Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовая работа по курсу «Дискретный анализ»: Методы сжатия данных

Студент: А. М. Титеев
Преподаватель: Н. А. Зацепин
Группа: М8О-408Б
Дата:
Оценка:
Подпись:

Москва, 2020

Условие

Необходимо реализовать два известных метода сжатия данных для сжатия одного файла.

Формат запуска должен быть аналогичен формату запуска программы gzip, должны быть поддержаны следующие ключи: c, d, k, l, r, t, 1, 9. Должно поддерживаться указание символа дефиса в качестве стандартного ввода.

Метод решения

Как и требуется в условии запуск программы аналогичен запуску утилиты gzip: ./main <ключи> <файлы> <ключи> <файлы> ...

Обработка входных данных

На первом этапе работы программа определяет наличие в поступившей строке ключей, директорий и файлов. При обработке ключей учитывается их взаимоперекрываение, как в утилите gzip: l и r имеют наибольший приоритет, далее идёт ключ t, после чего остальные. В случае, если новый ключ перекрывает по логике утилиты некоторые из уже имеющихся, то эти ключи деактивируются.

Если полученное слово из стандартного ввода не является ключом, то программа проверяет наличие директории с таким именем. Если такой директории нет, то считается, что это имя файла, и оно заносится в список файлов. Если директория с таким именем существует и подключён ключ r, то все файлы внутри этой директории добавляются в список.

После с файлами ведётся работа согласно введённым ключам.

Арифметическое кодирование

.

LZ77

Описание файлов программы

Код программы разбит на 11 файлов:

1. Arithmetic.h - Содержит перечисление методов и описание класса TArithmetic, необходимого для работы арифметической компрессии и декомпрессии.

2. Arithmetic.cpp - Содержит реализацию всех методов класса TArithmetic.

3. BFile.h - Содержит перечисление методов и описание класса TOutBinary и класса TInBinary, необходимых для записи в файл и чтения из файла соответственно.

4. BFile.cpp - Содержит реализацию всех методов классов TOutBinary и TInBinary.

5. Globals.h - Содержит в себе все необходимые глобальные переменные и библиотеки используемые несколькими файлами.
6. LZ77.h - Содержит перечисление методов и описание класса TLZ77, необходимого для работы алгоритма LZ77.
7. LZ77.cpp - Содержит реализацию всех методов класса TLZ77.
8. main_help.h - Содержит в себе перечисление и описание всех функций необходимых для препроцессинга перед началом работы алгоритмов компрессии и декомпрессии.
9. main_help.cpp - Содержит реализацию всех функций, необходимых для препроцессинга, описанных в файле main_help.h.
10. main.cpp - Содержит в себе алгоритм чтения файлов и ключей.
11. Makefile - Файл для сборки программы.

Основные типы данных

1. TArithmetic - класс, описывающий работу арифметического алгоритма компрессии и декомпрессии.
2. TOutBinary - класс обеспечивающий запись необходимого количества байт в файл.
3. TInBinary - класс обеспечивающий считывание необходимого количества байт из файла.
4. TLZ77 - класс, описывающий работу алгоритма LZ77.
5.

Описание методов и функций программы

Основные свойства и методы класса TArithmetic

public:

1.
2.
3.
4.
5.

6.

7.

8.

private:

1.

2.

3.

4.

5.

6.

7.

8.

Основные свойства и методы класса TOutBinary

public:

1.

2.

3.

4.

5.

6.

7.

8.

private:

1.

2.

3.

4.

5.

6.

7.

8.

Основные свойства и методы класса TInBinary

public:

1.

2.

3.

4.

5.

6.

7.

8.

private:

1.

2.

3.

4.

5.

6.

7.

8.

Основные свойства и методы класса TLZ77

public:

1.
2.
3.
4.
5.
6.
7.
8.

private:

1.
2.
3.
4.
5.
6.
7.
8.

Прочие функции

1. bool KeyManager(std::string) - Обрабатывает полученные ключи. В случае получения неизвестного ключа возвращает false, иначе true.
2. bool DifferensOfSizes(TInBinary*, std::string) - вывод для каждого файла размера сжатого, оригинального, коэффициента сжатия(%) и имя оригинального файла(ключ l). В случае повреждения. архива возвращает false, иначе true.
3. void WorkWithDirectory(std::string) - работает с директорией (ключ r).

4. void WorkWithFile(std::string) - работает с файлом (определяет наличие файла, принимает решение о компрессии или декомпрессии, выполняет прочие ключи).
5. bool IsDirectory(std::string, bool) - Проверяет, является ли файл директорией. Если файл является директорией, возвращает true, иначе false.
6. void PrintDirectoryErrors(std::string) - Уведомляет об ошибках.
7. bool IsArchive(std::string) - Проверяет, является ли файл архивом. Если файл является архивом, возвращает true, иначе false.
8. void Rename(std::string, std::string) - Изменяет название файла после успешной компрессии или декомпрессии.
9. void Delete(std::string) - Удаляет временный файл.
10. void MainDecompress(TInBinary*, std::string) - Отвечает за подготовку декомпрессинга.
11. void MainCompress(TInBinary*, std::string) - Отвечает за подготовку компрессинга.
12. unsigned long long int LZWCompress(TInBinary*, std::string, TOutBinary*) - Подготавливает LZW компрессию. Возвращает размер нового файла.
13. unsigned long long int LZ77Compress(TInBinary*, std::string, TOutBinary*) - Подготавливает LZ77 компрессию. Возвращает размер нового файла.
14. unsigned long long int ArithmeticCompress(TInBinary*, std::string) - Подготавливает арифметический компрессию. Возвращает размер нового файла.
15. void KeepSmall(unsigned long long int, unsigned long long int, unsigned long long int, std::string) - Сохраняет архив самого малого размера.
16. int main(int, char*) - Осуществляет чтение входных данных.

Исходный код

Тест производительности

Для тестирования производительности использовалась английская версия книги Война и мир, размером 3.2 Мб.

	Время архивации (с)	Время разъархивации (с)	Сжатый размер	Пиковое потребление памяти (Кб)	Коэффициент сжатия
ключ 1					
ключ 9					
нет ключей					
gzip					

Выводы

Благодаря выполнению данного проекта я научился основным принципам работы с файлами и директориями во время компрессии и декомпрессии. Так же я освоил основные правила и принципы работы компрессии и декомпрессии данных. Два построенных алгоритма дали мне необходимый базис навыков для работы с кастомными буферами. Так же я значительно улучшил свои навыки написания комплексных программ построения утилит, к примеру я научился реализовывать поддержку ключей и возможность работы нескольких алгоритмов.

Список литературы

1. Арифметическое кодирование - Arithmetic coding [Электронный ресурс]: ru.qwe.wiki URL: https://ru.qwe.wiki/wiki/Arithmetic_coding (дата обращения 28.08.2020)

2. Arithmetic Coding [Электронный ресурс]: users.cs.cf.ac.uk URL: https://users.cs.cf.ac.uk/Dave.Marshall/Multimedia/node213.html (дата обращения 16.09.2020)

3. LZ77 на C, реализация алгоритма LZ77 на C [Электронный ресурс]: algor.skyparadise.org URL: https://algor.skyparadise.org/read/14 (дата обращения 16.09.2020)