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 работа №5

по курсу «Защита информации»

на тему: «Алгоритм сжатия LZW»

Вариант № 1

Студент ИУ7-73Б (Группа) _____ (Подпись, дата) Лысцев Н. Д. (И. О. Фамилия)

Преподаватель _____ (Подпись, дата) Чиж И. С. (И. О. Фамилия)

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **3**

1 Аналитическая часть **4**

2 Конструкторский раздел **5**

3 Технологический раздел **6**

3.1 Требования к программному обеспечению 6

3.2 Средства реализации 6

3.3 Сведения о модулях программы 6

3.4 Тестирование 6

3.5 Реализации алгоритмов 6

ЗАКЛЮЧЕНИЕ **10**

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ **11**

ВВЕДЕНИЕ

Целью данной лабораторной работы является реализация программы сжатия документа.

Для достижения поставленной цели необходимо решить следующие задачи:

1) описать алгоритм LZW;

2) выбрать средства программной реализации;

3) реализовать данный алгоритм.

1 Аналитическая часть

LZW – алгоритм сжатия, основывающийся на поиске схожих символов в файле.

Алгоритм состоит из следующих шагов:

Кодирование

1) Все возможные символы заносятся в словарь. Во входную фразу X заносится первый символ сообщения.

2) Считать очередной символ Y из сообщения.

3) Если Y — это символ конца сообщения, то выдать код для X, иначе:

4) Если фраза XY уже имеется в словаре, то присвоить входной фразе значение XY и перейти к Шагу 2,

5) Иначе выдать код для входной фразы X, добавить XY в словарь и присвоить входной фразе значение Y. Перейти к Шагу 2.

Декодирование

1) Все возможные символы заносятся в словарь. Во входную фразу X заносится первый код декодируемого сообщения.

2) Считать очередной код Y из сообщения.

3) Если Y — это конец сообщения, то выдать символ, соответствующий коду X, иначе:

4) Если фразы под кодом XY нет в словаре, вывести фразу, соответствующую коду X, а фразу с кодом XY занести в словарь.

5) Иначе присвоить входной фразе код XY и перейти к Шагу 2.

Вывод

В этом разделе был рассмотрен алгоритм сжатия LZW.

2 Конструкторский раздел

На рисунке 2.1 представлена схема алгоритма LZW.

