Министерство науки и высшего образования РФ

ФГБОУ ВО

Московский авиационный институт

(национальный исследовательский университет)



Каф 304

Пояснительная записка к курсовому проекту

по курсу: «Программирование на языке высокого уровня»

Выполнил: Королев П.В

Группа: M30-210Б-18

Принял: доцент, к.т.н Роговцев А. А.

Москва, 2019

Оглавление

[Теоретическая часть 3](#_Toc27670708)

[Структурные схемы алгоритмов программы и подпрограмм 4](#_Toc27670709)

[**Основная программа** 4](#_Toc27670710)

[**Подпрограмма: process** 5](#_Toc27670711)

[**Подпрограмма: compare** 6](#_Toc27670712)

[**Подпрограмма: findWordsByLen** 7](#_Toc27670713)

[**Подпрограмма: toLowString** 8](#_Toc27670714)

[**Подпрограмма: toLowRegistr** 9](#_Toc27670715)

[**Подпрограмма: listInit** 10](#_Toc27670716)

[**Подпрограмма: listAdd** 11](#_Toc27670717)

[**Подпрограмма: listClear** 12](#_Toc27670718)

[Реализация на СИ программы и подпрограмм 13](#_Toc27670719)

[**Заголовочные файлы** 13](#_Toc27670720)

[mylib.h 13](#_Toc27670721)

[List.h 14](#_Toc27670722)

[**Исполняемые файлы** 15](#_Toc27670723)

[Основная программа: main.c 15](#_Toc27670724)

[Подпрограмма: process.c 17](#_Toc27670725)

[Подпрограмма: compare .c 18](#_Toc27670726)

[Подпрограмма: findWordsByLen.c 19](#_Toc27670727)

[Подпрограмма: toLowString.c 20](#_Toc27670728)

[Подпрограмма: toLowRegistr.c 21](#_Toc27670729)

[Подпрограмма: listInit.c 22](#_Toc27670730)

[Подпрограмма: listAdd.c 23](#_Toc27670731)

[Подпрограмма: listClear.c 24](#_Toc27670732)

[Пояснение к работе программы 25](#_Toc27670733)

[Результаты работы программы 26](#_Toc27670734)

# **Теоретическая часть**

**Детальный анализ поставленной задачи:**

Входные параметры программы: имя входного файла, имя выходного файла, длина слова.

Входной текстовый файл должен быть закодирован кодировкой ASCII WIN1251, включающий в себя тексты как английского, так и русского алфавита. Размер файла может быть любым.

Словом является набор символов, оканчивающийся одним из символов-разделителей: (пробел)\t\n,.")}]!?>;:

Максимальная длина слова русского языка 55.

Пример слова:

**Тетрагидропиранилциклопентилтетрагидропиридопиридиновые** (химич. вещ-во)

Перенос слова на новую строку не учитывается!

Для хранения слов наилучшим образом будет использование непрерывной области памяти, разделенной на слова одинаковой длины. Выделяется заранее большая область памяти, которая постепенно заполняется символами. В случае нехватки памяти, память выделяется снова, но уже большего размера.

Такой массив слов здесь и далее будем называть списком слов, или просто списком.

У такого списка есть параметры:

- вместимость слов (capacity)

