МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УО «ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра технологий программирования

КУРСОВАЯ РАБОТА

по дисциплине «Основы алгоритмизации и программирования»

по теме: «База репозиториев сервиса GitHub»

Выполнил: студент гр. 16-ИТ-3 Яблонский А.С.

Руководитель: ассистент кафедры ТП Магеров В.В.

Дата защиты:\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Полоцк, 2016 г.

**СОДЕРЖАНИЕ**

Введение

1. Анализ задания и постановка задач
2. Теоретические сведения
3. Проектирование программы
4. Реализация программы
5. Тестирование программы

Заключение

Список литературы

Приложение А

Приложение Б

Приложение В

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

Яблонский А.С 1610574

Разраб.

Яблонский А.С

Провер.

Магеров В.В

Реценз.

Н. Контр.

Утверд.

.

База репозиториев сервиса GitHub

Лит.

Листов

23

УО «ПГУ» 16-ИТ-3

**Введение**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

Курсовая работа

GitHub — крупнейшийвеб-сервис для хостинга IT-проектов и их совместной разработки. Основан на системе контроля версий Git и разработан на Ruby on Rails и Erlang компанией GitHub.

Создатели сайта называют GitHub «социальной сетью для разработчиков». Кроме размещения кода, участники могут общаться, комментировать правки друг друга, а также следить за новостями знакомых. С помощью широких возможностей Git программисты могут объединять свои репозитории — GitHub предлагает удобный интерфейс для этого и может отображать вклад каждого участника в виде дерева.

Для проектов есть личные страницы, небольшие Вики и система отслеживания ошибок. Прямо на сайте можно просмотреть файлы проектов с подсветкой синтаксиса для большинства языков программирования.

В конце 2008 года GitHub получил награду как «Лучший стартап-дебют». Первый частный репозиторий был создан 12 января 2008. К концу 2011 года в проекте уже было зарегистрировано более миллиона пользователей и более двух миллионов репозиториев. По состоянию на март 2017 года на сайте существовало более 58 миллионов репозиториев.

GitHub наиболее популярен среди Ruby-разработчиков. Кроме того, многие крупные IT-компании размещают свои официальные репозитории на этом сервисе: Google, Facebook, Twitter, Microsoft, LineageOS и Yahoo.

Данная работа посвящена изучению языку С/С++ и созданию базы данных сервиса GitHub. База данных является неотъемлемой частью каждой программы. Будь это сложная программа банковского учета или простая программа, содержащая имена студентов, которые учатся в университете.

Цель курсовой работы:

* приобрести навыки формальной математической постановки задач;
* алгоритмизация задач и программирование, отладка и выполнение на ЭВМ конкретных задач с использованием современных методов программирования;
* научиться характеризовать исходные данные;
* уметь анализировать решение задач;
* овладеть теоретическими знаниями основ программирования и алгоритмизации.

1. **Анализ задания и постановка задач**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

4

Курсовая работа

Цель курсовой работы является создание программы, которая содержала бы базу данных популярных репозиториев сервиса GitHub. Основная функция программы – работа с одно файловой базой данных. Для описания каждого репозитория используются следующие поля (примеры содержания некоторых полей приведены в скобках):

1. Название репозитория - поле представляющее собой строку до 36 символов, например: «howmanypeoplearearound»
2. Язык программирования использующийся в репозиторие – поле представляющее собой строку до 36, например: «Python»
3. Имя (ник-нейм) автор репозитория – поле представляющее собой строку до 36, например: «schollz»
4. Рейтинг репозитория – вещественное число, принимающее значения в диапазоне от 0.0 до 5.0 включительно с двумя знаками после запятой, например: «4.85»
5. Дата создания репозитория в формате дд.мм.гггг, например: «15.05.2000»
6. Дата последнего изменения репозитория в формате дд.мм.гггг, например: «15.05.2017»
7. Количество совершенных коммитов – целочисленное значение от 0 до 999 включительно, например: «26»
8. Количество найденных ошибок в коде репозитория – целочисленное значение от 0 до 999 включительно, например: «93»
9. Количество исправленных ошибок репозитория – целочисленное значение от 0 до 999 включительно, например: «8»
10. Число веток, находящихся в репозитории – целочисленное значение от 0 до 99 включительно, например: «3»

Программа должна иметь дружественный интерфейс и проводить проверку на правильность ввода данных. Программа должна осуществлять:

1. Запись и загрузку файла базы данных (текстовый файл).
2. Добавление новых записей, редактирование и удаление старых.
3. Сортировать записи по любому из полей базы данных.
4. Фильтровать записи по любому из полей базы данных.
5. Перемещать запись из старой позиции в новую.
6. Отображать весь список записей.
7. Отображать подробную информацию по одной записи.
8. **Теоретическая часть**

В информатике, список – это абстрактный тип данных, представляющий собой упорядоченный набор значений, в котором некоторое значение может встречаться более одного раза. Экземпляр списка является компьютерной реализацией математического понятия конечной последовательности. Экземпляры значений, находящихся в списке, называются элементами списка (англ. Item); если значение встречается несколько раз, каждое вхождение считается отдельным элементом.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

5

Курсовая работа

Термином список также называется несколько конкретных структур данных, применяющихся при реализации абстрактных списков, особенно связных списков. Одним из представителей связных списком является Стек. Стек – (англ. stack – стопка) – абстрактный тип данных, представляющий собой список элементов, организованных по принципу LIFO (англ. last in – first out, «последним пришёл – первым вышел»). Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно снять верхнюю.

