**МИНОБРНАУКИ 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 ПВ-21  
Ковалев Павел

Проверил: Буханов Д.Г.

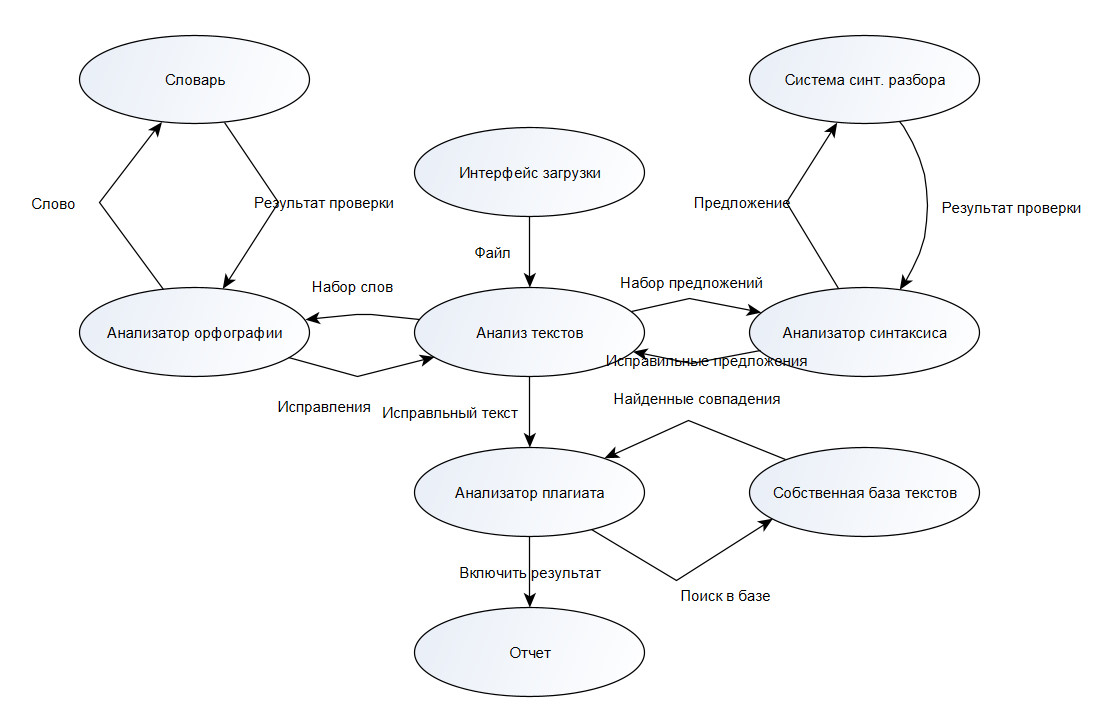
Белгород 2020

Вариант 8

**Цель работы:** приобретение практических навыков создания класса на языке С++.

**Задание**: Выполнить построение диаграммы объектов в соответствии с заданием варианта (табл. 1). По заданному описанию класса в соответствии с вариантом создать класс на языке С++ и переопределить указанные операции.

|  |  |  |
| --- | --- | --- |
| 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” |



