**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Отчет по домашнему заданию №2

**«Архивация данных»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Выполнил:** | |  | **Принял:** | |
| ФИО: | \_Цыпышев Т. А.\_\_\_\_\_ |  | ФИО: | \_Семёнов Д. В.\_\_\_\_\_\_ |
| Группа: | \_ИУ5-11Б\_\_\_\_\_\_\_\_\_\_ |  | Должность: | \_Преподаватель\_\_\_\_\_ |
| Дата: | \_20.01.2023\_\_\_\_\_\_\_\_ |  | Дата: | \_20.01.2023\_\_\_\_\_\_\_\_\_ |
| Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  | Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Москва, 2022 г.

**Описание алгоритма**

**Алгоритм:**

Пользователь вводит путь к оригинальному файлу, он передаётся в функцию Compress1. В ней создаётся словарь из 1-255 символов из таблицы ASCI. Далее программа начинает считывать байты из оригинального текста. Добавляем byte к слову. Если слова нет в словаре, то добавляем это слово в словарь, выводим номер слова без последнего байта в словаре и присваиваем слову значение byte. Выводим номер получившегося слова. Если это не конец файла, то делаем всё заново.

Иначе запускаем функцию Analysis. Эта функция будет разделять части выходного массива enc по количеству незначащих элементов в массиве. Мы проходимся по исходному массиву, если количество незначащих нулей в актуальном элементе больше максимального предыдущего, то мы вставляем в оригинальный массив ноль (является разделителем).

Далее спрашиваем у пользователя имя файла, куда сохранить закодированный текст и запускаем функцию Compress2. Она будет перекодировать закодированный массив без незначащих нулей и сохранять в файл. Мы проходим по закодированному массиву. Если актуальный элемент равен 0, то это разделитель и type равен кол-ву значащих битов в следующем элементе. Потом мы берём из буфера первые tmp\_len бит, если count + tmp\_len будет меньше 8, то мы просто прибавляем биты к byte, иначе добавляем часть, которая влезает в один байт, а оставшуюся добавим в последующих итерациях цикла. Потом сравниваем кол-во битов в byte и если она ровна 8, то сохраняем в файл. И так до конца закодированного массива.

**Описание функций**

1. **int main()** – здесь происходит вся магия, отсюда вызываются все функции
2. **int Find(int in[size\_dictionary][size\_word], int last\_position, int str[size\_word], int size)**

**Описание:**

Эта функция ищет первый массив str в массиве in.

**Переменные:**

* **int in[size\_dictionary][size\_word] –** двумерный массив, в котором нужно найти
* **int last\_position –** индекс первого незаписанного элемента в массиве **in**
* **int str[size\_word] –** массив, который нужно найти
* **int size –** индекс первого незаписанного элемента в массиве **str**

1. **int CountZero(int tmp)**

**Описание:**

Данная функция определяет кол-во незначащих битов в tmp.

**Переменные:**

* **int tmp –** переменная, у который определяют кол-во незначащих битов

1. **int CreateMask(int a, int b)**

**Описание:**

Данная фи-я создаёт битовую маску, которая в последующем будет применяться, для получения конкретной последовательности битов.

**Переменные:**

* **int a –** правая граница (с конца)
* **int b –** левая граница (с начала)

1. **int \*Analysis(int \*size\_enc, int enc[])**

**Описание:**

Данная функция добавляет разделители ('0') к enc при увеличении количества значащих битов у элемента массива.

**Переменные:**

* **int \*size\_enc –** указатель на размер массива enc
* **int enc[] –** массив, над которым будет происходить анализ

1. **int \*Compress1(char filename[], int \*index)**

**Описание:** фи-я читает и сжимает текст из файла 1 способом.

**Переменные:**

* **char filename[] –** путь к файлу
* **int \*index –** длина выходного массива

1. **void Compress2(char filename[], int size\_enc, int enc[])**

**Описание:** фи-я перекодирует ранее сжатые данные и записывает их в новый файл

**Переменные:**

* **char filename[] –** путь к файлу
* **int size\_enc –** длина входного массива
* **int enc[] –** ранее закодированный массив

