

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

Курсовая работа
по дисциплине «Программирование»
Тема: Обработка текстовой информации

Студент гр. 3312

Шарапов И. Д.

Преподаватель

Аббас С. А.

Санкт-Петербург
2023

Содержание

Цель работы	3
Задание (Вариант 51)	3
Постановка задачи и описание решения.....	3
Структура вызова функций	5
Описание переменных	5
Схема алгоритма.....	6
Текст программы.....	13
Контрольные примеры.....	16
Содержимое файлов.....	17
Примеры выполнения программы.....	19
Выводы	20

Цель работы

Законченное поэтапное решение содержательной задачи (постановка задачи, спецификация, выбор структур данных и разработка алгоритма, программная реализация, тестирование).

Задание (Вариант 51)

Ввести заданное количество ключевых слов и строку символов-разделителей. Затем вводится текст с неизвестным количеством строк. Ввод текста заканчивается, если после ввода строки в тексте окажется в любой последовательности все ключевые слова. Из строк введенного текста, в которых встречается хотя бы одно ключевое слово, удалить слово, имеющее минимальную длину. Вывести преобразованный текст.

Постановка задачи и описание решения

Решение предполагает считывание из файла и консоли: для этого у пользователя спрашивается режим и имя файла, в случае ввода из файла. Далее программа считывает количество ключевых слов и сами ключевые слова. В процессе считывания каждого слова, программа считает и запоминает хэш этого слова по следующему алгоритму.

Хэш – это число, полученное из слова, как результат хэш-функции. Каждому числу соответствует своё слово: у разных упорядоченных наборов символов разный хэш. Для вычисления хеша слова переберём по порядку все его символы. Каждый символ имеет числовой код, пусть это будет x . Возьмём два простых числа $r = 10^9 + 9$ и $t = 41$. Здесь r – модуль хэша, t – основание хэша. Заведём переменную $hash = 0$ и на каждом шаге перебора будем вычислять её значение по следующей формуле: $hash = (hash * t + x) \bmod r$. При таком алгоритме могут возникать так называемые коллизии – это когда два разных слова соответствуют одному числу. Это возникает из-за того, что мы берём значение по модулю r . Усовершенствуем наш алгоритм и заведём второй хэш $hash2$ с основанием $b = 47$. Теперь на каждом шаге мы будем также считать и

$hash2 = (hash2 * b + x) \bmod r$. Таким образом мы сводим вероятность коллизий к нулю.

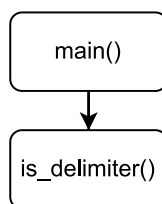
После считывания и обработки ключевых слов считаем строку разделителей. Далее начнём считывать текст построчно. Считав новую строчку, мы начнём перебирать её посимвольно и считать хэш между разделителями (слов). Когда мы находим разделитель (проверку на то, что символ разделитель выполним отдельной функцией), мы смотрим значение хэша «слова» и перебираем все ранее посчитанные хэши ключевых слов. Если хэши по обоим основаниям совпадают, то мы прибавляем 1 к счётчику этого ключевого слова. Также мы проверяем является ли это «слово» минимальным в строке. Если является, то мы запоминаем указатель на начало и конец этого слова. И обнуляем значение хэшей. Если символ не разделитель, то мы дальше продолжаем считать хэш по описанному ранее алгоритму. Обработав строчку нам нужно понять было ли в ней ключевое слово: для этого мы сложим все счётчики ключевых слов и сравним с тем значением, что у нас было на прошлом ходу. Если ничего не поменялось, то мы в массиве, в котором запоминали указатели на минимальное слово, присваиваем этим указателем значение на начало строки. И в самом конце работы над строчкой нам надо понять считывать ли дальше. Для этого переберём все счётчики ключевых слов, и если хотя бы какой-то из них равен нулю, то мы продолжим считывание.

После считывания и обработки строк осталось их только вывести. Спросим у пользователя тип вывода (в файл или консоль). Начнём перебирать все строки и выводить их элементы (символы). Если указатель на текущий элемент совпадёт с указателем на начало минимального слова, то мы перенесём указатель на конец слова и продолжим вывод символов. Если вывод был в файл, то выведем в консоль сообщение об успешном завершении работы программы.

Проверка слов на равенство с использованием хэшей, по сравнению с посимвольным сравнением, имеет лучшую асимптотику. $O(K * T + N * K)$

против $O(N^2 * K * T)$, где K , T – количество и длина ключевых слов, N – длина текста в символах.

