**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по домашнему заданию

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-11 |  | преподаватель каф. ИУ5 |
| Михалёв Ярослав |  | Козлов А.Д. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2021 г.

Постановка задачи

Слова текста из малых латинских букв записаны не менее чем через один пробел; текст оканчивается точкой. БЕЗ ИСПОЛЬЗОВАНИЯ конструкции STRING:

а) написать программу ввода такого текста с клавиатуры;

б) написать все слова, отличающиеся от последнего слова, и совпадающие с конечным отрезком из алфавита (z, yz, xyz и т.д.)

Разработка алгоритма

Для хранения текста используется динамический массив символов.

Проходим циклом по всем словам. Проверяем, если слово:

* Не состоит полностью из пробелов
* Отличается от последнего слова
* Совпадает с конечным отрезком из алфавита (z, yz, xyz и т.д.)

Если все условия выполняются, то печатаем это слово

**Описание входных, выходных и вспомогательных данных:**

Входные данные:

char size - количество символов в тексте

char \*text – динамический массив символов

Выходные данные:

Слова, удовлетворяющие условию задачи

**Список функций:**

void PrintText(char text [], int size)

Выводит массив символов на экран

Входные данные

* char text[] – массив символов
* int size - размер массива text

char\* GenerateText(int size)

Возвращает текст из случайных латинских символов

Входные данные

* int size – размер

Выходные данные

* char\* text - динамический массив из случайных латинских символов

char\* EnterText(int size)

Вводит текст с клавиатуры

Входные данные:

* int size – количество символов в тексте

Выходные данные;

* char\* text – текст

сhar\* ReadFile(const char\* title, int size)

Считывает текст из файла

Входные данные:

* const char\* title – название файла
* int size – количество символов в тексте

Выходные данные:

* char\* s – динамический массив символов

int GetLastEntry(char\* text, int size, char symbol)

// Возвращает индекс последнего вхождения символа в текст

Входные данные

* char\* text – текст
* int size – количество символов в тексте
* char symbol – символ

Выходные данные

* int i – индекс последнего вхождения символа в текст
* -1 – если текст не содержит данный символ

char\* GetLastWord(char\* text, int size, int& wordsize)

Возвращает последнее слово в тексте

* char\* text – текст
* int size – количество символов в тексте
* int& wordsize – указатель на перемененную

Выходные данные

* char\* result – последнее слово в тексте
* int& wordsize – длинна последнего слова

int SpacesCount(char\* text, int size)

Возвращает количество пробелов в тексте

Входные данные:

* char\* text – текст
* int size – размер текста

Выходные данные:

* int count – количество пробелов в тексте

int\* FindSpaces(char\* text, int size)

Возвращает массив с индексами пробелов

Входные данные:

* char\* text – текст
* int size – размер текста

Выходные данные:

* int\* SpacesIdx – массив с индексами пробелов текста text

bool IsOnlyContainSpaces(char\* word, int len)

Проверяет, состоит ли слово целиком из пробелов

Входные данные

* char\* word – слово
* int len – длинна слова

Выходные данные

* true - если слово целиком состоит из пробелов
* false - если в слове есть хотя бы один пробел

char\* Trim(char\* text, int idx1, int idx2, int& wordsize)

Обрезает текст по заданным границам

Входные данные:

* char\* text – текст
* int idx1 – левая граница
* int idx2 – правая граница
* int& wordsize – указатель на переменную

Выходные данные:

* char\* result – обрезанный текст
* int& wordsize – длинна обрезанного текста

bool CompareWord(char\* word1, int size1, char\* word2, int size2)

Сравнивает два слова

Входные данные:

* char\* word1 – первое слово
* int size1 – длина первого слова
* char\* word2 – второе слово
* int size2 – длина второго слова

Выходные данные:

* true – если слова совпадают,
* false – если слова отличаются

bool CheckWord(char\* word, int size)

