****

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

(**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**)» (**МАИ)**

Кафедра «Технология испытаний и эксплуатации»

**Курсовая работа**

Дисциплина «Технологии программирования»

ТЕМА

«Создание и обслуживание учебной базы данных на удаленном компьютере»

«База Данных абитуриентов ВУЗа»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

20902 – 01 81 01

Листов 43

Отметка о защите Задание выполнил  
\_\_\_\_\_\_\_ / Молчанова С. И. студент группы Т12О – 209Б – 19

«\_\_» 2021 г.

\_\_\_\_\_\_\_\_ / Атрашков Н. А.

«25» мая 2021 г.

2021

**Аннотация**

Программа **«База данных абитуриентов ВУЗа»** является курсовой работой по дисциплине «Технологии программирования» и предназначается для создания, редактирования и сериализации БД абитуриентов ВУЗа.

Программа реализуется на платформе интегрированной среды разработки программного обеспечения Microsoft visual studio 2019 community на языке C#.

Пояснительная записка состоит из шести параграфов и двух приложений:

1. Постановка задачи

В данном параграфе представлено задание своего варианта, а также условия и требования для его выполнения.

1. Описание алгоритма

В данном параграфе представлено описание основного алгоритма программного комплекса **«База данных абитуриентов ВУЗа»**, в котором разобраны необходимые действия для правильной работы программы.

1. Описание классов

В данном параграфе разобраны библиотеки классов, используемые в программе, приведены описания методов, а также приложена диаграмма классов.

1. Организация данных и диалога

В данном разделе разобрана организация входных и выходных данных программы.

1. Технические и программные средства

В данном параграфе разобраны технические и программные средства, используемые для разработки программы.

1. Результаты работы программы и их оценка

В данном параграфе приводится испытание и тестирование законченной программы.

1. Приложение А. Текст программы

В данном параграфе представлен весь текст программы

1. Приложение Б. Текст библиотеки

В данном параграфе представлен текст собственной библиотеки класса, предназначенный для работы с табличными данными

1. Приложение В. Техническое задание на программу  
   В данном параграфе представлено техническое задание на программу
2. Приложение Г. База данных

В данном параграфе представлена ссылка на файл, содержащий таблицу с информацией о абитуриентах.

**Содержание**

[**1.** **Постановка задачи** 5](#_Toc72787617)

[**2.** **Описание алгоритма** 6](#_Toc72787618)

[**3.** **Описание классов** 7](#_Toc72787619)

[3.1. Описание методов 7](#_Toc72787620)

[3.2. Диаграмма классов. 14](#_Toc72787621)

[**4.** **Организация данных и диалога** 15](#_Toc72787622)

[4.1. Организация входных данных 15](#_Toc72787623)

[4.2. Организация выходных данных 15](#_Toc72787625)

[**5.** **Технические и программные средства** 16](#_Toc72787626)

[5.1. Технические средства 16](#_Toc72787627)

[5.2. Программные средства 16](#_Toc72787628)

[**6.** **Результаты работы программы и их оценка** 17](#_Toc72787629)

[6.1. Объект испытаний 17](#_Toc72787630)

[6.2. Цель испытаний 17](#_Toc72787631)

[6.3. Требования к программе 18](#_Toc72787632)

[6.4. Требования к программной документации 18](#_Toc72787633)

[6.5. Методы испытаний 19](#_Toc72787634)

[6.6. Оценка результатов тестирования 19](#_Toc72787635)

[**Заключение** 20](#_Toc72787636)

[**Приложение А. Текст Программы** 21](#_Toc72787637)

[**Приложение Б. Текс библиотеки** 21](#_Toc72787638)

[**Приложение В. Техническое задание** 21](#_Toc72787639)

[**Приложение Г. База данных** 21](#_Toc72787640)

# **Постановка задачи**

Создание и редактирование базы данных, содержащей сведения об абитуриентах.

Требования к программному продукту: при реализации программного продукта необходимо использовать теоретические сведения по дисциплине «Технологии программирования», полученные в процессе изучения дисциплины. При разработке предусматривается использование собственных и сторонних библиотек, атрибутов.

# **Описание алгоритма**

Для решения поставленной задачи необходимо создать форматированную таблицу в виде двумерного массива, с использованием библиотеки класса FILEWORKING, а также выполнить следующие действия:

1. определить класс для работы с таблицей, содержащей в себе информацию о абитуриентах;
2. организовать считывание информации из файла;
3. организовать следующие функции:
   * открытие форматированной таблицы;
   * создание форматированной таблицы;
   * редактирование данных;
   * сериализацию данных;
   * передачу данных по сети;
   * форматированный вывод данных на экран;
   * формирование отчета о работе программы.

Способ передачи параметров в методы - на усмотрение разработчика программы.

\*\*Для каждого метода разработать систему кодов возврата, отражающую результат выполнения действий внутри метода, например, успешное выполнение, неправильный номер элемента, неправильный параметр, и т. п.

1. **Описание классов**

Класс work\_with\_file находится в библиотеке FILEWORKING, класс предназначен для работы с табличными данными.

Функционал класса:

1. Открытие файла, его создание;
2. Редактирование информации.
   1. Описание полей

* public string path — путь к файлу;
* объект класса FileStream, предназначенный для чтения файла – FileStream Stream;
* объект TextReader, который считывает символы из потока байтов – StreamReader Reader;
* вспомогательная строка для чтения файла и записи информации в массив – string line;
* двумерный массив для хранения информации – public string[,] info;
* количество столбцов в таблице – public int count\_of\_t;
* переменная-маркер, указывающая на пустоту или полноту файла – \_null;
  1. Описание методов

Ниже представлены описания методов библиотеки класса FILEWORKING .

1. Метод **reading()** предназначен для чтения файла и формирования двумерного массива строк с использованием информации, хранящейся в файле.

