Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Основы программирования»

Отчет по домашнему заданию
Вариант 14

Выполнил:		Проверил:
Студент группы ИУ5-15Б		преподаватель каф. ИУ5
Расулов А. Н.		Аксенова М. В.
Подпись и дата:		Подпись и дата:

Москва, 2023 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;

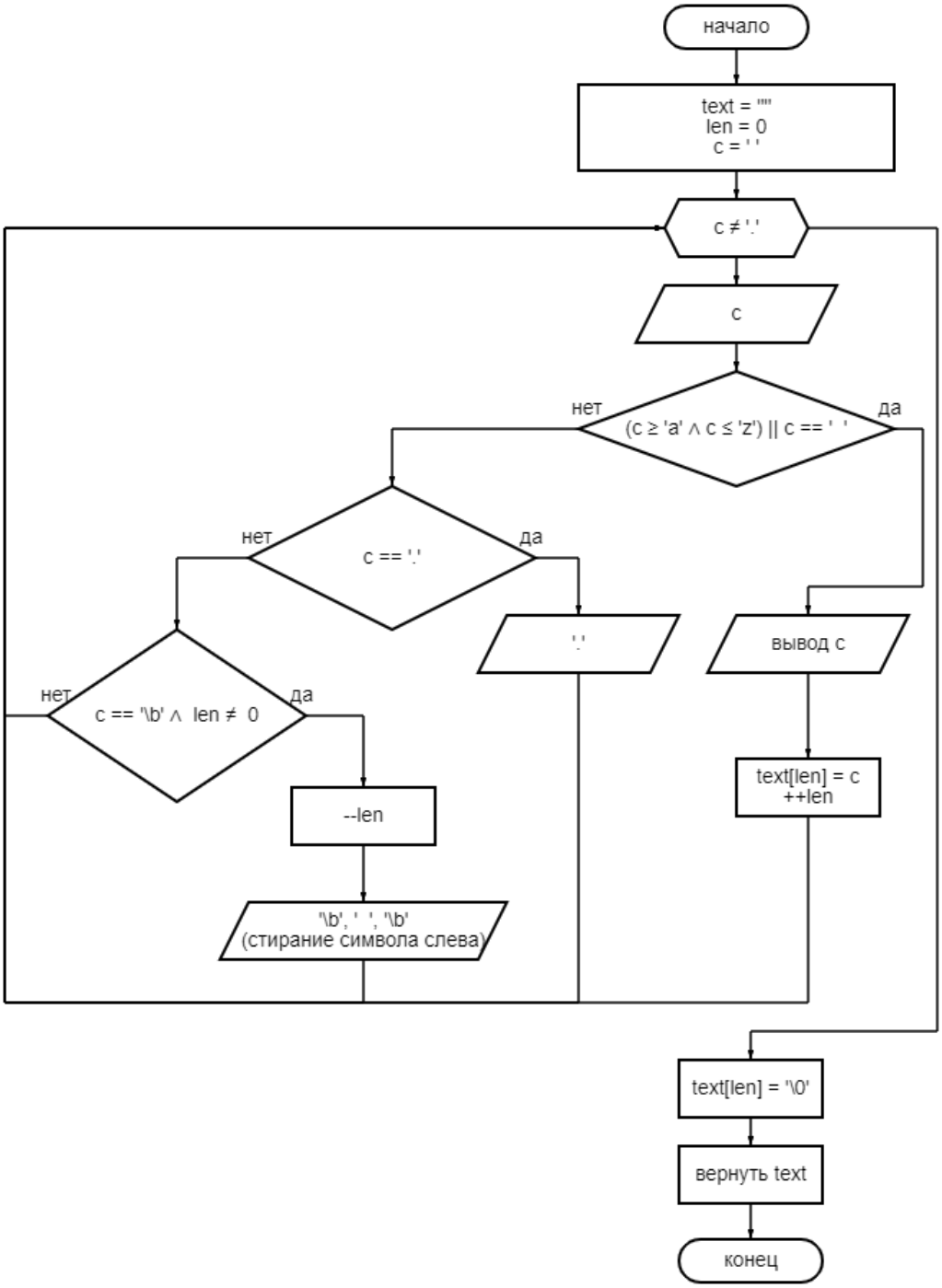
б) напечатать все слова, отличающиеся от последнего слова, и в которые каждая буква входит не менее двух раз.

