Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему:

«АНАЛИЗ АЛГОРИТМОВ СЖАТИЯ ФАЙЛОВ»

БГУИР КП 1-40 01 01  012  ПЗ

Студент: гр. 851002 Ковалевский М.Ю.

Руководитель: асс. Фадеева Е.Е.

Минск 2019

**СОДЕРЖАНИЕ**

СОДЕРЖАНИЕ ..……………………………………………………….….…….…4

ВВЕДЕНИЕ ...…………………………………………………………….………...5

1. АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ ……………………………………………………………………..…..…………….7

1.1 Актуальность сжатия …………………………………………….…...……….7

1.2 Программы для сжатия файлов......................................................................... 8

1.3 Описание популярных алгоритмов сжатия …….…………………………... 11

1.4. Формирование требований к проектируемому программному средству…14

1.5 Входные данные ................................................................................................14

2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРА-  
БОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ...………………………….…..15

2.1 Разработка используемых данных ……………………………………..….....15

2.2 Анализ требований к программному средству и разработка функциональных требований …………………………………………………………………………… 16

2.3 Разработка алгоритма программного средства …………………………….. 16

3. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА ……………………. 17

3.1 Общая схема программы………………………………………………………17

3.2 Алгоритм сжатия RLE…………………………………………………………18

3.3 Алгоритм сжатия Хаффмана…………………………………………….…….18

3.4 Алгоритм сжатия LZ77………………………………………………….……..19

3.5 Алгоритм сжатия Deflate………………………………………………….……19

3.6 Алгоритм декомпрессии файла, сжатого RLE…………………………….….19

3.7 Алгоритм декомпрессии файла, сжатого алгоритмом Хаффмана ……….…20

3.8 Алгоритм декомпрессии файла, сжатого LZ77…………………………….…20

3.9 Алгоритм декомпрессии файла, сжатого Deflate……………………………..21

3.10 Алгоритм построения графиков………………………………………...…....21

4. СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА …………………………...…....22

4.1 Описание модулей ……………………………………………………………...22

4.2 Список все процедур, их параметры и назначение…………………………...27

5. ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ ….……………………………………………………. 31

6. РУКОВОДСТВО ПО УСТАНОВКЕ И ИСПОЛЬЗОВАНИЮ …………….… 34

6.1 Основные требования для запуска данного программного средства…………..34

6.2 Руководство по установке .……………………………………………………..34

6.3 Пример использования программы …………………………............................34

ЗАКЛЮЧЕНИЕ …………………………………………………………...………...36

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ ……………………........…………37

ПРИЛОЖЕНИЯ .………………………………………………………….…………38

**ВВЕДЕНИЕ**

Мы даже не задумываемся, какой объем данных генерируем в течение дня, и где все эти данные хранятся. Благодаря интернету, Google в день обрабатывает больше данных, чем было написано во всех литературных работах на всех языках до появления интернета, и это при том, что к паутине подключена далеко не вся планета.

Первый жесткий диск на 5 МБ появился 60 лет назад. Он [весил](http://gagadget.com/15739-evolyutsiya-kompyuternyih-nositelej-informatsii/) около тонны и по размеру был сравним с крупным современным холодильником. Внутри массивного корпуса находилось 50 дисков диаметром 60 сантиметров или примерно 23 дюйма. Сегодня на таком пространстве помещаются две серверные стойки, а маленькое устройство в кармане может хранить несколько сотен гигабайт информации.

Сегодня люди генерируют огромное количество данных. Если учесть, что эти данные затем копируются в облако и копируются на другие носители, это только увеличивает объем памяти, необходимый для их хранения. Такие решения и технологии, как автономные автомобили, интеллектуальные фабрики, интернет вещей (IoT), блокчейн и домашняя автоматизация, будут генерировать дополнительные потоки данных, которые необходимо хранить.

Таким образом ценность сжатия данных невозможно переоценить. Без уменьшения размера файлов, большим компаниям, как и совершенно обычным людям, потребуется значительно больше места на накопителе. А это привело бы к дополнительным тратам на оборудование и его более быстрому износу.

Сжатие данных – это алгоритмическое преобразование [данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5), производимое с целью уменьшения занимаемого ими объёма. Применяется для более рационального использования устройств [хранения](https://ru.wikipedia.org/wiki/%D0%A5%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%B8) и [передачи](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D0%B4%D0%B0%D1%87%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) данных.

Все методы сжатия данных делятся на два основных класса:

* [Сжатие без потерь](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B1%D0%B5%D0%B7_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8C)
* [Сжатие с потерями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D1%81_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8F%D0%BC%D0%B8)

При использовании сжатия без потерь возможно полное восстановление исходных данных, сжатие с потерями позволяет восстановить данные с искажениями.

Сжатие без потерь обычно используется для передачи и хранения текстовых данных, компьютерных программ, реже - для сокращения объёма [аудио](https://ru.wikipedia.org/wiki/%D0%90%D1%83%D0%B4%D0%B8%D0%BE)- и [видеоданных](https://ru.wikipedia.org/wiki/%D0%92%D0%B8%D0%B4%D0%B5%D0%BE), [цифровых фотографий](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D1%8F_%D1%84%D0%BE%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) и т. п., в случаях, когда искажения недопустимы или нежелательны. Сжатие с потерями, обладающее значительно большей, чем сжатие без потерь, эффективностью, обычно применяется для сокращения объёма аудио- и видеоданных и цифровых фотографий в тех случаях, когда такое сокращение является приоритетным, а полное соответствие исходных и восстановленных данных не требуется.

Данная записка содержит следующие разделы курсовой работы по разработке ПО для проведения операций над матрицами:

1. Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству;

2. Анализ требований к программному средству и разработка функциональных требований;

3. Проектирование программного средства;

4. Создание программного средства;

5. Тестирование, проверка работоспособности и анализ полученных результатов;

6. Руководство по установке и использованию

**1 АНАЛИЗ ПРОТОТИПОВ, ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОМУ ПРОГРАММНОМУ СРЕДСТВУ**

**1.1 Актуальности сжатия**

В наше время сжатие данных используется повсеместно. Такие программы, как архиваторы, установлены на каждом компьютере. С Windows 10 в комплекте идет алгоритм сжатия LZX, доступный через PowerShell.

Знакомство со сжатыми файлами начинается с обычных картинок. Большинство изображений хранится в таком формате, как JPEG. Файлы такого формата могут иметь расширения .jpg, .jfif, .jpe или .jpeg. Однако из них .jpg является самым популярным на всех платформах.

Каждый день ведутся разработки новых алгоритмов, позволяющих сжать файл с меньшими потерями и с меньшим итоговым размером.

Сжатие данных получило широкое распространение вместе с интернетом и после изобретения алгоритмов Лемпелем и Зивом (алгоритмы LZ).

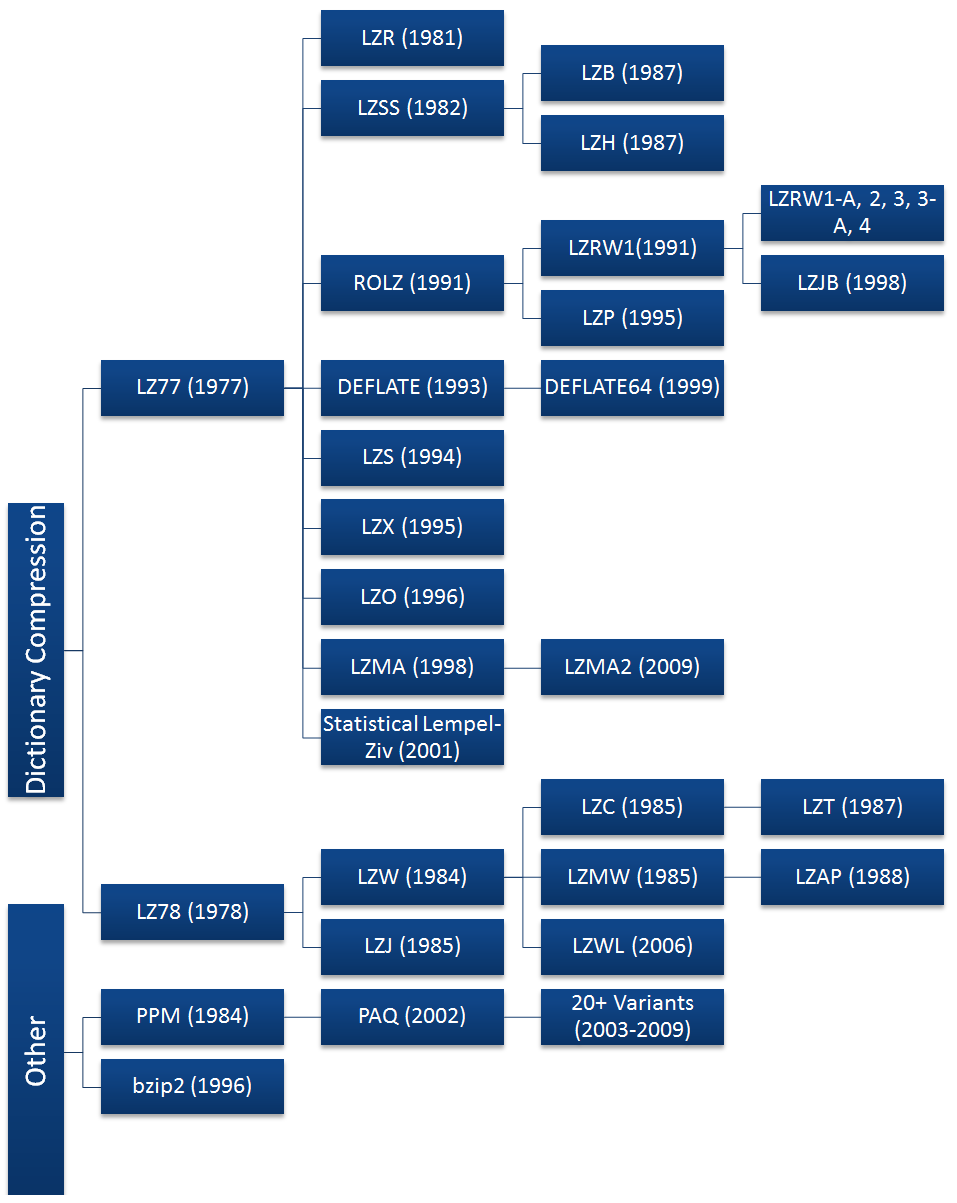


Рисунок 1.1 – Иерархия алгоритмов

**1.2 Программы для сжатия файлов**

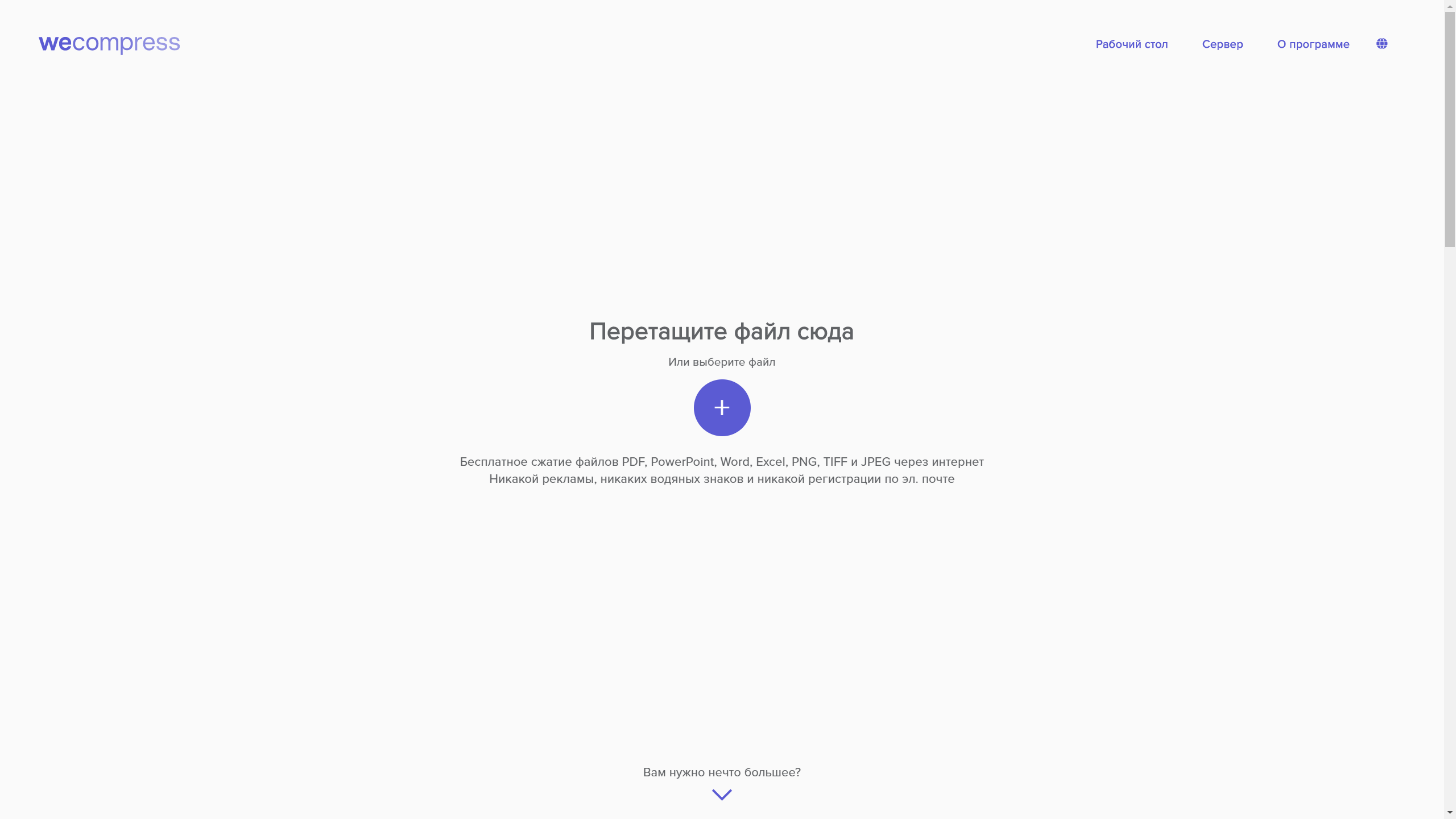
1.2.1 Онлайн сервис «Wecompress»[1].

Плюсы:

* Отличный сайт, позволяющий быстро сжать PDF, PNG, TIFF и JPEG. Приятный и интуитивно понятный пользовательский интерфейс. Содержит вкладку с инструкцией по эксплуатации для пользователя. Переведен на все основные языки мира.

Минусы:

* Онлайн программа, для доступа к ней необходимо подключение к сети, что не всегда является возможным. Сжатие происходит путем уменьшения качества файла без возможности восстановления.

Интерфейс изображен на рисунке 1.2 и 1.3.

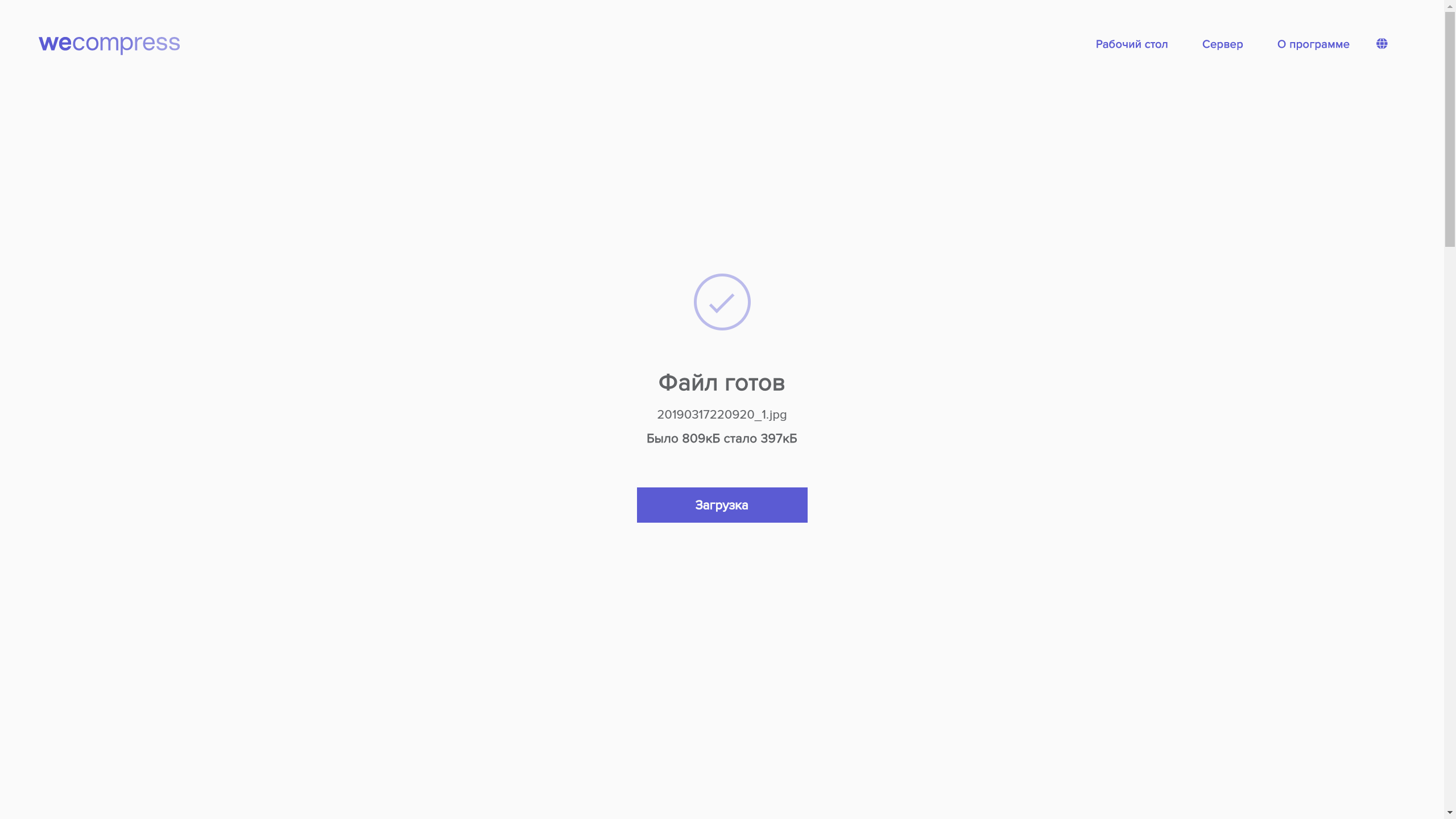
Рисунок 1.1 – Скриншот сайта “[www.wecompress.com](https://www.wecompress.com/ru/)” до сжатия файла

Рисунок 1.2 – Скриншот сайта “[www.wecompress.com](https://www.wecompress.com/ru/)” после сжатия файла

1.2.2 Архиватор «WinRAR»[2].

Плюсы:

* Разработчики поддерживают продукт и регулярно выпускают обновления;
* Создание архивов [RAR](https://ru.wikipedia.org/wiki/RAR) (с выбором формата — RAR4 или RAR5) и [ZIP](https://ru.wikipedia.org/wiki/ZIP) 2.0, их обновление и проверка целостности.
* Распаковка архивов [RAR](https://ru.wikipedia.org/wiki/RAR), а также [ARJ](https://ru.wikipedia.org/wiki/ARJ), [bz2](https://ru.wikipedia.org/wiki/Bzip2), [CAB](https://ru.wikipedia.org/wiki/CAB), [GZ](https://ru.wikipedia.org/wiki/GZ), [ISO](https://ru.wikipedia.org/wiki/ISO-%D0%BE%D0%B1%D1%80%D0%B0%D0%B7), [JAR](https://ru.wikipedia.org/wiki/JAR), [LZH](https://ru.wikipedia.org/wiki/LZH), [TAR](https://ru.wikipedia.org/wiki/TAR), [UUE](https://ru.wikipedia.org/wiki/UUE), [XZ](https://ru.wikipedia.org/wiki/XZ), [Z](https://ru.wikipedia.org/wiki/Z), [ZIP](https://ru.wikipedia.org/wiki/ZIP), [ZIPX](https://ru.wikipedia.org/wiki/WinZip), [7z](https://ru.wikipedia.org/wiki/7z), 001.
* Поддержка многоядерности на соответствующих [ЦП](https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BD%D1%82%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) при упаковке и распаковке.
* Полная поддержка имён файлов в [Юникоде](https://ru.wikipedia.org/wiki/%D0%AE%D0%BD%D0%B8%D0%BA%D0%BE%D0%B4).
* Создание [самораспаковывающихся](https://ru.wikipedia.org/wiki/%D0%A1%D0%B0%D0%BC%D0%BE%D1%80%D0%B0%D1%81%D0%BF%D0%B0%D0%BA%D0%BE%D0%B2%D1%8B%D0%B2%D0%B0%D1%8E%D1%89%D0%B8%D0%B9%D1%81%D1%8F_%D0%B0%D1%80%D1%85%D0%B8%D0%B2) (SFX) архивов;
* Возможность управления из [командной строки](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B0%D0%BD%D0%B4%D0%BD%D0%B0%D1%8F_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B0);
* Позволяет выбрать метод и степень сжатия.

Минусы:

* Является платным по истечении пробного периода.

