Курсовая работа на тему

«Программа-архиватор на основе библиотеки ZLib для C#»

Работу выполнили ученики 11 «В» класса СУНЦ МГУ:

Арутюнов Эдгар и Горбунов Андрей.

Оглавление

Постановка задачи2

Используемые алгоритмы и технологии2

Сравнение с другими архиваторами6

Руководство пользователя10

Перспективы технического развития11

Список литературы11

Постановка задачи

К сожалению большинство современных программ-архиваторов порой недостаточно гибки в плане функциональности для определённых нужд пользователя или не обладают набором готовых классов, процедур, функций и т.д., поэтому нами была поставлена задача, заключающаяся в проектировании, создании и тестировании собственной настраиваемой программы-архиватора на основе библиотеки «ZLib» для языка программирования «C#», имеющей API (англ. application programming interface).

Также был разработано техническое задание, описывающие функционал будущего архиватора, которое выглядит следующим образом:

Архиватор должен уметь:

1. Сжимать файлы и каталоги.
2. Собирать архив без сжатия.
3. Шифровать данные (со сжатием и без).
4. Исключать указанные пути.
5. Удалять файлы после их сжатия.
6. Распаковывать сжатый архив.

Используемые алгоритмы и технологии

Для выполнения поставленной нами задачи нам понадобились:

1. Интегрированная среда разработки Microsoft Visual Studio.
2. Библиотека zlib.net.dll («Деликатно ненавязчивая библиотека сжатия»), осуществляющая компрессию и декомпрессию.
3. Framework 3.0.
4. Прямые руки.

Вышеупомянутая библиотека осуществляет компрессию посредством одной из идей, потрясших мир ещё в 70-ые годы XX века, ‑ методом Зива-Лемпела. Он даёт эффективное сжатие и применяет подход, совершенно отличный от хаффмановского и арифметического. Естественно, с тех пор этот метод всячески усовершенствовался и лёг в основу наиболее эффективных алгоритмов сжатия без потерь. В основном, идея метода Зива-Лемпела используется в словарных алгоритмах компрессии (Словарное сжатие – сжатие, в котором группы последовательных символов или "фраз" заменяются кодом. Замененная фраза может быть найдена в некотором "словаре").

Алгоритм сжатия является адаптированным. Адаптированные модели представляют собой элегантную и эффективную технику, и обеспечивают сжатие по крайней мере не худшее производимого неадаптированными (словарь кодировщика построен не по входным данным) схемами. Оно может быть значительно лучше, чем у плохо соответствующих текстам статичных моделей. Адаптированные модели, в отличии от полуадаптиpованных (словарь строится во время предварительного просмотра файла), не производят их предварительного просмотра, являясь поэтому более привлекательными и лучше сжимающими. Т.о. алгоритмы моделей, описываемые в подразделах, пpи кодировании и декодировании будут выполнятся одинаково. Модель никогда не передается явно, поэтому сбой происходит только в случае нехватки под нее памяти.

Конкретный алгоритм сжатия, предоставленный библиотекой, — LZ77. LZ77 использует уже просмотренную часть сообщения как словарь. Чтобы достигнуть сжатия, алгоритм пытается заменить очередной фрагмент сообщения на указатель в содержимое словаря.

В качестве модели данных LZ77 использует "скользящее" по сообщению окно, разделенное на две неравных части. Первая часть, большая по размеру, включает уже просмотренную часть сообщения. Вторая, намного меньшая, — буфер, содержащий еще не закодированные символы. Как правило, размер окна — несколько килобайтов. Буфер намного меньше, обычно не больше, чем сто байтов. Алгоритм пытается найти фрагмент в словаре, который совпадает с содержанием буфера.

Алгоритм LZ77 выдает коды, состоящие из трех элементов:

* смещение в словаре относительно его начала подстроки, которая соответствует содержанию буфера;
* длина подстроки;
* первый символ в буфере после подстроки.

После этого алгоритм перемещает окно на длину совпадающей подстроки +1 символ влево.

Если совпадение не найдено, алгоритм выдает код *<0, 0, первый символ в буфере>* и продолжает свою работу. Хотя такое решение неэффективно, оно гарантирует, что алгоритм сможет закодировать любое сообщение.

Пример:

|  |  |  |
| --- | --- | --- |
| Сообщение | Подстрока | Код |
| Kabababababz |  | <0, 0, k> |
| Kabababababz |  | <0, 0, a> |
| Kabababababz |  | <0, 0, b> |
| Kabababababz | ab | <2, 2, a> |
| Kabababababz | babab | <2, 5, z> |

Таким образом алгоритм превращает входные данные в последовательность блоков, которые бывают разных видов: первый – необработанные данные и несколько байтов на заголовок. В худшем случае, когда сжатый блок становятся объёмнее, чем несжатый, производится замена на несжатый блок. В итоге имеем накладные расходы по пять байт на каждые 16 килобайт блока и расход 6 байт на головной заголовок на весь поток. В абсолютном худшем случае на однобайтовом потоке накладные расходы составляют 1100%(11 байт накладных расходов, 1 байт фактических данных). При достаточно больших размерах потока накладные расходы приближаются к предельному значению 0,03%.

