**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Алгоритмы и структуры данных»**

Тема: Очереди и стеки

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Сергеев А.Д. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2019

## Цель работы.

Ознакомиться с основными характеристиками и особенностями типов данных стек и очередь, изучить особенности их реализации на языке программирования C++. Разработать программу, использующую иерархические списки и их рекурсивную обработку, высчитывающую значение выражения.

## Задание.

Определить, имеет ли заданная в файле *F* символьная строка следующую структуру: *a* *D* *b* *D* *c* *D* *d* ..., где каждая строка *a*, *b*, *c*, *d*, ..., в свою очередь, имеет вид *x*1 *C* *x*2. Здесь *x*1 есть строка, состоящая из символов *A* и *B*, а *x*2 - строка, обратная строке *x*1 (т. е. если *x*1 = *ABABB*, то *x*2 = *BBABA*). Таким образом, исходная строка состоит только из символов *A*, *B*, *C* и *D*. Исходная строка может читаться только последовательно (посимвольно) слева направо.

## Основные теоретические положения.

Согласно определению, стек — это такой список, функциональная спецификация которого задается следующими определениями (множество непустых стеков обозначим как *Non*\_*null*\_*stack*):

1) *Create*: ® *Stack* (α);

2) *Null*: *Stack* (α) ® *Boolean*;

4) *Pop*: *Non*\_*null*\_*stack* (α) ® α Ä *Stack* (α).

5) *Push*: α Ä *Stack* (α) ® *Stack* (α)

## Выполнение работы.

Написание работы производилось на базе операционной системы Ubuntu, в среде CLion, а также с использованием библиотек qt и среды QTCreator.

Для выполнения поставленной задачи был создан класс visible\_stack, содержащий в себе начало и конец двусвязного списка. Стандартные методы работы со стеком применяются к его концу. Кроме стандартных методов он содержит метод toString, выводящий содержимое списка, начиная с первого элемента, на экран в виде строки и декорирующий эту строку при помощи специальных декоративных символов.

Класс lab3 содержит в себе алгоритм проверки введённой строки на предмет соответствия критериям задания (сразу или посимвольно). Также он содержит в статичных полях данные, необходимые для пошагового отображения состояния работы программы.

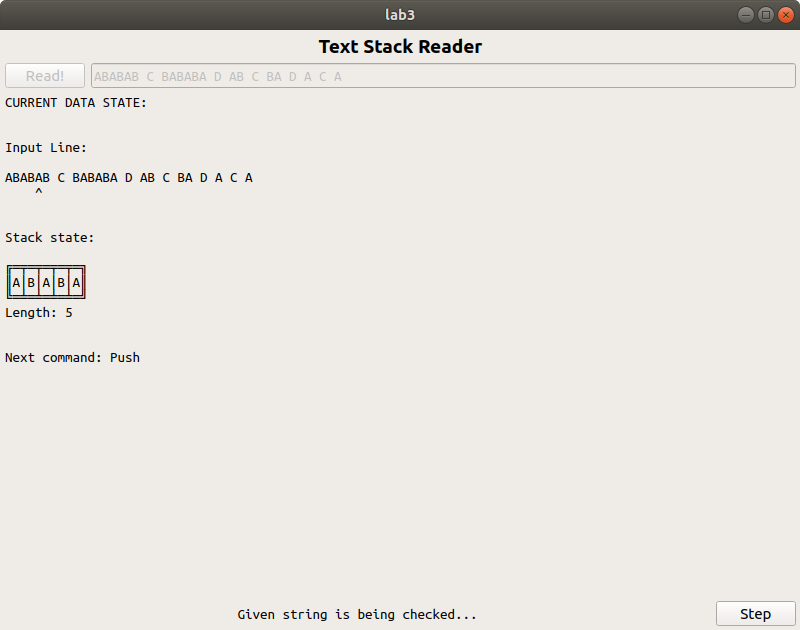
Класс universal\_exception представляет из себя контейнер для предоставления информации об исключении во время выполнения.

Для работы в консольном и gui режиме имеется два файла, содержащих процедуру main: main\_console.cpp и main\_gui.cpp. Во избежание некорректной работы, в make файле, используемом средой программирования CLion указан только main\_console.cpp, а в файле, используемом QT — только main\_gui.cpp.