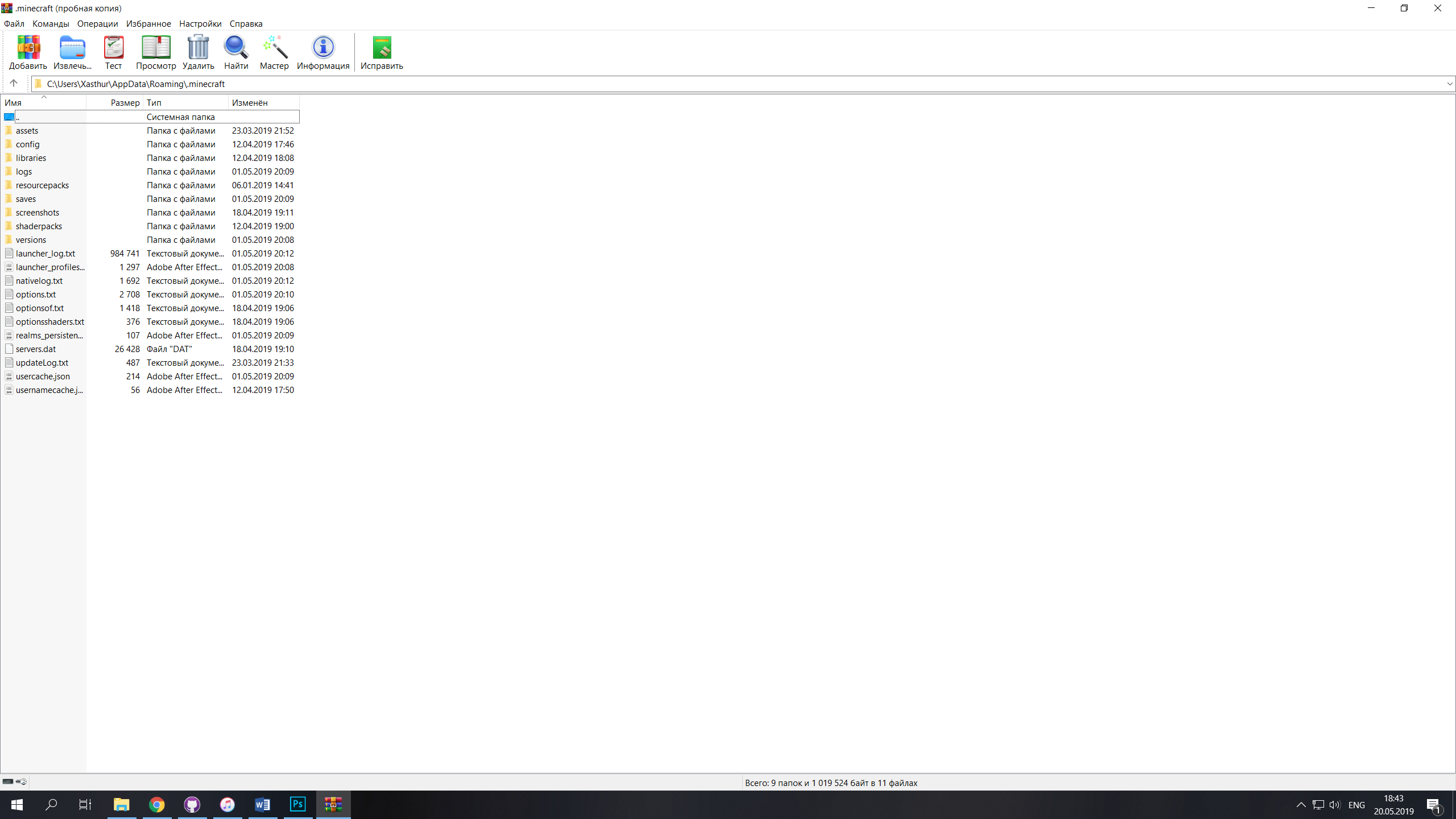
 Интерфейс программы показан на рисунках 1.3 и 1.4.

Рисунок 1.3 – Скриншот рабочего окна WinRAR

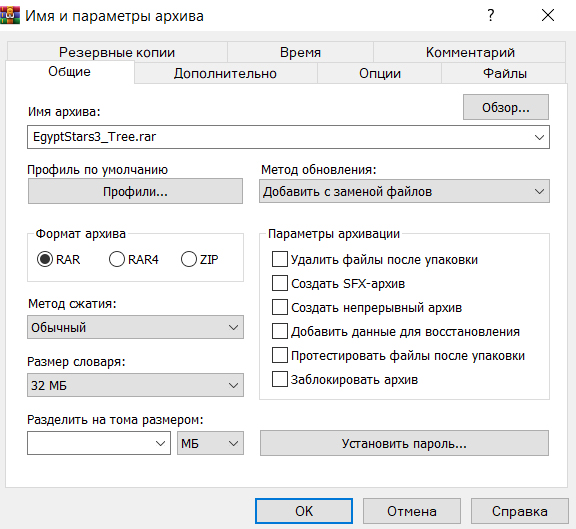


Рисунок 1.4 – Скриншот окна создания архива

**1.3 Описание популярных алгоритмов сжатия**

1.3.1 Алгоритм сжатия [RLE](http://en.wikipedia.org/wiki/Run-length_encoding) (Кодирование длин серий)[3].

RLE— [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) [сжатия данных](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), заменяющий повторяющиеся символы (серии) на один символ и число его повторов. Серией называется последовательность, состоящая из нескольких одинаковых символов. При кодировании строка одинаковых символов, составляющих серию, заменяется строкой, содержащей сам повторяющийся символ и количество его повторов.

После применения алгоритма RLE строка ABCABCABCDDDFFFFFF превратится в 1A1B1C1A1B1C1A1B1C3D6F.

Допустим, реализация метода RLE для записи длин серий использует переменную символьного типа «Char». В такую переменную можно записать числа от 0 до 255 включительно. Но функция Chr(x) позволяет записать символ с кодом от 0 до 127 включительно. Как же быть, если длина серии равна 128 символам и более? В этом случае серию разделяют на части так, чтобы длина части не превышала 127 символов. Например, серия, состоящая из 256 символов «A», будет закодирована следующей строкой (256=127+127+2): 127A127A2A

Плюсы:

* Один из наиболее быстрых алгоритмов сжатия;
* Эффективен для сжатия текстовых файлов с длинными сериями повторяющихся символов;
* Эффективен для сжатия простых графических изображений, таких как [иконки](https://ru.wikipedia.org/wiki/%D0%97%D0%BD%D0%B0%D1%87%D0%BE%D0%BA_(%D1%8D%D0%BB%D0%B5%D0%BC%D0%B5%D0%BD%D1%82_%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B0)) и графические рисунки;
* Звуковые данные, которые имеют длинные последовательные серии байт (такие как низкокачественные [звуковые семплы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BC%D0%BF%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) могут быть сжаты с помощью RLE после того, как к ним будет применено [Дельта-кодирование](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BB%D1%8C%D1%82%D0%B0-%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5).

Минусы:

* Плохо подходит для изображений с плавным переходом тонов, таких как фотографии.
* Не подходит для сжатия обычных текстов, где повторений незначительное количество.

1.3.2 Алгоритм сжатия Хаффмана[4].

Алгоритм Хаффмана — [жадный алгоритм](https://ru.wikipedia.org/wiki/%D0%96%D0%B0%D0%B4%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) оптимального [префиксного](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%84%D0%B8%D0%BA%D1%81%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) [кодирования](https://ru.wikipedia.org/wiki/%D0%AD%D0%BD%D1%82%D1%80%D0%BE%D0%BF%D0%B8%D0%B9%D0%BD%D0%BE%D0%B5_%D0%BA%D0%BE%D0%B4%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) алфавита с минимальной [избыточностью](https://ru.wikipedia.org/wiki/%D0%98%D0%B7%D0%B1%D1%8B%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C). В настоящее время используется во многих программах сжатия данных.

Этот метод кодирования состоит из двух основных этапов:

* Построение оптимального кодового дерева.
* Построение отображения код-символ на основе построенного дерева.

Идея алгоритма состоит в следующем: зная вероятности символов в сообщении, можно описать процедуру построения кодов переменной длины, состоящих из целого количества битов. Символам с большей вероятностью ставятся в соответствие более короткие коды. Коды Хаффмана обладают свойством [префиксности](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%84%D0%B8%D0%BA%D1%81%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4) (то есть ни одно кодовое слово не является префиксом другого), что позволяет однозначно их декодировать.

Плюсы:

* Очень эффективен при небольшом количестве уникальных символов;

Минусы:

* Для восстановления содержимого сжатого сообщения декодер должен знать таблицу частот, которой пользовался кодер. Следовательно, длина сжатого сообщения увеличивается на длину таблицы частот, которая должна посылаться впереди данных, что может свести на нет все усилия по сжатию;
* В процессе работы алгоритма сжатия вес узлов в дереве кодирования Хаффмана неуклонно растет, из-за чего при очень большом количество уникальных символов сжатие может быть неэффективно.

1.3.3 Алгоритм сжатия LZ77[5].

LZ77 – один из наиболее известных алгоритмов сжатия без потерь из семейства LZ\*. Использует словарный метод. В отличие от других методов уменьшения избыточности, таких как [RLE](https://ru.wikipedia.org/wiki/RLE) и [арифметическое сжатие](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D0%B8%D1%84%D0%BC%D0%B5%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B5_%D1%81%D0%B6%D0%B0%D1%82%D0%B8%D0%B5). LZ77 является алгоритмом со «скользящим окном», что эквивалентно неявному использованию словарного подхода, впервые предложенного в LZ78.

В кодируемых строках часто содержатся совпадающие длинные подстроки. Идея, лежащая в основе LZ77, заключается в замене повторений на ссылки на позиции в тексте, где такие подстроки уже встречались.

Информацию о повторении можно закодировать парой чисел - смещением назад от текущей позиции (offset) и длиной совпадающей подстроки (length). В таком случае, например, строка pabcdeqabcdepabcdeqabcde может быть представлена как pabcdeq⟨6,5⟩. Выражение ⟨6,5⟩ означает «вернись на 6 символов назад и выведи 5 символов».

Алгоритм LZ77 кодирует ссылки блоками из трёх элементов ⟨offset,length,next⟩. В дополнение к двум уже описанным элементам, новый параметр (next) означает первый символ после найденного совпадающего фрагмента. Если алгоритму не удалось найти совпадение, то считается, что offset=length=0.

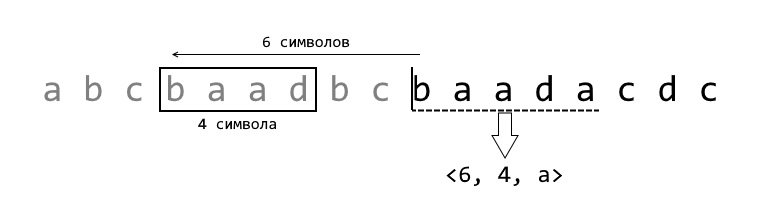
На рисунке 1.5 представлен принцип работы LZ77

Рисунок 1.5 – Принцип работы LZ77

Для декодирования LZ77 необходимо пройти по уже раскодированной строке назад, вывести необходимую последовательность, затем следующий символ.

Плюсы:

* Эффективен как для длинных серий символов, так и для строк, где повторения встречаются не в виде длинной последовательности символов.

Минусы:

* Невозможность кодирования подстрок, отстоящих друг от друга на расстоянии, большем длины словаря
* Длина подстроки, которую можно закодировать, ограничена размером буфера
* Малая эффективность при кодировании незначительного объёма данных

1.3.4 Алгоритм сжатия Deflate[6].

Deflate — это [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) [сжатия без потерь](https://ru.wikipedia.org/wiki/%D0%A1%D0%B6%D0%B0%D1%82%D0%B8%D0%B5_%D0%B1%D0%B5%D0%B7_%D0%BF%D0%BE%D1%82%D0%B5%D1%80%D1%8C), использующий комбинацию алгоритмов [LZ77](https://ru.wikipedia.org/wiki/LZ77) и [Хаффмана](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%A5%D0%B0%D1%84%D1%84%D0%BC%D0%B0%D0%BD%D0%B0).

Компрессия выполняется в два этапа:

* замена повторяющихся строк указателями (алгоритм LZ77);
* замена символов новыми символами, основываясь на частоте их использования (алгоритм Хаффмана).

Плюсы:

Наиболее эффективен при сжатии строки с большим количеством повторений.

Минусы:

Неэффективен для строк с малым количеством повторений.

**1.4. Формирование требований к проектируемому программному средству**

Подробно изучив алгоритмы сжатия данных и программы, позволяющие это сделать, я решил что должна из себя представлять программа, разработанная мною на языке Delphi в среде программирования «Embarcadero Delphi 10.3»:

1. математический функционал:

* Программа должна сжимать файлы, используя алгоритмы RLE, Хаффмана, LZ77 и Deflate;
* Программа должна создавать файлы декомпрессии из файлов, которые ею были сжаты;
* Программа должна подсчитывать время выполнения алгоритма и считывать размер сжатого файла.

1. прочие функциональные требования к программному средству:

* Программа должна сама определять уже сжатый файл и предлагать создать файл декомпрессии.
* Построение графиков сравнения времени выполнения и степени сжатия выбранных алгоритмов.

**1.5 Входные данные**

Входными данными для проектируемой программы может быть любой файл, нуждающийся в сжатии. Но важно, чтобы файл не был изначально сжат, потому что тогда нету никакого смысла в том, чтобы сжимать его еще раз. Для декомпрессии входной файл должен был ранее сжат данной программой, так как планируемая структура в сжатом файле не может быть построена в других программах или получена случайно.

**2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ**

**2.1** **Разработка используемых данных**

1) Соотношение алгоритма сжатия и расширения сжатого файла:

RLE - .xrle

Алгоритм Хаффмана - .xhfm

LZ77 - .xlz77

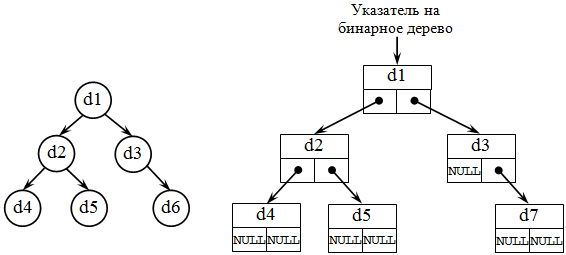
Deflate - .xdfl

2) Для хранения однотипной информации в большинстве случаев будет использоваться динамический массив.

3) Для реализации алгоритма Хаффмана потребуется использовать бинарное дерево.

Также в алгоритме Хаффмана будет использован динамический массив, состоящий из элементов типа THuffArrayElement.

Структура бинарного дерева и типа THuffArrayElement приведены на рисунках 2.1 и 2.2 соответственно.

Рисунок 2.1 – Структура бинарного дерева

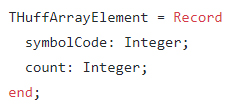


Рисунок 2.2 – Объявление типа THuffArrayElement

**2.2** **Анализ требований к программному средству и разработка фф функциональных требований**

В результате анализа требований к программному средству были составлены следующие функциональные требования:

* Корректность выполнения сжатия и декомпрессии с использованием следующих алгоритмов:

1. RLE;
2. Алгоритм Хаффмана;
3. LZ77;
4. Deflate;

* Поддержка открытия файлов с любым расширением
* Предпросмотр открытого файла
* Вывод результата декомпрессии в файл формата .txt
* Возможность не создавать на диске сжатые файлы
* Возможность выбрать любую комбинацию алгоритмов сжатия для сравнения
* Цена деления на графике должна зависеть от результатов сжатия
* Возможность масштабировать графики

**2.3 Разработка алгоритма программного средства**

Изначально вниманию пользователя представляется начальная страница, где для продолжения работы программы ему необходимо нажать на кнопку «Open file». После открытия файла программа по его расширению определяет, был ли файл сжат данной программой в прошлом. Если программа определяет, что файл был сжат ею, то активируется функционал, позволяющий преобразовать данный файл в исходный. В противном случае активируется функционал, позволяющий сжать данный файл и проанализировать использованные алгоритмы.

Функционал сжатия:

При нажатии на кнопку Compress будет произведено сжатие выбранного файла выбранными алгоритмами.

Если стоит галочка на пункте “Analyse”, то после завершения сжатия всеми выбранными алгоритмами будут построены графики, с помощью которых можно сравнить эффективность каждого алгоритма на выбранном файле.

Если стоит галочка на пункте “Export comp.”, то после завершения сжатия всеми выбранными алгоритмами файлы, созданные в процессе сжатия, удалены не будут.

Функционал декомпрессии:

При нажатии на кнопку Decompress(\*алгоритм, использованный для сжатия\*) будет произведена декомпрессия файла с использованием соответствующего алгоритма.

Если стоит галочка на пункте “Export as .txt”, то ,после завершения декомпрессии соответствующим алгоритмом, файл, созданный в процессе декомпрессии, удален не будет.

**3. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

**3.1 Общая схема программы**

Схема алгоритма программы представлена в *приложении 1*. Также она выполнена на листе формата А1.

Словесный алгоритм:

1. Программа начинается с создания пользовательского интерфейса
2. У пользователя есть возможность открыть файл
3. Программа определяет, был ли данный файл сжат ею ранее
4. Если программа определяет файл как сжатый ею ранее, то перейти к пункту 11
5. Программа определила файл как неизвестный, открывается возможность его сжать
6. Происходит выбор алгоритмов сжатия
7. Происходит сжатие файлов соответствующим алгоритмом и создание сжатых файлов
8. Если на пункте “Analyse” стояла галочка, то происходит вывод графиков, позволяющих сравнить выбранный алгоритмы по степени сжатия и времени
9. Если на пункте “Export comp.” не стояла галочка, то файлы, созданные в процессе сжатия, удаляются.
10. Перейти к пункту 2
11. Программа определила файл как сжатый ею ранее, открывается возможность провести декомпрессию.
12. Происходит декомпрессия выбранного файла
13. Если на пункте “Export as .txt” не стояла галочка, то файлы, созданные в процессе декомпрессии, удаляются.
14. Перейти к пункту 2

**3.2 Алгоритм сжатия RLE**

(Блок-схема алгоритма в *приложении 2*)

Словесный алгоритм:

1. Считывание текущего символа, если не достигли конца строки
2. Подсчет длины последовательности одинаковых символов до достижения длины последовательности равной 127 символов
3. Запись в файл длину последовательности и сам символ
4. Перейти к пункту 1

**3.3 Алгоритм сжатия Хаффмана**

(Блок-схема алгоритма в *приложении 3*)

При сжатии алгоритмом Хаффмана в начало файла должна быть помещена информация для восстановления дерева при декомпрессии(символы, записанные в порядке убывания их встречаемости). Данная таблица отделяется от сжатой информацией 2мя символами ESC(код 27). Также обрабатывается ситуация, когда в конце таблицы находится 3 символа ESC. Данная ситуация возможно в том случае, если в самом тексте встречается символ ESC и он является самым редким.

Также количество “бит”, записанных в строку, не всегда кратно 7. Последний “байт” дополняется нужным количеством 0, а в конец всей строки пишется количество 0, которое потребовалось добавить.

Словесный алгоритм:

1. Создание массива с данными о количестве каждого символа на основе полученной строки*(Приложение 4)*
2. Сортировка полученного массива по встречаемости*(Приложение 5)*
3. Построение дерева на основе полученного массива*(Приложение 6)*
4. Преобразование строки на основе построенного дерева*(Приложение 7)*
5. Запись в файл информации для восстановления дерева
6. Запись в файл преобразованной строки
7. Очистка дерева*(Приложение 8)*

**3.4 Алгоритм сжатия LZ77**

(Блок-схема алгоритма в *приложении 9*)

Данная реализация алгоритма LZ77 имеет буфер в 127 символов. Используется именно 127 символов, т.к функция Chr(x) возвращает символ с номером от 0 до 127.

Словесный алгоритм:

1. Считывание очередного символа, если не достигнут конец строки
2. Сравнение текущей последовательности с буфером. Если не найдено общей части, перейти к пункту 5
3. Добавление символа в последовательность, если не достигнут конец строки
4. Сохранение информации о смещении и длине повторяющейся части
5. Перейти к пункту 2
6. Запись в файл информации о смещении, длине повторяющейся части и символе, следующем за повторяющейся частью.
7. Очистка строки, которая содержит текущую последовательность.
8. Перейти к пункту 1

**3.5 Алгоритм сжатия Deflate**

(Блок-схема алгоритма в *приложении 10*)

Сжатие алгоритмом Deflate использует комбинацию LZ77 и алгоритма Хаффмана, которые были уже реализованы в программе.

Словесный алгоритм:

1. Преобразование строки с использованием алгоритма LZ77
2. Преобразование строки с использованием алгоритма Хаффмана

**3.6 Алгоритм декомпрессии файла, сжатого RLE**

(Блок-схема алгоритма в *приложении 11*)

Декомпрессия происходит путем последовательного считывания 2х символов. Код первого означает длину последовательности, состоящей из второго символа.

Словесный алгоритм:

1. Считывается последовательность из 2х символов, если не достигнут конце строки
2. Запись в строку 2го символа из последовательности количество раз, равное коду 1го символа.
3. Перейти к пункту 1

**3.7 Алгоритм декомпрессии файла, сжатого алгоритмом Хаффмана**

(Блок-схема алгоритма в *приложении 12*)

Создание массива происходит путем добавления в массив всех символов, находящихся в строке до последовательности символов, состоящей из 2х символов ESC(код 27). Если встречается последовательность состоящая из 3х символов ESC, то данный символ также заносится в массив. Далее на основе данного массива строится дерево, которое полностью повторяет дерево, полученное при сжатии. После этого происходит преобразование строки таким образом, что каждый символ записывается как его двоичная запись. С конца строки удаляются лишние 0, которые потребовались при сжатии. И происходит преобразование сжатого текста в исходный на основе построенного дерева.

