ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Отчет

по контрольному домашнему заданию

по направлению подготовки бакалавров 09.03.04 «Программная инженерия»

Выполнил
студент группы БПИ181

_____Т.Н. Сафин_____
И.О. Фамилия

_____
Подпись, Дата

Принял
Доцент ДПИ, к.т.н
Р.З. Ахметсафина

_____
Оценка

_____
Подпись, Дата

Москва 2020

Содержание

Постановка задачи

Разработать на языке **C++** программу, реализующую алгоритмы сжатия данных без потерь, получение архивного файла из исходного и разархивированного файла из архивного (упаковка файла и распаковка архива).

Алгоритмы:

1. алгоритм *Шеннона-Фано* (простой)

Провести вычислительный эксперимент для исследования эффективности реализованных алгоритмов сжатия без потерь (упаковка и распаковка) для файлов разного типа. Для проведения эксперимента с алгоритмами сжатия без потерь необходимо использовать набор из N файлов различных типов с именами 1.* … N.*.

В наборе присутствуют следующие файлы с размером 2 мегабайт:

1. 1.docx
2. 2.pptx
3. 3.pdf
4. 4.txt
5. 5.exe
6. 6.jpg
7. 7.bmp
8. 8.jpg (черно-белое изображение)
9. 9.bmp (черно-белое изображение)

Описание алгоритмов и использованных структур данных

1. **Архиватор**

Класс FileArchiver служит основным классом для запаковки/распаковки файлов. Он содержит указатель на переменную типа Algorithm – интерфейс, который реализуется каждым алгоритмом.

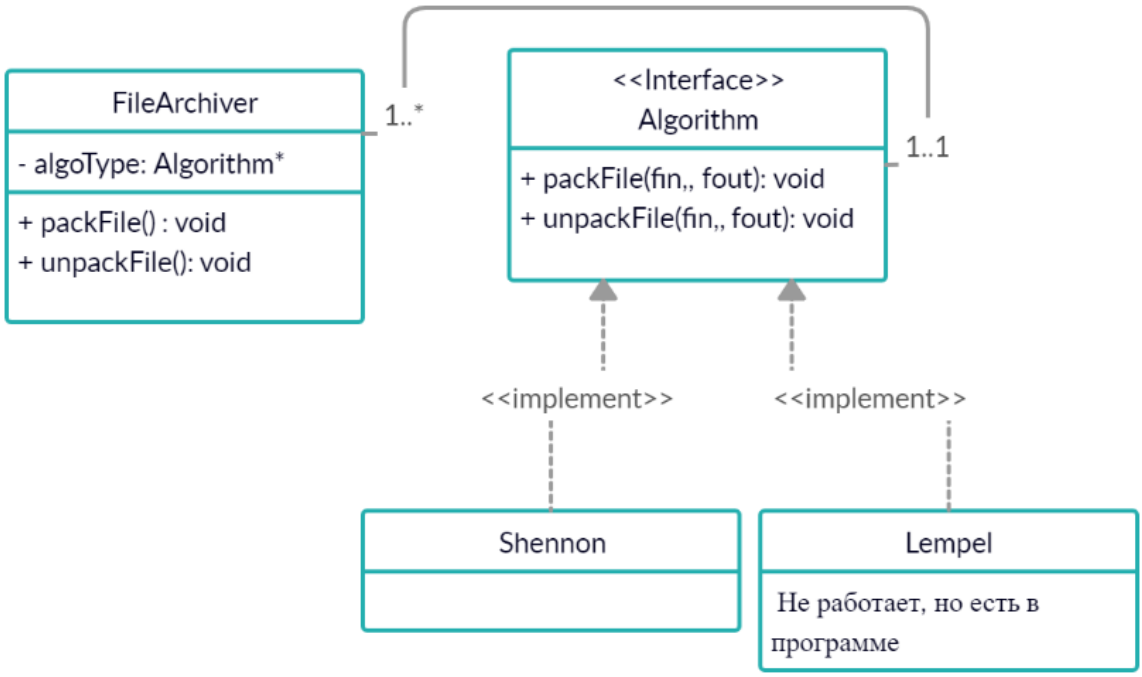

Рис.1 – Диаграмма классов программы

2. **Доп. методы**

Также в основном файле (main.cpp) содержаться методы для записи интересуемых нами данных в соответствующие csv-файлы.

