Додаток Б Тексти програмного коду

*студента групи ІП-81 І курсу*

*Касьяненка М.В.*

(Вид носія даних)

(Обсяг програми (документа), арк., Кб)

*14 арк, 36 Кб*

(Найменування програми (документа))

*Тексти програмного коду програмного забезпечення розв’язку систем лінійних рівнянь*

MyForm.h

#pragma once

#include <string>

#include <fstream>

#include <msclr\marshal\_cppstd.h>;

#include "before\_solving.h"

#include "system.h"

//#include "stdafx.h"

systema the\_system;

namespace Project {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

//friend void parce\_system();

/// <summary>

/// Сводка для MyForm

/// </summary>

public ref class MyForm : public System::Windows::Forms::Form

{

public:

MyForm(void)

{

InitializeComponent();

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~MyForm()

{

if (components)

{

delete components;

}

}

private: System::Windows::Forms::RichTextBox^ richTextBox1;

private: System::Windows::Forms::Label^ label1;

private: System::Windows::Forms::ComboBox^ comboBox1;

private: System::Windows::Forms::Button^ button1;

private: System::Windows::Forms::Button^ button2;

private: System::Windows::Forms::RichTextBox^ richTextBox2;

private: System::Windows::Forms::Label^ label3;

private: System::Windows::Forms::DataVisualization::Charting::Chart^ chart1;

private: System::Windows::Forms::TextBox^ textBox1;

private: System::Windows::Forms::Button^ button3;

protected:

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

void InitializeComponent(void)

{

System::Windows::Forms::DataVisualization::Charting::ChartArea^ chartArea1 = (gcnew System::Windows::Forms::DataVisualization::Charting::ChartArea());

System::Windows::Forms::DataVisualization::Charting::Series^ series1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

System::Windows::Forms::DataVisualization::Charting::Series^ series2 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

System::ComponentModel::ComponentResourceManager^ resources = (gcnew System::ComponentModel::ComponentResourceManager(MyForm::typeid));

this->richTextBox1 = (gcnew System::Windows::Forms::RichTextBox());

this->label1 = (gcnew System::Windows::Forms::Label());

this->comboBox1 = (gcnew System::Windows::Forms::ComboBox());

this->button1 = (gcnew System::Windows::Forms::Button());

this->button2 = (gcnew System::Windows::Forms::Button());

this->richTextBox2 = (gcnew System::Windows::Forms::RichTextBox());

this->label3 = (gcnew System::Windows::Forms::Label());

this->chart1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Chart());

this->textBox1 = (gcnew System::Windows::Forms::TextBox());

this->button3 = (gcnew System::Windows::Forms::Button());

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart1))->BeginInit();

this->SuspendLayout();

this->richTextBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 15.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->richTextBox1->Location = System::Drawing::Point(39, 145);

this->richTextBox1->Name = L"richTextBox1";

this->richTextBox1->Size = System::Drawing::Size(308, 240);

this->richTextBox1->TabIndex = 0;

this->richTextBox1->Text = L"";

this->label1->Anchor = System::Windows::Forms::AnchorStyles::Top;

this->label1->AutoSize = true;

this->label1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label1->Location = System::Drawing::Point(33, 93);

this->label1->Name = L"label1";

this->label1->Size = System::Drawing::Size(281, 31);

this->label1->TabIndex = 1;

this->label1->Text = L"Input Your System Here:";

this->comboBox1->FlatStyle = System::Windows::Forms::FlatStyle::System;

this->comboBox1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->comboBox1->FormattingEnabled = true;

this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(4) {

L"Метод Гаусса", L"Метод Жордана-Гаусса", L"Метод Вращения",

L"Графический Метод"

});

this->comboBox1->Location = System::Drawing::Point(167, 24);

this->comboBox1->Name = L"comboBox1";

this->comboBox1->Size = System::Drawing::Size(382, 39);

this->comboBox1->TabIndex = 2;

this->comboBox1->Text = L"Choose Your Preferred Method";

this->comboBox1->SelectedIndexChanged += gcnew System::EventHandler(this, &MyForm::comboBox1\_SelectedIndexChanged);

this->button1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button1->Location = System::Drawing::Point(109, 402);

this->button1->Name = L"button1";

this->button1->Size = System::Drawing::Size(178, 53);

this->button1->TabIndex = 4;

this->button1->Text = L"Check Input";