Словесный алгоритм:

1. Создание массива с данными о количестве каждого символа на основе таблицы в начале файла
2. Построение дерева на основе полученного массива
3. Преобразование строки на основе построенного дерева
4. Запись в файл преобразованной строки
5. Очистка дерева

**3.8 Алгоритм декомпрессии файла, сжатого LZ77**

(Блок-схема алгоритма в *приложении 13*)

Данная реализация алгоритма LZ77 имеет буфер в 127 символов. Используется именно 127 символов, т.к функция Chr(x) возвращает символ с номером от 0 до 127.

Словесный алгоритм:

1. Происходит считывание 3х символов из строки, если не достигнут её конец
2. Если 1 и 2 символы имеют код не равный 0, то происходит смещение в буфере и выводится требуемая последовательность
3. Выводится 3 символ
4. Перейти к пункту 1

**3.9 Алгоритм декомпрессии файла, сжатого Deflate**

(Блок-схема алгоритма в *приложении 14*)

Декомпрессия файла, сжатого алгоритмом Deflate, происходит путем преобразования строки из файла сначала с использованием алгоритма Хаффмана, а потом LZ77.

Словесный алгоритм:

1. Применение к строке алгоритма, позволяющего произвести декомпрессию строки, сжатой с использованием алгоритма Хаффмана.
2. Применение к строке алгоритма, позволяющего произвести декомпрессию строки, сжатой с использованием алгоритма LZ77.

**3.10 Алгоритм построения графиков**

(Блок-схема алгоритма в *приложении 15*)

Графики строятся на основе данных, которые были переданы в динамическом массиве после сжатия. Для добавления алгоритма к сравнению требуется увеличить длину массива и добавить соответствующую информацию. После добавления в графике появится новое поле, характеризующее добавленный алгоритм.

Словесный алгоритм:

1. Получение информации об использовавшихся в сжатии алгоритмах
2. Построение осей
3. Построение штрихов
4. Подсчет и вывод цены деления около штрихов
5. Построения столбцов на графике с их названиями
6. **СОЗДАНИЕ ПРОГРАММНОГО СРЕДСТВА**
   1. **Описание модулей**

Программа включает в себя 2 различных модуля:

4.1.1 MainForm

Этот модуль отвечает за основной графический интерфейс и логику программы. На ней происходит открытие файла(TSpeedButton, TOpenDialog), выбор алгоритмов сжатия(TCheckBox), предпросмотр файла(TMemo в режиме Read-only), кнопка начала сжатия/декомпрессии(TSpeedButton).

В данном модуле определены следующие типы:

* TFile = TextFile;

(Тип файла)

* THuffArrayElement = Record  
   symbolCode: Integer;  
   count: Integer;  
  end;

(Запись, использующаяся в массиве типа THuffArray, symbolCode – код символа, count – кол-во символов с кодом symbolCode)

* THuffArray = Array of THuffArrayElement;

(Массив, содержащий информацию о встречаемости символов в строке)

* HUFFTreePointer = ^HUFFTreeNode;
* HUFFTreeNode = Record  
   symbol: Integer;  
   left: HUFFTreePointer;  
   right: HUFFTreePointer;  
  end;

(Дерево, с помощь которого происходит кодирование Хаффмана)

* + 1. ChartForm

Данный модуль отвечает за показ графиков, которые строятся на основе информации, полученной во время сжатия файла. Сами графики строятся на компоненте типа TImage.

В данном модуле определены следующие типы:

* TAlgRec = Record  
   Name: String;  
   Data: Integer;  
  end;

(Запись, Name – название алгоритма, Data - затраченное время/размер файла)

* TGraphDataArray = Array of TAlgRec;

(Массив, содержащий информацию об алгоритме сжатия)

Информация из формы MainForm в ChartForm передается через процедуру LoadGraphData.

**4.2 Список всех процедур, их параметры и назначение:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Имя подпрограммы** | **Описание** | **Заголовок подпрограммы** | **Имя параметра** | **Назначение параметра** |
| Unit MainForm.pas | | | | |
| GetFileSize | Функция получения размера файла | function GetFileSize(FileName: String): Int64; | FileName | Имя файла |
| RLECompressString | Сжатие строки алгоритмом RLE | function RLECompressString(str: String):String; | str | Сжимаемая строка |
| RLECompress | Начало сжатия RLE | procedure RLECompress(s:string; newPath:String); | s | Рабочая строка |
| newPath | Путь сохранения |
| DisposeHuffTree | Очистка дерева Хаффмана | procedure DisposeHuffTree(var head: HuffTreePointer); | head | Указатель на голову дерева |
| HUFFGetCount | Заполнение массива символов | function HUFFGetCount(var arr:THuffArray; str: String):integer; | str | Строка для подсчета |
| HUFFSort | Сортировка массива | procedure HUFFSort(var arr:THuffArray); | arr | Массив сортируемый |
| HUFFCreateTree | Создание дерева Хаффмана | procedure HUFFCreateTree(var head: HuffTreePointer; arr:THuffArray); | head | Указатель на голову дерева |
| arr | Массив с вероятностями |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HUFFGetBSymbol | Преобразование символа в строку используя дерево Хаффмана | function HUFFGetBSymbol(head:HuffTreePointer; c:Char):String; | head | Указатель на голову дерева |
| c | Символ |
| StrBinToInt | Преобразование последовательности 0 и 1 в число десятичной сс | function StrBinToInt(s: String; pos,l:integer):Integer; | s | Строка сжатия |
| pos | Начальная позиция |
| l | Длина последовтельности |
| HUFFStringBinaryToChar | Запуск преобразования строки из 0 и 1 в символы | function HUFFStringBinaryToChar(s:String):String; | s | Строка сжатия |
| HUFFCompressString | Сжатие строки алгоритмом Хаффмана | function HUFFCompressString(head:HuffTreePointer; s: String):String; | head | Указатель на голову дерева |
| s | Строка сжатия |
| HUFFCompress | Начало сжатия Хаффмана | procedure HUFFCompress(s:String; newPath:String); | s | Строка |
| newPath | Путь сохранения |
| LZ77CompressString | Сжатие строки методом LZ77 | function LZ77CompressString(s:String):String; | s | Строка сжатия |
| LZ77Compress | Начало сжатия LZ77 | procedure LZ77Compress(s:String; newPath:String); | s | Рабочая строка |
| newPath | Путь сохранения |
| DFLCompress | Сжатие методом Deflate | procedure DFLCompress(s:String; newPath:String); | s | Рабочая строка |
| newPath | Путь сохранения |
| CompressionStart | Запуск сжатия в зависимости от выбранного типа | procedure CompressionStart(s:string;z:integer;var newPath,newName:String); | s | Строка для сжатия |
| z | Тип сжатия |
| newPath | Путь сохранения |
| newName | Новое имя |
| decompressRLEString | Декомпрессия RLE строки | function decompressRLEString(str:String):String; | str | Строка для декомпрессии |
| decompressRLE | Декомпрессия RLD | procedure decompressRLE(s,newPath:String); | s | Рабочая строка |
| newPath | Путь сохранения |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| IntToBin7 | Преобразование числа в строку, содержащую двоичную запись данного числа: | function IntToBin7(d: Integer): string; | d | Преобразуемое число |
| HUFFCharToStringBinary | Преобразование символов в строку 0 и 1 | function HUFFCharToStringBinary(s:string):String; | s | Строка для декомпрессии |
| getSymbolFromHuffTree | Поиск символа в дереве Хаффмана | function getSymbolFromHuffTree(head:HuffTreePointer; buffSequence:string):Integer; | head | Указатель на голову дерева |
| buffSequence | Последовательность |
| decodeHuffString | Расшифровка строки Хаффмана | function decodeHuffString(head:HuffTreePointer; s:string):String; | head | Указатель на голову дерева |
| s | Строка для декомпрессии |
| decompressHFM | Декомпрессия строки сжатой алгоритмом Хаффмана | procedure decompressHFM(s:string;newPath:String); | s | Рабочая строка |
| newPath | Путь сохранения |
| decompressLZ77String | Декомпрессия строки сжатой алгоритмом LZ77 | function decompressLZ77String(s:string):String; | s | Строка для декомпрессии |
| decompressLZ77 | Начало декомпрессии LZ77 | procedure decompressLZ77(s:string;newPath:String); | s | Рабочая строка |
| newPath | Путь сохранения |
| decompressDFL | Декомпрессия строки сжатой методом Deflate | procedure decompressDFL(s:string;newPath:String); | s | Рабочая строка |
| newPath | Путь сохранения |
| StartDecompression | Начало декомпрессии в зависимости от выбранного метода | procedure StartDecompression(path:String;z:integer); | path | Путь к файлу |
| z | Тип декомпрессии |
| Unit ChartForm.pas | | | | |
| LoadGraphData | Процедура загрузки данных в переменные | procedure LoadGraphData(TimeArr:TGraphDataArray;SizeArr:TGraphDataArray); | TimeArr | Массив времени |
| SizeArr | Массив размеров |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| drawDashes | Рисование штрихов на осях | procedure drawDashes(I:TImage;offsetX,offsetY:Integer); | I | Изображение TImage |
| offsetX | Смещение по X |
| offsetY | Смещение по Y |
| calcY | Подсчет Y | function calcY(I:TImage;offsetY,min,max:Integer;Element:TAlgRec):Integer; | I | Изображение TImage |
| offsetX | Смещение по X |
| offsetY | Смещение по Y |
| Min | Минимум на оси |
| Мax | Максимум на оси |
| Element | Информация о текущем столбце |
| drawMinMax | Вывод значений делений | procedure drawMinMax(I:TImage; offsetX,offsetY,min,max,T:Integer); | I | Изображение TImage |
| offsetX | Смещение по X |
| offsetY | Смещение по Y |
| Min | Минимум на оси |
| Мax | Максимум на оси |
| T | Тип графика |
| drawPillars | Построение столбцов | procedure drawPillars(I:TImage; offsetX,offsetY:Integer; DataArr:TGraphDataArray;T:Integer); | I | Изображение TImage |
| offsetX | Смещение по X |
| offsetY | Смещение по Y |
| DataArr | Массив с данными |
| T | Тип графика |
| drawAxises | Построение осей | procedure drawAxises(I:TImage;DA:TGraphDataArray;T:Integer); | I | Изображение TImage |
| DA | Массив с данными |
| T | Тип графика |
| drawSizeGraph | Построение графика размера | procedure drawSizeGraph(I:TImage;SA:TGraphDataArray); | I | Изображение TImage |
| SA | Массив размеров |
| drawTimeGraph | Построение графика времени | procedure drawTimeGraph(I:TImage;TA:TGraphDataArray); | I | Изображение TImage |
| TA | Массив времени |

1. **ТЕСТИРОВАНИЕ, ПРОВЕРКА РАБОТОСПОСОБНОСТИ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ**

Для того, чтобы соответствовать требованиям к проектируемому программному средству, необходимо, чтобы оно прошло некоторое тестирование, способное выявить его недостатки.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| №  Теста | Тестируемая функциональность | Последовательность  действий | Ожидаемый результат | Полученный результат |
| 1 | Сжатие алгоритмом Хаффмана с выводом файла | Запустить программы. Открыть сжимаемый файла. Поставить галочку на “Export comp.” и “Huffman” Нажать на кнопку Compress | Будет создан сжатый файл с расширением .xhfm. После чего произведена декомпрессия программой. Содержание начального файла должно полностью совпадать с тем, что получилось после декомпрессии. | Тест пройден |
| 2 | Сжатие алгоритмом RLE с выводом файла | Запустить программы. Открыть сжимаемый файла. Поставить галочку на “Export comp.” и “RLE” Нажать на кнопку Compress | Будет создан сжатый файл с расширением .xrle. После чего произведена декомпрессия программой. Содержание начального файла должно полностью совпадать с тем, что получилось после декомпрессии. | Тест пройден |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 3 | Сжатие алгоритмом LZ77 с выводом файла | Запустить программы. Открыть сжимаемый файла. Поставить галочку на “Export comp.” и “LZ77” Нажать на кнопку Compress | Будет создан сжатый файл с расширением .xlz77. После чего произведена декомпрессия программой. Содержание начального файла должно полностью совпадать с тем, что получилось после декомпрессии. | Тест пройден |
| 4 | Сжатие алгоритмом Deflate с выводом файла | Запустить программы. Открыть сжимаемый файла. Поставить галочку на “Export comp.” и “Deflate” Нажать на кнопку Compress | Будет создан сжатый файл с расширением .xdfl. После чего произведена декомпрессия программой. Содержание начального файла должно полностью совпадать с тем, что получилось после декомпрессии. | Тест пройден |
| 5 | Декомпрессия алгоритмом Хаффмана с выводом файла | Запустить программы. Открыть файл с расширением .xhfm. Поставить галочку на “Export as .txt” Нажать на кнопку Decompress | Будет создан файл с расширением .txt. Содержание начального файла должно полностью совпадать с тем, что получилось после декомпрессии | Тест пройден |
| 6 | Декомпрессия алгоритмом RLE с выводом файла | Запустить программы. Открыть файл с расширением .xrle. Поставить галочку на “Export as .txt” Нажать на кнопку Decompress | Будет создан файл с расширением .txt. Содержание начального файла должно полностью совпадать с тем, что получилось после декомпрессии | Тест пройден |
| 7 | Декомпрессия алгоритмом LZ77 с выводом файла | Запустить программы. Открыть файл с расширением .xlz77. Поставить галочку на “Export as .txt” Нажать на кнопку Decompress | Будет создан файл с расширением .txt. Содержание начального файла должно полностью совпадать с тем, что получилось после декомпрессии | Тест пройден |
| 8 | Декомпрессия алгоритмом Deflate с выводом файла | Запустить программы. Открыть файл с расширением .xdfl. Поставить галочку на “Export as .txt” Нажать на кнопку Decompress | Будет создан файл с расширением .txt. Содержание начального файла должно полностью совпадать с тем, что получилось после декомпрессии | Тест пройден |
| 9 | Попытка сжатия без выбранных алгоритмов | Запустить программы. Открыть сжимаемый файла. Нажать на кнопку Compress | Ничего не произойдет | Тест пройден |
| 10 | Попытка сжатия 4мя алгоритмами с последующим выводом графиков | Запустить программы. Открыть сжимаемый файла. Поставить галочку на “Analyse” Нажать на кнопку Compress | После процесса сжатия открылось окно с построенными графиками | Тест пройден |
| 11 | Попытка сжатия без выбора алгоритмов с последующим выводом графиков | Запустить программы. Открыть сжимаемый файла. Поставить галочку на “Analyse” Нажать на кнопку Compress | Откроются графики, содержащие информацию только о несжатом файле | Тест пройден |

**6. Руководство по установке и использованию**

**6.1 Основные требования для запуска данного программного средства:**

* + ОС: Версия Microsoft Windows от XP и выше;
  + Процессор: Pentium® III 800 МГц или AMD Athlon;
  + RAM: От 128 Мб;
  + Место на диске: от 3 Мб свободного места.

Исходя из данных требований следует, что данная программа может запускаться практически на любом компьютере.

**6.2 Руководство по установке**

Установка программы не требуется, т.к она является переносимым приложением.

Переносимое приложение — [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), которое для своего запуска не требует процедуры [установки](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) и может полностью храниться на съёмных носителях информации, что позволяет использовать данное ПО на многих компьютерах.

**6.3 Пример использования программы**

1. Открыть программу

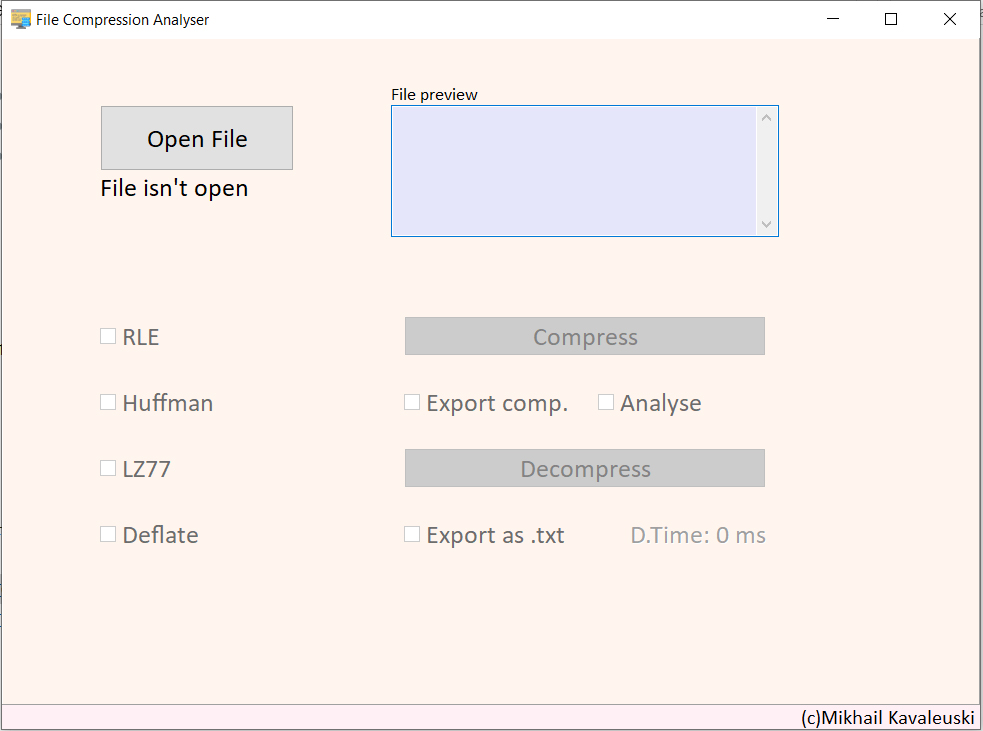


Рисунок 6.1 – Скриншот рабочего окна программы

2. Открыть файл для сжатия, выбрать все алгоритмы, нажать на кнопку Compress

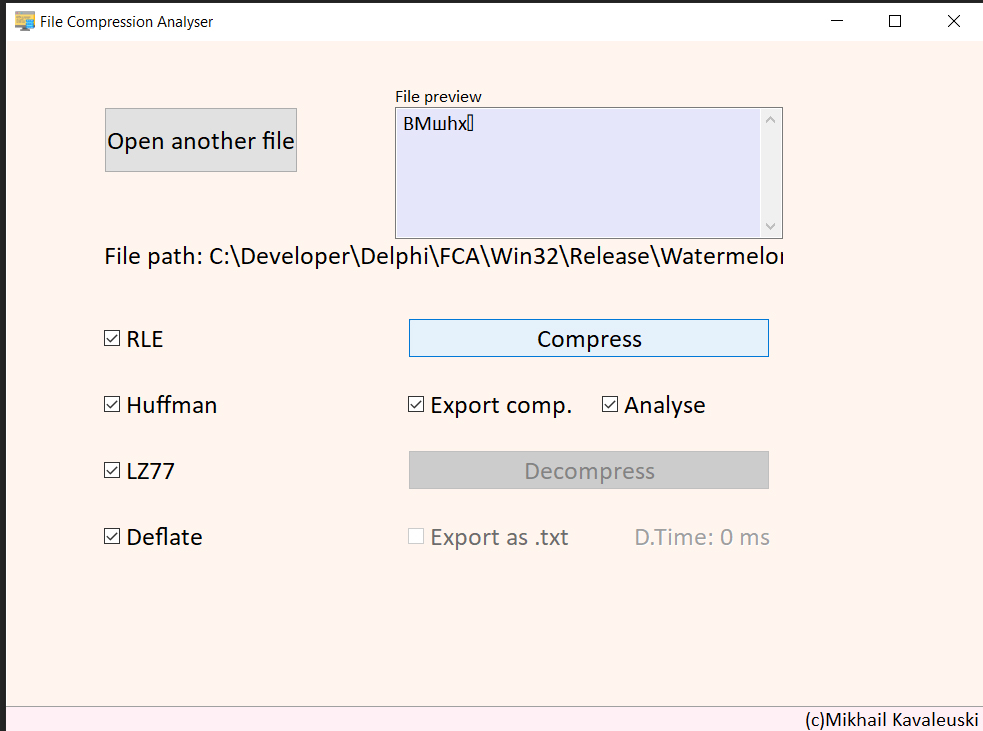
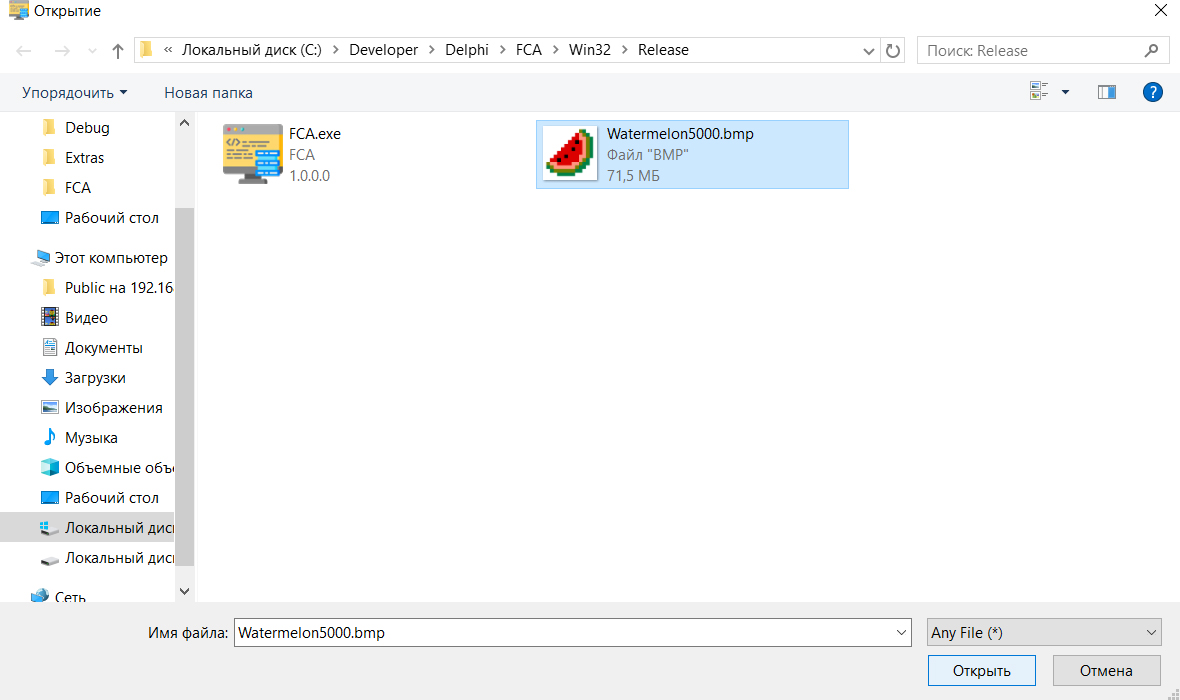


Рисунок 6.2 – Иллюстрация процесса запуска сжатия

3. После завершения процесса откроется окно, содержащее информацию, позволяющую проанализировать выбранные алгоритмы.

