МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра информационных систем управления**

**ЛАБОРАТОРНАЯ РАБОТА № 8 ЗАДАЧА № 2**

По дисциплине «Объектно-ориентированное программирование»

На тему «Динамические структуры в языке C++»

|  |
| --- |
| Выполнил студент гр. Б8219 |
| О.В. Константинов |
|  |
| Проверил старший преподаватель |
| Г.Л. Берёзкина |
|  |
| (зачтено/не зачтено) |

г. Владивосток

2016

# Аннотация

Данный отчет подготовлен в рамках задания по дисциплине «Объектно-ориентированное программирование». Отчет призван систематизировать знания об использовании и реализации классов в языке C++. Отчет содержит список выполненных заданий с реализацией на языке C++ в среде программирования MS Visual Studio 2016.

# Задача 2

## Файл *“string\_lib.h”*

#pragma once

#include <iostream>

#include <string>

class String {

private:

char \*str;

public:

String(char ch) { str = new char[2]; str[0] = ch; str[1] = '\0'; };

String(char \*\_str = "") { str = new char[strlen(\_str) + 1]; strcpy(str, \_str); };

String(const String &rhs) { str = new char[strlen(rhs.str) + 1]; strcpy(str, rhs.str); };

~String() { delete[] str; };

size\_t length() const { return strlen(str); };

const char \*c\_str() const { return str; };

String& operator = (const String&);

friend String operator + (const String &lhs, const String &rhs) { String ts = lhs; return ts += rhs; };

friend String operator - (const String &lhs, const String &rhs) { String ts = lhs; return ts -= rhs; };

String& operator += (const String&);

String& operator -= (const String&);

String operator ~() const;

friend bool operator == (const String &lhs, const String &rhs) { return !strcmp(lhs.str, rhs.str); };

friend bool operator != (const String &lhs, const String &rhs) { return strcmp(lhs.str, rhs.str); };

friend std::ostream& operator << (std::ostream &os, const String &obj) { return os << obj.str; };

friend std::istream& operator >> (std::istream&, String&);

};

String String::operator ~() const {

char \*ts = new char[strlen(str) + 1];

for (size\_t i = 1; i <= strlen(str); i++)

ts[strlen(str) - i] = str[i - 1];

ts[strlen(str)] = '\0';

return ts;

}

String& String::operator -= (const String& ths) {

size\_t k = 0, len = strlen(str);

for (size\_t i = 0, j; i < len; i++) {

for (j = 0; j < strlen(ths.str) && str[i] != ths.str[j]; j++);

if (str[i] != ths.str[j]) str[k++] = str[i];

}

str[k++] = '\0';

char \*ts = new char[k];

strcpy(ts, str);

delete str;

str = ts;

return \*this;

}

String& String::operator = (const String& rhs) {

if (this == &rhs) return \*this;

delete[] str;

str = new char[strlen(rhs.str) + 1];

strcpy(str, rhs.str);

return \*this;

}

String& String::operator += (const String& rhs) {

size\_t sz = strlen(str) + strlen(rhs.str);

char\* ts = new char[sz + 1];

for (size\_t i = 0; i < strlen(str); i++) ts[i] = str[i];

for (size\_t i = strlen(str), j = 0; i <= sz; i++, j++) ts[i] = rhs.str[j];

delete[] str;

str = ts;

return \*this;

}

std::istream& operator >> (std::istream& is, String& obj) {

char BUFF[2048];

is.getline(BUFF, sizeof BUFF);

obj = BUFF;

return is;

}

## Файл *“main.cpp”*

/\*

Задача 2. Строка символов. Реализовать операции:

сравнение строк (операция ==);

удаления из строки указанного символа (операция -);

переворота строки (операция ~).

Кроме того, членом класса сделать также функцию с именем strset() для удаления из первой строки всех символов, встречающихся во второй строке.

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "string\_lib.h"

#include <iostream>

#include <string.h>

using namespace std;

typedef bool(\*TEST\_FuncT)();

bool CompareArg(const String &S, const char \*C) {

#if \_DEBUG

cout << C << '[' << strlen(C) << ']' << " = "

<< S << '[' << S.length() << ']' << endl;

#endif

return S.length() == strlen(C) && strcmp(S.c\_str(), C) == 0 ? true : false;

}

bool TEST\_CreateStr() {

String s1;

String s2 = '1';

String s3 = "String";

String s4 = s3;

return s1.length() == 0

&& s2.length() == 1

&& s3.length() == 6

&& s4.length() == 6;

}

bool TEST\_AssignStr() {

String s1;

String s2 = "Новая строка";

s1 = "Привет";

bool res = CompareArg(s1, "Привет");

s1 = "Я изменил строку";

res = res && CompareArg(s1, "Я изменил строку");

s1 = s2;

s2 = "Строка изменилась";

res = res && CompareArg(s1, "Новая строка");

return res;

}

bool TEST\_RevertStr() {

String s1 = "дед Макар раком";

return CompareArg(~s1, "мокар ракаМ дед");

}

bool TEST\_RemoveSmbStr() {

String s1 = "Удалю $ и @ из строки";

s1 -= '$';

s1 = s1 - '@';

return CompareArg(s1, "Удалю и из строки");

}

bool TEST\_RemoveSubstrStr() {

String s1 = "Удалить буквы 'лож' т. е. ложка лол жоож";

s1 -= "лож";

return CompareArg(s1, "Удаить буквы '' т. е. ка ");

}

bool TEST\_EmptyStr() {

String s1 = "Эта строка станет пустой";

s1 = s1 - "Эта" - "строка станет" - "пуй";

return CompareArg(s1, "");

}

bool TEST\_CompareStr() {

String s1 = '1', s2 = "12", s3 = '1', s4 = "13";

return s1 != s2 && s1 == s3 && s2 != s4 && s2 == s2 && s4 != s2;

}

bool TEST\_ConcatenateStr() {

String s1 = "Я", s2 = " делаю", s3 = ' ', s4 = "тесты!";

s1 += s2;

s2 = s1 + s3;

s2 += s4;

return s2 == "Я делаю тесты!";

}

int main() {

TEST\_FuncT funcs[] = {

TEST\_CreateStr,

TEST\_AssignStr,

TEST\_RevertStr,

TEST\_RemoveSmbStr,

TEST\_RemoveSubstrStr,

TEST\_EmptyStr,

TEST\_CompareStr,

TEST\_ConcatenateStr

};

unsigned int i, size = sizeof(funcs) / sizeof(TEST\_FuncT);

setlocale(LC\_ALL, "Russian");

cout << "TEST COUNT " << size << endl;

for (i = 0; i < size; i++) {

cout << "TEST " << i + 1 << " ";

funcs[i]() ? cout << "OK" : cout << "FAIL";

cout << endl;

}

cout << "TEST IS FINISHED" << endl;

system("PAUSE");

return 0;

}