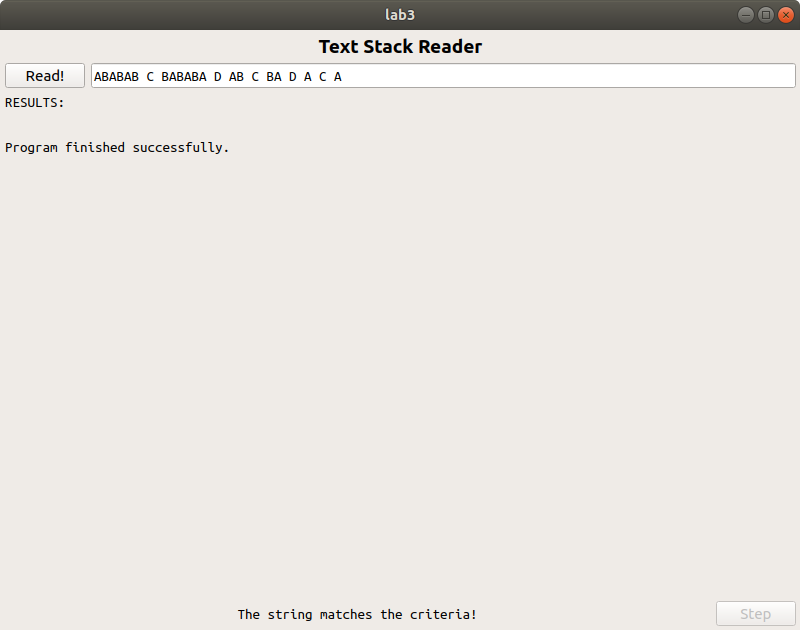
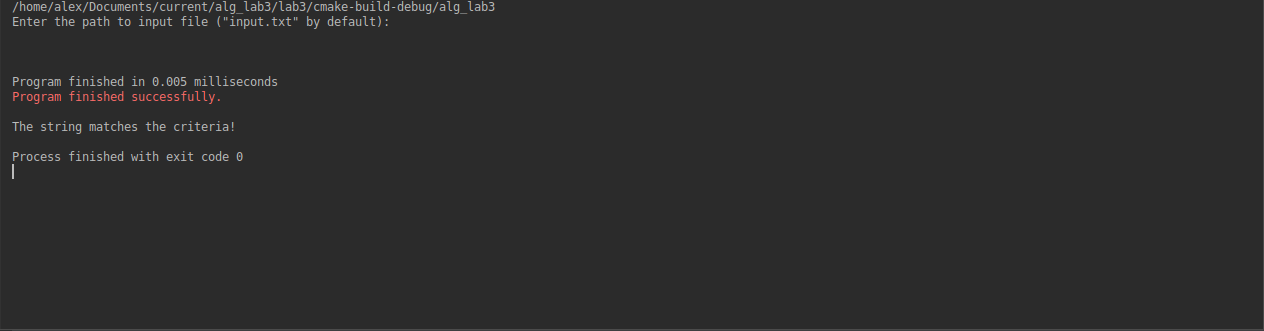
## Оценка эффективности алгоритма.

Алгоритм имеет линейную сложность, так как иерархический список строится, заполняется и рассчитывается один раз.