this->button1->UseVisualStyleBackColor = true;

this->button1->Click += gcnew System::EventHandler(this, &MyForm::button1\_Click);

this->button2->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button2->Location = System::Drawing::Point(403, 404);

this->button2->Name = L"button2";

this->button2->Size = System::Drawing::Size(228, 49);

this->button2->TabIndex = 5;

this->button2->Text = L"Solve The System ";

this->button2->UseVisualStyleBackColor = true;

this->button2->Click += gcnew System::EventHandler(this, &MyForm::button2\_Click);

this->richTextBox2->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 14.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->richTextBox2->Location = System::Drawing::Point(373, 145);

this->richTextBox2->Name = L"richTextBox2";

this->richTextBox2->Size = System::Drawing::Size(308, 240);

this->richTextBox2->TabIndex = 6;

this->richTextBox2->Text = L"";

this->label3->AutoSize = true;

this->label3->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->label3->Location = System::Drawing::Point(367, 93);

this->label3->Name = L"label3";

this->label3->Size = System::Drawing::Size(97, 31);

this->label3->TabIndex = 7;

this->label3->Text = L"Output:";

chartArea1->Name = L"ChartArea1";

this->chart1->ChartAreas->Add(chartArea1);

this->chart1->Location = System::Drawing::Point(373, 145);

this->chart1->Name = L"chart1";

series1->ChartArea = L"ChartArea1";

series1->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Line;

series1->Name = L"line1";

series2->ChartArea = L"ChartArea1";

series2->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Line;

series2->Name = L"line2";

this->chart1->Series->Add(series1);

this->chart1->Series->Add(series2);

this->chart1->Size = System::Drawing::Size(308, 240);

this->chart1->TabIndex = 8;

this->chart1->Text = L"chart1";

this->chart1->Visible = false;

this->chart1->Click += gcnew System::EventHandler(this, &MyForm::chart1\_Click);

this->textBox1->Font = (gcnew System::Drawing::Font(L"Times New Roman", 15.75F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->textBox1->Location = System::Drawing::Point(109, 502);

this->textBox1->Name = L"textBox1";

this->textBox1->Size = System::Drawing::Size(238, 32);

this->textBox1->TabIndex = 9;

this->textBox1->Text = L"filename";

this->button3->Font = (gcnew System::Drawing::Font(L"Times New Roman", 20.25F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(204)));

this->button3->Location = System::Drawing::Point(399, 489);

this->button3->Name = L"button3";

this->button3->Size = System::Drawing::Size(150, 51);

this->button3->TabIndex = 10;

this->button3->Text = L"Save to file";

this->button3->UseVisualStyleBackColor = true;

this->button3->Click += gcnew System::EventHandler(this, &MyForm::button3\_Click);

this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);

this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;

this->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;

this->ClientSize = System::Drawing::Size(727, 583);

this->Controls->Add(this->button3);

this->Controls->Add(this->textBox1);

this->Controls->Add(this->chart1);

this->Controls->Add(this->label3);

this->Controls->Add(this->richTextBox2);

this->Controls->Add(this->button2);

this->Controls->Add(this->button1);

this->Controls->Add(this->comboBox1);

this->Controls->Add(this->label1);

this->Controls->Add(this->richTextBox1);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::Fixed3D;

this->Icon = (cli::safe\_cast<System::Drawing::Icon^>(resources->GetObject(L"$this.Icon")));

this->ImeMode = System::Windows::Forms::ImeMode::Disable;

this->Name = L"MyForm";

this->Text = L"MyForm";

this->Load += gcnew System::EventHandler(this, &MyForm::MyForm\_Load);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->chart1))->EndInit();

this->ResumeLayout(false);

this->PerformLayout();

}

#pragma endregion

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void comboBox1\_SelectedIndexChanged(System::Object^ sender, System::EventArgs^ e) {

}

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

msclr::interop::marshal\_context context;

std::string str = context.marshal\_as<std::string>(richTextBox1->Text);

int how\_many = how\_many\_wariables(str);

int aquations = how\_many\_equations(str);

System::String^ managedString;

if (how\_many == aquations) {

managedString += how\_many;

managedString += " variables\n";

richTextBox2->Text = managedString;

the\_system.create\_system(str);

richTextBox2->Text += the\_system.matrix();

}

else {

richTextBox2->Text = "Number of equations is not equel to number of variables\n";