Разработка алгоритма

- `bool strEq(char* str1, char* str2)` – функция для сравнения двух строк str1 и str2. Если они равны, то она возвращает true, иначе false.
  Внутрии функции определены:
  - int i хранит в себе текущий индекс сравниваемых символов str1 и str2
- `char* getLastWord(char* str)` – функция для получения последнего слова из строки str. Внутри функции определены:
  - int a, b – соответственно левая и правая границы (индексы) последнего слова.
  - int sublen – длина подстроки (последнего слова)
  - char* res – переменная, которая хранит в себе последнее слово
- `bool eachLetterAppearsAtLeastTwice(char* str)` – функция, возвращающая true, если каждая буква строки str входит в нее не менее двух раз (пункт б задания); иначе false.
  Внутрии функции определены:
  - int i – итератор
  - int stats[256] – массив, хранящий в себе статистику используемых символов в строке. То есть ключ — это символ, а значение — его количество в строке str.
- `void strPrint(char* str)` – функция, выводящая строку str в консоль.
- `void strCpy(char* dest, char* src, int len)` – функция, копирующая строку src размером len в строку dest
- `int strLen(char* str)` – функция, возвращающая длину строки str. Внутрии функции определена переменная int i – длина строки str
- `char* getInput()` -- функция для получения ввода от пользователя, который удовлетворяет пункту а) задачи. Возвращает строку, которую ввел пользователь, которая удовлетворяет условию задачи.
  Внутри функции определены:
  - `char* text` – строка, которую ввел пользователь
  - `int len` -- ее длина
  - `char c` — текущий символ, вводимый пользователем
- `int main()`:
  - `char* text` – текст, введенный пользователем, который удовлетворяет условию задачи
  - `char* lastWord` -- последнее слово текста text
  - `char currWord[BUFF_SIZE]` – переменная для хранения одного слова из текста text
  - `bool isFirstWord` - перемененная отвечающая за то, выводится ли первое слово в консоль или нет (от этой переменной зависит, выводить ли пробел перед словом или нет)
  - `int currWordLen` – переменная для хранения длины слова currWord

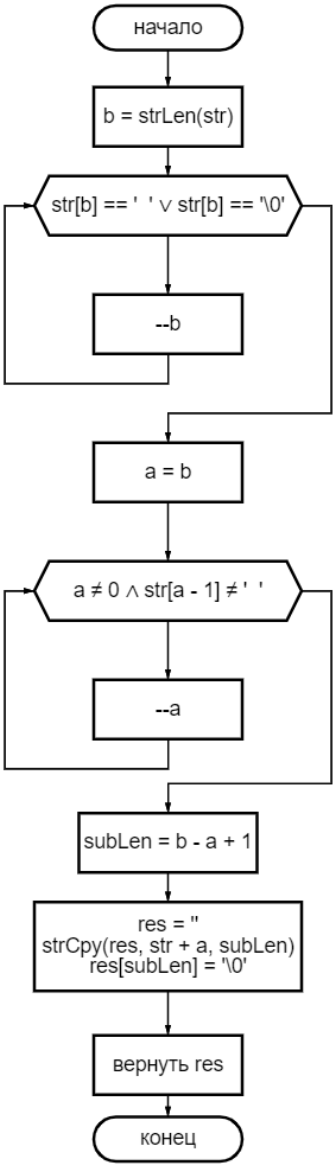
Блок-схема алгоритма

Блок-схема функции char* getInput():

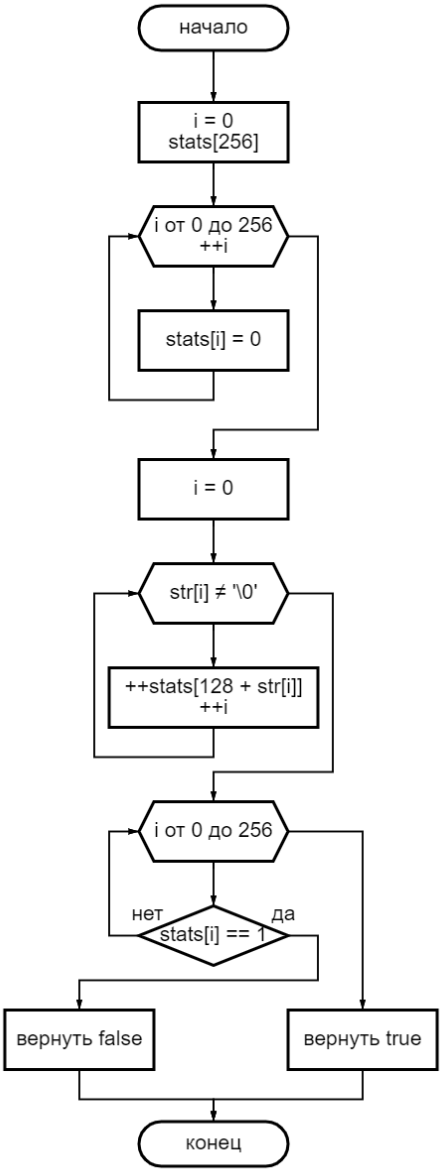


Блок-схема функций:

