Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по лабораторной работе №5

по курсу «Защита информации»

на тему: «Алгоритм сжатия информации»

Студент ИУ7-73Б _____ Лагутин Д. В.

(Группа) (Подпись, дата) (Фамилия И. О.)

Преподаватель _____ Чиж И. С.

(Подпись, дата) (Фамилия И. О.)

Москва, 2023 г.

Содержание

Введение

Целью работы является разработка алгоритма сжатия информации LZW. Обеспечить сжатие и распаковку произвольного файла с использованием разработанной программы, рассчитывать коэффициент сжатия. Предусмотреть работу программы с пустым, однобайтовым файлом.

Задачи:

1) изучить алгоритм сжатия;
2) разработать алгоритм работы программы;
3) реализовать и протестировать разработанное программное обеспечение.

1 Аналитическая часть

1.1 Сжатие информации

Сжатие информации — процесс уменьшения объема данных с сохранением возможности их полного восстановления.

При этом выделяют следующие классы сжатия информации:

1) без потери информации;
2) с потерей информации.

Преимущество методов сжатия с потерями над методами сжатия без потерь состоит в том, что первые делают возможной большую степень сжатия, продолжая удовлетворять поставленным требованиям, а именно — искажения должны быть в допустимых пределах чувствительности человеческих органов, физических чувств.

1.2 Алгоритм сжатия LZW

LZW — алгоритм сжатия без потерь. Основная идея метода заключается в замене наиболее часто встречаемых последовательностей байт на более короткие кодовые последовательности.

Кодовые последовательности получаются как двенадцатибитные индексы соответствующих записей в таблице замен и потому могут использоваться повторно.

1.2.1 Сжатие

Как следует из описания, сжатое сообщение получается из исходного последовательной заменой по таблице. Изначально таблица заполнена 256

возможными символами из таблицы ASCII.

На каждой итерации алгоритма ищется наибольшая подстрока, содержащаяся в таблице. Соответствующее кодовая последовательность заносится в результат, подстрока, состоящая из найденной и следующего символа добавляется в таблицу.

1.2.2 Распаковка

Преимуществом данного алгоритма является то, что таблица замены может быть получена и в ходе распаковки, поэтому она не записывается в результирующее сообщение. Это достигается за счет того, что аналогичные записи могут быть построены конкатенацией предыдущей записи с первым символом найденной. При этом возможна ситуация, когда подстрока не может быть заменена, что происходит, когда замена должна производится индексом, полученным на текущей итерации, поэтому значение формируется сложением предыдущей записи с ее первым символом.

