

**Московский государственный технический
университет им. Н. Э. Баумана**

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

Курс «Основы программирования»

Отчет по Домашней работе

Выполнил:

Студент группы ИУ5-11Б

Алехин Сергей

Подпись и дата:

Проверил:

Преподаватель каф. ИУ5

Правдина Анна Дмитриевна

Подпись и дата:

Москва, 2018 г.

Задание

Целью домашнего задания является закрепление приобретенных в процессе выполнения лабораторных работ практических навыков реализации на языке C++ следующих приемов структурного программирования:

1. Использование методов структурного проектирования программ при разработке алгоритма выполнения задания.
2. Использование структур и массивов из структур для хранения данных.
3. Использование методов структурного программирования при разработке библиотеки базовых функций для реализации отдельных блоков алгоритма.
4. Организация обмена данными между функциями с использованием указателей и ссылок.
5. Создание и обработка динамических массивов.
6. Отладка и тестирование программ.

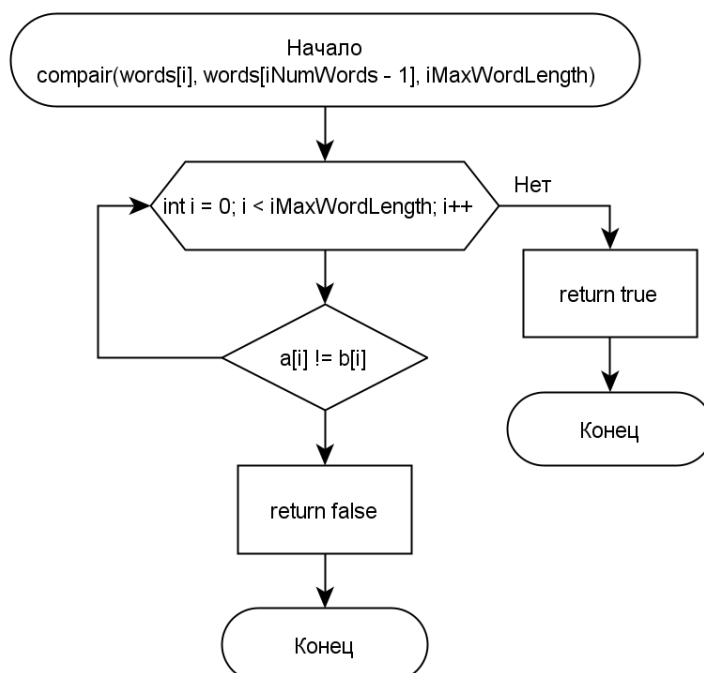
Разработка алгоритма

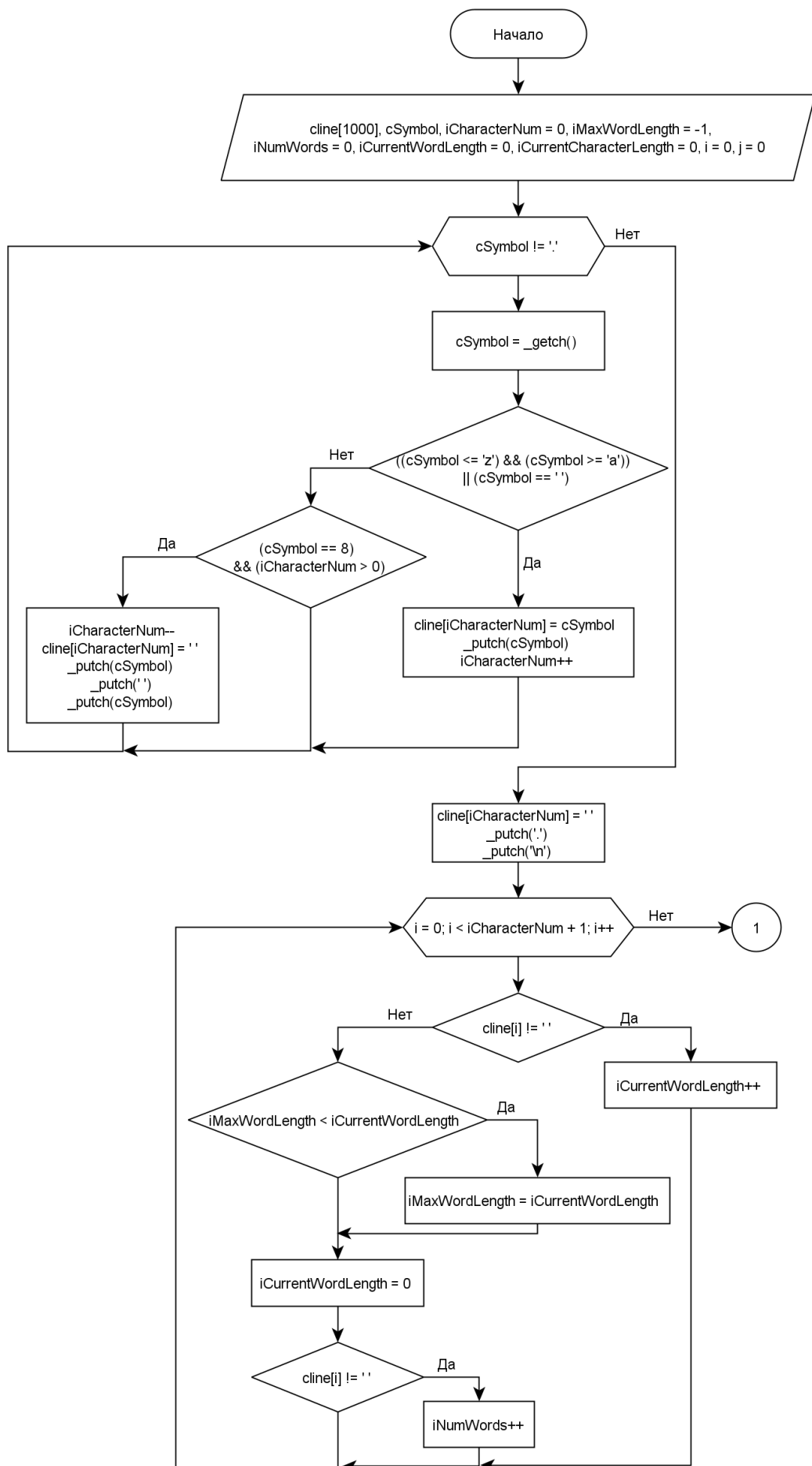
Входные переменные:

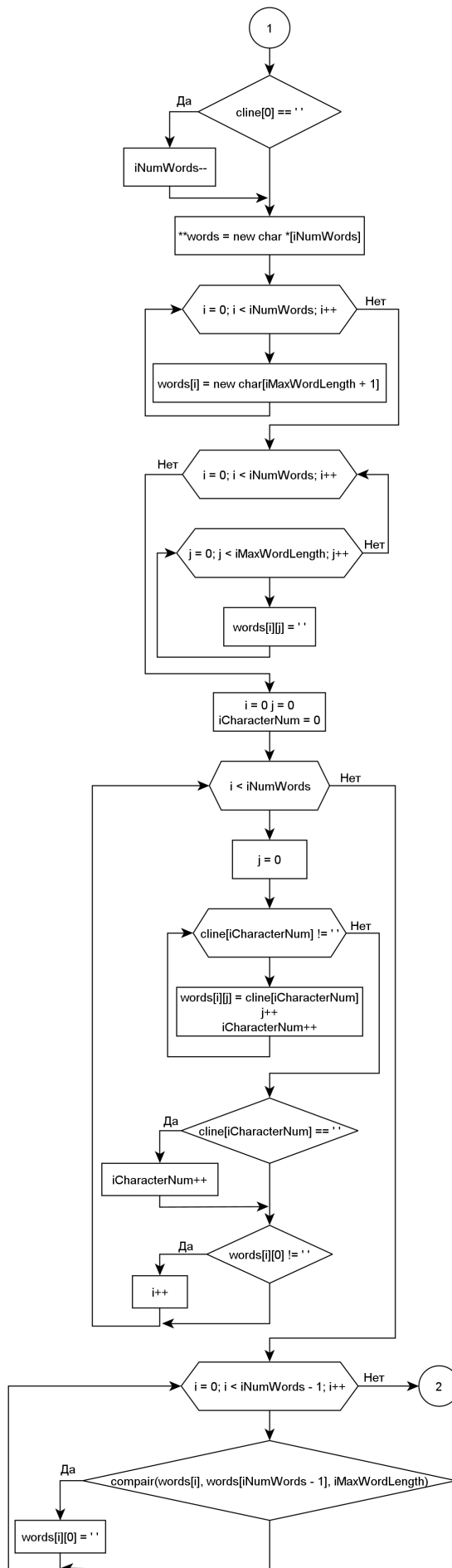
- 1) `char cline[]` – одномерный массив(входная строка);
- 2) `char cSymbol` – считываемый символ;
- 3) `int iCharacterNum` – номер считываемого символа;
- 4) `int iMaxWordLength` – максимальная длина слова;
- 5) `int iNumWords` – количество слов;
- 6) `int iCurrentWordLength` – длина считываемого символа;
- 7) `int iCurrentCharacterLength` – длина текущего слова;

