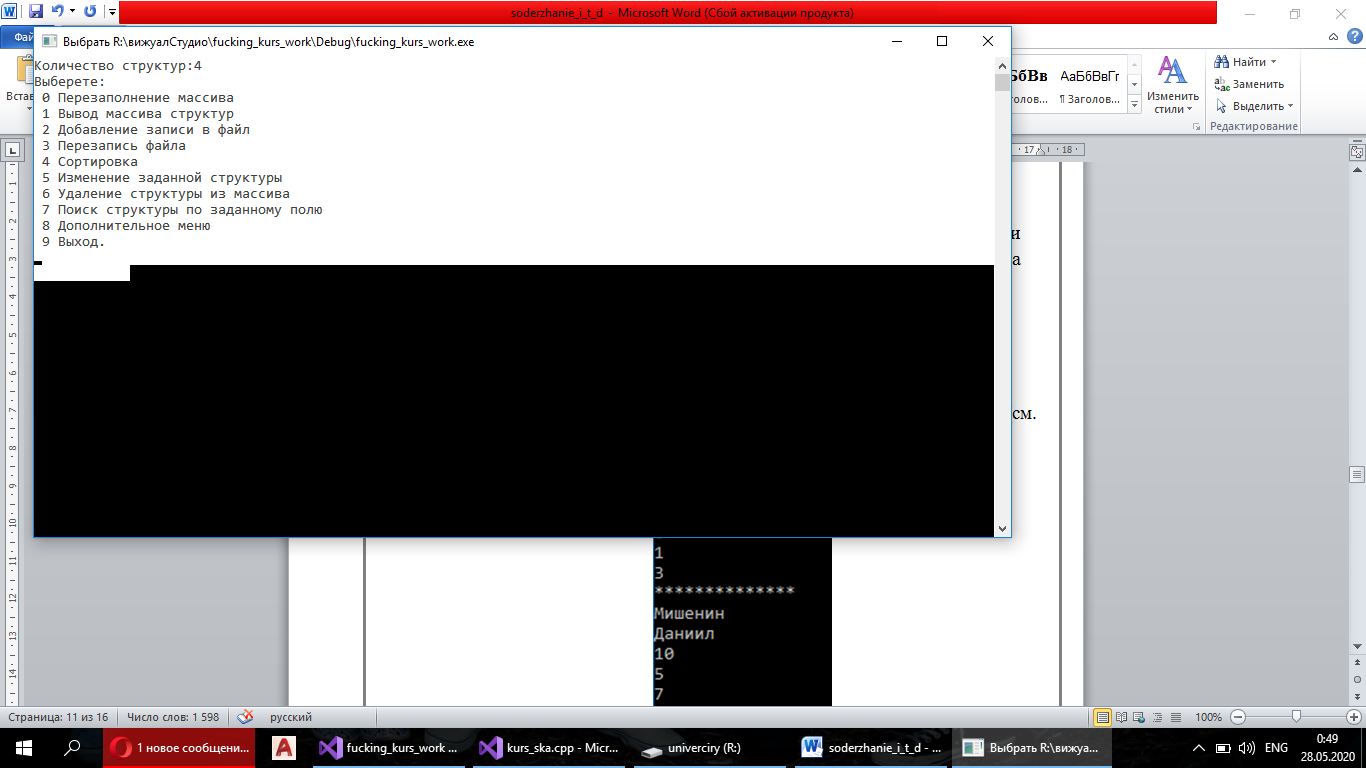


Рисунок 4.2

В пункте 0 осуществляется перезаполнение массива(см. рисунок 4.3)

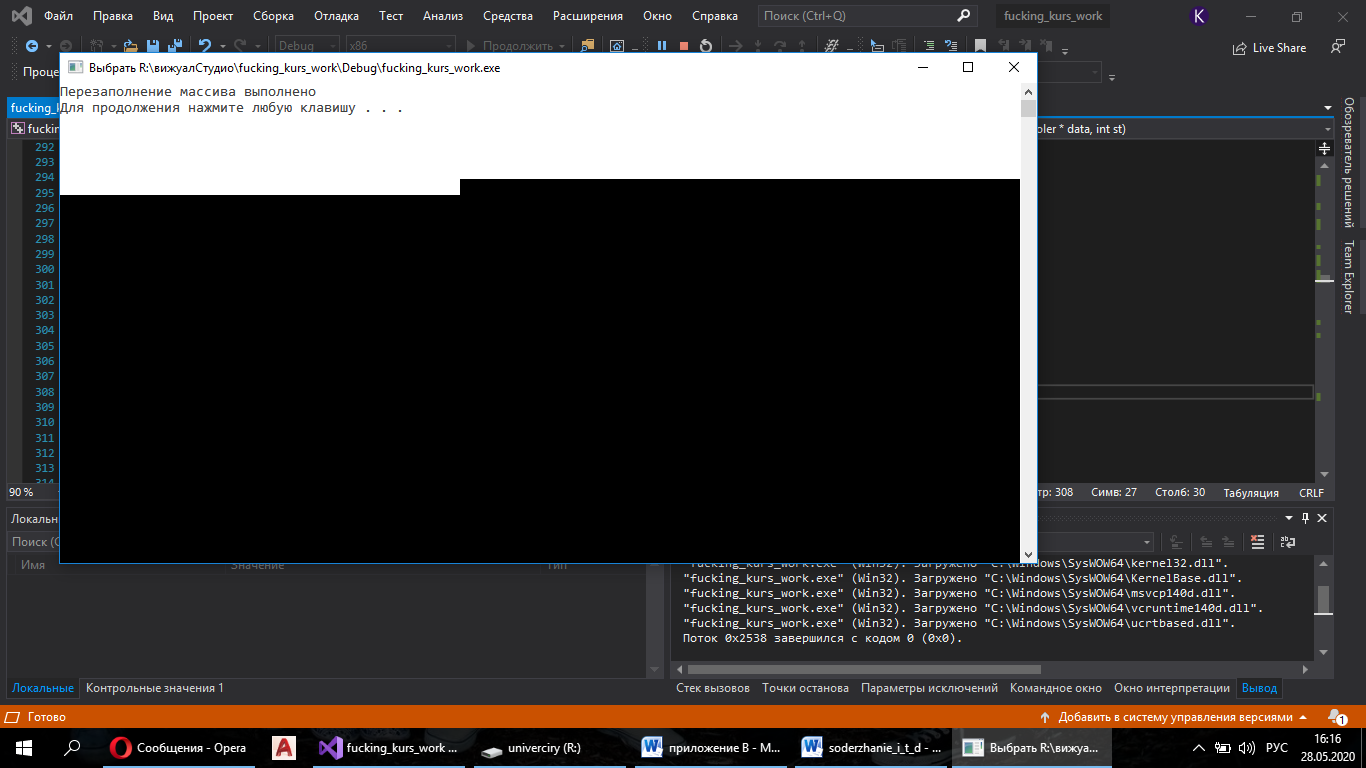


Рисунок 4.3

В пункте 1, осуществляется вывод данных массива структур на экран (см. рисунок 4.4)