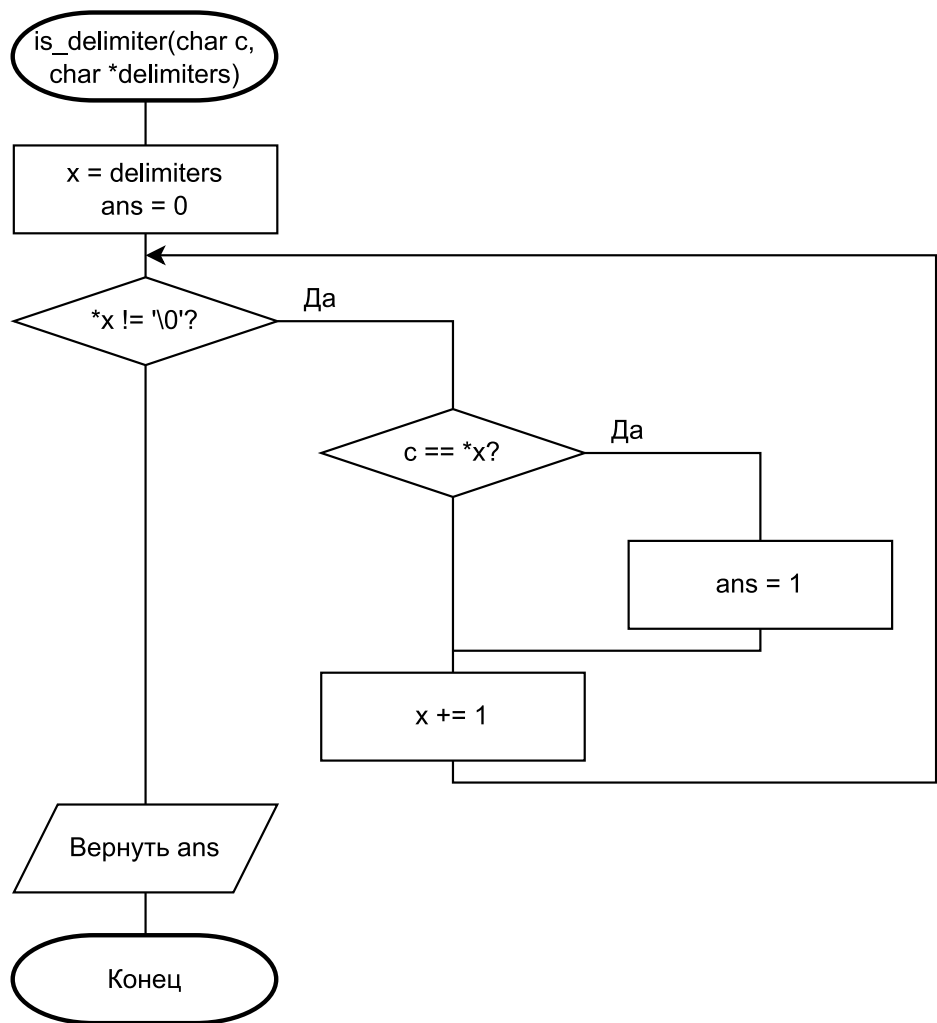
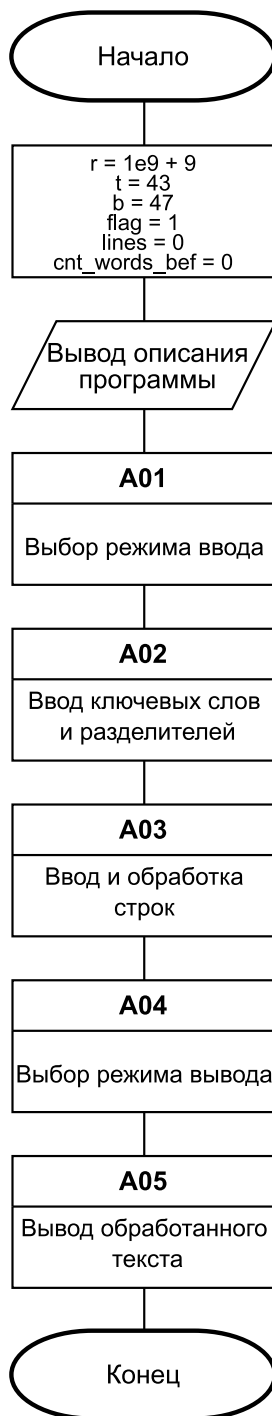
Структура вызова функций