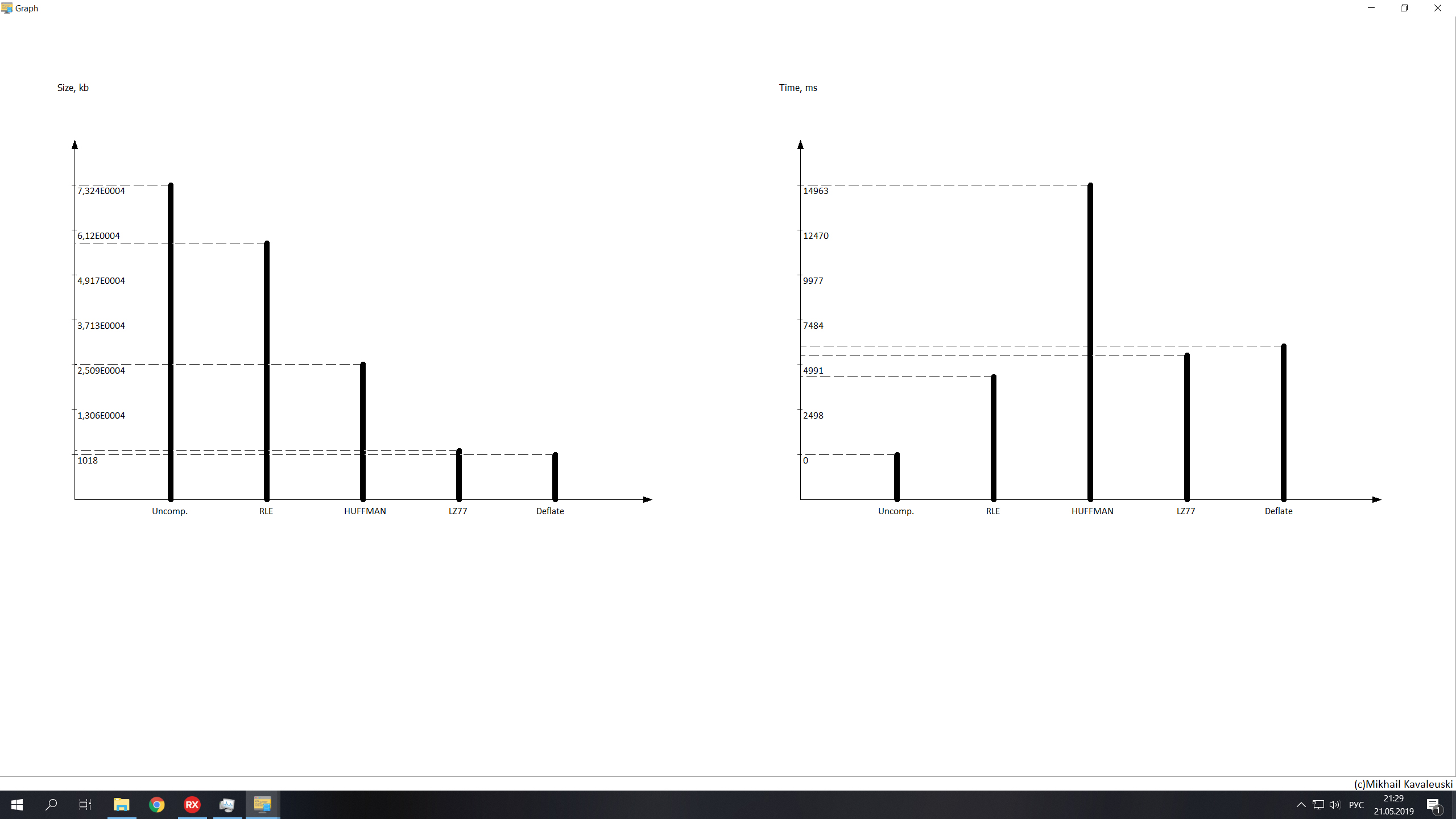


Рисунок 6.3 – Графики размера и времени

**ЗАКЛЮЧЕНИЕ**

В результате работы над курсовым проектом было создано исправно работающее приложение для сжатия файлов алгоритмами RLE, LZ77, Deflate и Хаффмана. Разработка приложения включала в себя различных методов сжатия файлов. Были изучены некоторые возможности создания приложений в Delphi 10 и формирование конкретных функциональных требований к программе на основе возможностей языка. Затем были разработаны структуры данных, разработана примерная архитектура приложения. Далее были детализированы все функции. Программа была отлажена и протестирована сначала разработчиком, а затем несколько раз обычными пользователями. После испытаний были внесены корректировки в интерфейсе, работе некоторых функций. Пройдя все вышеперечисленные этапы, на выходе получилась корректно работающее программное средство для сжатия файлов и анализа использовавшихся алгоритмов.

**СПИСОК ЛИТЕРАТУРЫ**

[1] wecompress.com [Электронный портал]. – Электронные данные. – Режим доступа: <https://www.wecompress.com/ru/>

[2] win-rar.com [Электронный портал]. – Электронные данные. – Режим доступа: <https://www.win-rar.com/start.html?&L=4>

[3] habr.com [Электронный портал]. – Электронные данные. – Режим доступа:  
<https://habr.com/ru/post/141827/>

[4] Wikipedia.org [Электронный портал]. – Электронные данные. – Режим доступа: <https://en.wikipedia.org/wiki/Huffman_coding>

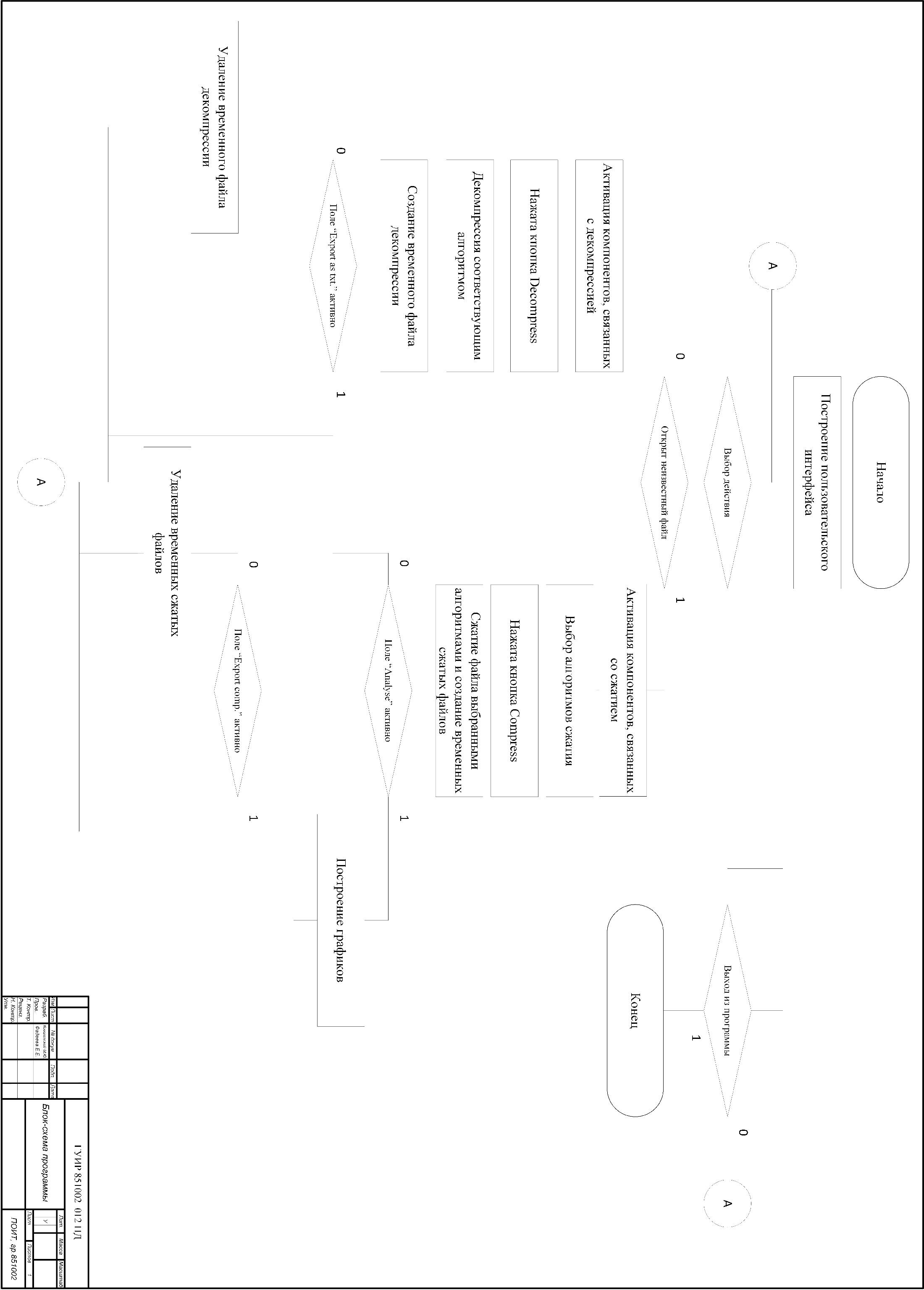
[5] neerc.ifmo.ru [Электронный портал]. – Электронные данные. – Режим доступа:  
<https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B_LZ77_%D0%B8_LZ78>

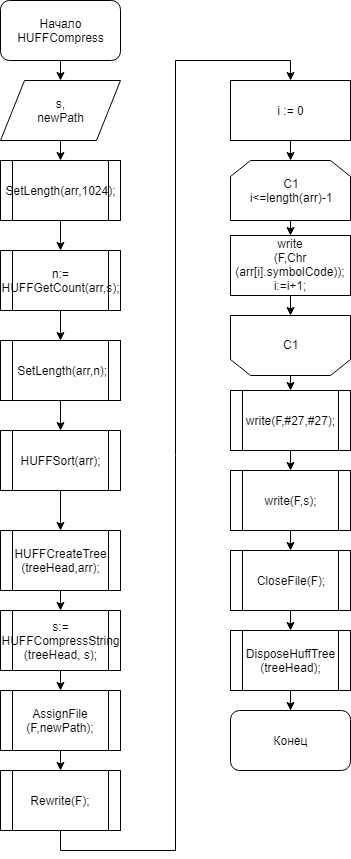
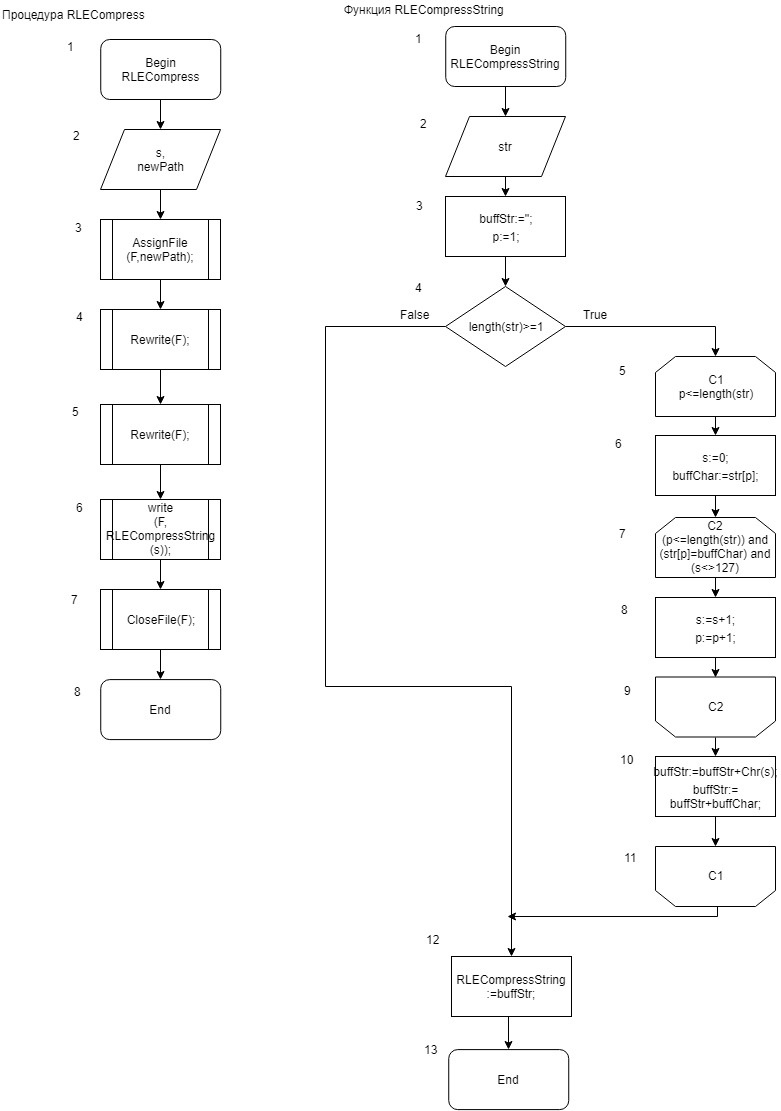
[6] Wikipedia.org [Электронный портал]. – Электронные данные. – Режим доступа:  
<https://ru.wikipedia.org/wiki/Deflate>

**ПРИЛОЖЕНИЯ**

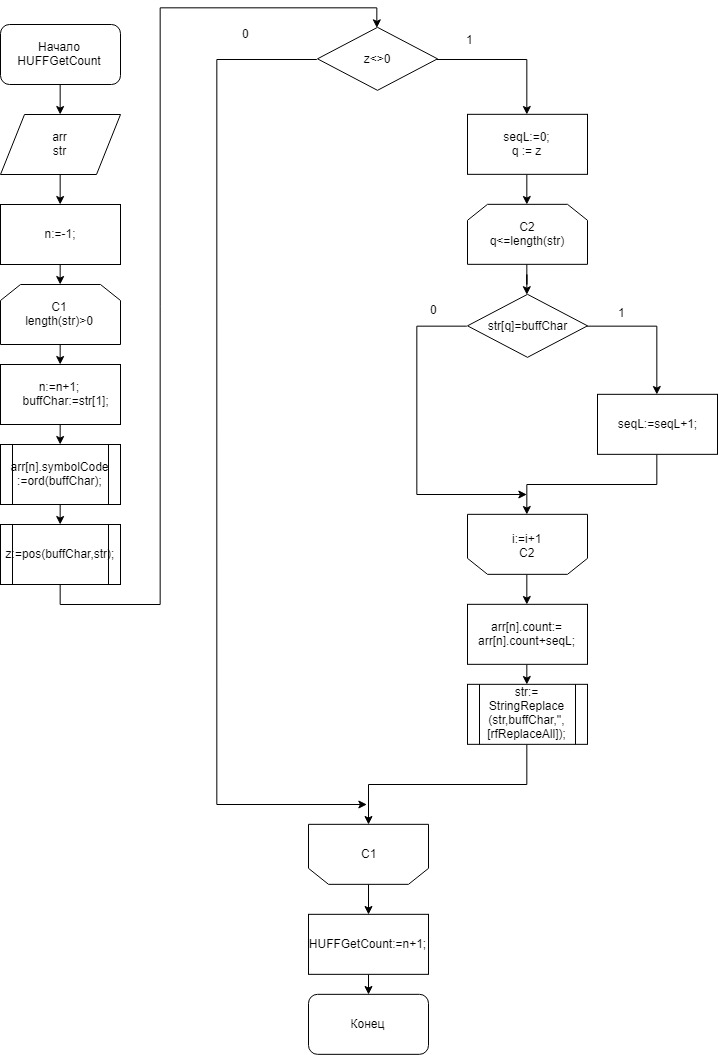
Приложение 1 (схема алгоритма программы на А1)

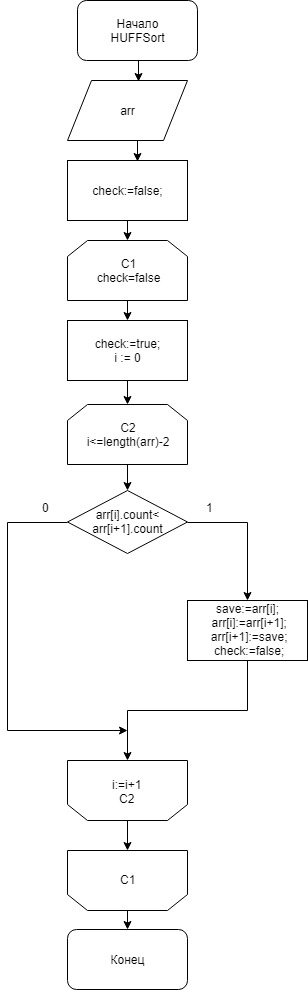
Приложение 2 (Алгоритм сжатия RLE)



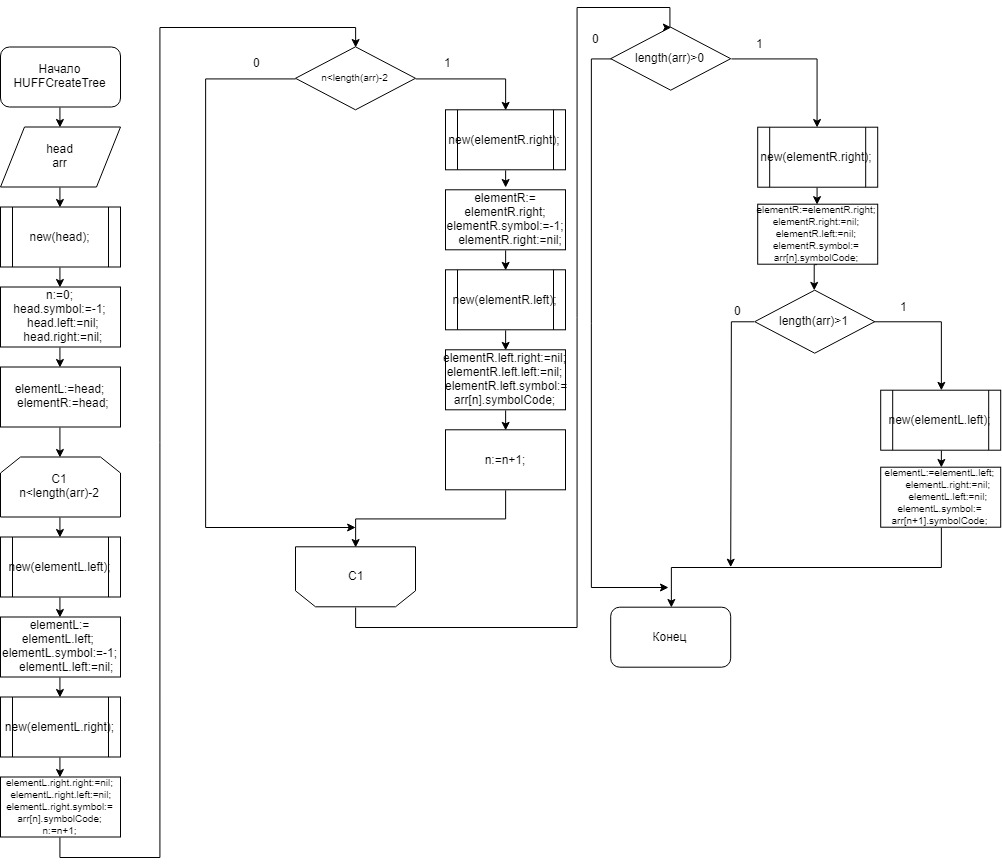
Приложение 3 (Алгоритм сжатия Хаффмана)

Приложение 4 (Создание массива с данными о количестве символов)