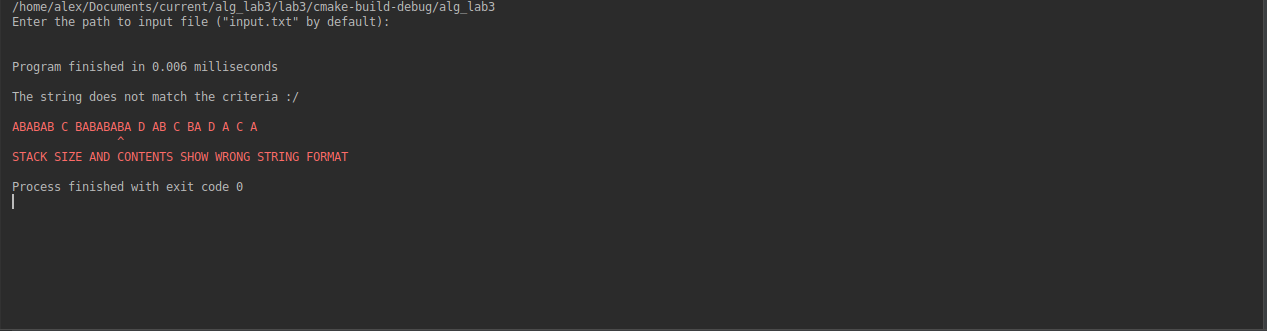
**Тестирование программы.**

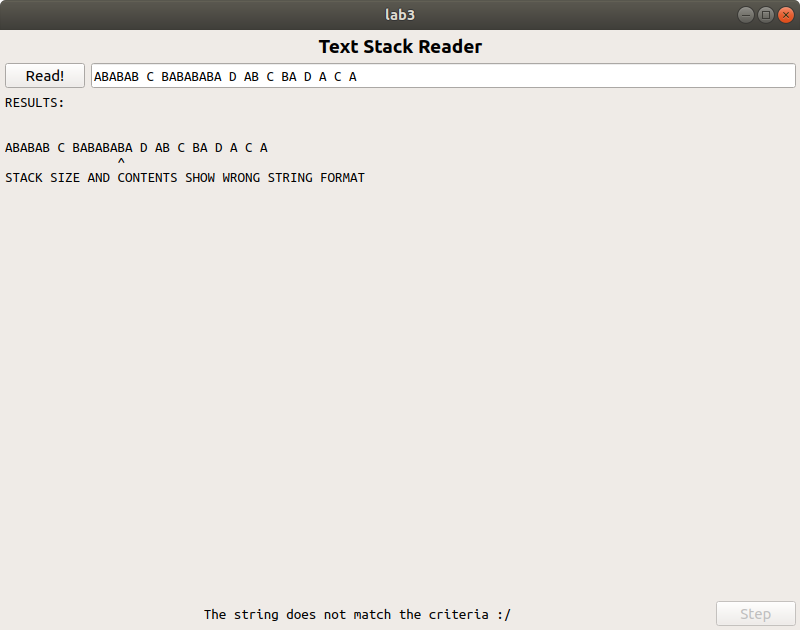
Ниже представлен снимок экрана работающей в режиме gui программы, а также результаты трёх различных тестов в консольном и gui режиме.

Ввод «ABABAB C BABABA D AB C BA D A C A» (корректный ввод):

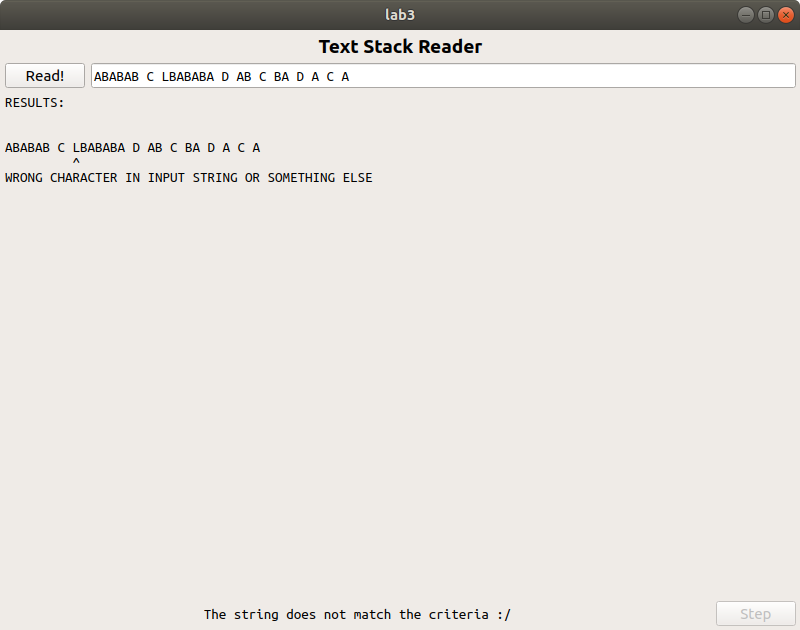
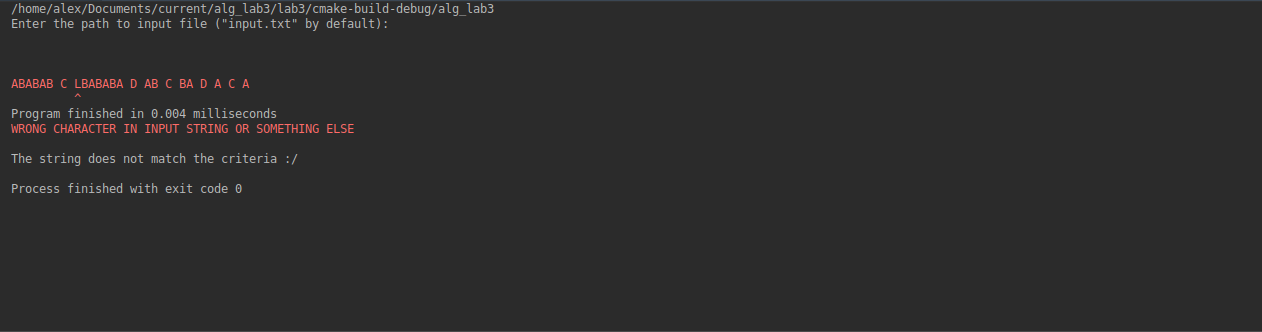


