**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по курсовой работе**

**по дисциплине «Программирование»**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0305 |  | Боев И.С. |
| Преподаватель |  | Павлом С.М. |

Санкт-Петербург

2020

**Цель работы**

Закрепление практических навыков в написании программ на языке программирования СИ, используя пройденный за семестр материал, а именно динамическую память, работу с файлами, обработку строк, разработку алгоритмов, работу с двухмерными массивами.

**Задание**

Ввести строку символов разделителей и текст, состоящий из заданного количества строк. В процессе ввода строк текста не записывать в них повторяющиеся после каждого слова символы разделители.

Из строк введенного текста сформировать другой текст, в который войдут только строки, в которые после слов, содержащих больше заданного в исходных данных количества одинаковых символов, будет вставлено новое введенное слово. В случае необходимости добавить любой символ из строки символов разделителей.

Вывести введённый и сформированный тексты.

**Постановка задачи и описание решения**

Алгоритм считывает из ввода (ввод может производиться через файл или консоль) построчно исходный текст, пропуская при этом в каждой строчке повторяющиеся после каждого слова символы-разделители. Параллельно с этим, он фиксирует в вводимых строчках позиции, куда нужно будет вставить слова для формирования строк нового текста, в виде пары чисел, отвечающих за номер строки исходного текста и номер символа.

После ввода, если есть строки, из которых нужно сформировать новый текст, построчно формируется новый текст. Каждая строка из исходного текста копируется в новую строку, считывая из ввода слова для вставки. Если такие слова отсутствуют, то вставляется нулевой символ строки символов-разделителей.

Исходный и, если был создан, сформированный тексты выводятся.

Алгоритм реализован в отдельной функции algorithm, которая будет описана ниже.

Программа при динамическом выделении памяти производит проверку на её выделение. Если он не была выделена, то работа алгоритма останавливается, в консоль выводится ошибка и меню снова вызывается. Выделяемая память очищается после того, как перестаёт использоваться (например, в конце функции algorithm).

В программе реализован интерфейс для удобного взаимодействия между пользователем или программой. При неверном вводе программа сообщит пользователю об этом и попросит ввести значение ещё раз.

При вводе чисел учитывается ситуация, когда могут быть случайно введены нечисловые символы. В этом случае, они не учитываются при вводе и программа продолжает своё выполнение.

Для каждого из представленных в программе массивов, количество выделенной памяти увеличивается с увеличением размера самого массива. Для этого используется функция realloc.

**Описание функции int main()**

В данной программе функция main выполняет роль меню. Пользователю даётся на выбор 3 вариант: показать информацию, относящуюся к алгоритму, выполнить алгоритм или закончить выполнение программы.

При выборе 2ого варианта, программа просит пользователя выбрать тип ввода данных: консольный или файловый. При выборе файлового ввода, программа запрашивает пользователя ввести название файла, из которого будут браться данные, и создаёт файл results.txt, куда будут записаны результаты.

В функцию algorithm передаются переменные типа FILE\* istream и ostream – потоки ввода и вывода соответственно.

**Описание переменных функции int main()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | n | int | Ввод числа выбора в меню |
| 2 | m | int | Ввод числа выбора типа ввода/вывода данных |
| 3 | istream | FILE\* | Количество строчек в формируемом двухмерном массиве |
| 4 | ostream | FILE\* | Количество столбцов в формируемом двухмерном массиве |

**Описание функции void info()**

Функция info выводит информацию о создателе программы, задаче алгоритма, входных и выходных данных.

**Описание переменных функции void info()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| - | - | - | - |

**Описание функции STR get\_str(FILE\* stream, STR message)**

Функция get\_str считывает из потока stream строку символов str и возвращает её. Если ввод производится через консоль, то функция выводит пользователю сообщение message.

В переменную c считывается первый символ, не являющийся символом переноса каретки или символом новой строки (\r или \n) посредством функции get\_next\_symb(stream). В цикле do… while символ c записывается в n-ый символ строки str (str[n]), n инкрементируется, и в переменную c считывается следующий символ из ввода, пока не будут введены ‘\n’ или ‘\r’.

После цикла, в n символ строки записывается нулевой байт – конец строки.

**Описание переменных функции STR get\_str(FILE\* stream, STR message)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | stream | FILE\* | Аргумент. Поток ввода |
| 2 | message | STR | Аргумент. Сообщение, выводящееся в консоль |
| 3 | str | char\* | Вводимая строка |
| 4 | n | int | Номер вводимого элемента |
| 5 | c | char | Символ, вводящийся в строку |

**Описание функции char get\_next\_symb(FILE\* stream)**

Назначение функции get\_next\_symb – получить следующий в потоке stream символ, не являющийся символом переноса каретки или символом новой строки, т.е. \n и \r соответственно, и вернуть его.

**Описание переменных функции char get\_next\_symb(FILE\* stream)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | stream | FILE\* | Аргумент. Поток ввода |
| 2 | c | char | Вводимый символ |

**Описание функции void clear\_f(FILE\* stream)**

Функция clear\_f убирает из потока stream любые символы, не являющиеся символами новой строки или переноса каретки.

В программе, используется только после ввода чисел посредством функции fscanf и scanf, чтобы убрать «лишние символы» из потока.

**Описание переменных функции void clear\_f(FILE\* stream)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | arr | int\*\* | Аргумент. Поток ввода |
| 2 | c | char | Вводимый символ |

**Описание функции void algorithm(FILE\* is, FILE\* os)**

Функция get\_str является функцией, в которой выполняется выполняется задача алгоритма. Данные вводятся из потока is и выводятся в поток os. При любой ошибке выводится сообщение, освобождается память, которая уже была динамически выделена под некоторые переменные и пользователь возвращается в меню.

Вводятся количество строчек исходного текста, количество одинаковых символов в одном слове и строка символов разделителей. Построчно вводится исходный текст pText в цикле for при помощи функции read\_text\_str. При этом, в функцию передаётся указатель на двухмерный массив p. Он представляет из себя массив пар чисел, где нулевой элемент – номер строки, а первый – номер позиции в строке, куда нужно вставить слово при формировании нового текста. Двухмерный массив обязательно заканчивается парой, где нулевой элемент равен -1.

После через функцию form\_text формируется новый текста fText. Если он не был сформирован, то ему присваивается значение NULL, и программа выведет сообщение «Текст не сформирован». Если произошла ошибка при формировании текста, то strfN принимает значение -1.

В конце алгоритма, если ввод/вывод являлся файловым, потоки is и os закрываются, динамически выделенная память освобождается.

