**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**  
**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №4

Дисциплина: Обьектно-ориентированное программирование

по теме «Классы»

Выполнил: ст. группы ВТ-22  
Фаракшин Никита Русланович

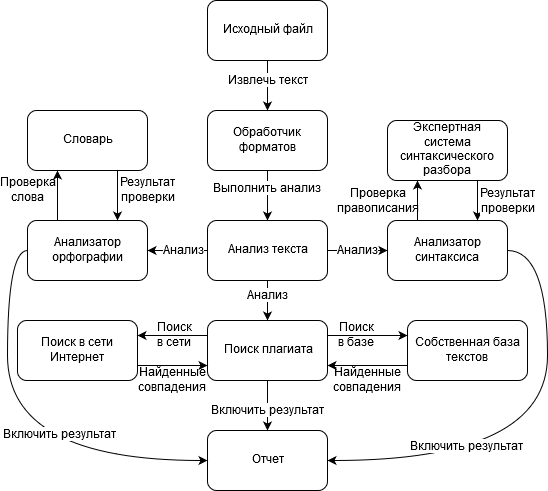
Проверил: Буханов Д.Г.

**Белгород 2019Цель работы**: приобретение практических навыков создания класса на языке C++.

|  |  |  |
| --- | --- | --- |
| **№** | **Задание 1** | **Задание 2** |
| 8 | Выполнить построение объектной модели следующей предметной области:  “анализатор текста, нахождение плагиата” | Создать класс для работы сo строками.  Разработать следующие элементы класса:  а. Поля:  char \*str;  int n.  б. Конструктор, позволяющий создать строку из символов.  в. Методы, позволяющие:  подсчитать количество пробелов в строке;  заменить в строке все прописные символы на строчные;  удалить из строки все знаки препинания.  г. Перегрузить (переопределить):  деление  операция “логический сдвиг вправо” |

**Задание 1.**

Анализатор текста включает в себя: проверку текста на орфографические и синтаксические ошибки, анализ семантики (ключевые слова, «водянистость» текста, и т.д.) и поиск плагиата как по внутренней базе документов, так и в сети интернет. Пользователь может загрузить файл в различных форматах, из которых система извлекает текст. По окончании проверки программа формирует для пользователя отчет о результатах.

**Задание 2.**

#include <iostream>

using namespace std;

class String{

    private:

    char \*str;

    int n;

friend const String operator/(const String& s1, const String& s2);

friend const String operator>>(const String& s1, const int& count);

    int is\_upper(char c){

        return (c >= 'A' && c <= 'Z' || c >= 'А' && c <= 'Я');

    }

    int is\_punctuation(char c){

*// Знаками препинания считаем символы ASCII с 65 по 79 и с 90 по 96*

        return (c >= 65 && c <= 79 || c >= 90 && c <= 96);

    }

    char to\_lower(char c){

        return c - ('a' - 'A');

    }

    void resize(int length){

        char \*new\_addr = new char [length];

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

            new\_addr[i] = str[i];

        }

        n = length;

        delete[] str;

        str = new\_addr;

    }

    public:

    String(char \*chars, int count){

        str = new char[count];

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

            str[i] = chars[i];

        }

        n = count;

    }

void print(){

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

cout << str[i];

}

cout << endl;

}

    int count\_spaces(){

        int count = 0;

        for (int i = 0; i<n; i++){

            if (str[i] == ' '){

                count++;

            }

        }

        return count;

    }

    void lower(){

        for (int i = 0; i<n; i++){

            if (is\_upper(str[i])){

                str[i] = to\_lower(str[i]);

            }

        }

    }

    void remove\_punctuations(){

int last = 0;

        for (int i = 0; i<n; i++){

            if (!is\_punctuation(str[i])){

                if (i != last){

                    str[last] = str[i];

                }

                last++;

            }

        }

        resize(last);

    }

};

const String operator/(const String& s1, const String& s2){

*/\* Оператор деления возвращает часть строки, находящуюся до первого вхождения s2.*

*Если s2 не входит в s1, то результат - вся строка s1 \*/*

    int i = 0, occurence = 0;

    int n1 = s1.n;

    int n2 = s2.n;

    char \*str1 = s1.str;

    char \*str2 = s2.str;

*// За границей n1-n2 не может быть вхождения строки s2, имеющей длину n2*

    while (i <= n1-n2 && !occurence){

*// Для каждого символа строки s1 пытаемся определить, является ли*

*//подстрока s1[i..i+n2-1] строкой s2*

        int j = 0;

        while (j < n2 && str1[i+j] == str2[j]){

            j++;

        }

        if (j == n2){

            occurence = 1;

        }

        i++;

    }

int new\_len;

    if (occurence){

        new\_len = i-1;

    }else{

        new\_len = n1;

    }

    String s3(str1, new\_len);

    return s3;

}

const String operator>>(const String& s1, const int& count){

*/\* Оператор возвращает строку, дополненную count пробелами слева \*/*

int new\_len = s1.n + count;

char \*new\_str = new char[new\_len];

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

new\_str[i] = ' ';

}

for (int j = 0; j<s1.n; j++){

new\_str[j+count] = s1.str[j];

}

    String s3(new\_str, new\_len);

    return s3;

}

int main() {

    return 0;

}