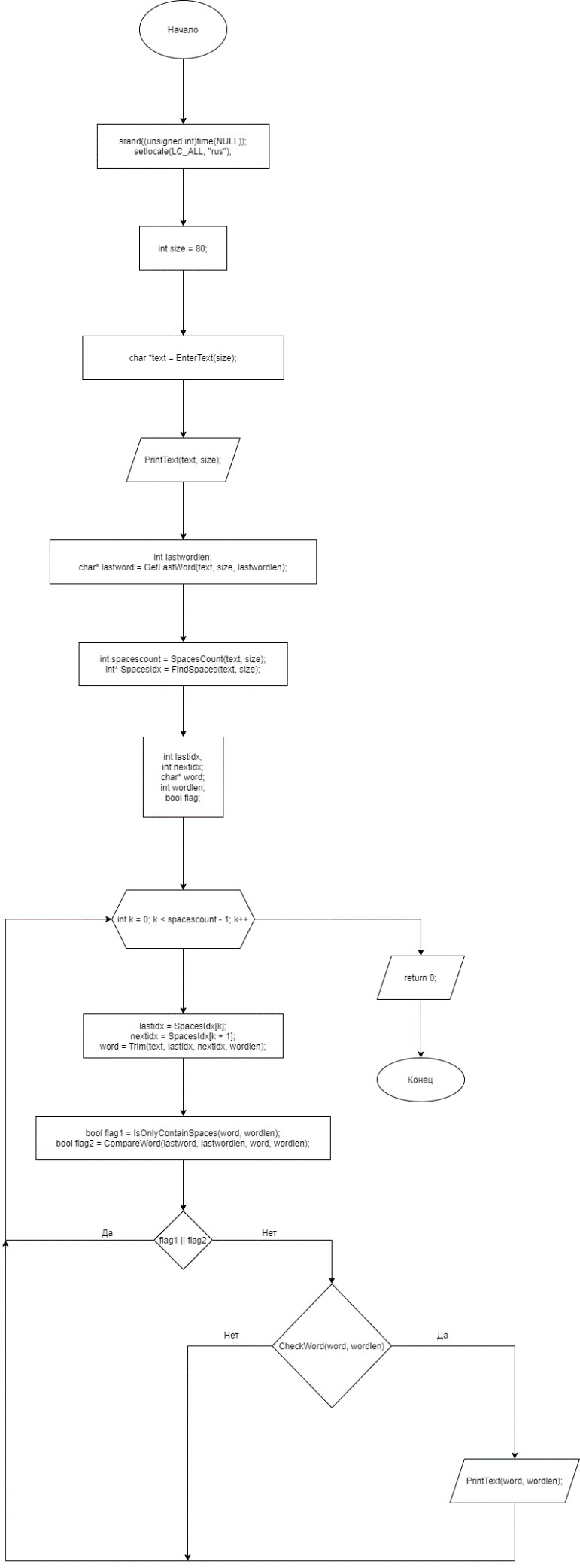
Проверяет, совпадает ли слово с конечным отрезком из алфавита(z, yz, xyz, ...)