}

}

private: System::Void button2\_Click(System::Object^ sender, System::EventArgs^ e) {

string res;

msclr::interop::marshal\_context context;

switch (comboBox1->SelectedIndex) {

case -1:

MessageBox::Show("Choose the solving method!", "error");

break;

case 0:

res = the\_system.hauss\_method();

richTextBox2->Text += "\nThe Solution:\n";

richTextBox2->Text += context.marshal\_as<System::String^>(res); break;

case 1:

res = the\_system.jordan\_hauss\_method();

richTextBox2->Text += "\nThe Solution:\n";

richTextBox2->Text += context.marshal\_as<System::String^>(res);

case 2:

res = the\_system.rotation\_method();

richTextBox2->Text += "\nThe Solution:\n";

richTextBox2->Text += context.marshal\_as<System::String^>(res);

break;

case 3:

if (the\_system.number\_of\_variables() != 2) {

MessageBox::Show("This Method is Inapropriate", "error");

break;

}

else {

//strin

double kx1, b1, kx2, b2;

the\_system.graphical\_method(kx1, b1, kx2, b2);

int resultx = the\_system.hauss\_method(1);

chart1->Visible = true;

for (double i = resultx-5; i < resultx+5; i += (double)1) {

chart1->Series["line1"]->Points->AddXY(i, kx1\*i + b1);

chart1->Series["line2"]->Points->AddXY(i, kx2\*i + b2);

}

break;

}

};

}

private: System::Void chart1\_Click(System::Object^ sender, System::EventArgs^ e) {

chart1->Visible = false;

}

private: System::Void button3\_Click(System::Object^ sender, System::EventArgs^ e) {

msclr::interop::marshal\_context context;

std::string fname = context.marshal\_as<std::string>(textBox1->Text);

if (fname.find(".txt") == string::npos) {

fname += ".txt";

}

std::string text = context.marshal\_as<std::string>(richTextBox2->Text);

ofstream the\_file(fname, ios::trunc);

the\_file << text;

MessageBox::Show("Saved succesfully!!", "YEEEEyy");

}

};

}

Before\_solving.h

#pragma once

#include <string>

using namespace std;

int how\_many\_equations(string raw\_system);

int how\_many\_wariables(string raw\_system);

bool is\_letter(char letter);

bool is\_number(char symbol);

void upper\_triangle\_matrix(double \*\*arr, int length, int height);

int zero\_row(double \*\*arr, int height);

Before\_solving.cpp

#include <string>

#include <iostream>

#include <iomanip>

using namespace std;

bool is\_letter(char letter) {

if (((letter >= 'a') && (letter <= 'z')) || ((letter >= 'A') && (letter <= 'Z'))) {

return true;

}

return false;

}

bool is\_number(char symbol) {

if ((symbol >= '0') && (symbol <= '9'))

return true;

return false;

}

int how\_many\_wariables(string raw\_system) {

string variables;

int how\_many = 0;

for (int i = 0; i < raw\_system.size(); i++) {

if ((variables.find(raw\_system[i]) == string::npos) && is\_letter(raw\_system[i])) {

variables += raw\_system[i];

how\_many++;

}

}

return how\_many;

}

void upper\_triangle\_matrix(double \*\*arr, int length, int height) {

double \*buff;

double multiplication;

for (int row = 0; row < height-1; row++) {

for (int i = row; i < height; i++) {

if (arr[i][row] != 0) {

buff = arr[row];

arr[row] = arr[i];

arr[i] = buff;

break;

}

}

multiplication = arr[row][row];

for (int i = row; i < (length); i++) {

arr[row][i] /= multiplication;

}

for (int i = row + 1; i < height; i++) {

multiplication = arr[i][row];

for (int j = row; j < length; j++) {

arr[i][j] -= (arr[row][j])\*multiplication;

}

}

}

//system("pause");

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

for (int j = 0; j < length; j++) {

cout << setw(8) << arr[i][j];

}

cout << endl;

}

}

int zero\_row(double \*\*arr, int height) {

int j;

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

for (j = 0; j < height; j++) {

if (arr[i][j] != 0) break;

}

if (j == height) return i;

}

return -1;

}

int get\_determinant(double \*\*arr, int size) {

double \*\*copy = new double\*[size];

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

copy[i] = new double[size];

}

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

for (int j = 0; j < size; j++)

copy[i][j] = arr[i][j];

