содержание

**ВВЕДЕНИЕ ………………………………………………………………………………..4**

**1. ПОСТАНОВКА ЗАДАЧИ ……………………………………………………………..5**

**2. РАЗАРАБОТКА АЛГОРИТМОВ……………………………………………………..8**

**3. РАЗРАБОТКА ПРОГРАММЫ………………………………………………………..11**

**3.1. Выбор средств программирования.....………………………………………….......11**

**3.2. Разработка модулей........……………………………………………………….........13**

**4. ТЕСТИРОВАНИЕ……………………………………………………………………....18**

**4.1. Описание входных и выходных данных.....……………………………………......18**

**4.2. Результаты тестирования.....………………………………………………………...18**

**ЗАКЛЮЧЕНИЕ......……………………………………...………………………………...28**

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ..........……………………...…......29**

**ПРИЛОЖЕНИЕ А ТЕКСТ ПРОГРАММЫ**

**ПРИЛОЖЕНИЕ Б ГРАФИЧЕСКИЙ МАТЕРИАЛ**

# ВВЕДЕНИЕ

База данных «Страны мира» предназначена для ведения учета информации о «крупнейших странах-автопроизводителях». 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 «Страны мира», представленной в отдельном текстовом файле.

Для достижения поставленной цели в курсовой работе необходимо выполнить решение задач, описанных в следующем пункте.

**1. ПОСТАНОВКА ЗАДАЧИ**

Для начала определим предметную область, чтобы знать, какие цели и задачи необходимо поставить. Предметная область, подлежащая изучению - «страны мира». В сферу этой предметной области попадают сами страны, информация об их экономическом и политическом взаимодействии, а также и о внутреннем состоянии.

Необходимо разработать приложение, которое будет обрабатывать массив структурированных данных «Страны мира» и которое будет, в том числе, вести учёт по самим странам. Бинарный файл исходных данных должен содержать следующие поля: название страны, название столицы, численность населения страны, площадь страны, код валюты стран. Однако структура «Страны мира» будет содержать дополнительно поле индекса и поле рассчитываемого значения. Стоит отметить также, что бинарный файл будет содержать рабочую строку, в которой будет указана размерность массива, т.е. 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.

В программе должно быть реализовано:

– дополнительное поле с рассчитываемым значением – «Плотность населения»;

– ввод информации из бинарного 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 индексно-бинарных файлов;

– поиск информации в массиве структур по любому полю;

– вычисление количества записей по задаваемому интервалу.

– структуры, способ хранения записей – бинарный файл.

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 предприятием 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 страны очень тесно связаны друг с другом в информационном плане. Существует множество различный баз данных между странами, причём совершенно в любой отрасли. Поэтому и эта база данных, которая описывается в этом курсовом проекте, ориентирована на использование единой базы данных, являющейся совокупностью структурированных данных, предназначенных для многоцелевого и многократного их применения. А также методов доступа к ним с целью уменьшения степени трудоёмкости, увеличения производительности, повышения точности и надёжности выполняемых операций и задач. Поэтому базу данных «страны мира» целесообразно автоматизировать. Заметим, что данная база данных носит глобальный характер, преимущественно о состоянии страны. Под этим понимается её числовые характеристики, такие как численность населения, площадь и плотность населения, и некоторые географические и экономические элементы, такие как название страны и ей столицы, а также код валюты.

Автоматизированная система позволит приводить в порядок информацию по базам данных стран мира и их состояния, возможность обновления в связи с изменением некоторых экономических, политических, топографических и других причин. Изменения всё-таки не так редки, как может показаться на первый взгляд, поэтому можно сказать, что это актуально. А также использование баз данных — возможность надёжного хранения информации.

**2. РАЗРАБОТКА АЛГОРИТМОВ**

Этот этап один из самых важных. После того, как мы осуществили постановку задачи, для дальнейшей работы нам будет необходимо разработать алгоритмы. Разработать алгоритм - это означает, что 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. Для решения поставленной задачи потребуется создать простой и удобный алгоритм взаимодействия пользователя с базой данных «страны мира». Для этого в основном модуле программы нужно реализовать меню. Алгоритм работы программы будет заключаться в следующем: в начале работы будет проводиться проверка на наличие файла с сохраненными ранее данными, если файл отсутствует в корневой папке, то указываем путь к файлу, т.е. полное имя файла.

Алгоритм ввода информации из файла в массив записей:

1. Открываем файл в корневой папке программы для чтения.
2. Если файл не найден, то указываем полное имя файла и открываем его.
3. Читаем рабочую строку из файла, получаем количество записей, расширяем

динамический массив на считанное значение количества записей.

1. В случае, если файл пустой:
   1. Записываем самую первую строку
   2. Сохраняем эти данные в бинарный файл
2. В случае, если файл не пустой:
   1. Читаем строки из файла пока процесс считывания данных не достигнет

конца файла.

5.2. Записываем прочитанные строки в массив данных

5.3. Закрываем файл.

6. Изменяем содержание индексно-бинарных файлов соответственно.

Также сразу опишем следующие алгоритмы.

Алгоритм вывода информации в текстовый или бинарный файл:

1. Открываем файл для записи.

2. Перезаписываем весь файл целиком после различных изменений записей, с указанием рабочей строки

3. Закрываем файл.

Стоит отметить, что рабочим файлом считается бинарный файл.

Алгоритм вывода информации в индексно-бинарные файлы:

1. Открываем файлы для записи.

2. Перезаписываем весь файл целиком после различных изменений записей в соответствии с сортировкой по полям.

3. Закрываем файлы.

После успешного прохождения всех этих действий осуществляется вызов меню, с помощью которого пользователь взаимодействует со всеми остальными возможностями программы, алгоритмы которых, приведены ниже.

Алгоритм добавления новых элементов:

1. Запрашиваем индекс записи, после которой необходимо добавить запись.

2. Запрашиваем количество новых записей.

3. Выделяем память для новых элементов структуры с увеличением размерности и отсортировываем массив для добавления. Другими словами, освобождаем место для записи, которую хотим ввести, опуская нижестоящие записи.

4..Запрашивается информация о странах и производится непосредственный её ввод.

5. Изменяем содержание рабочего файла по алгоритму ввода информации в текстовый или бинарный файл.

6. Изменяем содержание индексных бинарных файлов соответственно

Алгоритм вывода элементов массива на консоль:

1. Запускаем цикл для всех элементов массива.