* Прототип метода **public string[,] reading();**
* Параметры метода отсутствуют;
* Возвращаемое значение – ссылка (info) на сформированный массив;
* Описание алгоритма метода:
  + чтение первой строки файла;
  + если файл не пустой, то определяется количество столбцов в таблице, объявляется двумерный массив и организуется цикл для чтения файла начиная со второй строки и до его конца, в теле цикла выполняются следующие действия:

1. объявление строкового массива temp, который будет хранить информацию об одном абитуриенте;
2. объявление временного двумерного строкового массива info\_ и копирование в него исходного массива;
3. в переменную temp записывается информация из прочитанной строчки, которая была разделена посредством разделителя «;»;
4. копирование temp в конец массива info\_;
5. в исходный массив info копируется информация из временного массива info\_;
6. чтение следующей строчки;
   * иначе выводим: «Файл пуст, обработка невозможна! Выйдите из программы и повторите попытку или заполните таблицу, нажав соответствующий пункт меню!»;

На [рисунке 1](#рис1) представлена схема алгоритма чтения файла.

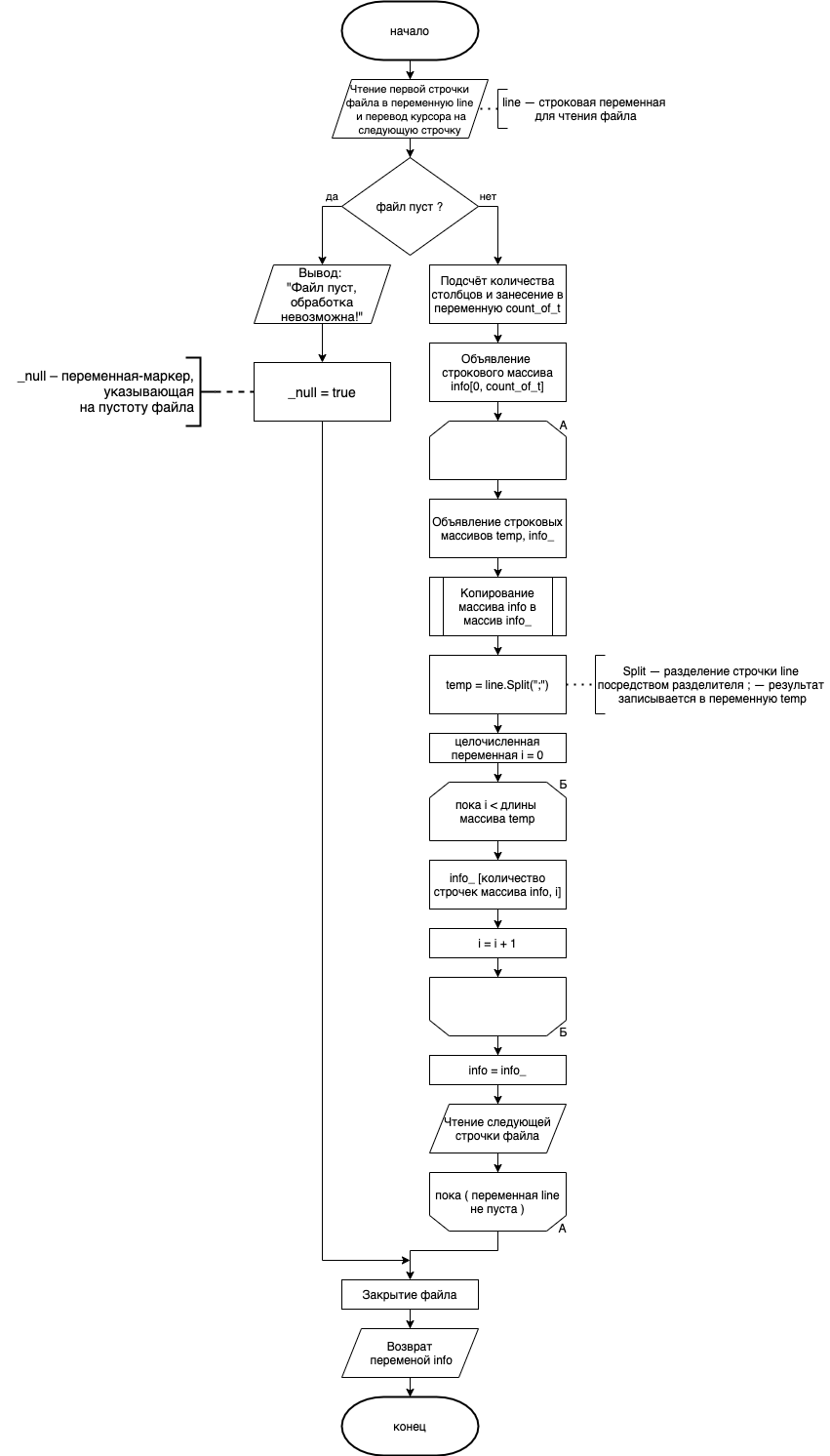


Рисунок 1 – Схема алгоритма чтения файла

1. Метод **edit\_list()** предназначен для изменения информации в таблице данных.
   * Прототип функции **public void edit\_list()**;
   * Параметры функции:  
     string surname – фамилия абитуриента, информацию о котором нужно изменить;
   * Возвращаемое значение отсутствует;
   * Описание алгоритма метода:
     1. поиск абитуриента в таблице;
     2. если абитуриент найден, то выполняются пункты 3-4, иначе вывод сообщения об отсутствии абитуриента в таблице и завершение метода и возвращение в основную программу;
     3. выбор информации, которую требуется изменить;
     4. изменение информации.

На [рисунке 2](#рис2) представлена схема алгоритма изменения информации в таблице данных.