**Описание переменных**

**#define size\_dictionary 100000 –** максимальное кол-во слов в словаре

**#define size\_out 1000000 –** максимальная длина массива с сжатыми данными

**#define size\_word 100 –** максимальная длина слов в словаре

**char filename[100] –** переменная, в которой храниться путь до файлов

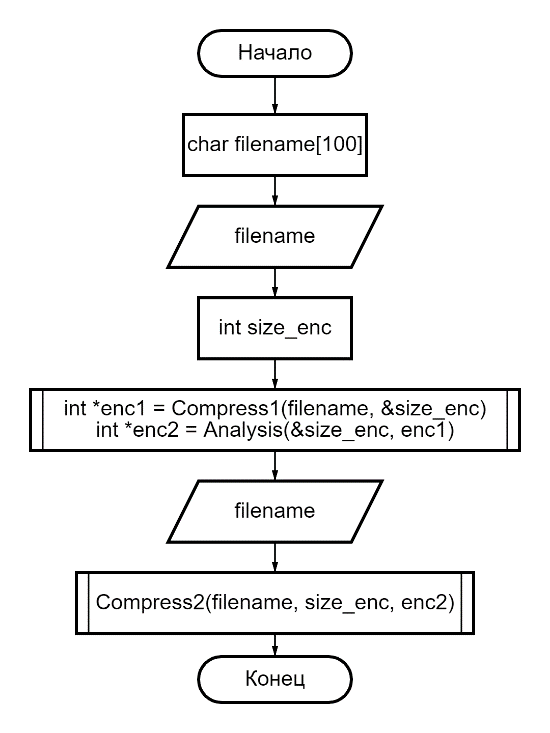
**int size\_enc –** длина сжатого массива данных

**int \*enc1 –** массив сжатых 1 способом данных

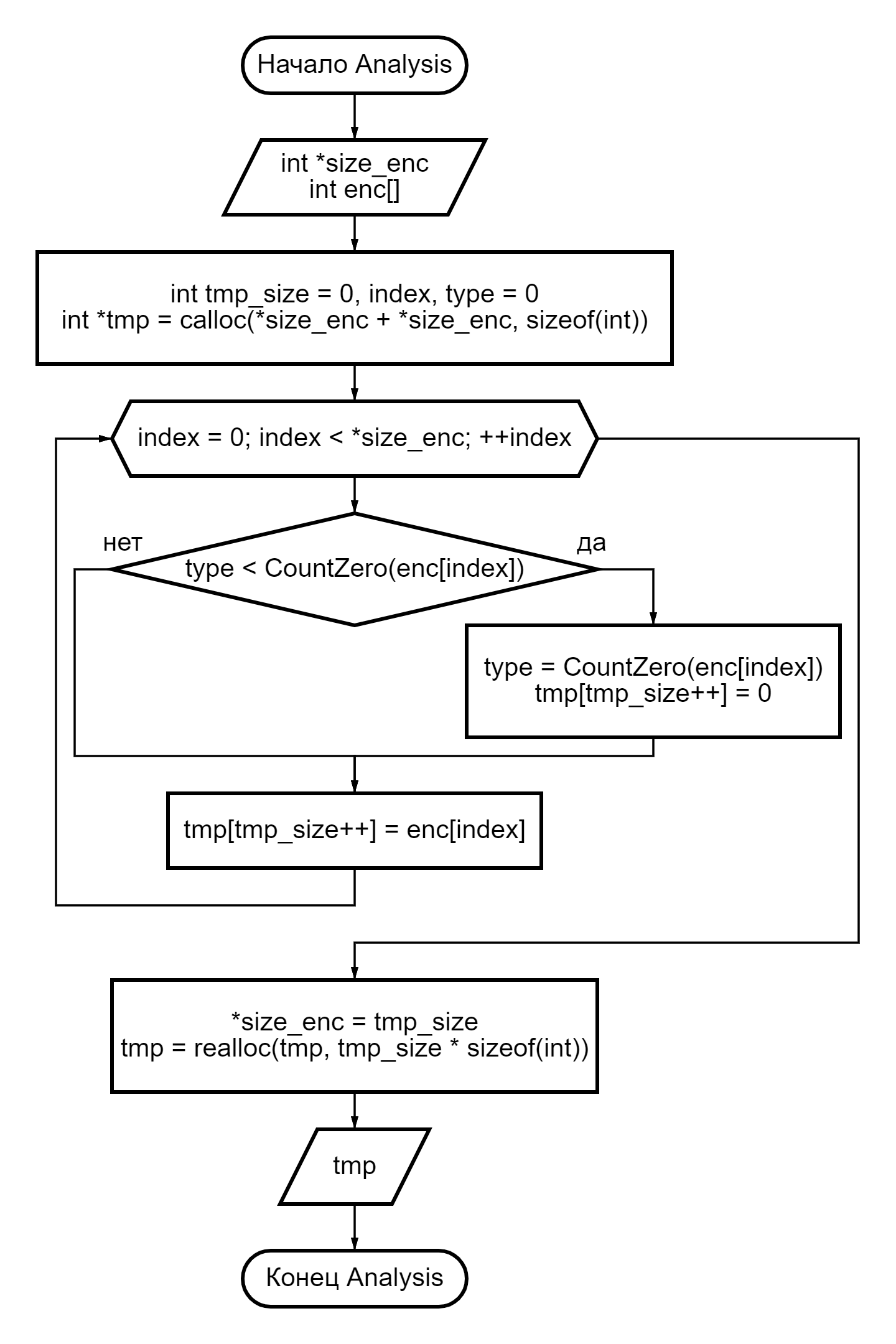
**int \*enc2 –** массив сжатых и проанализированных данных