Зачастую стек реализуется в виде однонаправленного списка (каждый элемент в списке содержит помимо хранимой информации в стеке указатель на следующий элемент стека). Но также часто стек располагается в одномерном массиве с упорядоченными адресами. Такая организация стека удобна, если элемент информации занимает в памяти фиксированное количество слов, например, 1 слово. При этом отпадает необходимость хранения в элементе стека явного указателя на следующий элемент стека, что экономит память.

Возможны три операции со стеком: добавление элемента, удаление элемента и чтение головного элемента:

1. При добавлении элемента создается новый элемент, указывающий на элемент, бывший до этого головой. Новый элемент теперь становится головным.
2. При удалении элемента убирается первый, а головным становится тот, на который был указатель у этого объекта (следующий элемент). При этом значение убранного элемента возвращается.
3. При чтении элемента извлекается верхний элемент, головным элементом становится следующий за ним.

К достоинствам любого связного списка можно отнести:

Изм.

Лист

№ докум.

Подпись

Дата

Лист

6

Курсовая работа

* размер ограничен только объёмом памяти компьютера;
* быстрое динамическое добавление и удаление элементов.

Как недостатки нужно отметить следующий факторы:

* сложность прямого доступа к элементу, а именно определения физического адреса по его индексу (порядковому номеру) в списке;
* на поля-указатели (указатели на следующий и предыдущий элемент) расходуется дополнительная память;
* некоторые операции со списками медленнее, чем с массивами, так как к произвольному элементу списка можно обратиться, только пройдя все предшествующие ему элементы.

**Указатель –** этопеременная, значением которой является адрес, по которому располагаются данные.

**Типизированный указатель** – указатель, содержащий адрес данных определенного типа (системного или пользовательского).

**Не типизированный указатель** – указатель, содержащий адрес данных неопределенного типа (просто адрес).

Классификация по области доступа определяется методом адресации принятой для семейства процессоров x86: адрес состоит из двух элементов: сегмент и смещение.

**Адрес** – это номер ячейки памяти, в которой или с которой располагаются данные.

**Динамическая память** – это область (блок) памяти выделенный для нужд программы в процессе работы программы (а не заранее).

Основными двумя действиями над динамической памятью являются: выделение и освобождение. В языке С функции для осуществления этих действий описаны в библиотеке **stdlib.h**.

**Структура** – это сложный тип данных представляющий собой упорядоченное в памяти множество элементов различного типа. Каждый элемент в структуре имеет свое имя и называется полем. Размер структуры определяется суммой размеров всех элементов.

**Файл** – именованная область данных на каком-либо носителе информации (жесткий диск, дискета, компакт-диск и т.д. и т.п.).

1. **Проектирование программы**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

7

Курсовая работа

При реализации базы данных GitHub репозиториев будет использован стек на основе однонаправленного списка.

Для удобной работы с функциями проекта и быстрой масштабируемости кода выделим основные модули и их функции для последующей реализации.

1. **Модуль работы с меню** 
   1. Функции для отрисовки меню
   2. Функции перемещения по элементам меню
2. **Модуль работы с GitRep структурой** 
   1. Функции ввода/вывода новой структуры
   2. Функцию сравнения двух структур
   3. Функцию подробной печати структуры
   4. Функцию сохранения структуры в файловый поток
3. **Модуль помощи ввода/вывода –** будет использован для организации проверок ввода данных, а также для отрисовки дружелюбного пользователю интерфейса
   1. Функции установки/получение положения курсора в консоли
   2. Функцию цветного вывода переменных
   3. Функцию изменения размера окна консоли
   4. Функцию удаления символов/строки отображенных в консоли
4. **Модуль работы со списком**
   1. Функции добавления, чтения и удаления элемента по индексу
   2. Функции сортировки и фильтрации
   3. Функции печати данных в табличном формате
5. **Модуль работы с файлами**
   1. Функции сохранения списка в файл
   2. Функции чтения списка из файла
6. **Файл кодов** - дополнительные заголовочные файлы для хранения константных переменных со значениями кодов, возвращаемых функциями.

После запуска программы, пользователю отображается главное меню состоящее из 4 пунктов:

1. **Database** – меню работы с файлом базы данных. В себе содержит следующие пункты:
   1. Load – загрузка базы данных из файла
   2. Save – сохранение базы данных в файл, если она не пуста
2. **Item** – меню работы элементами списка:
   1. Add – добавление нового элемента в указанную позицию
   2. Show – подробный просмотр элемента по указанной позиции
   3. Edit – редактирование элемента по указанной позиции
   4. Remove – удаление элемента с позиции

Изм.

Лист

№ докум.

