**ВВЕДЕНИЕ**

Сжатие данных – это преобразование данных для того, чтобы уменьшить занимаемый этими данными объем. Это нужно для рационального использования объема запоминающих устройств. Сжатие основывается на избыточности информации. Такая избыточность бывает двух видов:

* Повторение в файле фрагментов;
* Некоторые фрагменты в файле встречаются чаще других.

Так же сжатие делится на:

* Без потерь;
* С потерями.

Сжатие без потерь позволяет полностью восстанавливать исходные данные, а сжатие с потерями дает возможность восстановить данные с небольшими искажениями.

В этой курсовой работе я создам программное средство сжатия на основе алгоритма Хаффмана.

Данная пояснительная записка содержит следующие основные этапы:

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. **Сжатие Хаффмана**

Сжатие Хаффмана – это алгоритм эффективного кодирования информации без потерь. Идея, положенная в основу кодировании Хаффмана, основана на частоте появления символа в последовательности. Символ, который встречается в последовательности чаще всего, получает новый очень маленький код, а символ, который встречается реже всего, получает, наоборот, очень длинный код. Это нужно, так как мы хотим, чтобы, когда мы обработали весь ввод, самые частотные символы заняли меньше всего места (и меньше, чем они занимали в оригинале), а самые редкие — побольше (но так как они редкие, это не имеет значения).

Так же код Хаффмана обладает свойством префиксности (то есть ни одно кодовое слово не является префиксом другого), что позволяет однозначно их декодировать.

Допустим, у меня есть строка символов, тогда алгоритм сжатия выглядит так:

* Создаю статистику встречаемости(вероятности) каждого символа в строке;
* Создаю бинарное дерево, в котором корень – это сумма вероятностей предыдущих двух его потомков, в свою очередь составленных из сумм их потомков и так далее;
* Прохожу по дереву от корня до каждого символа, записывая «повороты», например, при переходе в левую ветку я записываю ноль, в правую – единицу. Как только я дохожу до конечного символа, записанная мною последовательность из нулей и единиц будет кодом данного символа.

Пример: строка “Мама мыла раму”

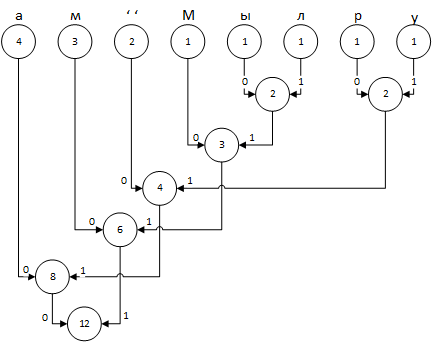
Создаю таблицу встречаемости:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| М | а | м | ы | л | р | у | ‘ ‘ |
| 1 | 4 | 3 | 1 | 1 | 1 | 1 | 2 |

Теперь сортирую данную таблицу по убыванию весов:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| а | м | ‘ ‘ | М | ы | л | р | у |
| 4 | 3 | 2 | 1 | 1 | 1 | 1 | 1 |

Теперь построю дерево по данной таблице, всегда объединяя два корня с наименьшими вероятностями:



*Рисунок 1 – Построение кодового дерева Хаффмана.*

Отсюда мы получим таблицы кодов:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| а | м | ‘ ‘ | М | ы | л | р | у |
| 00 | 10 | 010 | 110 | 1110 | 1111 | 0110 | 0111 |

Текст, зашифрованный в двоичном коде по ASCII таблице:

110011001110000011101101110000000100000111011001111101111101011111000000010000011110000111000001110110011110011 – 112 бит.