Входные данные

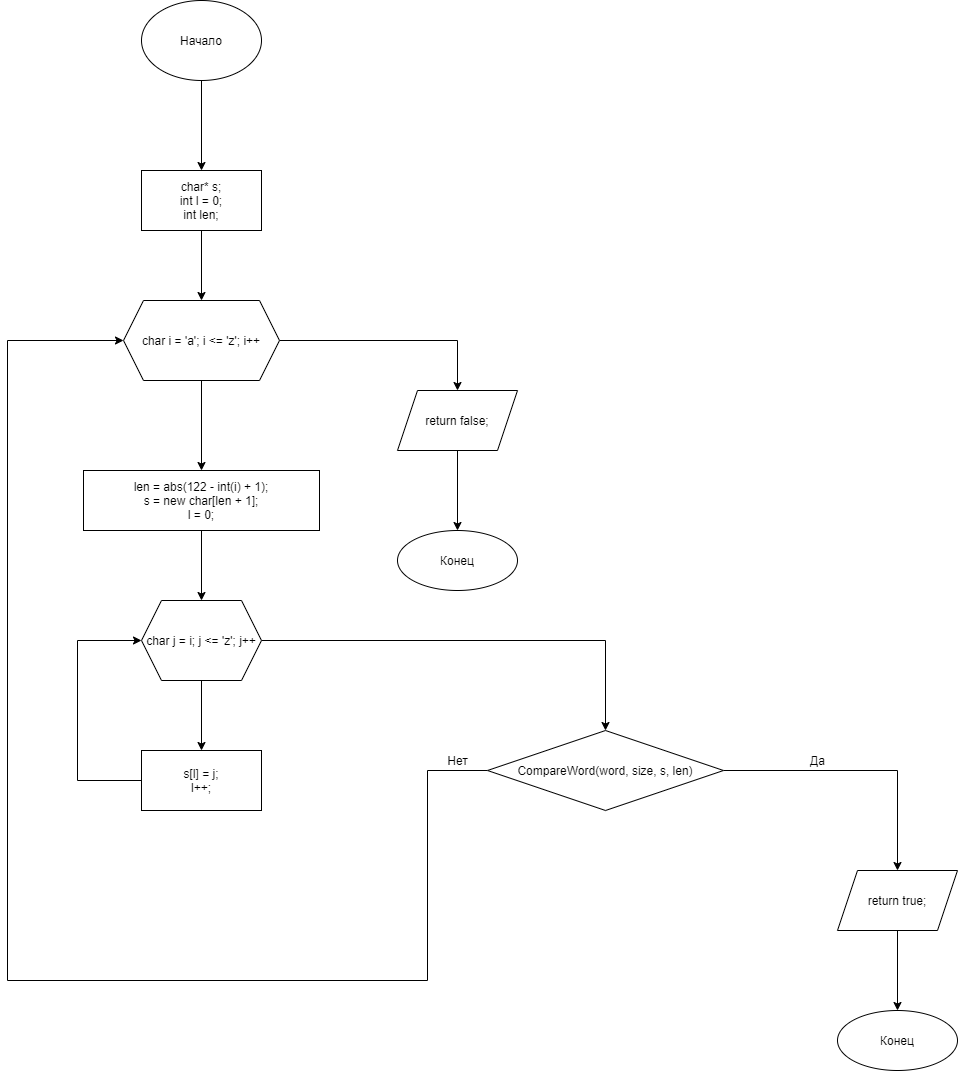
* char\* word – слово
* int size – длина слова