Подпись

Дата

Лист

8

Курсовая работа

1. **List** – меню обработки списка:
   1. Sort – запуск функции сортировки по указанному полю и в выбранном направлении
   2. Filter – фильтрация списка по указанному полю и введенному значению
   3. Search – поиск элемента списка по указанному полю и введенному значению
   4. Clean – полная очистка всего списка
2. **Exit** – заверение работы программы

\* Каждое подменю имеет пункт возврата в главное меню.

\* Критические операции, такие как полная очистка списка, выход без сохранения базы данных или загрузка новой базы данных без сохранения прежней, имеют подтверждение выполнения

При проектировании расположения элементов интерфейса использовался редактор изображений – Paint.



**Рисунок 1** – шаблон интерфейса с отображением таблицы данных и главным меню

1. **Реализация программы**

Для реализации базы данных содержащие репозитории GitHub была создана следующая структура:

Изм.

Лист

№ докум.

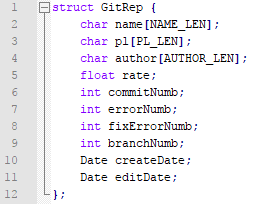
Подпись

Дата

Лист

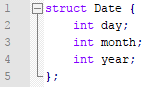
9

Курсовая работа



**Листинг 1.1** – структура элемента базы данных (NAME\_LEN, PL\_LEN, AUTHOR\_LEN – константные целочисленные переменные, определяющие размер полей)

Размер текстовых полей структуры GitRep - name, pl, author определен в целочисленных константах переменных - NAME\_LEN, PL\_LEN, AUTHOR\_LEN соответственно, для возможности быстрого изменения их размера. Дата создания и редактирования репозитория хранится в структуре Date (Листинг 1.2).

  
**Листинг 1.2** – структура для хранения даты

**void** setWindowSize(**int** width, **int** height); // установка размера окна консоли  
Color getConsoleTextColor(); // получение цвета текста консоли  
**void** setConsoleTextColor(Color color); // установка цвета текста консоли  
COORD getCursorPosition(); // получение позиции курсора консоли  
**void** setCursorTo(**int** x, **int** y); // установка позиции курсора консоли  
**int** printf(Color color, **const char** \*format, ...); // переопределенная функция вывода данных с выбором цвета  
**void** removeC(**int** n); // удаление n-числа символов  
**void** removeC(**int** n, **char** c); // замена n-числа символов на с-символ  
**void** clearLine(); // очистка строки консоли  
**void** beep(); // проигрывание системного звука

Изм.

Лист

№ докум.

Подпись

Дата

Лист

10

Курсовая работа

Листинг 3 – список публичных функций модуля **console\_io\_lib.cpp**

**int** saveList(StackList \*list); // функция сохранения списка в файл  
**int** loadList(StackList \*list); // функция чтения списка из файла

Листинг 4 – список публичных функций модуля **database.cpp**

**int** isDateCorrect(Date date); // функция проверки даты на корректность  
**int** compareDate(Date dateA, Date dateB); // функция сравнения дат  
**char** \*dateToString(**char** \*buffer, Date date); // функция перевода даты в строку

Листинг 5 – список публичных функций модуля **date.cpp**

**void** scanGitRep(GitRep \*gitRep); // ввод новой структуры GitRep  
**void** printGitRep(GitRep gitRep); // подробный вывода структуры GitRep в консоль  
**void** saveToFile(std::ofstream &dbFile, GitRep gitRep); // сохранение структуры GitRep в файл

**int** compareGitReps(GitRep \*a, GitRep \*b, GitRepField field, **bool** fullComp); // сравление двух структур GitRep по полю

**void** sortGitRepArrayList(GitRep \*\*gitRepList, **int** gitRepNumb, SortDirection dir, GitRepField field); // Сортировка массива по заданному полю и направлению

Листинг 6 – список публичных функций модуля **git\_rep\_io**

**.cpp**

**int** sizeOf(StackList \*list); // получение количсетва элементов списка  
**int** addItem(StackList \*stackList, GitRep data, **int** index); // добавление элемента в позицию  
**int** removeItem(StackList \*stackList, **int** index); // удаление элемента из списка по позиции  
**int** getItem(StackList \*stackList, StackListItem \*\*item, **int** index); // извлечение элемента из списка по позиции  
**int** readItem(StackList \*stackList, StackListItem \*\*item, **int** index); // чтение элемента из списка по позиции  
StackList \*newList(); // создание пустого списка  
**void** freeList(StackList \*stackList); // очистка списка и высвыбождение памяти  
**int** printList(StackList \*stackList); // печать всех элементов списка в табличной формате

**void** cleanList(StackList \*stackList); // удаление всех элементов списка  
**bool** isEmpty(StackList \*stackList); // проверка списка на пустоту

Изм.

Лист

№ докум.