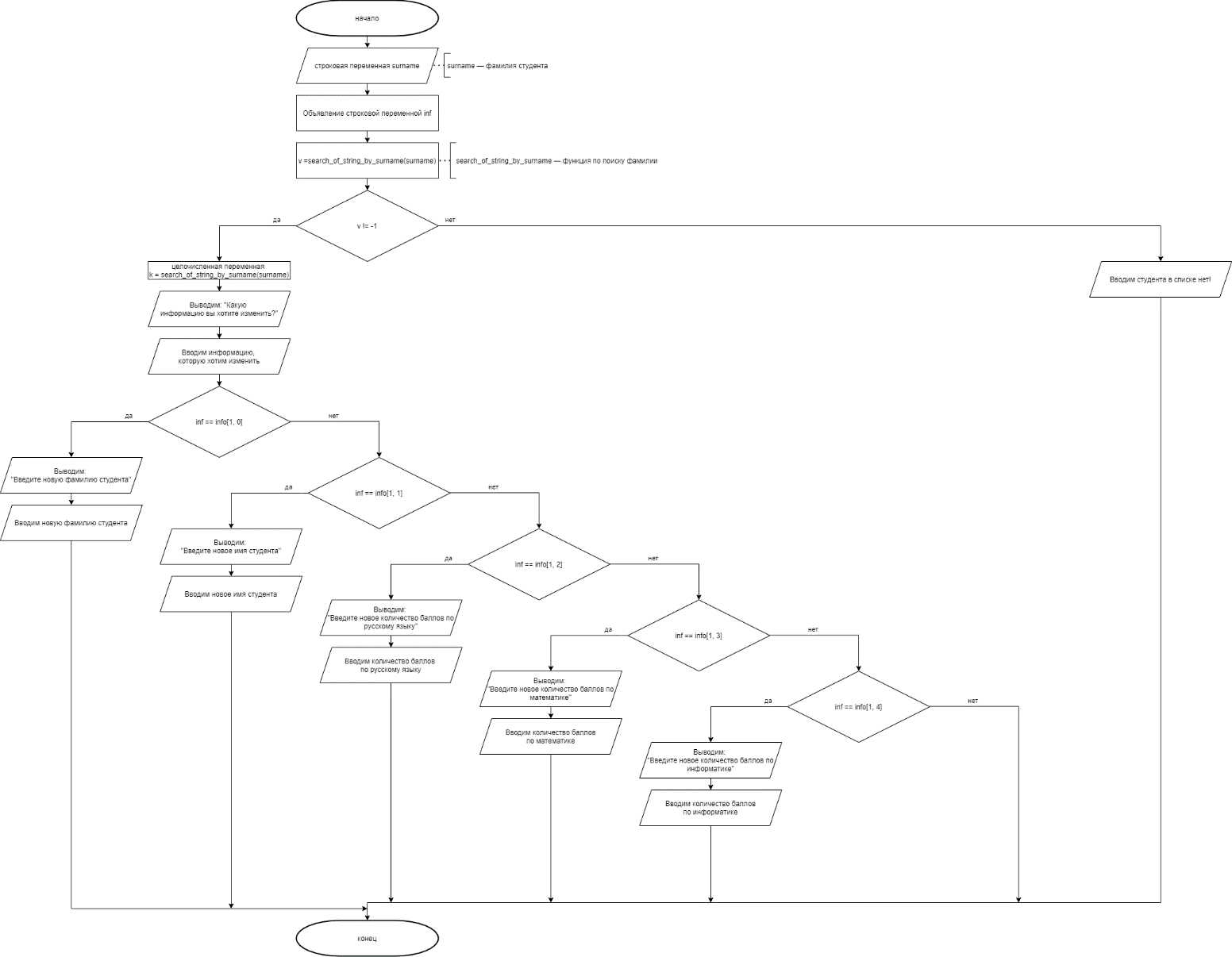


Рисунок 2 - Схема алгоритма изменения информации в таблице данных

1. Метод **del()** предназначена для удаления информации о абитуриенте.

* Прототип метода **public void del()**;
* Параметры метода:  
  string del\_ – фамилия абитуриента, информацию о котором нужно удалить;
* Возвращаемое значение отсутствует;
* Описание алгоритма метода:
  1. создание временного массива строк, с размерностью меньше на одну строку;
  2. копирование исходного массива во временный массив без строки с информацией о абитуриенте, сведения о котором нужно удалить;
  3. если абитуриента в таблице нет, то вывести на экран предупреждение "Такого абитуриента в списке нет!"

На [рисунке 3](#рис3) представлена схема алгоритма удаления страны из таблицы.