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

Алгоритм корректировки полей элементов массива:

1. Выбираем индекс элемента, для корректировки полей.
2. Выбираем режим корректировки данных: корректировка всей записи

целиком или частичная корректировка данных записи.

1. Если корректировка частичная, то
   1. . Выбираем поле для корректировки.
   2. . На основании выбора вносим новое значение выбранного поля.
2. Если корректировка полная, то
   1. . Поочерёдно вносим изменения каждого поля записи

5. Изменяем содержание рабочего файла по алгоритму ввода информации в текстовый или бинарный файл.

6. Изменяем содержание индексных бинарных файлов соответственно

Алгоритм поиска записи по значению ключевого поля:

1. Выбираем поле для поиска и вводим значение соответствующего поля.

2. Запускаем цикл для всех элементов массива.

3. Выводим запись с заданным значением на экран.

Алгоритм удаления выбранного элемента:

1. Находим запись, которую необходимо удалить одним из предложенных способов: удаление по индексу или поиск записи по полю.

2. Удаляем выбранный элемент из массива.

3. Уменьшение размерности массива и сдвигаем записи вверх, закрывая запись, которая должна быть удалена.

4. Запрос на ещё одно удаление.

5. Если удалять более не нужно, то запрос на сохранения изменений, иначе п.1.

6. Если необходимо сохранить изменения, то

6.1. Изменяем содержание рабочего файла по алгоритму ввода информации в текстовый или бинарный файл

6.2. Изменяем содержание индексных бинарных файлов соответственно.

7. Если сохранять изменения не надо, то

7.1. Аннулирование размерности динамического массива.

7.2. Далее действия идут по алгоритму ввода информации из бинарного файла в массив записей.

Алгоритм сортировки записей по выбранному полю:

1. Выбор поля для сортировки.
2. Открываем соответствующий индексно-бинарный файл.
3. Считываем информацию из индексно-бинарных файлов.
4. Закрываем соответствующий индексно-бинарный файл.

5. В соответствии со считанными индексами, выводим на экран отсортированные записи.

Алгоритм вывода записей, принадлежащему вводимому промежутку численности населения, и подсчёт количества таких записей.

1. Вводи промежуток численности населения

2. Запускаем цикл сравнения значений численности населения стран в соответствии с указанным промежутком.

3. Увеличиваем маячок количества записей, удовлетворяющих указанному промежутку.

4. Вывод всех стран, чья численность населения удовлетворяет указанному промежутку, и количества таких записей.

**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 – 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. iomanip - реализует инструменты для работы с форматированием вывода. conio.h – для функции \_getche(), которая обеспечивает мгновенный ввод символа и вывода его в консоль. Это обеспечивает быстрое и комфортное перемещение по программе. stdio.h – ещё одна библиотека для работы с файлами. cmath – для функции pow(), позволяющая возводить число в степень. Эта функция будет использоваться в блоках защиты от некорректного ввода.

Сейчас опишем разные ключевые переменные, которые встречаются в программе. Ключевыми переменными являются массив land[] типа lander и его размерность Kapazitat типа int. Kapazitat в некоторых функция и в блок-схемах (для сокращения) указывается как K. Крайне важен массив указателей на функции Arbeiter. Было решено организовать массив указателей на функции по той причине, что он крайне удобен для того, чтобы программу разбивать на модули и там определять основные функции. Если бы это организовывалось через switch, то тогда бы код программы был очень громоздким, со множеством различных вложенностей, что делало бы его не читаемым, даже с комментариями. Также стоит заметить, что даже в самих модулях присутствует switch, поэтому это было не хорошей идеей организовывать несколько switch в главном swicht. Такие переменные как ctrl, laufgang, Antwort типа char являются, так сказать, переменными выбора и ответа, хода по меню, хода по функции. Также из общих ключевых переменных можно выделить metaindex типа int. Эта переменная является ключевой, т.к. она много где используется и имеет различные функции, например, используя metaindex мы сдвигаем записи массива при удалении или добавлении записей. Также ключевой переменной является переменна маяка leuchturm типа int. Она используется для подсчёта совпадений и относительно её результата предпринимаются те или иные действия. Пожалуй, самым важным элементом является массив Zeitspanne типа char. Он предназначен для защиты от некорректного ввода любой информации. Он претерпевает на себе множество проверяющих условий, а также корректировку, после чего только идёт переопределение значений массива записей. Стоит отметить минус такого способа ввода значений в массив записей – громоздкость. Но всё же, плюс его полностью перевешивает – практически абсолютная защита от аварии программы. Практически, за исключение ахиллесовой пяты, которой является переполнение самого защитного массива. Если ввести более 65 символом, тогда программа потерпит аварию. Последними ключевыми переменными являются переменные для работы с файлами: wanderung – для чтения и файла, reisen – для запись в файл, IndexFileR – для чтения информации с индексно-бинарных файлов. IndexReisen – для запись в индексно-бинарные файлы.

Теперь опишем некоторые частные переменные. Переменная metaint типа int используется для сравнения преимущественно в функции поиска. Массив metamass[] типа int для считывания индексов с индексно-бинарного файла. Значения массива используются для сравнения, которое выводит записи в порядке индексов массива, что является сортировкой. Переменная x типа int определяет количество новых записей в функции добавления записей. Переменная DerNameDesTextes класса string является именем файла, который открывает программа. Переменные NiedrigeGrenze и HoheGrenze являются переменные границ промежутка, по которому происходит выборка. Переменная gang для режима корректировки данных. Массив TeilDel[] типа char выполняет туже функцию, что и Zeitspanne. Возникла необходимость в нём только по причине того, что необходимо было ввести данные, но значение Zeitspanne должно было сохраниться.

Стоит заметить, что при разработке программы использовался оператор перехода goto, причём многократно. Его плюс заключается в том, что можно избежать многократной вложенности. К тому же, в некоторых местах он упрощает написание самого кода и блок-схем. Также с помощью goto был выполнен дерзкий переход в функции удаления, когда был определён metaindex.

Ввод ответных переменных осуществляется через \_getche(). Это обеспечивает быстрое и удобное взаимодействие с программой. Минус: если уже введена переменная, то исправить ей нельзя, как это можно было бы сделать с cin >> - ввести значение, но не нажать Enter, a потом исправить.

Для форматированного вывода использовались функции setiosflags(ios::left) и setw(). Также была использована функция pow() для организации математической логики преобразования стоки в число.

Для сортировки использовалась пользовательский метод сортировки, похожий на метод пузырьком. Плюс: занимает мало места и прост. Минус: при огромной количестве данных, долго работает.