Подпись

Дата

Лист

11

Курсовая работа

**int** listToArray(StackList \*list, GitRep \*\*array, **int** \*size); // преобразование списка в массив  
**int** arrayToList(StackList \*list, GitRep \*array, **int** size); // преобразование массива в список

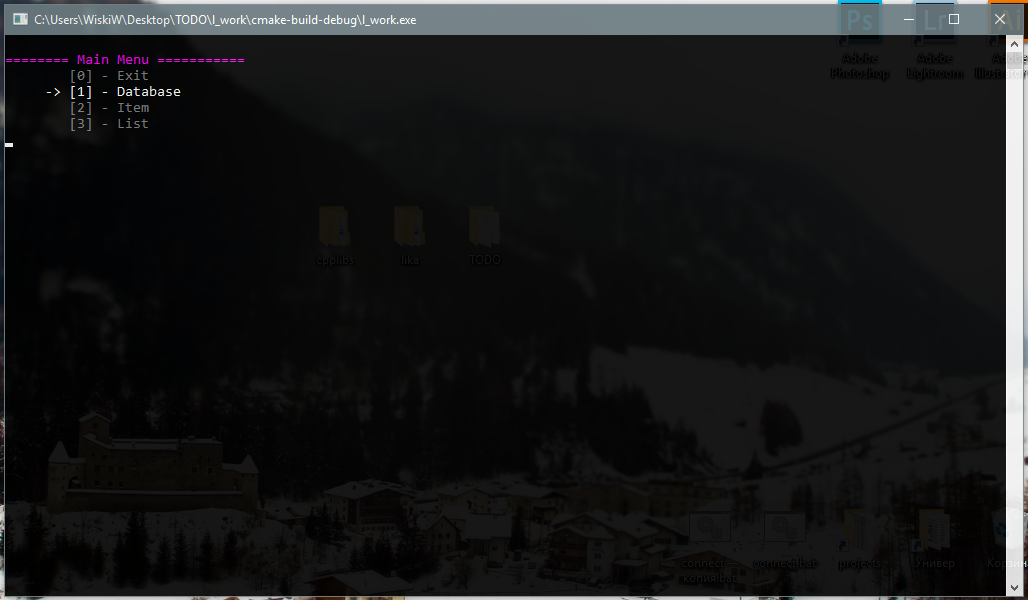
Листинг 7 – список публичных функций модуля **GitRepStackList.cpp**

**int** printMainMenu(); // вывод и обработка нажатий главного меню  
**int** printDatabaseMenu(); // вывод и обработка нажатий меню работы с файлом базы данных  
**int** confirmMenu(); // вывод и обработка нажатий уточняющего меню  
**int** printItemMenu(); // вывод и обработка нажатий меню работы с элементом  
**int** printListMenu(); // вывод и обработка нажатий меню работы со списком  
**int** printFieldMenu(); // вывод и обработка нажатий меню выбора поля структуры  
**int** printDirMenu(); // вывод и обработка нажатий меню выбора направления сортировки

Листинг 8 – список публичных функций модуля **menu.cpp**

1. **Тестирование программы**

В результате работы была написана программа, осуществляющая редактирование и обработку базы данных репозиториев GitHub. Основные экраны программы (Рисунок 2, Рисунок 2.1, Рисунок 2.2)



Изм.

Лист

№ докум.