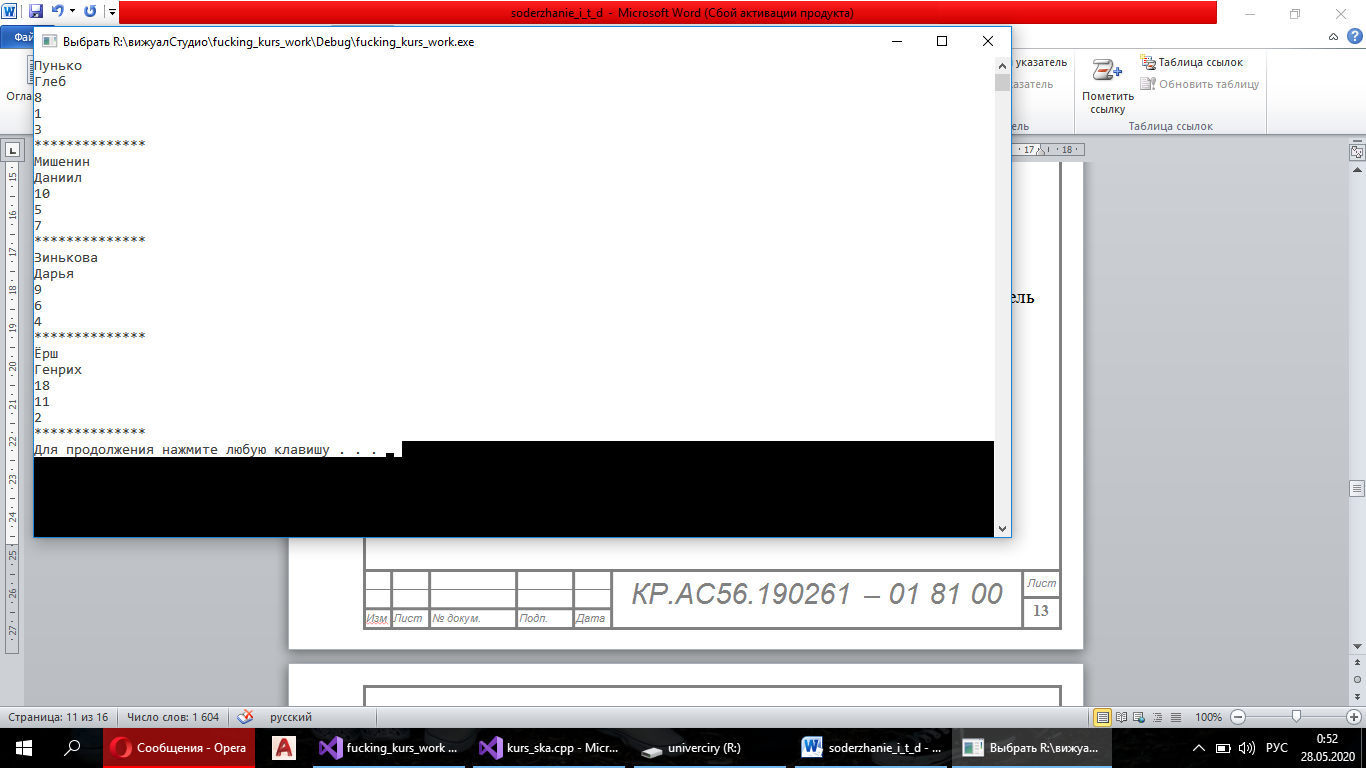


Рисунок 4.4

В пункте 2 осуществляется добавление элемента в массив, пользователь вводит элемент и заполняет поле для него (см. рисунок 4.5)

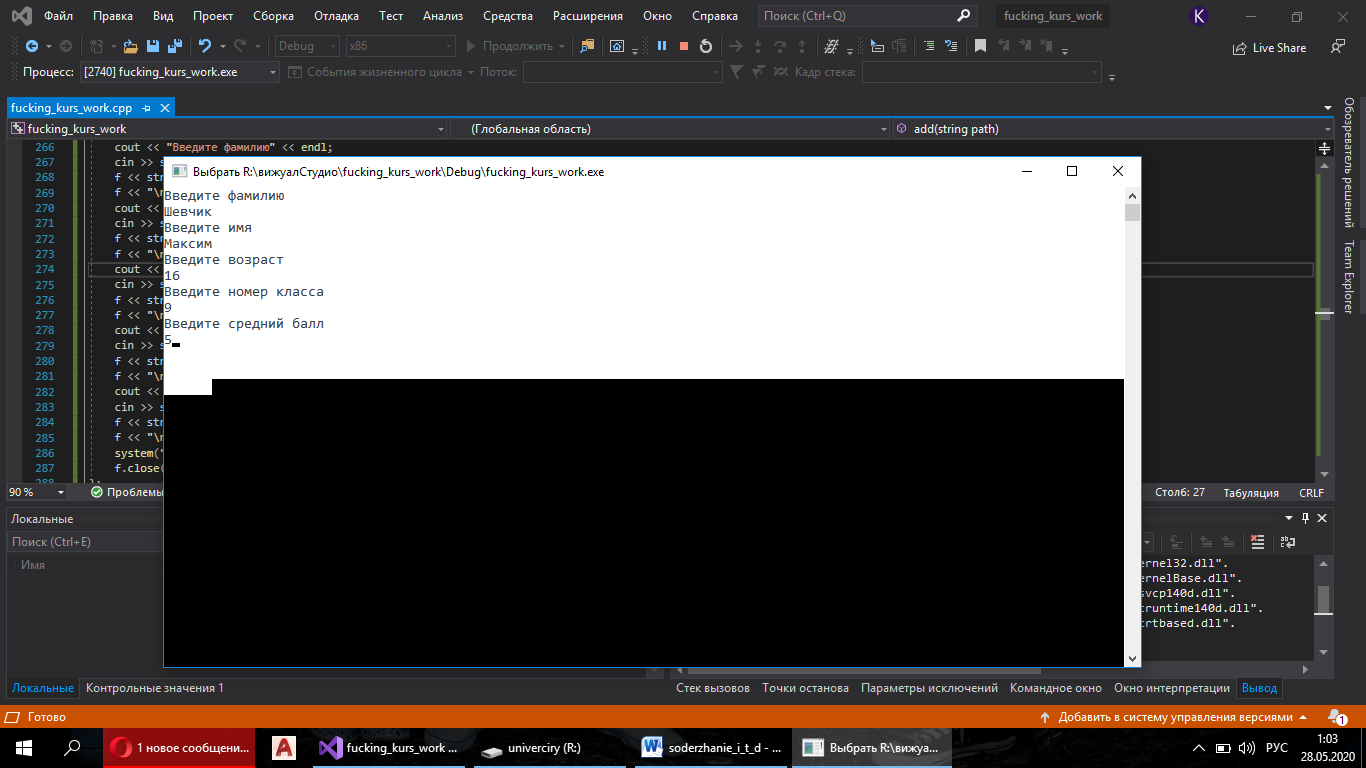


Рисунок 4.5

После перезаполнения массива(пункт меню — 0) можно увидеть следующий результат (см. рисунок 4.6)

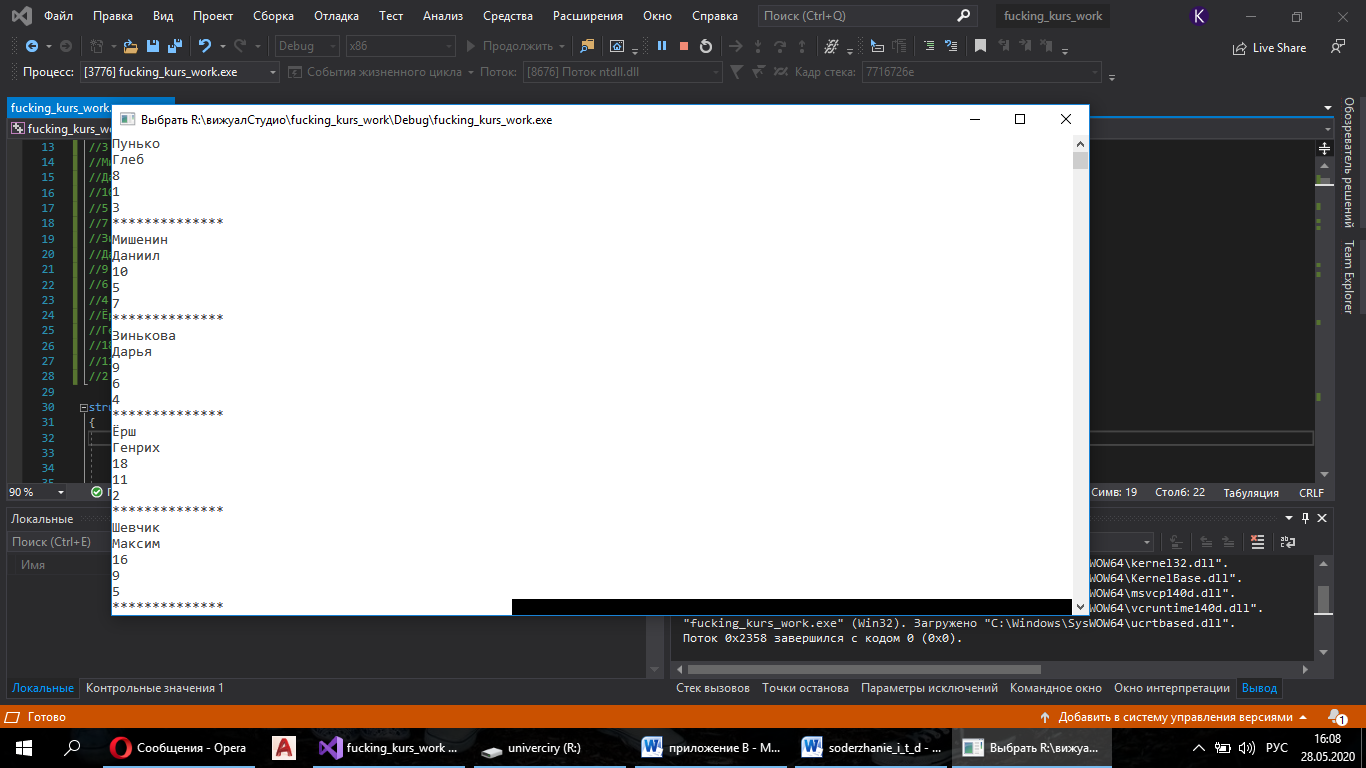


Рисунок 4.6

В пункте 3 осуществляется перезапись файла(см. рисунок 4.7)