Схема алгоритма

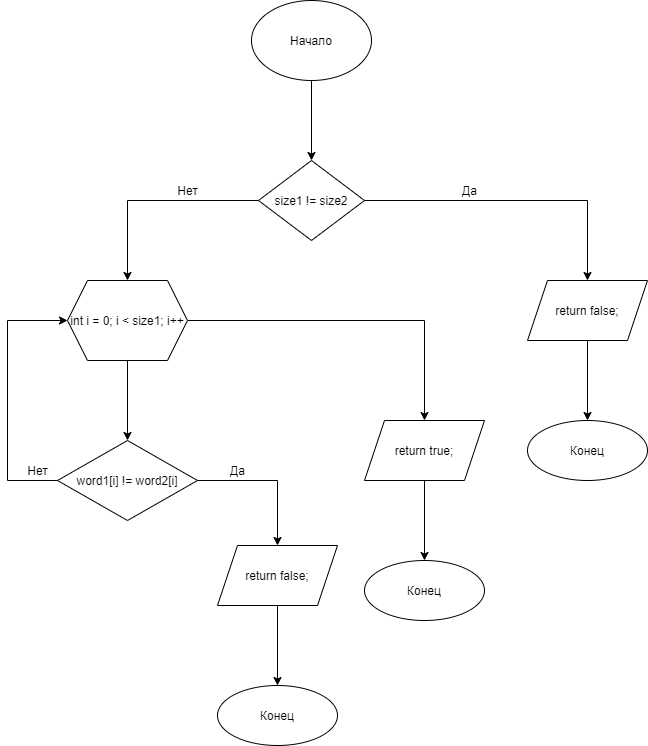
Main



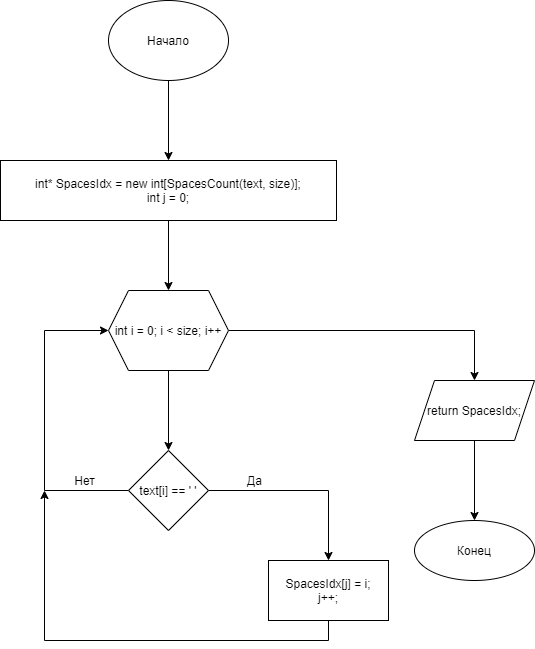
CheckWord



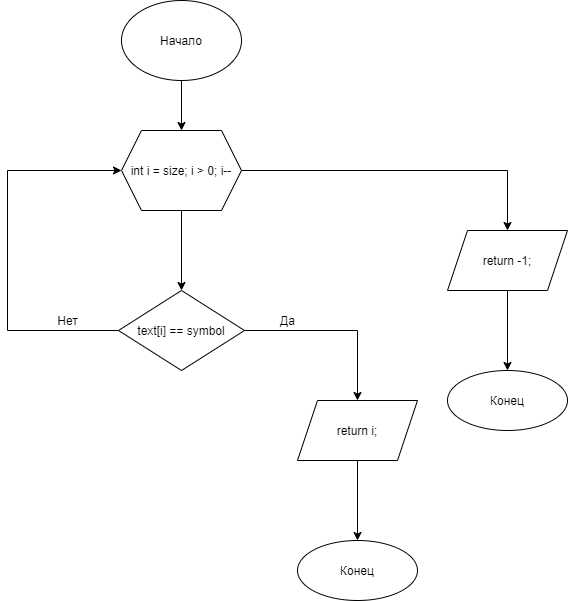
CompareWord



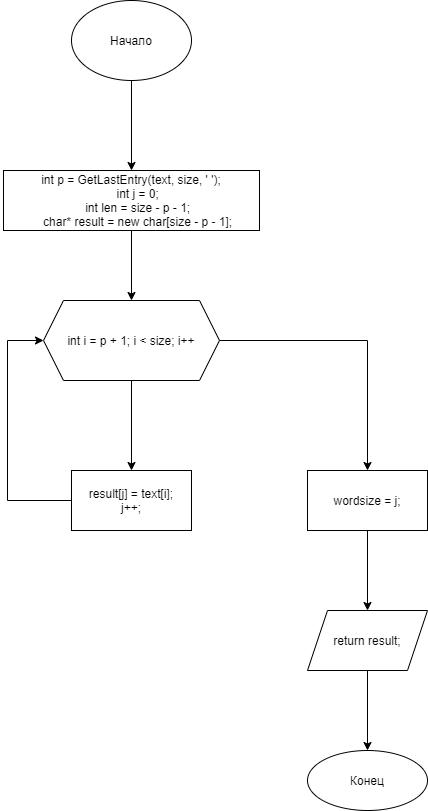
FindSpaces



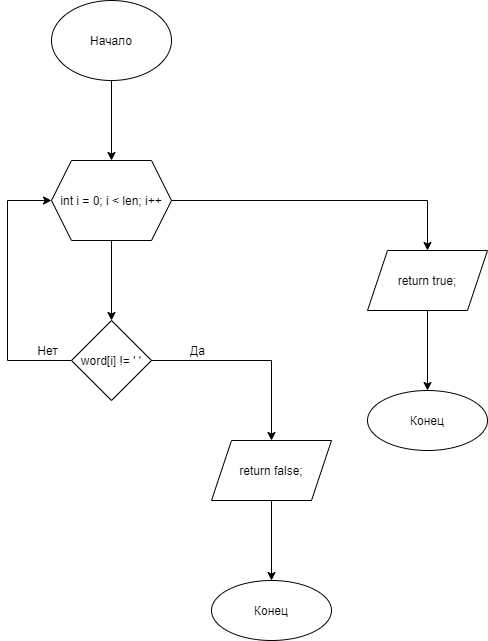
GetLastEntry



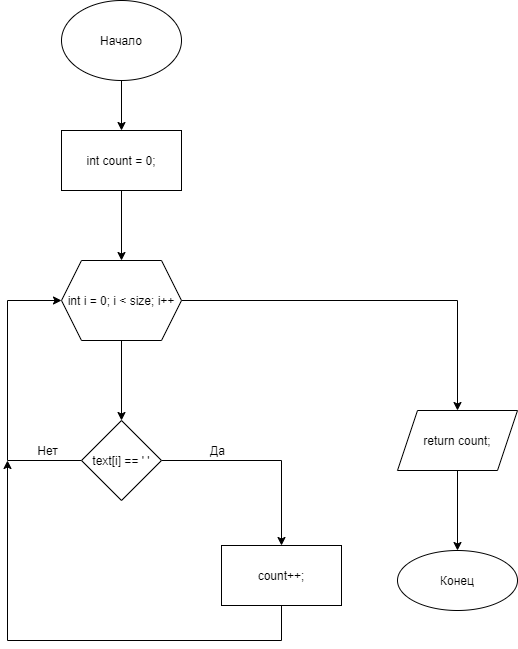
GetLastWord



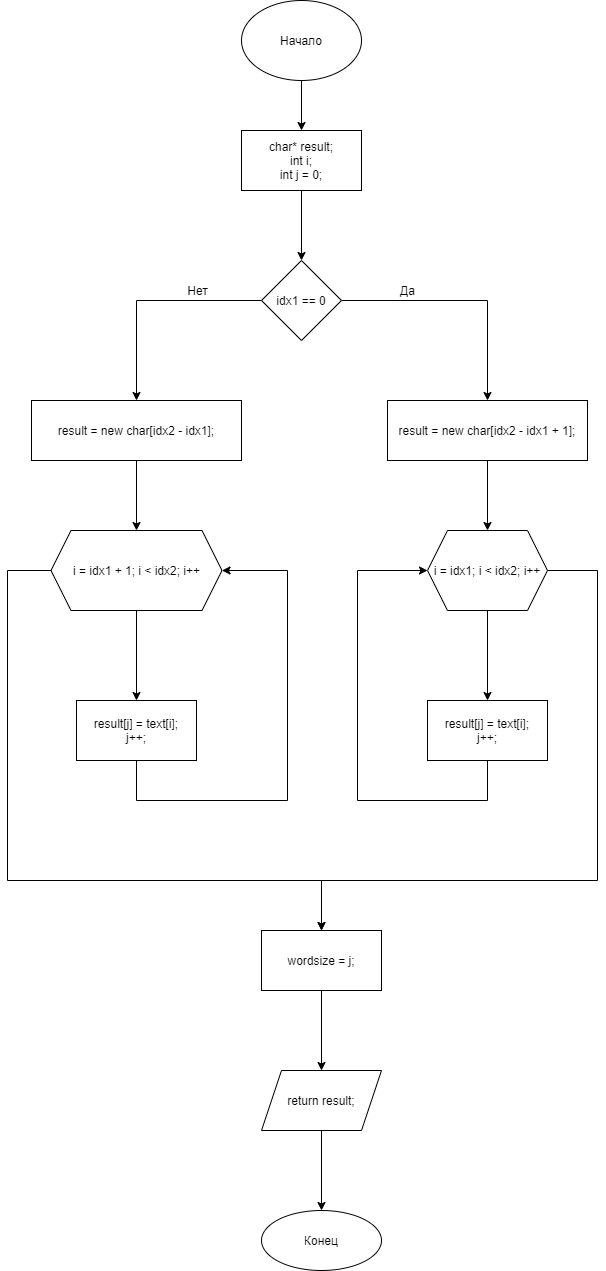
IsOnlyContainSpaces



SpacesCount



Trim



Текст программы

#include <iostream>

#include <cstdlib> // rand(), srand()

#include <fstream> // ifstream

#define \_CRT\_SECURE\_NO\_WARNINGS

**using** **namespace** std**;**

// Печает текст на экран

void PrintText**(**char text**[],** int size**)**

**{**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

cout **<<** text**[**i**];**

cout **<<** endl**;**

**}**

// Генерирует текст

char**\*** GenerateText**(**int size**)**

**{**

char**\*** text **=** **new** char**[**size**];**

const char alphanum1**[]** **=** "abcdefghijklmnopqrstuvwxyz"**;**

const char alphanum2**[]** **=** "abcdefghijklmnopqrstuvwxyz "**;**

// Первый символ не должен быть пробелом

text**[**0**]** **=** alphanum1**[**rand**()** **%** **(sizeof(**alphanum1**)** **-** 1**)];**