Методом пристального взгляда был определён оптимальный формат архива (структура тех самых блоков):

|  |  |
| --- | --- |
| Назначение | Размер |
| Тип архива | 1 байт |
| Длина заголовка (после сжатия и шифрования) | 4 байта |
| Заголовок (подробнее рассмотрим ниже) | N байт |
| Блок содержимого первого файла | N байт |
| Блок содержимого второго файла | N байт |
| ...... | ...... |
| Блок содержимого K-того файла | N байт |

Формат заголовка архива

|  |  |
| --- | --- |
| Назначение | Размер |
| Размер необработанного заголовка | 4 байта |
| Блок 1 | N байт |
| Блок 2 | N байт |
| ...... | ...... |
| Блок K | N байт |

Формат блока заголовка архива

|  |  |
| --- | --- |
| Назначение | Размер |
| Размер блока | 4 байта |
| Длина абсолютного пути | 4 байта |
| Абсолютный путь | N байт |
| Длина относительного пути | 4 байта |
| Относительный путь | N байт |
| Размер объекта после обработок | 8 байт |

Соответственно для технической реализации компрессии/декомпрессии нам, как программистам, необходимо было:

1. Считывание, перезаписывание информации, которую паковал/распаковывал LZ77.
2. Ведение всех заголовков и их обновление.
3. Взаимодействие между заголовками и поведением LZ77.
4. Своевременное удаление промежуточных данных, а также исходных данных по требованию.

Исходя из всего вышеперечисленного, можно сформировать алгоритм работы:

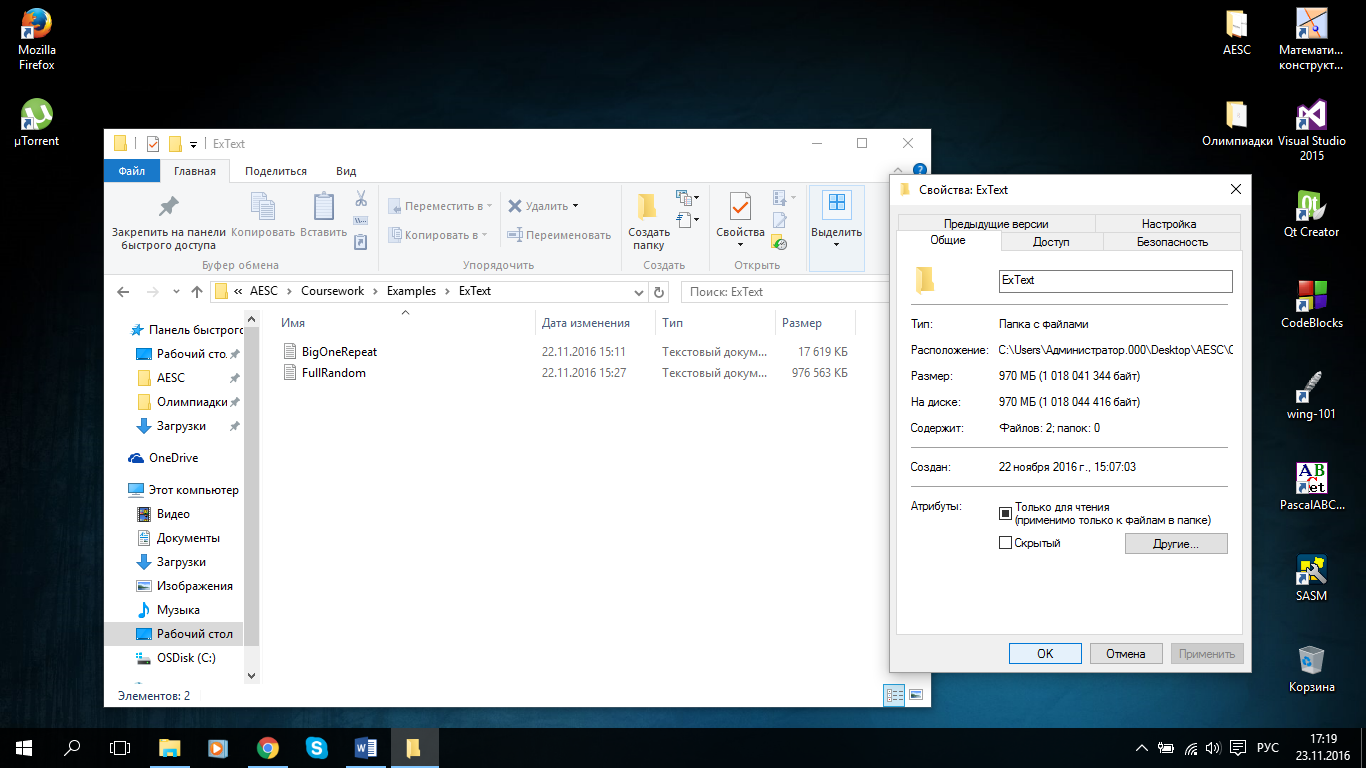
1. Определение типа архива (сжатый, зашифрованный).
2. Чтение списка объектов архивации.
3. Формирование на основе прочитанного списка и списка исключений полного списка архивируемых объектов.
4. Создание заголовка архива (в объектном виде).
5. Перебор полного списка объектов, лежащих в заголовке.
6. Обработка объекта, обновление данных о его размере после обработки в заголовке, запись во временный файл обработанного содержимого.
7. Сохранение заголовка в файл.
8. Обработка заголовка (сжатие, шифрование).
9. Сборка конечного файла архива.