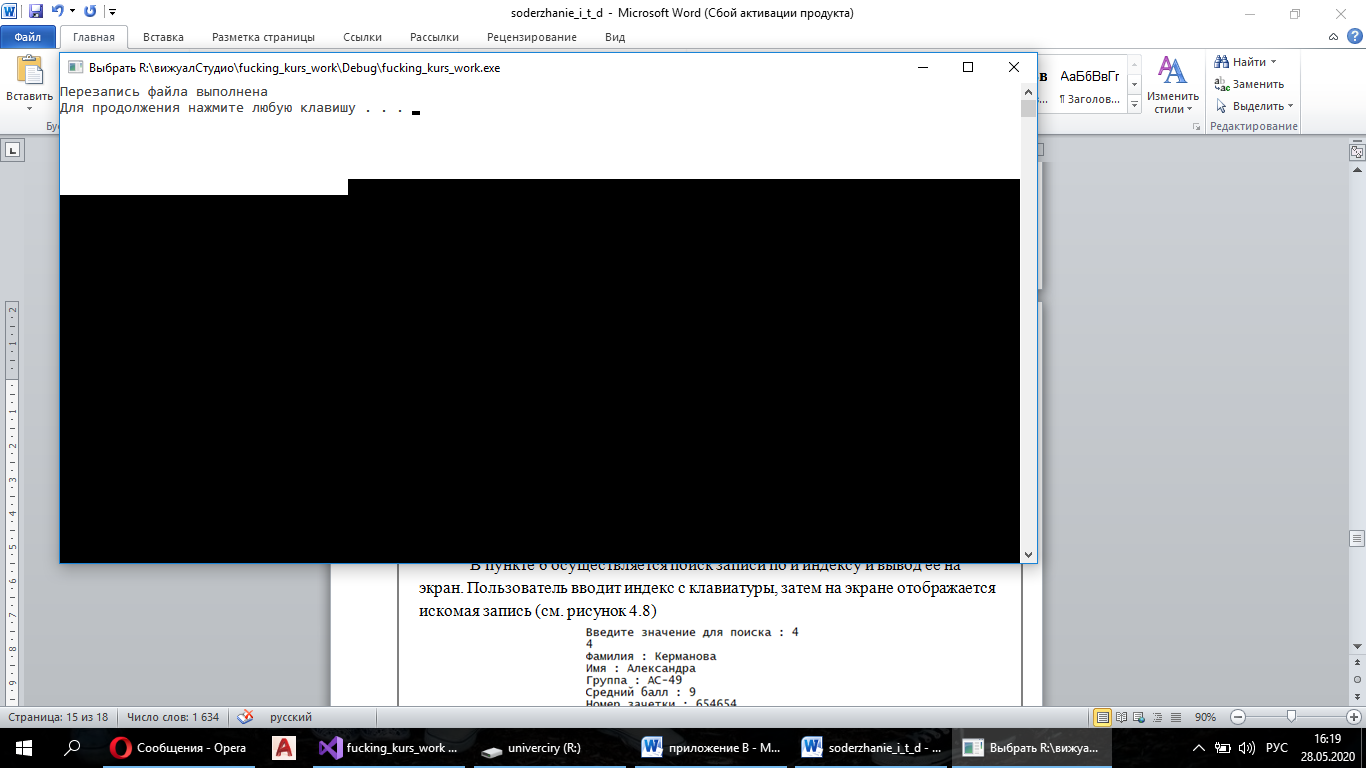


Рисунок 4.7

В пункте 4 осуществляется сортировка массива по выбранному полю. Поле, по которому будет осуществляться сортировка, вводится с клавиатуры (см. рисунок 4.7)

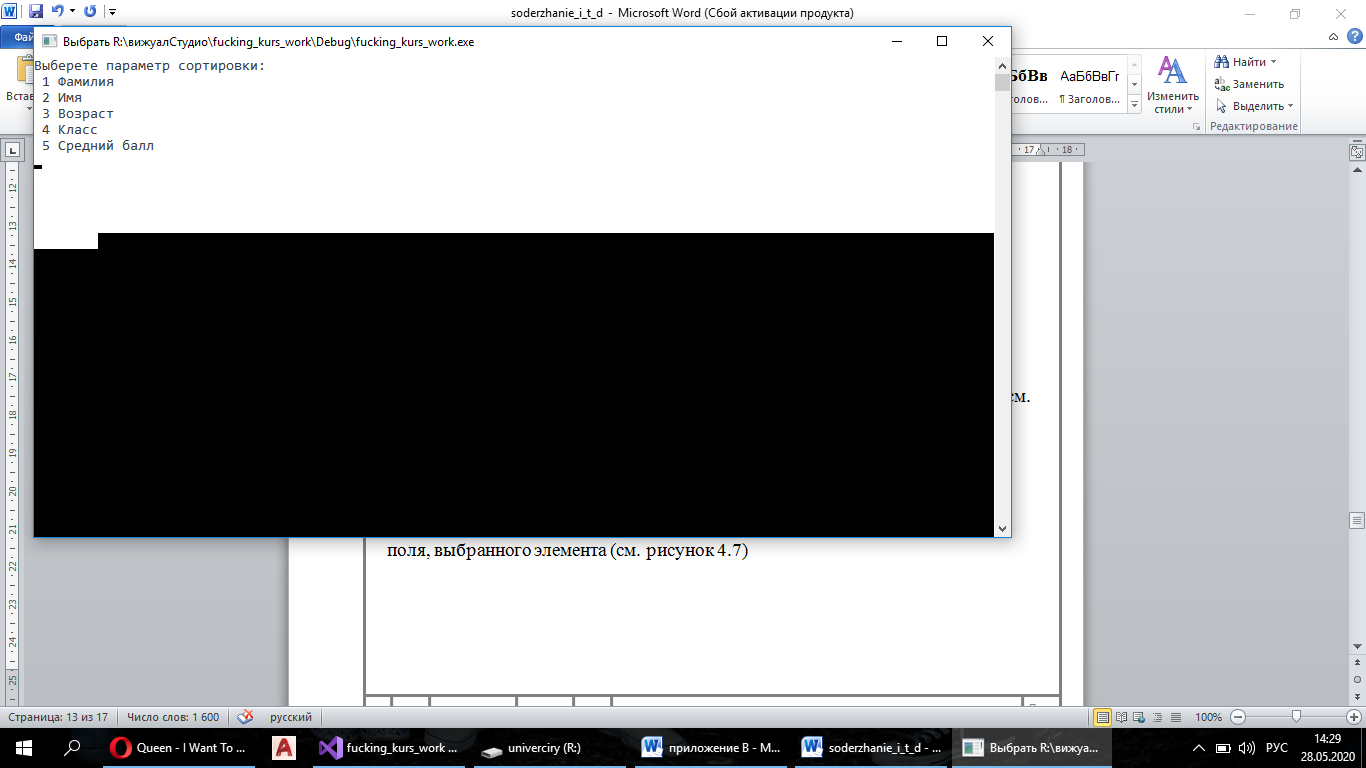
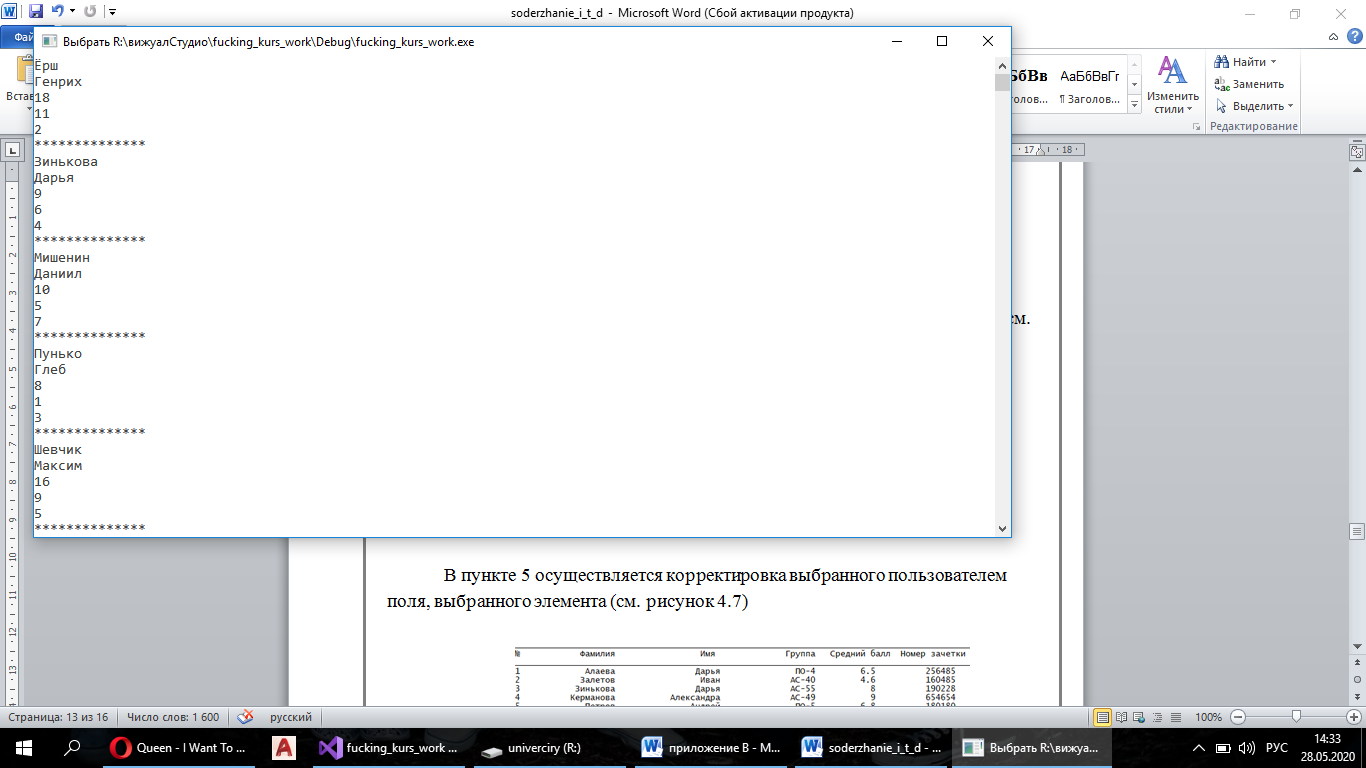