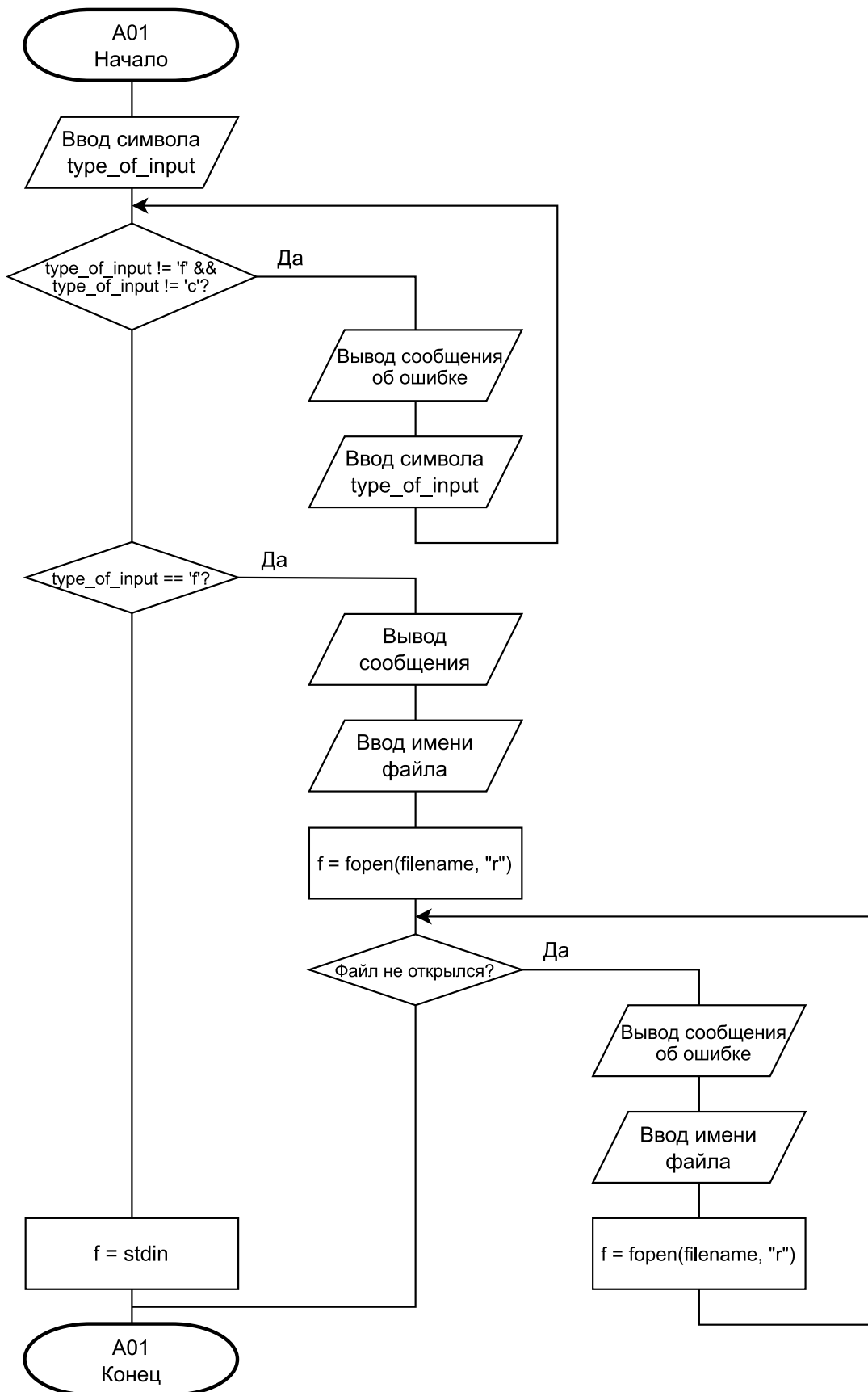


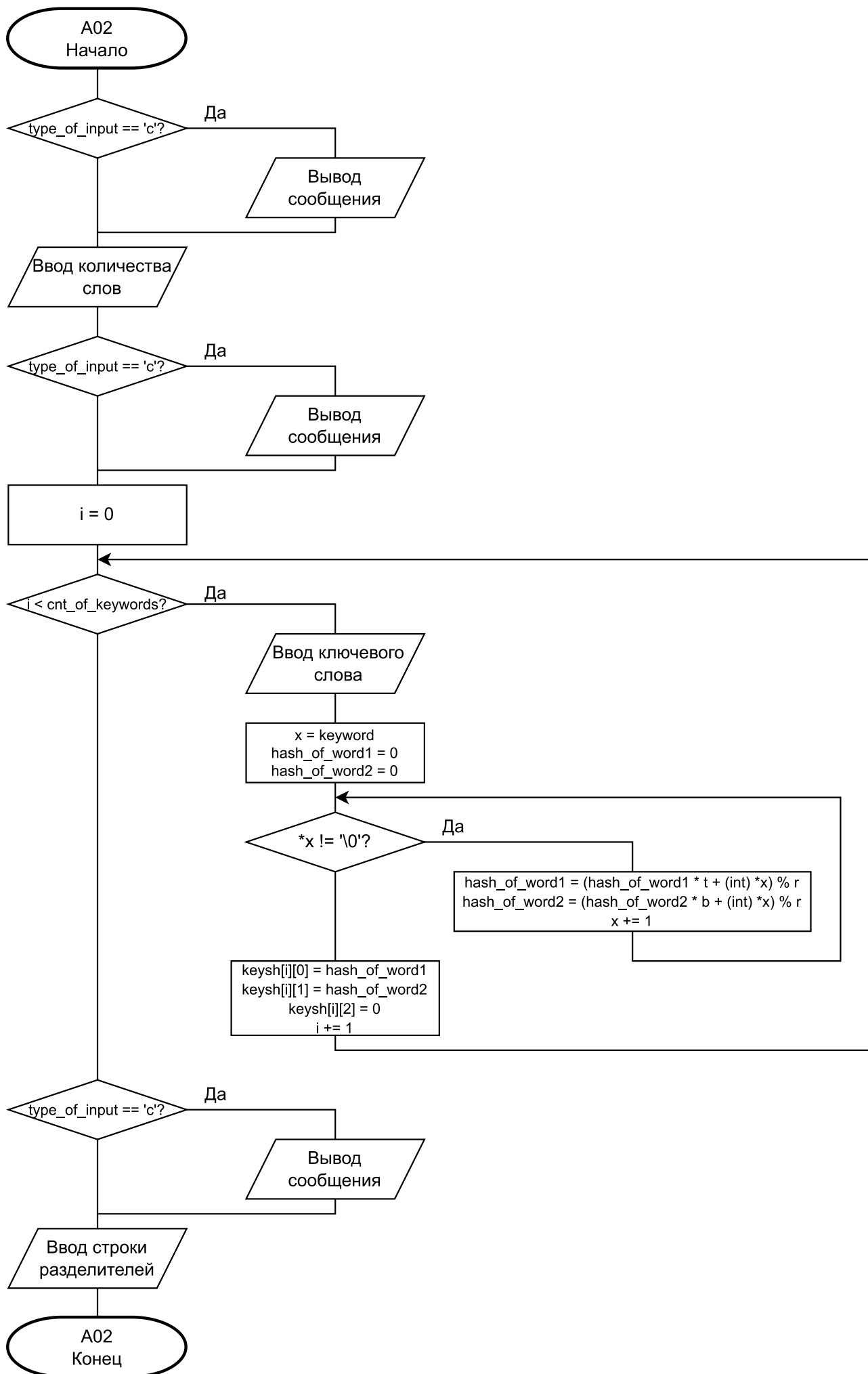
Описание переменных

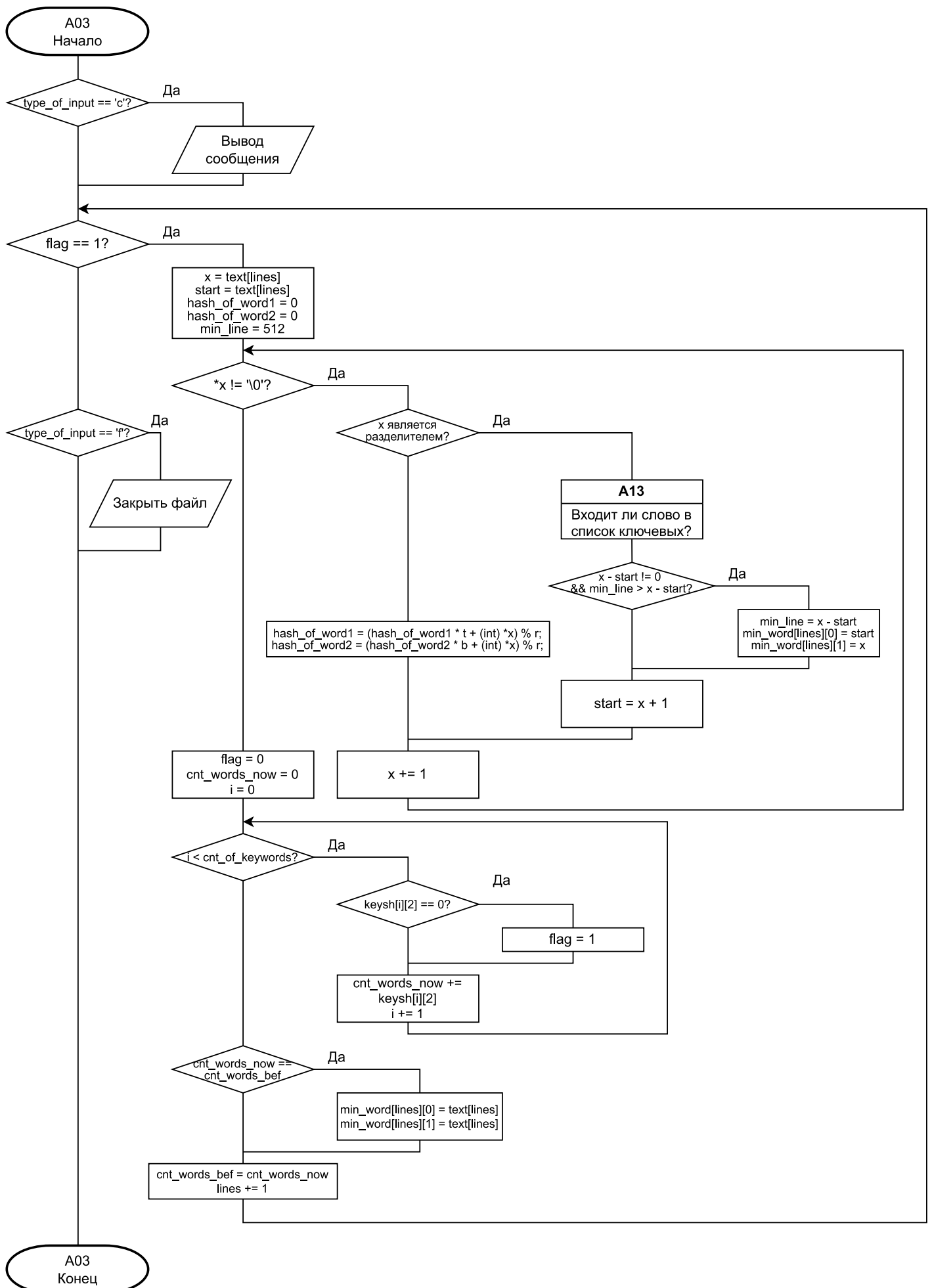
Функция <i>int is_delimiter(char c, char *delimiters)</i>			
№	Имя переменной	Тип	Назначение
1	c	char	Проверяем: есть ли этот символ в массиве
2	delimiters	char*	Массив разделителей
3	x	char*	Указатель для перебора всех элементов
4	ans	int	Логическая переменная
Функция <i>int main()</i>			
№	Имя переменной	Тип	Назначение
1	r	int	Модуль хеширования
2	t	int	Первое основание хеша
3	b	int	Второе основание хеша
4	type of input	char	Символьная переменная для типа ввода
5	filename	char[]	Имя файла
6	f	file	Указатель на файл
7	cnt of keywords	int	Количество ключевых слов
8	keyword	char[]	Ключевое слово
9	x	char*	Указатель на символ
10	hash of word1	long long	Первое значения хеша
11	hash of word2	long long	Второе значения хеша
12	keysh	int[][]	Двумерный массив для хранения значений хеша и счетчиков
13	delimiters	char[]	Массив для хранения разделителей
14	flag	int	Флаг для ограничения ввода
15	lines	int	Количество строк
16	text	char[][]	Двумерный массив для хранения текста
17	start	char*	Указатель на начало слова
18	min_line	int	Минимальная длина слова в строке
19	cnt_words_bef	int	Количество найденных слов до этого момента
20	cnt words now	int	Количество найденных слов сейчас
21	min_word	char*[][]	Двумерный массив для хранения указателей на начало и конец минимального слова в каждой строке
22	type of output	char	Символьная переменная для типа вывода