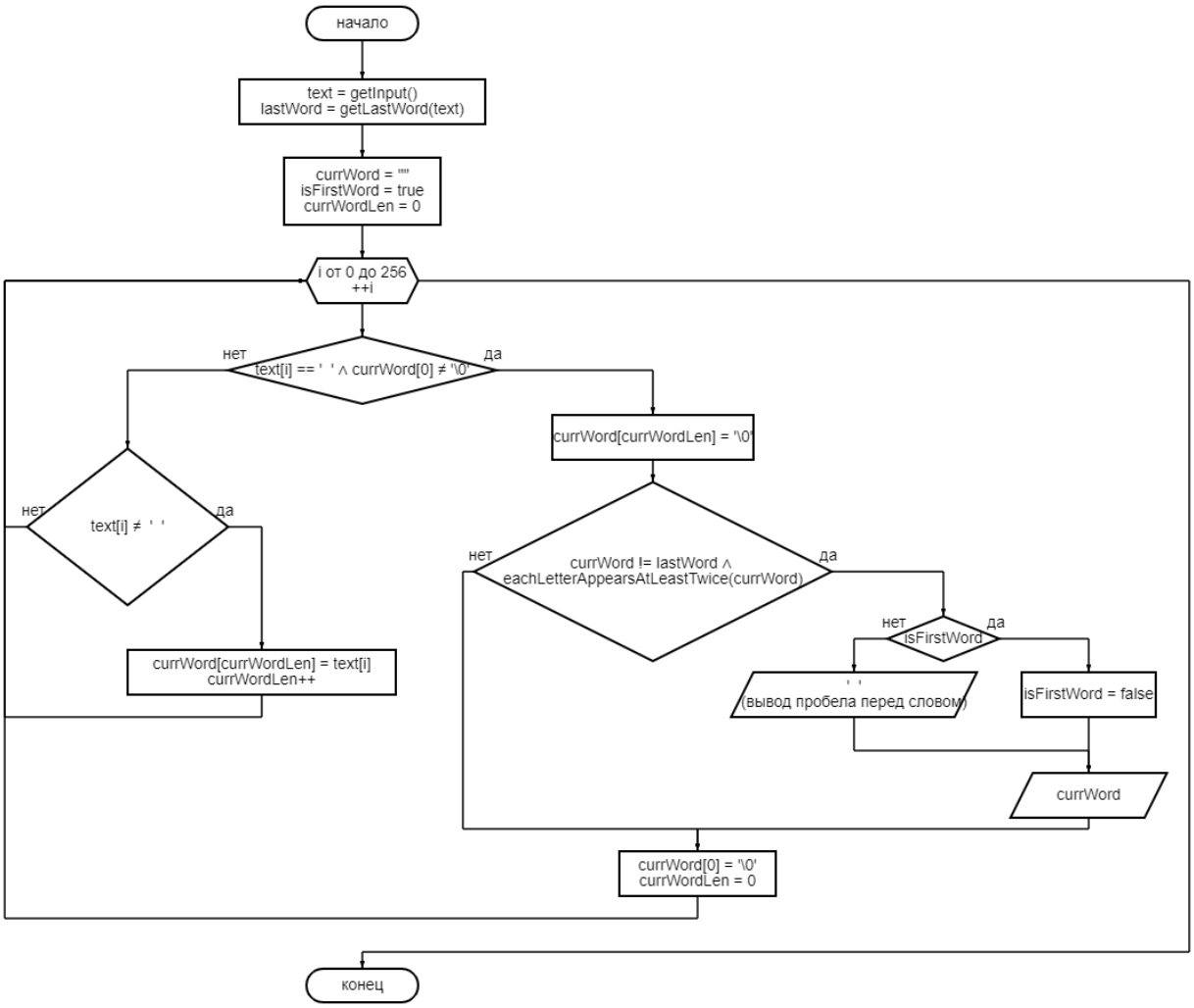
`char* getLastWord(char* str):`



`bool eachLetterAppearsAtLeastTwice(char* str):`



Блок-схема функции int main():