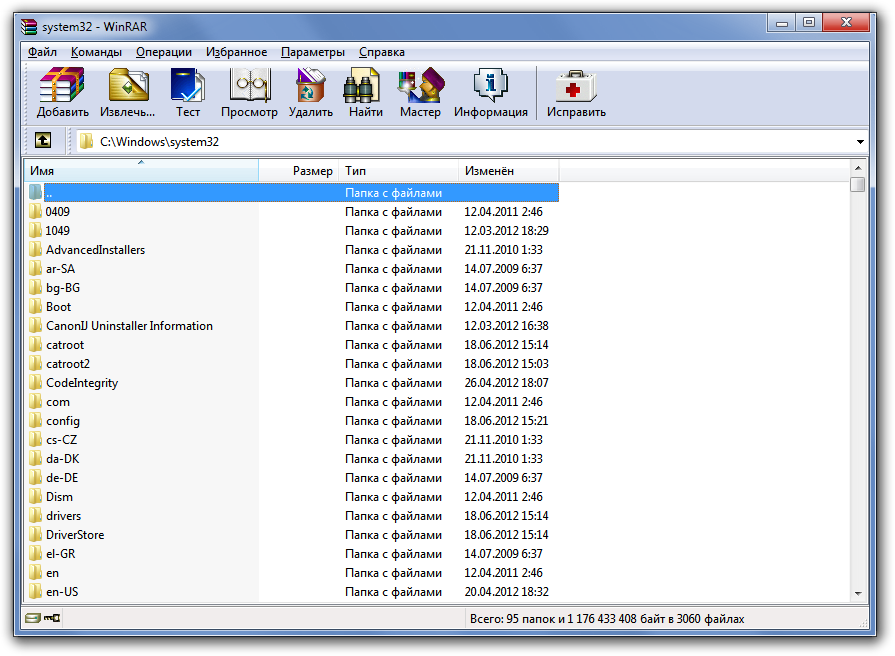
Текст, зашифрованный кодами:

110001000010101110111100010011000100111 – 39 бит.

Создаётся впечатление, что данный алгоритм очень эффективен, но на практике, для расшифровки такого сообщения в его начало прибавляется таблица вероятностей (для построения дерева), ее длина и длина сообщения, что прибавляет еще 20-30 % к весу выходного файла. И тогда на практике, эффективность алгоритма сжатия Хаффмана составляет, в среднем, 20-30%.

* 1. **Сравнение аналогов**

**1.2.1 WinRAR** – популярный архиватор файлов в форматы RAR и ZIP.



*Рисунок 2 – Внешний вид WinRAR*

В данном архиваторе при процессе архивации используется алгоритм Хаффмана для сжатия информации. Нужно понимать, что кроме данного алгоритма, в зависимости от настроек архивации параллельно используются и другие методы, для увеличения эффективности архивации.

**Плюсы:**

+ Скорость сжатия;

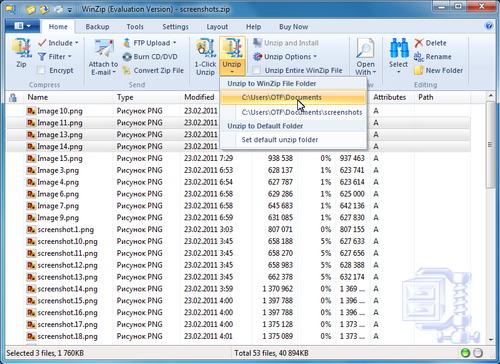
+ Вес выходного файла;

+ Минимальный, но оптимальный набор функций.

**Минусы:**

- Распространяется на платной основе.

В целом, я считаю, что это самый удобный архиватор, направленный на потребности обычных пользователей.

**1.2.2 WinZIP** – более свежий конкурент рассмотренного выше архиватора **WinRAR**, произведен компанией Corel с плотной интеграцией в оболочку Microsoft Windows.

*Рисунок 3 – Внешний вид WinZIP*

В данном архиваторе так же при архивации файлов используется метод сжатия Хаффмана. Нужно понимать, что кроме данного алгоритма, в зависимости от настроек архивации параллельно используются и другие методы, для увеличения эффективности архивации.

**Плюсы:**  
+ Многофункционален;

+ Эффективность архивации.

**Минусы:**

- Высокая стоимость;

- “Захламленный” функционал;

- Недружелюбный пользовательский интерфейс.

Я не использую данный архиватор потому что он неудобен, а так же из-за его большой стоимости.