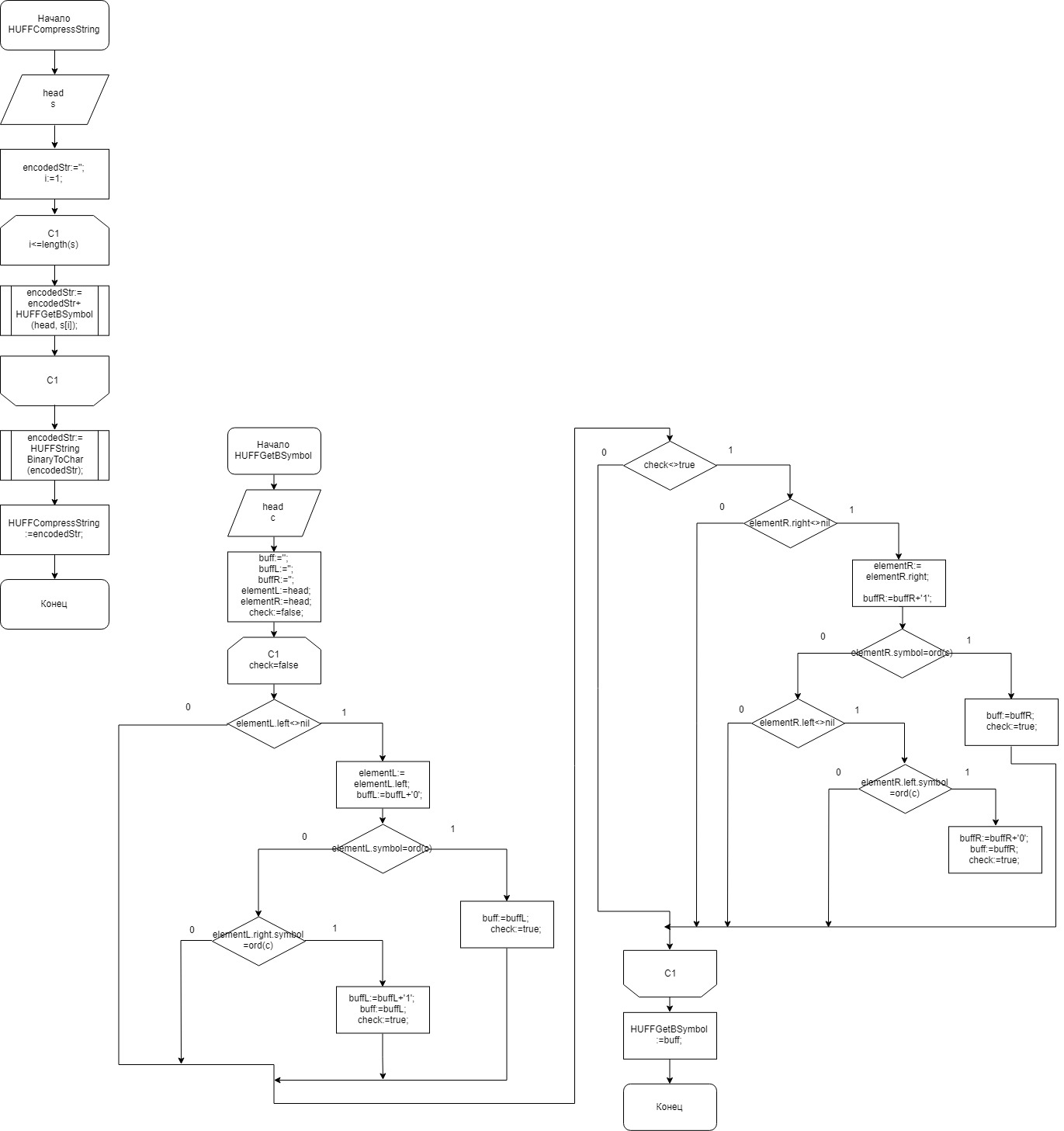


Приложение 5 (Сортировка полученного массива по встречаемости)

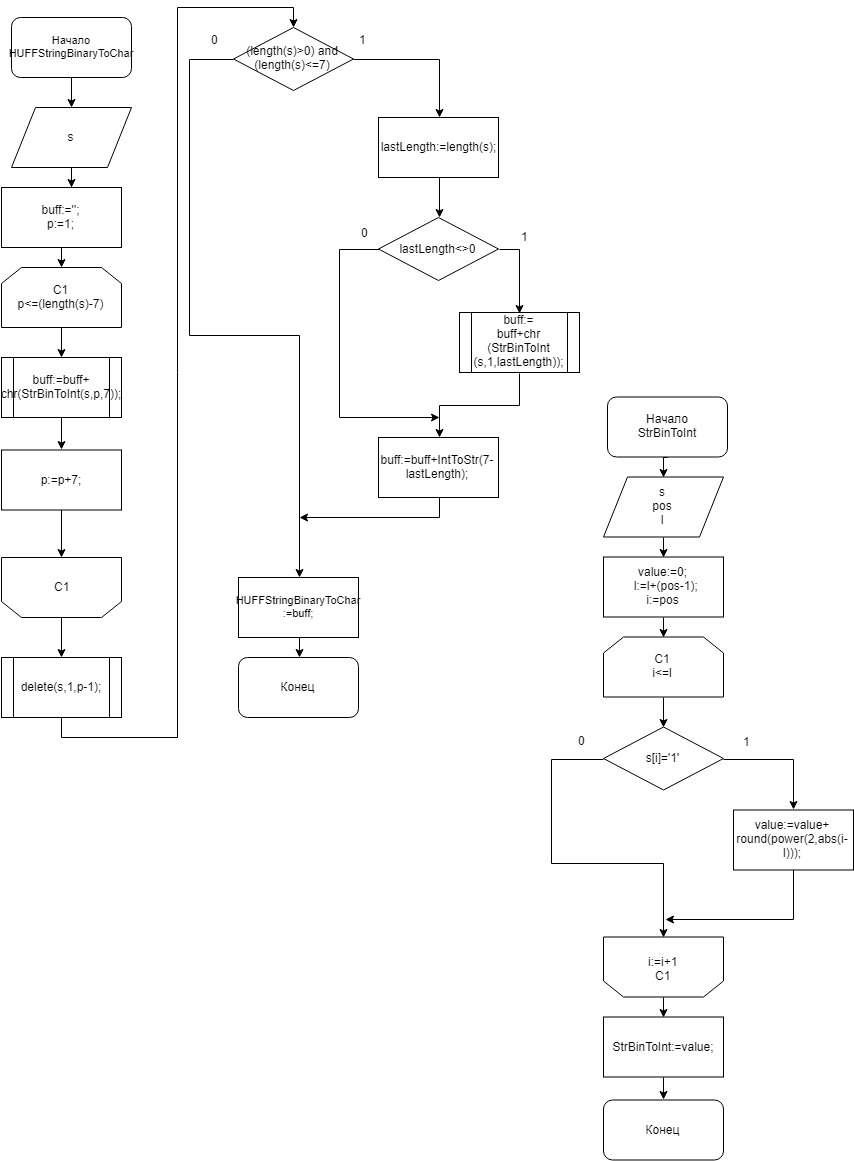
Приложение 6 (Построение дерева на основе полученного массива)



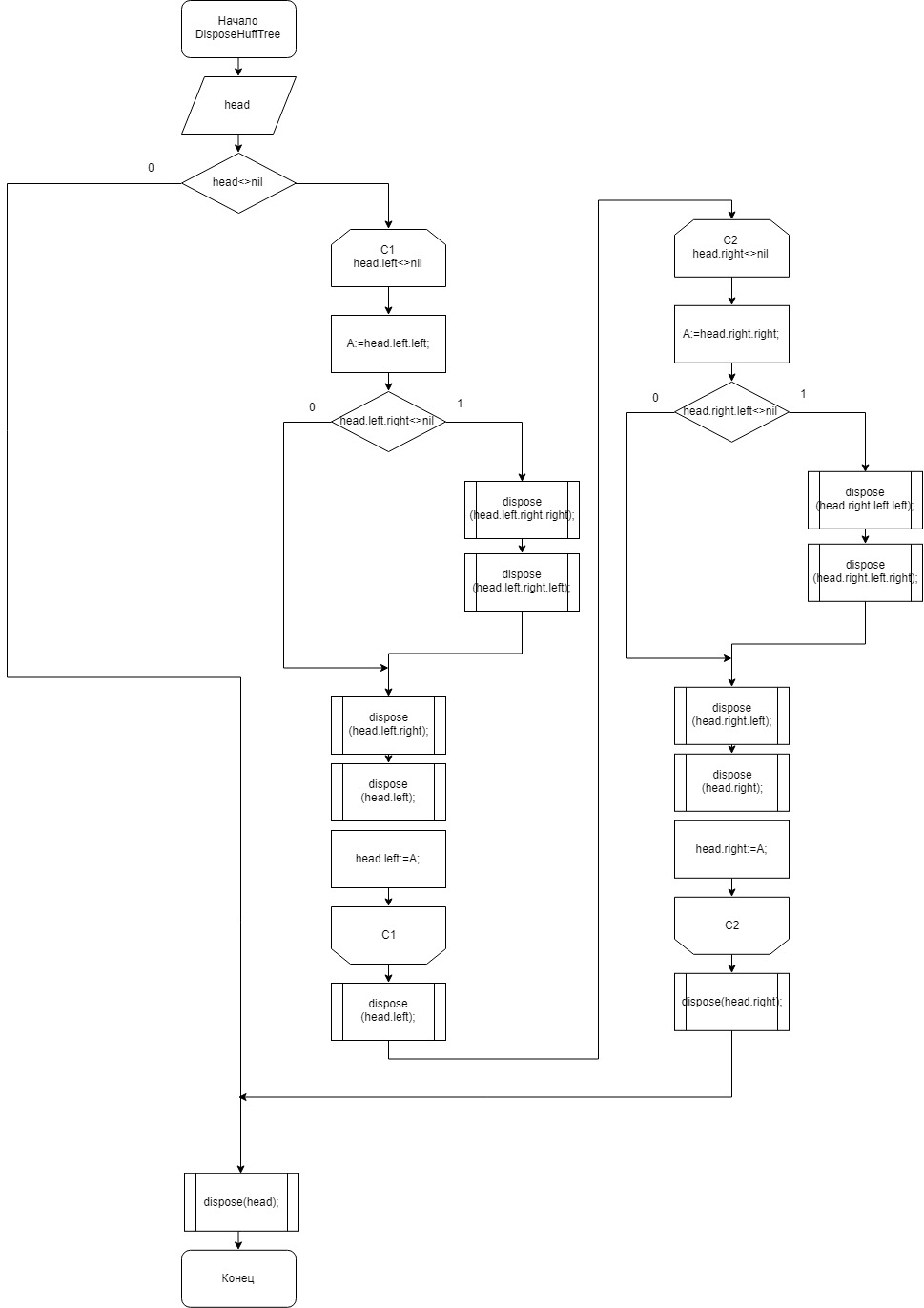
Приложение 7 (Преобразование строки на основе построенного дерева)



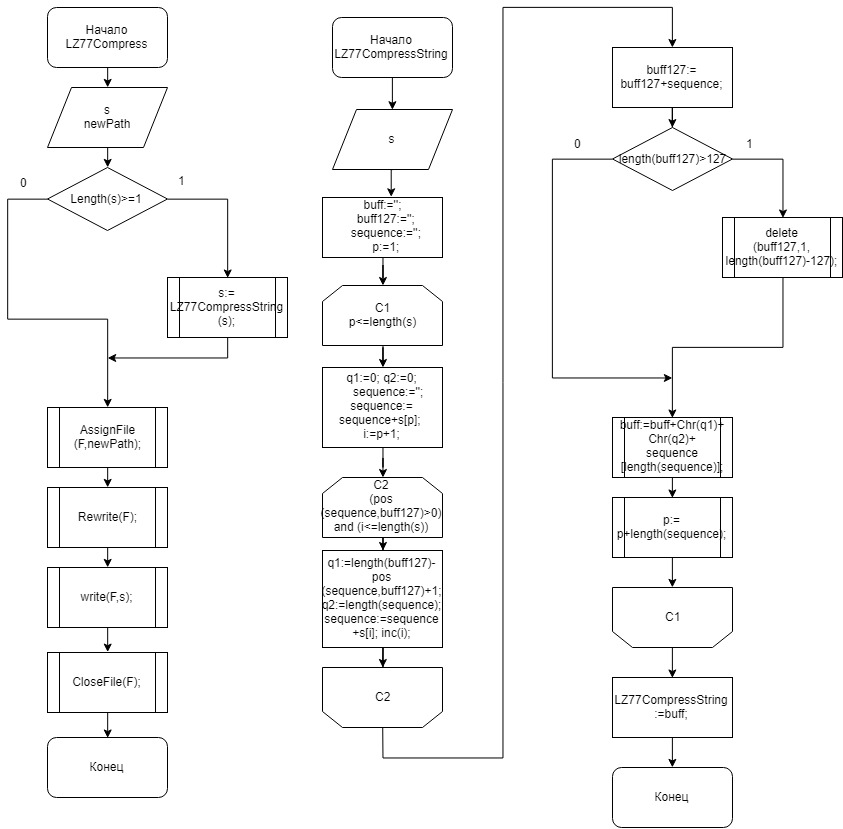
Приложение 7 (Преобразование строки на основе построенного дерева, продолжение)



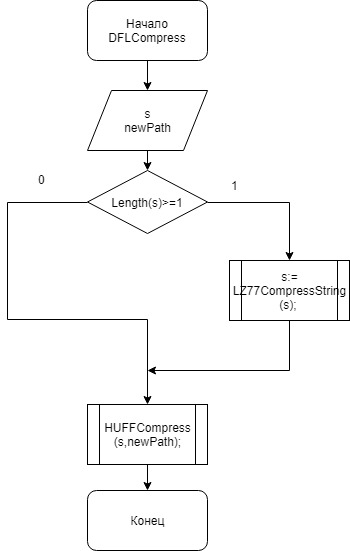
Приложение 8 (Очистка дерева)



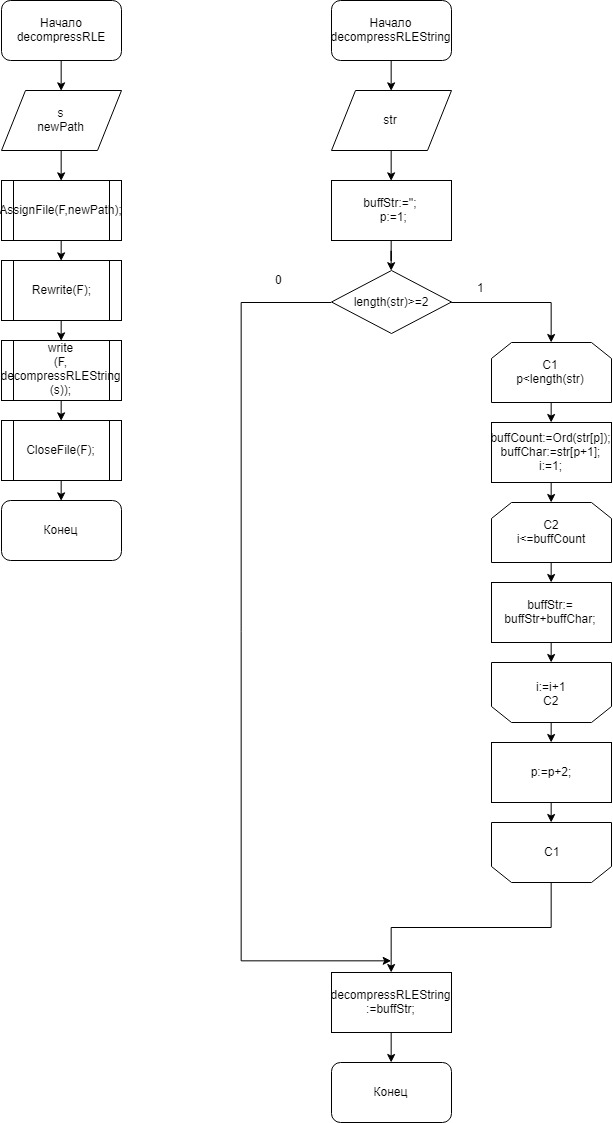
Приложение 9 (Алгоритм сжатия LZ77 )



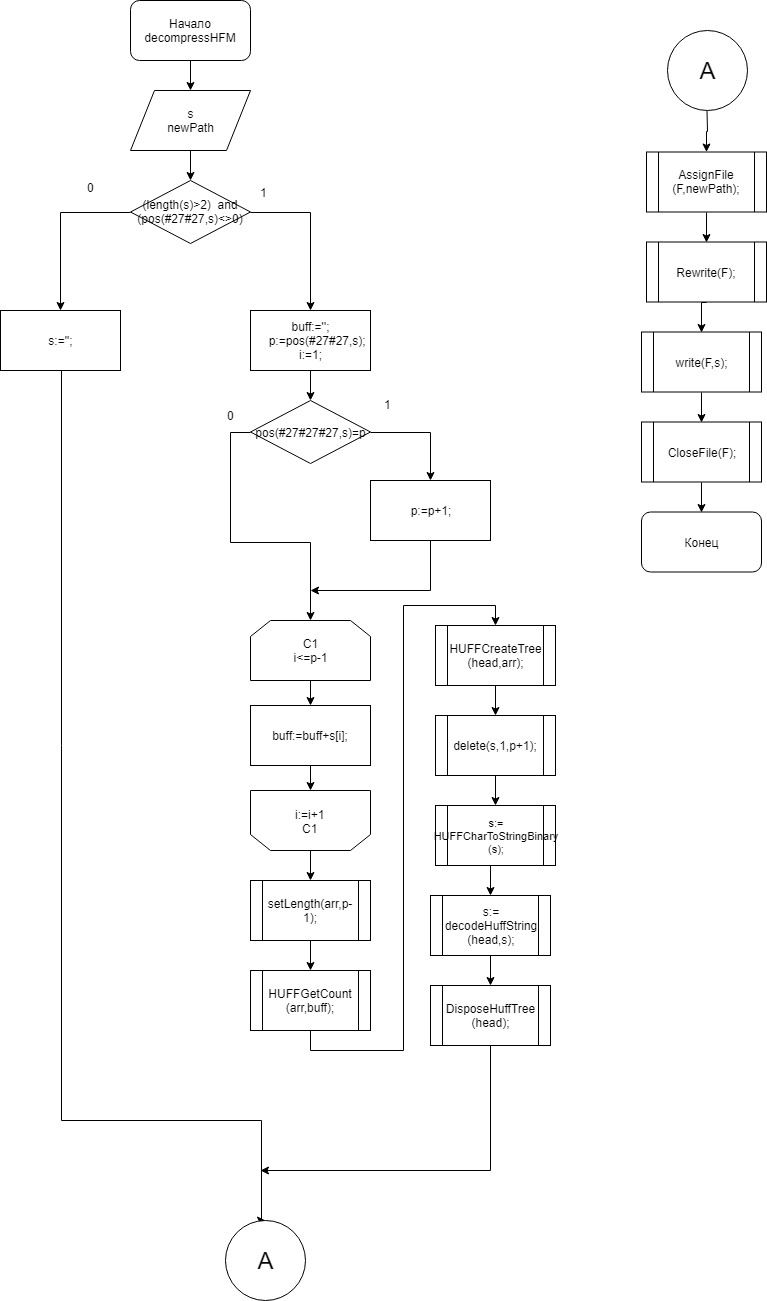
Приложение 10 (Алгоритм сжатия Deflate)



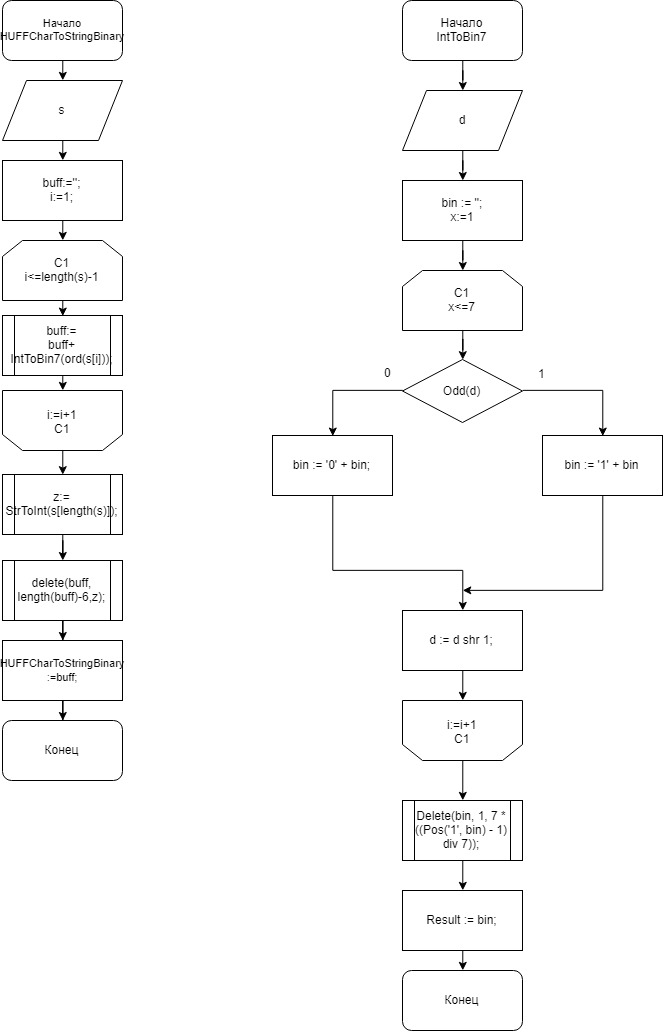
Приложение 11 (Алгоритм декомпрессии файла, сжатого RLE**)**