Отметим, почему не был организован массив указателей в самих функциях, где используется swicht. Причина тому, чтобы не загромождать библиотеки множеством функций. В случае бы такой организации, приходилось бы держать в голове все названия функций или долго искать их, чтобы что-то исправить

**3.2. Разработка модулей**

Программа разбита на 6 основных модулей: KursArbeit.cpp, Dynamik.h и Dynakim.cpp, Statik.h и Statik.cpp, Suchen.h и Suchen.cpp, Sortierung.h и Sortierung.cpp, Peripherie.h и Peripherie.cpp – и библиотеку lender.h. Это необходимо и рационально, т.к. весь код программы очень длинный. Для того, чтобы лучше ориентироваться в коде, программа и была разбита на модули.

**Модуль KursArbeit.cpp**

Является основным модулем программы, который содержит функцию чтений из файла, описанной в другом модуле, и функцию int main(), где содержится меню программы. На запрос о вводе пункта меню, вводится соответственно сам пункт. Далее, перед тем, как произойдёт распределение по организованному в этом модуле массиву указателей на функции, произойдёт проверка вводимых значений пункта. Если были введены некорректные данные, по типу символа, не относящегося к числу или несуществующего пункта меню, то выводится оповещение о некорректном вводе и предлагается ввести пункт заново. После уже корректного ввода идёт распределение по пунктам меню, каждый из которых указывает на соответствующую функцию. После выполнения работы функции, программа снова возвращается в основной модуль и выводит меню работы с файлом. Такой возврат будет продолжаться до тех пор, пока не будет избран пункт окончания работы программы.

**Библиотека lender.h**

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

**Модуль Dynamik.h**

Содержит в себе прототипы функций, описанных в “**Dynamik.cpp**”.

**Модуль Dynamik.cpp**

Содержит в себе реализацию всех функций, объявленных в модуле “**Dynamik.h**”. В основном это функции, которые изменяют содержание файла. Также содержит глобальные переменные записи в файл и чтения из файла, переменную имени файла. Описание функций представлено ниже.

1. **Функция void LesenDaten(lander\*& land, int& Kapazitat)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, **int& Kapazitat** - ссылка на размерность.

Назначение: чтение данных из бинарного файла и заполнения массива записей.

Также организует защиту от пустого файла, предложив заполнить первую строку.

Возвращаемое значение: значений не возвращает.

1. **Функция void AddierenDaten(lander\*& land, int& Kapazitat)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, **int& Kapazitat** - ссылка на размерность.

Назначение: добавление новых записей после указанной записи.

Возвращаемое значение: значений не возвращает.

**3. Функция**  **void SubstragierenDaten(lander\*& land, int& Kapazitat)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, **int& Kapazitat** - ссылка на размерность.

Назначение: удаление выбранного элемента из записей с запросом о сохранении изменений и возможностью возврата в исходное состояние.

Возвращаемое значение: значений не возвращает.

**4. Функция void SpeichernDaten(lander\*& land, int& Kapazitat)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, **int& Kapazitat** - ссылка на размерность.

Назначение: сохранение данных в тектсовый файл для наглядности.

Возвращаемое значение: значений не возвращает.

**Модуль Statik.h**

Содержит в себе прототипы функций, описанных в “**Statik.cpp**”.

**Модуль Statik.cpp**

Содержит в себе реализацию всех функций, объявленных в модуле “**Statik.h**”. В основном это простые основные функции, которые не изменяют файл. Описание функций представлено ниже.

1. **Функция void SreibenDaten(lander\*& land , int& Kapazitat)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, **int& Kapazitat** - ссылка на размерность..

Назначение: вывод данных массива записей в консоль.

Возвращаемое значение: значений не возвращает

1. **Функция void WahlDaten(lander\*& land , int& Kapazitat)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, **int& Kapazitat** - ссылка на размерность.

Назначение: вывод записей, которые удовлетворяют задаваемому промежутку численности населения, и подсчёт таких записей.

Возвращаемое значение: значений не возвращает.

**Модуль Suchen.h**

Содержит в себе прототипы функций, описанных в “ **Suchen.cpp** ”.

**Функция Suchen.cpp**

Содержит в себе реализацию всех функций, объявленных в модуле “**Suchen.h**”. В основном это функции, которые с поиском данных. Описание функций представлено ниже.

1. **Функция void SuchenDaten(lander\*& land , int& Kapazitat)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, **int& Kapazitat** - ссылка на размерность.

Назначение: поиск по любому полю и вывод соответствующих записей.

Возвращаемое значение: значений не возвращает.

**2.Функция int SuchenFurSubstragierung(lander\*& land, int& Kapazitat)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, **int& Kapazitat** - ссылка на размерность.

Назначение: поиск по любому полю и запрос на удаление записи.

Возвращаемое значение: возвращает целочисленное значение, интерпретируемое как индекс записи, которую необходимо удалить.

**Модуль Sortierung.h**

Содержит в себе прототипы функций, описанных в “ **Sortierung.cpp** ”.

**Модуль Sortierung.cpp**

Содержит в себе реализацию всех функций, объявленных в модуле “**Sortierung.h**”. В основном это функции, которые с сортировкой данных. Описание функций представлено ниже.

1. **Функция void SortieranDaten(lander\*& land , int& Kapazitat)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, **int& Kapazitat** - ссылка на размерность.

Назначение: вывод отсортированных данных.

Возвращаемое значение: значений не возвращает.

**2. Функция void SortierungNachVeranderungen(lander\*& land , int& K)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, **int& Kapazitat** - ссылка на размерность.

Назначение: перезапись индексно-бинарных файлов сразу, после каких-либо изменений в основном файле. При перезаписи данные сортируются по всем полям, а результаты сортировки в виде отсортированных индексов записываются в соответствующие файлы.

Возвращаемое значение: значений не возвращает.

**3. Функция int HeranSortierung(lander\*& land , int i, int j)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, счётчик цикла i, счётчик цикла j.

Назначение: «прямой ход сортировки» - выполняется непосредственную сортировку, вызываясь в цикле не один раз.

Возвращаемое значение: целочисленный 0 – аннулирование счётчика в том цикле, в котором вызывается функция.

**4. Функция void ZuruckSortierung(lander\*& land , int& Kapazitat)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса структуры, **int& Kapazitat** - ссылка на размерность.

Назначение: «обратный ход сортировки» - возвращает истинный порядок записей в соответствии с индексами. Другими словами, сортировка по индексам.

Возвращаемое значение: значений не возвращает.