Подпись

Дата

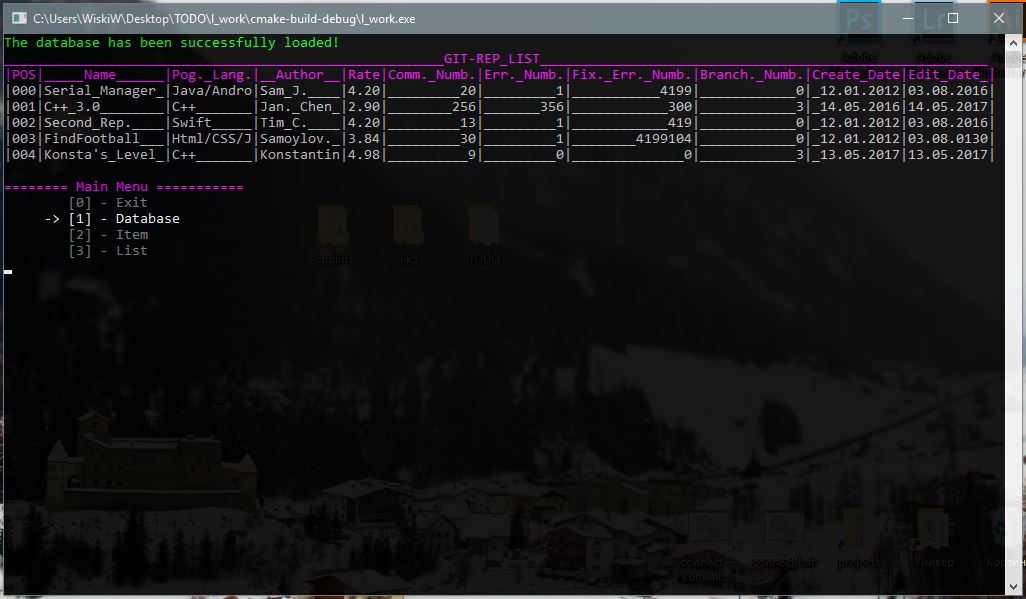
Лист

12

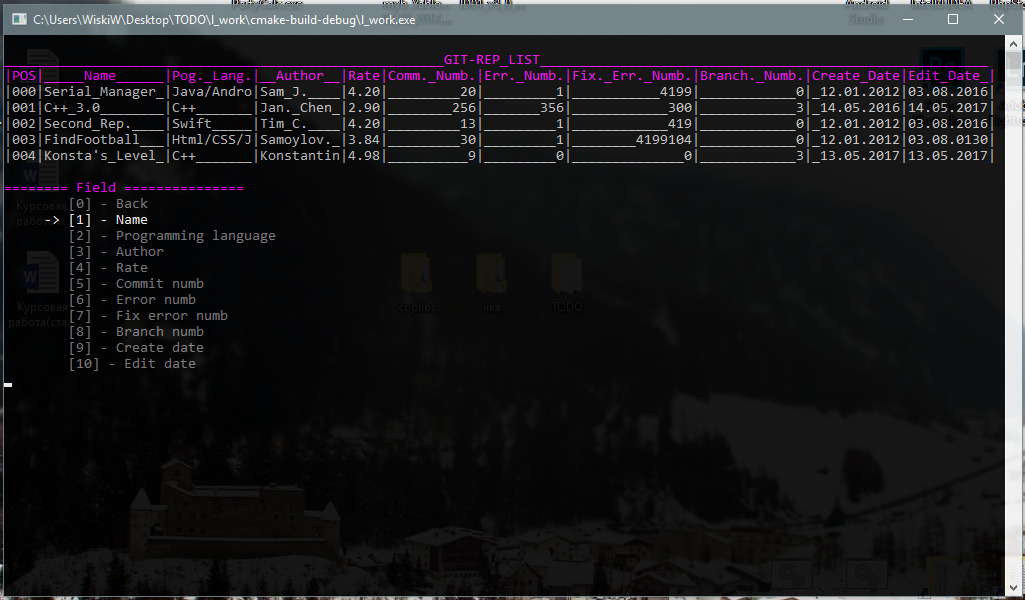
Курсовая работа

**Рисунок 2** – главное меню после запуска программы

Пользователь имеет возможность передвигаться по пунктам меню, как при помощи цифр от 0 до 9, так и при помощи стрелок UP, DOWN, LEFT, RIGHT клавиатуры. Выбор пункта меню осуществляется нажатием клавиши ENTER.



**Рисунок 2.1** – главное меню после успешной загрузки списка из базы данных



Изм.

Лист

№ докум.

Подпись

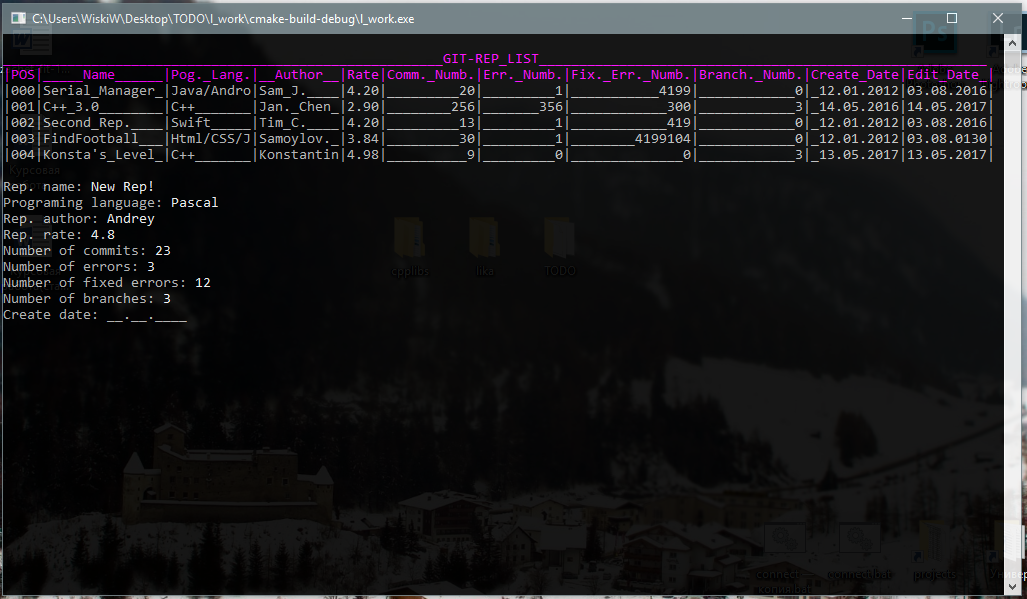
Дата

Лист

13

Курсовая работа

**Рисунок 2.2**– меню выбора поля сортировки



**Рисунок 2.3**– добавление нового элемента

На данном скриншоте (Рисунок 2.3) можно наблюдать функцию «помощь ввода», в частности помощь ввода даты создания. Данная функция ввода предоставляет шаблон, соответствующий форме заращиваемым данных. Проверка ввода осуществляется сразу после нажатия пользователем

клавиши клавиатуры, что предоставляет больше возможностей для вывода предупреждающих сообщений. Аналогичные проверки ввода присутствуют при заполнении каждого поля, что прощает задачу пользователю.

Изм.

Лист

№ докум.