Также в соображениях безопасности была придумана система шифрования архивов. Иными словами, архив можно распаковать только с помощью пароля. В нашем архиваторе производится симметричное шифрование по ключу, которым является пароль, известный только создателю архива и получателю.

Сравнение с другими архиваторами

Для сравнения мы взяли два самых популярных архиватора: «WinRar» и «7Zip». В ходе тестирования выяснилось, что время работы всех трёх архиваторов примерно одинаковы, поэтому интерес представляет только сравнение качества компрессии.

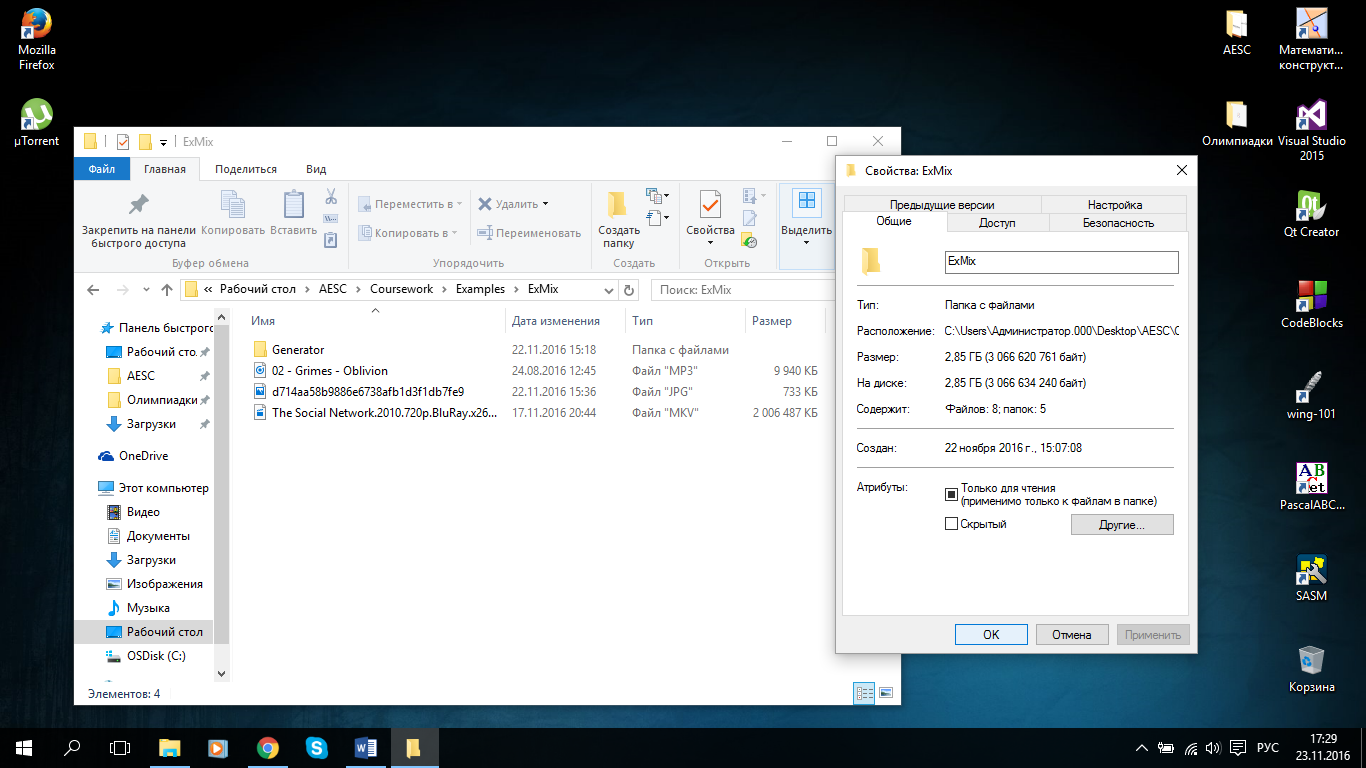
Итак, тест 1. В качестве архивируемого каталога выбрана папка с двумя текстовыми файлами: «BigOneRepeat», состоящий из повторяющегося набора символов (Подобный файл весом приблизительно 80000 кб наш архиватор способен сжать вплоть до 500 кб), и «FullRandom», состоящий из десяти в девятой степени случайных символов (очевидно, что такой файл сжимается очень плохо).