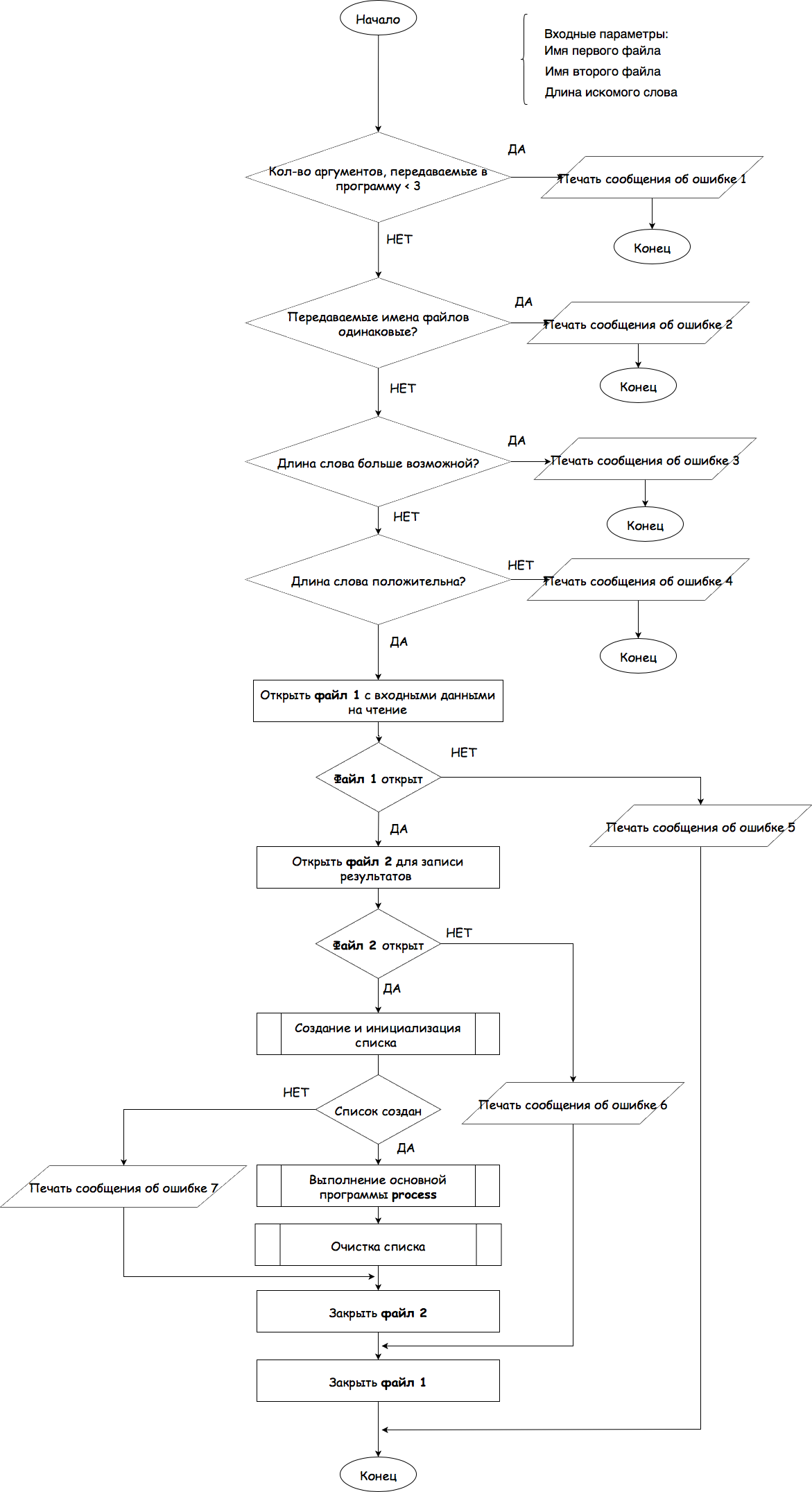
- текущее кол-во слов (numWords)

- размер в байтах (size)

