

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Системы автоматизированного проектирования | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Программирование | | |
| Курс | I | | | Группа | 404 |

Отчёт по лабораторной № 4

Вариант № 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 404 |  |  |  | Д. К. Азаров |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | И. Г. Корниенко |
|  |  | (дата, подпись) |  | А. К. Федин |

### 1 Постановка задачи

Найти в тексте все последовательности идущих подряд одинаковых символов и заменить их сигнатурой {символ, количество}. Минимальная длинна последовательности, которая может подвергаться замене, задается пользователем. Предусмотреть режим восстановления оригинального текста. Пример: текст «длинношеее животное» должен быть заменен текстом «длиннош{e, 3} животное».

### 2 Исходные данные

Строка

### 3 Особые ситуации

Необходимо рассмотреть следующую особую ситуацию:

количество повторяющихся символов, которые можно заменить сигнатурой должно быть больше нуля.

### 4 Математические методы и алгоритмы решения задач

Кодировка:

1. Проходимся по строкам, пока не встретим заданное кол-во повторяющихся символов;
2. Создаем новую строку и записываем в нее сигнатуру {буква,кол-во};
3. Заменяем повторяющиеся буквы основной строки новой строкой;
4. Повторяем пункт 1, начиная с символа после сигнатуры

Декодировка:

1. Проходимся по строкам, пока не встретим сигнатуру {буква,кол-во};
2. Заменяем сигнатуру на кол-во букв, указанных в сигнатуре;
3. Повторяем пункт 1, начиная с символа после последней вставленной буквы

### 5 Форматы представления данных

Таблица 1 – константы, используемые в программе

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя переменной** | **Тип** | **Значение** | **Описание** |
| ALL\_TEST\_PASSED | const int | 6 | Проверка: прошли ли все тесты проверку |

Таблица 2 – перечисляемые типы, используемые в программе

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя типа** | **Имя переменной** | **Значение** | **Описание** |
| ChoiceTypeInput | - | - | Используется для проверки выбора пользователя |
| CHOICE\_KEYBOARD | 1 | Используется для выбора способа ввода через клавиатуру |
| CHOICE\_FILE | 2 | Используется для выбора способа ввода через клавиатуру |
| Opening | - | - | Проверка открыт ли файл |
| FAILED | 0 | Файл не открыт |
| COMPLITE | 1 | Файл открыт |
| FileStatus | - | - | Проверка на существование файла |
| EXIST | 0 | Файл существует |
| NON\_EXISTENT | 1 | Файл не существует |
| TransformingChoice | - | - | Используется для проверки пользовательского ввода |
| CODING\_CHOICE | 1 | При выборе кодировки строк |
| DECODING\_CHOICE | 2 | При выборе декодировки строк |

Таблица 3 – Классы, используемые в программе

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя класса** | **Имя переменной** | **Тип** | **Описание** |
| Input | - | - | Абстрактный класс |
| Read | virtual string | Считывания данных в массив |
| KeyboardInput | Read | string | Считывание данных из консоли |
| FileInput | Read | string | Считывание данных из файла |

### 6 Структура программы

Таблица 4 – Функции, составляющие модуль save\_data.h

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| SaveInputString | Сохранение введённых данных |
| SaveChangeString | Сохранение преобразованных данных |

Таблица 5 – Функции, составляющие модуль formatting\_input.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| GetInt | Проверяет является ли введённое число цифрой и возвращает его целым числом |

Таблица 6 – Функции, составляющие модуль open\_file.h

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| OpenFileInput | Реализует открытие файлов для считывания |
| OpenFileOutput | Реализует открытие файлов для записи |

Таблица 7 – Функции, составляющие модуль menu.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| StringInputMenu | Меню выбора типа ввода |
| CodingOrDecodingMenu | Меню выбора типа работы со строками |
| EndProgramMenu | Выводит меню завершения программы |
| StringOrTestingMenu | Меню выбора работы со строкой или тестирования программы |
| Greeting | Приветствие |

Таблица 8 – Функции, составляющие модуль greeting.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Greeting | Выводит приветствие |

Таблица 9 – Функции, составляющие модуль string\_work.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| StringInterface | Реализует работу со строками |