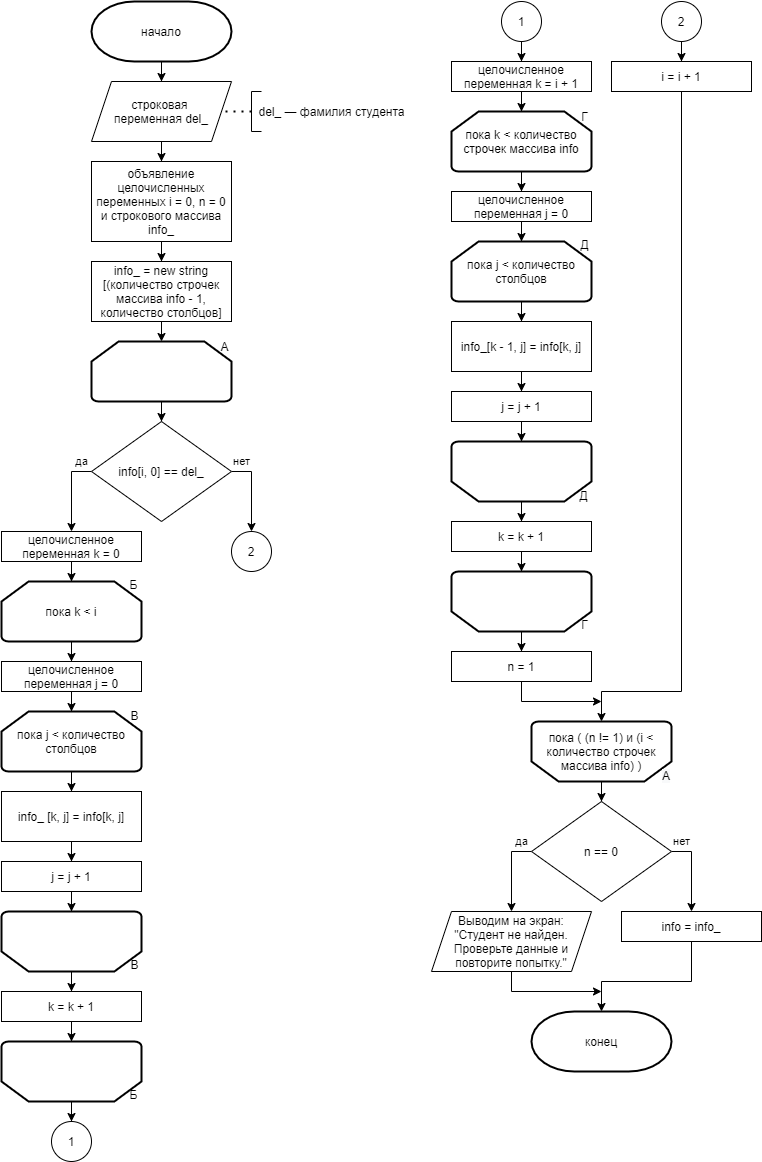


Рисунок 3 - Схема алгоритма удаления страны из таблицы

1. Функция **client()** предназначена для сериализации и передачи данных по сети.

* Прототип метода **public static void client()**;
* Параметры метода:  
  path – путь к файлу, который необходимо сериализовать и передать;
* Возвращаемое значение отсутствует;
* Описание метода.  
  Для реализации данной функции необходим атрибут. Атрибут – [Serializable] применяется к классу work\_w\_file. Данный атрибут позволяет сериализовать данные, используя стандартный класс .NET.   
  Передача сериализованного файла по локальной (глобальной) сети была организована с помощью протокола TCP. Также данную функцию можно реализовать посредством использования протокола UPD, однако для данной курсовой работы, TCP оказался наиболее подходящих по следующим причинам:
  + TCP устанавливает соединение перед передачей данных и обеспечивает правильную доставку пакетов, в то время как UDP не подтверждает, приняты ли переданные данные или нет;
  + TCP очень надежен для передачи полезных данных, так как он принимает подтверждение отправленной информации.
  1. ****Диаграмма классов  
     На [рисунке 4](#рис5) представлена полная диаграмма классов.

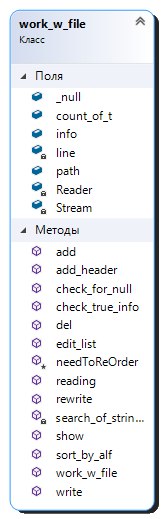
****

Рисунок 4 – Диаграмма классов

1. **Организация данных и диалога**
   1. Организация входных данных  
      Входными данными является текстовый файл, содержащий отформатированную таблицу информации о абитуриентах ВУЗа, каждая строка файла содержит поля представленные в [таблице 1](#табл). Поля отделяются точкой запятой друг от друга.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Фамилия | Имя | Отчество | Математика | Информатика | Русский язык |
| Атрашков | Никита | Андреевич | 100 | 120 | 80 |
| Петров | Иван | Алексеевич | 30 | 45 | 65 |
| Осипова | Наталья | Олеговна | 67 | 70 | 90 |

Таблица 1 – Пример входных данных

Файл, содержащий входные данные, может быть создан.

* 1. Организация выходных данных

Выходными данными программы является бинарный файл, содержащий форматированную обработанную информацию.

1. **Технические и программные средства**
   1. Технические средства

При тестировании программы использовался компьютер,

который обладает следующими техническими параметрами:

* процессор с частотой 1,6 ГГц;
* оперативная память объемом 8 Гб;
* 480 ГБ свободного пространства на жестком диске.
  1. Программные средства

При проведении тестов над программой «Государство» использовался компьютер под управлением ОС MS Windows 7.

1. **Результаты работы программы и их оценка**
   1. Объект испытаний

Объектом испытаний является программа **«База Данных абитуриентов ВУЗа»** для использования сотрудниками деканата. Программа предназначена для работы с таблицей: сортировка элементов, добавление и удаление элементов, редактирование информации.

Для проверки работоспособности программы была разработана система тестов, проверяющая основной функционал программной реализации.

* 1. Цель испытаний

Цель испытаний программы **«База Данных абитуриентов ВУЗа»** состоит в том, чтобы оценить работу программы, а именно проверить, соответствует ли характеристики реализованной программы функциональным требованиям, представленным в техническом задании, и перечисленных в разделе «Требования к программе» данного документа. Проверяется корректная работоспособность программной реализации при ей эксплуатации обычным пользователем.

* 1. Требования к программе

Во время проведения испытаний необходимо проверить следующий функционал программы **«База Данных абитуриентов ВУЗа»**:

* + открытие форматированной таблицы;
  + создание форматированной таблицы;
  + редактирование данных;
  + сериализацию данных;
  + передачу данных по сети;
  + форматированный вывод данных на экран;
  + формирование отчета о работе программы.
  1. Требования к программной документации

Состав программной документации:

* «База Данных абитуриентов ВУЗа». Техническое задание.
* «База Данных абитуриентов ВУЗа». Пояснительная записка.
  1. Методы испытаний

При тестировании программы использовался метод «чёрного ящика». Суть этого метода заключается в том, что тестировщик не располагает данными о коде программы и тестирует её, только используя её документальное описание и набор входных данных. Такой метод позволяет проводить проверку специалисту, не относящемуся к группе разработчиков, и не требует специальных знаний при проведении тестирования.