**Модуль Peripherie.h**

Содержит в себе прототипы функций, описанных в “ **Peripherie.cpp** ”.

**Модуль Peripherie.cpp**

Содержит в себе реализацию всех функций, объявленных в модуле “ **Peripherie.h**”. В основном это различные вспомогательные функции. Описание функций представлено ниже.

1. **Функция void Mutze()**

Входные параметры: нет принимаемых параметров.

Назначение: вывод элементов шапки в консоль.

Возвращаемое значение: значений не возвращает.

1. **Функция void Prufung()**

Входные параметры: нет принимаемых параметров.

Назначение: оповещение о некорректном вводе и вывод небольшого руководства по вводу.

Возвращаемое значение: значений не возвращает.

1. **Функция void Wartung()**

Входные параметры: нет принимаемых параметров.

Назначение: организация задержки.

Возвращаемое значение: значений не возвращает.

1. **Функция void Schreiben(lander\*& land , int i)**

Входные параметры: **lander\*& land** - ссылка на указатель на объект класса

счётчик цикла i.

Назначение: вывод i-той записи.

Возвращаемое значение: значений не возвращает.

1. **Функция void KorrigierenDaten(char Zeitspanne[], int gang)**

Входные параметры: **char Zeitspanne** – массив char, строка, **int gang** –режим

корректировки.

Назначение: коррекция введённых данных.

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 массив. Затем на экране появляется меню для работы с нашим массивом. В ходе выполнения программы происходит изменение информации, хранимой в массиве и в файле. Немного об ограничениях. Для работы программы требуется рабочая строка с указанным там точным количеством записей. Ограничение по длине для названия страны и её столицы составляет 25 символов, для кода валюты – 3 символа. Числовые переменные имеют usigned, т.к. площадь и численность населения не могут быть отрицательными, а значит и плотность не может быть отрицательной. Однако пользователю не нужно будет держать это в голове, т.к. в программе реализована защита от некорректного ввода. Более того реализована и корректировка вводимых данных для корректной работы программы.

Файл с информацией Lender.txt (для наглядности) (см.рисунок 4.1):

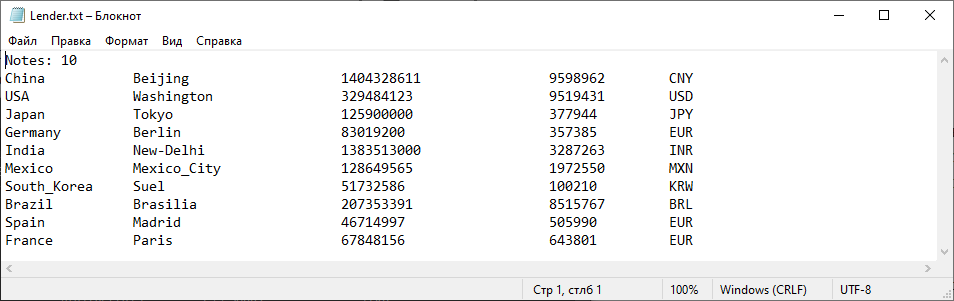


Рисунок 4.1 - Файл с информацией

**4.2. Результаты тестирования**

При запуске программа автоматически считывает данные с файла, который хранится в корневой папке. В случае, если в корней папки программы нет бинарного файла, то выводится оповещение о том, что файл не найден, и запрос, что необходимо вручную ввести полное имя файла, для работы с ним (см.рисунок 4.2).

Рисунок 4.2 – Оповещение о ненайденном файле

После открытия и считывания данных с файла выводится меню (см.рисунок 4.3) для непосредственной работы со считанными данными. В меню предлагается выбрать один из 9 пунктов, считая нулевой пункт – выход из программы. Ввод производится с помощью функции \_getche(). После этого происходит проверка на введённые символы.

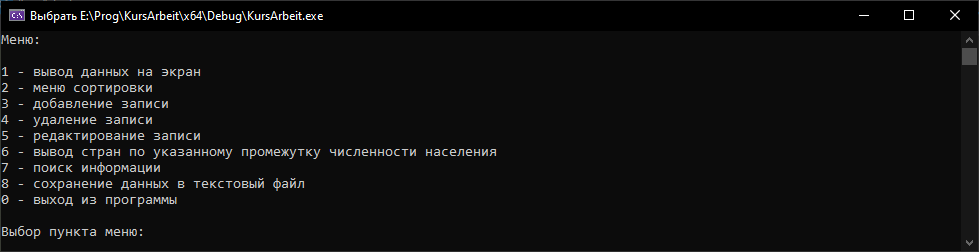
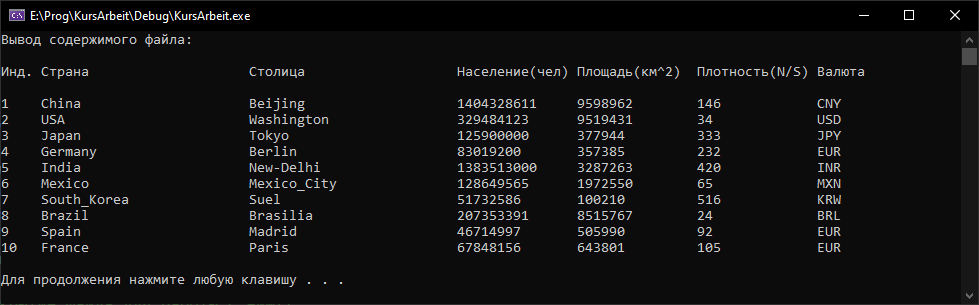


Рисунок 4.3 – основное меню

В пункте 1 осуществляется вывод данных в консоль (см.рисунок 4.4).