Таблица 10 – Функции, составляющие модуль test.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| Test | Проверяет успешность всех тестов |
| TestingProgram | Тестирует программу |

Таблица 11 – Функции, составляющие модуль input.cpp

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| InputType | Обрабатывает выбор типа ввода |

Таблица 12 – Функции, составляющие модуль encode\_string.h

|  |  |
| --- | --- |
| **Имя** | **Описание** |
| CodingString | Кодирует строку |
| DecodingString | Декодирует строку |
| CodingOrDecodingChoice | Обрабатывает выбор типа работы со строками |

### 7 Блок-схем алгоритмов программы



Рисунок 1 – Блок-схема алгоритма кодировки строки



Рисунок 2 – Блок-схема алгоритма декодирования строки

### 8 Описание хода выполнения лабораторной работы

1. В ходе лабораторной работы было создано решение (Solution) в интегрированной среде разработки Microsoft Visual Studio C++ 2017. В нем был создан проект.
2. В ходе выполнения программы была выявлена проблема: не была предусмотрена декодировка строк.
3. В ходе выполнения программы была выявлена проблема: при выборе минимального кол-ва букв, которые должны быть закодированы, равным 1, последний символ не кодировался. Для решения данной проблемы было добавлено дополнительная обработка данного события.
4. В ходе выполнения программы была выявлена проблема: при выборе минимального кол-ва букв, которые должны быть закодированы, равным 1, кодировался особый символ ‘\n’. Для решения данной проблемы было добавлено дополнительная обработка данного события.
5. В ходе выполнения программы была выявлена проблема: после выбора типа ввода приходилось освобождать выделенную память в отдельной функции. Для решения данной проблемы было принято решение использовать умный указатель unique\_ptr<>.

### 9 Результаты работы программы

В результате программа выводит преобразованную строку.

### 10 Исходный текст программы

[Начало main.cpp ---]

#include <iostream>

#include <string>

#include "main\_menu.h"

using namespace std;

enum ChoiceTypeInput {

CHOICE\_STRING = 1,

CHOICE\_TESTING

};

int GetInt(void);

void Test(void);

void StringInterface(void);

int main() {

setlocale(LC\_ALL, "Russian");

Greeting();

int choice = 0;

string choice\_repeat = "";

do {

do {

StringOrTestingMenu();

choice = GetInt();

switch (choice) {

case CHOICE\_STRING:

StringInterface();

break;

case CHOICE\_TESTING:

Test();

break;

default:

cout << "Такого выбора нет. Попробуйте еще раз" << endl;

}

} while ( (choice != CHOICE\_STRING) && (choice != CHOICE\_TESTING) );

do {

EndProgramMenu();

cin >> choice\_repeat;

if (choice\_repeat.compare("2") && choice\_repeat.compare("1")) {

cout << "Такого выбора нет" << endl;

}

} while (choice\_repeat.compare("2") && choice\_repeat.compare("1"));

} while (choice\_repeat.compare("2"));

return 0;

}

[--- Конец main.cpp

[Начало string\_work.cpp ---]

#include <iostream>

#include <string>

#include "input.h"

#include "save\_data.h"

#include "encode\_string.h"

using namespace std;

unique\_ptr<Input> ChoiceInputType(bool \*user\_selected\_file\_input);

int GetInt(void);

void StringInterface(void) {

bool user\_selected\_file\_input = false;

unique\_ptr<Input> inp = ChoiceInputType(&user\_selected\_file\_input);

string input\_string = inp->Read();

string transforming\_string = "";

if (CodingOrDecodingChoice() == CODING\_CHOICE) {

cout << "При каком минимальном кол-ве повторяющихся символов вы хотите заменять их на {буква, кол-во}?" << endl;

int number\_of\_characters = 0;

do {

cout << "Кол-во: ";

number\_of\_characters = GetInt();

if (number\_of\_characters <= 0) {

cout << "Количество должно быть больше 0. Попробуйте другое число." << endl;

}

} while (number\_of\_characters <= 0);

transforming\_string = CodingString(input\_string, number\_of\_characters);

} else {

transforming\_string = DecodingString(input\_string);