****

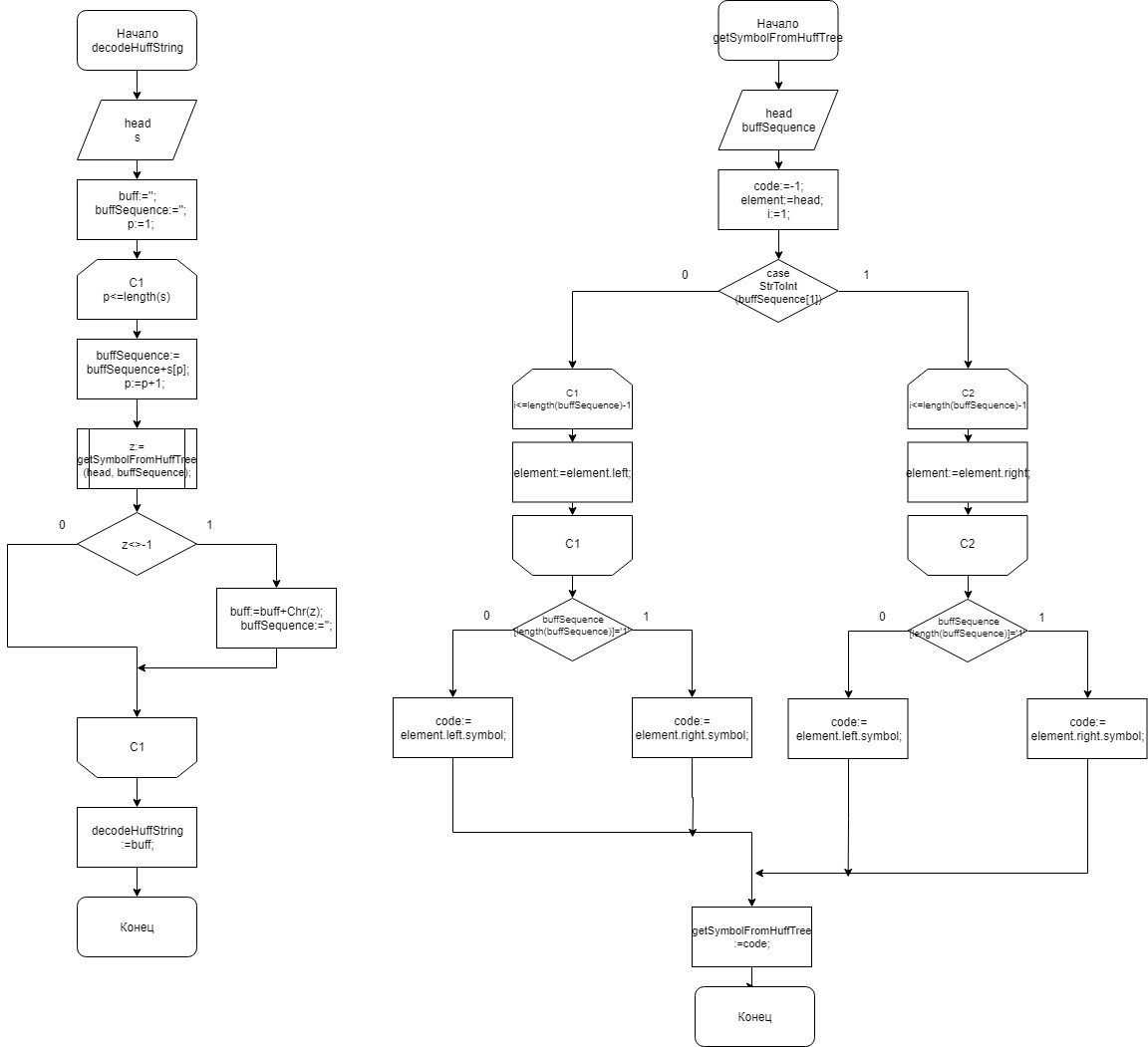
Приложение 12 (Алгоритм декомпрессии файла, сжатого Хаффманом**)**



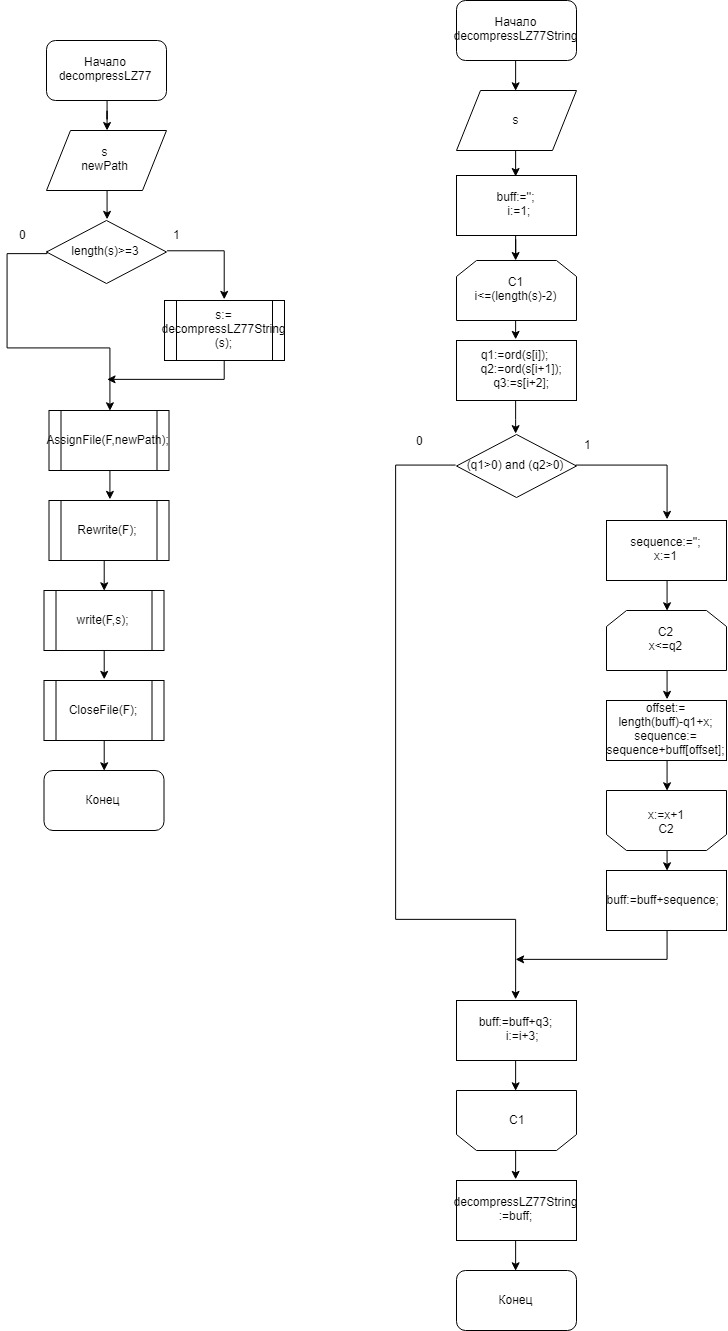
Приложение 12 (Алгоритм декомпрессии файла, сжатого Хаффманом, продолжение**)**

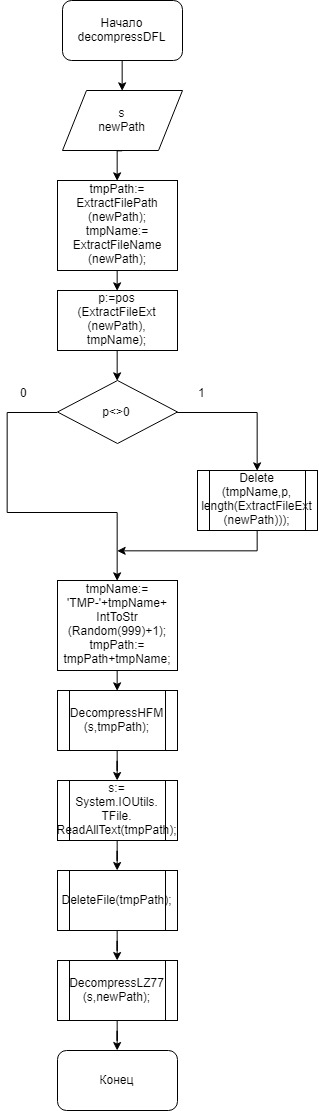


Приложение 12 (Алгоритм декомпрессии файла, сжатого Хаффманом, продолжение)

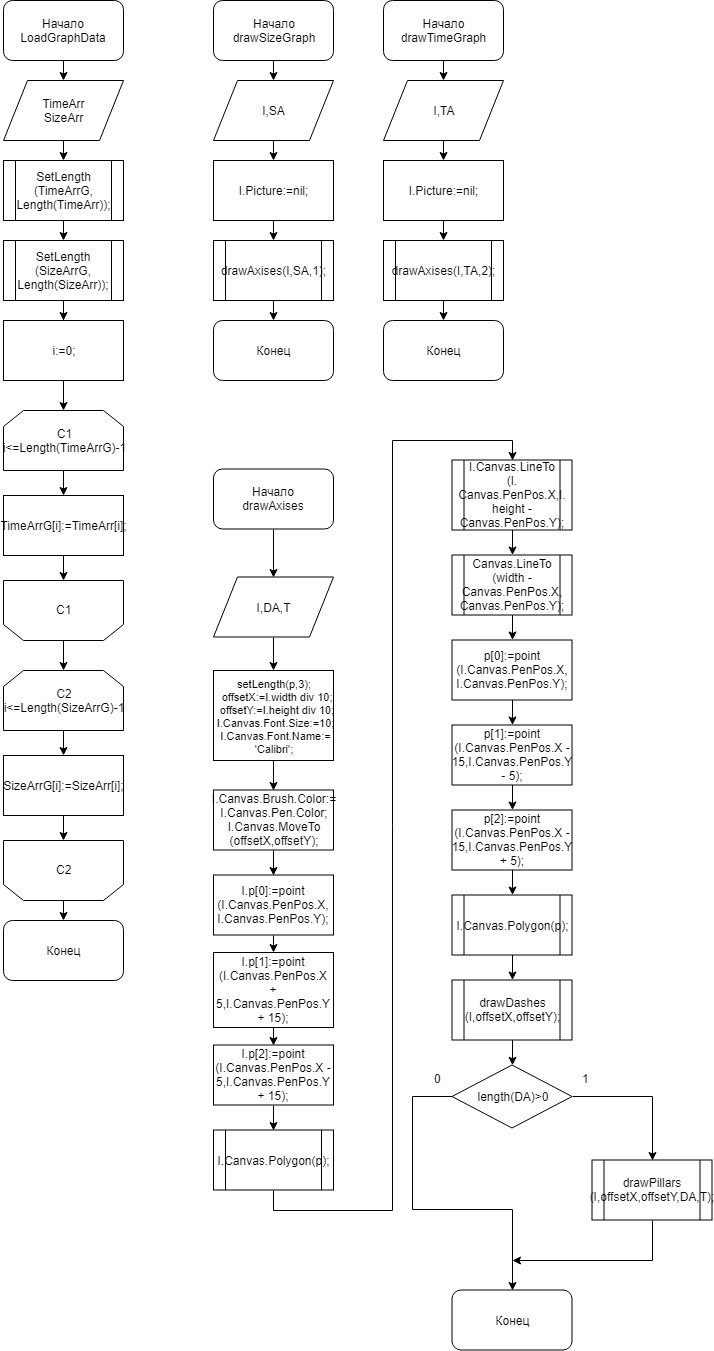


Приложение 13 (Алгоритм декомпрессии файла, сжатого LZ77**)**

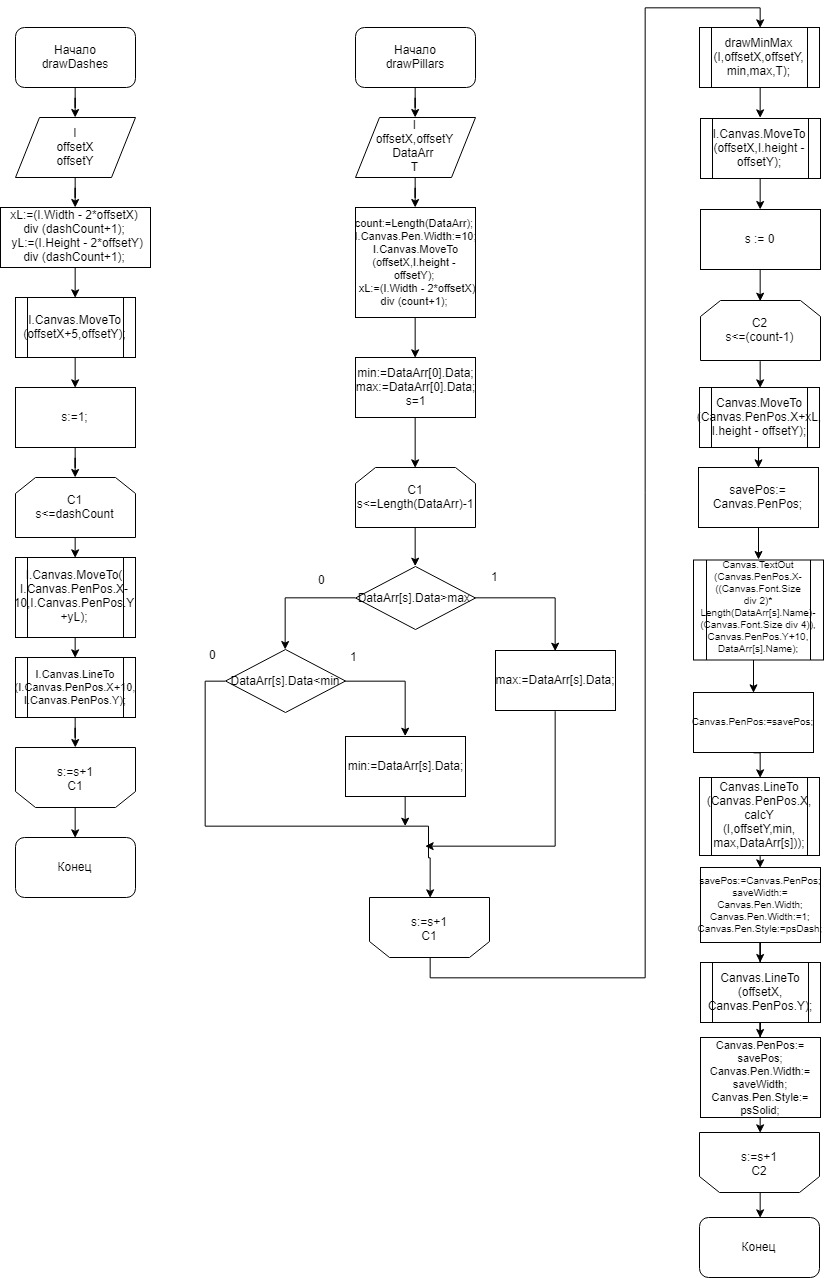


Приложение 14 (Алгоритм декомпрессии файла, сжатого Deflate**)**

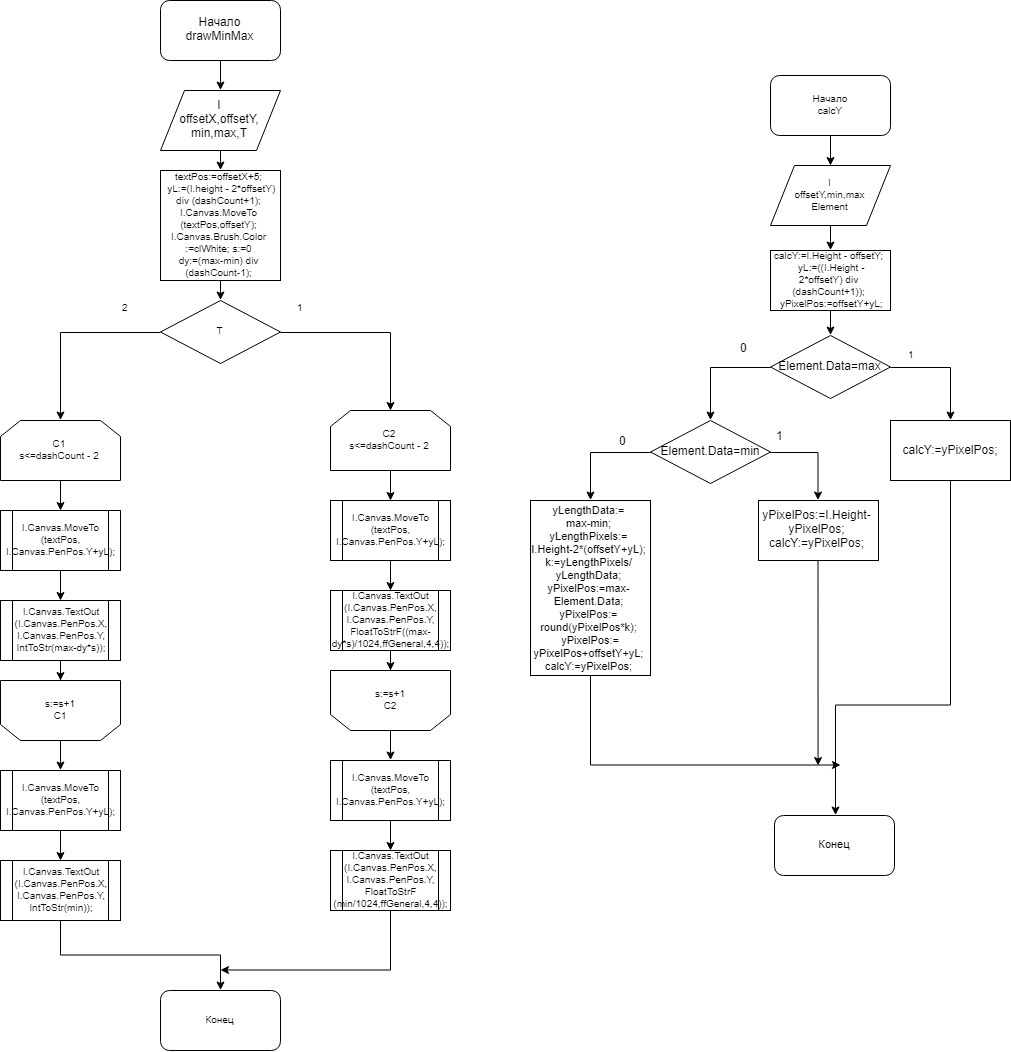
Приложение 15 (Построение графика)



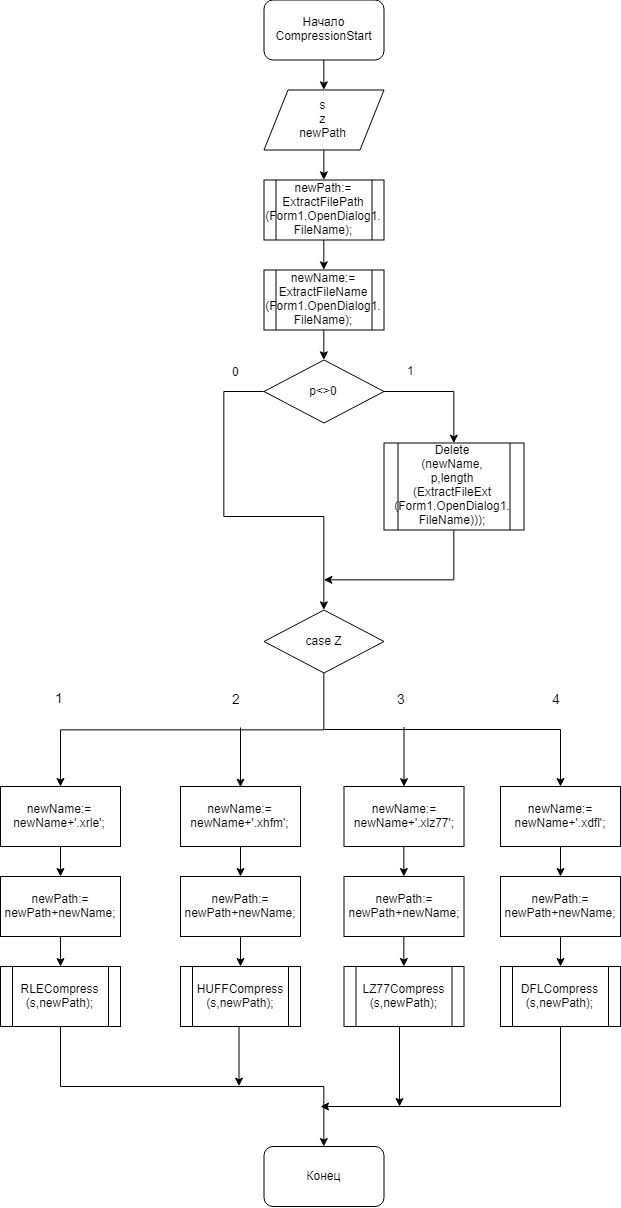
Приложение 15 (Построение графика, продолжение)