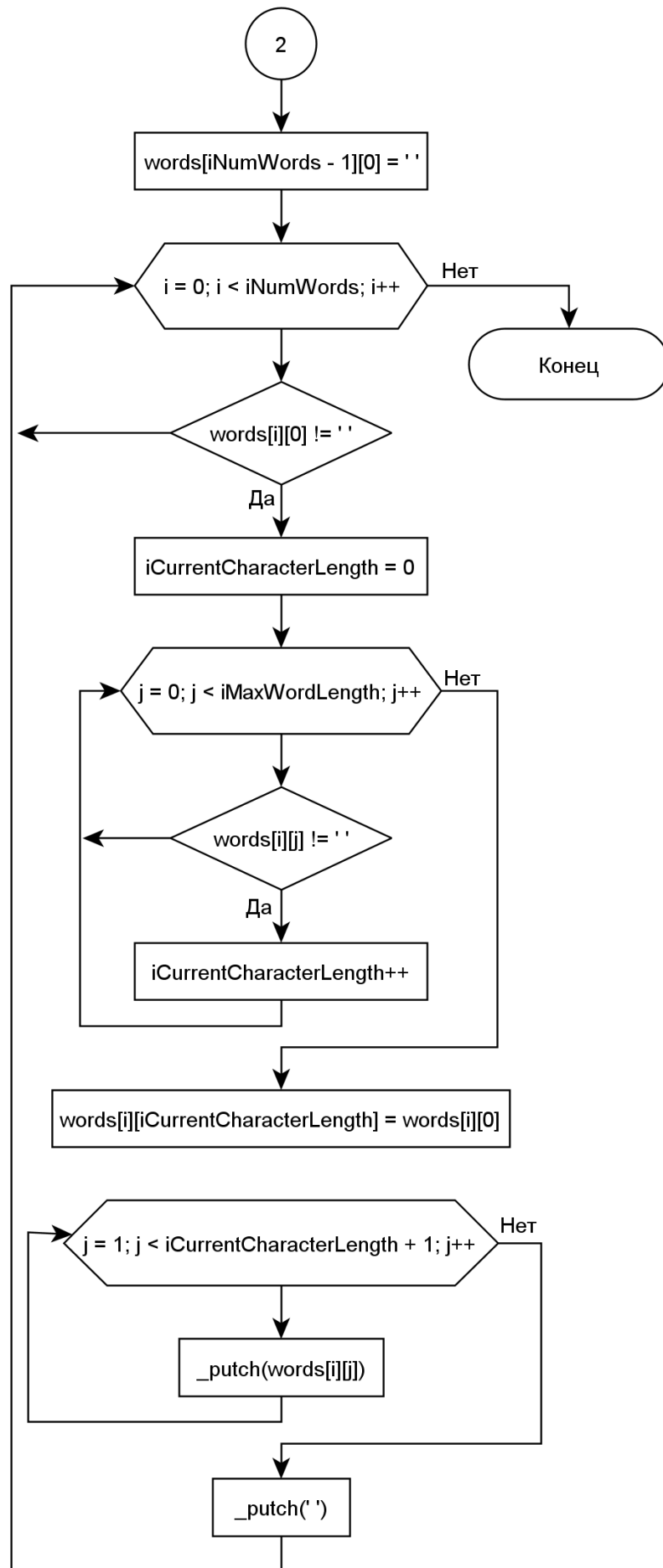
Функции:

- 1) `bool compair` – сравнение слов;
 - a. *Входные переменные:*
 - i. `char *a` – первое слово;
 - ii. `char *b` – второе слово;
 - iii. `int iMaxWordLength` – максимальная длина слова.
 - b. *Возвращаемое значение:*
 - i. `bool fasle` или `bool true` – одинаковые слова или нет;