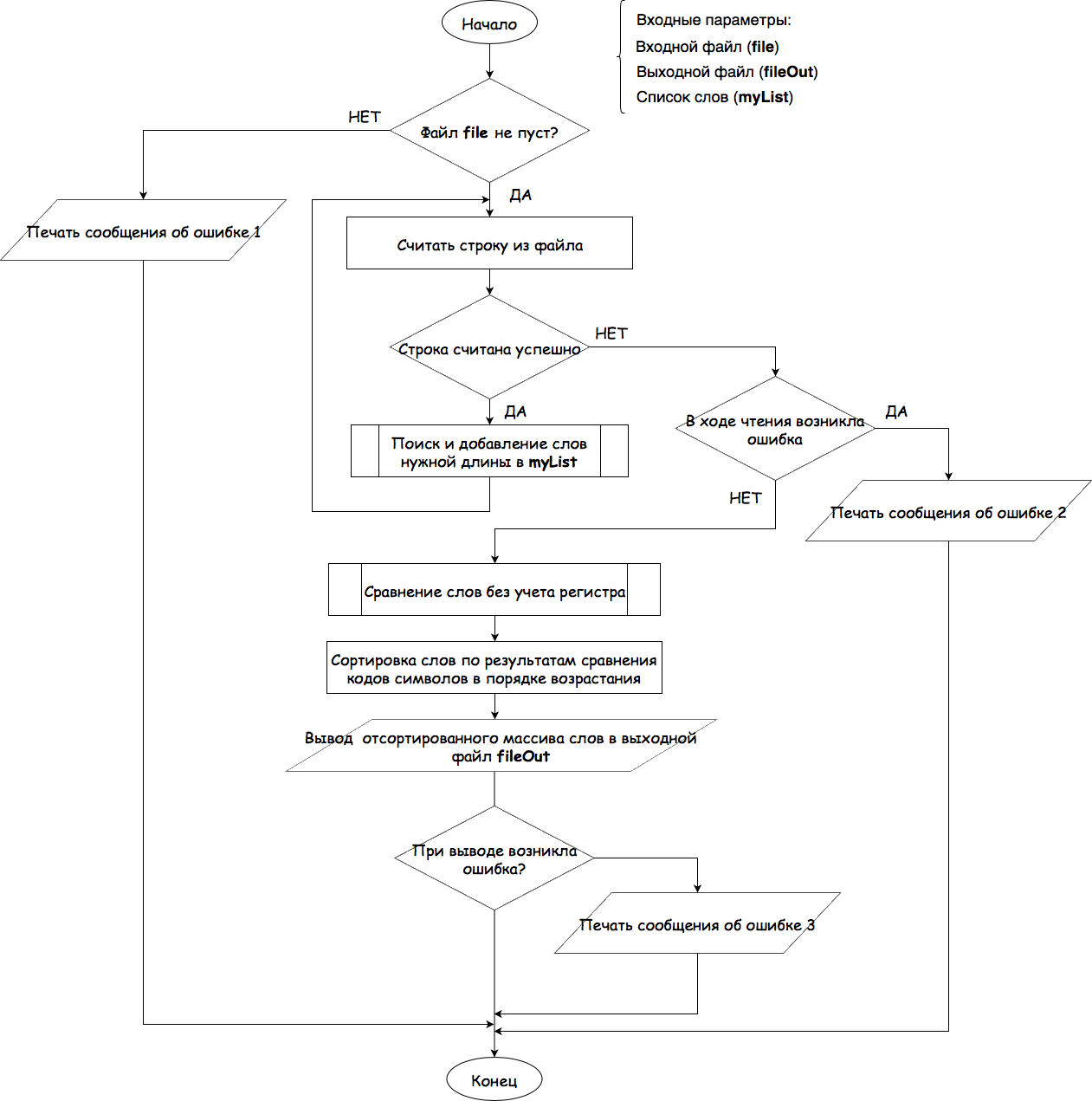
Пример списка: Петя\0Маша0\Ваня0\Миша0\

# **Структурные схемы алгоритмов программы и подпрограмм**

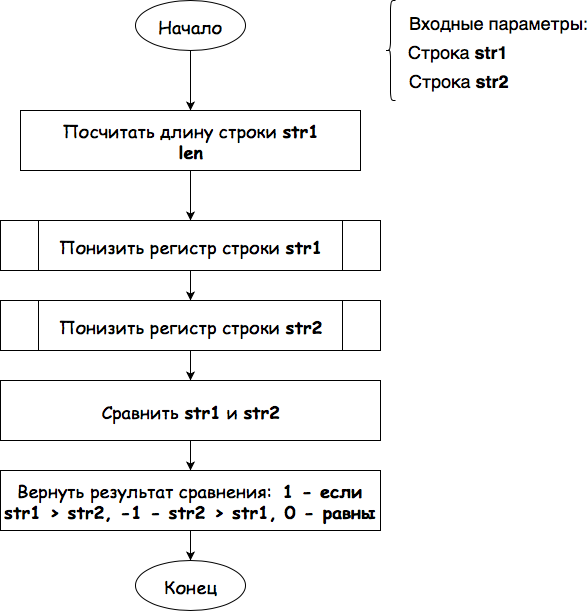
## **Основная программа**



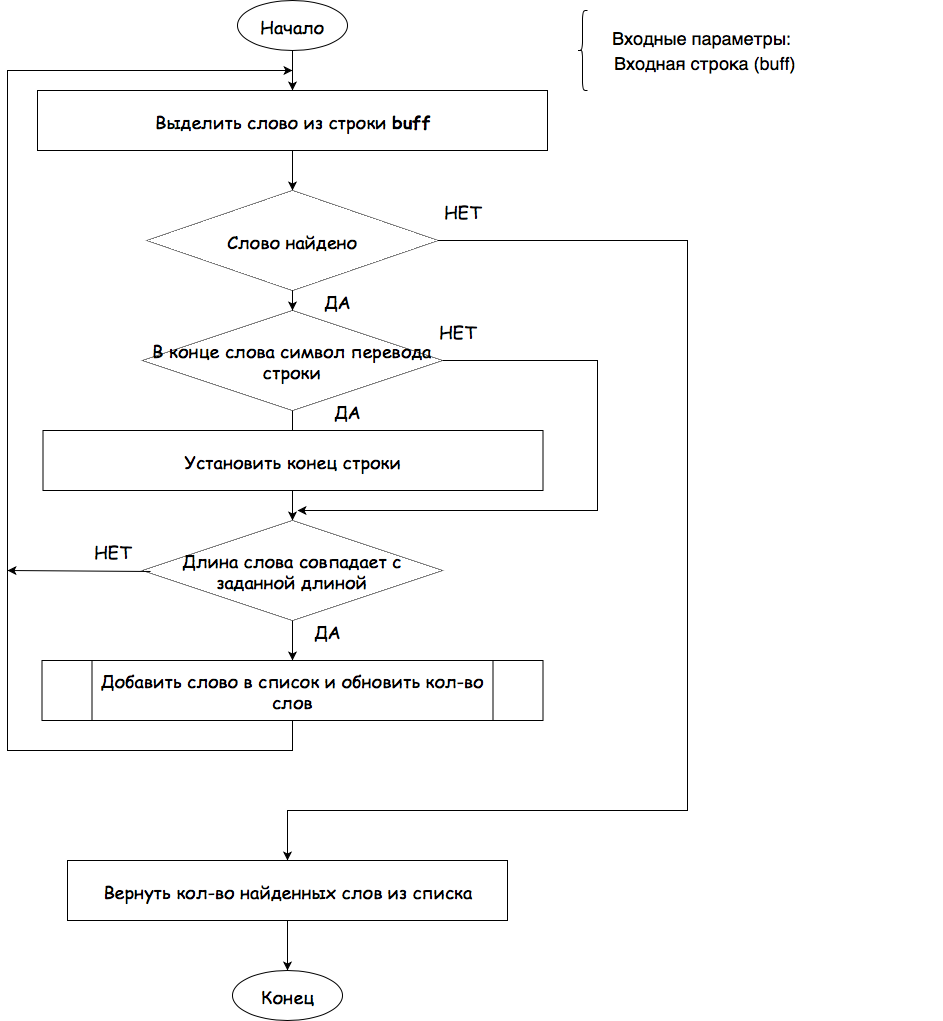
## **Подпрограмма: process**



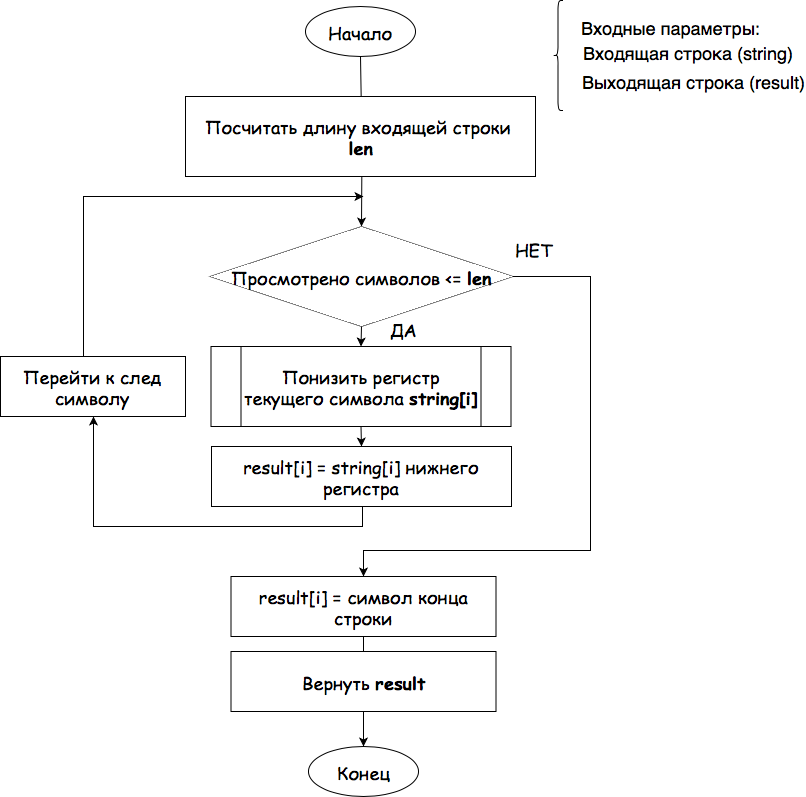
## **Подпрограмма: compare**



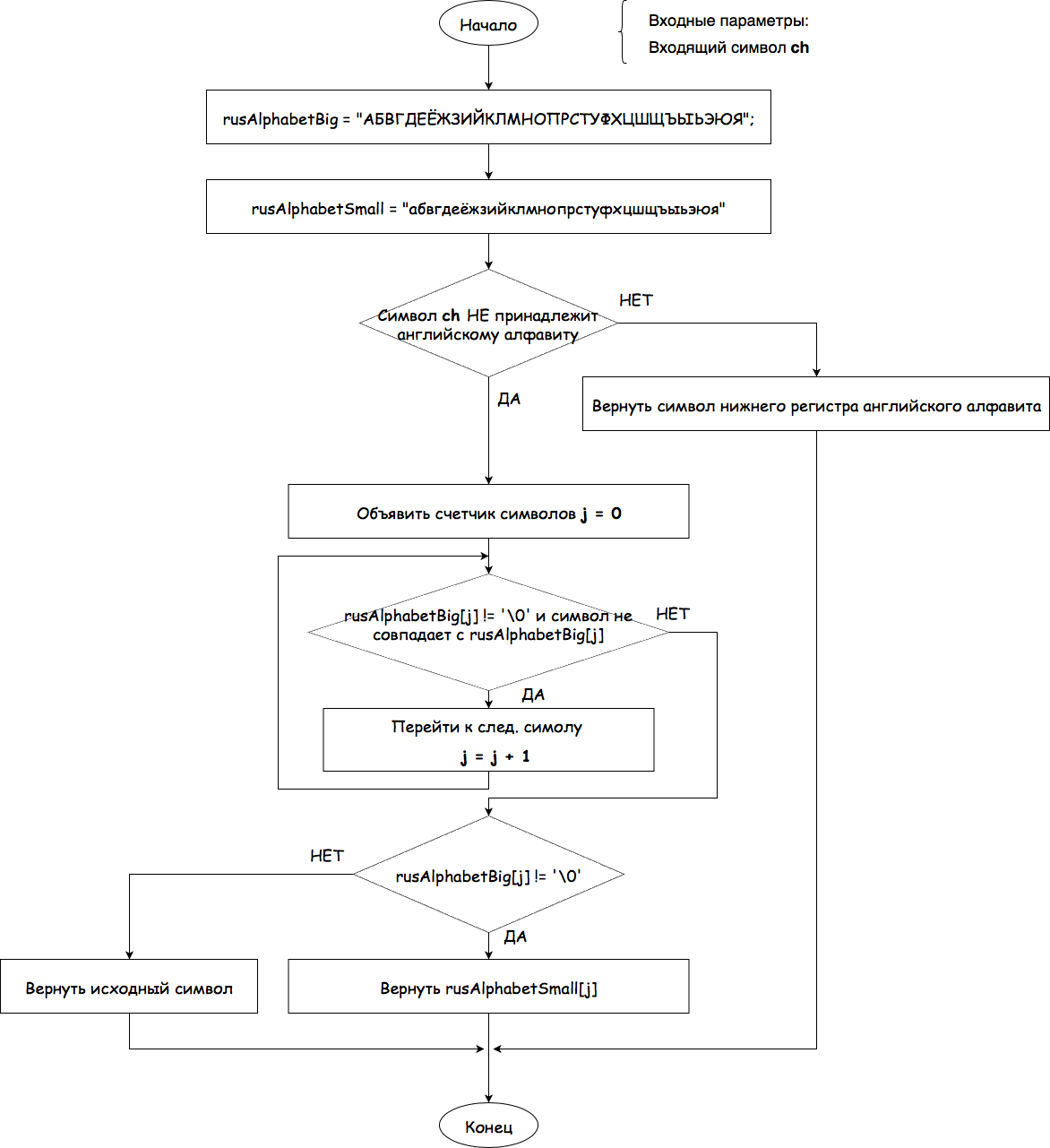
## **Подпрограмма: findWordsByLen**



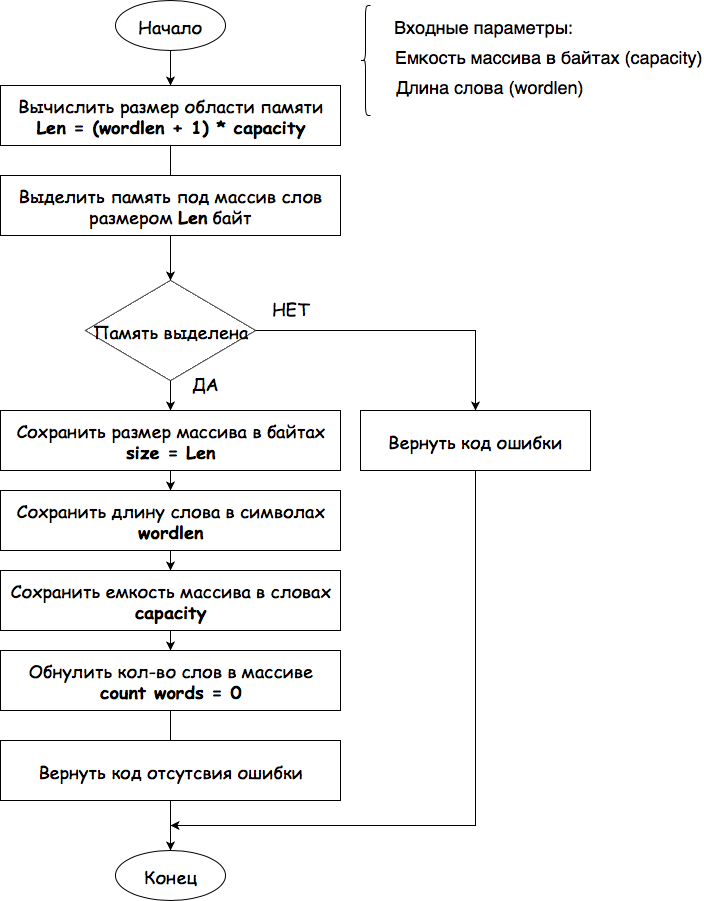