**Схема алгоритма**

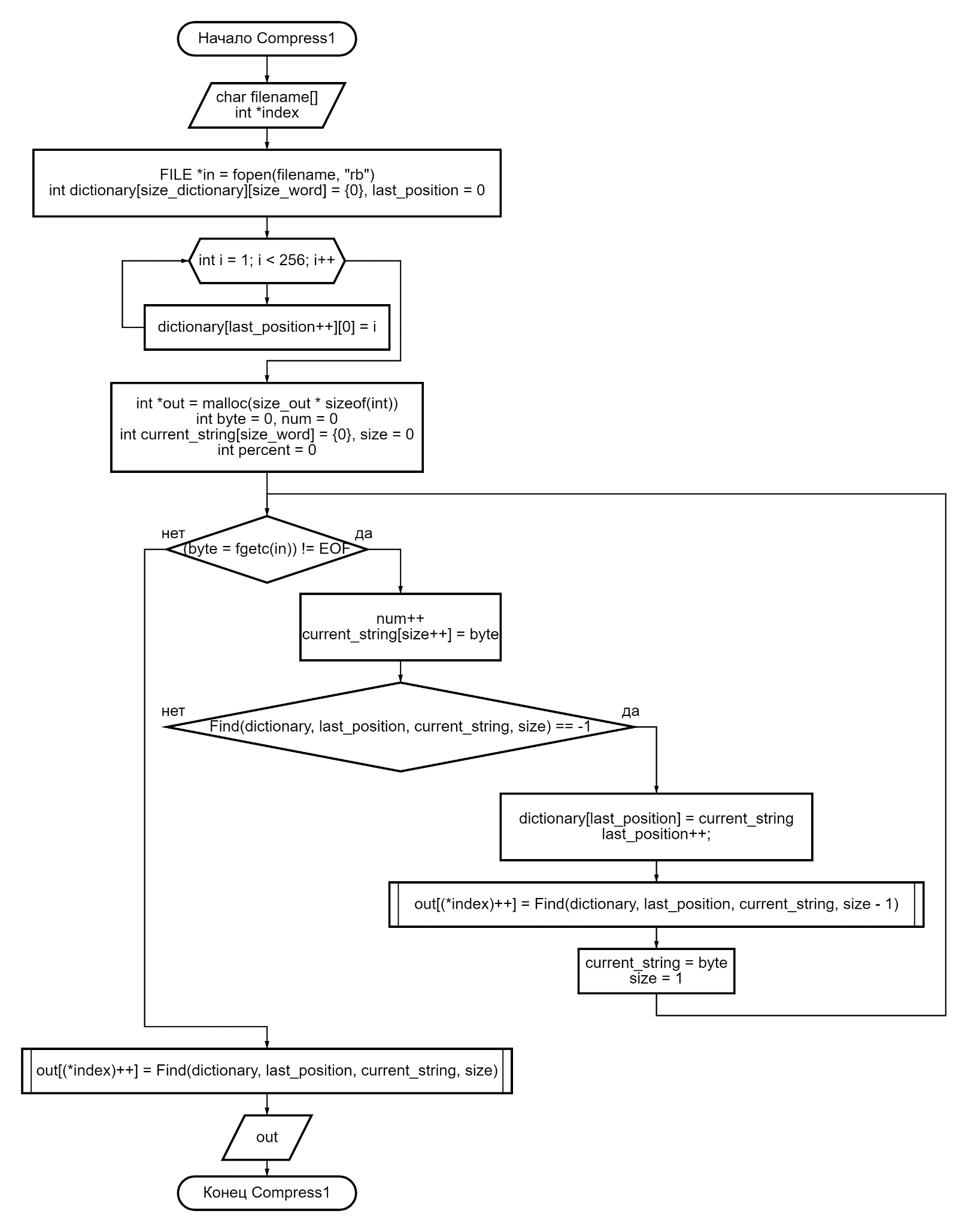
Main