*Демонстрация объёма файлов теста №1.*

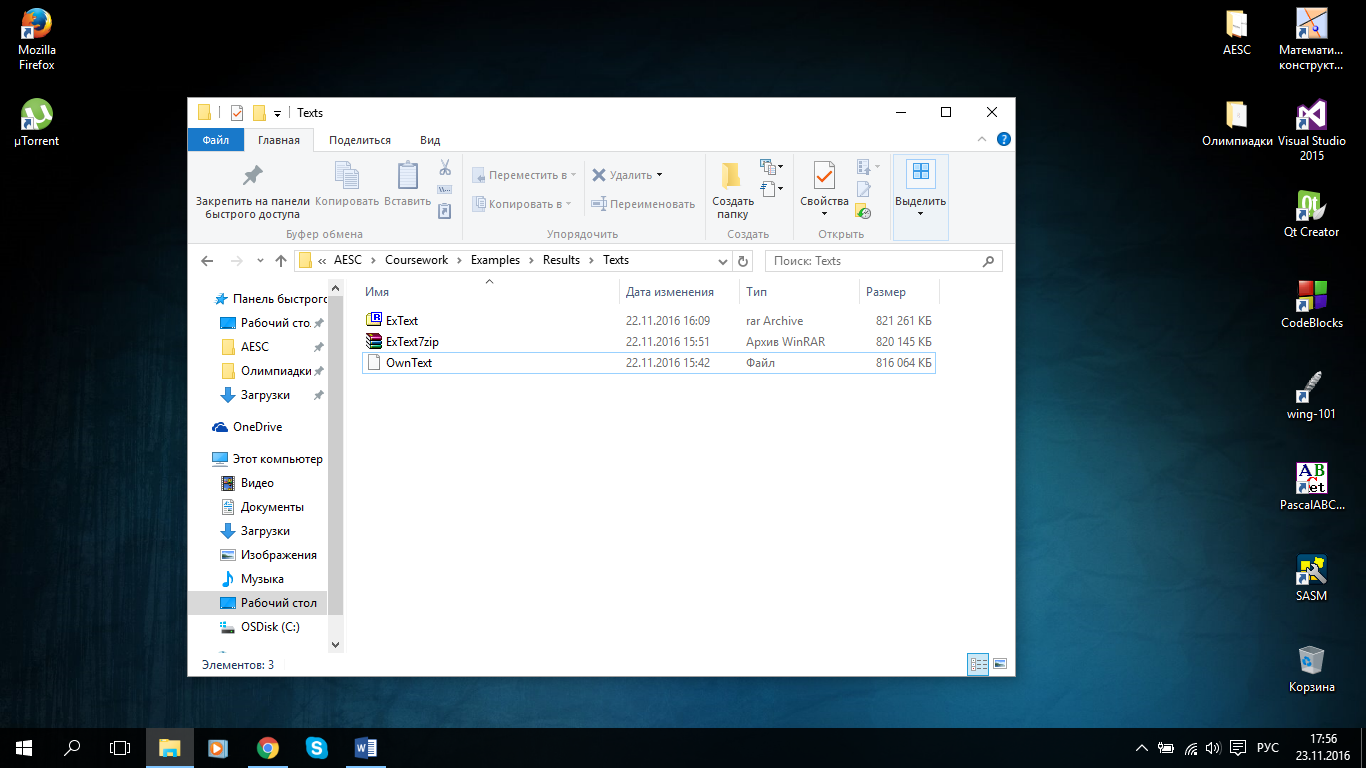
Для теста №2 был собран каталог «ExMix», в который помещены: программа, которая создавала текст для предыдущего теста вместе с этими текстами, аудиозапись, графическое изображение и видеозапись.

Для теста №3 была выбрана только аудиозапись из вышеприведённого каталога «ExMix».