**Описание переменных функции void algorithm(FILE\* is, FILE\* os)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | is | FILE\* | Аргумент. Поток ввода |
| 2 | os | FILE\* | Аргумент. Поток вывода |
| 3 | strN | int | Количество строк в исходном тексте |
| 4 | strfN | int | Количество строк в сформированном тексте |
| 5 | repN | int | Количество одинаковых символов в одном слове |
| 6 | del | STR | Строка символов-разделителей |
| 7 | pText | TEXT | Исходный текст |
| 8 | fText | TEXT | Сформированный текст |
| 9 | p | int\*\* | Двухмерный массив позиций для вставки слов |

**Описание функции STR read\_text\_str(FILE\* stream, int\*\*\* p, STR del, CI repN, CI i, CI strN)**

Функция read\_text\_str считывает i-тую строку исходного текста из потока stream. Ввод заканчивается, если программа достигла конца строки (‘\n для консольного ввода и ‘\r’ для файлового ввода) или конца файла EOF. Если EOF был встречен до того, как закончился ввод всего текста (if c == EOF && i != strN – 1), то функция возвращает NULL.

Перед циклом создаётся счётчик counter (counter = counter\_alloc), являющийся массивом пар чисел, первое из которых – код введённого символа, а второе – количество повторений этого символа. Счётчик всегда заканчивается парой чисел, где первое из них равно нулю.

В цикле while посимвольно считывается строка из потока ввода, пока введённый символ не равен EOF или символам переноса каретки или новой строки.

В первой итерации цикла если ввод является файловым, то символ считывается посредством функции get\_next\_symb, чтобы исключить случайный ввод символов ‘\n’ или ‘\r’.

Если введённый символ является символом-разделителем, что определяется через вызов функции str\_repeat\_check(del, c), тогда проверяется счётчик символов (counter\_repeat\_check(counter, repN)). Если функция возвращает true, то в двухмерный массив \*p добавляется позиция рассматриваемого символа. После, счётчик обнуляется (counter = free\_counter(), создаётся новый. Если символ разделитель не был ранее введён (проверка в str\_repeat\_check(dCheck, c), то он добавляется в строку текста str и строку введённых символов-разделителей dCheck (dCheck = add\_symb(&dCheck, c).

Если введённый символ является EOF, ‘\r’ или ‘\n\’, то происходит то же самое, но вместо создания нового счётчик и проверки символа-разделителя, в строку str записывается символ конца строки ‘\0’.

Во всех иных случаях, введённый символ – часть слова, поэтому количество его повторений в счётчике counter увеличивается на 1 или становится 1 (counter = count\_symb(&counterm c)). В str[n] записывается символ, n инкрементируется.

По завершению цикла освобождается память, выделенная под строку введённых символов-разделителей и возвращается str.

**Описание переменных функции void algorithm(FILE\* is, FILE\* os)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | stream | FILE\* | Аргумент. Поток ввода |
| 2 | p | int\*\*\* | Аргумент. Указатель на двухмерный массив позиций исходного текста для вставки слов |
| 3 | del | STR | Аргумент. Строка символов-разделителей |
| 4 | repN | CI | Аргумент. Количество повторений символов в слове |
| 5 | i | CI | Аргумент. Номер записываемой строки исходного текста |
| 6 | strN | CI | Аргумент. Количество строк исходного текста |
| 7 | n | int | Количество введённых символов |
| 8 | str | char\* | Вводимая строка |
| 9 | counter | int\*\* | Счётчик повторений символов в слове |
| 10 | c | char | Вводимый символ |
| 11 | dCheck | char\* | Строка символов-разделителей, введённых ранее в строке |

**Описание функции int\*\* add\_int\_p(int\*\*\* parr, int x, int y)**

Функция add\_int\_p добавляет в двухмерный массив позиций для вставки слов в исходный текст новую позицию, где x – номер строки, а y – номер позиции в строке.

Цикл while проходит через все элементы двухмерного массива, увеличивая каждый раз i на 1. Когда был достигнут его конец ((\*parr)[i][0] == -1), то на это место вставляется пара чисел x и y, а нулевой элемент следующей пары (номер i + 1) приравнивается к -1, чтобы обозначить конец двухмерного массива.

Функция возвращает двухмерный массив \*parr.

**Описание переменных функции int\*\* add\_int\_p(int\*\*\* parr, int x, int y)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | parr | int\*\*\* | Аргумент. Указатель на двухмерный массив позиций в строках для вставки слов |
| 2 | x | int | Аргумент. Номер строки |
| 3 | y | int | Аргумент. Номер позиции в строке |
| 4 | i | int | Номер строки двухмерного массива, куда нужно вставить пару чисел x и y |

**Описание функции int\*\* counter\_alloc()**

Функция counter\_alloc возвращает указатель на память, выделенную под пустой двухмерный массив-счётчик, каждая строка которого состоит из пары чисел – кода символа и количества его повторений. Нулевой элемент нулевой строки приравнивается к нулю, чтобы обозначить конец счётчика.

**Описание переменных функции int\*\* counter\_alloc()**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | arr | int\*\* | Двухмерный массив-счётчик |

**Описание функции int\*\* free\_counter(int\*\*\* arr)**

Функция free\_counter освобождает память, выделенную под двухмерный массив-счётчик и возвращает NULL.

**Описание переменных функции int\*\* free\_counter(int\*\*\* arr)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | parr | int\*\*\* | Аргумент. Указатель на двухмерный массив-счётчик |
| 2 | i | int | Номер строки |

**Описание функции char\* add\_symb(char\*\* s, const char c)**

Функция add\_symb добавляет в конец строки \*s символ c на i-тую позицию, на позицию i+1 – символ окончания строки ‘\0’ и возвращает строку \*str.

**Описание переменных функции char\* add\_symb(char\*\* s, const char c)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | str | char\*\* | Аргумент. Указатель на строку |
| 2 | c | char | Аргумент. Символ для вставки в конец строки |
| 3 | i | int | Номер позиции для вставки символа |

**Описание функции int\*\* count\_symbol(int\*\*\* m, char c)**

Функция count\_symb добавляет единицу к количеству повторений символа c в двухмерном массиве-счётчике \*m. Если символ не был раньше введён, то он добавляется в \*m, количество его повторений становится равным 1 и обозначается конец массива((\*m)[i+1][0] = 0).

Функция возвращает двухмерный массив \*m.

**Описание переменных функции int\*\* count\_symbol(int\*\*\* m, char c)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | m | int\*\*\* | Аргумент. Указатель на двухмерный массив-счётчик |
| 2 | c | char | Аргумент. Повторяющийся символ |
| 3 | i | int | Номер строки двухмерного массива |

**Описание функции bool counter\_repeat\_check(int\*\* counter, CI repN)**

Функция counter\_repeat\_check проверяет, есть ли в двухмерном массиве-счётчике counter символ, повторяющийся в тексте более repN раз

Функция возвращает true, если такой символ был найден, и false, если не был найден (если counter[i][0] == 0)

**Описание переменных функции bool counter\_repeat\_check(int\*\* counter, CI repN)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | counter | int\*\* | Аргумент. Двухмерный массив-счётчик |
| 2 | repN | int | Аргумент. Количество повторений одного символа в слове |
| 3 | i | int | Номер строки двухмерного массива |

**Описание функции bool str\_repeat\_check(STR s, const char c)**

Функция str\_repeat\_check проверяет, существует ли символ c в строке s.