2 Конструкторская часть

2.1 Схема алгоритма

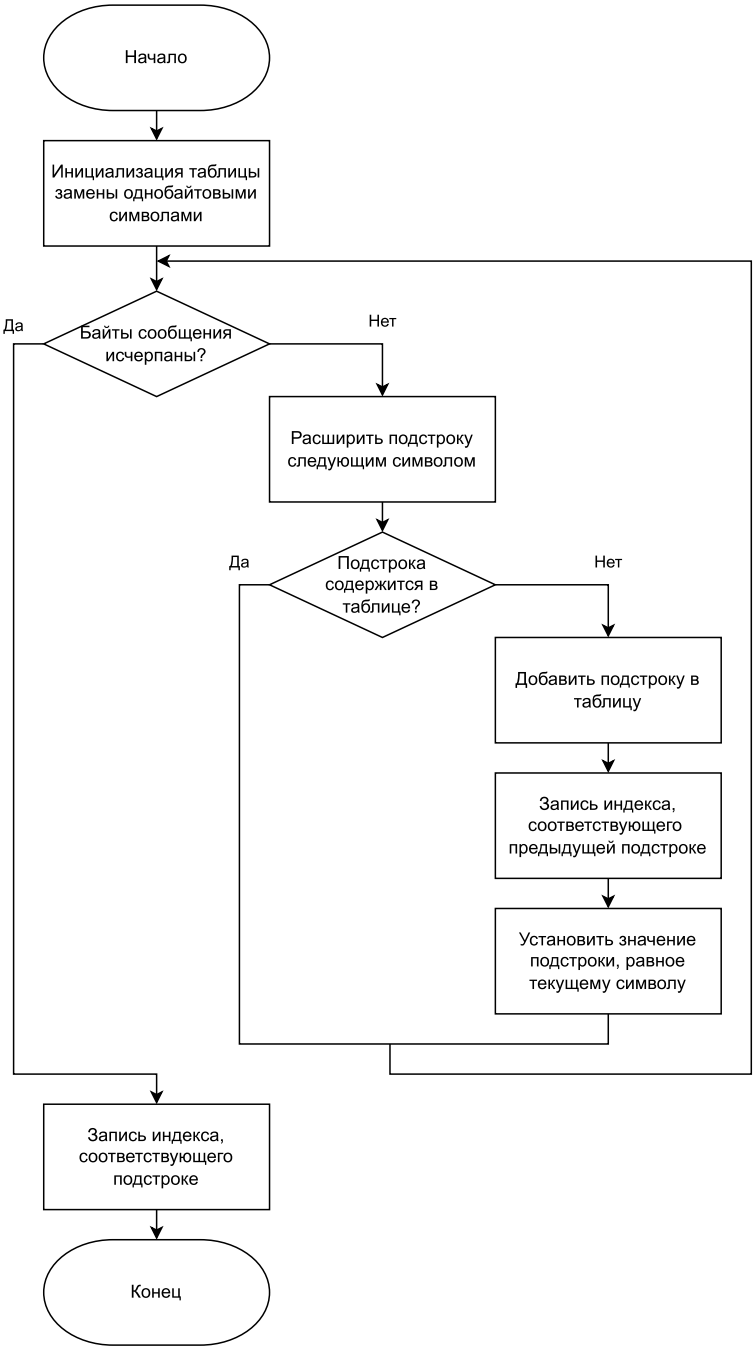

Рисунок 2.1 – Алгоритм сжатия

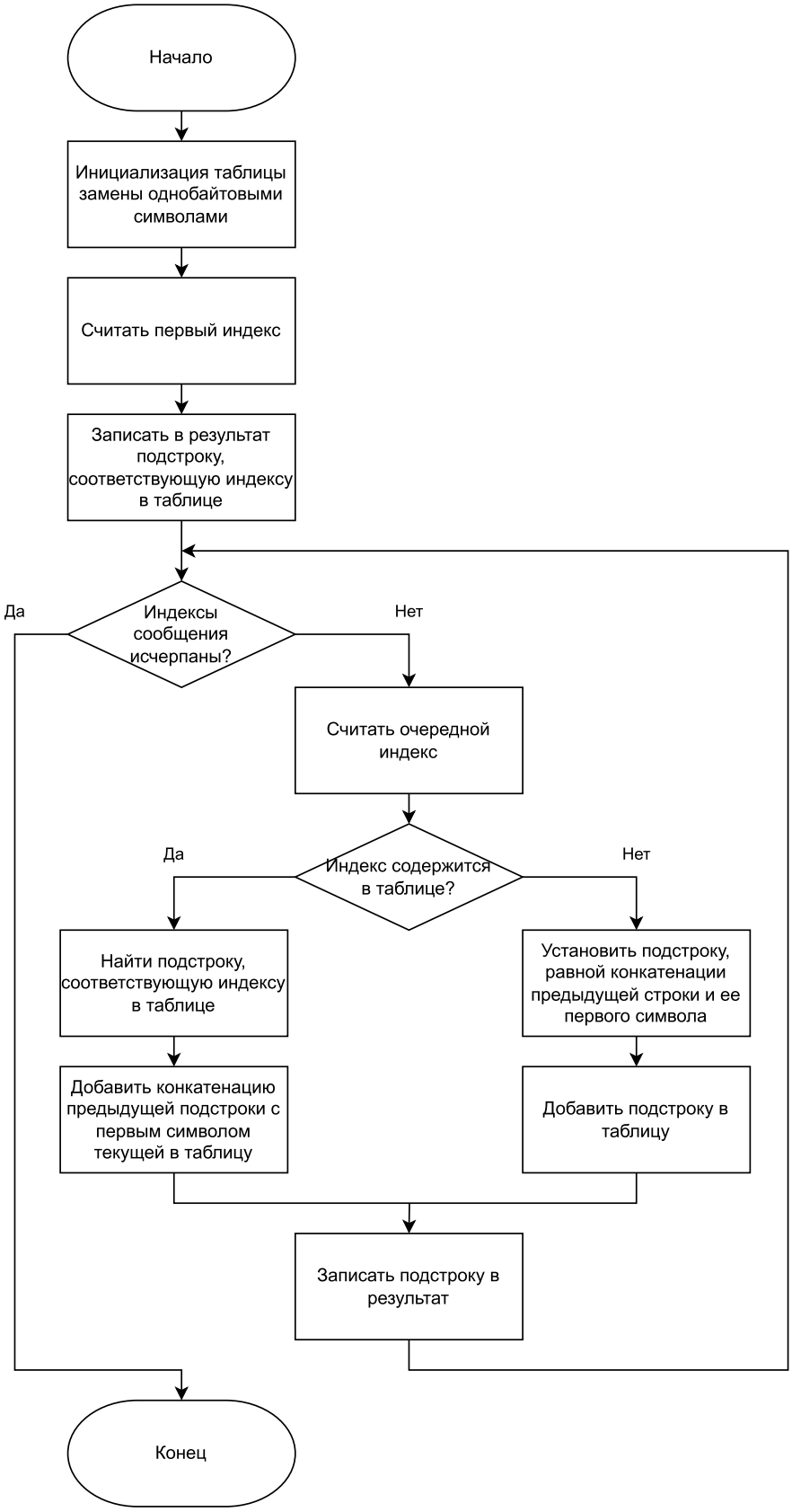

Рисунок 2.2 – Алгоритм распаковки

3 Технологическая часть

3.1 Описание программного обеспечения

Для реализации машины использовался язык C++.

Листинг 3.1 – Метод, реализующий сжатие

```
std::string LZW::compress(const std::string &origin)
{
    if (origin.empty())
        return origin;

    std::string out;
    WTable table;
    BufferWriter writer(out);
    std::string sequence = "", last = "";

    for (char c : origin)
    {
        last = sequence;
        sequence += c;

        if (!table.present(sequence))
        {
            writer.add(table.get(last));
            table.add(sequence);
            sequence = c, last = "";
        }
    }

    writer.add(table.get(sequence));
    writer.flush();

    return out;
}
```

Листинг 3.2 – Метод, реализующий распаковку

```
std::string LZW::decompress(const std::string &origin)
{
    if (origin.empty())
        return origin;

    std::string out;
    RTable table;
    BufferReader reader(origin);
    unsigned short code = reader.read();
    std::string previous, current;

    previous = table.get(code);
    out += previous;

    while (reader.rest())
    {
        code = reader.read();

        if (table.present(code))
        {
            current = table.get(code);