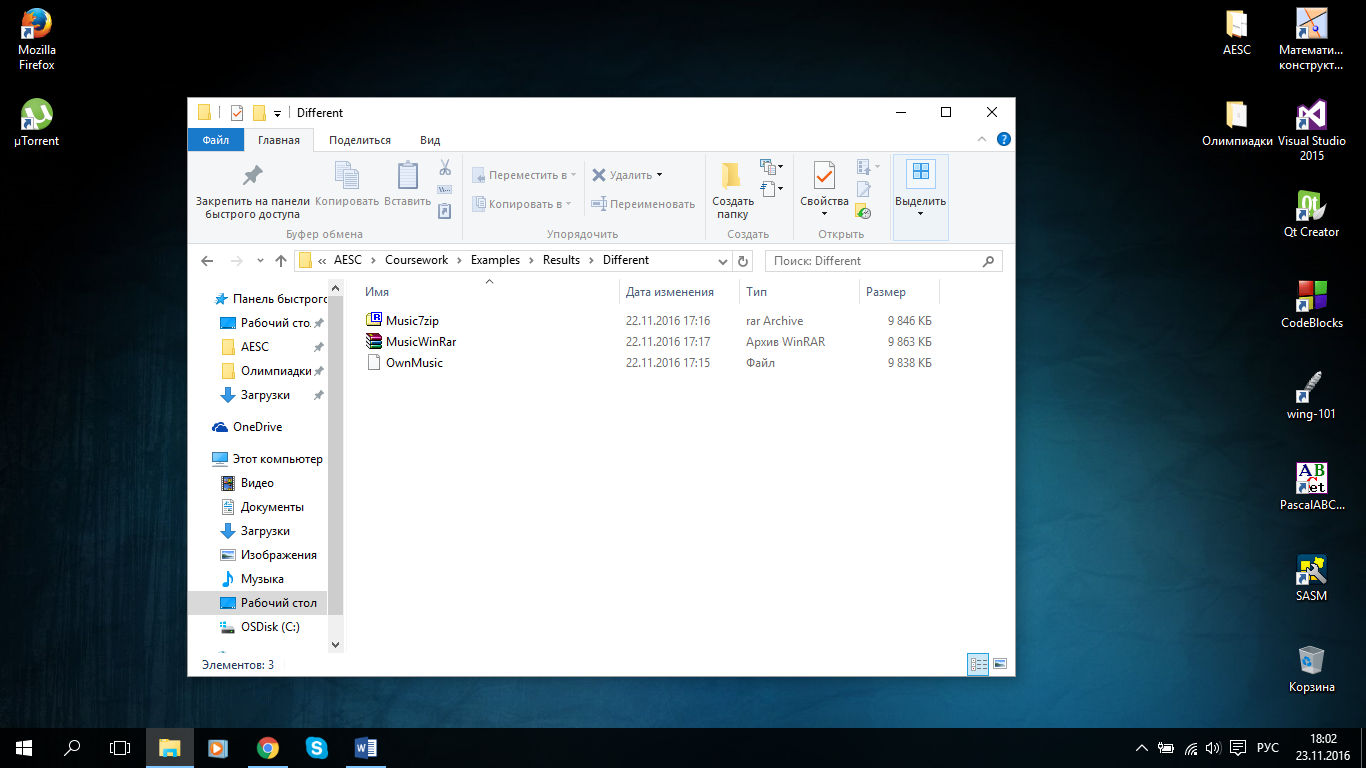
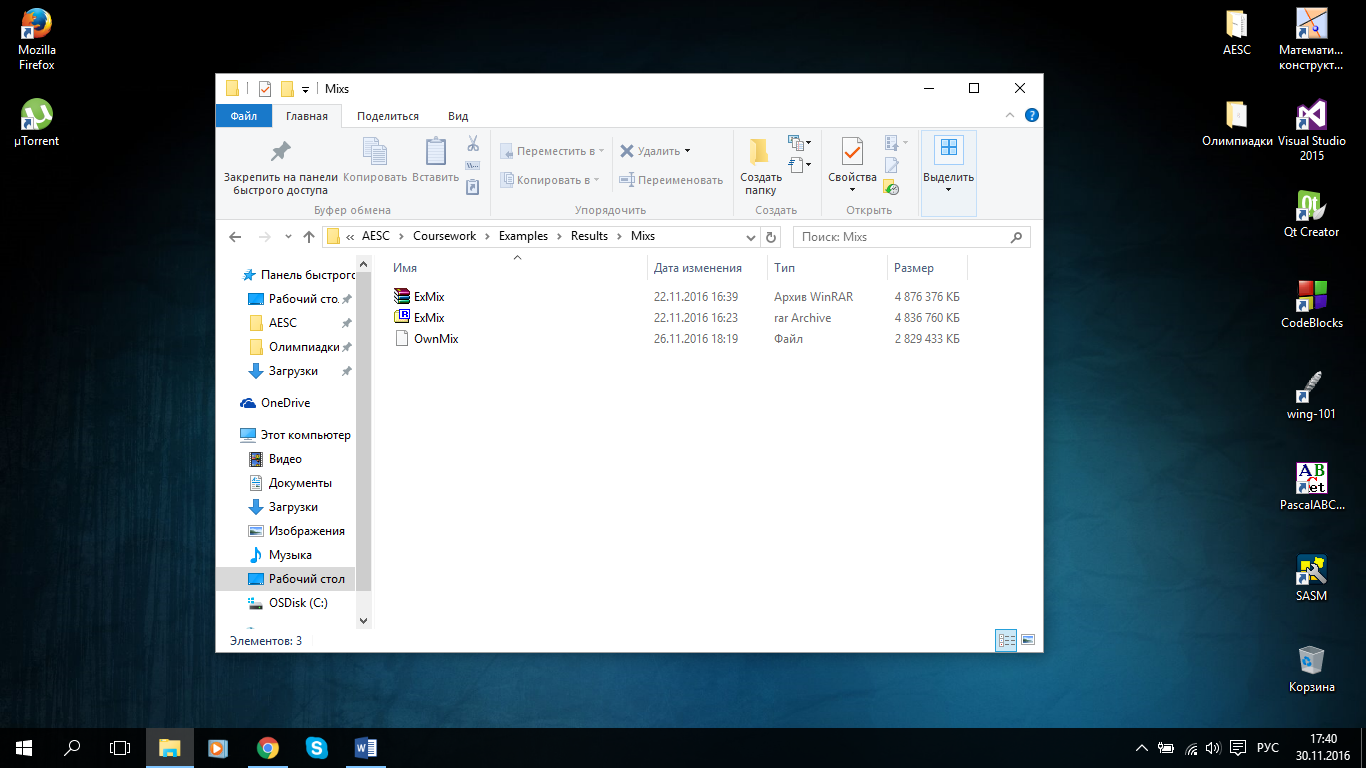
*Демонстрация объёмов каталога для теста №2*