Функция возвращает true, если такой символ был найден, и false, если не был найден (если s[i] == ‘\0’).

**Описание переменных функции bool str\_repeat\_check(STR s, const char c)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | s | STR | Аргумент. Строка символов |
| 2 | c | const char | Аргумент. Символ |
| 3 | i | int | Номер символа в строке |

**Описание функции TEXT form\_text(FILE\* stream, TEXT t, int\*\* p, int\* k, STR del)**

Функция form\_text формирует новый текст из строк исходного текста t, в которые будут вставлены введённые из потока stream слова.

Цикл while продолжается, пока не будет достигнут конец двухмерного массива позиций p (p[m][0] == -1).

Если рассматривается строка, которая будет вставлена в формирующийся текст(p[m][0] совпадает с номер i рассматриваемой строки исходного текста t), то создаётся строка str, в которую во вложенном цикле while посимвольно будет записываться строка t[i]. Если позиция этой строки (i и j) совпадает с позицией из двухмерного массива позиций p (p[m][0] и p[m][1]), тогда на место этой позиции записывается слово через функцию insert\_word или, если слово не было введено, добавляется первый символ строки символов-разделителей (по условию задачи, нужно вставить любой символ-разделитель). Переменная увеличивается на размер вставленной строки, m икрементируется.

Каждый символ t[i][j] записывается в символ str[j + ind]. Вложенный цикл заканчивается икрементированием j.

После окончания записи, символу str[j + ind] присваивается значение символа конца строки ‘\0’ и \*k-той строке формирующегося текста fT присваивается строка str. Значение \*k увеличивается на 1.

В конце всего цикла while i инкрементируется для перехода к рассмотрению следующей строки исходного текста.

Функция возвращает сформированный текст fT или NULL, если он не был сформирован. В случае, когда происходит ошибка выделения памяти, функция возвращает NULL и присваивает \*k значение -1, т.к. иначе программа может считать, что функция не сформировала новый текст.

**Описание переменных функции TEXT form\_text(FILE\* stream, TEXT t, int\*\* p, int\* k, STR del)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | stream | FILE\* | Аргумент. Поток ввода |
| 2 | t | TEXT | Аргумент. Исходный текст |
| 3 | p | int\*\* | Аргумент. Двухмерный массив позиций для вставки слов |
| 4 | k | int\* | Аргумент. Указатель на количество строк в формирующемся тексте |
| 5 | del | STR | Аргумент. Строка символов-разделителей |
| 6 | fT | TEXT | Формирующийся текст |
| 7 | ind | int | Количество вставленных символов в строку |
| 8 | str | char\* | Строка исходного текста со вставленными словами |
| 9 | m | int | Номер следующей пары чисел массива p |
| 10 | i | int | Номер строки исходного текста |
| 11 | j | int | Номер символа строки исходного текста |
| 12 | b | int | Количество вставленных символов в функции insert\_word |

**Описание функции int insert\_word(FILE\* stream, char\*\* s, int index, bool last)**

Функция insert\_word посимвольно считывает из потока stream слово и посимвольно вставляет его в строку \*s, начиная с позиции index.

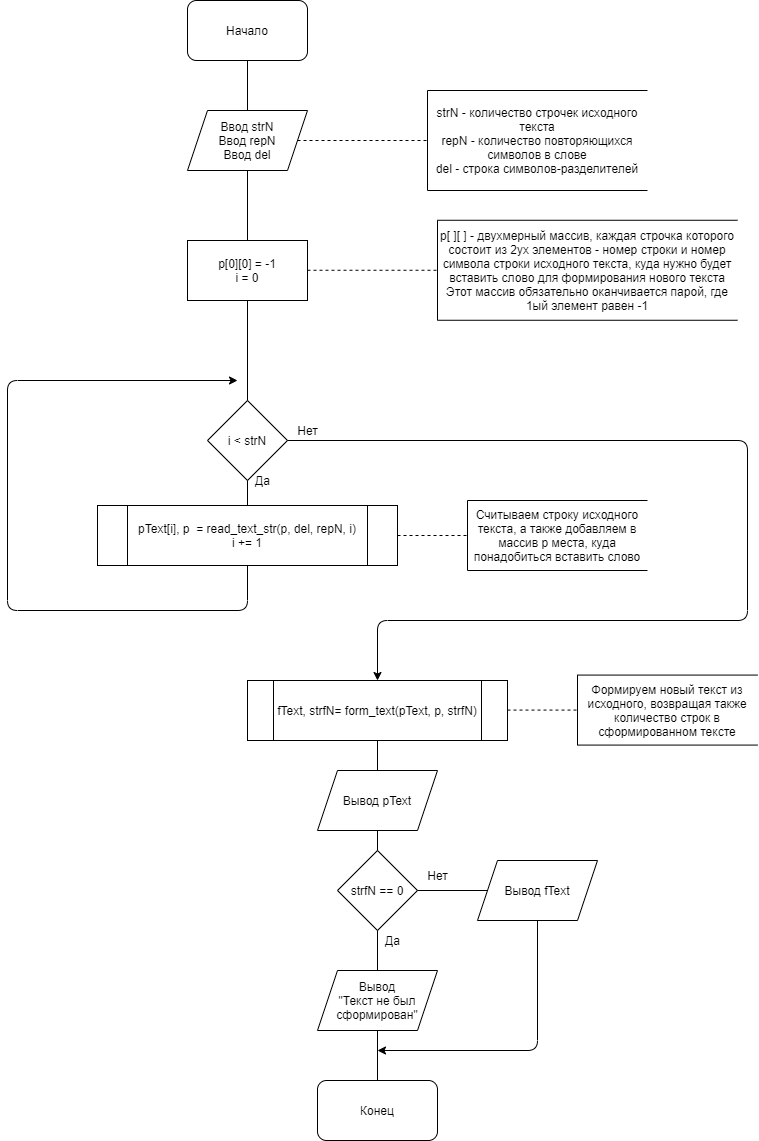
Если файловый, а не консольный, то вместо fgetc для записи первого символа используется используется get\_next\_symb, чтобы пропустить символы ‘\n’ и ‘\r’.