**for** **(**int i **=** 1**;** i **<** size **-** 1**;** i**++)**

text**[**i**]** **=** alphanum2**[**rand**()** **%** **(sizeof(**alphanum2**)** **-** 1**)];**

// Последний символ не должен быть пробелом

text**[**size **-** 1**]** **=** alphanum1**[**rand**()** **%** **(sizeof(**alphanum1**)** **-** 1**)];**

**return** text**;**

**}**

// Вводим текст с клавиатуры

char**\*** EnterText**(**int size**)**

**{**

char**\*** text **=** **new** char**[**size**];**

cout **<<** "Ввведите текст: "**;**

cin**.**getline**(**text**,** size **+** 1**);**

**return** text**;**

**}**

// Считывает текст из файла

char**\*** ReadFile**(**const char**\*** title**,** int size**)**

**{**

char**\*** s **=** **new** char**[**size**];**

char ch**;**

ifstream fin**;**

fin**.**open**(**title**);**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

**{**

fin**.**get**(**ch**);**

s**[**i**]** **=** ch**;**

**}**

fin**.**close**();**

**return** s**;**

**}**

// Возвращает индекс последнего вхождения символа в текст

int GetLastEntry**(**char**\*** text**,** int size**,** char symbol**)**

**{**

**for** **(**int i **=** size**;** i **>** 0**;** i**--)**

**if** **(**text**[**i**]** **==** symbol**)**

**return** i**;**

**return** **-**1**;**

**}**

// Возвращает последнее слово в тексте

char**\*** GetLastWord**(**char**\*** text**,** int size**,** int**&** wordsize**)**

**{**

int p **=** GetLastEntry**(**text**,** size**,** ' '**);**

int j **=** 0**;**

int len **=** size **-** p **-** 1**;**

char**\*** result **=** **new** char**[**size **-** p **-** 1**];**

**for** **(**int i **=** p **+** 1**;** i **<** size**;** i**++)**

**{**

result**[**j**]** **=** text**[**i**];**

j**++;**

**}**

wordsize **=** j**;**

**return** result**;**

**}**

// Возвращает количество пробелов в тексте

int SpacesCount**(**char**\*** text**,** int size**)**

**{**

int count **=** 0**;**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

**{**

**if** **(**text**[**i**]** **==** ' '**)**

count**++;**

**}**

**return** count**;**

**}**

// Возвращает массив с индексами пробелов

int**\*** FindSpaces**(**char**\*** text**,** int size**)**

**{**

int**\*** SpacesIdx **=** **new** int**[**SpacesCount**(**text**,** size**)];** // Массив с индексами пробелов

int j **=** 0**;**

**for** **(**int i **=** 0**;** i **<** size**;** i**++)**

**{**

**if** **(**text**[**i**]** **==** ' '**)**

**{**

SpacesIdx**[**j**]** **=** i**;**

j**++;**

**}**

**}**

**return** SpacesIdx**;**

**}**

// Возвращает true - если слово целиком состоит из пробелов, в обратном случае возвращает false

bool IsOnlyContainSpaces**(**char**\*** word**,** int len**)**

**{**

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

**if** **(**word**[**i**]** **!=** ' '**)**

**return** **false;**

**return** **true;**

**}**

// Обрезает текст по заданным границам

char**\*** Trim**(**char**\*** text**,** int idx1**,** int idx2**,** int**&** wordsize**)**

**{**

char**\*** result**;**

int i**;**

int j **=** 0**;**

**if** **(**idx1 **==** 0**)**

**{**

result **=** **new** char**[**idx2 **-** idx1 **+** 1**];**

**for** **(**i **=** idx1**;** i **<** idx2**;** i**++)**

**{**

result**[**j**]** **=** text**[**i**];**

j**++;**

**}**

**}**

**else** **{**

result **=** **new** char**[**idx2 **-** idx1**];**

**for** **(**i **=** idx1 **+** 1**;** i **<** idx2**;** i**++)**

**{**

result**[**j**]** **=** text**[**i**];**

j**++;**

**}**

**}**

wordsize **=** j**;**

**return** result**;**

**}**

// Сравненивает два слова

bool CompareWord**(**char**\*** word1**,** int size1**,** char**\*** word2**,** int size2**)**

**{**

**if** **(**size1 **!=** size2**)** **return** **false;**

**for** **(**int i **=** 0**;** i **<** size1**;** i**++)**

**if** **(**word1**[**i**]** **!=** word2**[**i**])** **return** **false;**

**return** **true;**

**}**

/\*

Проверяем, совпадает ли слово с конечным отрезком из алфавита(z, yz, xyz, ...)