<main.cpp>

1. #include <QCoreApplication>
2. #include <QTextCodec>
3. #include <QTextStream>
4. #include <iostream>
5. *using* *namespace* std;
6. *class* **String**
7. {
8. *private*:
9. char \*str;
10. int n;
11. int **isUpper**(char c){*//Возв* *True* *если* *A...Z* *false* *a...z*
    1. *return* (c >= 'A' && c <= 'Z');
12. }
13. int **isPunctuation**(char c){*//Проверка* *на* *знаки* *препинания*
    1. *//* *Знаками* *препинания* *считаем* *символы* *ASCII* *с* *65* *по* *79* *и* *с* *90* *по* *96*
    2. *return* ((c >= 65 && c <= 79) || (c >= 90 && c <= 96));
14. }
15. char **toLower**(char c){*//Возв* *строчного* *символа* *алфавита*
    1. *return* c - ('a' - 'A');
16. }
17. *//Увеличения* *или* *уменьшения* *строки*
18. void **resizeString**(int length){
    1. char\* newStr = *new* char [length];
    2. *for* (int i = 0; i<length; i++){
       * 1. newStr[i] = str[i];
       1. }
       2. n = length;
       3. *delete*[] str;
       4. str = newStr;
19. }
20. */\** *Оператор* *деления* *возвращает* *часть* *строки,* *находящуюся* *до* *первого* *вхождения* *s2.*
    1. *Если* *s2* *не* *входит* *в* *s1,* *то* *результат* *-* *вся* *строка* *s1* *\*/*
21. *friend* *const* String *operator* /(*const* String& s1, *const* String& s2){
    1. int i = 0, occurence = 0;
    2. int n1 = s1.n;
    3. int n2 = s2.n;
    4. char \*str1 = s1.str;
    5. char \*str2 = s2.str;
    6. *//* *За* *границей* *n1-n2* *не* *может* *быть* *вхождения* *строки* *s2,* *имеющей* *длину* *n2*
    7. *while* (i <= n1-n2 && !occurence){
       1. *//* *Для* *каждого* *символа* *строки* *s1* *пытаемся* *определить,* *является* *ли*
    8. *//подстрока* *s1[i..i+n2-1]* *строкой* *s2*
       1. int j = 0;
       2. *while* (j < n2 && str1[i+j] == str2[j]){
       3. j++;
       4. }
       5. *if* (j == n2){
       6. occurence = 1;
       7. }
       8. i++;
    9. }
    10. int new\_len;
    11. *if* (occurence){
        1. new\_len = i-1;
    12. }*else*{
        1. new\_len = n1;
    13. }
    14. String s3(str1, new\_len);
    15. *return* s3;
22. }
23. *friend* *const* String *operator* >>(*const* String& s1, *const* int& count){
    1. */\** *Оператор* *возвращает* *строку,* *дополненную* *count* *пробелами* *слева* *\*/*
    2. int new\_len = s1.n + count;
    3. char \*new\_str = *new* char[new\_len];
    4. *for* (int i = 0; i<count; i++){
       1. new\_str[i] = ' ';
    5. }
    6. *for* (int j = 0; j<s1.n; j++){
       1. new\_str[j+count] = s1.str[j];
    7. }
    8. String s3(new\_str, new\_len);
    9. *return* s3;
24. }
25. *public*:
26. **String**(char \*chars,int count) {
    1. str = *new* char[count];
    2. n = count;
    3. *for* (long int index = 0; index < count; index++) {
       1. str[index] = chars[index];
    4. }
27. }
28. void **showString**(){
    1. *for* (long int i = 0; i < n; i++) {
       1. cout << str[i];
    2. }
    3. cout << endl;
29. }
30. int **getN**(){
    1. *return* n;
31. }
32. char\* **getString**(){
    1. *return* str;
33. }
34. long int **countSpace**(){*//Счет* *пробелов*
    1. long int count = 0;
    2. *for* (long int i = 0; i < n; i++) {
       1. *if*(str[i] == ' ')
       2. count++;
    3. }
    4. *return* count;
35. }
36. *//Преоброзования* *прописных* *в* *строчные*
37. void **lower**(){
    * 1. *for* (int i = 0; i<n; i++){
      2. *if* (isUpper(str[i])){
         1. str[i] = toLower(str[i]);
      3. }
      4. }
    1. }
38. void **removePunctuationsToSring**(){
    * 1. int last = 0;
      2. *for* (int i = 0; i<n; i++){
      3. *if* (!isPunctuation(str[i])){
         1. *if* (i != last){
            1. str[last] = str[i];
         2. }
         3. last++;
      4. }
      5. }
      6. resizeString(last);
    1. }
39. };
40. int **main**(int argc, char \*argv[])
41. {
42. #ifdef Q\_OS\_WIN32
43. QTextCodec::setCodecForLocale(QTextCodec::codecForName("IBM 866"));
44. #endif
45. #ifdef Q\_OS\_LINUX
46. QTextCodec::setCodecForLocale(QTextCodec::codecForName("UTF-8"));
47. #endif
48. QCoreApplication app(*argc*, argv);
49. *return* app.exec();
50. }