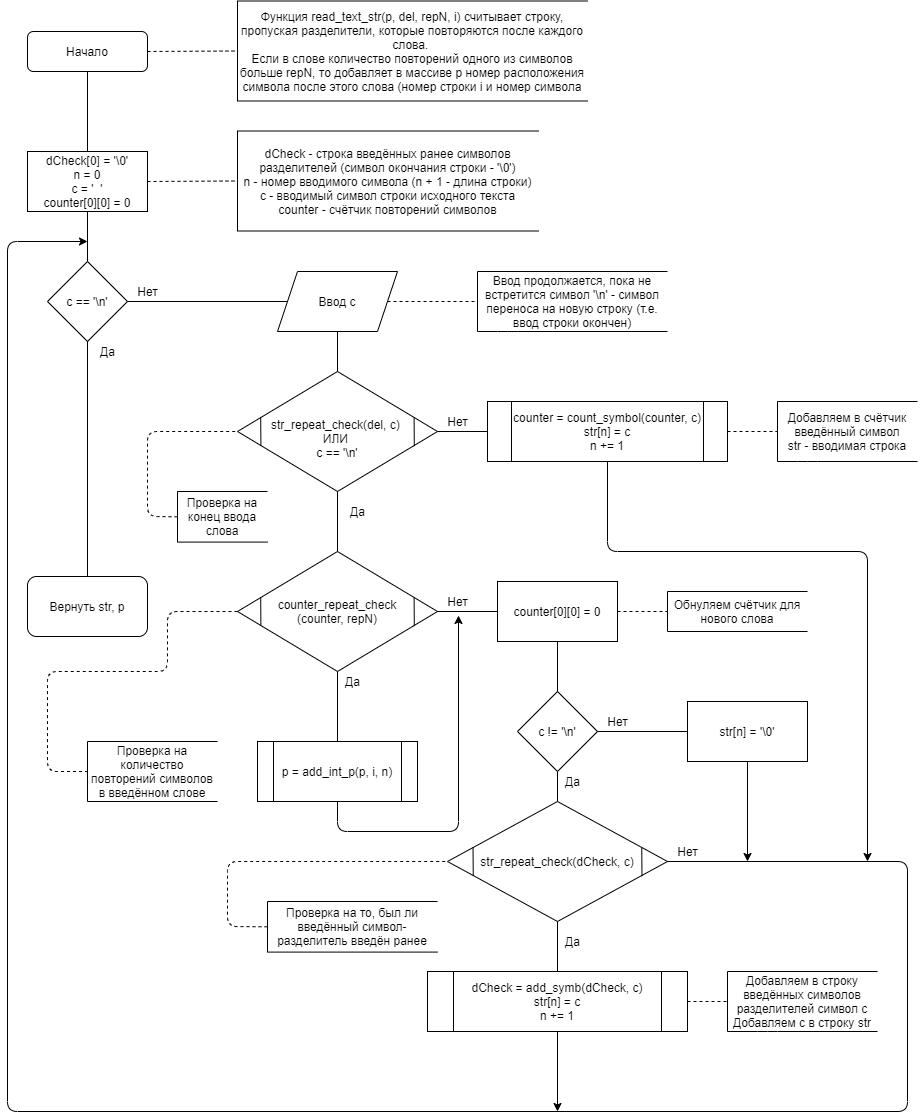
Посимвольная запись происходит в цикле while, пока символ не станет равным ‘\n’, ‘\r’ или EOF. В каждой итерации в переменную с записывается новый символ, а n увеличивается на 1.

Функция возвращает n – количество введённых символов.

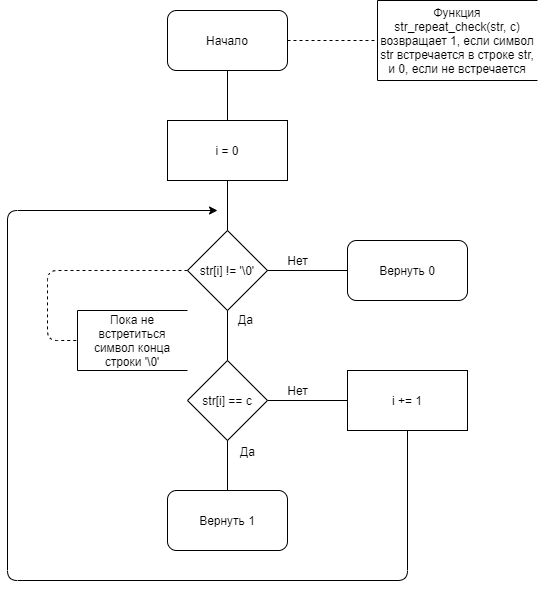
**Описание переменных функции TEXT form\_text(FILE\* stream, TEXT t, int\*\* p, int\* k, STR del)**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | stream | FILE\* | Аргумент. Поток ввода |
| 2 | s | char\*\* | Аргумент. Указатель на строку |
| 3 | index | int | Аргумент. Позиция, с которой начинается запись слова в s |
| 4 | n | int | Количество введённых символов |
| 5 | c | char | Вводимый символ |

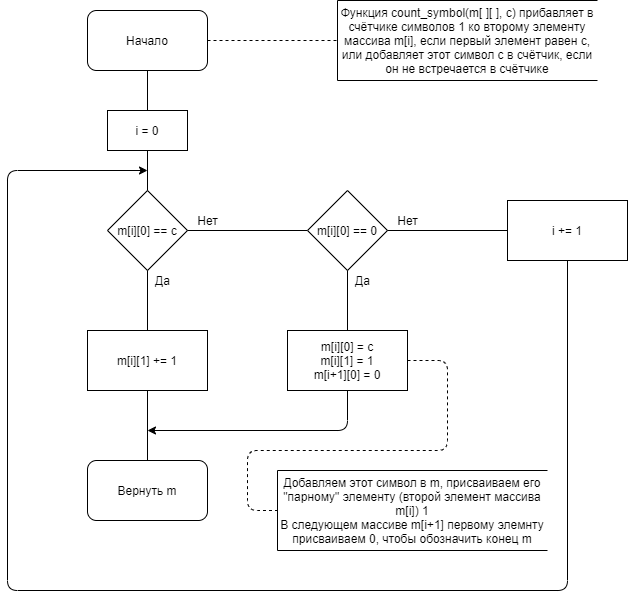
**Схема алгоритма (функция algorithm)**

**Схема алгоритма функции read\_text\_str**

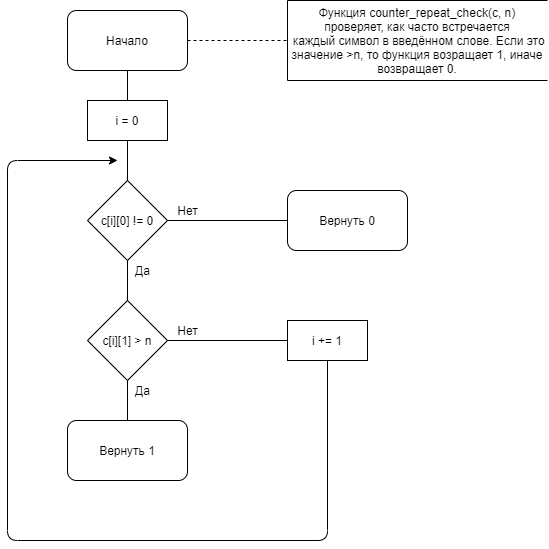
**Схема алгоритма функции str\_repeat\_check**