Ввод «ABABAB C BABABABA D AB C BA D A C A» (некорректно задана строка):





Ввод «ABABAB C LBABABA D AB C BA D A C A» (недопустимый символ в строке):



## Выводы.

В ходе выполнения лабораторной работы были изучены такие структуры данных как стек и очередь, а также методы их обработки. Была реализована программа на С++, использующая стек, которая анализирует строку, определяя ее соответствие условию поставленной задачи.

# Приложение А Исходный код программы

**Файл main\_console.cpp:**

#include <bits/stdc++.h>

#include "universal\_exception.h"

#include "lab3.h"

using namespace std;

string readFromFile(string fileName);

int main() {

try {

string input = readFromFile("input.txt");

double millis = lab3::process(input, false);

cout << endl << "Program finished in " << millis << " milliseconds" << endl;

cerr << endl << lab3::getMessage() << endl;

cout << endl << lab3::getAnswer() << endl;

} catch (universal\_exception& UE) {

cerr << UE.toString() << endl;

return EXIT\_ERROR\_CODE;

}

return 0;

}

string readFromFile(string fileName) {

string name;

cout << "Enter the path to input file (\"" + fileName + "\" by default):" << endl;

getline(cin, name);

if (!name.empty()) fileName = name;

ifstream infile(fileName);

if (!infile || infile.fail()) {

throw universal\_exception(INPUT\_FILE\_EXCEPTION\_CODE);

}

string input;

getline(infile, input);

infile.close();

if (input.empty()) {

throw universal\_exception(INPUT\_FILE\_EXCEPTION\_CODE);

}

return input;

}

**Файл main\_gui.cpp:**

#include "mainwindow.h"

#include <QApplication>

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

QApplication a(argc, argv);

MainWindow w;

w.setWindowTitle("lab3");

w.show();

return a.exec();

}

**Файл lab3.h:**

#ifndef ALG\_LAB1\_LAB3\_H

#define ALG\_LAB1\_LAB3\_H

#include <bits/stdc++.h>

#include "visible\_stack.h"

#define ANSWER\_PROCESS "Given string is being checked..."

#define ANSWER\_TRUE "The string matches the criteria!"

#define ANSWER\_FALSE "The string does not match the criteria :/"

#define After 'A'

#define Before 'B'

#define Cetera 'C'

#define Dawn 'D'

#define COMMAND\_PUSH "Push"

#define COMMAND\_POP "Pop"

#define COMMAND\_CHANGING "Checking stack"

#define COMMAND\_REBUILDING "Clearing stack"

#define COMMAND\_SKIPPING "Skipping"

#define COMMAND\_ERROR "Unknown"

#define COMMAND\_FINISHING "Finishing"

#define MESSAGE\_OK "Program finished successfully."

using namespace std;

class lab3 {

private:

static unsigned long currentPos; // -1 for gui

static string source;

static string nextCommand;

static string answer;

static string message;

static visible\_stack<char>\* stack;

static bool init;

static bool stackOperationalMode;

static bool considerNextAction();

static void considerMessage();

static bool unfriendlyStep();

public:

static unsigned long getCurrentPos();

static const string& getSource();

static const string& getNextCommand();

static const string& getAnswer();

static const string& getMessage();

static const visible\_stack<char>\* getStackState();

static bool isInit();

static double process(string& input, bool friendly); // no time for gui

static void friendlyStep();

};