Рисунок 4.7

Протестируем сортировку по полю «Фамилия», результат записывается в индексный файл «KursIndecses.txt», выводим отсортированный массив

(см. рисунок 4.8)



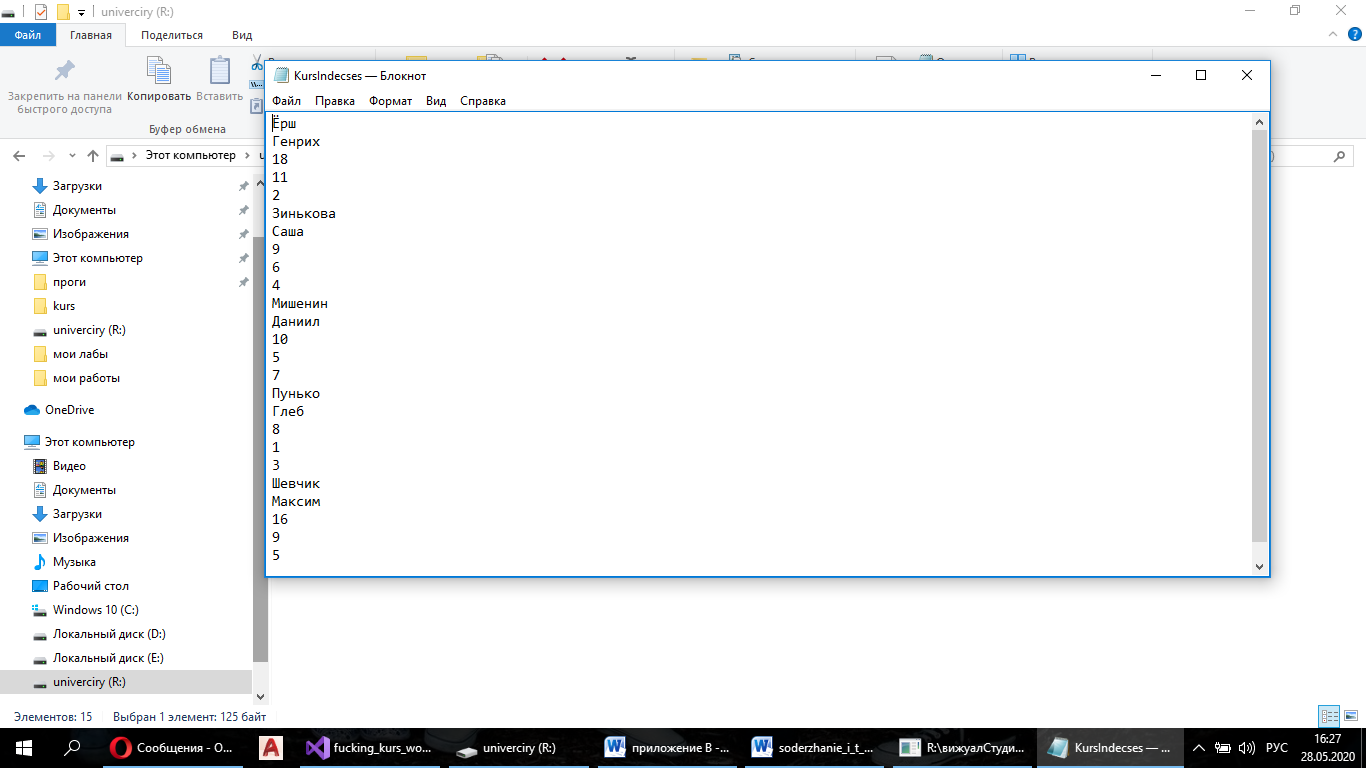


Рисунок 4.8

В пункте 5 осуществляется корректировка выбранного пользователем поля, выбранного элемента (см. рисунок 4.9)

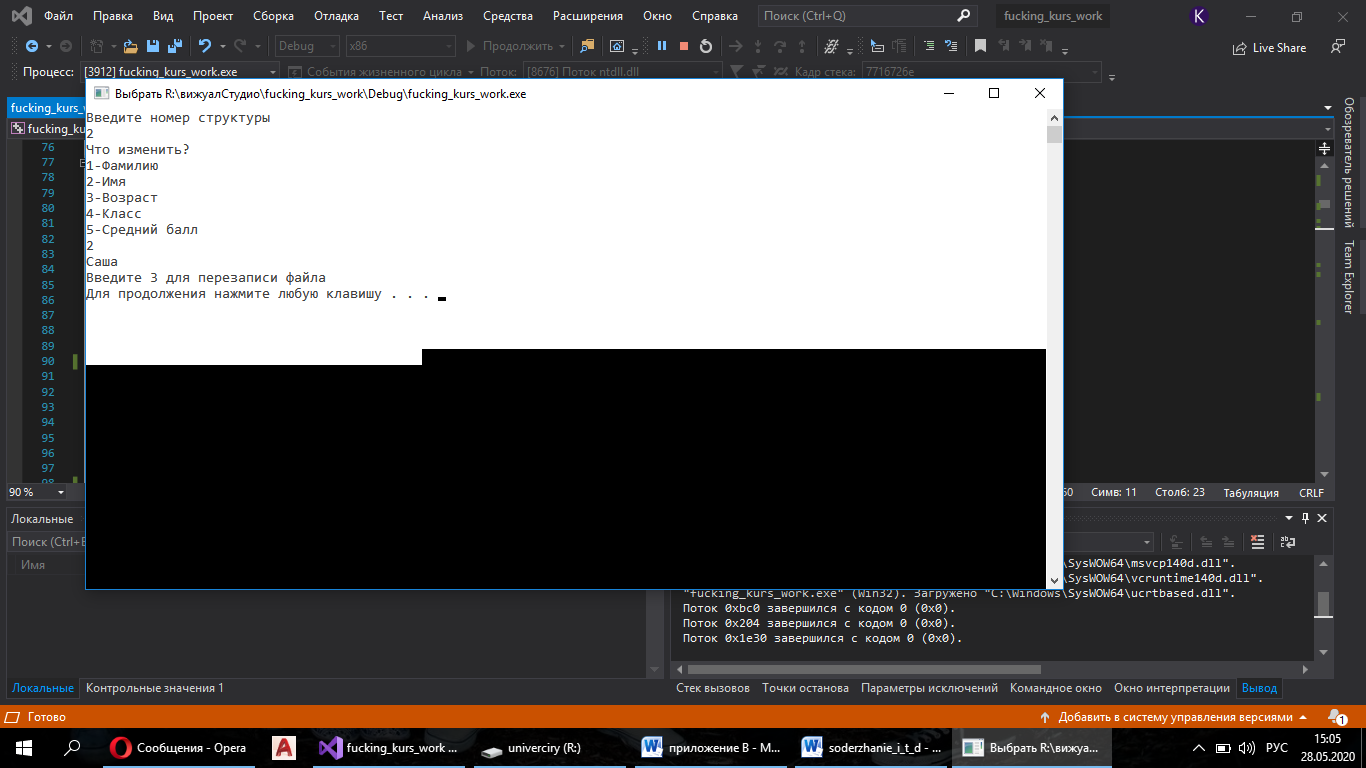


Рисунок 4.9

Выведем результат работы вышеописанных действий(см. рисунок 4.10)