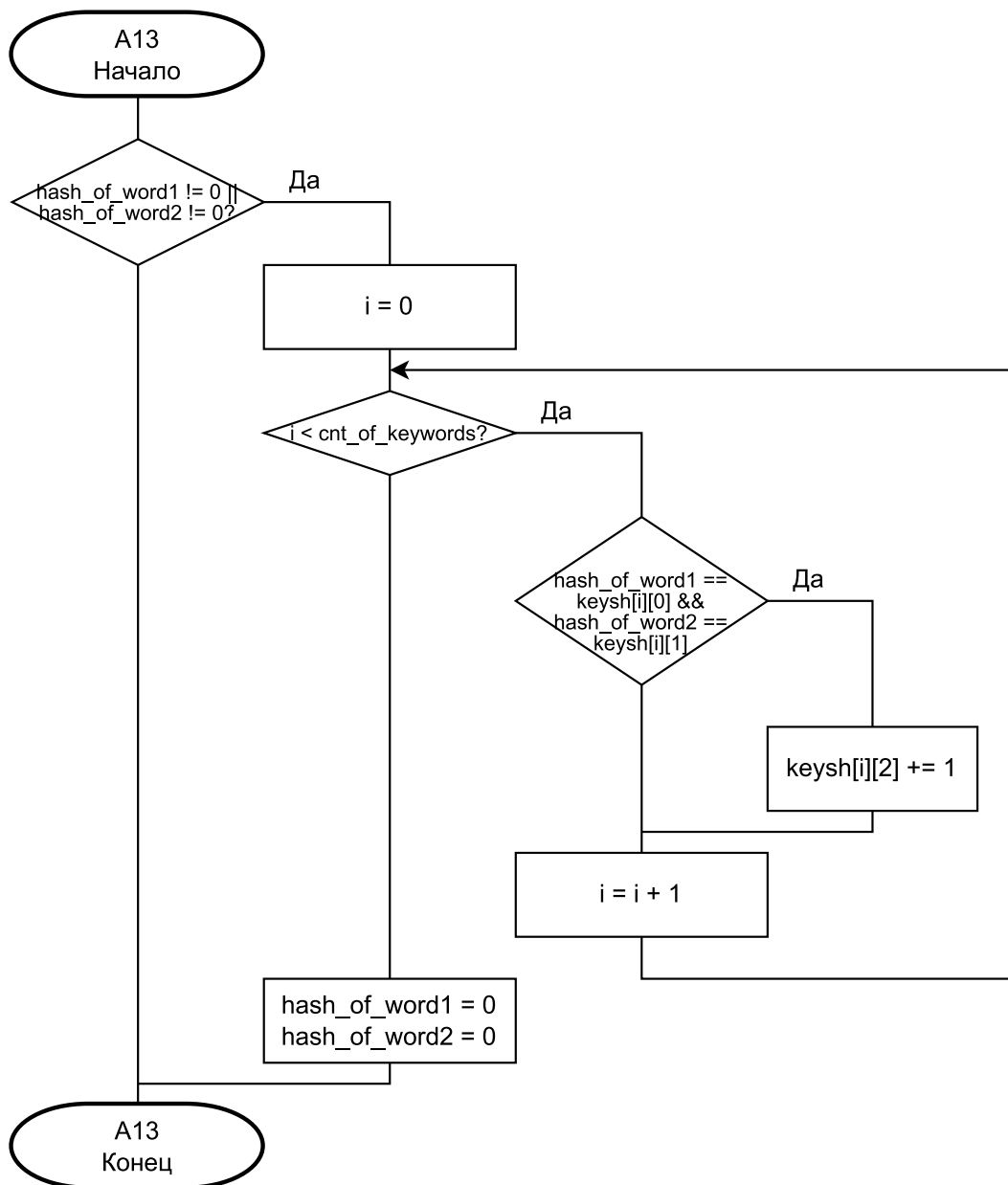
Схема алгоритма

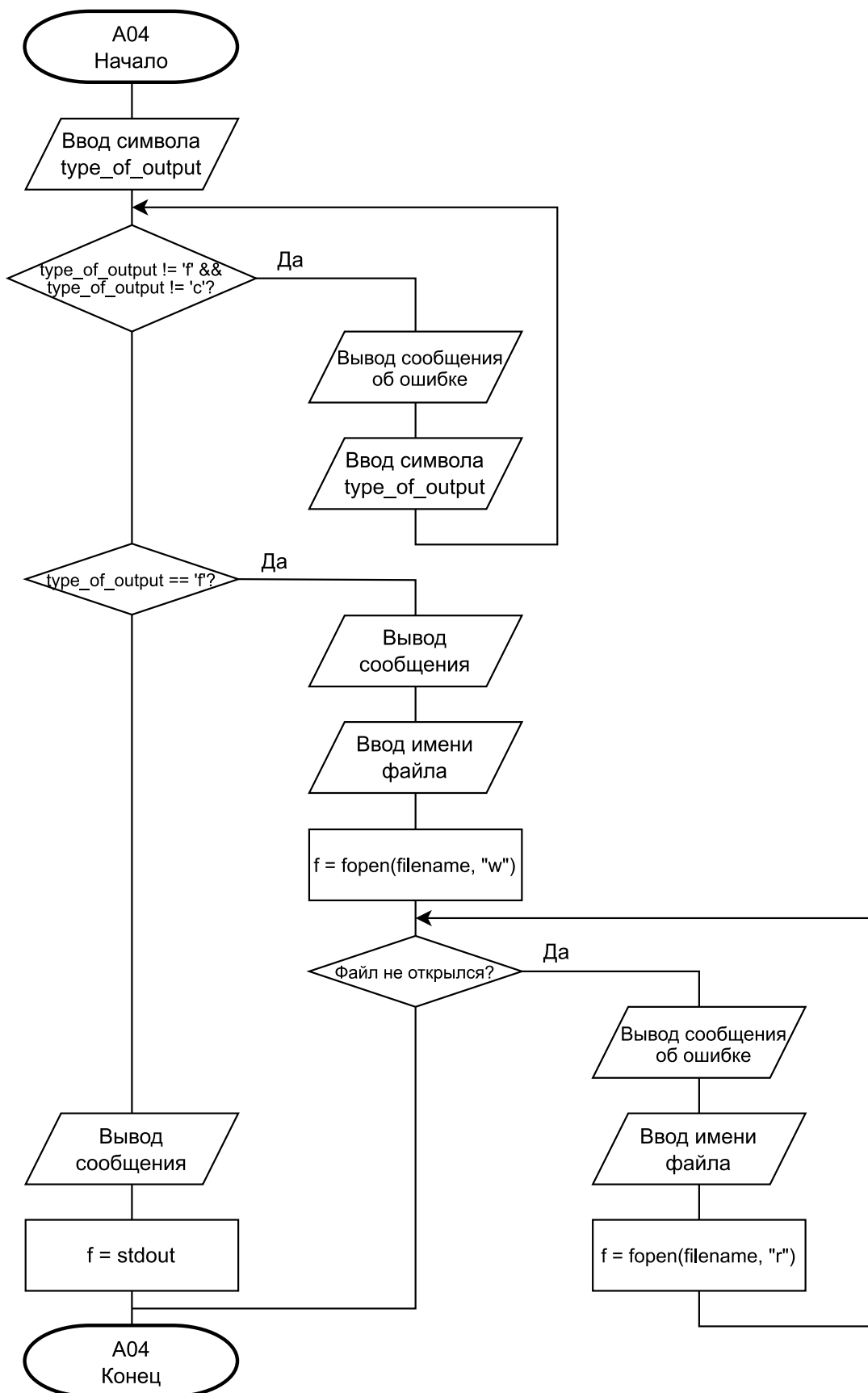


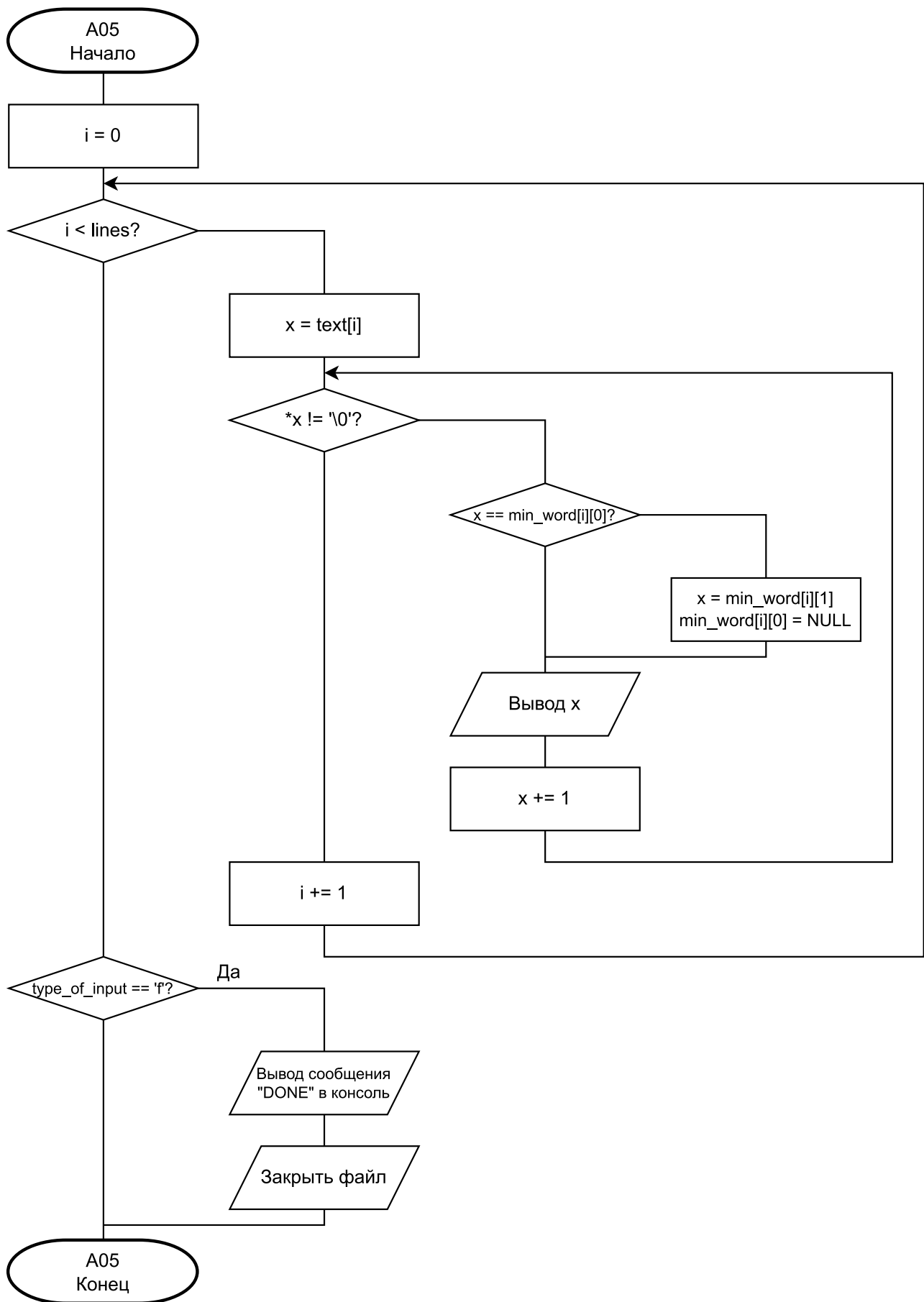












Текст программы

```
#include <stdio.h>

#define HASH_MOD 1000000009 /* Hash module */
#define HASH_B1 43 /* Hash base 1 */
#define HASH_B2 47 /* Hash base 2 */
#define MAXLEN_N 128 /* Max file name length */
#define MAXLEN_K 128 /* Max keyword length */
#define MAXKEYS 256 /* Max keywords count */
#define MAXLEN_D 64 /* Max delimiters length */
#define MAXLEN_S 512 /* Max text lines length */
#define MAXLINES 512 /* Max lines count */

/* The function checks if there is "c" in the array delimiters */
int is_delimiter(char c, char *delimiters);

int main() {
    /* Block of variables */
    int r; /* Module of hash */
    int t; /* First hash base */
    int b; /* Second hash base */
    char type_of_input; /* Character variable for input type ('f' or 'c') */
    char filename[MAXLEN_N]; /* String of filename */
    FILE *f; /* File pointer */
    int cnt_of_keywords; /* Count of keywords */
    char keyword[MAXLEN_K]; /* String of keyword */
    char *x; /* Character pointer */
    long long hash_of_word1; /* Long integer for first hash value */
    long long hash_of_word2; /* Long integer for second hash value */
    int keysh[MAXKEYS][3]; /* 2D array for storing hash values and counts */
    char delimiters[MAXLEN_D]; /* Array to store delimiters */
    int flag; /* Flag for control input */
    int lines; /* Count of lines */
    char text[MAXLINES][MAXLEN_S]; /* 2D array to store text */
    char *start; /* Character pointer for the start of a word */
    int min_line; /* Minimum word length in line */
    int cnt_words_bef; /* Count of founded words before */
    int cnt_words_now; /* Count of founded words now */
    char *min_word[MAXLINES][2]; /* 2D array to store pointers to the beginning and end of the minimum word in each line */
    char type_of_output; /* Character variable for output type ('f' or 'c') */

    /* Block of initialization */
    r = HASH_MOD;
    t = HASH_B1;
    b = HASH_B2;
    flag = 1;
    lines = 0;
    cnt_words_bef = 0;

    /* Info */
    printf("Hello! The program removes the minimum words in the lines that contain keywords.\n");

    /* Block of input */
    /* Prompt the user for input type (file or console) */
    printf("Input from file or console? (f/c)\n");
    type_of_input = getchar();

    /* Validate the input type; loop until a valid input is provided */
    while (type_of_input != 'f' && type_of_input != 'c') {
        printf("Something went wrong! Please enter 'f' or 'c':\n");
        type_of_input = getchar();
    }

    /* Process based on the input type */
    if (type_of_input == 'f') {
        /* Prompt user for the file name and attempt to open the file */
        printf("Please enter the file name (limit: %i chars):\n",
            MAXLEN_N);
        scanf("%s", filename);