#endif //ALG\_LAB1\_LAB3\_H

**Файл lab3.cpp:**

#include "lab3.h"

unsigned long lab3::currentPos; // -1 for gui

string lab3::source;

string lab3::nextCommand;

string lab3::answer;

string lab3::message;

visible\_stack<char>\* lab3::stack;

bool lab3::init = false;

bool lab3::stackOperationalMode;

unsigned long lab3::getCurrentPos() {

return currentPos;

}

const string& lab3::getSource() {

return source;

}

const string& lab3::getNextCommand() {

return nextCommand;

}

const string& lab3::getAnswer() {

return answer;

}

const string& lab3::getMessage() {

return message;

}

const visible\_stack<char>\* lab3::getStackState() {

return stack;

}

bool lab3::isInit() {

return init;

}

double lab3::process(string& input, bool friendly) {

source = input;

currentPos = 0;

stack = new visible\_stack<char>();

stackOperationalMode = true;

init = true;

answer = ANSWER\_PROCESS;

if (!friendly) {

bool result = true;

clock\_t sTime = clock();

for (; currentPos < source.size(); currentPos++) {

result = unfriendlyStep();

if (!result) break;

}

clock\_t eTime = clock();

if (result) {

answer = ANSWER\_TRUE;

message = MESSAGE\_OK;

} else {

answer = ANSWER\_FALSE;

considerMessage();

}

init = false;

return ((double) (eTime - sTime) / CLOCKS\_PER\_SEC ) \* 1000.0;

}

return 1L;

}

bool lab3::unfriendlyStep() {

switch (source[currentPos]) {

case After:

case Before:

if (stackOperationalMode) {

stack->push(source[currentPos]);

} else if (stack->isEmpty() || stack->pop() != source[currentPos]) {

return false;

}

break;

case Cetera:

stackOperationalMode = !stackOperationalMode;

break;

case Dawn:

if (!stack->isEmpty()) return false;

stackOperationalMode = !stackOperationalMode;

break;

default:

if (!isblank(source[currentPos])) {

return false;

}

break;

}

return true;

}

void lab3::friendlyStep() {

if (unfriendlyStep()) {

currentPos++;

if (considerNextAction()) {

message = MESSAGE\_OK;

answer = ANSWER\_TRUE;

init = false;

}

} else {

considerMessage();

answer = ANSWER\_FALSE;

init = false;

}

}

bool lab3::considerNextAction() {

if (currentPos < source.size()) {

switch (source[currentPos]) {

case After:

case Before:

if (stackOperationalMode) {

nextCommand = COMMAND\_PUSH;

} else {

nextCommand = COMMAND\_POP;

}

break;

case Cetera:

nextCommand = COMMAND\_CHANGING;

break;

case Dawn:

nextCommand = COMMAND\_REBUILDING;

break;

default:

if (isblank(source[currentPos])) {

nextCommand = COMMAND\_SKIPPING;

} else {

nextCommand = COMMAND\_ERROR;

}

break;

}

return false;

} else {

nextCommand = COMMAND\_FINISHING;

return true;

}

}

void lab3::considerMessage() {

switch (source[currentPos]) {

case After:

case Before:

case Dawn:

message = (new universal\_exception(10, currentPos, &source))->toString();

break;

default:

message = (new universal\_exception(11, currentPos, &source))->toString();

break;

}

}

**Файл visible\_stack.h:**

#ifndef ALG\_LAB1\_VISIBLE\_STACK\_H

#define ALG\_LAB1\_VISIBLE\_STACK\_H

#include <bits/stdc++.h>

#include "universal\_exception.h"

#define EMPTY\_STACK "Stack is empty!"

using namespace std;

template <typename T>

class visible\_stack {

private:

class stack\_element {

private:

T value;

stack\_element\* next;

stack\_element\* previous;

public:

stack\_element(T& value, stack\_element \*previous);

virtual ~stack\_element();

T& getValue();

void setNext(stack\_element \*next);

stack\_element \*getNext();

stack\_element \*getPrevious();

};

int size;

stack\_element\* first;

stack\_element\* last;

public:

visible\_stack();

~visible\_stack();

void push(T& element);

T pop();

string toString() const;

bool isEmpty();

const int\* getStackSize() const;

};

template<typename T>

visible\_stack<T>::stack\_element::stack\_element(T& value, visible\_stack::stack\_element\* previous) {

this->value = value;

this->previous = previous;

this->next = nullptr;