Результаты для каждого из архиваторов были сохранены под соответствующим названием.



*Архивы теста №1*

*Архивы из теста №2*



*Архивы из теста №3*

Таким образом можно подытожить результаты сравнений:

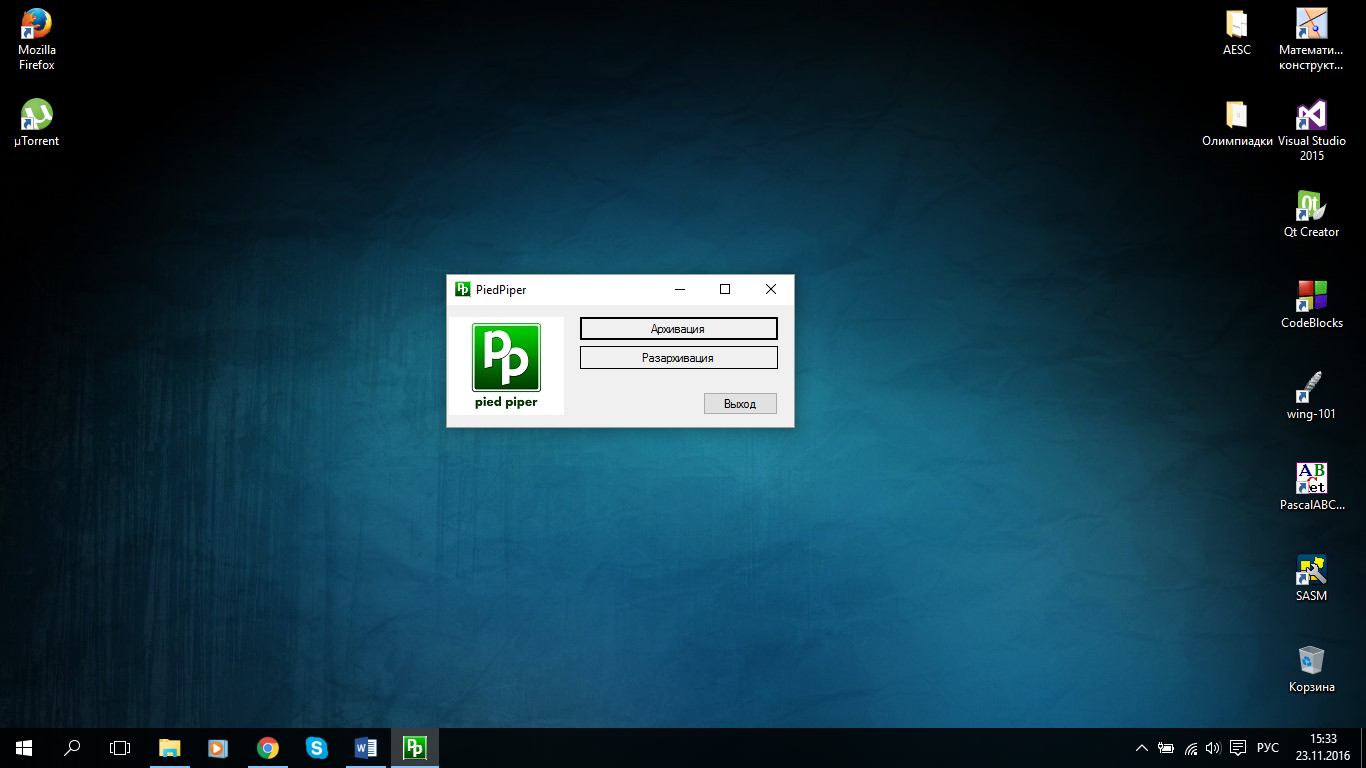
|  |  |
| --- | --- |
| Тест №1 (Текстовые файлы) | |
| Архиватор | Вес архива при объёме несжатого файла 970 МБ |
| WinRar | 802 МБ |
| 7Zip | 800 МБ |
| PiedPiper | 796 МБ |
| Тест №2 (Комплексное сжатие) | |
| Архиватор | Вес архива при объёме несжатого файла 2,85 ГБ |
| WinRar | 4,61 ГБ |
| 7Zip | 4,65 ГБ |
| PiedPiper | 2,69 ГБ |
| Тест №3 (Аудиофайлы) | |
| Архиватор | Вес архива при объёме несжатого файла 9,70 МБ |
| WinRar | 9,61 МБ |
| 7Zip | 9,63 МБ |
| PiedPiper | 9,6 МБ |

Опираясь на вышеприведённые аргументы, можно судить о лучшей степени сжатия нашим архиватором.

