ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доцент		В. А. Кузнецов
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 5.2

ОБРАБОТКА БИТОВЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

по курсу:

ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4326		Г. С. Томчук
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1 Постановка задачи

Задача: реализовать программную функцию на языке C/C++, выполняющую поставленную задачу. Вариант задания, пример входных и выходных данных представлен в таблице 1. Глобальные параметры использовать запрещено; допустимо использование дополнительных функций.

Обязательно использование поразрядных операций для выполнения задания, математические операции с индексами и счетчиками разрешены.

Таблица 1 – Вариант

N	Текст задания	Вход	Выход
2	**Упаковка слов энтропийного кода** Дана таблица кодовых слов CT. Кодовые слова имеют переменную длину. Каждое кодовое слово не превышает в размере и хранится в блоке 32 бит, для каждого слова указана его длина. Входная последовательность M представляет собой индексы кодовых слов, записанных в произвольной форме, количество кодовых слов также может быть передано. Реализовать функцию записи кодовых слов в битовую последовательность C так, чтобы кодовые слова с индексами M следовали друг за другом. Порядок записи бит в C произвольный, запись может начинаться со старших или младших бит.	CT[0]: 01, 2 CT[1]: 100, 3 CT[2]: 1011, 4 CT[3]: 1101, 4 CT[4]: 111101100, 9 M: 1,0,0,0,2,1,4,3,2,0,0,1	C: 10001010 11011100 11110110 01101101 10101100

2 Схема алгоритма решения

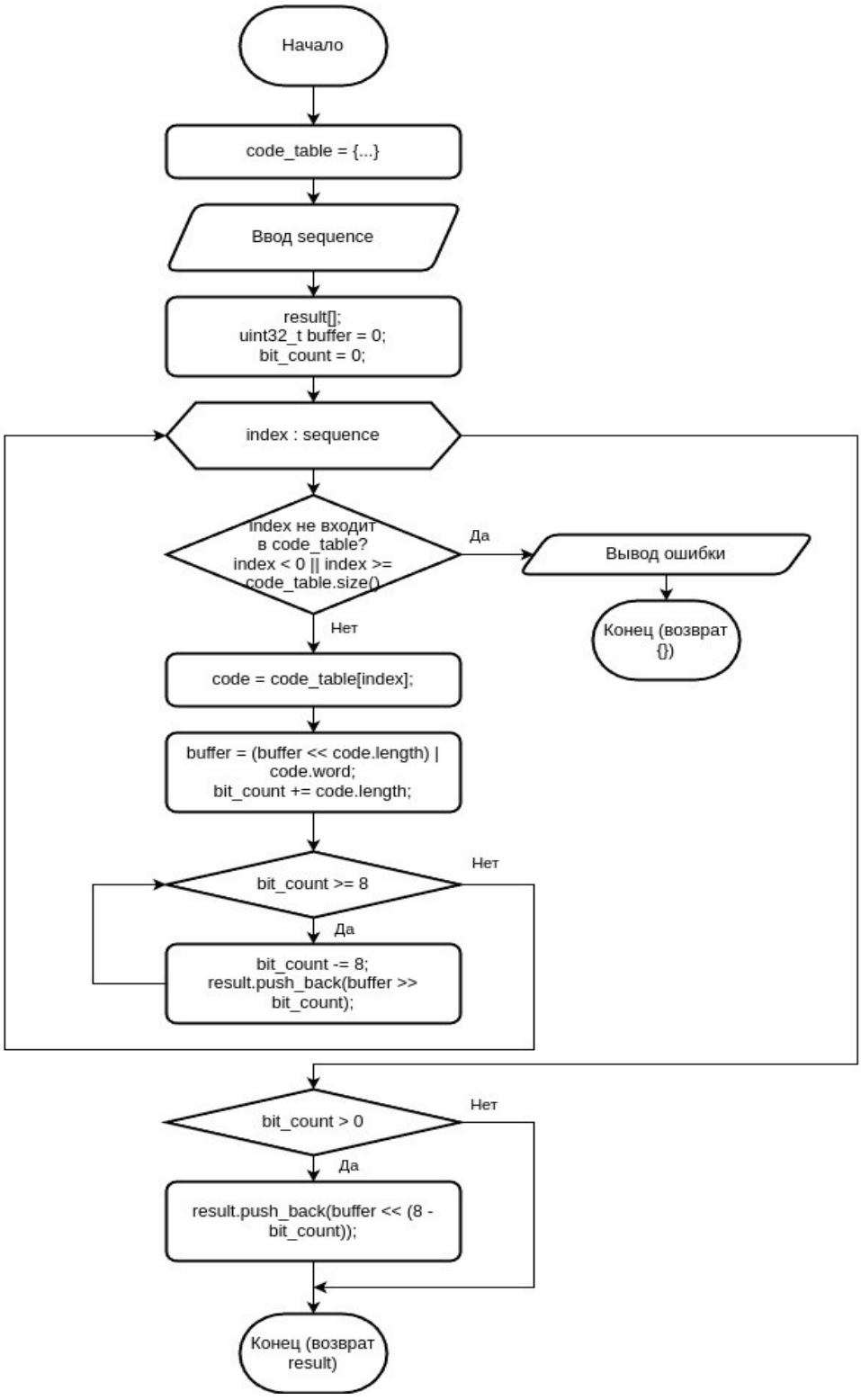

Рисунок 1 – Блок-схема алгоритма

3 Полное описание реализованных функций

3.1 pack_code_words

Функция pack_code_words упаковывает последовательность кодовых слов из таблицы в битовую последовательность. Принимает следующие аргументы:

1. const std::vector<CodeWord> &code_table: Таблица кодовых слов. Каждый элемент таблицы содержит кодовое слово и его длину.
2. const std::vector<int> &sequence: Последовательность индексов, указывающих, какие кодовые слова использовать для упаковки.