            table.add(previous + current[0]);
        }
        else // Handle mirror sequence c.*c.*c (skip of index)
        {
            current = previous + previous[0];
            table.add(current);
        }

        out += current;
        previous = current;
    }

    return out;
}
```

3.2 Тестирование

Таблица 3.1 – Тестирование алгоритма сжатия

№	Исходные данные	Ожидаемый результат	Фактический результат
1	Пустой файл	Пустой файл	Пустой файл
2	<31>	<31><00>	<31><00>
3	<31><31><31> <31><31><31> <31>	<31><00><10> <01><11><03>	<31><00><10> <01><11><03>
4	<61><62><63>	<61><20><06> <63><00>	<61><20><06> <63><00>
5	<61><62><63> <61><62><63>	<61><20><06> <63><00><10> <63><00>	<61><20><06> <63><00><10> <63><00>

Таблица 3.2 – Тестирование алгоритма распаковки

№	Исходные данные	Ожидаемый результат	Фактический результат
1	Пустой файл	Пустой файл	Пустой файл
2	<31><00>	<31>	<31>
3	<31><00><10> <01><11><03>	<31><31><31> <31><31><31> <31>	<31><31><31> <31><31><31> <31>
4	<61><20><06> <63><00>	<61><62><63>	<61><62><63>
5	<61><20><06> <63><00><10> <63><00>	<61><62><63> <61><62><63>	<61><62><63> <61><62><63>
6	<31>	Ошибка	Ошибка

Заключение

Была разработана программа сжатия информации с использованием алгоритма LZW.

Были решены следующие задачи:

1) изучен алгоритм сжатия;
2) разработан алгоритм работы программы;
3) реализовано и протестировано разработанное программное обеспечение.