* 1. Оценка результатов тестирования

Система тестов программы приведена в [таблице 2](#табл2).

Таблица 2 – Система тестов

|  |  |  |
| --- | --- | --- |
| Назначение | Входные данные | Фактическая реакция |
| Проверка реакции программы на ввод пункта меню, который соответствует требованиям | 1 | Фактическая реакция совпала с ожидаемой |
| 2 |
| 9 |
| Проверка реакции программы на ввод пункта меню, который не соответствует требованиям (является символом, не принадлежит диапазону) | s | Программа вывела предупреждение " Неправильное значение, оно должно быть в пределах диапазона 1...10"запросила пользователя ввести пункт заново |
| 100 |
| as |

Продолжение таблицы 2

|  |  |  |
| --- | --- | --- |
| Проверка реакции программы на пустой файл | пустой файл | Программа вывела предупреждение "Файл пуст, обработка невозможна! Выйдите из программы и повторите попытку или заполните таблицу, нажав соответствующий пункт меню!"  запросила пользователя закрыть программу или заполнить таблицу самостоятельно. |
| Проверка реакции программы на непустой файл | непустой файл | Фактическая реакция совпала с ожидаемой |
| Проверка реакции программы на поиск абитуриента, которого нет в БД | фамилия абитуриента, который отсутствует в БД | Программа вывела предупреждение "Такого абитуриента в списке нет!" |

# **Заключение**

В процессе разработки программы были полностью освоены принципы работы с текстовыми файлами, их чтением, обработкой и записью. Также были изучены способы форматирования, способы представления табличных данных, а также их передача как по локальной, так и по глобальной сети.

# **Приложение А. Текст Программы**

1. **using** System;
2. **using** FILEWORKING;
3. **using** System.Collections.Generic;
4. **using** System.Linq;
5. **using** System.Text;
6. **using** System.IO;
7. **using** System.Runtime.Serialization.Formatters.Binary;
8. **using** System.Net.Sockets;
9. **using** System.Net;
10. **using** System.Threading;
12. **namespace** nikita\_kr\_final
13. {
14. **class** Programm
15. {
16. **public** **static** **void** client(**string** path)
17. {
18. TcpClient eclient = **new** TcpClient("95.165.6.147", 55555);
19. NetworkStream writerStream = eclient.GetStream();
21. // класс для [де]сериализации данных
22. BinaryFormatter format = **new** BinaryFormatter();
23. **byte**[] buf = **new** **byte**[1024];
25. **int** count;
27. // открытие файла
28. FileStream fs = **new** FileStream(path, FileMode.Open);
29. BinaryReader br = **new** BinaryReader(fs);
31. // размер файла
32. **long** k = fs.Length;
34. // передача реального размера
35. format.Serialize(writerStream, k.ToString());
37. // сериализация и передача файла по 1024 байта
38. **while** ((count = br.Read(buf, 0, 1024)) > 0)
39. {
40. format.Serialize(writerStream, buf);
41. }
43. Console.WriteLine("Объект сериализован");
44. Console.WriteLine("Файл отправлен!");
46. fs.Close();
47. writerStream.Close();
48. }
50. // вывод инфомации об авторе
51. **public** **static** **void** info()
52. {
53. Console.WriteLine("—————————————————————————————————————");
54. Console.WriteLine("|          МАИ Институт №12         |");
55. Console.WriteLine("|         Группа Т12О-209Б-19       |");
56. Console.WriteLine("|           Атрашков Никита         |");
57. Console.WriteLine("|           Курсовая работа         |");
58. Console.WriteLine("—————————————————————————————————————");
59. Console.WriteLine();
60. }
62. // меню программы
63. **public** **static** **void** menu()
64. {
65. Console.WriteLine("Что вы хотите сделать?");
66. Console.WriteLine("1. Вывести данные на экран.");
67. Console.WriteLine("2. Произвести очистку данных от некорректной и недостоверной информации.");
68. Console.WriteLine("3. Добавление информации о новом студенте.");
69. Console.WriteLine("4. Отсортировать список по алфавиту.");
70. Console.WriteLine("5. Редактирование информации о существующем студенте.");
71. Console.WriteLine("6. Удаление информации о существующем студенте.");
72. Console.WriteLine("7. Отправить данные.");
73. Console.WriteLine("8. Выход.");
74. Console.WriteLine();
75. }