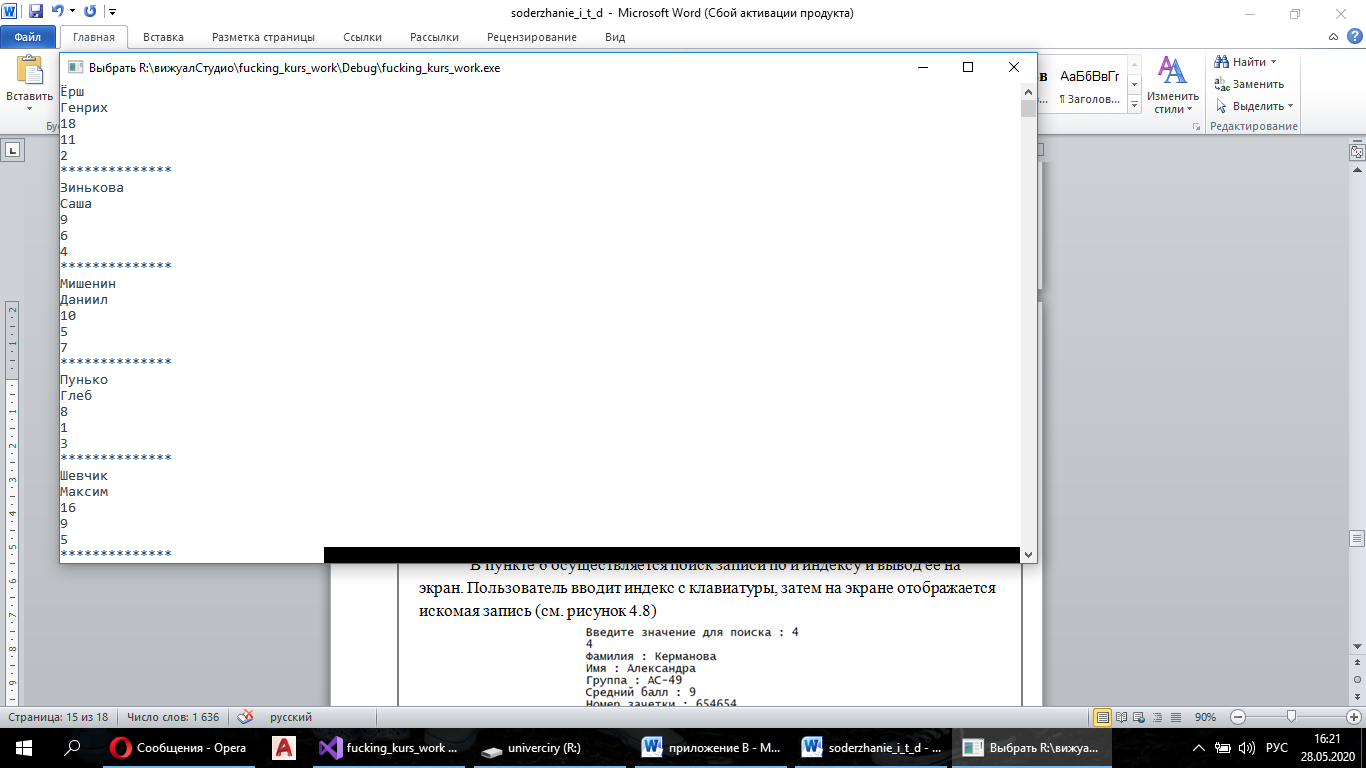
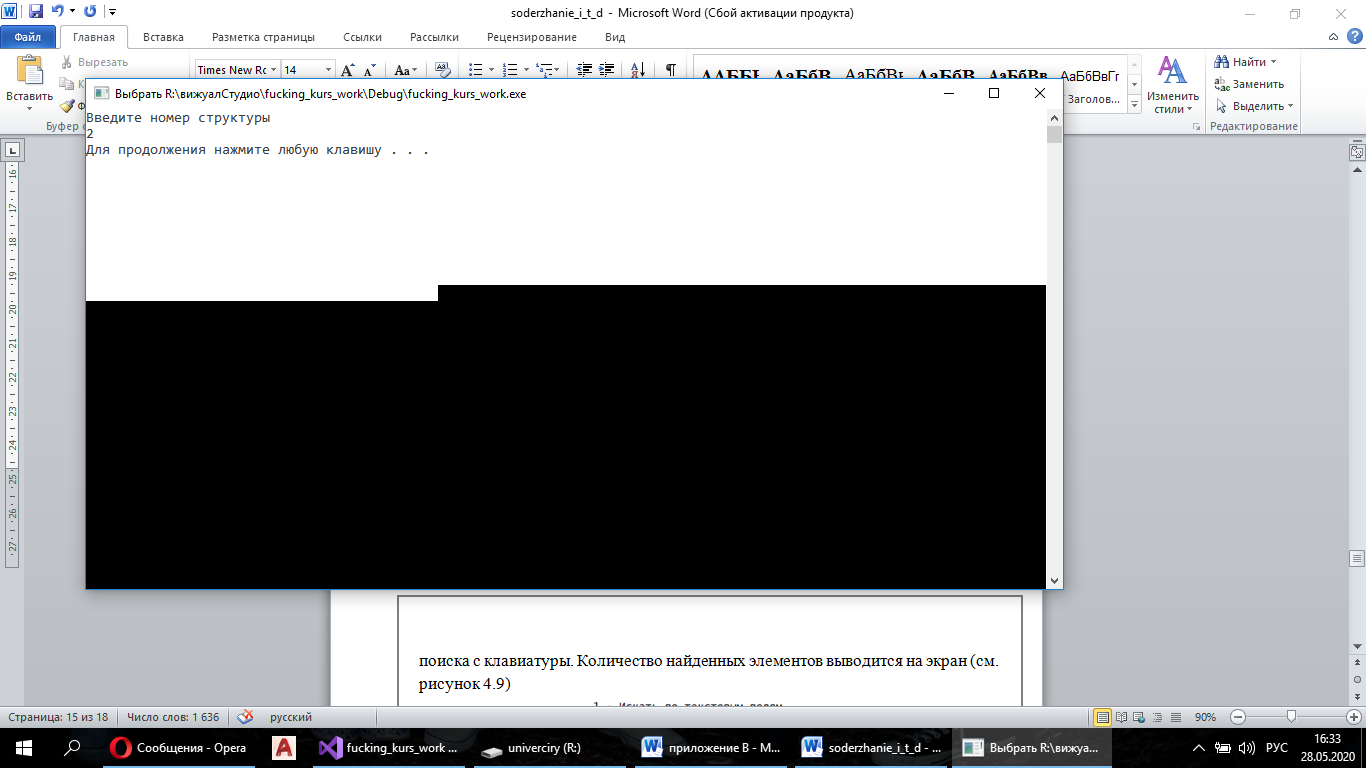


Рисунок 4.10

В пункте 6 осуществляется удаление структуры из массива. Пользователь вводит индекс с клавиатуры, затем на экране отображается искомая запись (см. рисунок 4.11)