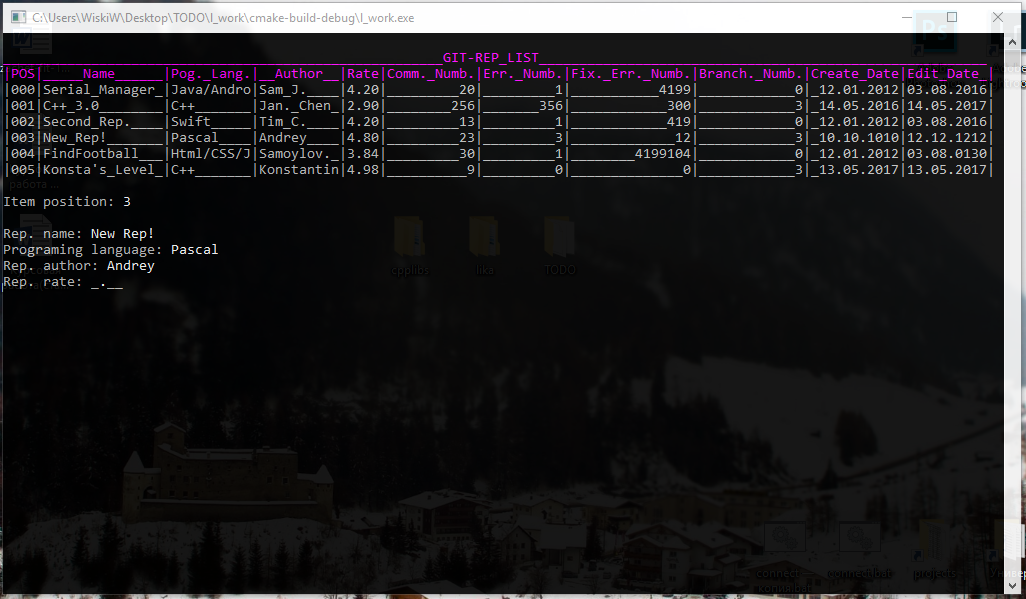
Подпись

Дата

Лист

14

Курсовая работа



Изм.

Лист

№ докум.

Подпись

Дата

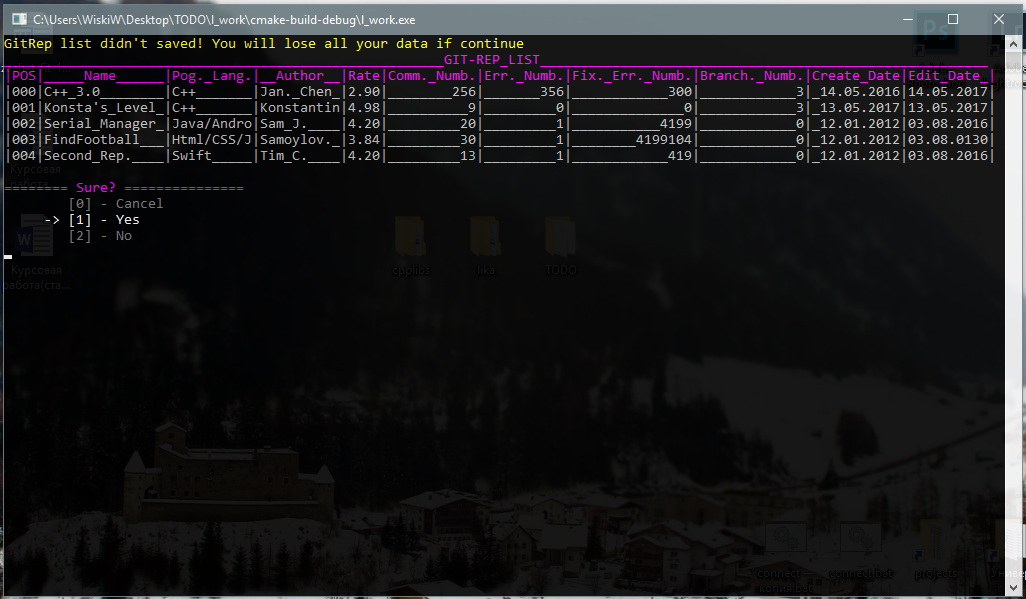
Лист

15

Курсовая работа

**Рисунок 2.3** – редактирование элемента

Интерфейс программы проектировался с расчетом на максимальную функциональность, информативность и высокую скорость работы. В следствии решено было не оснащать его дополнительными укрощающими элементами.

При внесении изменений в загруженную базу данных и попытке выхода из программы, пользователь получает предупреждение, о возможной потере данных (Рисунок 2.4).

**Рисунок 2.4** – запрос подтверждения действия

Аналогичное предупреждение пользователь увидит при попытке загрузить новый файл базы данных без сохранения открытого списка.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

16

Курсовая работа

1. **Заключение**

В данной курсовой работе была поставлена цель, использовав полученные знания языка программирования Cи, реализовать работоспособное программное обеспечение, способное хранить в себе информацию о репозиториях сервиса GitHub, редактировать существующие записи, а также корректно обрабатывать новые данные.

Удалось реализовать ПО, которое способно управлять базой данных: добавлять, редактировать, удалять, искать, сортировать, фильтровать данные.

Данное по способно сохранять данные в текстовый файл в виде псевдографической таблицы. Реализована корректная обработка потенциальных ошибок с информированием о них пользователя.

