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 3](#__RefHeading___Toc801_316780651)

[2 Схема алгоритма решения 4](#__RefHeading___Toc809_316780651)

[3 Полное описание реализованных функций 5](#__RefHeading___Toc807_316780651)

[3.1 split\_words 5](#__RefHeading___Toc3907_3112333877)

[3.2 find\_indices 5](#__RefHeading___Toc3913_3112333877)

[3.3 transform\_to\_lower 6](#__RefHeading___Toc3911_3112333877)

[3.4 main 6](#__RefHeading___Toc3909_3112333877)

[4 Листинг программы 7](#__RefHeading___Toc805_316780651)

[5 Результаты тестирования программы 8](#__RefHeading___Toc803_316780651)

1. Постановка задачи

Задача: реализовать программную функцию на языке C/С++, 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 |

1. Схема алгоритма решения

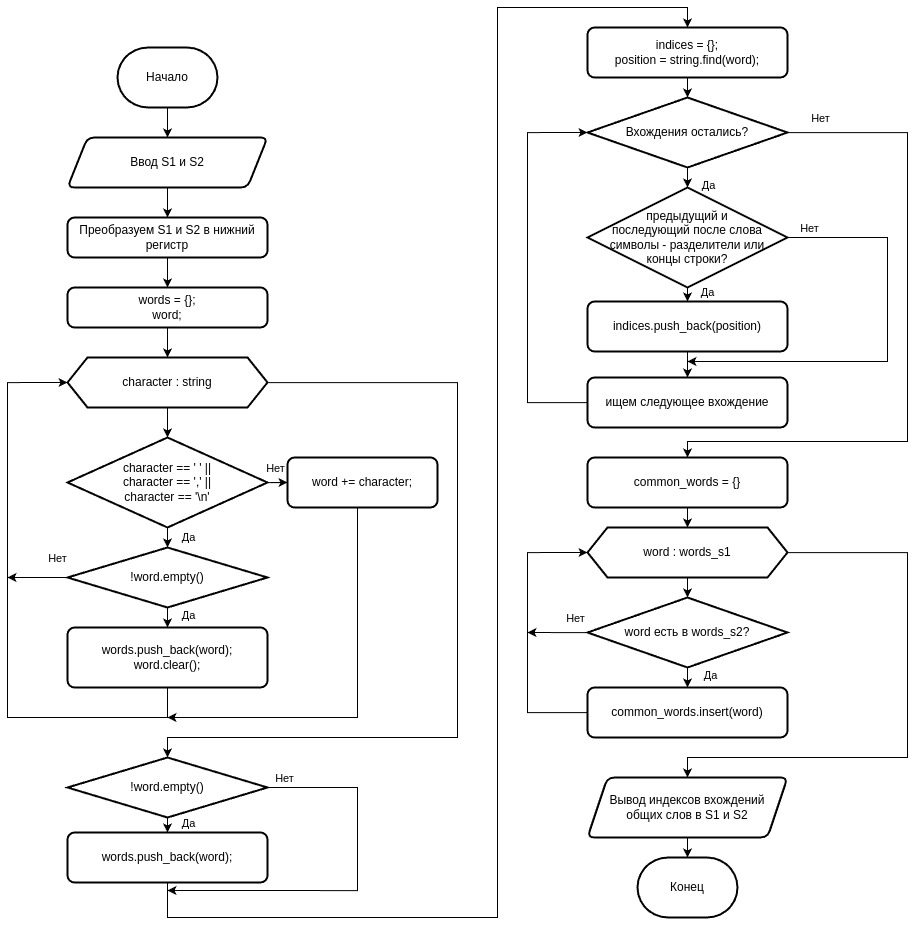


Рисунок 1 – Блок-схема алгоритма

1. Полное описание реализованных функций
   1. split\_words

Функция split\_words разделяет строку на слова, используя пробелы, запятые и переносы строк в качестве разделителей. Принимает следующие аргументы:

1. const std::string &string — строка, которую нужно разделить на слова.