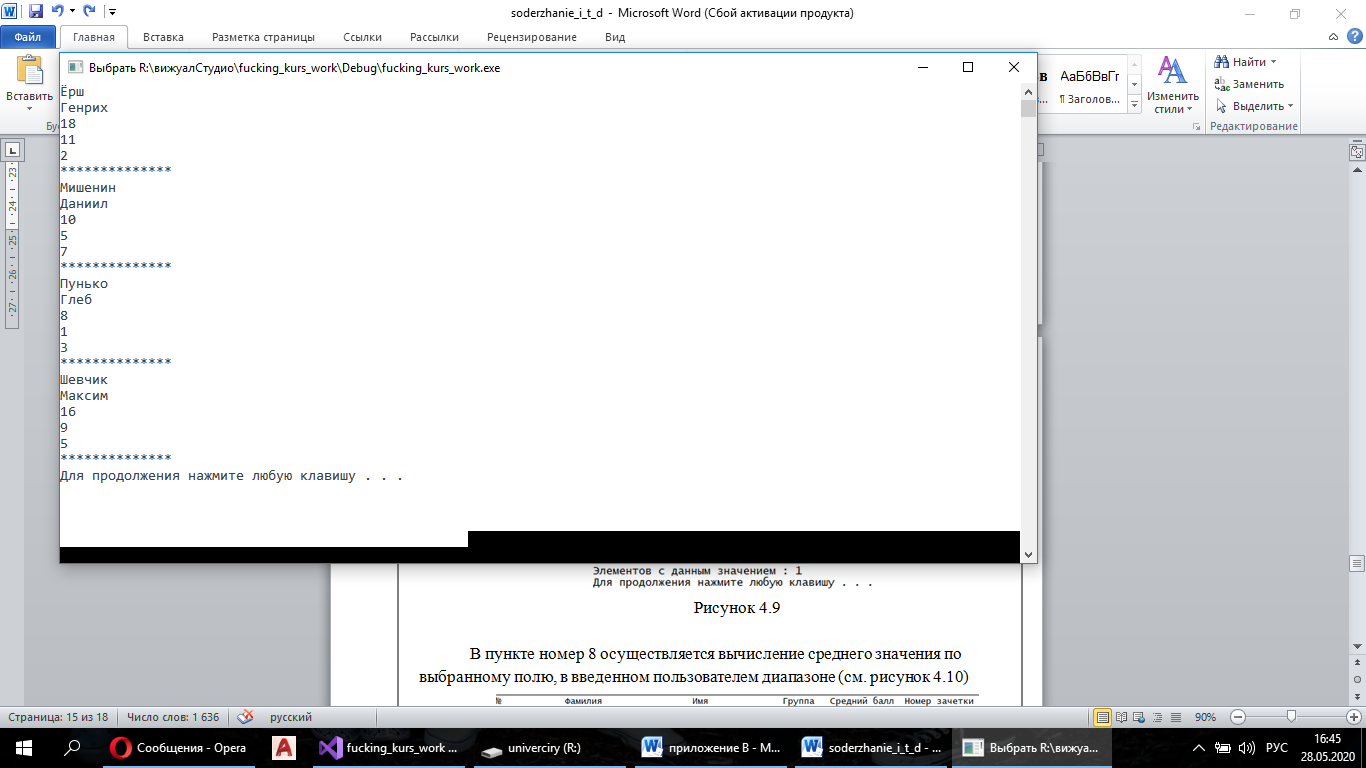


Рисунок 4.11

В пункте номер 7 осуществляется поиск элемента с заданным значением. Пользователь выбирает поле для поиска и вводит значение для поиска с клавиатуры. Найденный элемент выводится на экран (см. рисунок 4.12)

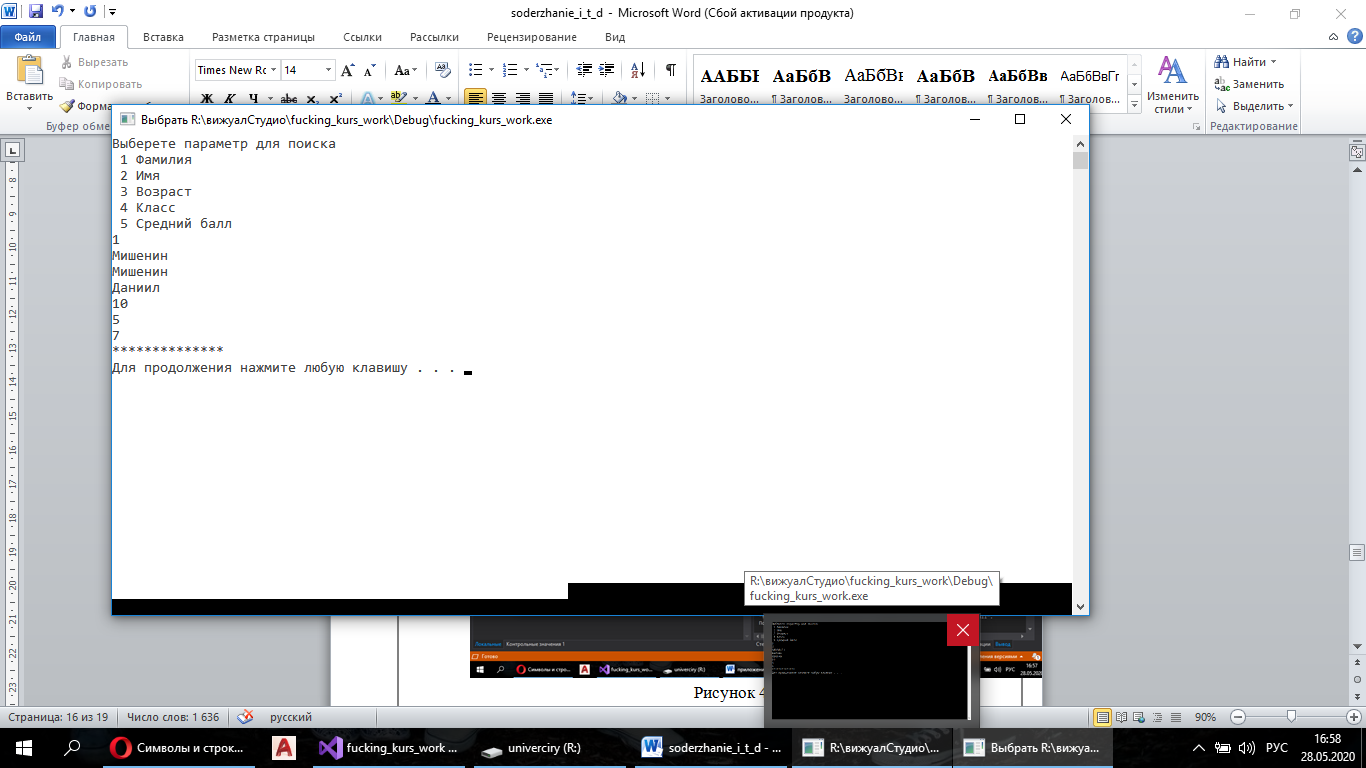


Рисунок 4.12

В пункте номер 8 выводится второе меню для решения дополнительных заданий (см. рисунок 4.13)

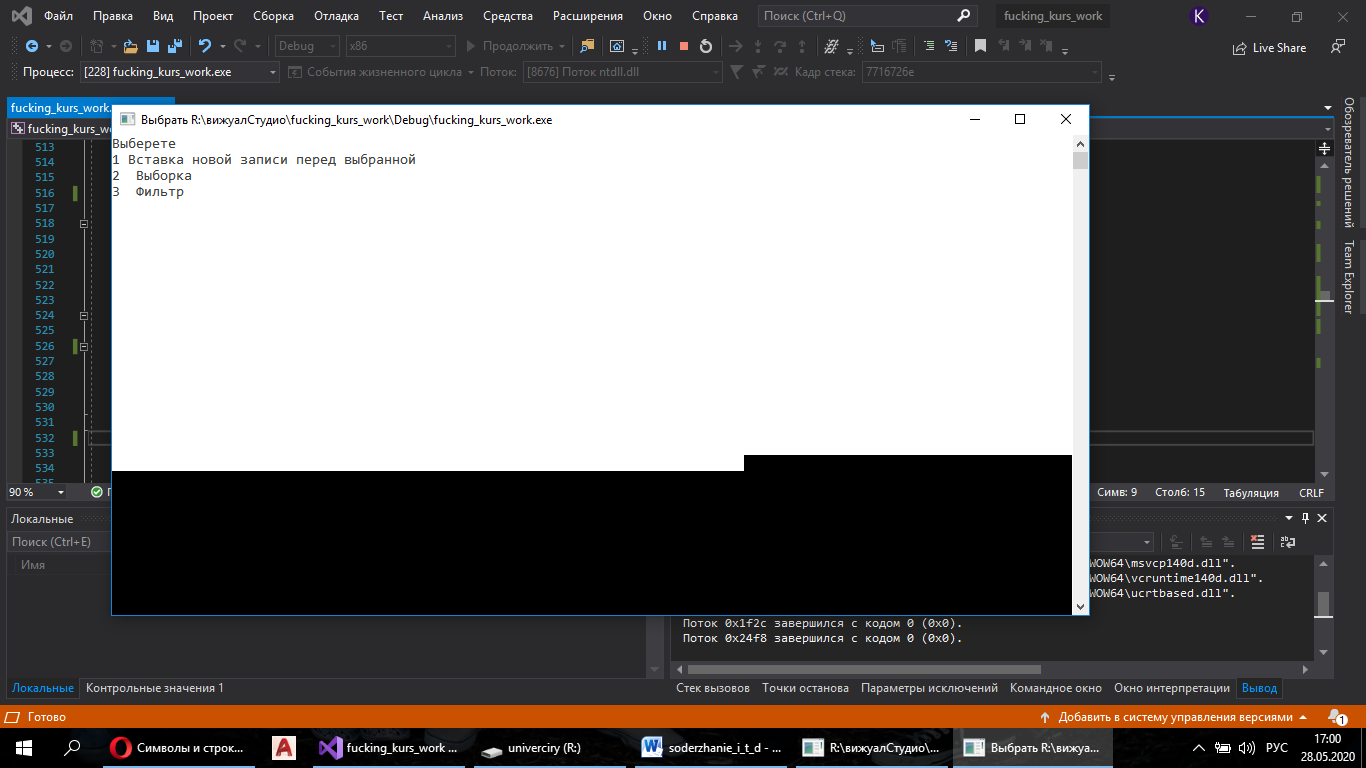
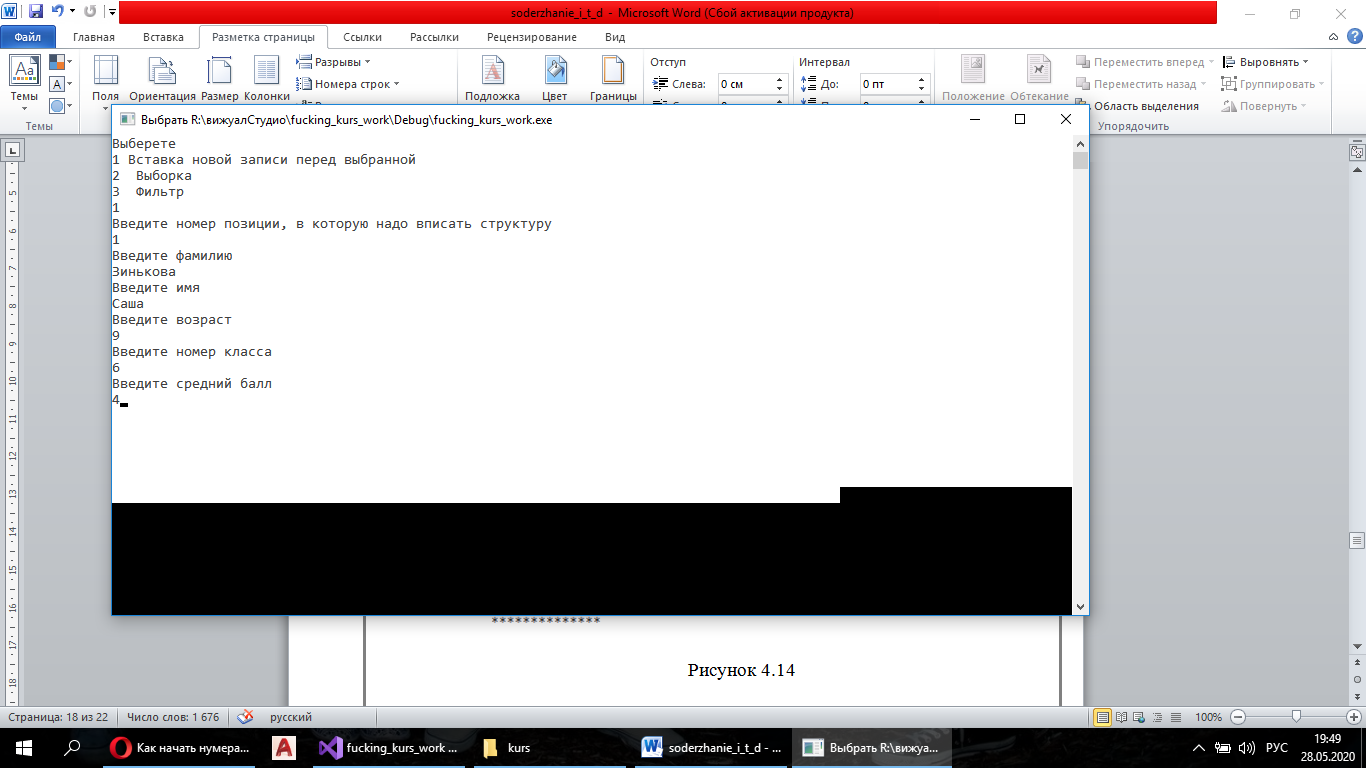