        f = fopen(filename, "r");

        /* Continue prompting until a valid file is opened */
    }
}
```

```

while (f == NULL) {
    printf("Something went wrong! Perhaps such a file does not exist."
           "\nPlease enter the file name again:\n");
    scanf("%s", filename);
    f = fopen(filename, "r");
}
} else {
    /* If console input, set the file pointer to standard input */
    f = stdin;
}

/* For console input, prompt for the count of keywords */
if (type of input == 'c')
    printf("Please enter the count of keywords (limit: %i):\n",
           MAXKEYS);
fscanf(f, "%i", &cnt_of_keywords);

/* For console input, prompt for each keyword and compute hash values */
if (type of input == 'c')
    printf("Please enter the keywords (limit: %i chars):\n", MAXLEN_K);
for (int i = 0; i < cnt_of_keywords; ++i) {
    fscanf(f, "%s", keyword);
    x = keyword;
    hash_of_word1 = 0;
    hash_of_word2 = 0;

    /* Compute double hash values for each character in the keyword */
    while (*x != '\0') {
        hash_of_word1 = (hash_of_word1 * t + (int) *x) % r;
        hash_of_word2 = (hash_of_word2 * b + (int) *x) % r;
        ++x;
    }

    /* Store the double hash values and initialize count in the keysh array */
    keysh[i][0] = hash_of_word1;
    keysh[i][1] = hash_of_word2;
    keysh[i][2] = 0;
}

/* For console input, prompt for the line of delimiter characters */
if (type of input == 'c')
    printf("Please enter the line of delimiter characters (limit: %i
chars):\n",
           MAXLEN_D);
fgets(delimiters, MAXLEN_D, f);
fgets(delimiters, MAXLEN_D, f);

/* For console input, prompt for the lines of text with specified limits */
if (type of input == 'c')
    printf("Please enter the lines of text (limit of line's length: %i;
limit\n"
           "count of lines: %i):\n", MAXLEN_S, MAXLINES);

/* Block of main logic */
/* Loop through lines of text from the input file or console */
while (flag == 1 && fgets(text[lines], MAXLEN_S, f)) {
    x = text[lines];
    start = text[lines];
    hash_of_word1 = 0;
    hash_of_word2 = 0;
    min_line = MAXLEN_S;

    /* Iterate through characters in the current line */
    while (*x != '\0') {
        /* Check if the character is a delimiter */
        if (is_delimiter(*x, delimiters) == 1) {
            /* Process the word if hash values are not zero */
            if (hash_of_word1 != 0 || hash_of_word2 != 0) {
                /* Check if the word matches any keyword and update counts */
                for (int i = 0; i < cnt_of_keywords; ++i) {
                    if (hash_of_word1 == keysh[i][0] &&
                        hash_of_word2 == keysh[i][1]) {
                        ++keysh[i][2];
                    }
                }
                /* Reset hash values for the next word */
                hash_of_word1 = 0;
                hash_of_word2 = 0;
            }
        }
        ++x;
    }
}