Текст программы

Файл Hw.cpp

```
#include <conio.h>
#include "str.h"
using namespace std;
int main()
{
    char cline[1000], cSymbol;
    int iCharacterNum = 0, iMaxWordLength = -1, iNumWords = 0, iCurrentWordLength = 0,
    iCurrentCharacter = 0, iCurrentCharacterLength = 0, i = 0, j = 0;
    do
    {
        cSymbol = _getch();
        if (((cSymbol <= 'z') && (cSymbol >= 'a')) || (cSymbol == ' '))
        {
            cline[iCharacterNum] = cSymbol;
            _putch(cSymbol);
            iCharacterNum++;
        }
        else
        {
            if ((cSymbol == 8) && (iCharacterNum > 0))
            {
                iCharacterNum--;
                cline[iCharacterNum] = ' ';
                _putch(cSymbol);
                _putch(' ');
                _putch(cSymbol);
            }
        }
    } while (cSymbol != '.');
    cline[iCharacterNum] = ' ';
    _putch('.');
    _putch('\n');
    for (i = 0; i < iCharacterNum + 1; i++)
    {
        if (cline[i] != ' ') iCurrentWordLength++;
        else
        {
            if (iMaxWordLength < iCurrentWordLength) iMaxWordLength =
iCurrentWordLength;
            iCurrentWordLength = 0;
            if (cline[i - 1] != ' ') iNumWords++;
        }
    }
    if (cline[0] == ' ') iNumWords--;
    char **words = new char *[iNumWords];
    for (i = 0; i < iNumWords; i++)
        words[i] = new char[iMaxWordLength + 1];
    for (i = 0; i < iNumWords; i++)
        for (j = 0; j < iMaxWordLength; j++)
            words[i][j] = ' ';
    i = 0; j = 0;
    iCharacterNum = 0;
    while (i < iNumWords)
    {
        j = 0;
        while (cline[iCharacterNum] != ' ')
```

```

        {
            words[i][j] = cline[iCharacterNum];
            j++;
            iCharacterNum++;
        }
        if (cline[iCurrentCharacter] == ' ') iCharacterNum++;
        if (words[i][0] != ' ') i++;
    }
    for (i = 0; i < iNumWords - 1; i++)
        if (compair(words[i], words[iNumWords - 1], iMaxWordLength))
            words[i][0] = ' ';
    words[iNumWords - 1][0] = ' ';
    for (i = 0; i < iNumWords; i++)
    {
        if (words[i][0] != ' ')
        {
            iCurrentCharacterLength = 0;
            for (j = 0; j < iMaxWordLength; j++)
                if (words[i][j] != ' ') iCurrentCharacterLength++;
            words[i][iCurrentCharacterLength] = words[i][0];
            for (j = 1; j < iCurrentCharacterLength + 1; j++)
                _putch(words[i][j]);
            _putch(' ');
        }
    }
    _getch();
    return 0;
}

```

Файл *str.h*

```

#pragma once
bool compair(char *, char *, int);

```

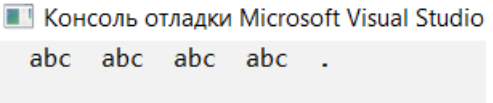
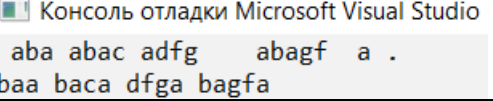
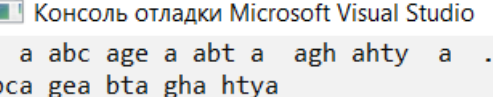
Файл *func.cpp*

```

bool compair(char *a, char *b, int iMaxWordLength)
{
    for (int i = 0; i < iMaxWordLength; i++)
        if (a[i] != b[i]) return false;
    return true;
}

```

Анализ результатов

№	Входные данные	Полученный результат
1	Все слова похожи на последнее	
2	Нет слов похожих на последнее	
3	Есть слова похожие на последнее, и не похожие на последнее.	

Вывод: Мы научились обрабатывать и выводить на экран только нужный нам текст с помощью `getch` и `putch`, после этого создавать двумерный динамический массив, обрабатывать слова, а после выводить их на экран.