Программа имеет дружественный интерфейс и проводит проверку на правильность ввода данных.

**Программа позволяет осуществить:**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

17

Курсовая работа

* Запись и загрузку файла базы данных (текстовый файл).
* Добавление новых записей, редактирование и удаление старых.
* Сортировать записи по любому из полей базы данных.
* Фильтрация записи по любому из полей базы данных.
* Поиск записи по любому из полей базы данных.
* Перемещать запись из старой позиции в новую.
* Отображать весь список записей в консоль.
* Форматированный вывод всех записей в консоль
* Отображать подробную информацию для каждой записи.

Поставленные задачи были полностью достигнуты, что свидетельствует о полном усвоении материала и приобретении навыков алгоритмизации задач и программирования.

**СПИСОК ЛИТЕРАТУРЫ**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

17

Курсовая работа

1. Брюс Эккель «Думаем на С++»
2. Скотт Мэйерс «Эффективный С++»
3. Конспект лекций по дисциплине «Языки программирования».

**Приложение А**

Изм.

Лист

№ докум.

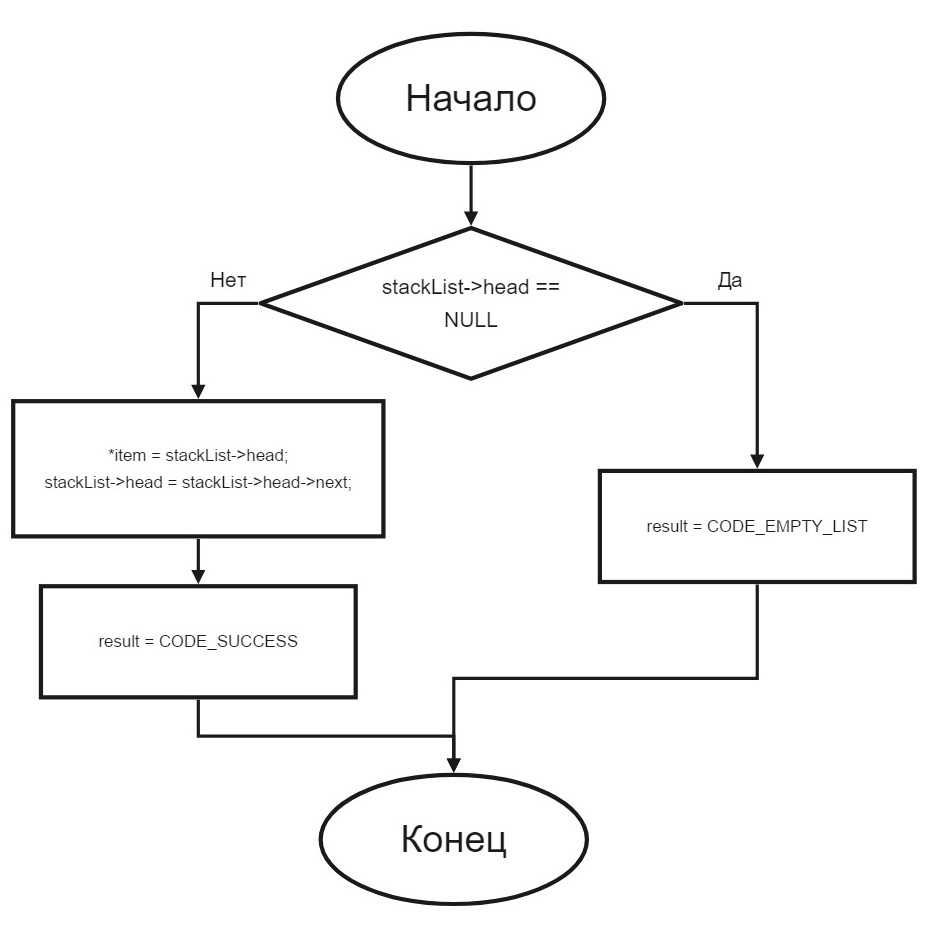
Подпись

Дата

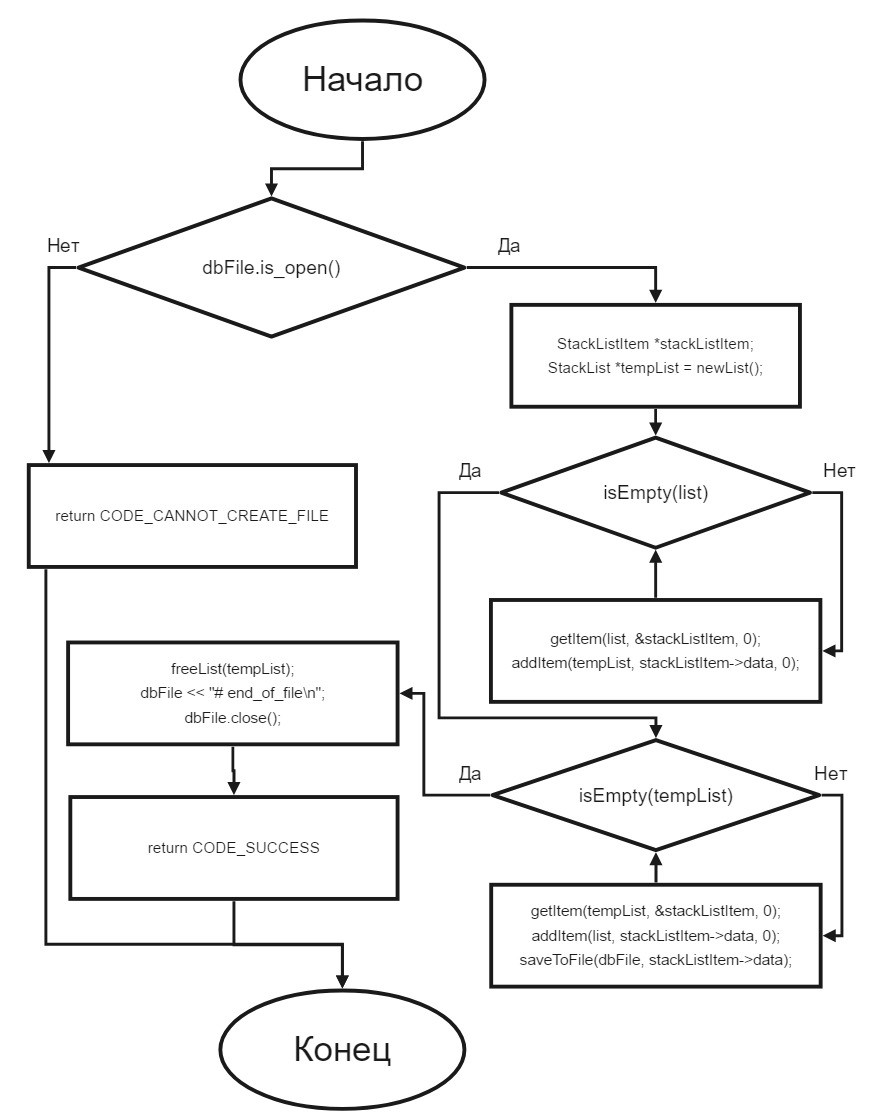
Лист

18

Курсовая работа



**Рисунок A.1** – блок схема алгоритма функции getItem

****

Изм.

Лист

№ докум.

Подпись

Дата

Лист

19

Курсовая работа

