**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” |

о

<main.cpp>

#include <QCoreApplication>

#include <QTextCodec>

#include <QTextStream>

#include <iostream>

*using* *namespace* std;

*class* **String**

{

*private*:

char \*str;

int n;

int **isUpper**(char c){*//Возв* *True* *если* *A...Z* *false* *a...z*

*return* (c >= 'A' && c <= 'Z');

}

int **isPunctuation**(char c){*//Проверка* *на* *знаки* *препинания*

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

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

}

char **toLower**(char c){*//Возв* *строчного* *символа* *алфавита*

*return* c - ('a' - 'A');

}

*//Увеличения* *или* *уменьшения* *строки*

void **resizeString**(int length){

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

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

newStr[i] = str[i];

}

n = length;

*delete*[] str;

str = newStr;

}

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

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

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

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;

}

*friend* *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;

}

*public*:

**String**(char \*chars,int count) {

str = *new* char[count];

n = count;

*for* (long int index = 0; index < count; index++) {

str[index] = chars[index];

}

}

void **showString**(){

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

cout << str[i];

}

cout << endl;

}

int **getN**(){

*return* n;

}

char\* **getString**(){

*return* str;

}

long int **countSpace**(){*//Счет* *пробелов*

long int count = 0;

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

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

count++;

}

*return* count;

}

*//Преоброзования* *прописных* *в* *строчные*

void **lower**(){

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

*if* (isUpper(str[i])){

str[i] = toLower(str[i]);

}

}

}

void **removePunctuationsToSring**(){

int last = 0;

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

*if* (!isPunctuation(str[i])){

*if* (i != last){

str[last] = str[i];

}

last++;

}

}

resizeString(last);

}

};

int **main**(int argc, char \*argv[])

{

#ifdef Q\_OS\_WIN32

QTextCodec::setCodecForLocale(QTextCodec::codecForName("IBM 866"));

#endif

#ifdef Q\_OS\_LINUX

QTextCodec::setCodecForLocale(QTextCodec::codecForName("UTF-8"));

#endif

QCoreApplication app(*argc*, argv);

*return* app.exec();

}