## **Подпрограмма: toLowString**



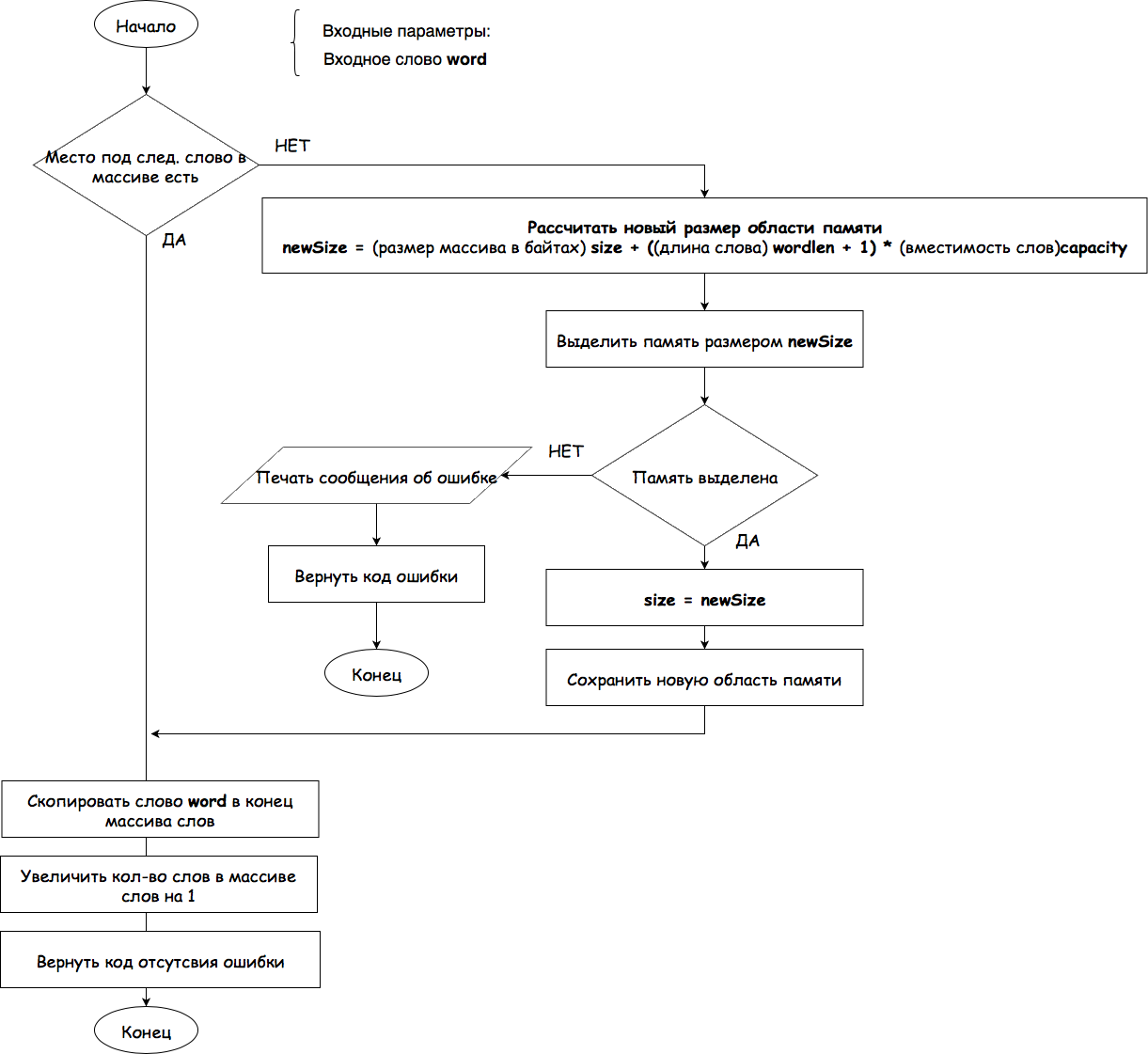
## **Подпрограмма: toLowRegistr**



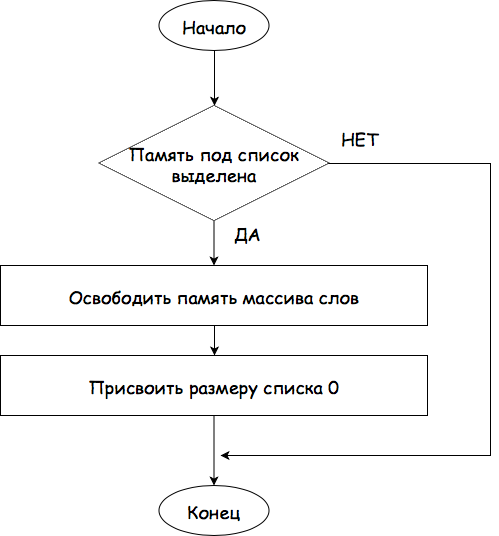
## **Подпрограмма: listInit**



## **Подпрограмма: listAdd**



## **Подпрограмма: listClear**



# **Реализация на СИ программы и подпрограмм**

## **Заголовочные файлы**

### mylib.h

#include <stdio.h>

#include "List.h"

#include <string.h>

#include <ctype.h>

#include <stdlib.h>

#ifndef KURSOVAYA\_MYLIB\_H

#define KURSOVAYA\_MYLIB\_H

typedef unsigned char BOOL;

#define TRUE 1

#define FALSE 0

static const STR\_LEN = 1024;

/// Функция toLowString

/// \param result - переменная, куда будет записан результат

/// \param string - переменная, откуда берется строка (источник)

/// \return - возвращает указатель на строку с символами нижнего регистра

char\* toLowString(char\* result, char\* string);

/// Функция isEnglishLetter

/// \param ch - исходный символ

/// \return - возвращает 1, если ch символ английского алфавита, иначе 0

BOOL isEnglishLetter(char ch);

/// Функция toLowRegistr

/// \param ch - исходный символ

/// \return - возвращает символ, преобразованый в нижний регистр

char toLowRegistr(char ch);

/// Функция findWordsByLen

/// \param mylist - список, который заполненяется позициями на начала слов нужной длины

/// \param buff - буффер, откуда производится считывание данных

/// \param len - нужная длина слова

/// \return

int findWordsByLen(MyList\* mylist, char\* buff);