Рисунок 4.4 – вывод содержания файла

В пункте 2 осуществляется сортировка данных. При нажатии на ‘2’ выводится меню сортировки (см.рисунок 4.5)

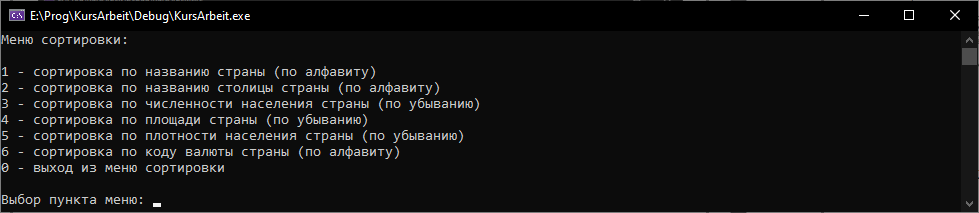


Рисунок 4.5 – меню сортировки данных

Для наглядности выведем в консоль результаты двух сортировок: сортировку по алфавиту – для слов (см.рисунок 4.6) – , а также сортировку по убыванию – для чисел (см.рисунок 4.7)

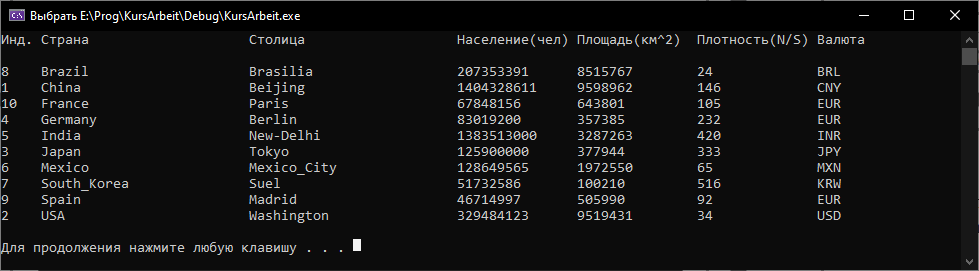


Рисунок 4.6 – сортировка по алфавиту

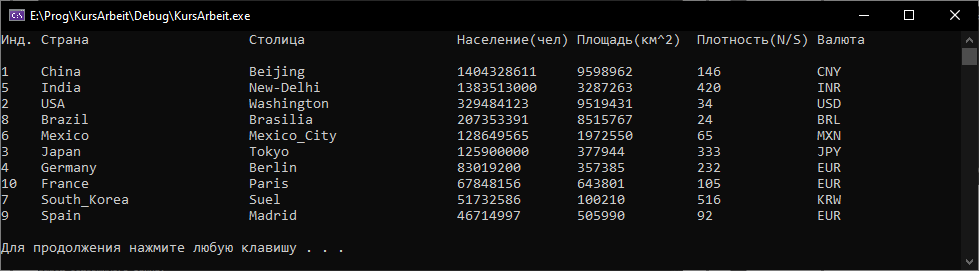


Рисунок 4.7 – сортировка по убыванию

В пункте 3 осуществляется добавление записей. Для того, чтобы добавить запись необходимо ввести следующую информацию. Необходимо, во-первых, указать индекс, после которой будет происходить добавление новой записи. Предварительно перед этим выведутся все данные для того, чтобы необходимый индекс не нужно было запоминать, а найти его в пункте добавления, не используя другие пункты. Во-вторых, необходимо ввести количество новых записей, которые необходимо добавить в массив, а в дальнейшем записать в файл. После ввода этих данных, выводится запрос на ввод данных в массив (см.рисунок 4.8). На ввод во всей программе в любом месте распространяется защита от некорректного ввода, по\тому самое время пояснить, что считается некорректным вводом. В этой программе некорректным вводом может считаться несуществующий пункт меню или некорректный символ – при вводе единичного символа (преимущественно в меню, а также в местах, где необходимо дать однозначный ответ одним символом). Некорректным вводом также считается слишком длинное значение совершенно любого поля. Для каждого поля определён их объём символов. Для ввода слов некорректным вводом считается символы, которые не относятся к символам латиницы и символам « - » и « \_ ». Для числа некорректным вводом считается любые символы, не относящиеся к нулю.

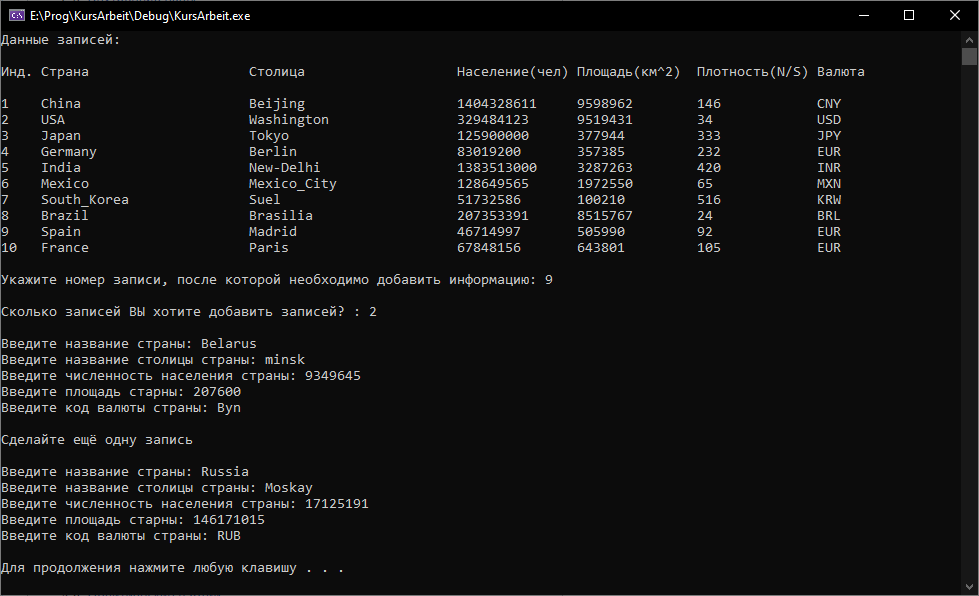


Рисунок 4.8 – добавление записей

К слову заметим работу функции корректировки вводимых данных (см.рисунок 4.9). Слово «minsk» было написано с прописной буквы, однако в массиве оно записалось с заглавной буквы. Точно также отработано и с кодом валюты страны. Повышение регистора символа производится также после символов « - » и « \_ ».

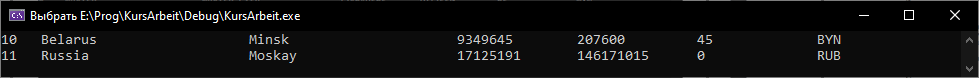


Рисунок 4.9 – результат корректировки данных

Также на счёт нуля в плотности населения у России. Были намеренно введены неверные данные. В площадь страны вводилась её численность населения, а в численность населения – её площадь. Это было сделано для демонстрации функции корректировки данных. Также значения плотности населения относятся к целочисленному типу, поэтому дробная часть не выводится.

В пункте номер 4 осуществляется удаление элементов. Первично выводится меню удаления (см.рисунок 4.10), где предлагается выбрать способ удаления: удаление по индексу или удаление по полю (поиск данных для удаления).