Возвращает std::vector<std::string> — вектор строк, каждая из которых является словом из исходной строки. Работа функции происходит следующим образом:

1. Создаётся пустой вектор words и временная строка word.
2. Для каждого символа в строке проверяется, является ли он разделителем (пробел, запятая, перенос строки).
3. Если символ является разделителем и word не пуст, добавляем word в words и очищаем word.
4. Если символ не является разделителем, добавляем его к word.
5. В конце добавляем последнее слово в words, если оно не пустое.
   1. find\_indices

Функция find\_indices находит все индексы вхождения слова в строку. Принимает следующие аргументы:

1. const std::string &string — строка, в которой ищем вхождения слова.
2. const std::string &word — слово, индексы вхождения которого ищем.

Возвращает std::vector<int> — вектор индексов, где слово встречается в строке. Работа функции происходит следующим образом

1. Создаётся пустой вектор indices.
2. Используем string.find для поиска первого вхождения слова в строке.
3. Пока находятся вхождения, проверяем, является ли предыдущий и последующий после слова символы разделителями или концом строки.
4. Если условие выполняется, добавляем индекс начала слова в indices.
5. Ищем следующее вхождение, начиная с позиции position + 1.
   1. transform\_to\_lower

Функция transform\_to\_lower преобразует все символы строки в нижний регистр. Принимает следующие аргументы:

1. const std::string &string — строка, которую нужно преобразовать.

Возвращает std::string — строка, преобразованная в нижний регистр. Работа функции происходит следующим образом:

1. Копируем исходную строку в новую строку lower.
2. Используем std::transform для преобразования каждого символа строки в нижний регистр с помощью функции ::tolower.
   1. main
3. Программа запрашивает ввод строк S1 и S2 от пользователя.
4. Преобразуем S1 и S2 в нижний регистр с помощью transform\_to\_lower.
5. Разделяем S1 и S2 на слова с помощью функции split\_words.
6. Для каждого слова в S1 и S2 находим индексы вхождений в исходные строки, используя функцию find\_indices.
7. Находим слова, которые есть в обеих строках, используя unordered\_set.
8. Для каждого общего слова выводим индексы их вхождений в S1 и S2.
9. Листинг программы

Листинг 1

1. Результаты тестирования программы

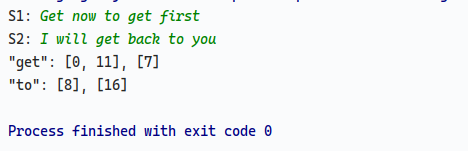


Рисунок 2

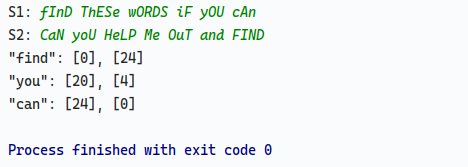


Рисунок 3

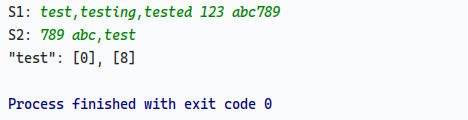


Рисунок 4

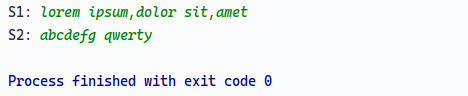


Рисунок 5

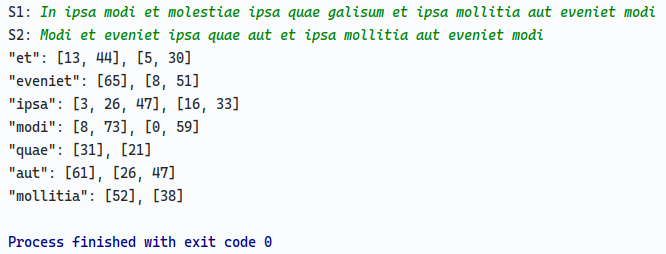


Рисунок 6