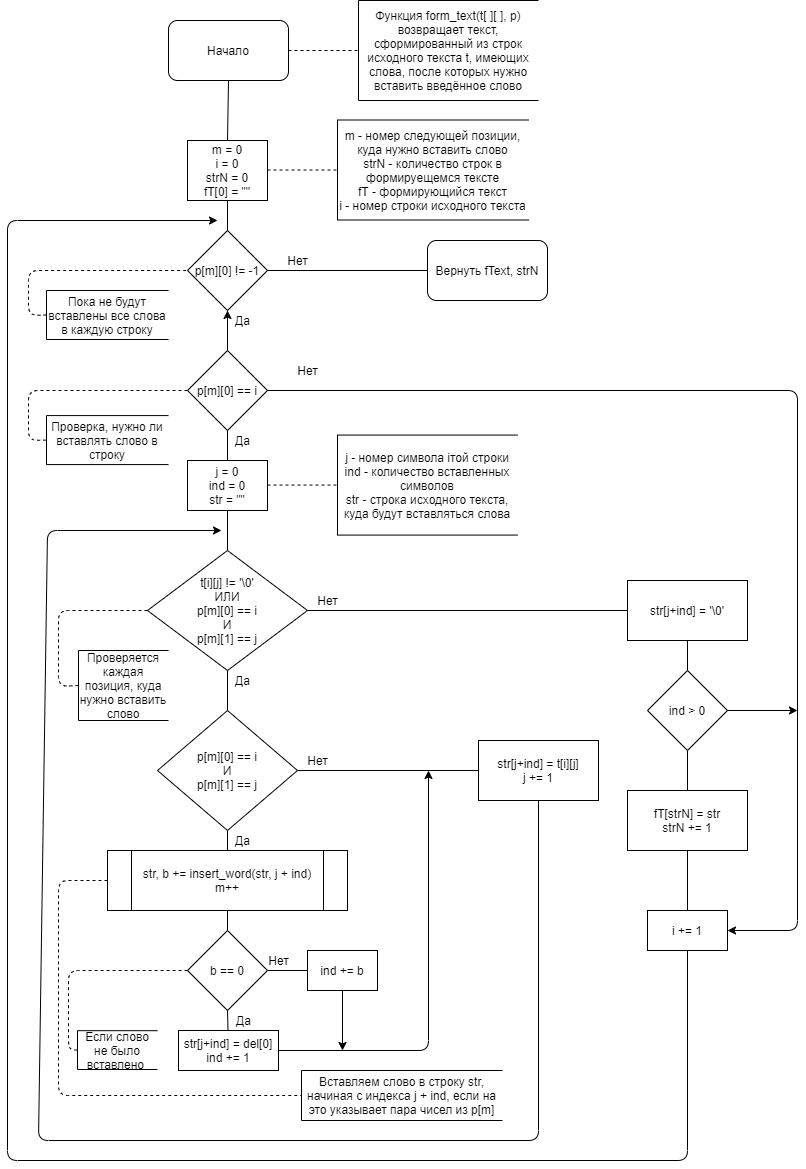


**Схема алгоритма функции count\_symbol**

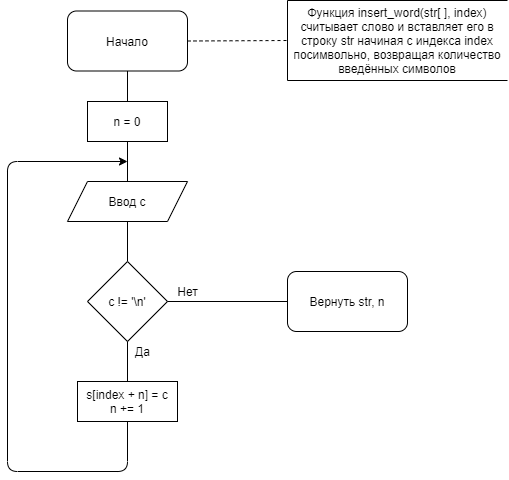
****

**Схема алгоритма функции counter\_repeat\_check**

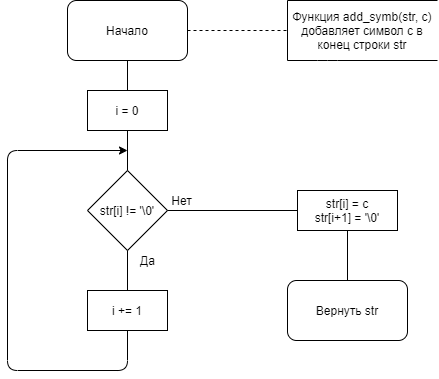
****

**Схема алгоритма функции fText**

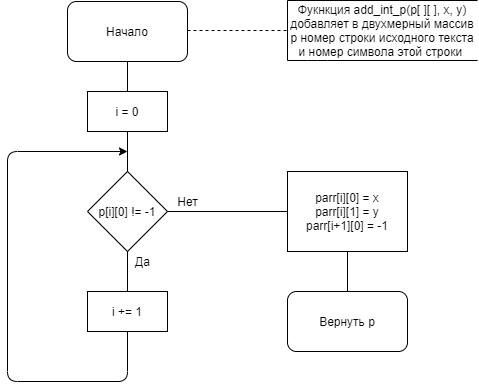
**Схема алгоритма функции insert\_word**