3. **Подсчёт времени**

В качестве инструмента замера используется high_resolution_clock из библиотеки chrono.

4. **Шеннон-Фано**

Вся логика прописана в файле Shannon.cpp.

Шаги упаковки файла:

1. Находим частоты встречаемых символов в входном файле;
2. Строим код основываясь на частотах символов;
3. Записываем словарь (символ – код) в выходной файл;
4. Кодируем каждый символ из исходного файла соответствующим кодом. Полученный код разбивается по 8 битов, из которых мы получаем

символ, который записываем в выходной файл. (Учитывается то, что количество символов полученного кода не делится на цело на 8).

Шаги распаковки файла:

1. Считываем словарь (символ – код) из входного файла;
2. Заменяем каждый код на соответствующий символ;
3. Записываем этот символ в выходной файл;

5. Стандартные труктуры данных

В самой программе также используются стандартные структуры данных: vector, map.

План эксперимента

Для проведения эксперимента была реализована программа, которая выполняет сжатие и восстановление файлов. Файлы предварительно создаются (в нашем случае весом 2 МБ) и расфасовываются для удобности по папкам (каждая папка отдельный файл).

Для получения достоверных результатов упаковку и распаковку каждого файла каждым методом выполним 10 раз.

Для автоматизации подсчета данных в программе предусмотрено вывод в csv-файл:

1. Энтропии файлов
2. Размера исходного и полученного файлов
3. Среднее время работы за определенное количество экспериментов (в нс)
4. Коэффициент сжатия

Далее из полученных данных построим графики и таблицы в программе MS Excel. В конце на полученных данных сделаем выводы.

Использованные аппаратные средства для проведения эксперимента

1. Операционная система - Windows 10 Домашняя
2. Процессор - Intel® Core™ i7-10510U, 1,8 ГГц, 4 ядра
3. Видеокарта - Intel® UHD Graphics 620
4. Оперативная память - 16 ГБ
5. Хранение данных - 512 ГБ SSD