Руководство пользователя

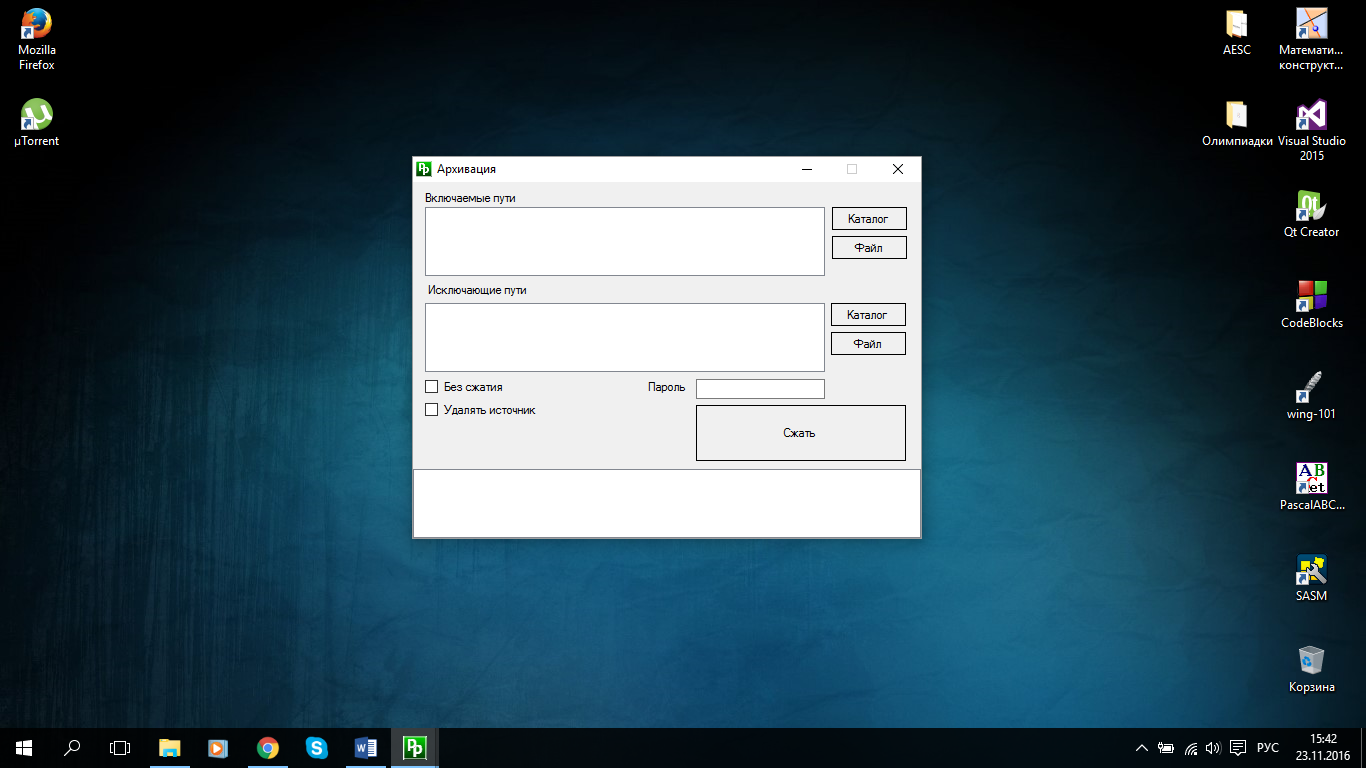
Программа состоит из главного окна (рис.1) и нескольких всплывающих окон:

1. Главное меню (рис. 1). Соответственно именно в нём пользователь может выбрать архивацию/разархивацию.