Для этого проходимся по каждому слову (z, yz, xyz, ...) и проверяем, если

оба слова совпадают, то возвращает true

\*/

bool CheckWord**(**char**\*** word**,** int size**)**

**{**

char**\*** s**;**

int l **=** 0**;**

int len**;**

**for** **(**char i **=** 'a'**;** i **<=** 'z'**;** i**++)**

**{**

len **=** abs**(**122 **-** int**(**i**)** **+** 1**);**

s **=** **new** char**[**len **+** 1**];**

l **=** 0**;**

**for** **(**char j **=** i**;** j **<=** 'z'**;** j**++)**

**{**

s**[**l**]** **=** j**;**

l**++;**

**}**

**if** **(**CompareWord**(**word**,** size**,** s**,** len**))** **return** **true;**

**}**

**return** **false;**

**}**

int main**()**

**{**

srand**((**unsigned int**)**time**(NULL));**

// Исправляем русский язык в консоли

setlocale**(**LC\_ALL**,** "rus"**);**

// Количество символов в тексте

int size **=** 80**;**

char **\***text **=** ReadFile**(**"text.txt"**,** size**);** // Считываем текст из файла

// char\* text = GenerateText(size); // Генерируем текст

// char\* text = new char[size];

// cin.getline(text, 80); // Вводим текст из консоли

PrintText**(**text**,** size**);** // Печатаем текст

// Длина последнего слова

int lastwordlen**;**

// Последнее слово

char**\*** lastword **=** GetLastWord**(**text**,** size**,** lastwordlen**);**

// Количество пробелов в тексте

int spacescount **=** SpacesCount**(**text**,** size**);**

// Массив из индексов пробелов

int**\*** SpacesIdx **=** FindSpaces**(**text**,** size**);**

int lastidx**;**

int nextidx**;**

char**\*** word**;**

int wordlen**;**

bool flag**;**

**for** **(**int k **=** 0**;** k **<** spacescount **-** 1**;** k**++)**

**{**

lastidx **=** SpacesIdx**[**k**];**

nextidx **=** SpacesIdx**[**k **+** 1**];**

word **=** Trim**(**text**,** lastidx**,** nextidx**,** wordlen**);**

// Проходимся по каждому слову в тексте, кроме последнего

// Если слово состоит только из проблелов - пропускаем

bool flag1 **=** IsOnlyContainSpaces**(**word**,** wordlen**);**

// Если слово совпадает с последним словом - пропускаем

bool flag2 **=** CompareWord**(**lastword**,** lastwordlen**,** word**,** wordlen**);**

**if** **(**flag1 **||** flag2**)** **continue;**

// Если слово удовлетворяет всем условиям - печатаем

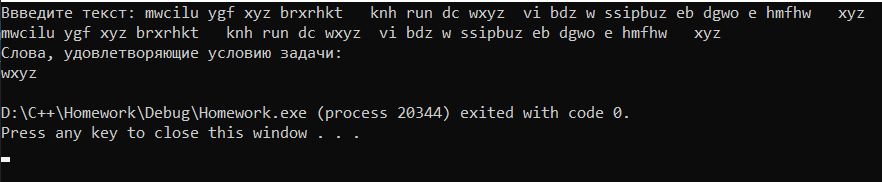
**if** **(**CheckWord**(**word**,** wordlen**))** PrintText**(**word**,** wordlen**);**

**}**

**return** 0**;**

**}**

Анализ результатов



Слово ‘xyz’ не подходит, так как оно совпадает с последним словом – ‘xyz’

Слово ‘wxyz’ подходит, так как оно

* Не совпадает последним словом ‘xyz’
* Совпадает с конечным отрезком из алфавита (z, yz, xyz и т.д.)

Вывод

Я научился

* Работать с динамическими массивами
* Обрабатывать динамические массивы
* Выводить динамические массивы на экран
* Работать с текстовыми файлами