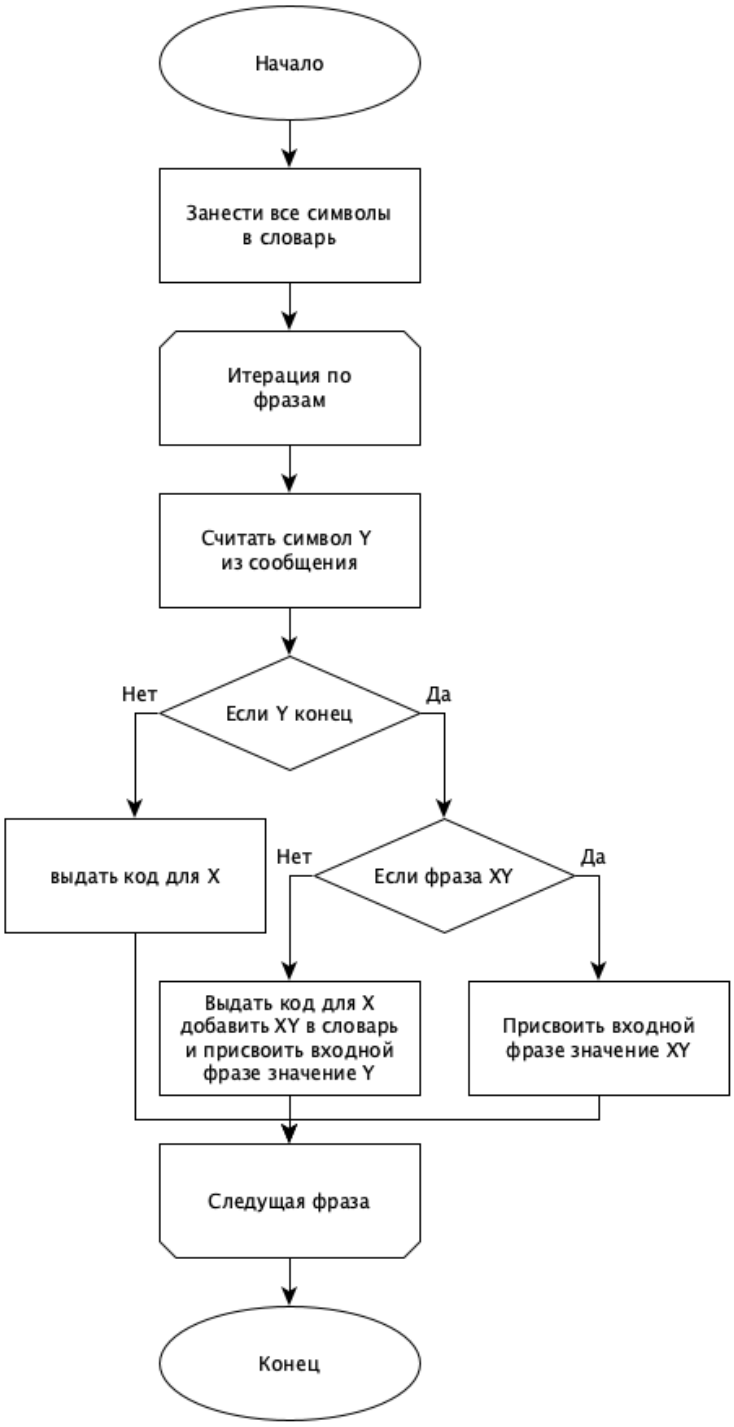

Рисунок 2.1 – Схема алгоритма сжатия LZW

Вывод

В данном разделе была представлена схема алгоритма LZW.

3 Технологический раздел

В данном разделе будут перечислены требования к программному обеспечению, средства реализации и листинги кода.

3.1 Требования к программному обеспечению

К программе предъявляется ряд требований:

- программа должна предоставлять возможность сжатия и разжатия произвольного файла;
- программа должна рассчитывать коэффициент сжатия;
- программа должна корректно работать с пустым однобайтовым файлом;

3.2 Средства реализации

В качестве языка программирования для этой лабораторной работы был выбран $C++$ [1].

3.3 Сведения о модулях программы

Программа состоит из трех модулей:

- `main.cpp` — файл, содержащий точку входа в программу;
- `LZW.cpp` — модуль, реализующий алгоритм LZW;
- `TrieTree.cpp` — модуль, реализующий префиксное дерево.

3.4 Тестирование

Все тесты с файлами были успешно пройдены.

3.5 Реализации алгоритмов

На листинге 3.1 представлена функция, реализующая сжатие по алгоритму LZW.

Листинг 3.1 – Функция, реализующая сжатие по алгоритму LZW

```
vector<uint8_t> LZW::_compress(const vector<uint8_t> &data) {
    LZW::_initDict(this->_dict);


    vector<pair<uint32_t, uint8_t>> tmpResult;

    vector<uint8_t> s;

    for (uint8_t currByte: data) {
        vector<uint8_t> tmp = s;
        tmp.push_back(currByte);

        if (this->_dict.contains(tmp)) {
            s = tmp;
        } else {
            uint32_t code = this->_dict.encode(s);
            tmpResult.emplace_back(code,
               LZW::_getBitsToRepresentInteger(code));
            this->_dict.insert(tmp);
            s = vector<uint8_t>(1, currByte);
        }
    }

    uint32_t code = this->_dict.encode(s);
    tmpResult.emplace_back(code,
       LZW::_getBitsToRepresentInteger(code));

    return LZW::_vec32ToVec8(tmpResult);
}
```

Листинг 3.2 – Функция, реализующая разжатие по алгоритму LZW (начало)

```
vector<uint8_t> LZW::_decompress(const vector<uint8_t> &data) {
    LZW::_initDict(this->_dict);
    vector<vector<uint8_t>> tmpKeys;
    LZW::_initVectorKeys(tmpKeys);

    vector<uint8_t> result;

    vector<uint8_t> s;

    uint32_t prevCode = codes[0];
    vector<uint8_t> entry = this->_getKeyByCode(tmpKeys,
       prevCode);
    result.insert(result.end(), entry.begin(), entry.end());

    for (int i = 1; i < data.size(); ++i) {

        uint32_t currCode = codes[i];
        entry = this->_getKeyByCode(tmpKeys, currCode);

        if (!entry.empty()) {
            result.insert(result.end(), entry.begin(),
               entry.end());

            uint8_t ch = entry[0];

            vector<uint8_t> tmp = {ch};
            vector<uint8_t> newEntry =
               this->_getKeyByCode(tmpKeys, prevCode);
            tmp.insert(tmp.begin(), newEntry.begin(),
               newEntry.end());

            this->_dict.insert(tmp);
            tmpKeys.push_back(tmp);

            prevCode = currCode;
        } else {
            vector<uint8_t> oldEntry =
               this->_getKeyByCode(tmpKeys, prevCode);
            oldEntry.push_back(oldEntry[0]);
```

Листинг 3.3 – Функция, реализующая разжатие по алгоритму LZW (конец)

```
            result.insert(result.end(), oldEntry.begin(),
               oldEntry.end());

            this->_dict.insert(oldEntry);
            tmpKeys.push_back(oldEntry);

            prevCode = currCode;
        }
    }

    return result;
}
```

Вывод

В данном разделе были перечислены требования к программному обеспечению, средства реализации и листинги кода.

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были решены следующие задачи:

1) описан алгоритм LZW;

2) выбраны средства программной реализации;

3) реализован данный алгоритм.

Цель работы, а именно реализация программы сжатия документа, также была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Справочник по языку C++ [Электронный ресурс]. — Режим доступа: https://learn.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=msvc-170 (дата обращения: 28.09.2022).