}

template<typename T>

visible\_stack<T>::stack\_element::~stack\_element() {

free(this->next);

free(this->previous);

}

template<typename T>

T& visible\_stack<T>::stack\_element::getValue() {

return value;

}

template<typename T>

void visible\_stack<T>::stack\_element::setNext(visible\_stack::stack\_element \*next) {

stack\_element::next = next;

}

template <typename T>

typename visible\_stack<T>::stack\_element \*visible\_stack<T>::stack\_element::getNext() {

return this->next;

}

template <typename T>

typename visible\_stack<T>::stack\_element \*visible\_stack<T>::stack\_element::getPrevious() {

return this->previous;

}

template<typename T>

const int\* visible\_stack<T>::getStackSize() const {

return &size;

}

template<typename T>

visible\_stack<T>::visible\_stack() {

this->size = 0;

this->first = nullptr;

this->last = nullptr;

}

template<typename T>

visible\_stack<T>::~visible\_stack() {

free(this->first);

free(this->last);

}

template<typename T>

void visible\_stack<T>::push(T& element) {

stack\_element\* SE = new stack\_element(element, last);

if (size == 0) {

first = SE;

} else {

last->setNext(SE);

}

size++;

last = SE;

}

template<typename T>

T visible\_stack<T>::pop() {

if (size == 0)

throw universal\_exception(1);

stack\_element \*decapitation = last;

T value = decapitation->getValue();

last = decapitation->getPrevious();

if (size > 1) {

last->setNext(nullptr);

} else {

first = nullptr;

}

free(decapitation);

size--;

return value;

}

template<typename T>

string visible\_stack<T>::toString() const {

string out;

if (size > 0) {

out += "╔";

for (int i = 1; i < size \* 2; i++) {

out += (i % 2 == 0) ? "╤" : "═";

}

out += "╗";

out += "\n║";

stack\_element \*current = first;

for (int j = 0; j < size-1; j++) {

out += string(1, (T) current->getValue()) + "│";

current = current->getNext();

}

out += string(1, (T) current->getValue()) + "║";

out += "\n";

out += "╚";

for (int i = 1; i < size \* 2; i++) {

out += (i % 2 == 0) ? "╧" : "═";

}

out += "╝";

} else {

out = EMPTY\_STACK;

}

return out;

}

template<typename T>

bool visible\_stack<T>::isEmpty() {

return size == 0;

}

#endif //ALG\_LAB1\_VISIBLE\_STACK\_H

**Файл universal\_exception.h:**

#ifndef ALG\_LAB1\_UNIVERSAL\_EXCEPTION\_H

#define ALG\_LAB1\_UNIVERSAL\_EXCEPTION\_H

#include <bits/stdc++.h>

#define EXIT\_ERROR\_CODE 42

#define NULL\_POS -1

#define OUTPUT\_FILE\_EXCEPTION\_CODE -1

#define OUTPUT\_FILE\_EXCEPTION\_STRING "THE EXCEPTION OCCURS WHILE CREATING OUTPUT FILE, APPLICATION TERMINATED"

#define INPUT\_FILE\_EXCEPTION\_CODE -2

#define INPUT\_FILE\_EXCEPTION\_STRING "THE EXCEPTION OCCURS WHILE READING INPUT FILE, APPLICATION TERMINATED"

#define STACK\_SIZE\_EXCEPTION\_CODE 1

#define STACK\_SIZE\_EXCEPTION\_STRING "POP METHOD INVOCED ON 0 LENGTH STACK, APPLICATION TERMINATED"

#define STACK\_MISMATCH\_EXCEPTION\_CODE 10

#define STACK\_MISMATCH\_EXCEPTION\_STRING "STACK SIZE AND CONTENTS SHOW WRONG STRING FORMAT"

#define STRANGE\_INPUT\_EXCEPTION\_CODE 11

#define STRANGE\_INPUT\_EXCEPTION\_STRING "WRONG CHARACTER IN INPUT STRING OR SOMETHING ELSE"

#define UNHANDLED\_EXCEPTION\_STRING "SOME UNHANDLED EXCEPTION HAPPENS, APPLICATION TERMINATED"

using namespace std;

class universal\_exception : public std::exception {

private:

string cause;

string\* source;

long sourcePos;

public:

const char\* what() const noexcept override;

universal\_exception(int cause\_num, long sourcePos, string\* source) noexcept(true);

explicit universal\_exception(int cause\_num) noexcept(true);