**Рисунок A.2** – блок схема алгоритма функции saveList

**Приложение Б**

Изм.

Лист

№ докум.

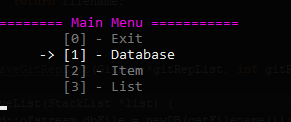
Подпись

Дата

Лист

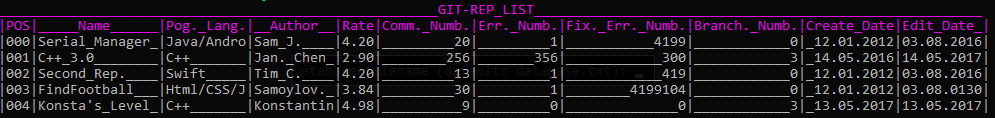
20

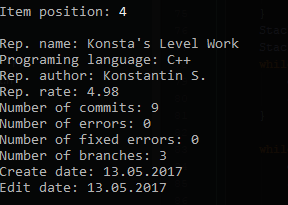
Курсовая работа



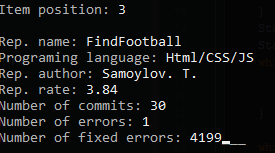
**Рисунок Б.1** – Главное меню

https://gyazo.com/b45401ee7bd6a27e9693c7250e0bd1b7.png  
**Рисунок Б.2** – Поле ввода названия файла базы данных

  
**Рисунок Б.3** – Табличное отображение базы данных



**Рисунок Б.4** – Подробное отображение элемента базы данных



Изм.

Лист

№ докум.

Подпись

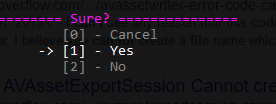
Дата

Лист

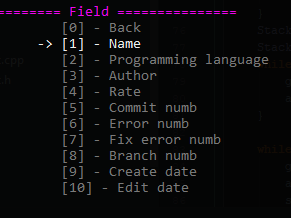
21

Курсовая работа

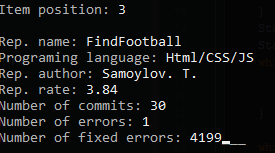
**Рисунок Б.5** – Редактирование элемента базы данных



**Рисунок Б.6** – Меню подтверждения действия



**Рисунок Б.7** – Меню выбора поля сортировки, поиска или фильтрации



Изм.

Лист

№ докум.

Подпись

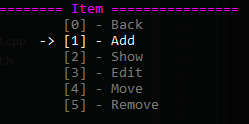
Дата

Лист

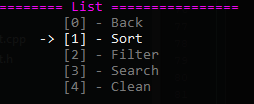
22

Курсовая работа

**Рисунок Б.8** – Заполнение полей нового элемента



**Рисунок Б.9** – Меню действий с элементами базы данных



**Рисунок Б.10** – Меню действий с списком элементов базы данных

**Приложение В**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

23

Курсовая работа

Листинг приложения находиться на диске в папке «source\_code».