Результаты экспериментов - таблицы и графики

Рис. 2 – частота символов в файле 1.docx

Рис. 3 – частота символов в файле 2.pptx

Рис. 4 – частота символов в файле 3.pdf

Рис.5 – частота символов в файле 4.txt

Рис. 6 – частота символов в файле 5.exe

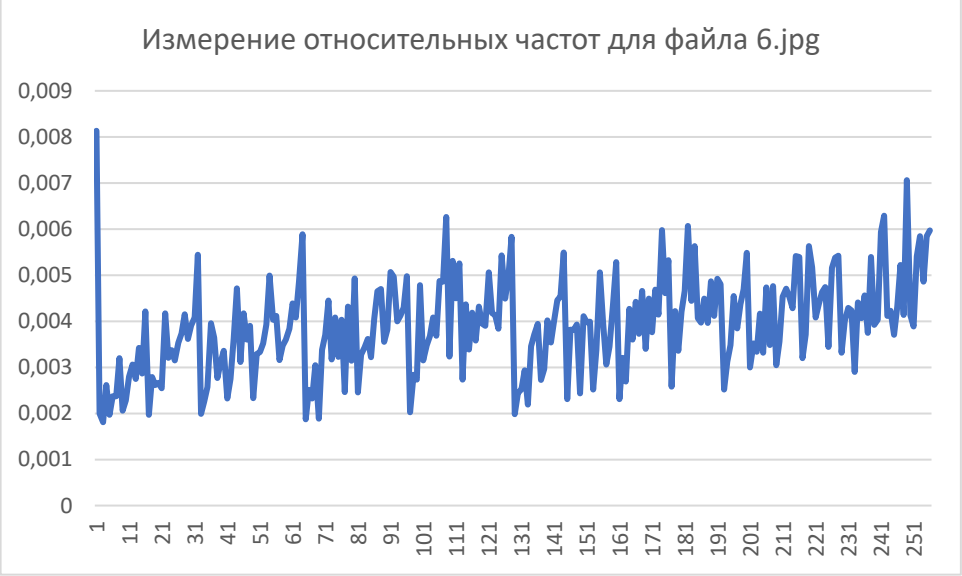

Рис. 7 – частота символов в файле 6.jpg

Рис. 8 – частота символов в файле 7.bmp

Рис. 9 – частота символов в файле 8.jpg

Рис. 10 – частота символов в файле 9.bmp

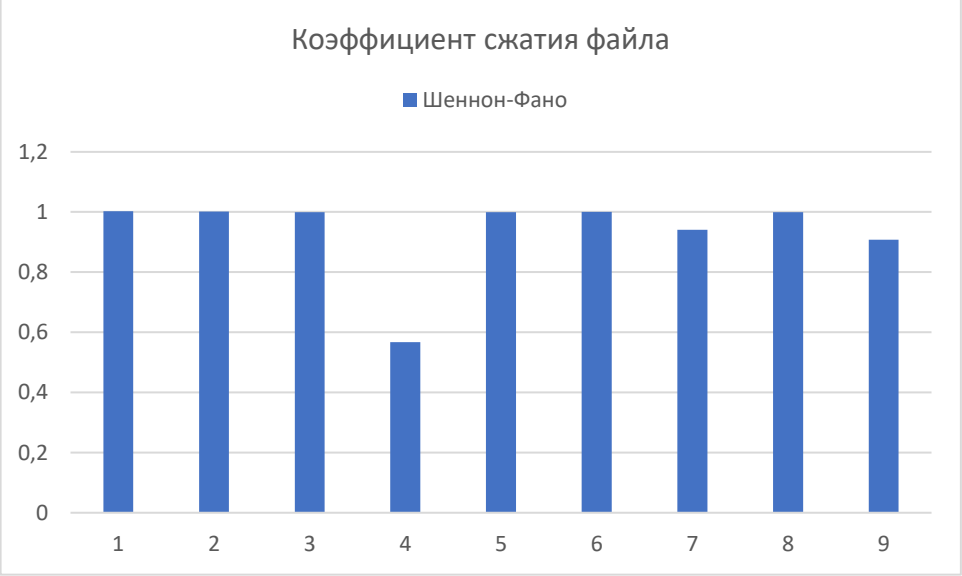

Рис. 11 – коэффициент сжатия файлов

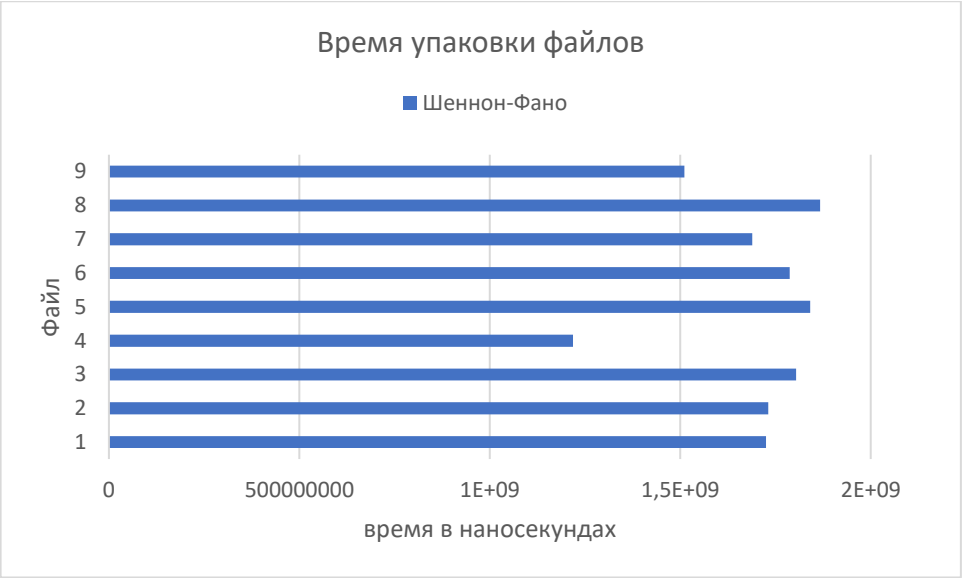

Рис.12 – время упаковки файлов в наносекундах

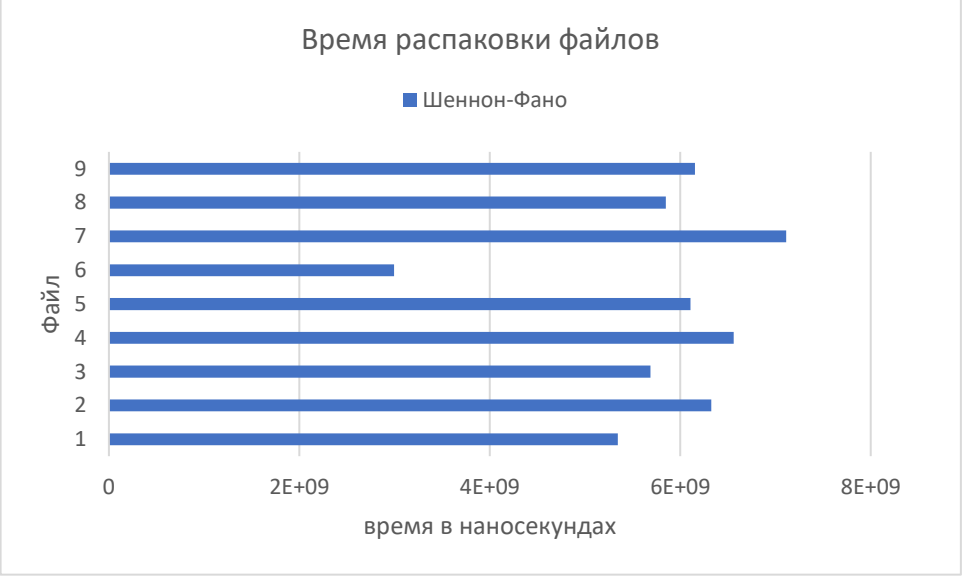

Рис.12 – время распаковки файлов в наносекундах

Имя файла	S1	H	Шеннон-Фано	
			S2	K
1.docx	2092534	7,99165	2097541	1,00239
2.pptx	2080708	7,98564	2084193	1,00167
3.pdf	2241312	7,94198	2240723	0,999737
4.txt	2081430	4,21107	1180796	0,5673
5.exe	2102104	7,96272	2100790	0,999375
6.jpg	2095025	7,94928	2096136	1,00053
7.bmp	2102438	7,47009	1979180	0,941374
8.jpg	2074467	7,93798	2074311	0,999925
9.bmp	2081838	7,20509	1889480	0,907602

Таб. 1 – коэффициенты сжатия файлов

Имя файла	S1	H	Шеннон-Фано	
			tu	tp
1.docx	2092534	7,99165	1,73E+09	6,15E+09
2.pptx	2080708	7,98564	1,73E+09	5,85E+09
3.pdf	2241312	7,94198	1,8E+09	7,11E+09
4.txt	2081430	4,21107	1,22E+09	2,99E+09
5.exe	2102104	7,96272	1,84E+09	6,11E+09
6.jpg	2095025	7,94928	1,79E+09	6,56E+09
7.bmp	2102438	7,47009	1,69E+09	5,69E+09
8.jpg	2074467	7,93798	1,87E+09	6,33E+09
9.bmp	2081838	7,20509	1,51E+09	5,34E+09