```

```

        /* Update minimum word information if conditions are met */
        if (x - start != 0 && min_line > x - start) {
            min_line = x - start;
            min_word[lines][0] = start;
            min_word[lines][1] = x;
        }
        start = x + 1; /* Move start pointer to the next character */
    } else {
        /* Update hash values for the current word */
        hash_of_word1 = (hash_of_word1 * t + (int) *x) % r;
        hash_of_word2 = (hash_of_word2 * b + (int) *x) % r;
    }
    ++x;
}

/* Update flag and word count information */
flag = 0;
cnt_words_now = 0;

/* Check if any keyword count is still zero, set the flag accordingly */
for (int i = 0; i < cnt_of_keywords; ++i) {
    if (keysh[i][2] == 0) flag = 1;
    cnt_words_now += keysh[i][2];
}

/* Update minimum word information if the current word count matches the
previous one */
if (cnt_words_now == cnt_words_bef) {
    min_word[lines][0] = text[lines];
    min_word[lines][1] = text[lines];
}

cnt_words_bef = cnt_words_now; /* Update the previous word count */
lines++; /* Move to the next line in the text */
}

/* Close the file if input is from a file */
if (type_of_input == 'f') fclose(f);

/* Block of output */
/* Prompt the user for output type (file or console) */
printf("Output to file or console? (f/c)\n");

/* If the input was from a file, consume an extra newline character from the
buffer */
if (type_of_input == 'f') getchar();

/* Get the user's choice for output type */
type_of_output = getchar();

/* Validate the output type; loop until a valid input is provided */
while (type_of_output != 'f' && type_of_output != 'c') {
    printf("Something went wrong! Please enter '\f' or '\c':\n");
    type_of_output = getchar();
}

/* Process based on the output type */
if (type_of_output == 'f') {
    /* Prompt user for the file name and attempt to open the file for writing */
    printf("Please enter the file name (limit: %i chars):\n",
           MAXLEN N);
    scanf("%s", filename);
    f = fopen(filename, "w");

    /* Continue prompting until a valid file is opened */
    while (f == NULL) {
        printf("Something went wrong! Please enter the file name again:\n");
        scanf("%s", filename);
        f = fopen(filename, "w");
    }
} else {
    /* If console output, set the file pointer to standard output */
    printf("Processed text:\n");
    f = stdout;
}

```

```

/* Process and output each line of text, excluding the minimum words */
for (int i = 0; i < lines; ++i) {
    x = text[i];
    while (*x != '\0') {
        /* Skip characters within the minimum words and update the pointer */
        if (x == min_word[i][0]) {
            x = min_word[i][1];
            min_word[i][0] = NULL;
        }
        /* Output the character to the file or console */
        fprintf(f, "%c", *x);
        ++x;
    }
}

/* If the output type is a file, display a completion message, close the file,
and finalize */
if (type_of_output == 'f') {
    printf("DONE");
    fclose(f);
}

return 0;
}

int is_delimiter(char c, char *delimiters) {
    char *x;
    int ans;
    x = delimiters;
    ans = 0;
    while (*x != '\0') { /* Iterating until the end of delimiters */
        if (c == *x) ans = 1;
        ++x;
    }
    return ans; /* Return 1 if the element in the array 0 else */
}

```

Контрольные примеры

№	Исходные данные	Результаты
1	f input1.txt f output1.txt	DONE
2	f input2.txt c	Processed text: Ryaba Hen Once upon time there lived an old man and an old woman. And they had hen, called Ryaba. One day the hen laid an egg not a simple one, but golden one. The old man hit it and hit it, but couldn' break it. The old woman hit it and hit it, but couldn't break it. A mouse was running by, swang its tail, the egg fell and broke. The old man crying, the old woman is crying, but the hen is clucking, "Don' cry, grandpa, don't cry, grandma. I'll lay you another egg, not a golden one, but a simple one!"

3	c 4 are simpler, programmer Java, There are many other programming languages out there with a host of cool features that make developing applications relatively easy. You might have heard of some of these languages like Python, Java, and JavaScript. However, if you are new to programming and want a career in it, you need to learn C++ first. This is because other modern languages, while convenient and simpler, rob you of the ability to learn important concepts that any successful programmer should know. f output3.txt	DONE
---	---	------

Содержимое файлов

input1.txt ×

1 4

2 of

3 and

4 in

5 caught

6 (, .)

7 The New York Times

8 Styles's 71 Most Stylish 'People' of 2023

9 OK, some weren't people, but they all made us talk:

10 about what we wear, how we live and how we express ourselves.

11 The "people" on this list – who are presented in no particular

12 order – reflect the ways that the Styles desk defines its coverage:

13 high and low; fun and serious; curious and open-minded; reveling in

14 characters; appreciating the material world; inviting everyone to the party.

15 Many were recognized for being Styles-ish, others for being stylish.

16 (Several could not have done it without the help of stylists, costume

17 designers and other crews.) Lots came from the worlds of politics, film,

18 TV, music, sports and fashion. But a few caught our attention in less expected

19 places, like courtrooms. Some had great hair. Some had singular accessories.

20 One person had both – and was mistaken for a duchess in disguise. Certain

21 people might surprise you or (we hope) inspire heated debate. After all,

22 one thing they had in common is they made us talk: about what we wear,

23 how we live and how we express ourselves.

24

✓ 1 ^ v

CRLF windows-1251 4 spaces

```
output1.txt X
1 The New York Times
2 Styles's Most Stylish 'People' of 2023
3 OK, some weren't people, but they all made us talk:
4 about what wear, how we live and how we express ourselves.
5 The "people" on this list who are presented in no particular
6 order - reflect the ways that the Styles desk defines its coverage:
7 high and low; fun and serious; curious and open-minded; reveling
8 characters; appreciating the material world; inviting everyone to the party.
9 Many were recognized for being Styles-ish, others for being stylish.
10 (Several could not have done without the help of stylists, costume
11 designers and other crews.) Lots came from the worlds politics, film,
12 TV, music, sports and fashion. But few caught our attention in less expected
13
```