* 1. **Постановка задачи**

Проектируемое программное средство сжатия должно эффективно уменьшать вес входных данных; так же оно должно никак не повреждать входные данные; программное средство должно уметь разжимать файлы, сжатые в данной программе; должно быть удобным и иметь максимально понятный для пользователя интерфейс; так же необходима возможность вызова помощи по данной программе; важно и отсутствие лишних функций, присутствующих в различных аналогах; необходимо наличие минимальных настроек, хотя бы возможность удалять уже заархивированные файлы. Для того, чтобы программное средство отвечало данным требованиям, в нем должны быть реализованы следующие функции:

* Сжатие файлов;
* Разжатие файлов;
* Справка по данному программному средству;
* Отображение прогресса в процессе сжатия и разжатия.

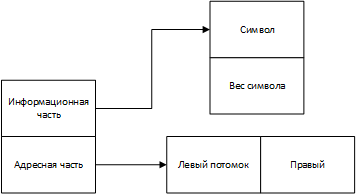
Так как данная программа не требует продвинутых функций, для ее написания был выбран язык Delphi, так как данным языком программирования я владею на достаточном уровне.

# МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

**2.1 Проектирование динамических структур**

Любой файл – это, по сути, массив байтов. При этом, каждый элемент данного массива содержит значение от 0 до 255. Поэтому, для алгоритма Хаффмана не важно, файл какого типа я имею на входе (текстовый, изображение и т.д.).

Мною было принято решение в качестве структуры, которая будет хранить информацию о вероятности символа и его самого, использовать дерево. В информативной части будут находится сам символ (можно использовать число типа Byte, потому что в таблице каждому символу соответствует код), и его вес (вероятность, встречаемость). В адресной части будут содержатся две ссылки: на правого и левого потомка, необходимыми для построения кодового дерева.



*Рисунок 4 – Структура данных*

Всего таких деревьев может быть (максимум) 256, то есть символ 00 – минимальный, а символ FF – максимальный (В шестнадцатеричной системе счисления).

|  |  |  |  |
| --- | --- | --- | --- |
| Node[0] | Node[1] | … | Node[255] |

*Таблица 1 – Массив деревьев*

**2.1.1 Структура нетипизированных файлов**

В ходе анализа алгоритмов и исходя из аналогов мною было принято решение принять следующую структуру сжимаемых файлов:

* + - 1. Первые четыре байта содержат размер исходного файла;
      2. Следующий байт содержит количество уникальных байтов или длину таблицу встречаемости;
      3. Далее идет таблица встречаемости – один байт на символ (его код), затем 4 байта на его «вес» в файле;
      4. Как только заканчивается таблица встречаемости – начинается зашифрованная последовательность байтов;
      5. Так как коды Хаффмана обычно меньше одного байта, то при записи закодированного потока байтов нужно использовать битовую арифметику, а именно логических операции и логические сдвиги.

**2.2 Описание функциональности ПС**

Данное программное средство должно предоставлять возможность пользователю выбора действия: сжатия и разжатия. Так же, при нажатии необходимых кнопок (сжатия или разжатия) должно открываться диалоговое окно, в котором пользователь сможет выбрать необходимый файл. Должна присутствовать вкладка “Настройки”, в которой пользователь сможет выбрать режим работы программы, а именно – удалять ли файлы после сжатия/разжатия для экономии места на диске.

Для того, чтобы пользователю не казалось, что программа “зависла” в результате выполнения операций над файлами необходимо, чтобы в процессе сжатия или разжатия на экран выводился прогресс. Удобнее всего сделать это в виде так называемой строки прогресса (Progress Bar).

При невозможности сжатия или разжатия файла, а также после успешного завершения данных процессов уведомлять пользователя об этом.

Так же, для помощи пользователю должна быть возможность вызвать справку по данному приложению. Это должно происходить при нажатии на стандартную кнопку F1 или же при нажатии кнопки “Помощь” в меню.

Тогда:

* По нажатию на кнопку “Сжать” открывается диалоговое окно, в котором пользователь выбираем нужный ему файл;
* По нажатию на кнопку “Разжать” открывается диалоговое окно, в котором пользователь выбирает нужный ему файл;
* По нажатию на кнопку “Настройки” открывается новое окно, в котором есть Checkbox, который позволяет включить или выключить режим экономии места;
* По нажатию на клавишу F1 или на кнопку “Помощь” открывается текстовое окно с информацией, которая может помочь пользователю;
* Уведомление об ошибках или об успешном завершении процессов сжатия и разжатия.