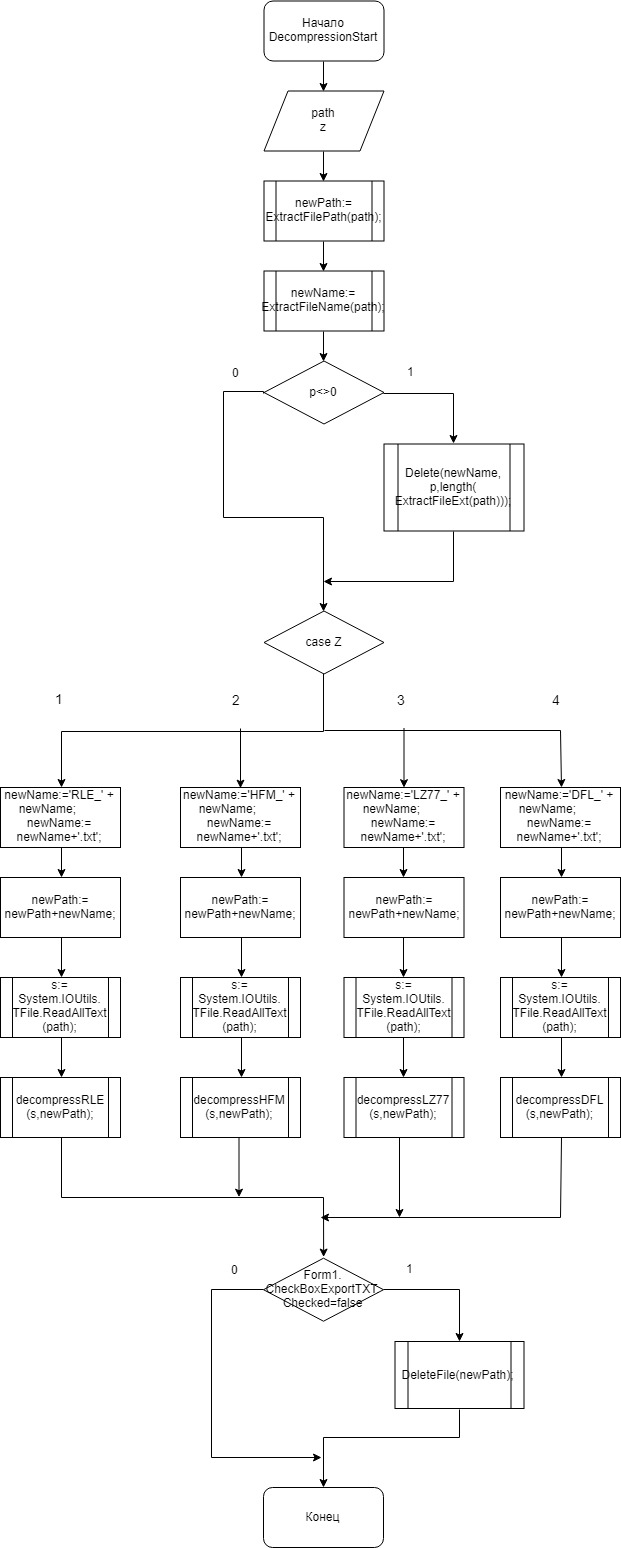
Приложение 15 (Построение графика, продолжение)



Приложение 16 (Процедура начала сжатия)



Приложение 17 (Процедура начала декомпрессии)



Приложение 18 (Код программы)

unit MainForm;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.ExtCtrls, Vcl.Menus,

System.ImageList, Vcl.ImgList, Vcl.Buttons, Math, System.IOUtils, System.Diagnostics, ChartForm;

const

outputFiles = false;

type

TFile = TextFile;

THuffArrayElement = Record

symbolCode: Integer;

count: Integer;

end;

THuffArray = Array of THuffArrayElement;

HUFFTreePointer = ^HUFFTreeNode;

HUFFTreeNode = Record

symbol: Integer;

left: HUFFTreePointer;

right: HUFFTreePointer;

end;

type

TForm1 = class(TForm)

Label1: TLabel;

GridPanelMain: TGridPanel;

OpenDialog1: TOpenDialog;

OpenButton: TSpeedButton;

Memo1: TMemo;

OpenLabel: TLabel;

LabelFilePreview: TLabel;

LabelFilePath: TLabel;

CompressionButton: TSpeedButton;

CheckBoxRLE: TCheckBox;

CheckBoxHUFF: TCheckBox;

CheckBoxA: TCheckBox;

DecompressionButton: TSpeedButton;

CheckBoxExportTXT: TCheckBox;

CheckBoxLZ77: TCheckBox;

CheckBoxDeflate: TCheckBox;

DTimeLabel: TLabel;

CheckBoxExpComp: TCheckBox;

procedure OpenButtonClick(Sender: TObject);

procedure FormCreate(Sender: TObject);

procedure Label1Click(Sender: TObject);

procedure CompressionButtonClick(Sender: TObject);

procedure DecompressionButtonClick(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

implementation

{$R \*.dfm}

function GetFileSize(FileName: String): Int64;

var FS: TFileStream;

begin

FS := TFileStream.Create(FileName, fmOpenRead);

Result := FS.Size;

FS.Free;

end;

//n-Full File Path

procedure printFileInfo(n:String);

begin

Writeln('File Info:');

writeln('Drive = '+ExtractFileDrive (n));

writeln('Catalog = '+ExtractFileDir (n));

writeln('Path = '+ExtractFilePath (n));

writeln('Name = '+ExtractFileName (n));

writeln('Extention = '+ExtractFileExt (n));

writeln('Size = '+IntToStr(GetFileSize(n)),' bytes');

end;

//-------RLE-------

function RLECompressString(str: String):String;

var

s,p: Integer;

buffChar: Char;

buffStr: String;

begin

buffStr:='';

p:=1;

if length(str)>=1 then

begin

while p<=length(str) do

begin

s:=0;

buffChar:=str[p];

while (p<=length(str)) and (str[p]=buffChar) and (s<>127) do

begin

s:=s+1;

p:=p+1;

end;

// write(s,buffChar);

buffStr:=buffStr+Chr(s);

buffStr:=buffStr+buffChar;

end;

end;

RLECompressString:=buffStr;

end;

procedure RLECompress(s:string; newPath:String);

var

F: TFile;

begin

AssignFile(F,newPath);

Rewrite(F);

write(F,RLECompressString(s));

CloseFile(F);

end;

//-----------------

//-----HUFFMAN-----

procedure DisposeHuffTree(var head: HuffTreePointer);

var

A:HuffTreePointer;

begin

if head<>nil then

begin

while head.left<>nil do

begin

A:=head.left.left;

if head.left.right<>nil then

begin

dispose(head.left.right.right);

dispose(head.left.right.left);

end;

dispose(head.left.right);

dispose(head.left);

head.left:=A;

end;

dispose(head.left);

while head.right<>nil do

begin

A:=head.right.right;

if head.right.left<>nil then

begin

dispose(head.right.left.left);

dispose(head.right.left.right);

end;

dispose(head.right.left);

dispose(head.right);

head.right:=A;

end;

dispose(head.right);

end;

dispose(head);

end;

function HUFFGetCount(var arr:THuffArray; str: String):integer;

var

n,q,seqL,z: Integer;

buffChar: char;

begin

// n:=-1;

// while length(str)>0 do

// begin

// n:=n+1;

// buffChar:=str[1];

// arr[n].symbolCode:=ord(buffChar);

// while pos(buffChar,str)<>0 do

// begin

// q:=pos(buffChar,str);

// seqL:=0;

// while (q<=length(str)) and (str[q]=buffChar) do

// begin

// seqL:=seqL+1;

// q:=q+1;

// end;

//

// arr[n].count:=arr[n].count+seqL;

// delete(str, pos(buffChar,str), seqL);

// end;

//

// end;

// HUFFGetCount:=n+1;

n:=-1;

while length(str)>0 do

begin

n:=n+1;

buffChar:=str[1];

arr[n].symbolCode:=ord(buffChar);

z:=pos(buffChar,str);

if z<>0 then

begin

seqL:=0;

for q := z to length(str) do

begin

if str[q]=buffChar then

begin

seqL:=seqL+1;

end;

end;

arr[n].count:=arr[n].count+seqL;

str:=StringReplace(str,buffChar,'',[rfReplaceAll]);

end;

end;

HUFFGetCount:=n+1;

end;

procedure HUFFSort(var arr:THuffArray);

var

i: integer;

save:THuffArrayElement;

check:Boolean;

begin

check:=false;

while check=false do

begin

check:=true;

for i := 0 to length(arr)-2 do

begin

if arr[i].count<arr[i+1].count then

begin

save:=arr[i];

arr[i]:=arr[i+1];

arr[i+1]:=save;

check:=false;

end;

end;

end;

end;

procedure HUFFCreateTree(var head: HuffTreePointer; arr:THuffArray);

var

elementL,elementR: HuffTreePointer;

n:integer;

begin

n:=0;

new(head);

head.symbol:=-1;

head.left:=nil;

head.right:=nil;

elementL:=head;

elementR:=head;

while n<length(arr)-2 do

begin

new(elementL.left);

elementL:=elementL.left;

elementL.symbol:=-1;

elementL.left:=nil;

new(elementL.right);

elementL.right.right:=nil;

elementL.right.left:=nil;

elementL.right.symbol:=arr[n].symbolCode;

n:=n+1;

if n<length(arr)-2 then

begin

new(elementR.right);

elementR:=elementR.right;

elementR.symbol:=-1;

elementR.right:=nil;

new(elementR.left);

elementR.left.right:=nil;

elementR.left.left:=nil;

elementR.left.symbol:=arr[n].symbolCode;

n:=n+1;

end;

end;

if length(arr)>0 then

begin

new(elementR.right);

elementR:=elementR.right;

elementR.right:=nil;

elementR.left:=nil;

elementR.symbol:=arr[n].symbolCode;

if length(arr)>1 then

begin

new(elementL.left);

elementL:=elementL.left;

elementL.right:=nil;

elementL.left:=nil;

elementL.symbol:=arr[n+1].symbolCode;

end;

end;

end;

procedure OutputTree(head:HuffTreePointer);

var

element: HuffTreePointer;

check: Boolean;

begin

writeln('Left Branch');

element:=head;

check:=true;

while check=true do

begin

check:=false;

if element.left<>nil then

begin

element:=element.left;

if element.right<>nil then

begin

write(chr(element.right.symbol));

check:=true;

end else

begin

writeln(chr(element.symbol));

end;

end;

end;

writeln('Right Branch');

element:=head;

check:=true;

while check=true do

begin

check:=false;

if element.right<>nil then

begin

element:=element.right;

if element.left<>nil then

begin

write(chr(element.left.symbol));

check:=true;

end else

begin

writeln(chr(element.symbol));

end;

end;

end;

writeln;

end;

function HUFFGetBSymbol(head:HuffTreePointer; c:Char):String;

var

buff,buffL,buffR: String;

elementL, elementR: HuffTreePointer;

check: Boolean;

begin

buff:='';

buffL:='';

buffR:='';

elementL:=head;

elementR:=head;

check:=false;

while check=false do

begin

if elementL.left<>nil then

begin

elementL:=elementL.left;

buffL:=buffL+'0';

if elementL.symbol=ord(c) then

begin

buff:=buffL;

check:=true;

end else

if elementL.right<>nil then

begin

if elementL.right.symbol=ord(c) then

begin

buffL:=buffL+'1';

buff:=buffL;

check:=true;

end;

end;

end;

if check<>true then

begin

if elementR.right<>nil then

begin

elementR:=elementR.right;

buffR:=buffR+'1';

if elementR.symbol=ord(c) then

begin

buff:=buffR;

check:=true;

end else

if elementR.left<>nil then

begin

if elementR.left.symbol=ord(c) then

begin

buffR:=buffR+'0';

buff:=buffR;

check:=true;

end;

end;

end;

end;

end;

// writeln(c,'\_',buff);

HUFFGetBSymbol:=buff;

end;

//function StrBinToInt(s: String; l:integer):Integer;

//var

// i: integer;

// value: integer;

//begin

// value:=0;

// for i:=1 to l do

// begin

// if s[i]='1' then

// begin

//// value:=value+round(exp(abs(i-8)\*ln(2)));

// value:=value+round(power(2,abs(i-l)));

// end;

// end;

//// write(' ',value);

// StrBinToInt:=value;

//end;

//function HUFFStringBinaryToChar(s:String):String;

//var

// buff:String;

// lastLength: Integer;

//begin

// buff:='';

//

// while length(s)>7 do

// begin

// buff:=buff+chr(StrBinToInt(s,7));

// delete(s,1,7);

// end;

//// writeln;

// {

// Next part adds (lastLength) additional 0 to the beginning of the next sequence

// }

// if (length(s)>0) and (length(s)<=7) then

// begin

// lastLength:=length(s);

// if lastLength<>0 then

// begin

// buff:=buff+chr(StrBinToInt(s,lastLength));

// end;

// buff:=buff+IntToStr(7-lastLength);

// end;

//

//

// HUFFStringBinaryToChar:=buff;

//end;

function StrBinToInt(s: String; pos,l:integer):Integer;

var

i: integer;

value: integer;

begin

value:=0;

l:=l+(pos-1);

for i:=pos to l do

begin

if s[i]='1' then

begin

value:=value+round(power(2,abs(i-l)));

end;

end;

StrBinToInt:=value;

end;

function HUFFStringBinaryToChar(s:String):String;

var

buff:String;

lastLength,p: Integer;

begin

buff:='';

p:=1;

while p<=(length(s)-7) do

begin

buff:=buff+chr(StrBinToInt(s,p,7));

p:=p+7;

end;

{

Next part adds (lastLength) additional 0 to the beginning of the next sequence

}

delete(s,1,p-1);

if (length(s)>0) and (length(s)<=7) then

begin

lastLength:=length(s);

if lastLength<>0 then

begin

buff:=buff+chr(StrBinToInt(s,1,lastLength));

end;

buff:=buff+IntToStr(7-lastLength);

end;

HUFFStringBinaryToChar:=buff;

end;

function HUFFCompressString(head:HuffTreePointer; s: String):String;

var

i: Integer;

encodedStr: String;

begin

encodedStr:='';

for i:=1 to length(s) do

begin

encodedStr:=encodedStr+HUFFGetBSymbol(head, s[i]);

end;

//writeln(encodedStr);

encodedStr:=HUFFStringBinaryToChar(encodedStr);

HUFFCompressString:=encodedStr;

end;

procedure HUFFCompress(s:String; newPath:String);

var

i,n: Integer;

F: TFile;

arr: THuffArray;

treeHead: HuffTreePointer;

begin

SetLength(arr,1024);

n:=HUFFGetCount(arr,s);

SetLength(arr,n);

HUFFSort(arr);

HUFFCreateTree(treeHead,arr);

//OutputTree(treeHead);

s:=HUFFCompressString(treeHead, s);

AssignFile(F,newPath);

Rewrite(F);

//HuffTable in file

for i := 0 to length(arr)-1 do

begin

write(F,Chr(arr[i].symbolCode));

end;

write(F,#27,#27);

//

write(F,s);

CloseFile(F);

Finalize(arr);

DisposeHuffTree(treeHead);

end;

//-----------------

//-------LZ77-----

function LZ77CompressString(s:String):String;

var

i,p,q1,q2:integer;

buff,buff127,sequence:String;

begin

buff:='';

buff127:='';

sequence:='';

p:=1;

while p<=length(s) do

begin

q1:=0;

q2:=0;

sequence:='';

sequence:=sequence+s[p];

i:=p+1;

while (pos(sequence,buff127)>0) and (i<=length(s)) do

begin

q1:=length(buff127)-pos(sequence,buff127)+1;

q2:=length(sequence);

sequence:=sequence+s[i];

inc(i);

end;

buff127:=buff127+sequence;

if length(buff127)>127 then

begin

delete(buff127,1,length(buff127)-127);

end;

buff:=buff+Chr(q1)+Chr(q2)+sequence[length(sequence)];

// buff:=buff+IntToStr(q1)+'/'+IntToStr(q2)+'/'+sequence[length(sequence)]+' ';

// delete(s,1,length(sequence));

p:=p+length(sequence);

end;

LZ77CompressString:=buff;

end;

procedure LZ77Compress(s:String; newPath:String);

var

F: TFile;

begin

if Length(s)>=1 then

begin

s:=LZ77CompressString(s);

end;

AssignFile(F,newPath);

Rewrite(F);

write(F,s);

CloseFile(F);

end;

//-----------------

//-----DEFLATE-----

procedure DFLCompress(s:String; newPath:String);

begin

if Length(s)>=1 then

begin

s:=LZ77CompressString(s);

end;

HUFFCompress(s,newPath);

end;

//-----------------

procedure CompressionStart(s:string;z:integer;var newPath,newName:String);

var

p: Integer;

begin

newPath:=ExtractFilePath(Form1.OpenDialog1.FileName);

newName:=ExtractFileName(Form1.OpenDialog1.FileName);

p:=pos(ExtractFileExt(Form1.OpenDialog1.FileName),newName);

if p<>0 then

begin

Delete(newName,p,length(ExtractFileExt(Form1.OpenDialog1.FileName)));

end;

case z of

1:

begin

newName:=newName+'.xrle';

newPath:=newPath+newName;

RLECompress(s,newPath);

write('RLE ');

end;

2:

begin

newName:=newName+'.xhfm';

newPath:=newPath+newName;

HUFFCompress(s,newPath);

write('Huffman ');

end;

3:

begin

newName:=newName+'.xlz77';

newPath:=newPath+newName;

LZ77Compress(s,newPath);

write('LZ77 ');

end;

4:

begin

newName:=newName+'.xdfl';

newPath:=newPath+newName;

DFLCompress(s,newPath);

write('Deflate ');

end;

end;

s:=System.IOUtils.TFile.ReadAllText(newPath);

writeln('Compression Result:');

if outputFiles then

begin

writeln(s);

end;

writeln;

printFileInfo(newPath);

writeln;

writeln;

end;

procedure TForm1.CompressionButtonClick(Sender: TObject);

var

s: String;

newPath,newName: String;

stopWatch: TStopWatch;

TA,SA:ChartForm.TGraphDataArray;

begin

stopWatch.Create;

SetLength(SA,1);

SetLength(TA,1);

SA[0].Name:='Uncomp.';

SA[0].Data:=GetFileSize(Self.OpenDialog1.FileName);

TA[0].Name:='Uncomp.';

TA[0].Data:=0;

s:=System.IOUtils.TFile.ReadAllText(Self.OpenDialog1.FileName);

AllocConsole;

writeln('Uncompressed text:');

if outputFiles then

begin

writeln(s);

end;

writeln;

printFileInfo(Self.OpenDialog1.FileName);

writeln;

writeln;

if Self.CheckBoxRLE.Checked then

begin

SetLength(SA,Length(SA)+1);

SetLength(TA,Length(TA)+1);

stopWatch.Reset;

stopWatch.Start;

CompressionStart(s,1,newPath,newName);

SA[Length(TA)-1].Name:='RLE';

SA[Length(TA)-1].Data:=GetFileSize(newPath);

TA[Length(TA)-1].Name:='RLE';

TA[Length(TA)-1].Data:=stopWatch.ElapsedMilliseconds;

stopWatch.Stop;

if Self.CheckBoxExpComp.Checked=false then

begin

DeleteFile(NewPath);

end;

end;

if Self.CheckBoxHUFF.Checked then

begin

SetLength(SA,Length(SA)+1);

SetLength(TA,Length(TA)+1);

SA[Length(TA)-1].Name:='HUFFMAN';

TA[Length(TA)-1].Name:='HUFFMAN';

stopWatch.Reset;

stopWatch.Start;

CompressionStart(s,2,newPath,newName);

SA[Length(TA)-1].Data:=GetFileSize(newPath);

TA[Length(TA)-1].Data:=stopWatch.ElapsedMilliseconds;

stopWatch.Stop;

if Self.CheckBoxExpComp.Checked=false then

begin

DeleteFile(NewPath);

end;

end;

if Self.CheckBoxLZ77.Checked then

begin

SetLength(SA,Length(SA)+1);

SetLength(TA,Length(TA)+1);

SA[Length(TA)-1].Name:='LZ77';

TA[Length(TA)-1].Name:='LZ77';

stopWatch.Reset;

stopWatch.Start;

CompressionStart(s,3,newPath,newName);

SA[Length(TA)-1].Data:=GetFileSize(newPath);

TA[Length(TA)-1].Data:=stopWatch.ElapsedMilliseconds;

stopWatch.Stop;

if Self.CheckBoxExpComp.Checked=false then

begin

DeleteFile(NewPath);

end;

end;

if Self.CheckBoxDeflate.Checked then

begin

SetLength(SA,Length(SA)+1);

SetLength(TA,Length(TA)+1);

SA[Length(TA)-1].Name:='Deflate';

TA[Length(TA)-1].Name:='Deflate';

stopWatch.Reset;

stopWatch.Start;

CompressionStart(s,4,newPath,newName);

SA[Length(TA)-1].Data:=GetFileSize(newPath);

TA[Length(TA)-1].Data:=stopWatch.ElapsedMilliseconds;

stopWatch.Stop;

if Self.CheckBoxExpComp.Checked=false then

begin

DeleteFile(NewPath);

end;

end;

if Self.CheckBoxA.Checked then

begin

ChartForm.LoadGraphData(TA,SA);

if ChartForm.chForm.Showing=false then

begin

ChartForm.chForm.Show;

end else

begin

ChartForm.chForm.Refresh;

end;

end;

//FreeConsole;

end;

//-------RLE-------

function decompressRLEString(str:String):String;

var

i,p: Integer;

buffChar: Char;

buffCount: Integer;

buffStr: String;

begin

buffStr:='';

p:=1;

if length(str)>=2 then

begin

while p<length(str) do

begin

buffCount:=Ord(str[p]);

buffChar:=str[p+1];

for i:=1 to buffCount do

begin

buffStr:=buffStr+buffChar;

end;

p:=p+2;

end;

end;

decompressRLEString:=buffStr;

end;

procedure decompressRLE(s,newPath:String);

var

F:TFile;

begin

AssignFile(F,newPath);

Rewrite(F);

write(F,decompressRLEString(s));

CloseFile(F);

end;