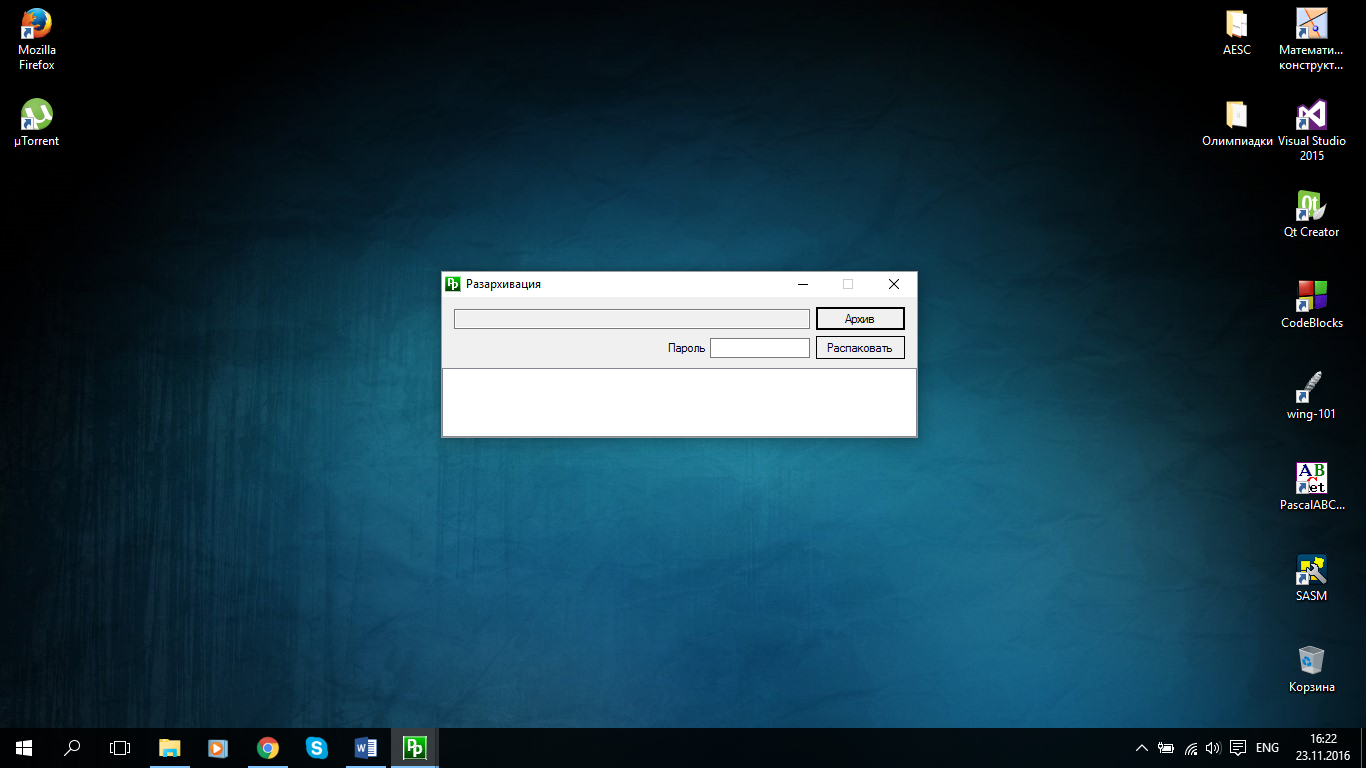
*Рис. 1*

1. Меню архивации (рис. 2). Интуитивно-понятное меню, демонстрирующее функции гибкого (благодаря наличию исключающих путей) комплексного архивирования каталогов и файлов. При желании можно установить параметры с говорящими названиями «Без сжатия» и «Удалять источник».



*Рис. 2*

1. Меню разархивации (рис. 3). Необходимо просто указать через каталог путь к архиву, пароль, если требуется (иначе, ничего не произойдёт), а также указать путь сохранения после нажатия кнопки «распаковать».



*Рис. 3*

1. Окно ошибки (Рис. 4). Если что-то идёт не так: файл повреждён, отсутствует и т.д. Пользователь способен выбрать, как поступить с процессом архивации.



*Рис. 4*

Руководство программиста

В программе реализованы следующие функции и алгоритмы:

1. Управление потоками ввода вывода для записи файла, в процессе которой производится сжатие файла и добавление в заголовок архива.
2. Гибкая система каталогов. Реализована возможность сборки архива как из каталога, так и из целой папки. Для удобства имеется возможность добавить «исключения», в виде каталогов или файлов.
3. Сборка архива и распаковка с помощью собственной структуры «Head», отвечающей за вышеупомянутые заголовки. Алгоритм собирает всю информацию для заголовка архива, затем сжимает его, и после всех операций перезаписывания файла-архива обновляет заголовок, по которому впоследствии происходит разархивация.
4. Несколько параметров архивации: архивация с сжатием и без него.
5. Симметричное шифрование архивов. Файл предварительно шифруется с помощью ключа, прежде чем быть сжатым.
6. Интуитивно понятный дизайн.

Перспективы технического развития:

* Добавление сохранения информации о файле (дата создания/изменения, права доступа).
* Добавление многопоточности.
* Возможность добавления комментариев к архиву.
* Ассоциирование файлов с программой.

Литература:

* http://www.compression.ru/ — теоретический материал по алгоритмам сжатия без потерь (Моделирование для сжатия текстов (Modeling for Text Compression) T. Bell, I.H. Witten, J.G. Cleary).
* http://www.msdn.microsoft.com/ — официальное руководство разработчика от «Microsoft».
* http://zlib.net/zlib\_tech.html — официальная документация библиотеки ZLib.
* http://www.wikipedia.org/ — куда же без неё?