#endif //KURSOVAYA\_MYLIB\_H

### List.h

#include <stdlib.h>

#include <mm\_malloc.h>

#include <string.h>

#include <errno.h>

#include <stdio.h>

typedef struct MyList

{

unsigned char wordlen; // длина слова в символах

size\_t size; // размер массива в байтах

unsigned short capacity; // емкость массива в словах

unsigned int countWords; // текущее кол-во слов в массиве

char\* data;

} MyList;

int listInit(MyList\* list, unsigned short capacity, unsigned char wordlen);

void listShow(const MyList\* list);

void listClear(MyList\* list);

int listAdd(MyList \*myList, const char\* word);

## **Исполняемые файлы**

### Основная программа: main.c

int main (int argc, char\*\* argv) {  
 if(argc < 3)  
 {  
 printf("\nОшибка! Не достает кол-во входных аргументов! Передача парметров должна осущеcтвляться в виде:\n"  
 "\*\* Первый аргумент - входной файл\n"  
 "\*\* Воторй аргумент - выходной файл\n"  
 "\*\* Третий аргумент - длина искомых слов (макс. длина возможного искомого слова: 55 букв)\n ");  
 return -1;  
 }  
  
 if(strcmp(argv[1], argv[2]) == 0)  
 {  
 printf("\nОшибка! Путь и имя выходного файла не должны совпадать с входным файлом!");  
 return -2;  
 }  
 if(atoi(argv[3]) > MAX\_LEN\_WORD)  
 {  
 printf("\nОшибка! Слово указанной длины ни в русском, ни в английском языке не существует, хотите ли продолжить? Y,y - да, иначе - нет\n"  
 "Введите с клавиатуры: ");  
 char ch;  
 ch = getchar();  
 if(ch != 'Y' && ch != 'y')  
 {  
 printf("\nЗавершение работы программы.");  
 return -3;  
 }  
 }  
  
 if(atoi(argv[3]) <= 0)  
 {  
 printf("\nОшибка! Длина искомого слова должна быть положительной!");  
 return -4;  
 }  
  
 FILE\* file = NULL;  
 FILE\* fout = NULL;  
 MyList myList;  
  
 int wordlen = atoi(argv[3]);  
 int error = 0;  
  
 const char\* fName = argv[1];  
 const char\* fOutName = argv[2];  
  
 unsigned short listCapacity = 10; // вместимость списка  
 //Создание и инициализация списка  
 if((error = listInit(&myList, listCapacity, wordlen)) != 0)  
 {  
 printf("Ошибка выделения памяти");  
 return error;  
 }  
  
 //Открытие файла на чтение  
 if((file = fopen(fName, "r")) == NULL)  
 {  
 perror("Ошибка, файл для чтения не открыт!");  
 listClear(&myList);  
 return -1;  
 }  
  
 if((fout = fopen(fOutName, "w")) == NULL)  
 {  
 perror("Ошибка, не удалось открыть файл на запись!");  
 fclose(file);  
 listClear(&myList);  
 return -4;  
 }  
  
 process(file, fout, &myList);  
  
 //Закрытие файлов  
 fclose(fout);  
 fclose(file);  
  
 //Очистка списка, высвобождение памяти  
 listClear(&myList);  
  
 return 0;  
}

### Подпрограмма: process.c

void process(FILE\* file, FILE\* fout, MyList\* myList)  
{  
 char buff[STR\_LEN];  
 if(fgets(buff, STR\_LEN, file) == NULL)  
 {  
 if(ftell(file) == 0)  
 {  
 printf("Входной файл пуст");  
 return;  
 }

//иначе, мы ничего не можем сделать, даже если ftell неправильно сработал!  
 else  
 {  
 perror("Ошибка при работе с входым файлом!");  
 return;  
 }  
  
 }  
  
 do{  
 //поиск слов заданной длины  
 findWordsByLen(myList, buff);  
 }while (fgets(buff, STR\_LEN, file) != NULL);  
  
 if(ferror(file))  
 {  
 perror("Ошибка чтения из файла!");  
 return;  
 }  
  
  
 qsort(myList->data, myList->countWords, myList->wordlen + 1, compare);  
  
 for(int i = 0; i < myList->countWords; i ++)  
 {  
 fwrite(myList->data + i \* (myList->wordlen + 1), myList->wordlen , 1, fout);  
 if(ferror(fout))  
 {  
 perror("Ошибка записи в выходной файл!");  
 return;  
 }  
  
 fputc('\n',fout);  
 }  
}

### Подпрограмма: compare .c

int compare (void\* \_str1, void\* \_str2)  
{  
 char\* str1 = (char\*)\_str1;  
 char\* str2 = (char\*)\_str2;  
  
 size\_t len = strlen(str1);  
  
 char buff1[len];  
 char buff2[len];  
  
 return strcmp(toLowString(buff1, str1), toLowString(buff2, str2));  
}

### Подпрограмма: findWordsByLen.c

int findWordsByLen(MyList\* mylist, char\* buff)

{

assert(mylist != NULL);

assert(buff != NULL);

const char\* endWordSymb = " \t,;:!?.)}>";

char\* pWord; // указатель на лексему

pWord = strtok(buff, endWordSymb); // выделяем слово и

//сохраняем указатель на него

while (pWord != NULL)