CRLF windows-1251 4 spaces

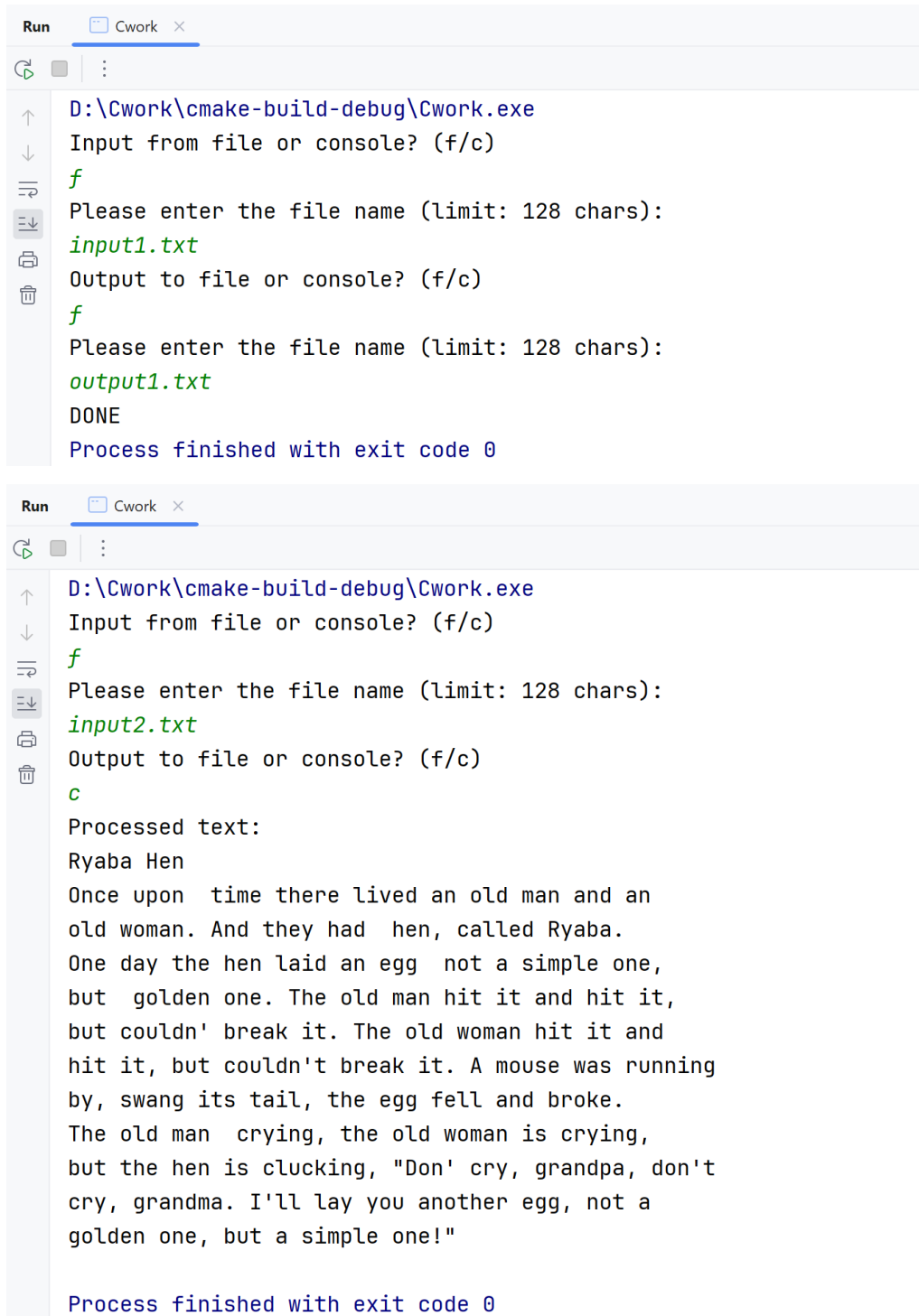
```
input2.txt X
1 3
2 none
3 old
4 hen
5 " '
6 Ryaba Hen
7 Once upon a time there lived an old man and an
8 old woman. And they had a hen, called Ryaba.
9 One day the hen laid an egg - not a simple one,
10 but a golden one. The old man hit it and hit it,
11 but couldn't break it. The old woman hit it and
12 hit it, but couldn't break it. A mouse was running
13 by, swang its tail, the egg fell and broke.
14 The old man is crying, the old woman is crying,
15 but the hen is clucking, "Don't cry, grandpa, don't
16 cry, grandma. I'll lay you another egg, not a
17 golden one, but a simple one!"
18
```

CRLF windows-1251 4 spaces

```
output3.txt X
1 There many other programming languages out there with
2 a host of cool features that make developing applications
3 relatively easy. You might have heard of some of these
4 languages like Python, Java, JavaScript.
5 However, if you are new to programming and want career
6 in it, you need to learn C++ first. This is because other
7 modern languages, while convenient and simpler, rob you
8 the ability to learn important concepts that any successful
9 programmer should
10
```

CRLF windows-1251 4 spaces

Примеры выполнения программы



```
Run Cwork x
D:\Cwork\cmake-build-debug\Cwork.exe
Input from file or console? (f/c)
f
Please enter the file name (limit: 128 chars):
input1.txt
Output to file or console? (f/c)
f
Please enter the file name (limit: 128 chars):
output1.txt
DONE
Process finished with exit code 0
```



```
Run Cwork x
D:\Cwork\cmake-build-debug\Cwork.exe
Input from file or console? (f/c)
f
Please enter the file name (limit: 128 chars):
input2.txt
Output to file or console? (f/c)
c
Processed text:
Ryaba Hen
Once upon time there lived an old man and an
old woman. And they had hen, called Ryaba.
One day the hen laid an egg not a simple one,
but golden one. The old man hit it and hit it,
but couldn' break it. The old woman hit it and
hit it, but couldn't break it. A mouse was running
by, swang its tail, the egg fell and broke.
The old man crying, the old woman is crying,
but the hen is clucking, "Don' cry, grandpa, don't
cry, grandma. I'll lay you another egg, not a
golden one, but a simple one!"
Process finished with exit code 0
```

```
Run Cwork x
D:\Cwork\cmake-build-debug\Cwork.exe
Input from file or console? (f/c)
c
Please enter the count of keywords (limit: 256):
4
Please enter the keywords (limit: 128 chars):
are
simpler,
programmer
Java,
Please enter the line of delimiter characters (limit: 64 chars):

Please enter the lines of text (limit of line's length: 512; limit
count of lines: 512):
There are many other programming languages out there with
a host of cool features that make developing applications
relatively easy. You might have heard of some of these
languages like Python, Java, and JavaScript.
However, if you are new to programming and want a career
in it, you need to learn C++ first. This is because other
modern languages, while convenient and simpler, rob you of
the ability to learn important concepts that any successful
programmer should know.
Output to file or console? (f/c)
f
Please enter the file name (limit: 128 chars):
output3.txt
DONE
Process finished with exit code 0
```

Выводы

В работе использован только один заголовочный файл стандартной библиотеки. `<stdio.h>` используется для ввода и вывода из файла и консоли.