**Схема алгоритма функции add\_symb**

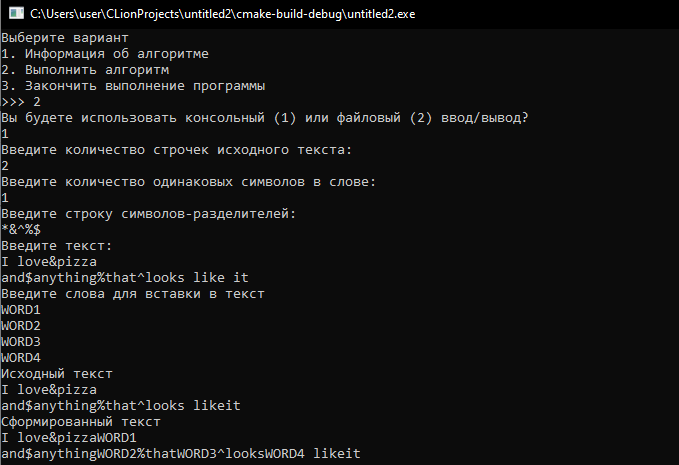


**Схема алгоритма функции add\_int\_p**

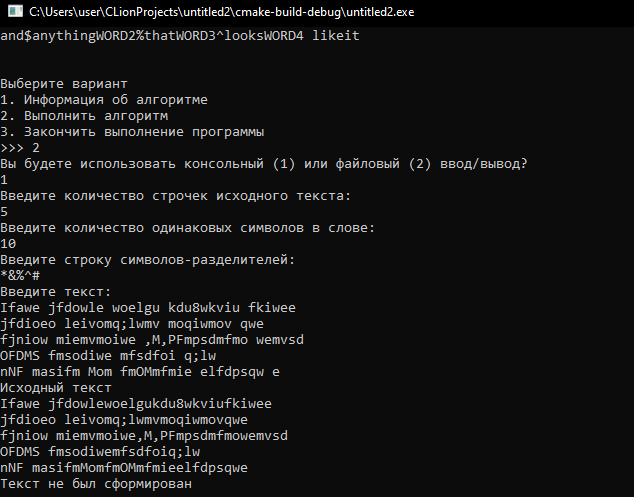


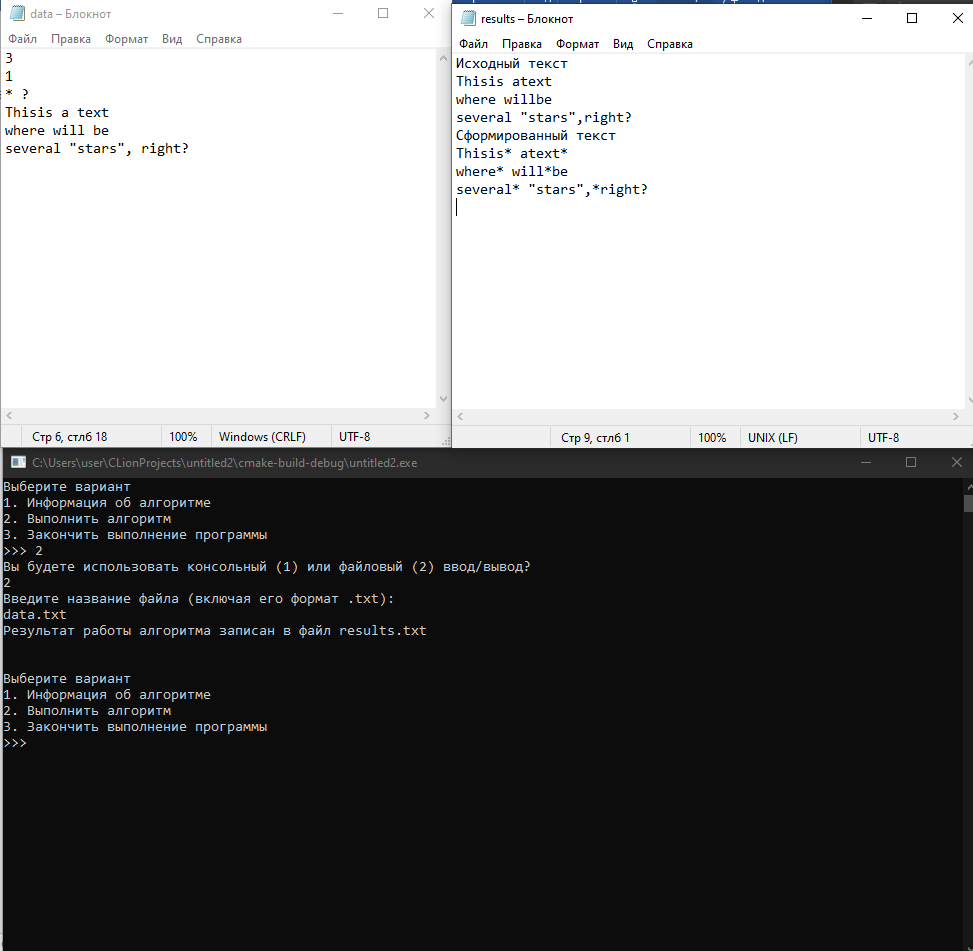
**Контрольные примеры**

**Пример 1:**

****

**Пример 2:**

****

**Пример 3:**

**Текст программы**

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include <locale.h>

typedef const char\*\* TEXT;

typedef const char\* STR;

typedef const int CI;

//--------------------------------------------------------------------------------------------------------------------------------------------

//---------ГЛАВНАЯ ЧАСТЬ----------------------------------------------------------------------------------------------------------------------

//--------------------------------------------------------------------------------------------------------------------------------------------

//Объявление функций

void info();

void algorithm(FILE\* is, FILE\* os);

STR get\_str(FILE\* stream, STR message);

char get\_next\_symb(FILE\* stream){

char c;

do{

c = fgetc(stream);

} while(c == '\n' || c == '\r');

return c;

}

void clear\_f(FILE\* stream){

char c;

do{

c = fgetc(stream);

} while(c != '\n' && c != "\r");

}

//ГЛАВНАЯ ФУНКЦИЯ-----------------------------------------------------------------------------------------------------------------------------

//--------------------------------------------------------------------------------------------------------------------------------------------

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Rus");

int n = 0;

//Выводит меню, пока пользователь не введёт символ для закрытия программы

while(1){

printf("Выберите вариант\n"

"1. Информация об алгоритме\n"

"2. Выполнить алгоритм\n"

"3. Закончить выполнение программы\n");

printf(">>> "); scanf("%d", &n);

clear\_f(stdin);

switch(n){

case 1:{

info(); break;

}

case 2:

{

FILE\* istream = NULL;

FILE\* ostream = NULL;

int m;

while(istream == NULL && istream == NULL) {

puts("Вы будете использовать консольный (1) или файловый (2) ввод/вывод?");

scanf("%d", &m);

clear\_f(stdin);

if(m == 1) {

istream = stdin;

ostream = stdout;

}

else if(m == 2) {

istream = fopen(get\_str(stdin, "Введите название файла (включая его формат .txt): "), "r");

ostream = fopen("results.txt", "w");

if (istream == NULL || ostream == NULL){

puts("ОШИБКА ОПРЕДЕЛЕНИЯ ТИПА ВВОДА!"); break;

}

}

else

puts("Неправильный ввод! Вы можете вводить только символы 1 или 2!");

}

if(istream != NULL && ostream != NULL)

algorithm(istream, ostream);

break;

}

case 3:

return 0;

default:

puts("Вы ввели неправильный символ! Вы можете ввести только 1, 2 или 3!\n");

}

}

}

//ИНФОРМАЦИЯ----------------------------------------------------------------------------------------------------------------------------------

//Вывод описание алгоритма, т.е. его задачу, ввод и вывод

void info()

{

printf("\nАлгоритм был разработан студентом СПбГЭТУ ЛЭТИ Боевом Иваном (группа 0305)\n"

"\nЗадача алгоритма\n"

"Из строк исходного массива сформировать другой текст, в который войдут только строки из исходного текста, "

"в которых после слов, содержащих больше заданного в исходных данных количества одинаковых символов, "

"будет вставлено новое введенное слово.\n"

"\nВходные данные\n"

"1) Количество строк исходного текста\n"

"2) Количество одинаковых символов\n"

"3) Строка символов-разделителей\n"

"4) Исходный текст (в процессе ввода строк текста в них не записываются повторяющиеся после каждого слова символы разделители)\n"

"5) Слова, которые будут вставлены в формируемый текст\n"

"\nВывод\n"

"1) Исходный текст\n"

"2) Сформированный текст, или сообщение о том, что текст не был сформирован\n\n");

}

//Считывает из потока символьную строку и возвращает её.