//coppied code fron upper triangle matrix;

double determinant = 1;

double \*buff;

double multiplication;

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

for (int i = row; i < size; i++) {

if (copy[i][row] != 0) {

buff = copy[row];

copy[row] = copy[i];

copy[i] = buff;

break;

}

}

multiplication = copy[row][row];

determinant \*= multiplication;

for (int i = row; i < (size); i++) {

copy[row][i] /= multiplication;

}

for (int i = row + 1; i < size; i++) {

multiplication = copy[i][row];

for (int j = row; j < size; j++) {

copy[i][j] -= (copy[row][j])\*multiplication;

}

}

}

multiplication \*= copy[size - 1][size - 1];

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

delete[] copy[i];

}

delete[]copy;

return determinant;

}

int how\_many\_equations(string raw\_system) {

int counter = 0;

if (raw\_system[raw\_system.size() - 1] != '\n') {

raw\_system += '\n';

}

for (int i = 0; i < raw\_system.size(); i++) {

if (raw\_system[i] == '\n') {

counter++;

}

}

return counter;

}

Systema.h

#include <string>

#pragma once

using namespace std;

class systema {

private:

string variables;

double \*\*arr;

int height;

int length;

public:

int hauss\_method(int k);

int hauss\_single\_solution(int k);

string hauss\_method();

string hauss\_single\_solution();

string jordan\_hauss\_method();

string jordan\_hauss\_single\_solution();

string rotation\_method();

void graphical\_method(double &kx1, double &b1, double &kx2, double &b2);

void create\_system(string str);

System::String^ matrix();

int number\_of\_variables();

};

System.cpp

#include "system.h"

#include "before\_solving.h"

#include <stdlib.h>

#include <string>

#include <msclr\marshal\_cppstd.h>

//#include "system.h"

//#include "MyForm.h"

using namespace std;

void systema::create\_system(string str) {

height = how\_many\_wariables(str);

length = height + 1;

// create the array of zeroes

if (arr != nullptr) delete[] arr;

arr = new double\*[height];

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

arr[i] = new double[length];

}

for (int i = 0; i < height; i++)

for (int j = 0; j < length; j++)

arr[i][j] = 0;

//getting the list of variables

variables = "";

for (int i = 0; i < str.size(); i++) {

if ((variables.find(str[i]) == string::npos) && is\_letter(str[i])) {

variables += str[i];

}

}

//making matrix

double cnumber = 0; //Number we are going to put into the matrix

int row = 0; // Number of te row in matrix;

//string help;

bool plus = true; // shows is number positive or negative

bool coma = false; // shows presence of coma

double multiply\_coma = 1;

str += "\n";

for (int i = 0; i < str.size(); i++) {

if (is\_number(str[i])) {

if (!coma) {

cnumber \*= 10;

cnumber += (int)str[i] - 48;

}

else{

multiply\_coma \*= 0.1;

cnumber += ((int)str[i] - 48) \* multiply\_coma;

}

}

else if ((is\_letter(str[i])) && (variables.find\_first\_of(str[i]) != string::npos)) {

if (cnumber == 0) {

cnumber = 1;

}

arr[row][variables.find\_first\_of(str[i])] = plus ? cnumber : cnumber \* (-1);

cnumber = 0;

plus = true;

coma = false;

multiply\_coma = 1;

}

else if (str[i] == '-') {

plus = false;

}

else if ((str[i] == '.')||(str[i] == ',')) {

coma = true;

}

else if (str[i] == '\n') {

arr[row][length - 1] = plus ? cnumber : cnumber \* (-1);

row++;

cnumber = 0;

plus = true;

coma = false;

multiply\_coma = 1;

}

}

}

System::String^ systema::matrix() {

System::String^ the\_str;

string help;

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

for (int j = 0; j < length; j++) {

help = to\_string(arr[i][j]);

for (int i = help.size() - 1; i >= 0; i--) {

if (help[i] != '0') break;

else {

help.erase(i);

}

}

if(help[help.size()-1] == '.') help.erase(help.size() - 1);

while(help.size() <= 8){

help += " ";

}

msclr::interop::marshal\_context context;

the\_str += context.marshal\_as<System::String^>(help);

//the\_str += marshal\_as<String^>(help);

//the\_str += arr[i][j];

//the\_str += " ";

}

the\_str += "\n";

}

return the\_str;

}

int systema::number\_of\_variables() {

return variables.size();

}