Табл. 2 – время упаковки/распаковки файлов

Сравнительный анализ алгоритмов по эффективности сжатия файлов разных типов, по скорости работы

Сразу можно заметить, что не всегда сжатие по алгоритму Шеннона-Фано работает положительно. Иногда может получиться так, что выходной файл больше исходного (коэффициент сжатия больше 1) (рис. 10).

При этом этот исход в основном связен не с алгоритмом, а с исходный файлом. Заметим то, что в файлах, где почти равномерное распределение частот символов коэффициент сжатия почти 1. А вот в некоторых всё наоборот: распределение неравномерное и коэффициент сжатия хороший (рис. 5, 8, 10, 11).

Алгоритм хоть и быстрый по упаковке, но распаковывает раза в 3 дольше.

Заключение

Некоторые из приведенных файлов уже имеют внутри себя сжатую структуру и наше сжатие может просто увеличить размер файла (например jpg), так как алгоритм не оптимален для такой структуры.

В наше время существует множество различных вариаций архиваторов (7-zip, WinRAR, Ark и т.д.). Каждый из них пытается использовать оптимальные алгоритмы упаковки и распаковки файлов. Эти алгоритмы должны учитывать разные факторы, в частности, как мы заметили, они должны подстроиться под входной файл.

Очевидно, что изначально алгоритмы не были оптимальны, но благодаря множеству тестов, они улучшаются.

Использованные источники

1. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. - М.: ДИАЛОГ-МИФИ, 2003.- 384 с.
2. Шилдт Герберт. C++. Базовый курс. – М.: Диалектика/Вильямс – 2019. – 624 с.