//Если пользователь выбрал консольный ввод, то также выводит сообщение

STR get\_str(FILE\* stream, STR message)

{

char\* str;

int n;

char c;

if(stream == stdin)

puts(message);

c = get\_next\_symb(stream);

n = 0;

str = (char\*)malloc(1 \* sizeof(char));

do{

if(c == EOF)

return NULL;

str[n] = c;

if (!(str = (char \*) realloc(str, (++n + 1) \* sizeof(char))))

return NULL;

c = fgetc(stream);

} while(c != '\n' && c != '\r');

str[n] = '\0';

return str;

}

//Просит пользователя ввести тип ввода, которым он будет пользоваться.

//Если это файл, то запрашивает ввести его название

//--------------------------------------------------------------------------------------------------------------------------------------------

//---------АЛГОРИТМ---------------------------------------------------------------------------------------------------------------------------

//--------------------------------------------------------------------------------------------------------------------------------------------

//Func declarations

STR read\_text\_str(FILE\* stream, int\*\*\* p, STR del, CI repN, CI i, CI strN);

TEXT form\_text(FILE\* stream, TEXT t, int\*\* p, int\* k, STR del);

//Выводит текст в консоль

void print\_text(FILE\* stream, TEXT t, STR message, CI n){

fprintf(stream, message);

for(int i = 0; i < n; i++)

fprintf(stream, "%s\n", \*(t+i));

}

bool enter\_number(FILE\* stream, STR message, int\* n){

if(stream == stdin)

puts(message);

if(fscanf(stream, "%d", n) == EOF)

return false;

return true;

}

void free\_text(TEXT t, CI n){

if(t){

for(int i = 0; i < n; i++)

if(t[i])

free(t[i]);

free(t);

}

}

void free\_algorithm(STR message, FILE\* stream1, FILE\* stream2, TEXT t1, CI n1, TEXT t2, CI n2, STR str1, int\*\* p){

puts(message);

if(stream1)

if(stream1 != stdin)

fclose(stream1);

if(stream2)

if(stream2 != stdout)

fclose(stream2);

if(str1 != NULL)

free(str1);

if(p) {

for(int i = 0; p[i][0] != -1; i++)

free(p[i]);

free(p);

}

free\_text(t1, n1);

free\_text(t2, n2);

}

//ГЛАВНАЯ ФУНКЦИЯ-----------------------------------------------------------------------------------------------------------------------------

//--------------------------------------------------------------------------------------------------------------------------------------------

void algorithm(FILE\* is, FILE\* os)

{

int strN = 0; //количество строчек в исходном тексте

int strfN = 0; //количество строчек в формируещемся текста

int repN = 0; //количество повторений символов

STR del = NULL; //строка символов разделителей

TEXT pText = NULL; //исходный текст

TEXT fText = NULL; //формирующийся текст

int\*\* p = NULL;

//Начало ввода из потока (файла/консоли)

if(!enter\_number(is, "Введите количество строчек исходного текста: ", &strN) || strN < 1)

return free\_algorithm("ОШИБКА ВВОДА КОЛИЧЕСТВА СТРОК ТЕКСТА!", is, os, pText, strN, fText, strfN, del, p);

clear\_f(is);

if(!enter\_number(is, "Введите количество одинаковых символов в слове: ", &repN) || repN < 0)

return free\_algorithm("ОШИБКА ВВОДА КОЛИЧЕСТВА ОДИНАКОВЫХ СИМВОЛОВ!", is, os, pText, strN, fText, strfN, del, p);

clear\_f(is);

del = get\_str(is, "Введите строку символов-разделителей: ");

if(!del || del[0] == '\0')

return free\_algorithm("ОШИБКА ВВОДА СТРОКИ СИМВОЛОВ РАЗДЕЛИТЕЛЕЙ!", is, os, pText, strN, fText, strfN, del, p);

if(!(pText = (TEXT)malloc(strN \* sizeof(STR))))

return free\_algorithm("ОШИБКА ВЫДЕЛЕНИЯ ПАМЯТИ ДЛЯ ИСХОДНОГО ТЕКСТА!", is, os, pText, strN, fText, strfN, del, p);

if(!(p = (int\*\*)malloc(sizeof(int\*))))

return free\_algorithm("ОШИБКА ВЫДЕЛЕНИЯ ПАМЯТИ ДЛЯ МАССИВА УКАЗАТЕЛЕЙ!", is, os, pText, strN, fText, strfN, del, p);

if(!(\*p = (int\*)malloc(2 \* sizeof(int))))

return free\_algorithm("ОШИБКА ВЫДЕЛЕНИЯ ПАМЯТИ ДЛЯ МАССИВА УКАЗАТЕЛЕЙ!", is, os, pText, strN, fText, strfN, del, p);

p[0][0] = -1;

if(is == stdin)

puts("Введите текст:");

for(int i = 0; i < strN; i++) {

\*(pText+i) = read\_text\_str(is, &p, del, repN, i, strN);

if(\*(pText+i) == NULL)

return free\_algorithm("ОШИБКА ВВОДА СТРОКИ ИСХОДНОГО ТЕКСТА!", is, os, pText, i, fText, strfN, del, p);

}

fText = form\_text(is, pText, p, &strfN, del);

if(strfN == -1)

return free\_algorithm("ОШИБКА ФОРМИРОВАНИЯ ТЕКСТА!", is, os, pText, strN, fText, strfN, del, p);

print\_text(os, pText, "Исходный текст\n", strN);

if(!strfN)

fprintf(os, "Текст не был сформирован\n");

else

print\_text(os, fText, "Сформированный текст\n", strfN);

if(is != stdin)

puts("Результат работы алгоритма записан в файл results.txt");

free\_algorithm("\n", is, os, pText, strN, fText, strfN, del, p);

}

//--------------------------------------------------------------------------------------------------------------------------------------------

//-----------ВВОД СТРОКИ----------------------------------------------------------------------------------------------------------------------

//--------------------------------------------------------------------------------------------------------------------------------------------

//Объявления функций

int\*\* add\_int\_p(int\*\*\* parr, int i, int y);

int\*\* counter\_alloc();

int\*\* free\_counter(int\*\*\* arr);

char\* add\_symb(char\*\* s, const char c);

int\*\* count\_symbol(int\*\*\* m, char c);

bool counter\_repeat\_check(int\*\* counter, CI repN);

bool str\_repeat\_check(STR s, const char c);

//ГЛАВНАЯ ФУНКЦИЯ--------------------------------------------------------------------------------------------------------------------------------

//--------------------------------------------------------------------------------------------------------------------------------------------

STR read\_text\_str(FILE\* stream, int\*\*\* p, STR del, CI repN, CI i, CI strN)

{

int n; //количество введенных символов

char\* str; //вводимая строка

int\*\* counter; //счётчик букв в слове

char c; //вводимый символ

char\* dCheck; //введённые разделители

if(!(str = (char\*)malloc(sizeof(char))))

return NULL;

if(!(counter = counter\_alloc()))

return NULL;

if(!(dCheck = (char\*)malloc(sizeof(char))))

return NULL;

dCheck[0] = '\0';

n = 0;

c = 0;

while(c != '\n' && c != '\r' && c != EOF) {

if(stream != stdin && n == 0) //если это файловый ввод и вводится первый символ

c = get\_next\_symb(stream);

else

c = fgetc(stream);

if(c == EOF && i != strN - 1)

return NULL;

else if (str\_repeat\_check(del, c) || c == '\r' || c == '\n' || c == EOF) //если введенный символ - конец строки