}

cout << "Преобразованное предложение" << endl;

cout << transforming\_string;

if (user\_selected\_file\_input != true) {

SaveInputString(input\_string);

}

SaveChangeString(transforming\_string);

}

[--- Конец string.cpp]

[Начало open\_file.cpp ---]

#include <iostream>

#include <fstream>

#include <string>

#include <filesystem>

using namespace std;

enum Opening {

FAILED,

COMPLITE

};

ofstream OpenFileOutput(void) {

string path = "";

bool file\_open\_complit = false;

ofstream file;

error\_code ec;

do {

cout << "Введите путь к файлу" << endl;

getline(cin, path);

if (ifstream(path)) {

if (!filesystem::is\_regular\_file(path, ec)) {

cout << "Этот файл не может быть открыт. Попробуйте еще раз" << endl;

continue;

}

cout << "Этот файл существует. Хотите его перезаписать?" << endl;

string choice;

do {

cout << "1 - Да" << endl

<< "2 - Нет" << endl

<< "Выбор: ";

cin >> choice;

if (choice.compare("2") && choice.compare("1")) {

cout << "Такого выбора нет." << endl

<< "Хотите перезаписать файл?" << endl;

}

} while (choice.compare("2") && choice.compare("1"));

if (!choice.compare("2")) {

continue;

}

}

file.open(path);

if (!filesystem::is\_regular\_file(path, ec)) {

cout << "Этот файл не может быть открыт." << endl;

continue;

}

if (!file.is\_open()) {

cout << "Этот файл не может быть открыт." << endl;

continue;

}

file\_open\_complit = true;

} while (!file\_open\_complit);

return file;

}

ifstream OpenFileInput(void) {

cout << "Введите путь к файлу" << endl;

string path;

int opening\_file = 0;

error\_code ec;

ifstream file;

do {

do {

getline(cin, path);

if (!filesystem::is\_regular\_file(path, ec)) {

cout << "Этот файл не может быть открыт." << endl

<< "Введите другой путь или имя файла: ";

}

} while (!filesystem::is\_regular\_file(path, ec));

file.open(path);

if (file.is\_open()) {

opening\_file = COMPLITE;

} else {

cout << "Этот файл не может быть открыт." << endl

<< "Введите другой путь или имя файла: ";

opening\_file = FAILED;

}

} while (opening\_file == FAILED);

return file;

}

[--- Конец open\_file.cpp]

[Начало menu.cpp ---]

#include <iostream>

using namespace std;

void StringInputMenu(void) {

cout << endl << "Как вы хотите ввести строку?" << endl

<< "1)Консольный ввод" << endl

<< "2)Ввод из файла" << endl

<< "Ваш выбор: ";

}

void EndProgramMenu(void) {

cout << endl << "Хотите повторить программу?" << endl

<< "1 - Да" << endl

<< "2 - Нет" << endl

<< "Ваш выбор: ";

}

void StringOrTestingMenu(void) {

cout << endl << "Что вы хотите?" << endl

<< "1)Преобразование строки" << endl

<< "2)Протестировать программу" << endl

<< "Ваш выбор: ";

}

void CodingOrDecodingMenu(void) {

cout << endl << "Что вы хотите?" << endl

<< "1)Закодировать строку" << endl

<< "2)Декодировать строку" << endl

<< "Ваш выбор: ";

}

[--- Конец menu.cpp]

[Начало input.cpp ---]

#include <iostream>

#include <string>

#include <fstream>

#include <ctime>

#include "open\_file.h"

#include "input.h"

using namespace std;

enum ChoiceTypeInput {

CHOICE\_KEYBOARD = 1,

CHOICE\_FILE

};

enum Opening {

FAILED,

COMPLITE

};

int GetInt(void);

void StringInputMenu(void);

string KeyboardInput::Read() {

cout << "Введите предложение (чтобы остановить ввод введите 'Q')" << endl;

string input\_string;

string line = "";

int i = 1;

do {

line = "";

cout << i++ << ")";

getline(cin, line);

if (line != "Q") {

input\_string = input\_string + line + '\n';

}

} while (line != "Q");

return input\_string;