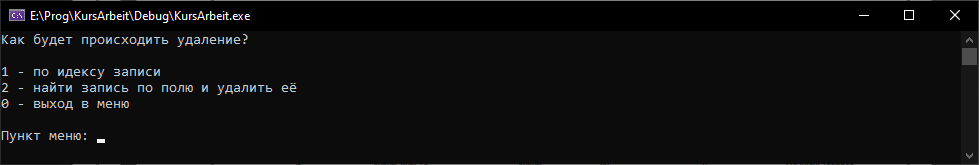


Рисунок 4.10 – меню удаления

Удаление по индексу: для начала выводится весь список данных, затем запрашивается индекс записи, которую необходимо удалить (см. рисунок 4.11)

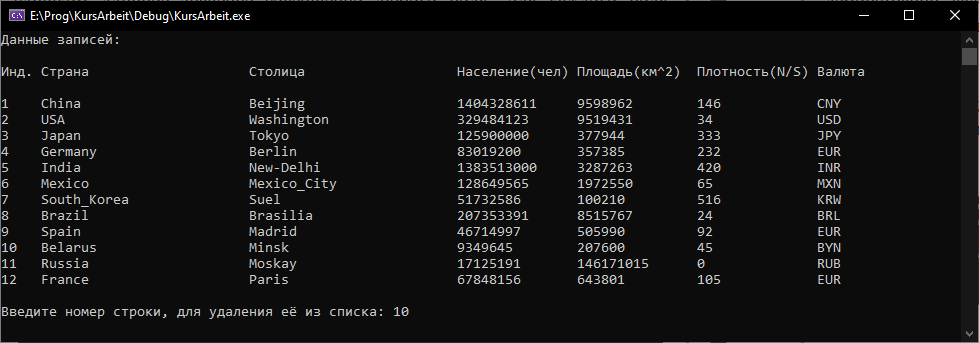


Рисунок 4.11 – удаление по индексу

После удаления выводится результат удаления в виде нового массива данных (см.рисунок 4.12). Далее выводится запрос на повторное удаление. Для того, чтобы продемонстрировать возможности другого пункта удаления, ответим “y”. При таком ответе нам снова открывается меню удаления, и мы выбираем пункт удаления по полю (см.рисунок 4.13). Там выводится меню поиска для удаления.

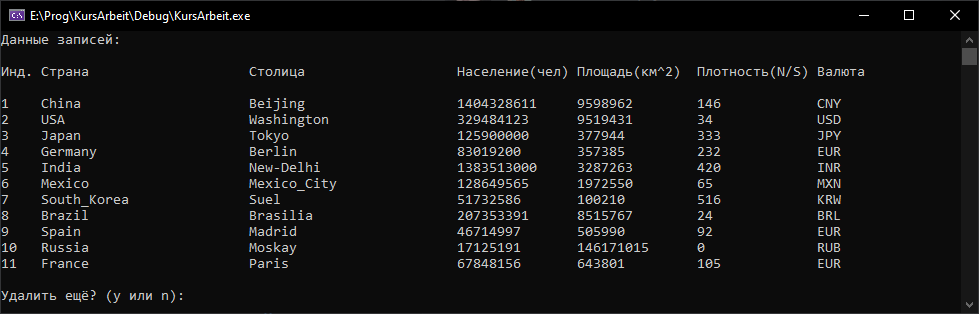


Рисунок 4.12 – массив после удаления

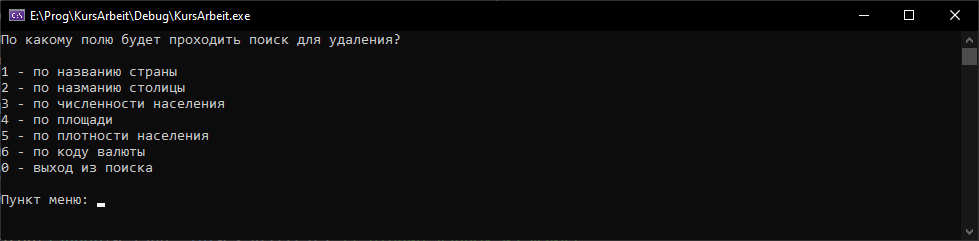


Рисунок 4.13 – меню поиска для удаления

Выберем, для примера, удаление по названию столицы. Введём «Suel». Соглашаемся на удаление этого элемента (см. рисунок 4.14). В случае отказа от удаления, с данными ничего не происходит и открывается снова меню удаления.

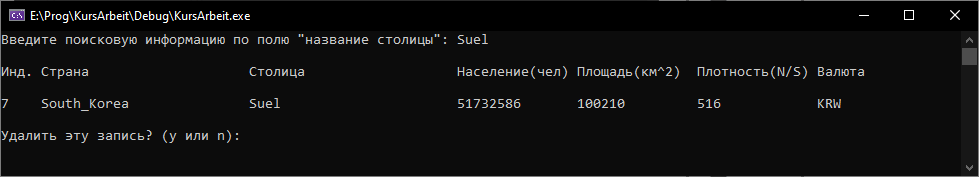


Рисунок 4.14 – удаление по полю (одно совпадение)

После этого нам снова выводится весь массив данных, без удалённой записи. Далее снова соглашаемся на повторное удаление. Далее снова заходим в пункт удаление по полю. В меню поиска для удаления активируем пункт удаление по коду валюты страны и вводим «EUR». Почему так? Всё дело в том, что код валюты у стран может совпадать, а удалить надо одну страну. Поэтому для начала выведутся все записи с этим кодом валюты, а потом необходимо указать индекс записи, которую необходимо удалить (см.рисунок 4.15).

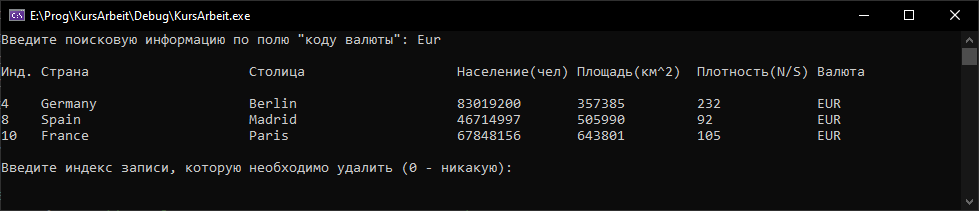


Рисунок 4.15 – удаление по полю (несколько совпадений)

Заметим, что можно ввести «0», для того, чтобы не удалять ни одну запись. Но мы введём «8». Далее нам снова выводится изменённый массив и снова появляется запрос на повторение удаления. В этот раз отвечаем «n». Далее нам выводится запрос на перезапись данных (см. рисунок 4.16). Для начала ответим “y”.