78. **public** **static** **void** Main(**string**[] args)
79. {
80. info();
81. **int** m;
82. work\_w\_file obj = **new** work\_w\_file("/Users/gulya/Desktop/cours1.txt");
83. obj.reading();
85. **do**
86. {
87. menu();
88. **bool** Tr;
89. **do**
90. {
91. Console.Write("Введите номер пункта: ");
92. Tr = Int32.TryParse(Console.ReadLine(), **out** m);
93. **if** (Tr != **true** || m < 1 || m > 8)
94. Console.WriteLine("Неправильное значение, оно должно быть в пределах диапозона 1...8");
95. **if** (obj.\_null == **true**)
96. {
97. **do**
98. {
99. Console.WriteLine("Файл пуст. Обработка невозможна. Закройте программу и повторите попытку или заполните таблицу, выбрав соответствующий пункт меню!");
100. menu();
101. Console.Write("Введите номер пункта: ");
102. Tr = Int32.TryParse(Console.ReadLine(), **out** m);
103. **if** (Tr != **true** || m < 1 || m > 8)
104. Console.WriteLine("Неправильное значение, оно должно быть в пределах диапозона 1...8");
105. } **while** (obj.\_null != **false** && m != 3);
106. }
108. } **while** (Tr != **true** || m < 1 || m > 8);
110. **if** (m == 1) { obj.show(); }
112. **else** **if** (m == 2)
113. {
114. obj.check\_true\_info();
115. }
116. **else** **if** (m == 3)
117. {
118. **if** (obj.\_null == **true**)
119. {
120. obj.add\_header();
121. obj = **new** work\_w\_file("/Users/gulya/Desktop/cours1.txt");
122. obj.reading();
123. }
125. **string**[] str = **new** **string**[obj.count\_of\_t];
126. Console.WriteLine("Введите Фамилию студента:");
127. str[0] = Console.ReadLine();
129. Console.WriteLine("Введите Имя студента:");
130. str[1] = Console.ReadLine();
132. Console.WriteLine("Введите Отчество студента:");
133. str[2] = Console.ReadLine();
135. Console.WriteLine("Введите баллы студента по Русскому языку:");
136. str[3] = Console.ReadLine();
138. Console.WriteLine("Введите баллы студента по Математике:");
139. str[4] = Console.ReadLine();
141. Console.WriteLine("Введите баллы студента по Информатике:");
142. str[5] = Console.ReadLine();
144. obj.add(str);
145. }
146. **else** **if** (m == 4)
147. {
148. obj.sort\_by\_alf();
149. obj.show();
150. }
151. **else** **if** (m == 5)
152. {
153. Console.WriteLine("Введите фамилию студента, информацию о котором требуется редактировать:");
154. **string** str = Console.ReadLine();
155. obj.edit\_list(str);
157. }
158. **else** **if** (m == 6)
159. {
160. Console.WriteLine("Введите фамилию студента, информацию о котором требуется удалить:");
161. **string** str = Console.ReadLine();
162. obj.del(str);
163. }
164. **else** **if** (m == 7)
165. {
166. obj.write();
167. client(obj.path);
168. }
170. } **while** (m != 8);
172. obj.write();
173. Environment.Exit(0);
174. }
175. }
176. }