}

string FileInput::Read() {

ifstream file;

string input\_string = "";

bool correctness\_string = 0;

do {

cout << endl << "Ввод предложения." << endl;

file = OpenFileInput();

string line;

while (!file.eof()) {

getline(file, line);

input\_string += line;

}

if (!input\_string.length()) {

cout << "Файл ничего не содержит. Попробуйте еще раз." << endl;

continue;

}

correctness\_string = COMPLITE;

cout << endl << "Введённое предложение:" << endl

<< input\_string << endl;

file.close();

} while (correctness\_string == FAILED);

return input\_string;

}

unique\_ptr<Input> ChoiceInputType(bool \*user\_selected\_file\_input) {

int choice = 0;

do {

StringInputMenu();

choice = GetInt();

if (choice == CHOICE\_KEYBOARD) {

return unique\_ptr<KeyboardInput>(new KeyboardInput);

} else if (choice == CHOICE\_FILE) {

\*user\_selected\_file\_input = true;

return unique\_ptr<FileInput>(new FileInput);

} else {

cout << "Такого выбора нет" << endl;

}

} while ((choice != CHOICE\_KEYBOARD) && (choice != CHOICE\_FILE));

exit(0);

}

[--- Конец input.cpp]

[Начало greeting.cpp ---]

#include <iostream>

using namespace std;

void Greeting(void){

cout << "Лабораторная работа № 4" << endl

<< "Для преобразования строк используется язык C++" << endl

<< "Задача найти в тексте все последовательности идущих подряд одинаковых символов"

"и заменить их сигнатурой{ символ, количество }.Минимальная длинна"

"последовательности, которая может подвергаться замене, задается"

"пользователем." << endl

<< "Студент группы 404, Азаров Даниил Константинович. 2021 год" << endl;

}

[--- Конец greeting.cpp]

[Начало formatting\_input.cpp ---]

#include <iostream>

#include <limits>

using namespace std;

int GetInt(void) {

int input = 0;

cin >> input;

while (cin.fail()) {

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Ошибка ввода. Введите число." << endl;

cin >> input;

}

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

return input;

}

[--- Конец formatting\_input.cpp]

[Начало encode\_string.cpp ---]

#include <iostream>

#include <string>

#include <cctype>

using namespace std;

enum TransformingChoice {

CODING\_CHOICE = 1,

DECODING\_CHOICE

};

int GetInt(void);

void CodingOrDecodingMenu(void);

string CodingString(string input\_string, int number\_of\_characters) {

int last\_change\_position = 0;

for (int i = 0; i < (input\_string.length() - 1); i++) {

int first\_repeated\_character = i;

if (input\_string[i] == '\n') {

continue;

}

while ( (i != (input\_string.length() - 1)) && (input\_string[i] == input\_string[i + 1]) ) {

i++;

}

if (((i + 1) - first\_repeated\_character) >= number\_of\_characters) {

string new\_string = "{";

new\_string += input\_string[first\_repeated\_character];

new\_string += ",";

new\_string += to\_string((i + 1) - first\_repeated\_character);

new\_string += "}";

input\_string.replace(first\_repeated\_character, ((i + 1) - first\_repeated\_character), new\_string);

i = (first\_repeated\_character - 1) + new\_string.length();

last\_change\_position = i;

}

}

if ( (number\_of\_characters == 1) && (last\_change\_position == (input\_string.length() - 1)) ) {

string new\_string = "{";

new\_string += input\_string[last\_change\_position + 1];

new\_string += ",1}";

input\_string.replace(last\_change\_position + 1, new\_string.length() - 1, new\_string);

}

return input\_string;

}

string DecodingString(string input\_string) {

const int ASCII\_OFFSET = 48;

for (int i = 0; i < (input\_string.length() - 3); i++) {

if ((input\_string[i] == '{') && (input\_string[i + 2] == ',') && (isdigit(input\_string[i + 3]))) {

int safe\_index = i;

int number\_of\_characters = 0;

char letter = input\_string[i + 1];

while (isdigit(input\_string[i + 3])) {

number\_of\_characters = number\_of\_characters \* 10 + static\_cast<int>(input\_string[i + 3]) - ASCII\_OFFSET;

i++;