**2.3. Спецификация функциональных требований**

Во время разработки данного программного средства должны быть реализованы следующие функции:

2.3.1. Фукция сжатия файлов;

2.3.2. Функция разжатия файлов, сгенерированных в данном программном средстве;

2.3.3. Функция настроек, в которой будет присутствовать пункт, отвечающий за экономию места на хранилище;

2.3.4. Функция помощи, в которой будет описан базовый алгоритм взаимодействия пользователя с программой;

2.3.4 Функция отображения достигнутого прогресса в процессе выполнения главных операций (сжатие и разжатие);

2.3.5. Функция отображения ошибок, полученных в ходе выполнения операций;

2.3.6. Функция отображения отчета об успешном завершении главных процессов;

**3. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

Схема алгоритма программы представлен в *приложении 1.* Так же он выполнен на листе, формата А1.

Ниже представлен словесный алгоритм:

3.1 Программа начинается с создания интерфейса программы;

3.2 Пользователь может выбрать один из процессов: сжатие, разжатие, настройки и помощь;

3.3 После выбора операции «сжатие» пользователь увидит диалоговое окно, в котором он сможет выбрать файл для сжатия;

3.4 Если файл занят другой программой или же недоступен для чтения, то на экран необходимо вывести сообщение об ошибке и вернутся к выбору операции;

3.5 Если файл доступен, тогда начинается процесс сжатия. Сначала собирается статистика встречаемости байт в файле. Затем собранная статистика помещается в массив. В этом массиве строится дерево кодированое, по алгоритму, расположенному в *приложении 2.* После построения дерево, производится кодировка каждого символа и запись сжатого потока байтов в файл. После завершения сжатия на экран выводится сообщение об успешном сжатии

3.6. После выбора операции «разжатие» пользователь так же увидит диалоговое окно, в котором он сможет выбрать необходимый файл;

3.7 Если файл занят другой программой или же он поврежден, на экран выведется сообщение об ошибке;

3.8 Если файл доступен, тогда начинается процесс разжатия. Сначала читается заголовок файла (О нем подробнее дальше). Заголовок записывается в таблицу встречаемости. Затем по ней так же строится кодовое дерево и начинается разжатие файла по алгоритму, расположенному в *приложении 3.* После разжатия поток байтов сохраняется в файл, с именем и разрешением, сгенерированным по исходномуфайлу.

3.9 После выбора операции «Настройки» должно появляться новое окно, в котором будет настройка, отвечающая за режим экономии памяти.

3.10 После выбора операции «Помощь» на экране должна появится текстовая справка по данному программному средству.

Далее будут рассмотрены некоторые словесные описания алгоритмов модулей, которые требуют пояснений:

**Алгоритм сжатия файла:**

3.1 Передаем имя сжимаемого файла в программу;

3.2 Запускаю процедуру создания статистики появления символа в файле;

3.3 Запуск процедуры создания сжимаемого файла;

3.3.1 Сортирую массив деревьев по убыванию весов;

3.3.2 Подсчет количества уникальных байт (из 256 возможных);

3.3.3 Создаем дерево (*Приложение 5*);

3.3.4 Создаем кодовые слова для кодирования входного файла (*Приложение 6*);

3.3.5 Записываем сам файл (*Приложение 7*);

3.3.6 Очищаем дерево;

3.3.7 Очищаем статистику файла;

**Алгоритм разжатия файла:**

3.1 Чтение заголовка файла

3.2 Инициализирую объект со статистикой файла;

3.3 Включаю частоты байт в массив;

3.4 Создаю дерево;

3.5 Вызываю процедуру разжатия файла (*Приложение 8*);

3.6 Очищаю дерево.

Подробнее данный алгоритм в *приложении 9*.

1. **Создание программного средства**

Спецификация функциональных требований и спроектированная архитектура программного средства служат фундаментом, на котором основывается выбор наиболее подходящих технологий для разработки программного средства. Успешное и обоснованное завершение данных этапов позволит создать расширяемое, надежное и функциональное приложение, призванное решать поставленные задачи.