{

if (counter\_repeat\_check(counter, repN)) //проверка на одинаковые буквы в слове

if (!(\*p = add\_int\_p(p, i, n)))

return NULL;

counter = free\_counter(&counter); //освобождаем память, выделенную под счётчик

if (c == '\r' || c == '\n' || c == EOF)

str[n] = '\0';

else {

if (!(counter = counter\_alloc())) //создаём новый

return NULL;

if (!str\_repeat\_check(dCheck, c)) //если разделитель не повторяется

{

if (!(dCheck = add\_symb(&dCheck, c)))

return NULL; //добавялем его в повторенные

if (!(str = (char \*) realloc(str, (n + 2) \* sizeof(char))))

return NULL;

str[n++] = c; //вписываем разделитель в текст

}

}

}

else{

if (!(counter = count\_symbol(&counter, c))) //подсчитываем символ

return NULL;

if (!(str = (char \*) realloc(str, (n + 2) \* sizeof(char))))

return NULL;

str[n++] = c; //вставляем символ в текст

}

}

free(dCheck);

return str;

}

//Проверяет счётчик повторений символов в слове

//Если в введённом слове есть символ, количество повторений которой больше

//заданного, то функция возвращает true. Иначе - false

bool counter\_repeat\_check(int\*\* c, CI repN)

{

int i;

i = 0;

while(c[i][0] != 0){

if(c[i++][1] > repN)

return true;

}

return false;

}

//Проверяет строку s на повторение символа c.

//Если символ c есть в строке, функция возвращает true.

//Если символа в строке нет, функция возвращает false

bool str\_repeat\_check(STR s, const char c)

{

int i = 0;

while(s[i] != '\0')

if(c == s[i++])

return true;

return false;

}

//Добавляет символ c в строку s

char\* add\_symb(char\*\* str, const char c)

{

int i = 0;

while((\*str)[i] != '\0')

i++;

if(!(\*str = (char\*)realloc(\*str, (i + 2) \* sizeof(char))))

return NULL;

(\*str)[i] = c; (\*str)[i+1] = '\0';

return \*str;

}

//Добавляет указатель на символ строки в массив указателей

int\*\* add\_int\_p(int\*\*\* parr, int x, int y)

{

int i = 0;

while((\*parr)[i][0] != -1)

i++;

if(!(\*parr = (int\*\*)realloc(\*parr, (i + 2) \* sizeof(int\*))))

return NULL;

(\*parr)[i] = malloc(2 \* sizeof(int));

(\*parr)[i][0] = x; (\*parr)[i][1] = y;

(\*parr)[i+1] = malloc(2 \* sizeof(int));

(\*parr)[i+1][0] = -1;

return \*parr;

}

//Создаёт счётчик

int\*\* counter\_alloc()

{

int\*\* arr;

arr = (int\*\*)malloc(sizeof(int\*));

arr[0] = (int\*)malloc(2 \* sizeof(int));

arr[0][0] = 0; arr[0][1] = 0;

return arr;

}

//В счётчике m обавляет +1 к количеству

//повторений символа c

int\*\* count\_symbol(int\*\*\* m, char c)

{

int i = 0;

while((\*m)[i][0] != (int)c)

if((\*m)[i][0] == 0)

{

(\*m)[i][0] = (int)c; (\*m)[i][1] = 1;

int\*\* temp = (int\*\*)realloc(\*m, (i + 2) \* sizeof(int\*));

if(temp == NULL)

{

puts("REALLOC ОШИБКА!!!");

return NULL;

}

\*m = temp;

if(!((\*m)[i+1] = (int\*)malloc(2 \* sizeof(int))))

return NULL;

(\*m)[i+1][0] = 0;

return \*m;

}

else

i++;

(\*m)[i][1]++;

return \*m;

}

//Освобождает память, выделенную для счётчика

int\*\* free\_counter(int\*\*\* arr)

{

int i = 0;

if(\*arr == NULL)

return NULL;

while((\*arr)[i][0] != 0){

i++;

}

while(i){

free((\*arr)[i]);

i--;

}

free(\*arr[0]);

free(\*arr);

return NULL;

}

//--------------------------------------------------------------------------------------------------------------------------------------------

//----------ФУНКЦИЯ ФОРМИРОВАНИЯ ТЕКСТА-------------------------------------------------------------------------------------------------------

//--------------------------------------------------------------------------------------------------------------------------------------------

//Объявление функций

int insert\_word(FILE\* stream, char\*\* s, int index, bool last);

//ГЛАВНАЯ ФУНКЦИЯ-----------------------------------------------------------------------------------------------------------------------------

//--------------------------------------------------------------------------------------------------------------------------------------------

TEXT form\_text(FILE\* stream, TEXT t, int\*\* p, int\* k, STR del) {

TEXT fT;

int ind;

char \*str;

int m = 0;

\*k = 0;

int i, j;

fT = NULL;

m = 0;

i = 0;

while (p[m][0] != -1) {

if(p[m][0] == i)

{

str = (char \*) malloc(sizeof(char));

ind = 0;

j = 0;

while (t[i][j] != '\0' || p[m][0] == i && p[m][1] == j) {

if (p[m][0] == i && p[m][1] == j) { //если мы встретили строку, куда надо вставить слово

if (!m && stream == stdin)

puts("Введите слова для вставки в текст");

int b = insert\_word(stream, &str, j + ind, p[m+1][0] == -1);

if(b == 0){

str[j + ind] = del[0];

ind++;

}

else

ind += b;

m++;

}

str = (char \*) realloc(str, (j + ind + 2) \* sizeof(char));

str[j + ind] = t[i][j];

j++;

}

str[j + ind] = '\0';

if (!fT)

fT = (TEXT) malloc(sizeof(STR));

else

fT = (TEXT) realloc(fT, (\*k + 1) \* sizeof(STR));

if (!fT) {

\*k = -1;

return NULL;

}

\*(fT + \*k) = str;

(\*k)++;

}

i++;

}

return fT;

}

//Вставляет слово из потока в вводимую строку s

int insert\_word(FILE\* stream, char\*\* s, int index, bool last)

{

int n = 0;

char c;

if(stream != stdin)

c = get\_next\_symb(stream);

else

c = fgetc(stream);

while(c != '\n' && c != '\r' && c != EOF) {

\*s = (char \*) realloc(\*s, (index + n + 1));

(\*s)[index + n] = c;

n++;

c = fgetc(stream);

}

return n;

}

**Вывод**

В результате работы были закреплены практические навыки в написании программ на языке программирования СИ, а именно в разработке алгоритмов, обработке строк, работе с двухмерными массивами, работе с файлами.