~universal\_exception() override;

string toString() const;

};

#endif //ALG\_LAB1\_UNIVERSAL\_EXCEPTION\_H

**Файл universal\_exception.cpp:**

#include "universal\_exception.h"

const char \*universal\_exception::what() const noexcept {

return toString().c\_str();

}

universal\_exception::universal\_exception(int cause\_num, long sourcePos, string\* source) noexcept(true) : std::exception() {

this->source = source;

this->sourcePos = sourcePos;

switch (cause\_num) {

case OUTPUT\_FILE\_EXCEPTION\_CODE:

this->cause = OUTPUT\_FILE\_EXCEPTION\_STRING;

break;

case INPUT\_FILE\_EXCEPTION\_CODE:

this->cause = INPUT\_FILE\_EXCEPTION\_STRING;

break;

case STACK\_SIZE\_EXCEPTION\_CODE:

this->cause = STACK\_SIZE\_EXCEPTION\_STRING;

break;

case STACK\_MISMATCH\_EXCEPTION\_CODE:

this->cause = STACK\_MISMATCH\_EXCEPTION\_STRING;

break;

case STRANGE\_INPUT\_EXCEPTION\_CODE:

this->cause = STRANGE\_INPUT\_EXCEPTION\_STRING;

break;

default:

this->cause = UNHANDLED\_EXCEPTION\_STRING;

break;

}

}

universal\_exception::universal\_exception(int cause\_num) noexcept(true) : universal\_exception(cause\_num, NULL\_POS, nullptr) {}

universal\_exception::~universal\_exception() = default;

string universal\_exception::toString() const {

string out;

if (sourcePos != NULL\_POS) {

out += \*source + "\n";

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

out += ' ';

}

out += "^";

out += "\n" + cause;

} else {

out = cause;

}

return out;

}

**Файл mainwindow.h:**

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include "lab3.h"

#include "ui\_mainwindow.h"

using namespace std;

namespace Ui {

class MainWindow;

}

class MainWindow : public QMainWindow

{

Q\_OBJECT

public:

explicit MainWindow(QWidget \*parent = 0);

~MainWindow();

private slots:

void build();

void step();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

**Файл mainwindow.cpp:**

#include "mainwindow.h"

MainWindow::MainWindow(QWidget \*parent) : QMainWindow(parent), ui(new Ui::MainWindow) {

ui->setupUi(this);

connect(ui->input\_button, SIGNAL (clicked()), this, SLOT (build()));

connect(ui->step\_button, SIGNAL (clicked()), this, SLOT (step()));

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::build() {

string input = ui->input->text().toStdString();

ui->input->setEnabled(false);

ui->input\_button->setEnabled(false);

ui->step\_button->setEnabled(true);

lab3::process(input, true);

step();

}

void MainWindow::step() {

if (lab3::isInit()) {

lab3::friendlyStep();

string definition;

definition = "CURRENT DATA STATE:\n\n\n";

definition += "Input Line:\n\n";

definition += lab3::getSource();

definition += "\n";

for (unsigned long i = 0; i < lab3::getCurrentPos() - 1; i++) {

definition += " ";

}

definition += "^\n\n\n";

definition += "Stack state:\n\n";

definition += lab3::getStackState()->toString();

definition += "\n";

definition += "Length: ";

definition += to\_string(\*lab3::getStackState()->getStackSize());

definition += "\n\n\n";

definition += "Next command: ";

definition += lab3::getNextCommand();

ui->output->setText(QString::fromStdString(definition));

string answer = lab3::getAnswer();

ui->answer->setText(QString::fromStdString(answer));

} else {

ui->input->setEnabled(true);

ui->input\_button->setEnabled(true);

ui->step\_button->setEnabled(false);

string definition;

definition = "RESULTS:\n\n\n";

definition += lab3::getMessage();

ui->output->setText(QString::fromStdString(definition));

string answer = lab3::getAnswer();

ui->answer->setText(QString::fromStdString(answer));

}

}

**Файл mainwindow.ui:**

<?xml version="1.0" encoding="UTF-8"?>

<ui version="4.0">

<class>MainWindow</class>

<widget class="QMainWindow" name="MainWindow">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>800</width>