# **Приложение Б. Текс библиотеки**

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Linq;
4. **using** System.Text;
5. **using** System.IO;
6. **using** System.Runtime.Serialization.Formatters.Binary;
8. **namespace** FILEWORKING
9. {
10. // атрибут для возможности сериализации объектов класса
11. [SerializableAttribute]
12. **public** **class** work\_w\_file
13. {
14. // путь до файла
15. **public** **string** path;
17. // переменные для чтения файла
18. FileStream Stream;
19. StreamReader Reader;
21. // строчка для чтения файла
22. **string** line;
24. // форматированная таблица
25. **public** **string**[,] info;
27. // количество столбцов
28. **public** **int** count\_of\_t;
30. // переменная-маркер – пустота файла
31. **public** **bool** \_null = **false**;
33. // конструктор класса
34. **public** work\_w\_file(**string** path)
35. {
36. **this**.path = path;
37. **this**.Stream = **new** FileStream(**this**.path, FileMode.OpenOrCreate, FileAccess.ReadWrite);
38. **this**.Reader = **new** StreamReader(Stream);
39. }
41. // проверка на пустоту файла
42. **public** **bool** check\_for\_null(**string** line)
43. {
44. **if** (line != **null**) **return** **true**;
45. **else** **return** **false**;
46. }
48. // чтение файла
49. **public** **string**[,] reading()
50. {
51. // чтение первой строчки
52. line = Reader.ReadLine();
54. // определение полноты/пустоты файла
55. **if** (check\_for\_null(line) == **true**)
56. {
57. // определение количества стобцов таблицы
58. **this**.count\_of\_t = (line.Split(**new** **char**[] { ';' })).Length;
59. **this**.info = **new** **string**[0, count\_of\_t];
61. // построчная запись таблицы в двумерный массив
62. **do**
63. {
64. **string**[] temp = **new** **string**[count\_of\_t];
65. **string**[,] info\_ = rewrite();
67. temp = line.Split(**new** **char**[] { ';' });
69. **for** (**int** i = 0; i < temp.Length; i++)
70. {
71. info\_[info.GetLength(0), i] = temp[i];
72. }
74. info = info\_;
75. line = Reader.ReadLine();
77. } **while** (check\_for\_null(line) == **true**);
78. }
80. **else**
81. {
82. Console.WriteLine("Файл пуст, обработка невозможна! Выйдите из программы и повторите попытку или заполните таблицу, нажав соответствующий пункт меню!");
83. \_null = **true**;
84. }
86. Console.WriteLine();
88. // закрытие файла
89. Reader.Close();
90. Stream.Close();
92. **return** info;
93. }
95. **public** **void** add\_header()
96. {
97. **string** header = "Фамилия";
98. header += ";";
99. header += "Имя";
100. header += ";";
101. header += "Отчество";
102. header += ";";
103. header += "Русский язык";
104. header += ";";
105. header += "Математика";
106. header += ";";
107. header += "Информатика";
109. FileStream f = **new** FileStream(**this**.path, FileMode.Truncate, FileAccess.ReadWrite);
110. StreamWriter sw = **new** StreamWriter(f);
111. sw.WriteLine(header);
112. sw.Close();
113. f.Close();
114. }
116. // добавление строчки
117. **public** **void** add(**string**[] str)
118. {
119. **string**[,] info\_ = rewrite();
121. **for** (**int** i = 0; i < count\_of\_t; i++)
122. {
123. info\_[info.GetLength(0), i] = str[i];
124. }
126. **this**.info = info\_;
127. }
129. // удаление строчки
130. **public** **void** del(**string** del\_)
131. {
132. **int** i = 0;
133. **int** n = 0;
134. **string**[,] info\_ = **new** **string**[(info.GetLength(0) - 1), count\_of\_t];
136. **do**
137. {
138. **if** (info[i, 0] == del\_)
139. {
141. **for** (**int** k = 0; k < i; k++)
142. {
143. **for** (**int** j = 0; j < count\_of\_t; j++)
144. {
145. info\_[k, j] = info[k, j];
146. }
147. }
149. **for** (**int** k = i + 1; k < info.GetLength(0); k++)
150. {
151. **for** (**int** j = 0; j < count\_of\_t; j++)
152. {
153. info\_[k - 1, j] = info[k, j];
154. }
155. }
157. n = 1;
158. }
159. **else** i++;
161. } **while** ((n != 1) && (i < info.GetLength(0)));
163. **if** (n == 0)
164. {
166. Console.WriteLine("Студент не найден. Проверьте данные и повторите попытку.");
167. }
169. **else** info = info\_;
170. }
172. // перезапись (копирование)
173. **public** **string**[,] rewrite()
174. {
175. **string**[,] info\_ = **new** **string**[(info.GetLength(0) + 1), count\_of\_t];
177. **for** (**int** i = 0; i < info.GetLength(0); i++)
178. {
179. **for** (**int** j = 0; j < count\_of\_t; j++)
180. {
181. info\_[i, j] = info[i, j];
182. }
183. }
185. **return** info\_;
186. }
188. // вывод информации на экран
189. **public** **void** show()
190. {
191. Console.WriteLine();
193. **for** (**int** i = 0; i < info.GetLength(0); i++)
194. {
195. **for** (**int** j = 0; j < info.GetLength(1); j++)
196. {
197. Console.Write($"{info[i, j],-15}");
198. }
199. Console.WriteLine();
200. }
201. Console.WriteLine();
202. }
204. // запись массива обратно в файл
205. **public** **void** write()
206. {
207. FileStream f = **new** FileStream(**this**.path, FileMode.Truncate, FileAccess.ReadWrite);
208. StreamWriter sw = **new** StreamWriter(f);
209. **for** (**int** i = 0; i < info.GetLength(0); i++)
210. {
211. **string** s = info[i, 0];
212. **for** (**int** j = 1; j < count\_of\_t; j++)
213. {
214. s = s + ';' + info[i, j];
215. }
216. sw.WriteLine(s);
217. }
219. sw.Close();
220. f.Close();
221. }
223. //вспомогательная функция для сортировки по алфавиту
224. **protected** **static** **bool** needToReOrder(**string** s1, **string** s2)
225. {
226. **for** (**int** i = 0; i < (s1.Length > s2.Length ? s2.Length : s1.Length); i++)
227. {
228. **if** (s1.ToCharArray()[i] < s2.ToCharArray()[i]) **return** **false**;
229. **if** (s1.ToCharArray()[i] > s2.ToCharArray()[i]) **return** **true**;
230. }
231. **return** **false**;
232. }
234. // сортировка по алфавиту
235. **public** **void** sort\_by\_alf()
236. {
237. **string**[] temp = **new** **string**[6];
238. **for** (**int** i = 2; i < info.GetLength(0); i++)
239. {
241. **for** (**int** j = 2; j < info.GetLength(0) - 1; j++)
242. {
243. **if** (needToReOrder(info[j, 0], info[j + 1, 0]))
244. {
245. **for** (**int** k = 0; k < info.GetLength(1); k++)
246. {
247. temp[k] = info[j, k];
248. }
250. **for** (**int** k = 0; k < info.GetLength(1); k++)
251. {
252. info[j, k] = info[j + 1, k];
253. }
255. **for** (**int** k = 0; k < info.GetLength(1); k++)
256. {
257. info[j + 1, k] = temp[k];
258. }
259. }
260. }
261. }
263. Console.ReadKey();
264. }
266. // поиск фамилии
267. **private** **int** search\_of\_string\_by\_surname(**string** surname)
268. {
269. **int** k = 0;
270. **int** n = 0;

273. **while** ((k < info.GetLength(0)) && (n == 0))
274. {
275. **if** (info[k, 0] == surname) { n++; }
276. k++;
277. }
278. **if** (n != 0) **return** k - 1;
279. **else** **return** -1;
280. }

283. // изменение информации
284. **public** **void** edit\_list(**string** surname)
285. {
286. **string** inf;
287. **if** (search\_of\_string\_by\_surname(surname) != -1)
288. {
289. **int** k = search\_of\_string\_by\_surname(surname);
290. Console.WriteLine("Какую информацию вы хотите изменить?");
291. inf = Console.ReadLine();
293. **if** (inf == info[1, 0])
294. {
295. Console.WriteLine("Введите новую фамилию студента");
296. info[k, 0] = Console.ReadLine();
297. }
298. **else** **if** (inf == info[1, 1])
299. {
300. Console.WriteLine("Введите новое имя студента");
301. info[k, 1] = Console.ReadLine();
302. }
303. **else** **if** (inf == info[1, 2])
304. {
305. Console.WriteLine("Введите новое количество баллов по русскому языку");
306. info[k, 2] = Console.ReadLine();
307. }
308. **else** **if** (inf == info[1, 3])
309. {
310. Console.WriteLine("Введите новое количество баллов по математике");
311. info[k, 3] = Console.ReadLine();
312. }
313. **else** **if** (inf == info[1, 4])
314. {
315. Console.WriteLine("Введите новое количество баллов по информатике");
316. info[k, 4] = Console.ReadLine();
317. }
318. }
319. **else** Console.WriteLine("Такого студента в списке нет!");
320. }
322. // проверка на истинность информации
323. **public** **void** check\_true\_info()
324. {
325. **for** (**int** i = 1; i < info.GetLength(0); i++)
326. {
327. **if** ((**int**.Parse(info[i, 3]) >= 0) && (**int**.Parse(info[i, 4]) >= 0) && (**int**.Parse(info[i, 5]) > 0) && (**int**.Parse(info[i, 3]) <= 100) && (**int**.Parse(info[i, 4]) <= 100) && (**int**.Parse(info[i, 5]) <= 100))
328. {
330. }
331. **else** { **this**.del(info[i, 0]); }
332. }
333. }
334. }
335. }