Текст программы

header.h

```
#include <conio.h>
using namespace std;

bool strEq(char* str1, char* str2);
char* getLastWord(char* str);
bool eachLetterAppearsAtLeastTwice(char* str);
void strPrint(char* str);
void strCpy(char* dest, char* src, int len);
int strLen(char* str);

char* getInput();

#define BUFF_SIZE 256
```

source.cpp

```
#include "header.h"

char* getInput() {
	char* text = new char[BUFF_SIZE];
	int len = 0;

	char c = ' ';
	while (c != '.')
	{
		c = _getch();

		if ((c >= 'a' && c <= 'z') || c == ' ')
		{
```

```
                        _putch(c);
                        text[len] = c;
                        ++len;
                }
                else if (c == '.')
                        _putch('.');
                else if (c == '\b' && len != 0)
                {
                        --len;
                        _putch('\b');
                        _putch(' ');
                        _putch('\b');
                }
        }
        _putch('\n');
        text[len] = '\0';
        return text;
}
bool strEq(char* str1, char* str2) {
        int i = 0;
        while (str1[i] != '\0' && str2[i] != '\0') {
                if (str1[i] != str2[i])
                        return false;
                ++i;
        }
        if (str1[i] == '\0' && str2[i] == '\0')
                return true;
        return false;
}
void strCpy(char* dest, char* src, int len) {
        for (int i = 0; i < len; ++i)
                dest[i] = src[i];
}
int strLen(char* str) {
        int i = 0;
        while (str[i] != '\0')
                ++i;
        return i;
}
char* getLastWord(char* str) {
        int b = strLen(str);
        while (str[b] == ' ' || str[b] == '\0') {
                --b;
        }
        int a = b;
        while (a != 0 && str[a - 1] != ' ') {
                --a;
        }

        int subLen = b - a + 1;

        char* res = new char[subLen + 1];
        strCpy(res, str + a, subLen);
        res[subLen] = '\0';
        return res;
}
bool eachLetterAppearsAtLeastTwice(char* str) {
        int i = 0, stats[256];
        for (; i < 256; ++i)
                stats[i] = 0;
        i = 0;
        while (str[i] != '\0') {
                ++stats[128 + str[i]];  // Сам символ может иметь отрицательное значение. Поэтому мы
сдвигаем его значение на 128, чтобы сделать его в пределах [0; 255]
                ++i;
        }
        for (i = 0; i < 256; ++i)
                if (stats[i] == 1)
                        return false;
        return true;
}
```

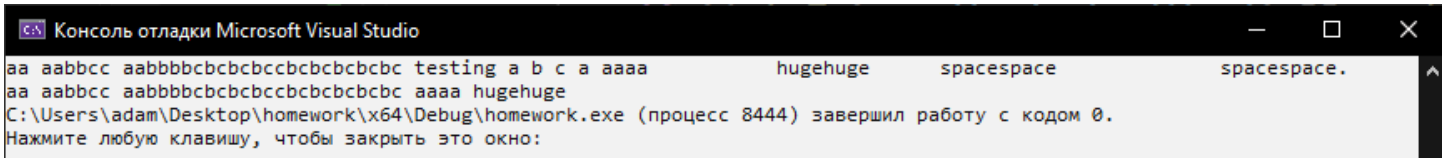
```
void strPrint(char* str) {
	for (int i = 0; str[i] != '\0'; ++i)
		_putch(str[i]);
}
```

main.cpp

```
#include "header.h"
int main()
{
	char* text = getInput();
	char* lastWord = getLastWord(text);

	char currWord[BUFF_SIZE] = "";
	bool isFirstWord = true;
	int currWordLen = 0;
	for (int i = 0; i < BUFF_SIZE; ++i) {
		if (text[i] == ' ' && currWord[0] != '\0')
		{
			currWord[currWordLen] = '\0';
			if (!strEq(currWord, lastWord) && eachLetterAppearsAtLeastTwice(currWord))
			{
				if (isFirstWord)
					isFirstWord = false;
				else
					_putch(' ');
				strPrint(currWord);
			}
			// Reset curr word
			currWord[0] = '\0';
			currWordLen = 0;
		}
		else if (text[i] != ' ')
			currWord[currWordLen++] = text[i];
	}
	// Don't forget to free memory
	delete[] text;
	delete[] lastWord;
}
```

Анализ результатов




```
.

C:\Users\adam\Desktop\homework\x64\Debug\homework.exe (процесс 8148) завершил работу с кодом 0.
```

```
hfusduhfiifh.

C:\Users\adam\Desktop\homework\x64\Debug\homework.exe (процесс 1764) завершил работу с кодом 0.
```

```
                aaaa lol                    lolo        f aaaa.
lolo
C:\Users\adam\Desktop\homework\x64\Debug\homework.exe (процесс 5368) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно:
```

```
                    .

C:\Users\adam\Desktop\homework\x64\Debug\homework.exe (процесс 9436) завершил работу с кодом 0.
```

```
esse iuiu test                          esse.
iuiu
C:\Users\adam\Desktop\homework\x64\Debug\homework.exe (процесс 12348) завершил работу с кодом 0.
```