Рисунок 4.13

В пункте номер 8.1 осуществляется вставка новой структуры перед выбранной (см. рисунок 4.14)



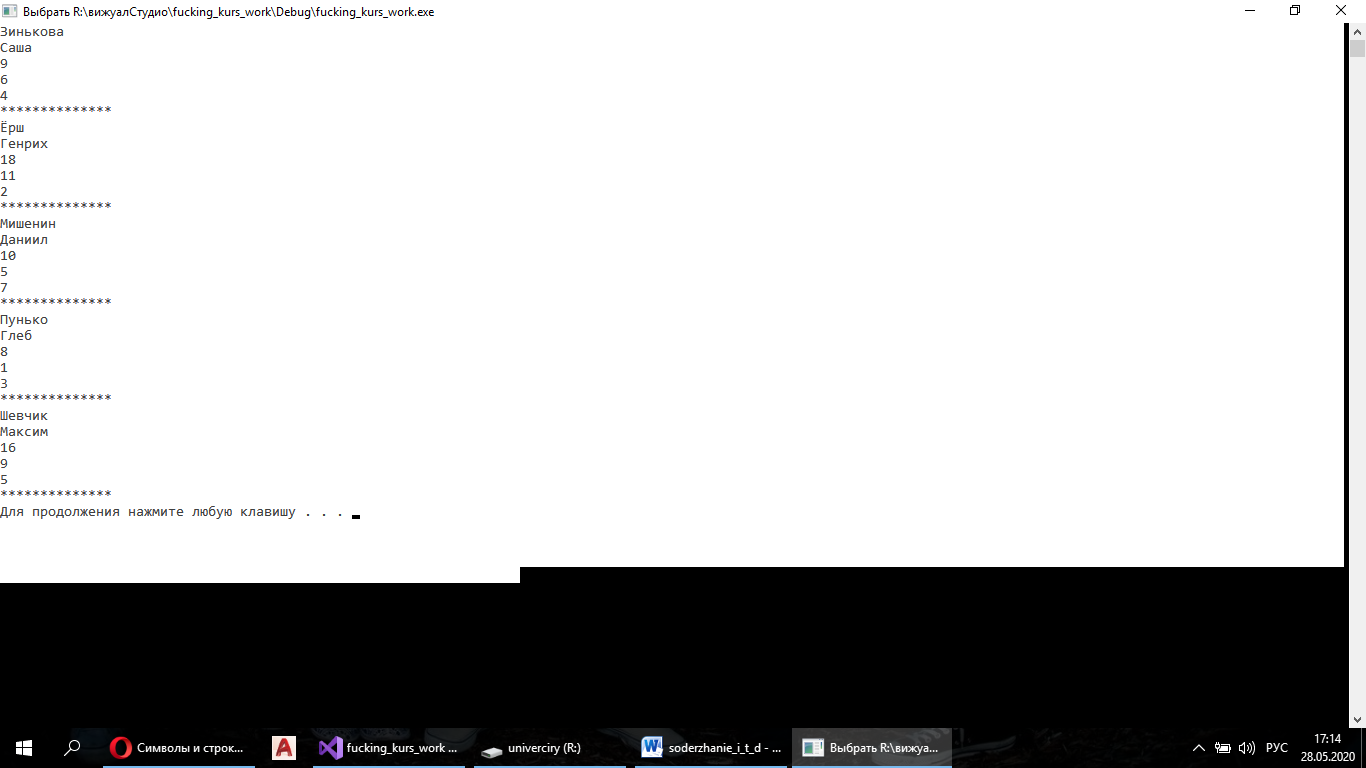


Рисунок 4.14

В пункте номер 8.2 осуществляется выборка. Пользователь вводит левую и правую границы диапазонов и выбирает поле, по которому будет искаться среднее значение (см. рисунок 4.15)