# **Приложение В. Техническое задание**

****

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

(**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**)» (**МАИ)**

Кафедра «Технология испытаний и эксплуатации»

**Программный комплекс**

**«База данных студентов ВУЗа»**

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

20902 – 01 81 01

Листов 7

Согласовано Задание принял к исполнению  
\_\_\_\_\_\_\_ / Молчанова С.И. студент группы

«\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

\_\_\_\_\_\_\_\_ /Атрашков Н. А.

«06» апреля 2021 г.

2021

**Введение**

Программный комплекс «**База данных студентов ВУЗа**» (далее DataBaseOfStudent) может быть использован сотрудниками деканата для формирования отчетов.

1. **Основание для разработки**

Основанием для разработки является учебный план по дисциплине  
«Технологии программирования».

1. **Назначение разработки**

Программа DataBaseOfStudent является курсовой работой по дисциплине «Технологии программирования» и предназначается для создания, редактирования и сериализации БД студентов ВУЗа.

1. **Требования к программе**

**3.1. Требования к функциональным характеристикам**

Программа DataBaseOfStudent должна выполнять следующие функции:

* организацию меню;
* вывод сведений об авторе;
* вывод краткой теории к каждому разделу;
* открытие форматированной таблицы;
* создание форматированной таблицы;
* редактирование данных;
* сериализацию данных;
* передачу данных по сети;
* форматированный вывод данных на экран;
* формирование отчета о работе программы;
* выдачу диагностических сообщений;
* создание интуитивно понятного интерфейса.
  1. **Организация входных и выходных данных**

Организация входных данных

Входными данными для программы являются:

* целое положительное число – ключ меню;
* текстовый файл, содержащий отформатированную таблицу информации о студентах университета, с полями, представленными в таблице 1;

Таблица 1 – Пример входных данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Фамилия | Имя | Отчество | Русский язык | Математика | Информатика |
| Атрашков | Никита | Андреевич | 92 | 86 | 90 |
| Иванов | Алексей | Юрьевич | 72 | 68 | 84 |
| Петрова | Мария | Сергеевна | 80 | 55 | 67 |

Организация выходных данных

Выходными данными для программы являются:

* текстовая информация, содержащая краткую теорию по каждому разделу на экране;
* результаты работы команд;
* сохраненная форматированная таблица в файле бинарного (двоичного) формата;
* перечень диагностических сообщений;
* отчет о работе программы.
  1. **Требования к надёжности**  
      Для обеспечения надежного функционирования программы необходимо предусмотреть проверку ключей меню, открытия файла и записи в файл и выдачу диагностических сообщений при обнаружении ошибок.
  2. **Условия эксплуатации**

Для эксплуатации программы не требуется специально подготовленный пользователь.

* 1. **Требования к составу и параметрам технических средств**

Для функционирования программы необходимо наличие следующих технических средств:

* процессор Intel® Core™ i5 inside™;
* не менее 512 Мб оперативной памяти;
* монитор SVGA с минимальным разрешением 800х600 пикселей;
* манипулятор типа «мышь» и клавиатура.

**3.6. Требования к информационной и программной совместимости**

Для разработки программы DataBaseOfStudent необходим следующий инструментарий:

* MS VisualStudio 2019 Community – платформа выбрана для использования языка С#;
* операционная система MS Windows 7 или выше.

1. **Требования к программной документации**

Состав программной документации

* «Программа «**База данных студентов ВУЗа**». Техническое задание.
* «Программа «**База данных студентов ВУЗа**». Пояснительная записка.

1. **Стадии и этапы разработки**

**5.1. Эскизный проект**

Обзор существующих программных решений.

Предварительная разработка структуры входных и выходных данных.

Разработка общего описания алгоритма.

Срок окончания: 7-9 недели

**5.2. Технический проект**

Уточнение структуры входных и выходных данных.

Разработка средств обработки ошибок и выдачи диагностических сообщений.

Окончательное определение конфигурации технических и программных средств.

Разработка пояснительной записки.

Срок окончания: 11-12 недели

* 1. **Рабочий проект**

Написание программы.

Тестирование и отладка программы.

Разработка программной документации.

Разработка программы и методики испытаний.

Корректировка программы и программной документации по результатам испытаний.

Срок окончания:13-16 недели

* 1. **Внедрение**

Защита курсовой работы.

Срок окончания:17-18 недели

1. **Порядок контроля и приёмки**

Контроль и приёмка разработки осуществляется в соответствии с программой и методикой испытаний, изложенными в пояснительной записке.

1. **Перечень сокращений**

SVGA – Super Video Graphics Array – улучшенный видеографический адаптер, обеспечивающий разрешающую способность 1280х1024 точек и более чем 256 цветов.

Мб – Мегабайт – единица измерения объема информации.

1. **Перечень используемой литературы**

* Курс лекций по дисциплине «Объектно-ориентированное программирование» Молчанова С.И.
* Документация C#. Microsoft
* Полное руководство по языку программирования С# 9.0 и платформе .NET 5  
  Электронный ресурс: https://metanit.com
* Основы С#  
  Электронный ресурс: https://habr.com

# **Приложение Г. База данных**

https://mairu-my.sharepoint.com/:t:/g/personal/emsidorenkova\_mai\_ru/ESoFWoPY8\_FJuBYuMypnHIkB-OV8WQYwGMUn2EaJ8fC50w?e=dlfJQG