Возвращает std::vector<uint8_t> — вектор байтов, представляющих упакованную битовую последовательность. Работа функции происходит следующим образом:

1. Создает пустой вектор result для хранения результата.
2. Инициализирует переменную buffer (тип uint32_t) для временного хранения битов и переменную bit_count для отслеживания количества битов в буфере.
3. Проходит по каждому индексу в sequence
   3.1. Проверяет, что индекс находится в допустимом диапазоне. Если нет, выводит сообщение об ошибке и возвращает пустой результат.
   3.2. Извлекает кодовое слово из code_table по данному индексу.
   3.3. Добавляет кодовое слово в buffer путем сдвига и побитового ИЛИ.
   3.4. Увеличивает bit_count на длину добавленного кодового слова.
   3.5. Пока в buffer есть полный байт (8 бит), извлекает его и добавляет в result.
4. Если после обработки всех индексов остаются неиспользованные биты в buffer, добавляет их в result.
5. Возвращает result.

3.2 main

1. Инициализирует таблицу кодовых слов code_table.
2. Считывает последовательность индексов кодовых слов из ввода пользователя

3. Вызывает pack_code_words с таблицей кодовых слов и последовательностью индексов.
4. Если результат упаковки пустой, возвращает 1.
5. Выводит результат в виде битовых последовательностей по байтам.

4 Листинг программы

Листинг 1

```
#include <iostream>
#include <vector>
#include <bitset>
#include <sstream>

struct CodeWord {
    uint32_t word; // Кодовое слово, хранится в 32 битах
    int length; // Длина кодового слова в битах
};

std::vector<uint8_t> pack_code_words(const std::vector<CodeWord> &code_table,
const std::vector<int> &sequence) {
    std::vector<uint8_t> result;
    uint32_t buffer = 0; // Буфер для накопления бит
    int bit_count = 0; // Количество бит в буфере

    for (int index : sequence) {
        // Проверка индекса
        if (index < 0 || index ≥ code_table.size()) {
            std::cerr << "Ошибка: в таблице отсутствует слово с индексом " <<
index;
            return {};
        }

        const CodeWord &code = code_table[index];

        // Добавляем кодовое слово в буфер
        buffer = (buffer << code.length) | code.word;
        bit_count += code.length;

        // Пока в буфере есть полный байт, выгружаем его в результат
        while (bit_count ≥ 8) {
            bit_count -= 8;
            result.push_back(buffer >> bit_count);
        }
    }

    // Если остались неиспользованные биты, добавляем их в результат
    if (bit_count > 0)
        result.push_back(buffer << (8 - bit_count));

    return result;
}

int main() {
    // Таблица кодовых слов
    std::vector<CodeWord> code_table = {
            {0b01,        2},
            {0b100,       3},
            {0b1011,      4},
            {0b1101,      4},
            {0b111101100, 9}
    };
```

Продолжение листинга 1

```
    // Последовательность индексов кодовых слов
    std::vector<int> sequence;
    std::string input;
    std::cout << "M: ";
    std::getline(std::cin >> std::ws, input);
    std::stringstream ss(input);
    std::string index;
    while (ss >> index) sequence.push_back(std::stoi(index));

    // Получаем упакованную битовую последовательность
    std::vector<uint8_t> packed_bits = pack_code_words(code_table, sequence);
    if (packed_bits.empty()) return 1;

    // Выводим результат в виде битовых последовательностей по байтам
    for (uint8_t byte : packed_bits) {
        std::bitset<8> bits(byte);
        std::cout << bits << std::endl;
    }

    return 0;
}
```

5 Результаты тестирования программы

```
43  int main() {
44      // Таблица кодовых слов
45      std::vector<CodeWord> code_table = {
46              {0b01,        2},
47              {0b100,       3},
48              {0b1011,      4},
49              {0b1101,      4},
50              {0b111101100, 9}
51      };
```

```
Run:  main5.2
/home/grigorijtomczuk/Desktop/suai/op/cmake-build-debug/main5.2
M: 1 0 0 0 2 1 4 3 2 0 0 1
10001010
11011100
11110110
01101101
10101100

Process finished with exit code 0
```

Рисунок 2

```
43  int main() {
44      // Таблица кодовых слов
45      std::vector<CodeWord> code_table = {
46              {0b01,        2},
47              {0b100,       3},
48              {0b1011,      4},
49              {0b1101,      4},
50              {0b111101100, 9}
51      };
```

```
Run:  main5.2
/home/grigorijtomczuk/Desktop/suai/op/cmake-build-debug/main5.2
M: 4 2 3 3 4 1 0 1
11110110
01011110
11101111
10110010
00110000

Process finished with exit code 0
```

Рисунок 3

Рисунок 4

Рисунок 5