//-----------------

//-----HUFFMAN-----

function IntToBin7(d: Integer): string;

var

x: Integer;

bin: string;

begin

bin := '';

for x := 1 to 7 do

begin

if Odd(d) then

begin

bin := '1' + bin

end else

begin

bin := '0' + bin;

end;

d := d shr 1;

end;

Delete(bin, 1, 7 \* ((Pos('1', bin) - 1) div 7));

Result := bin;

end;

function HUFFCharToStringBinary(s:string):String;

var

buff:String;

i,z:integer;

begin

buff:='';

for i:=1 to length(s)-1 do

begin

buff:=buff+IntToBin7(ord(s[i]));

end;

z:=StrToInt(s[length(s)]);

delete(buff,length(buff)-6,z);

HUFFCharToStringBinary:=buff;

end;

function getSymbolFromHuffTree(head:HuffTreePointer; buffSequence:string):Integer;

var

code,i:integer;

element:HuffTreePointer;

begin

code:=-1;

element:=head;

case StrToInt(buffSequence[1]) of

0:

begin

for i:=1 to length(buffSequence)-1 do

begin

element:=element.left;

end;

if buffSequence[length(buffSequence)]='1' then

begin

code:=element.right.symbol;

end else

begin

code:=element.left.symbol;

end;

end;

1:

begin

for i:=1 to length(buffSequence)-1 do

begin

element:=element.right;

end;

if buffSequence[length(buffSequence)]='1' then

begin

code:=element.right.symbol;

end else

begin

code:=element.left.symbol;

end;

end;

end;

getSymbolFromHuffTree:=code;

end;

function decodeHuffString(head:HuffTreePointer; s:string):String;

var

buff, buffSequence:string;

z,p:integer;

begin

buff:='';

buffSequence:='';

p:=1;

while p<=length(s) do

begin

buffSequence:=buffSequence+s[p];

p:=p+1;

z:=getSymbolFromHuffTree(head, buffSequence);

if z<>-1 then

begin

buff:=buff+Chr(z);

buffSequence:='';

end;

end;

decodeHuffString:=buff;

end;

procedure decompressHFM(s:string;newPath:String);

var

F:TFile;

i,p:integer;

arr:THuffArray;

head:HUFFTreePointer;

buff:String;

begin

if (length(s)>2) and (pos(#27#27,s)<>0) then

begin

buff:='';

p:=pos(#27#27,s);

if pos(#27#27#27,s)=p then

begin

p:=p+1;

end;

for i:=1 to p-1 do

begin

buff:=buff+s[i];

end;

setLength(arr,p-1);

HUFFGetCount(arr,buff);

HUFFCreateTree(head,arr);

//OutputTree(head);

delete(s,1,p+1);

s:=HUFFCharToStringBinary(s);

s:=decodeHuffString(head,s);

DisposeHuffTree(head);

end else

begin

s:='';

end;

Finalize(arr);

AssignFile(F,newPath);

Rewrite(F);

write(F,s);

CloseFile(F);

end;

//-----------------

//------LZ77-------

function decompressLZ77String(s:string):String;

var

q1,q2,i,x,offset:Integer;

q3:Char;

buff,sequence:String;

begin

buff:='';

i:=1;

while i<=(length(s)-2) do

begin

q1:=ord(s[i]);

q2:=ord(s[i+1]);

q3:=s[i+2];

if (q1>0) and (q2>0) then

begin

sequence:='';

for x:=1 to q2 do

begin

offset:=length(buff)-q1+x;

sequence:=sequence+buff[offset];

end;

buff:=buff+sequence;

end;

buff:=buff+q3;

i:=i+3;

end;

decompressLZ77String:=buff;

end;

procedure decompressLZ77(s:string;newPath:String);

var

F:TFile;

begin

if length(s)>=3 then

begin

s:=decompressLZ77String(s);

end;

AssignFile(F,newPath);

Rewrite(F);

write(F,s);

CloseFile(F);

end;

//-----------------

//-----DEFLATE-----

procedure decompressDFL(s:string;newPath:String);

var

tmpPath,tmpName:String;

p:Integer;

begin

tmpPath:=ExtractFilePath(newPath);

tmpName:=ExtractFileName(newPath);

p:=pos(ExtractFileExt(newPath),tmpName);

if p<>0 then

begin

Delete(tmpName,p,length(ExtractFileExt(newPath)));

end;

tmpName:='TMP-'+tmpName+IntToStr(Random(999)+1);

tmpPath:=tmpPath+tmpName;

DecompressHFM(s,tmpPath);

s:=System.IOUtils.TFile.ReadAllText(tmpPath);

DeleteFile(tmpPath);

DecompressLZ77(s,newPath);

end;

//-----------------

procedure StartDecompression(path:String;z:integer);

var

p: Integer;

s:string;

newPath,newName: String;

begin

AllocConsole;

Form1.Memo1.Clear;

newPath:=ExtractFilePath(path);

newName:=ExtractFileName(path);

p:=pos(ExtractFileExt(path),newName);

if p<>0 then

begin

Delete(newName,p,length(ExtractFileExt(path)));

end;

case z of

1:

begin

newName:='RLE\_' + newName;

newName:=newName+'.txt';

newPath:=newPath+newName;

s:=System.IOUtils.TFile.ReadAllText(path);

decompressRLE(s,newPath);

write('RLE ');

end;

2:

begin

newName:='HFM\_' + newName;

newName:=newName+'.txt';

newPath:=newPath+newName;

s:=System.IOUtils.TFile.ReadAllText(path);

decompressHFM(s,newPath);

write('Huffman ');

end;

3:

begin

newName:='LZ77\_' + newName;

newName:=newName+'.txt';

newPath:=newPath+newName;

s:=System.IOUtils.TFile.ReadAllText(path);

decompressLZ77(s,newPath);

write('LZ77 ');

end;

4:

begin

newName:='DFL\_' + newName;

newName:=newName+'.txt';

newPath:=newPath+newName;

s:=System.IOUtils.TFile.ReadAllText(path);

decompressDFL(s,newPath);

write('Deflate ');

end;

end;

s:=System.IOUtils.TFile.ReadAllText(newPath);

writeln('Decompression Result:');

if outputFiles then

begin

writeln(s);

end;

Form1.Memo1.Text:=s;

writeln;

printFileInfo(newPath);

writeln;

writeln;

if Form1.CheckBoxExportTXT.Checked=false then

begin

DeleteFile(newPath);

end;

end;

procedure TForm1.DecompressionButtonClick(Sender: TObject);

var

stopWatch: TStopWatch;

begin

stopWatch.Create;

stopWatch.Reset;

stopWatch.Start;

if (Self.OpenDialog1.FileName<>'') then

begin

if ExtractFileExt(Form1.OpenDialog1.FileName)='.xrle' then

begin

StartDecompression(Self.OpenDialog1.FileName,1);

end else

if ExtractFileExt(Form1.OpenDialog1.FileName)='.xhfm' then

begin

StartDecompression(Self.OpenDialog1.FileName,2);

end else

if ExtractFileExt(Form1.OpenDialog1.FileName)='.xlz77' then

begin

StartDecompression(Self.OpenDialog1.FileName,3);

end else

if ExtractFileExt(Form1.OpenDialog1.FileName)='.xdfl' then

begin

StartDecompression(Self.OpenDialog1.FileName,4);

end;

end;

Self.DTimeLabel.Caption:='D.Time: '+IntToStr(stopWatch.ElapsedMilliseconds)+' ms';

stopWatch.Stop;

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

Self.GridPanelMain.Color:=clWebSeashell;

Self.OpenLabel.Caption:='File isn''t open';

openDialog1.InitialDir := GetCurrentDir;

Self.Memo1.Color:=clWebLavender;

Self.Memo1.Text:='';

Self.LabelFilePath.Hide;

Self.LabelFilePreview.Caption:='File preview';

Self.CheckBoxRLE.Enabled:=False;

Self.CheckBoxHUFF.Enabled:=False;

Self.CheckBoxLZ77.Enabled:=False;

Self.CheckBoxExpComp.Enabled:=False;

Self.CheckBoxA.Enabled:=False;

Self.CheckBoxExportTXT.Enabled:=False;

Self.CompressionButton.Enabled:=False;

Self.DecompressionButton.Enabled:=False;

Self.DTimeLabel.Enabled:=False;

end;

procedure TForm1.OpenButtonClick(Sender: TObject);

var

s:integer;

begin

if OpenDialog1.Execute() then

begin

if ChartForm.chForm.Showing=true then

begin

ChartForm.chForm.Hide;

end;

Self.Memo1.Lines.LoadFromFile(Self.OpenDialog1.FileName);

Self.OpenButton.Caption:='Open another file';

Self.OpenLabel.Hide;

Self.LabelFilePath.Show;

Self.LabelFilePath.Caption:='File path: ' + OpenDialog1.FileName;

s:=0;

if ExtractFileExt(Self.OpenDialog1.FileName)='.xrle' then

begin

s:=1;

end else

if ExtractFileExt(Self.OpenDialog1.FileName)='.xhfm' then

begin

s:=2;

end else

if ExtractFileExt(Self.OpenDialog1.FileName)='.xlz77' then

begin

s:=3;

end else

if ExtractFileExt(Self.OpenDialog1.FileName)='.xdfl' then

begin

s:=4;

end;

if s=0 then

begin

Self.DecompressionButton.Enabled:=False;

Self.DecompressionButton.Caption:='Decompress';

Self.CompressionButton.Enabled:=True;

Self.CheckBoxRLE.Enabled:=True;

Self.CheckBoxHUFF.Enabled:=True;

Self.CheckBoxLZ77.Enabled:=True;

Self.CheckBoxDeflate.Enabled:=True;

Self.CheckBoxA.Enabled:=True;

Self.CheckBoxExpComp.Enabled:=True;

Self.CheckBoxExportTXT.Enabled:=False;

Self.DTimeLabel.Caption:='D.Time: 0 ms';

Self.DTimeLabel.Enabled:=False;

end else

begin

Self.DecompressionButton.Enabled:=True;

case s of

1:

begin

Self.DecompressionButton.Caption:='Decompress(RLE)';

end;

2:

begin

Self.DecompressionButton.Caption:='Decompress(Huffman)';

end;

3:

begin

Self.DecompressionButton.Caption:='Decompress(LZ77)';

end;

4:

begin

Self.DecompressionButton.Caption:='Decompress(Deflate)';

end;

end;

Self.CompressionButton.Enabled:=False;

Self.CheckBoxRLE.Enabled:=False;

Self.CheckBoxHUFF.Enabled:=False;

Self.CheckBoxLZ77.Enabled:=False;

Self.CheckBoxDeflate.Enabled:=False;

Self.CheckBoxA.Enabled:=False;

Self.CheckBoxExpComp.Enabled:=False;

Self.CheckBoxExportTXT.Enabled:=True;

Self.DTimeLabel.Caption:='D.Time: 0 ms';

Self.DTimeLabel.Enabled:=True;

end;

end;

end;

procedure TForm1.Label1Click(Sender: TObject);

var

TA,SA:ChartForm.TGraphDataArray;

i:Integer;

begin

if Self.Label1.Caption='(c)Mikhail Kavaleuski ' then

begin

Self.Label1.Caption:='^\_^';

end else

begin

Self.Label1.Caption:='(c)Mikhail Kavaleuski '

end;

SetLength(SA,5);

SetLength(TA,5);

TA[0].Name:='-';

TA[0].Data:=0;

SA[0].Name:='-';

SA[0].Data:=1024\*50;

for i := 1 to Length(TA)-1 do

begin

TA[i].Name:='Anime';

TA[i].Data:=1000\*i;

SA[i].Name:=IntToStr(i);

SA[i].Data:=1024\*5\*i;

end;

ChartForm.LoadGraphData(TA,SA);

if ChartForm.chForm.Showing=false then

begin

ChartForm.chForm.Show;

end else

begin

ChartForm.chForm.Refresh;

end;

end;

end.

unit ChartForm;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.WinXCtrls,

Vcl.StdCtrls;

const

dashCount = 7;

type

TAlgRec = Record

Name: String;

Data: Integer;

end;

TGraphDataArray = Array of TAlgRec;

// TSizeGraphArray = Array of TAlgRec;

type

TForm2 = class(TForm)

GridPanel1: TGridPanel;

Label1: TLabel;

TimeGraphImage: TImage;

SizeGraphImage: TImage;

SizeLabel: TLabel;

TimeLabel: TLabel;

procedure FormCreate(Sender: TObject);

procedure FormResize(Sender: TObject);

procedure FormShow(Sender: TObject);

procedure FormPaint(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

procedure LoadGraphData(TimeArr:TGraphDataArray;SizeArr:TGraphDataArray);

var

chForm: TForm2;

implementation

{$R \*.dfm}

procedure drawDashes(I:TImage;offsetX,offsetY:Integer);

var

xL,yL,s: Integer;

begin

xL:=(I.Width - 2\*offsetX) div (dashCount+1);

yL:=(I.Height - 2\*offsetY) div (dashCount+1);

with I do

begin

Canvas.MoveTo(offsetX+5,offsetY);

for s := 1 to dashCount do

begin

Canvas.MoveTo(Canvas.PenPos.X-10,Canvas.PenPos.Y+yL);

Canvas.LineTo(Canvas.PenPos.X+10,Canvas.PenPos.Y);

end;

// Canvas.MoveTo(Width - offsetX,Height - offsetY + 5);

// for s := 1 to dashCount do

// begin

// Canvas.MoveTo(Canvas.PenPos.X-xL,Canvas.PenPos.Y-10);

// Canvas.LineTo(Canvas.PenPos.X,Canvas.PenPos.Y+10);

// end;

end;

end;

function calcY(I:TImage;offsetY,min,max:Integer;Element:TAlgRec):Integer;

var

s,buff,yL,yPixelPos,yLengthData,yLengthPixels:Integer;

k:Real;

savePos:TPoint;

begin

calcY:=I.Height - offsetY;

yL:=((I.Height - 2\*offsetY) div (dashCount+1));

yPixelPos:=offsetY+yL;

if Element.Data=max then

begin

calcY:=yPixelPos;

end else

if Element.Data=min then

begin

yPixelPos:=I.Height-yPixelPos;

calcY:=yPixelPos;

end else

begin

yLengthData:=max-min;

yLengthPixels:=I.Height-2\*(offsetY+yL);

k:=yLengthPixels/yLengthData;

yPixelPos:=max-Element.Data;

yPixelPos:=round(yPixelPos\*k);

yPixelPos:=yPixelPos+offsetY+yL;

calcY:=yPixelPos;

end;

end;

procedure drawMinMax(I:TImage; offsetX,offsetY,min,max,T:Integer);

var

yL,dy,s,textPos:Integer;

begin

textPos:=offsetX+5;

yL:=(I.height - 2\*offsetY) div (dashCount+1);

I.Canvas.MoveTo(textPos,offsetY);

I.Canvas.Brush.Color:=clWhite;

case T of

1:

begin

dy:=(max-min) div (dashCount-1);

for s := 0 to (dashCount-2) do

begin

I.Canvas.MoveTo(textPos,I.Canvas.PenPos.Y+yL);

I.Canvas.TextOut(I.Canvas.PenPos.X,I.Canvas.PenPos.Y,FloatToStrF((max-dy\*s)/1024,ffGeneral,4,4));

end;

I.Canvas.MoveTo(textPos,I.Canvas.PenPos.Y+yL);

I.Canvas.TextOut(I.Canvas.PenPos.X,I.Canvas.PenPos.Y,FloatToStrF(min/1024,ffGeneral,4,4));

end;

2:

begin

dy:=(max-min) div (dashCount-1);

for s := 0 to (dashCount-2) do

begin

I.Canvas.MoveTo(textPos,I.Canvas.PenPos.Y+yL);

I.Canvas.TextOut(I.Canvas.PenPos.X,I.Canvas.PenPos.Y,IntToStr(max-dy\*s));

end;

I.Canvas.MoveTo(textPos,I.Canvas.PenPos.Y+yL);

I.Canvas.TextOut(I.Canvas.PenPos.X,I.Canvas.PenPos.Y,IntToStr(min));

end;

end;

end;

procedure drawPillars(I:TImage; offsetX,offsetY:Integer; DataArr:TGraphDataArray;T:Integer);

var

count, s, xL, min, max,saveWidth:Integer;

savePos: TPoint;

begin

count:=Length(DataArr);

I.Canvas.Pen.Width:=10;

I.Canvas.MoveTo(offsetX,I.height - offsetY);

xL:=(I.Width - 2\*offsetX) div (count+1);

min:=DataArr[0].Data;

max:=DataArr[0].Data;

for s := 1 to Length(DataArr)-1 do

begin

if DataArr[s].Data>max then

begin

max:=DataArr[s].Data;

end else

if DataArr[s].Data<min then

begin

min:=DataArr[s].Data;

end;

end;

drawMinMax(I,offsetX,offsetY,min,max,T);

I.Canvas.MoveTo(offsetX,I.height - offsetY);

with I do

begin

for s := 0 to (count-1) do

begin

Canvas.MoveTo(Canvas.PenPos.X+xL,I.height - offsetY);

savePos:=Canvas.PenPos;

Canvas.TextOut(Canvas.PenPos.X-((Canvas.Font.Size div 2)\*Length(DataArr[s].Name)-(Canvas.Font.Size div 4)),Canvas.PenPos.Y+10,DataArr[s].Name);

Canvas.PenPos:=savePos;

Canvas.LineTo(Canvas.PenPos.X,calcY(I,offsetY,min,max,DataArr[s]));

savePos:=Canvas.PenPos;

saveWidth:=Canvas.Pen.Width;

Canvas.Pen.Width:=1;

Canvas.Pen.Style:=psDash;

Canvas.LineTo(offsetX,Canvas.PenPos.Y);

Canvas.PenPos:=savePos;

Canvas.Pen.Width:=saveWidth;

Canvas.Pen.Style:=psSolid;

end;

end;

end;

procedure drawAxises(I:TImage;DA:TGraphDataArray;T:Integer);

var

p:Array of TPoint;

offsetX, offsetY:Integer;

begin

setLength(p,3);

offsetX:=I.width div 10;

offsetY:=I.height div 10;

I.Canvas.Font.Size:=10;

I.Canvas.Font.Name:='Calibri';

with I do

begin

Canvas.Brush.Color:=Canvas.Pen.Color;

Canvas.MoveTo(offsetX,offsetY);

p[0]:=point(Canvas.PenPos.X,Canvas.PenPos.Y);

p[1]:=point(Canvas.PenPos.X + 5,Canvas.PenPos.Y + 15);

p[2]:=point(Canvas.PenPos.X - 5,Canvas.PenPos.Y + 15);

Canvas.Polygon(p);

Canvas.LineTo(Canvas.PenPos.X,height - Canvas.PenPos.Y);

Canvas.LineTo(width - Canvas.PenPos.X,Canvas.PenPos.Y);

p[0]:=point(Canvas.PenPos.X,Canvas.PenPos.Y);

p[1]:=point(Canvas.PenPos.X - 15,Canvas.PenPos.Y - 5);

p[2]:=point(Canvas.PenPos.X - 15,Canvas.PenPos.Y + 5);

Canvas.Polygon(p);

end;

drawDashes(I,offsetX,offsetY);

if length(DA)>0 then

begin

drawPillars(I,offsetX,offsetY,DA,T);

end;

end;

procedure drawSizeGraph(I:TImage;SA:TGraphDataArray);

begin

I.Picture:=nil;

drawAxises(I,SA,1);

end;

procedure drawTimeGraph(I:TImage;TA:TGraphDataArray);

begin

I.Picture:=nil;

drawAxises(I,TA,2);

end;

var

TimeArrG,SizeArrG:TGraphDataArray;

//------MAIN------

procedure LoadGraphData(TimeArr:TGraphDataArray;SizeArr:TGraphDataArray);

var

i:Integer;

begin

SetLength(TimeArrG,Length(TimeArr));

SetLength(SizeArrG,Length(SizeArr));

for i := 0 to Length(TimeArrG)-1 do

begin

TimeArrG[i]:=TimeArr[i];

end;

for i := 0 to Length(SizeArrG)-1 do

begin

SizeArrG[i]:=SizeArr[i];

end;

end;

procedure TForm2.FormCreate(Sender: TObject);

var

i:Integer;

begin

Self.DoubleBuffered:=True;

SetLength(TimeArrG,0);

SetLength(SizeArrG,0);

end;

procedure TForm2.FormPaint(Sender: TObject);

begin

drawSizeGraph(Self.SizeGraphImage,SizeArrG);

drawTimeGraph(Self.TimeGraphImage,TimeArrG);

end;

procedure TForm2.FormResize(Sender: TObject);

begin

Self.refresh;

end;

procedure TForm2.FormShow(Sender: TObject);

begin

//

end;

end.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | | |
|  | | | | Текстовые документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| БГУИР КП 1–40 01 01 012ПЗ | | | | Пояснительная записка | | | | 86 с. | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | | Графические документы | | | |  | | | |
|  | | | |  | | | |  | | | |
| ГУИР 851002  012 ПД | | | | Схема программы | | | | Формат А1 | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  | | | |  | | | |  | | | |
|  |  |  |  |  | БГУИР КП 1-40 01 01 012 Д1 | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Анализ алгоритмов сжатия  файлов  Ведомость курсового  проекта |  | | | | Лист | Листов |
| Разраб. | | Ковалевский М.Ю. |  |  |  |  | |  | 83 | 83 |
| Пров. | | Фадеева Е.Е. |  |  | Кафедра ПОИТ  Гр. 851002 | | | | | |
|  | |  |  |  |