{

if(pWord[strlen(pWord) - 1] == '\n') // Если в конце слова \n

// поставить вместо него \0

pWord[strlen(pWord) - 1] = '\0';

if(pWord[(int)strlen(pWord)-1] == '"')

pWord[(int)strlen(pWord)-1] = '\0';

if(strlen(pWord) == mylist->wordlen) // Если длина текущего равна // len

listAdd(mylist, pWord); // Добавить в конец списка

pWord = strtok (NULL, endWordSymb); // Выделение очередной части // строки (слова)

}

return mylist->countWords;

}

### Подпрограмма: toLowString.c

char\* toLowString(char\* result, char\* string)

{

size\_t len = strlen(string);

size\_t i;

for(i = 0; i < len; i ++)

result[i] = toLowRegistr(string[i]);

result[i] = '\0';

return result;

}

### Подпрограмма: toLowRegistr.c

char toLowRegistr(char ch)  
{  
 assert(ch != '\0');  
  
 char\* rusAlphabetBig = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦШЩЪЫЬЭЮЯ";  
 char\* rusAlphabetSmall = "абвгдеёжзийклмнопрстуфхцшщъыьэюя";  
  
 //Если символ аншлийского алфавита  
 if((ch >= 65 && ch <= 90) || (ch >= 97 && ch <= 122))  
 return (char)tolower(ch);  
  
 int j;  
 for(j = 0; rusAlphabetBig[j] != '\0' && ch != rusAlphabetBig[j]; j ++);  
  
 if(rusAlphabetBig[j] != '\0')  
 return rusAlphabetSmall[j];  
 else  
 return ch;  
}

### Подпрограмма: listInit.c

int listInit(MyList\* list, unsigned short capacity, unsigned char wordlen)

{

assert(list != NULL);

int buflen = (wordlen + 1) \* capacity;

list->data = (char\*) malloc(buflen);

if(list->data == NULL)

return -5;

list->size = buflen;

list->wordlen = wordlen;

list->capacity = capacity;

list->countWords = 0;

return 0;

}

### Подпрограмма: listAdd.c

int listAdd(MyList\* list, const char\* word)

{

assert(list != NULL);

if((list->countWords + 1) \* (list->wordlen + 1) > list->size)

{

size\_t newSize = list->size + (list->wordlen + 1) \* list->capacity;

char\* data = (char\*)realloc(list->data, newSize);

if(data != NULL)

{

list->size = newSize;

list->data = data;

}

else

{

perror("Ошибка выделения памяти");

return -5;

}

}

strcpy(list->data + list->countWords \* (list->wordlen + 1), word);

list->countWords++;

return 0;

}

### Подпрограмма: listClear.c

void listClear(MyList\* list)

{

assert(list != NULL);

if(list != NULL)

{

free(list->data);

list->data = NULL;

list->size = 0;

}

}

# **Пояснение к работе программы**

Перед запуском программы убедитесь, что тексты программ закодированы кодировкой WIN1251.

Исходные файлы должны быть закодированы также в кодировке WIN1251.

Программа вызывается из консоли командной строкой.

Пример: Kursovaya.exe file.ANSI fileOut.ANSI 4

\*\* Первый аргумент - входной файл

\*\* Воторй аргумент - выходной файл

\*\* Третий аргумент - длина искомых слов

Программа написана на macOS в среде разработки JetBrains CLion 2018.3.4

# **Результаты работы программы**

**1) Подадим на вход текстовый файл (ищем слово длины 5):**

У Клары Карл украл каралл...

На дворе трава на траве - дрова...

**На выходе получим:**

дворе

дрова

Клары

трава

траве

украл

**Замечание:** поскольку в задании было сказано отсортировать в алфавитном порядке, то программа намеренно сортирует без учета регистра букв, то есть преобразуя в нижний регистр строки, и сортируя именно в алфавитном порядке. Вывод же в файл программа осуществляет в том виде, в каком было найдено слово без изменения регистра!

2) **Подадим на вход текстовый файл (ищем слово длины 4):**

My favorite hero from Marvel's films is Grood. He said "I am Grood"

Watson - is the best friend for Sherock Holmes.

Doctor Strange form Marvel's film livеs in the Blacker Street 221B, Sherlock holmes lives in the Backer Street 221B...

**На выходе получим:**

221B

221B

best

film

form

from

hero

said

**Замечание:** программа не должна учитывать повторение одно и того же слова, как например 221B, поскольку эти два слова встречены в разном контексте, и встречаются в разных местах. Реализация без учета повторений потребует добавление алгоритма, удаляющего повторные слова - но в техническом задании этого не требуется, а значит - такой результат нас устраивает.

3) **Подадим на вход тот же текстовый файл п.2 (ищем слово длины 6):**

**На выходе получим:**

Backer

Doctor

friend

holmes

Holmes

Street

Street

Watson

3) **Подадим на вход тот же текстовый файл п.2 (ищем слово длины 8):**

**На выходе получим:**

favorite

Marvel's

Marvel's

Sherlock