<height>600</height>

</rect>

</property>

<property name="windowTitle">

<string>MainWindow</string>

</property>

<widget class="QWidget" name="centralWidget">

<widget class="QWidget" name="verticalLayoutWidget">

<property name="geometry">

<rect>

<x>0</x>

<y>0</y>

<width>801</width>

<height>601</height>

</rect>

</property>

<layout class="QVBoxLayout" name="verticalLayout">

<property name="leftMargin">

<number>5</number>

</property>

<property name="topMargin">

<number>5</number>

</property>

<property name="rightMargin">

<number>5</number>

</property>

<property name="bottomMargin">

<number>5</number>

</property>

<item>

<widget class="QLabel" name="name">

<property name="font">

<font>

<pointsize>14</pointsize>

<weight>75</weight>

<bold>true</bold>

</font>

</property>

<property name="text">

<string>Text Stack Reader</string>

</property>

<property name="alignment">

<set>Qt::AlignCenter</set>

</property>

</widget>

</item>

<item>

<layout class="QHBoxLayout" name="header\_layout">

<item>

<widget class="QPushButton" name="input\_button">

<property name="sizePolicy">

<sizepolicy hsizetype="Preferred" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="text">

<string>Read!</string>

</property>

</widget>

</item>

<item>

<widget class="QLineEdit" name="input">

<property name="sizePolicy">

<sizepolicy hsizetype="Expanding" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="font">

<font>

<family>Ubuntu Mono</family>

</font>

</property>

</widget>

</item>

</layout>

</item>

<item>

<widget class="QLabel" name="output">

<property name="sizePolicy">

<sizepolicy hsizetype="Expanding" vsizetype="Expanding">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="font">

<font>

<family>Ubuntu Mono</family>

</font>

</property>

<property name="alignment">

<set>Qt::AlignLeading|Qt::AlignLeft|Qt::AlignTop</set>

</property>

</widget>

</item>

<item>

<layout class="QHBoxLayout" name="menu\_layout">

<item>

<widget class="QLabel" name="answer">

<property name="sizePolicy">

<sizepolicy hsizetype="Expanding" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="font">

<font>

<family>Ubuntu Mono</family>

</font>

</property>

<property name="alignment">

<set>Qt::AlignCenter</set>

</property>

</widget>

</item>

<item>

<widget class="QPushButton" name="step\_button">

<property name="enabled">

<bool>false</bool>

</property>

<property name="sizePolicy">

<sizepolicy hsizetype="Preferred" vsizetype="Preferred">

<horstretch>0</horstretch>

<verstretch>0</verstretch>

</sizepolicy>

</property>

<property name="text">

<string>Step</string>

</property>

</widget>

</item>

</layout>

</item>

</layout>

</widget>

</widget>

</widget>

<layoutdefault spacing="6" margin="11"/>

<resources/>

<connections/>

</ui>

**Файл CMakeLists.txt:**

cmake\_minimum\_required(VERSION 3.12)

project(alg\_lab3)

set(CMAKE\_CXX\_STANDARD 17)

add\_executable(alg\_lab3 main\_console.cpp lab3.cpp lab3.h universal\_exception.cpp universal\_exception.h visible\_stack.h)

**Файл lab3.pro:**

#-------------------------------------------------

#

# Project created by QtCreator 2019-10-22T14:34:31

#

#-------------------------------------------------

QT += core gui

greaterThan(QT\_MAJOR\_VERSION, 4): QT += widgets

TARGET = lab3

TEMPLATE = app

# The following define makes your compiler emit warnings if you use

# any feature of Qt which has been marked as deprecated (the exact warnings

# depend on your compiler). Please consult the documentation of the

# deprecated API in order to know how to port your code away from it.

DEFINES += QT\_DEPRECATED\_WARNINGS

# You can also make your code fail to compile if you use deprecated APIs.

# In order to do so, uncomment the following line.

# You can also select to disable deprecated APIs only up to a certain version of Qt.

#DEFINES += QT\_DISABLE\_DEPRECATED\_BEFORE=0x060000 # disables all the APIs deprecated before Qt 6.0.0

SOURCES += \

main\_gui.cpp \

mainwindow.cpp \

universal\_exception.cpp \

lab3.cpp

HEADERS += \

mainwindow.h \

visible\_stack.h \

lab3.h \

universal\_exception.h

FORMS += \

mainwindow.ui