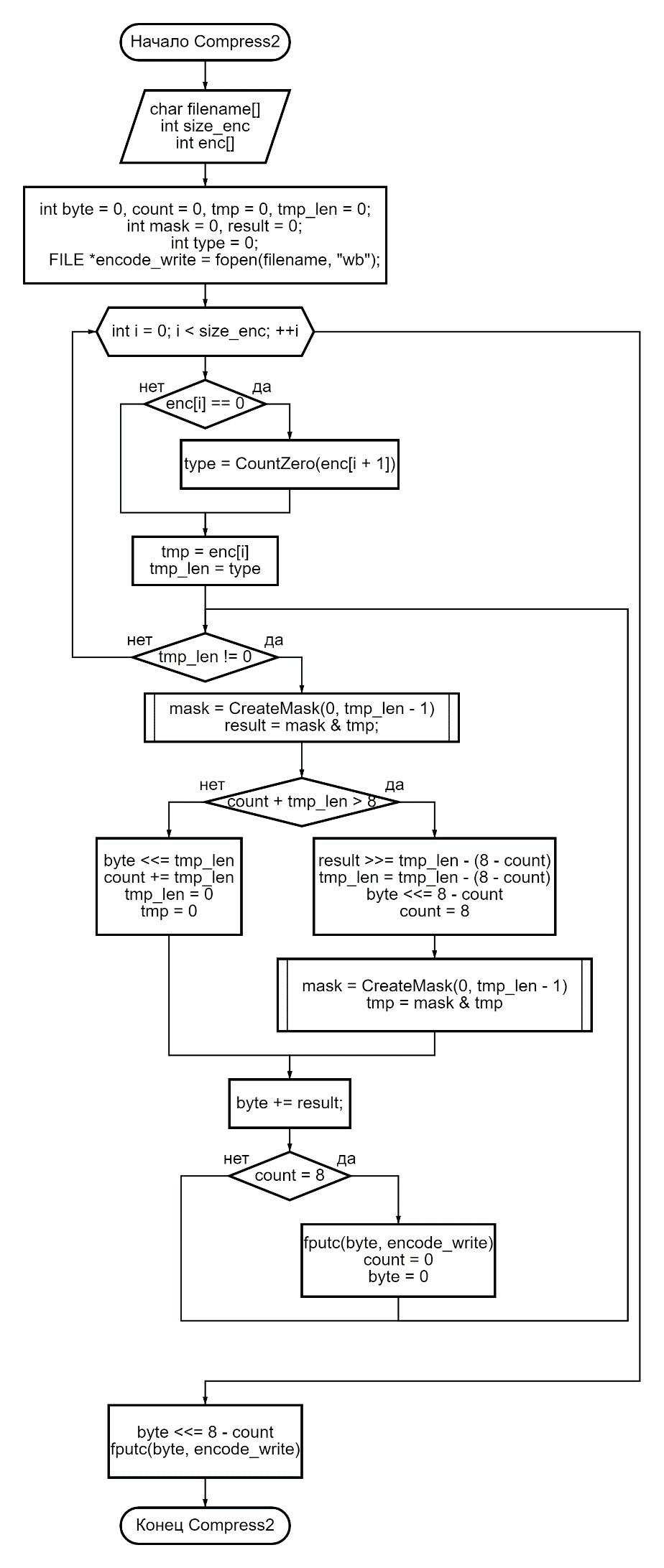
Analysis



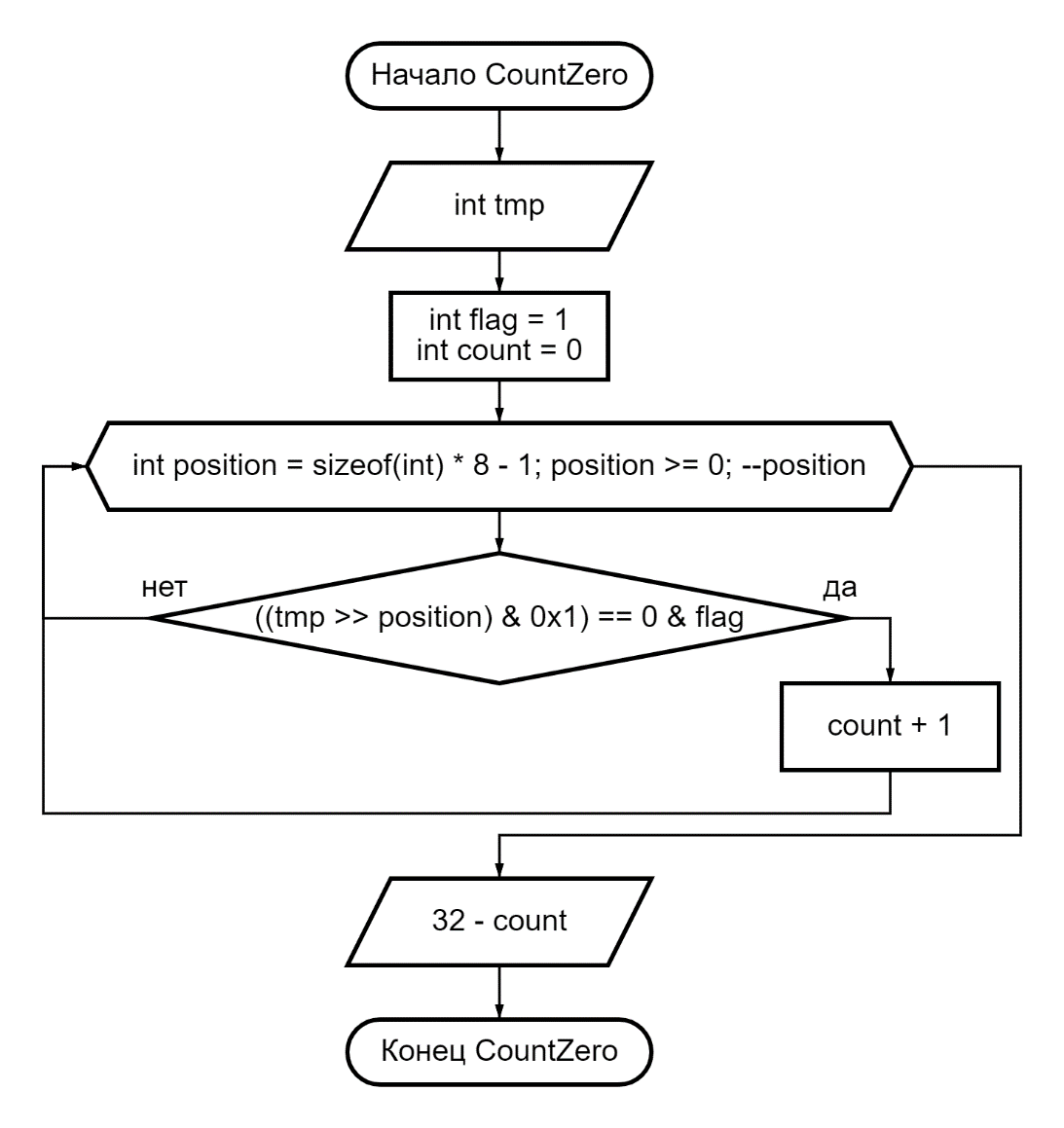
Compress 1