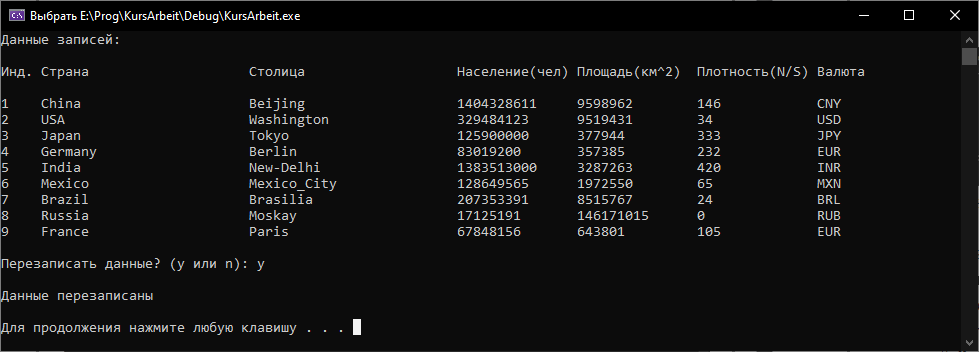


Рисунок 4.16 – перезапись после удаления

Ответив «n» можно отменить все действия в разделе удаления.

В пункте номер 5 осуществляется редактирование записи. Для начала выводятся все записи. Далее запрашивается индекс записи, которую необходимо отредактировать, затем режим редактирования: полное редактирование или частичное редактирование. Выберем полное редактирование (см. рисунок 4.17).

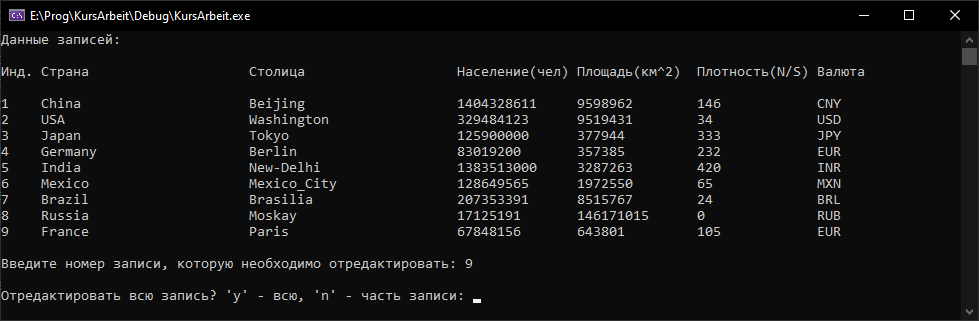


Рисунок 4.17 – редактирование записи

Далее выведется эта записи. После этого будет поочерёдно выводится запрос на изменение каждого поля записи (рисунок 4.18). Теперь опробуем частичное редактирование записи. Вспомним про недействительные данные России и изменим их (см. рисунок 4.19).

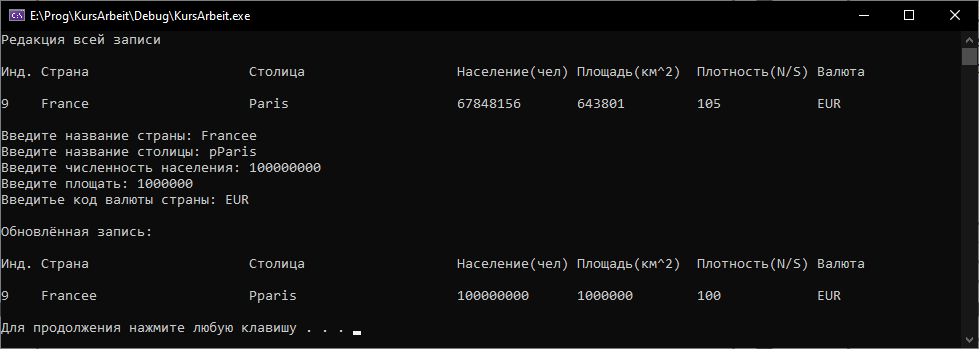


Рисунок 4.18 – полное редактирование записи

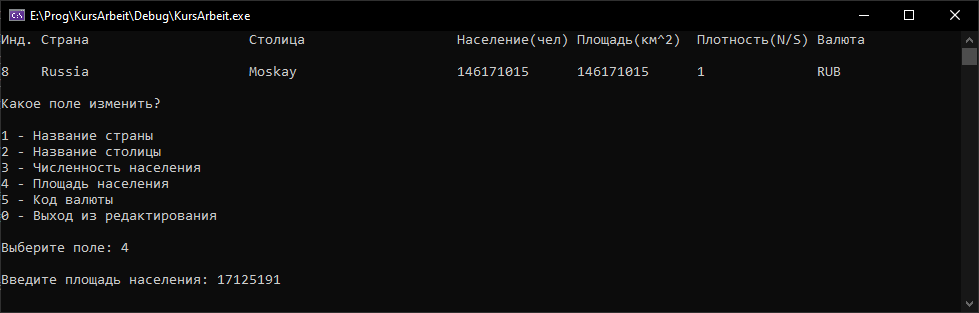


Рисунок 4.19 – частичное редактирование записи

В пункте номер 6 производится выборка по указываемому промежутку численности населения. В начале запрашивается ввод на границы промежутка. Затем происходит сама выборка с выводом элементов, которые попадаю в промежуток. А также ведётся подсчёт таких записей (см.рисунок 4.20).

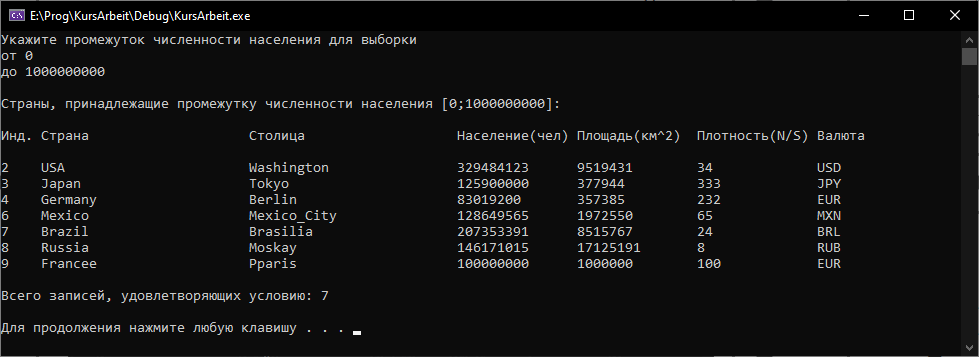


Рисунок 4.20 – выборка по численности населения

В пункте номер 7 производится поиск информации по полю. В начале выводится меню поиска (см. рисунок 4.21). Далее выбираем пункт меню поиска. Выберем пункт кода валюты страны и снова введём «EUR» (см.рисунок 4.22).