}

if (input\_string[i + 3] == '}') {

input\_string.replace(safe\_index, ( (i + 3) - safe\_index + 1), number\_of\_characters, letter);

i--;

}

}

}

return input\_string;

};

int CodingOrDecodingChoice() {

int choice = 0;

do {

CodingOrDecodingMenu();

choice = GetInt();

switch (choice) {

case CODING\_CHOICE:

return CODING\_CHOICE;

case DECODING\_CHOICE:

return DECODING\_CHOICE;

default:

cout << "Такого выбора нет" << endl;

}

} while ( (choice != CODING\_CHOICE) && (choice != DECODING\_CHOICE) );

exit(0);

}

[--- Конец encode\_string.cpp]

[Начало save\_data.cpp ---]

#include <iostream>

#include <fstream>

#include <string>

#include "open\_file.h"

using namespace std;

void SaveInputString(string input\_string) {

cout << endl << "Хотите сохранить введённый текст в файл?" << endl;

string choice = "";

do {

cout << "1 - Да" << endl

<< "2 - Нет" << endl

<< "Выбор: ";

cin >> choice;

if (choice.compare("2") && choice.compare("1")) {

cout << "Такого выбора нет." << endl

<< "Хотите сохранить введённый текст в файл?" << endl;

}

} while (choice.compare("2") && choice.compare("1"));

if (choice.compare("2")) {

ofstream file;

file = OpenFileOutput();

file << input\_string << endl;

file.close();

}

};

void SaveChangeString(string transforming\_string) {

cout << endl << "Хотите сохранить в файл преобразованную строку?" << endl;

string choice = "";

do {

cout << "1 - Да" << endl

<< "2 - Нет" << endl

<< "Выбор: ";

cin >> choice;

if (choice.compare("2") && choice.compare("1")) {

cout << "Такого выбора нет." << endl

<< "Хотите сохранить в файл преобразованную строку?" << endl;

}

} while (choice.compare("2") && choice.compare("1"));

if (choice.compare("2")) {

ofstream file;

file = OpenFileOutput();

file << transforming\_string << endl;

file.close();

}

}

[--- Конец save\_data.cpp]

[Начало open\_file.h ---]

#pragma once

#include <fstream>

std::ofstream OpenFileOutput(void);

std::ifstream OpenFileInput(void);

[--- Конец open\_file.h]

[Начало save\_data.h ---]

#pragma once

void SaveInputString(string input\_string);

void SaveChangeString(string transforming\_string);

[--- Конец save\_data.h]

[Начало main\_menu.h ---]

#pragma once

void EndProgramMenu(void);

void StringOrTestingMenu(void);

void Greeting(void);

[--- Конец main\_menu.h]

[Начало input.h ---]

#pragma once

#include <string>

using namespace std;

class Input {

public:

virtual string Read() = 0;

virtual ~Input() = default;

};

class KeyboardInput final : public Input {

public:

string Read();

};

class FileInput final : public Input {

public:

string Read();

};

[--- Конец input.h]

[Начало encode\_string.h ---]

#pragma once

#include <iostream>

#include <string>

enum TransformingChoice {

CODING\_CHOICE = 1,

DECODING\_CHOICE

};

std::string CodingString(std::string input\_string, int number\_of\_characters);

std::string DecodingString(std::string input\_string);

int CodingOrDecodingChoice(void);

[--- Конец encode\_string.h]

[Начало test.h ---]

#include <iostream>

#include <string>

#include "encode\_string.h"

using namespace std;

const int ALL\_TEST\_PASSED = 6;

enum IncorrectSorting {

ARR\_BUBBLE\_SORT,

ARR\_SELECTION\_SORT,

ARR\_INSERTION\_SORT,

ARR\_SHELL\_SORT

};

bool TestingDecoding(string input\_string, string expected\_string, int value\_test\_passed) {

string output\_string = DecodingString(input\_string);

if (output\_string != expected\_string) {

cout << endl << "Тест" << value\_test\_passed + 1 << " не пройден. Проверялась декодировка предложения" << endl;

if (output\_string.length() != expected\_string.length()) {

cout << "Длина преобразованной строки не равна длине ожидаемой преобразованной строки." << endl;