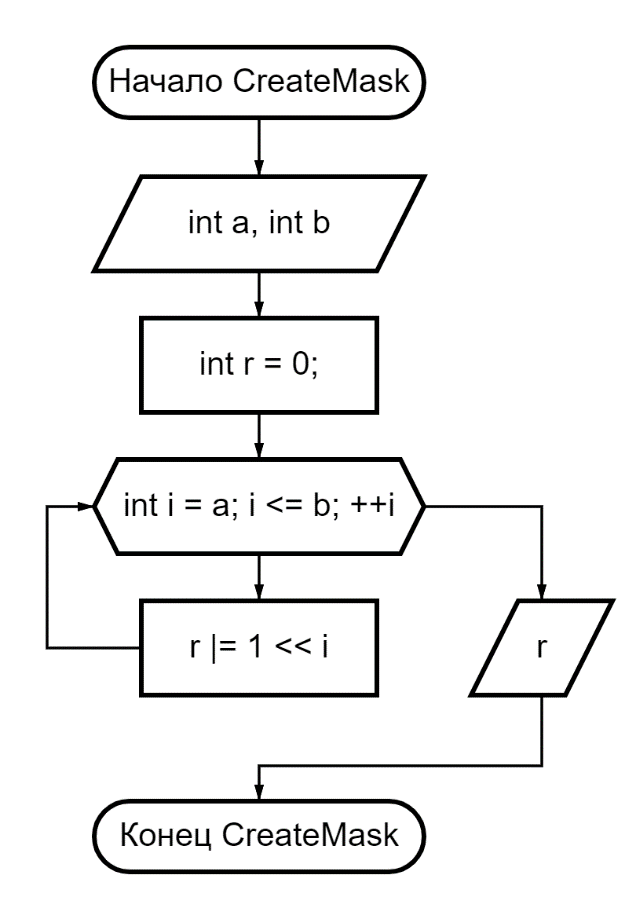
Compress 2



CountZero



CreateMask



Find