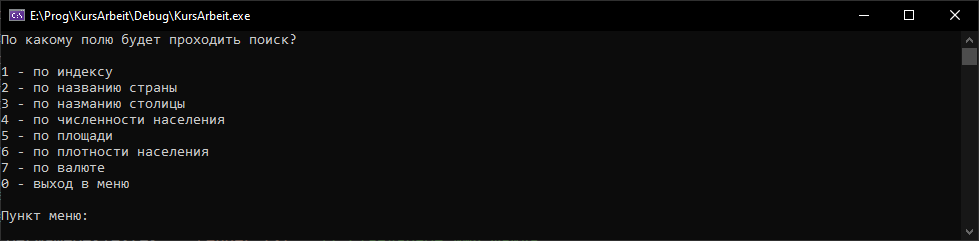


Рисунок 4.21 – меню поиска

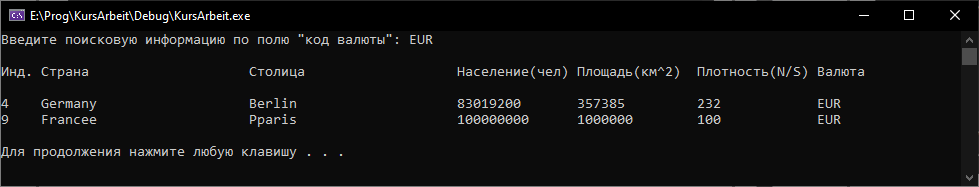


Рисунок 4.22 – результат поиска

В пункт номер 8 производится сохранений всех изменений в текстовый файл, где и объявляется об успешном сохранении (см.рисунок 4.23)

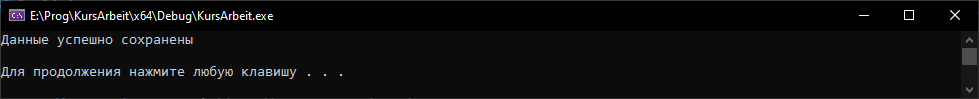


Рисунок 4.23 – сохранение данных в текстовый файл

Пункт 0 – завершение программы.

Оповещение о некорректном вводе (см.рисунок 4.24)

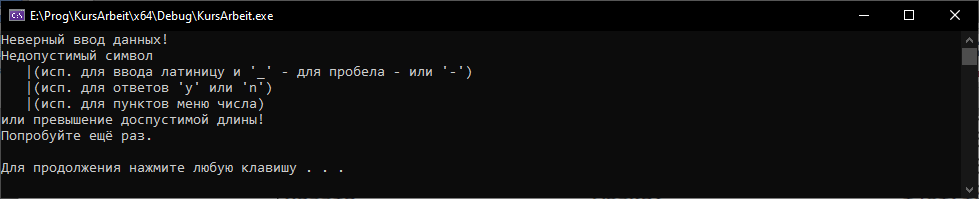


Рисунок 4.24 – оповещение о некорректном вводе

Рассмотрим частный случай, если открытый файл пустой. В том случае необходимо ввести первую строку (см.рисунок 4.25).

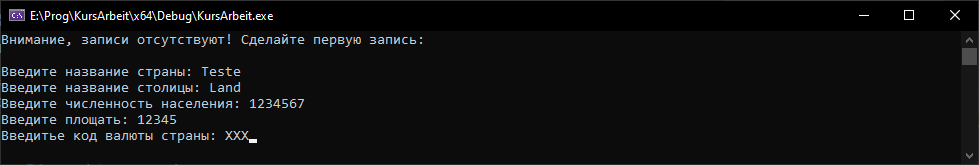


Рисунок 4.25 – ввод первой записи

Теперь рассмотрим всё то, что содержится в текстовом (см.рисунок 4.26) и бинарном (см.рисунок 4.27) файлах после всех изменений, которые были продемонстрированы выше.

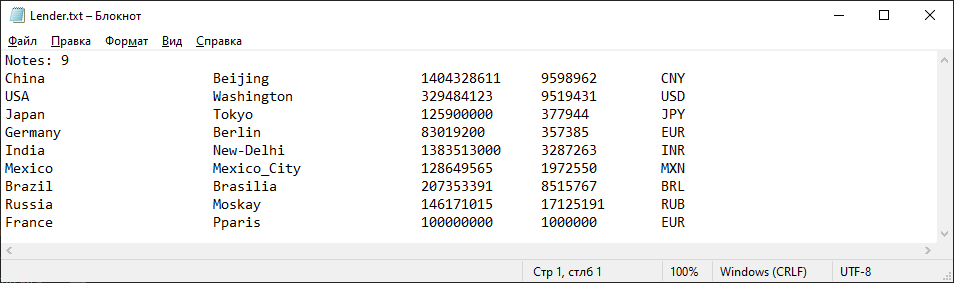


Рисунок 4.26 – содержание текстового файла после изменений

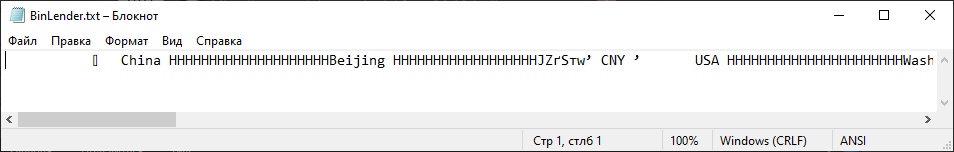


Рисунок 4.27 – содержание бинарного файла после изменений

**ЗАКЛЮЧЕНИЕ**

Результатом является работающая программа, содержащая в себе базу данных «страны мира». 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. С/С++ 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 С++ [Электронный ресурс]. – Режим доступа: <https://vk.com/doc175602161_570916906?hash=283d1eb040622>

[11b2f&dl=95eda26bfa5d45e1c8](https://vk.com/doc175602161_570916906?hash=283d1eb04062211b2f&dl=95eda26bfa5d45e1c8) – Дата доступа: 05.05.2021.

3. С/С++ 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. ГОСТ 19.504 – 79. ЕСПД. Руководство программиста. Требования к содержанию и оформлению.

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