} else {

cout << "Ошибка в ";

for (int i = 0; i < output\_string.length(); i++) {

if (output\_string[i] != expected\_string[i]) {

cout << i + 1 << " ";

}

}

cout << "элементе(ах) преобразованной строки." << endl;

}

cout << "Введённая строка: " << input\_string << endl;

cout << "Ожидаемая преобразованная строка: " << expected\_string << endl;

cout << "Преобразованная строка: " << output\_string << endl;

return false;

}

return true;

}

bool TestingCoding(string input\_string, string expected\_string, int number\_of\_characters, int value\_test\_passed) {

string output\_string = CodingString(input\_string, number\_of\_characters);

if (output\_string != expected\_string) {

cout << endl << "Тест" << value\_test\_passed + 1 << " не пройден. Проверялась кодировка предложения" << endl;

if (output\_string.length() != expected\_string.length()) {

cout << "Длина преобразованной строки не равна длине ожидаемой преобразованной строки." << endl;

}

else {

cout << "Ошибка в ";

for (int i = 0; i < output\_string.length(); i++) {

if (output\_string[i] != expected\_string[i]) {

cout << i + 1 << " ";

}

}

cout << "элементе(ах) преобразованной строки." << endl;

}

cout << "Введённая строка: " << input\_string << endl;

cout << "Ожидаемая преобразованная строка: " << expected\_string << endl;

cout << "Преобразованная строка: " << output\_string << endl;

return false;

}

return true;

}

void Test(void) {

int value\_test\_passed = 0;

string input\_string = "Hello, my best friendd! )))";

int number\_of\_characters = 2;

string expected\_string = "He{l,2}o, my best frien{d,2}! {),3}";

if (TestingCoding(input\_string, expected\_string, number\_of\_characters, value\_test\_passed) == true) {

value\_test\_passed++;

}

input\_string = "He{l,2}o, my best frien{d,2}! {),3}";

number\_of\_characters = NULL;

expected\_string = "Hello, my best friendd! )))";

if (TestingDecoding(input\_string, expected\_string, value\_test\_passed) == true) {

value\_test\_passed++;

}

input\_string = "I lovvvveeeee programming!!!!!";

number\_of\_characters = 4;

expected\_string = "I lo{v,4}{e,5} programming{!,5}";

if (TestingCoding(input\_string, expected\_string, number\_of\_characters, value\_test\_passed) == true) {

value\_test\_passed++;

}

input\_string = "I lo{v,4}{e,5} programming{!,5}";

number\_of\_characters = NULL;

expected\_string = "I lovvvveeeee programming!!!!!";

if (TestingDecoding(input\_string, expected\_string, value\_test\_passed) == true) {

value\_test\_passed++;

}

input\_string = "sorting laboratory is veeeeeeeeeeeeeeeeeery bad\n";

number\_of\_characters = 1;

expected\_string = "{s,1}{o,1}{r,1}{t,1}{i,1}{n,1}{g,1}{ ,1}{l,1}{a,1}{b,1}{o,1}{r,1}{a,1}{t,1}{o,1}{r,1}{y,1}{ ,1}{i,1}{s,1}{ ,1}{v,1}{e,18}{r,1}{y,1}{ ,1}{b,1}{a,1}{d,1}\n";

if (TestingCoding(input\_string, expected\_string, number\_of\_characters, value\_test\_passed) == true) {

value\_test\_passed++;

}

input\_string = "{s,1}{o,1}{r,1}{t,1}{i,1}{n,1}{g,1}{ ,1}{l,1}{a,1}{b,1}{o,1}{r,1}{a,1}{t,1}{o,1}{r,1}{y,1}{ ,1}{i,1}{s,1}{ ,1}{v,1}{e,18}{r,1}{y,1}{ ,1}{b,1}{a,1}{d,1}";

number\_of\_characters = NULL;

expected\_string = "sorting laboratory is veeeeeeeeeeeeeeeeeery bad";

if (TestingDecoding(input\_string, expected\_string, value\_test\_passed) == true) {

value\_test\_passed++;

}

if (value\_test\_passed == ALL\_TEST\_PASSED) {

cout << endl << "Все тесты пройдены" << endl;

}

}

[--- Конец test.h]