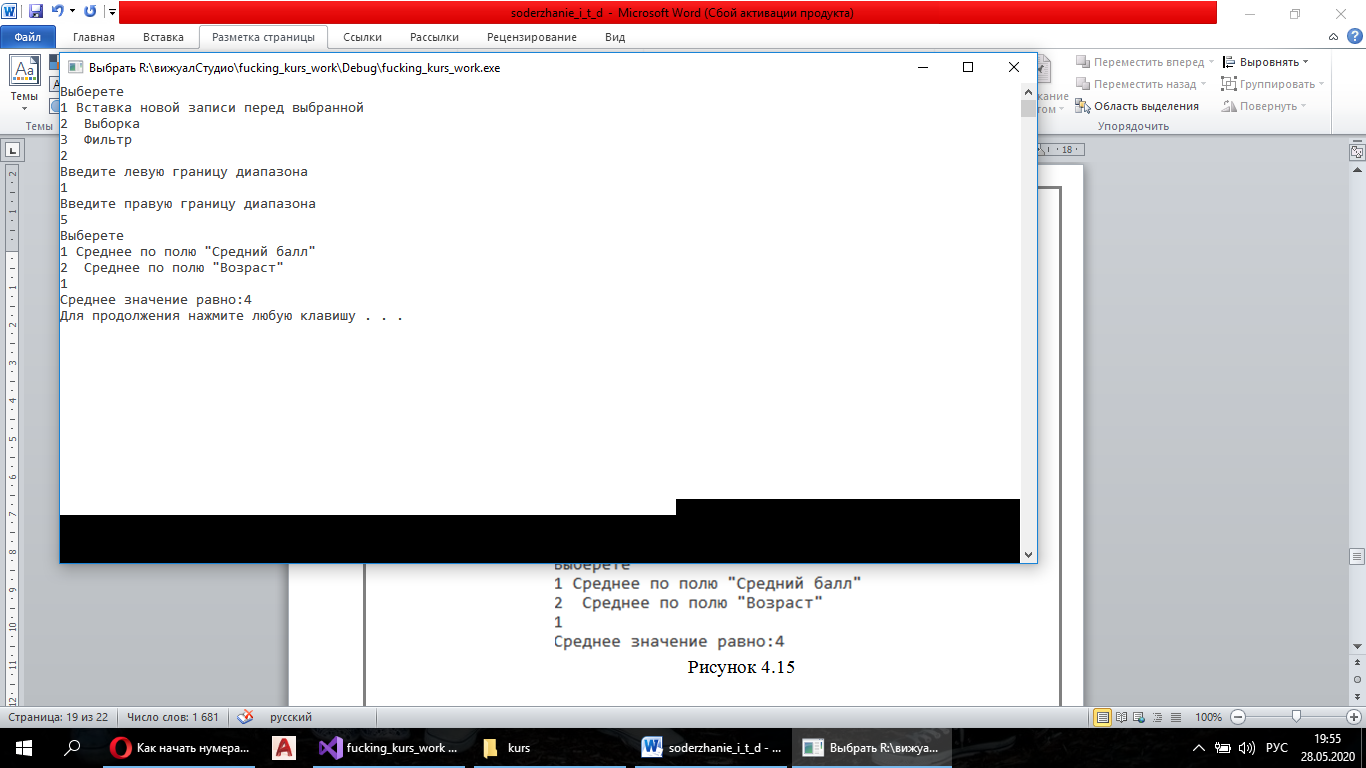


Рисунок 4.15

В пункте номер 8.3 осуществляется фильтрация структур по выбранному полю больших или меньших относительно введенного пользователем числа (см. рисунок 4.16)

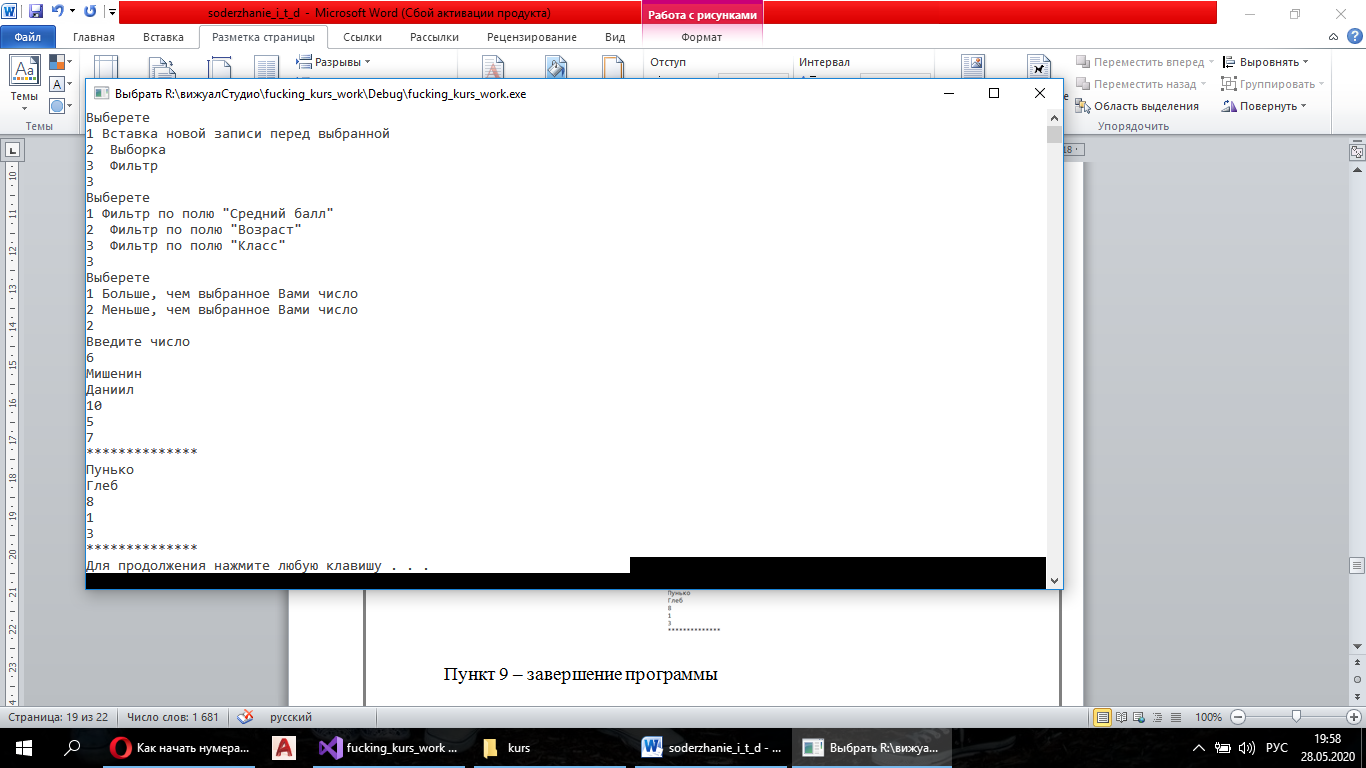


Рисунок 4.16

Пункт 9 – завершение программы(см. рисунок 4.17)

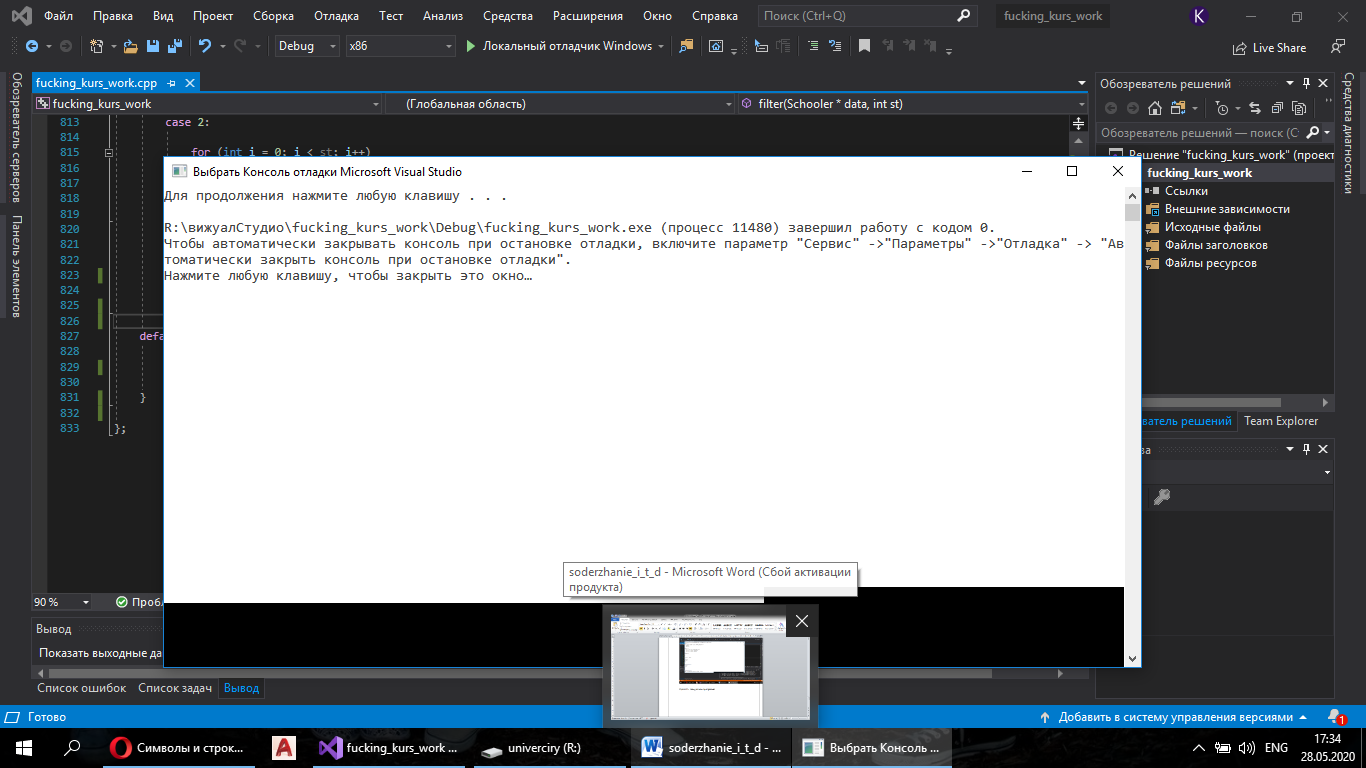


Рисунок 4.17

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была получена работающая программа, в которой были реализованы все функции в соответствии с предоставленным заданием. Программа работает исправно, вероятность возникновения ошибок и исключений сведена к минимуму. Все условия и ограничения были соблюдены. Программа предоставляет удобную среду для работы со списками школьников. Вести учет школьников становится гораздо проще благодаря встроенному функционалу, возможности добавления, изменения, удаления, поиску, сортировке, просмотру статистики по совокупностям школьников.

В ходе выполнения работы были практически закреплены знания о работе с массивами структурированных данных, инструментами файлового ввода и вывода, основными языковыми средствами языка C++.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

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

3. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем.