**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Сибирский государственный университет науки и технологий**

**имени академика М.Ф. Решетнёва»**

ИИТК/09.03.01/\Информатика и вычислительная техника/Автоматизированные системы обработки информации и управления

институт/факультет/подразделение

Информатики и вычислительной техники

кафедра

**Определенные интегралы**

Преподаватель \_\_14.05.2022\_\_\_\_\_ Семенкина М.Е.

подпись, дата инициалы, фамилия

Обучающиеся \_\_БИА21-01 \_ \_\_\_14.05.2022\_\_\_

номер группы подпись, дата Р. В. Сухих,

А.С.Мкртчян,

Е.В. Карапузин,

П.А. Хурасев.

инициалы, фамилии

Красноярск 2022

Введение

В нашей работе рассматривается стандартная и хорошо всем знакомая задача – решение дифференциальных уравнений.

На данный момент существует множество решений дифференциальных уравнений, мы выбрали лишь четыре из них: Метод Рунге-Кутта 2 порядка, Рунге-Кутта 4 порядка и Рунге-Кутта Фельберга.

В программе вы можете выбрать нужные вам методы для решения и посмотреть результат вычислений для каждого шага.

Для вычислений дифференциальных уравнений необходимы – «Шаг», границы интегрирования и сама формула.

Цели

* Изучить различные способы вычисления дифференциальных уравнений.
* Выделить четыре наиболее актуальных способа решения.
* Создать программу, использующую эти методы, которая поможет пользователю получить решение дифференциальных уравнений любым из выбранных методов.

Ход работы

**Функционал кнопки «Рассчитать» представлен ниже:**

private: System::Void panel1\_Click(System::Object^ sender, System::EventArgs^ e) {

bool f = true;

richTextBox1->Text += L"Рассчёт начался\n";

for (int i = 0; i < 4; i++) {

chart1->Series[i]->Points->Clear(); //Очистка графика

}

dataGridView1->Rows->Clear(); //Очистка таблиц

dataGridView3->Rows->Clear();

dataGridView4->Rows->Clear();

dataGridView5->Rows->Clear();

progressBar1->Value = 0; //Очистка прогресса

marshal\_context^ marshal = gcnew marshal\_context();

if (textBox3->Text != "") accuracy = atof(marshal->marshal\_as<const char\*>(textBox3->Text));

else f = false;

if (textBox1->Text != "") x0 = atof(marshal->marshal\_as<const char\*>(textBox1->Text));

else f = false;

if (textBox2->Text != "") y0 = atof(marshal->marshal\_as<const char\*>(textBox2->Text));

else f = false;

int maximum\_progress = 0;

if (Euler) maximum\_progress += 25;

if (RungeKutta2) maximum\_progress += 25;

if (RungeKutta4) maximum\_progress += 25;

if (RungeKuttaFehlberg) maximum\_progress += 25;

progressBar1->Maximum = maximum\_progress;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

if (f) {

result = y0;

if (Euler) {

for (int i = 0; i < 11; i++)

{

double x = 0.1 \* i;

if (checkBox5->Checked)

result = ODE\_Euler(UserFunc, x0, result, accuracy, x);

else

result = ODE\_Euler(Func, x0, result, accuracy, x);

if (i % 5 == 0) {

chart1->Series[0]->Points->Add(x, result);

}

dataGridView1->Rows->Add();

dataGridView1->Rows[i]->Cells[0]->Value = x;

dataGridView1->Rows[i]->Cells[1]->Value = result;

x0 = x;

progressBar1->Value += 1;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

progressBar1->Value += 14;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

result = y0;

if (RungeKutta2) {

for (int i = 0; i < 11; i++)

{

double x = 0.1 \* i;

if (checkBox5->Checked)

result = ODE\_RungeKutta2(UserFunc, x0, result, accuracy, x);

else

result = ODE\_RungeKutta2(Func, x0, result, accuracy, x);

if (i % 5 == 0) {

chart1->Series[1]->Points->Add(x, result);

}

dataGridView3->Rows->Add();

dataGridView3->Rows[i]->Cells[0]->Value = x;

dataGridView3->Rows[i]->Cells[1]->Value = result;

x0 = x;

progressBar1->Value += 1;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

progressBar1->Value += 14;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

result = y0;

if (RungeKutta4) {

for (int i = 0; i < 11; i++)

{

double x = 0.1 \* i;

if (checkBox5->Checked)

result = ODE\_RungeKutta4(UserFunc, x0, result, accuracy, x);

else

result = ODE\_RungeKutta4(Func, x0, result, accuracy, x);

if (i % 5 == 0) {

chart1->Series[2]->Points->Add(x, result);

}

dataGridView4->Rows->Add();

dataGridView4->Rows[i]->Cells[0]->Value = x;

dataGridView4->Rows[i]->Cells[1]->Value = result;

x0 = x;

progressBar1->Value += 1;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

progressBar1->Value += 14;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

result = y0;

if (RungeKuttaFehlberg) {

for (int i = 0; i < 11; i++)

{

double x = 0.1 \* i;

if (checkBox5->Checked)

result = ODE\_RungeKuttaFehlberg(UserFunc, x0, result, x, accuracy, 1e-8);

else

result = ODE\_RungeKuttaFehlberg(Func, x0, result, x, accuracy, 1e-8);

if (i % 5 == 0) {

chart1->Series[3]->Points->Add(x, result);

}

dataGridView5->Rows->Add();

dataGridView5->Rows[i]->Cells[0]->Value = x;

dataGridView5->Rows[i]->Cells[1]->Value = result;

x0 = x;

progressBar1->Value += 1;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

progressBar1->Value += 14;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

label9->Text = "100%";

}

else {

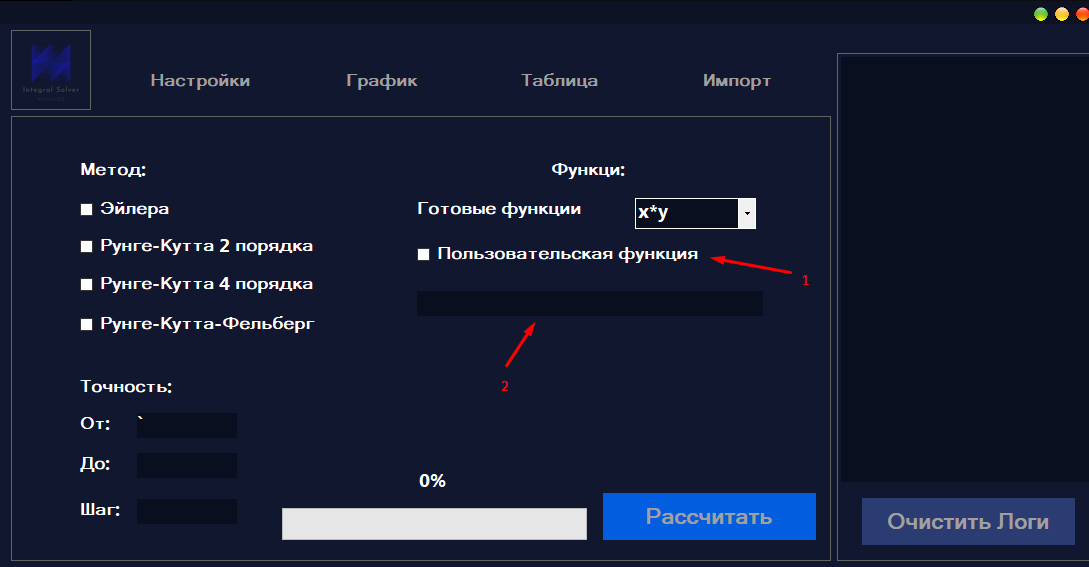
richTextBox1->Text += L"Ошибка, не заполненый все формы!\n";

}

richTextBox1->Text += L"Рассчёт окончен\n";

}

«bool f = true;» - отвечает за корректность заполненных данных.

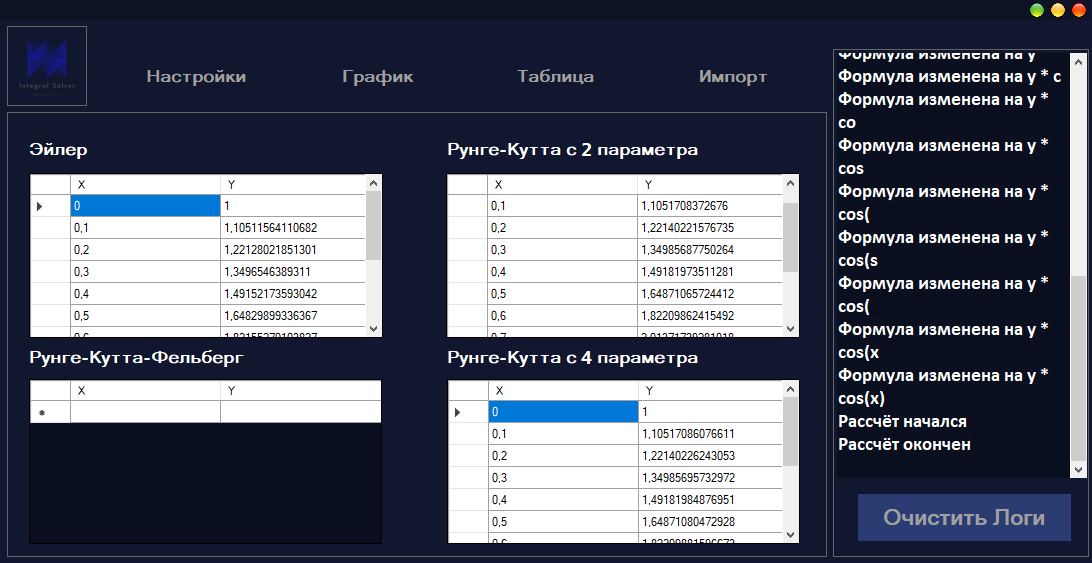


*Если человек выбрал пользовательскую функцию и поставил галочку под цифрой 1, то он может ввести свою формулу под цифрой 2.* *Например: y \* cos(x).Тогда все методы будут решать именно это уравнение.Если пользовательская функция не выбрана, то график будет строиться по готовой выбранной функции.*

dataGridView1->Rows->Add(); - добавляет новую строку.

dataGridView1->Rows[i]->Cells[0]->Value = x; - заполнение по Ox

dataGridView1->Rows[i]->Cells[1]->Value = result; - заполнение по Oy, y – результат решения дифференциального уравнения. (Скриншот 2)



*Скриншот №2*

Аналогичный принцип в методах решения дифференциальных уравнений - Ругне-Кутта 2 порядка, Ругне-Кутта 4 порядка и Ругне-Кутта-Фельберг.

**Реализация кнопки «Пользовательская функция» представлена ниже:**

private: System::Void checkBox5\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {  
richTextBox1->Text += L"Выбрана пользовательская функция\n";  
if (checkBox5->Checked) {  
MessageBox::Show("Пользовательская функция недоступна для метода Рунге-Кутта-Фельберга", "Внимание!", MessageBoxButtons::OK, MessageBoxIcon::Warning);  
richTextBox1->Text += L"!!!Пользовательская функция недоступна для метода Рунге-Кутта-Фельберга!!!\n";  
checkBox4->Checked = false;  
checkBox4->Enabled = false;  
}  
else {  
checkBox4->Checked = true;  
checkBox4->Enabled = true;  
}  
}

Функция отвечает за блокировку метода Рунге-Кутта-Фельберга, ввиду того, что данный метод недоступен для решения пользовательской функции.

Реализация кнопки выгрузки данных:

private: System::Void panel2\_Click(System::Object^ sender, System::EventArgs^ e) {

richTextBox1->Text += L"Началась выгрузка данных\n";

marshal\_context^ marshal = gcnew marshal\_context();

if (checkBox10->Checked) {

Sheet\* sheet1 = Data->addSheet(L"Euler");

sheet1->writeStr(1, 0, L"X");

sheet1->writeStr(1, 1, L"Y");

for (int i = 0; i < dataGridView1->Rows->Count - 1; i++) {

std::string buffer;

buffer = marshal->marshal\_as<const char\*>(dataGridView1->Rows[i]->Cells[0]->Value->ToString());

std::wstring buffer1;

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 0, buffer1.c\_str());

buffer1 = L"";

buffer = marshal->marshal\_as<const char\*>(dataGridView1->Rows[i]->Cells[1]->Value->ToString());

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 1, buffer1.c\_str());

}

richTextBox1->Text += L"Данные Метода Эйлера успешно выгружены!\n";

}

if (checkBox9->Checked) {

Sheet\* sheet1 = Data->addSheet(L"RungeKutta2");

sheet1->writeStr(1, 0, L"X");

sheet1->writeStr(1, 1, L"Y");

for (int i = 0; i < dataGridView3->Rows->Count - 1; i++) {

std::string buffer;

buffer = marshal->marshal\_as<const char\*>(dataGridView3->Rows[i]->Cells[0]->Value->ToString());

std::wstring buffer1;

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 0, buffer1.c\_str());

buffer1 = L"";

buffer = marshal->marshal\_as<const char\*>(dataGridView3->Rows[i]->Cells[1]->Value->ToString());

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 1, buffer1.c\_str());

}

richTextBox1->Text += L"Данные Метода Рунге-Кутта 2 порядка успешно выгружены!\n";

}

if (checkBox8->Checked) {

Sheet\* sheet1 = Data->addSheet(L"RungeKutta4");

sheet1->writeStr(1, 0, L"X");

sheet1->writeStr(1, 1, L"Y");

for (int i = 0; i < dataGridView4->Rows->Count - 1; i++) {

std::string buffer;

buffer = marshal->marshal\_as<const char\*>(dataGridView4->Rows[i]->Cells[0]->Value->ToString());

std::wstring buffer1;

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 0, buffer1.c\_str());

buffer1 = L"";

buffer = marshal->marshal\_as<const char\*>(dataGridView4->Rows[i]->Cells[1]->Value->ToString());

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 1, buffer1.c\_str());

}

richTextBox1->Text += L"Данные Метода Рунге-Кутта 4 порядка успешно выгружены!\n";

}

if (checkBox7->Checked) {

Sheet\* sheet1 = Data->addSheet(L"RungeKuttaFehlberg");

sheet1->writeStr(1, 0, L"X");

sheet1->writeStr(1, 1, L"Y");

for (int i = 0; i < dataGridView5->Rows->Count - 1; i++) {

std::string buffer;

buffer = marshal->marshal\_as<const char\*>(dataGridView5->Rows[i]->Cells[0]->Value->ToString());

std::wstring buffer1;

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 0, buffer1.c\_str());

buffer1 = L"";

buffer = marshal->marshal\_as<const char\*>(dataGridView5->Rows[i]->Cells[1]->Value->ToString());

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 1, buffer1.c\_str());

}

richTextBox1->Text += L"Данные Метода Рунге-Кутта-Фельберга порядка успешно выгружены!\n";

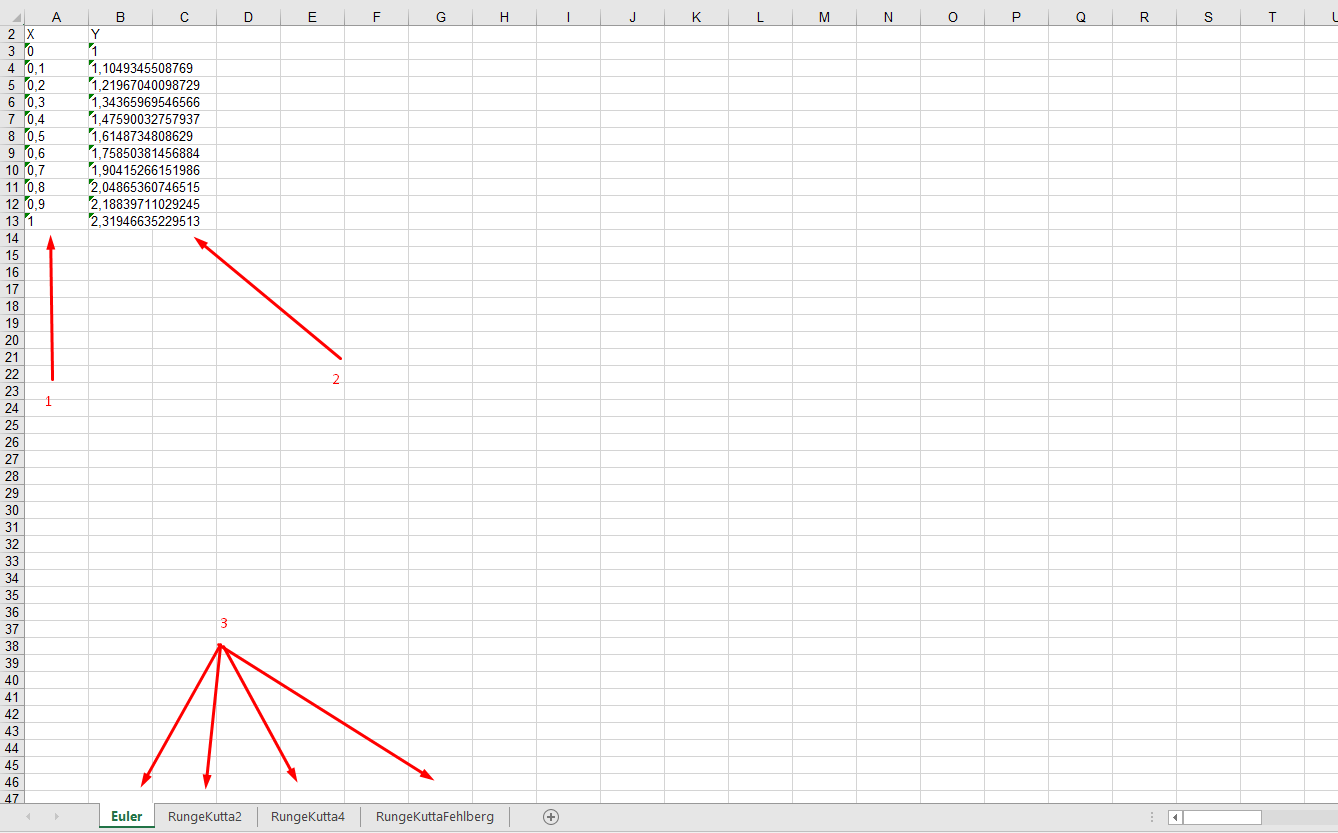
}

Data->save(L"Data.xls");

Data->release();

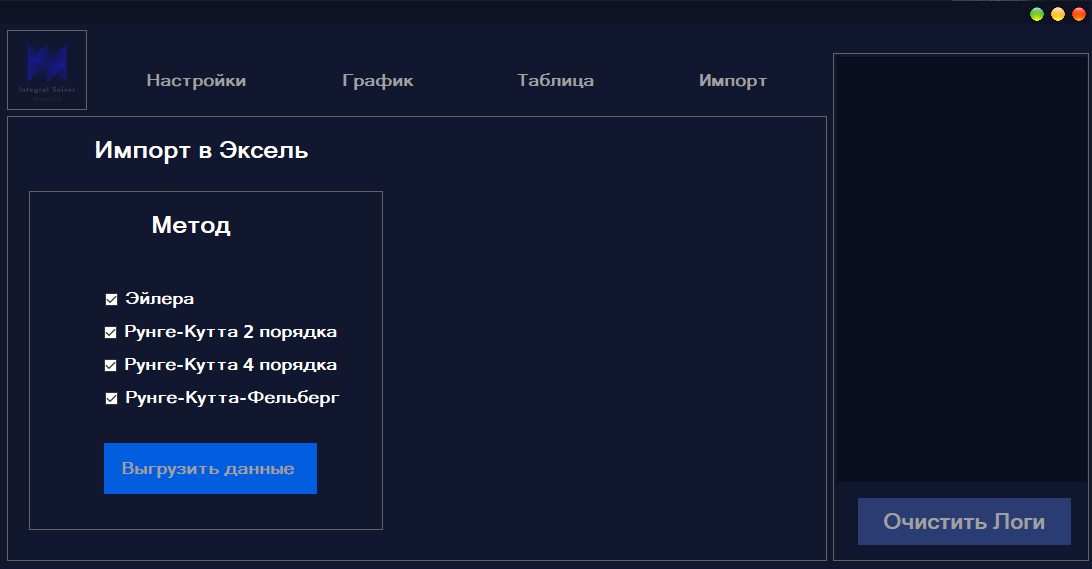
richTextBox1->Text += L"Выгрузка данных завершена!\n";

}



*Под цифрой 1- определенный интеграл со значения «от» и «до». 2 – результаты вычислений.Под цифрой 3 – результаты каждого метода, расположенные на отдельном листе файла Exel.Аналогично выгрузка работает для методов Рунге-Кутта 2 порядка, Рунге-Кутта 4 порядка и Рунге-Кутта-Фельберга.*

Exel-файл создается в папке с кодом проекта с названием «Data.xls».В нём будет находиться 4 листа с названиями метода, в них будут находиться результаты вычислений каждого метода, выбранного галочкой (Скриншот 3).



*Скриншот №3*

**Функция, устанавливающая размер окна при загрузке 1100 x 570 Пикселей:**

private: System::Void IntegralSolver\_Load(System::Object^ sender, System::EventArgs^ e) {  
this->Size = System::Drawing::Size(1100, 570);  
}

**Реализация кнопки «Закрыть»**

private: System::Void pictureBox2\_Click(System::Object^ sender, System::EventArgs^ e) {  
Close();  
}

**Реализация кнопки «Свернуть»**

private: System::Void pictureBox4\_Click(System::Object^ sender, System::EventArgs^ e) {  
this->WindowState = FormWindowState::Minimized;  
}

**Реализация перетаскивания приложения на экране:**

private: System::Void Panel\_MouseDown(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {  
this->dragging = true;  
this->offset = Point(e->X, e->Y);  
}  
private: bool dragging;  
private: Point offset;  
private: System::Void Shell\_Load(System::Object^ sender, System::EventArgs^ e) {  
this->dragging = false;  
}  
private: System::Void Panel\_MouseMove(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {  
if (this->dragging) {  
Point currentScreenPos = PointToScreen(e->Location);  
Location = Point(currentScreenPos.X - this->offset.X, currentScreenPos.Y - this->offset.Y);  
}  
};  
  
private: System::Void Panel\_MouseUp(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {  
this->dragging = false;  
}

**Включение анимации наведения курсора на вкладку:**

}  
private: System::Void panel5\_MouseLeave(System::Object^ sender, System::EventArgs^ e) {  
panel5->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(17)), static\_cast<System::Int32>(static\_cast<System::Byte>(23)),  
static\_cast<System::Int32>(static\_cast<System::Byte>(46)));  
back1 = true;  
timer1->Enabled = true;  
}

**Анимация наведения курсора на вкладку:**

private: System::Void timer1\_Tick(System::Object^ sender, System::EventArgs^ e) {  
if (back1) {  
if (r1 != 17) r1+=2;  
if (g1 != 23) g1-=19;  
if (b1 != 46) b1-=15;  
}  
else {  
if (r1 != 15) r1-=2;  
if (g1 != 99) g1+=19;  
if (b1 != 196) b1+=15;  
}  
panel6->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(r1)), static\_cast<System::Int32>(static\_cast<System::Byte>(g1)),  
static\_cast<System::Int32>(static\_cast<System::Byte>(b1)));  
if (r1 == 15 && g1 == 99 && b1 == 196) timer1->Enabled = false;  
if (r1 == 17 && g1 == 23 && b1 == 46) timer1->Enabled = false;  
}

**Набор готовых функций – дифференциальных уравнений:**

double Func(double x, double y)  
{  
if (IntegralSolver::comboBox1->Text == "x \* y")  
return x \* y;  
if (IntegralSolver::comboBox1->Text == "y \* cos(x)")  
return y \* cos(x);  
if (IntegralSolver::comboBox1->Text == "sin(y) \* cos(x)")  
return sin(y) \* cos(x);  
if (IntegralSolver::comboBox1->Text == "1 / (x \* y - 1)")  
return 1 / (x \* y - 1);  
if (IntegralSolver::comboBox1->Text == "x \* x / (y \* y)")  
return x \* x / (y \* y);  
}

**Реализация интегрирования:**

double dx(double x, double y, double z)  
{  
return 10.0 \* (y - x);  
}  
  
double dy(double x, double y, double z)  
{  
return x \* (28.0 - z) - y;  
}  
  
double dz(double x, double y, double z)  
{  
return x \* y - 8.0 \* z / 3.0;  
}

**Реализация метода Эллера для решения дифференциальных уравнений:**

double ODE\_Euler(std::function<double(double, double)> f, double x0, double y0, double h, double x)  
{  
double xnew, ynew, result;  
if (x <= x0)  
result = y0;  
else if (x > x0)  
{  
do  
{  
if (h > x - x0) h = x - x0;  
ynew = y0 + f(x0, y0) \* h;  
xnew = x0 + h;  
x0 = xnew;  
y0 = ynew;  
} while (x0 < x);  
result = ynew;  
}  
return result;  
}

Причём, при помощи парадигмы функционального программирования данный метод может работать, как с пользовательскими функциями, так и с готовыми функциями, потому что получает на вход заданную функцию.Аналогично для реализации метода Рунге-Кутта 2 порядка, 4-го порядка и Рунге-Кутта Фельберга.

**Реализация пользовательской функции:**

double UserFunc(double x, double y) {  
marshal\_context^ marshal = gcnew marshal\_context();  
std::string buffer;  
buffer = marshal->marshal\_as<const char>(IntegralSolver::textBox4->Text);  
  
for (int i = 0; i < buffer.size(); i++) {  
if (buffer[i] == 'x') {  
buffer.erase(i, 1);  
buffer.insert(i, std::to\_string(x));  
}  
if (buffer[i] == 'y') {  
buffer.erase(i, 1);  
buffer.insert(i, std::to\_string(y));  
}  
}  
  
for (int i = 0; i < buffer.size(); i++) {  
if (buffer[i] == ',') buffer[i] = '.';  
}  
char cstr = new char[buffer.length() + 1];  
strcpy(cstr, buffer.c\_str());  
eatspaces(cstr); //удаляем все пробелы из строки  
return expr(cstr);  
}

Берётся строка из поля для ввода,все x и y заменяются числами «от» и «до».

**Реализация функции eatspaces:**

void eatspaces(char\* str)  
{  
int i = 0; //индекс места в строке "куда копировать"  
int j = 0; //индекс места в строке "откуда копировать"  
while (((str + i) =(str + j++)) != '\0') //цикл, пока очередной символ не '\0'  
if (\*(str + i) != ' ') //увеличиваем i, если символ не пробел  
i++;  
return;  
}

**Ниже представлена реализация парсинга математических выражений (перевода строки в готовую формулу):**

double expr(char\* str)

{

double value = 0.0; //здесь сохраняем результат

int index = 0; //текущая позиция символа

value = term(str, index); //получить первый элемент

for (; ; ) //бесконечный цикл, выход внутри

{

switch (\*(str + index++)) //выбрать действие на основе текущего символа

{

case '\0': //конец строки, возвращаем значение

return value;

case '+': //знак плюс, прибавляем элемент к value

value += term(str, index);

break;

case '-': //знак минус, вычитаем элемент из value

value -= term(str, index);

break;

default: //все остальное не котируется

int i = index;

while (--i > 0)

MessageBox::Show("Неправильно задана формула.", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

}

}

double term(char\* str, int& index)

{

double value = 0.0; //здесь накапливается значение результата

value = power(str, index); //получить первое число элемента

//выполняем цикл до тех пор, пока имеем допустимую операцию

while ((\*(str + index) == '\*') || (\*(str + index) == '/'))

{

if (\*(str + index) == '\*')

value \*= power(str, ++index);

if (\*(str + index) == '/')

value /= power(str, ++index);

}

return value;

}

double power(char\* str, int& index)

{

double value = 0.0;

value = trigon(str, index);

while (\*(str + index) == '^')

{

value = pow(value, trigon(str, ++index)); //возводим в степень

}

return value;

}

double trigon(char\* str, int& index)

{

int buf\_index = 0;

int temp\_index = index; //переменная для хранения индекса (чтобы если что вернуть индекс без изменений)

char\* p\_str = 0; //временный указатель для сравнения символов

double value = 0;

while (isalpha(\*(str + temp\_index)))

{

buf\_index++; //сколько букв

temp\_index++; //текущий индекс

}

if (!buf\_index) //если нет ни одной буквы, то возвращаем число

{

value = number(str, index);

return value;

}

else //иначе смотрим, являются ли буквы чем-нибудь этим

{

p\_str = new char[buf\_index + 1]; //а для этого создаем временную строку, чтобы сравнить

p\_str[buf\_index] = '\0';

strncpy(p\_str, str + index, buf\_index);

}

if (strcmp(p\_str, "sin") == 0) //синус в градусах

{

value = sin(3.141592 / 180 \* number(str, temp\_index));

index = temp\_index;

delete[] p\_str; //не забываем удалить временную строку

return value;

}

else if (strcmp(p\_str, "cos") == 0) //косинус в градусах

{

value = cos(3.141592 / 180 \* number(str, temp\_index));

index = temp\_index;

delete[] p\_str; //не забываем удалить временную строку

return value;

}

else if (strcmp(p\_str, "tan") == 0) //тангенс в градусах

{

value = tan(3.141592 / 180 \* number(str, temp\_index));

index = temp\_index;

delete[] p\_str; //не забываем удалить временную строку

return value;

}

else

{

return value;

}

}

double number(char\* str, int& index)

{

double value = 0.0; //хранит результирующее значение

if (\*(str + index) == '(')

{

char\* p\_substr = 0;

p\_substr = extract(str, ++index);

value = expr(p\_substr);

delete[] p\_substr;

return value;

}

//продуманский цикл, превращает символы в число

while (isdigit(\*(str + index))) //цикл накапливает ведущие цифры

value = 10 \* value + (\*(str + index++) - '0');

if (\*(str + index) != '.') //если не цифра, проверяем на десятичную точку

return value;

double factor = 1.0; //множитель для десятичных разрядов

//еще один продуманский цикл, возвращает десятичную часть

while (isdigit(\*(str + (++index)))) //выполнять цикл, пока идут цифры

{

factor \*= 0.1;

value = value + (\*(str + index) - '0') \* factor;

}

return value;

}

char\* extract(char\* str, int& index)

{

char buffer[MAX]; //временное пространство для подстроки

char\* p\_str = 0; //указатель на новую строку для возврата

int numL = 0; //счетчик найденных левых скобок

int buf\_index = index; //сохранить начальное значение index

do

{

buffer[index - buf\_index] = \*(str + index); //копируем символ текущей строки в подстроку

switch (buffer[index - buf\_index]) //смотрим, чо это за символ

{

case ')':

if (numL == 0)

{

buffer[index - buf\_index] = '\0'; //если счетчик скобочек верный, ставим символ конца строки

++index; //устанавливаем индекс на следующий за скобочкой элемент

p\_str = new char[index - buf\_index];

if (!p\_str)

{

MessageBox::Show("Ошибка при выделении памяти.", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

strcpy\_s(p\_str, index - buf\_index, buffer); //и копируем подстроку в новую память

return p\_str;

}

else

numL--; //уменьшаем счетчик скобок

break;

case '(':

numL++; //соответственно увеличиваем

break;

}

} while (\*(str + index++) != '\0'); //устанавливаем индекс в следующий элемент

MessageBox::Show("Выход за пределы выражения, возможно, плохой ввод.", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Warning);

return p\_str;

}

Код программы:

#pragma once

#include <windows.h>

#include <string>

#include <msclr/marshal.h>

#include <functional>

#include <vector>

#include "libxl.h"

const unsigned int MAX = 400; //максимальный размер буфера

int r1 = 17, g1 = 23, b1 = 46;

int r2 = 17, g2 = 23, b2 = 46;

int r3 = 17, g3 = 23, b3 = 46;

int r4 = 17, g4 = 23, b4 = 46;

bool back1 = false;

bool back2 = false;

bool back3 = false;

bool back4 = false;

namespace IntegralSolverV2 {

using namespace System;

using namespace System::ComponentModel;

using namespace System::Collections;

using namespace System::Windows::Forms;

using namespace System::Data;

using namespace System::Drawing;

using namespace msclr::interop;

using namespace libxl;

/// <summary>

/// Сводка для IntegralSolver

/// </summary>

double Func(double x, double y);

double dx(double x, double y, double z);

double dy(double x, double y, double z);

double dz(double x, double y, double z);

double ODE\_Euler(std::function<double(double, double)> f, double x0, double y0, double h, double x);

double ODE\_RungeKutta2(std::function<double(double, double)> f, double x0, double y0, double h, double x);

double ODE\_RungeKutta4(std::function<double(double, double)> f, double x0, double y0, double h, double x);

double ODE\_RungeKuttaFehlberg(std::function<double(double, double)> f, double x0, double y0, double x, double h, double tolerance);

double UserFunc(double x, double y);

void eatspaces(char\* str); //функция для удаления пробелов

double expr(char\* str); //функция, вычисляющая выражение

double term(char\* str, int& index); //функция для анализа элемента

double trigon(char\* str, int& index); //функция для вычисления тригонометрических значений

double power(char\* str, int& index); //функция для извлечения степени

double number(char\* str, int& index); //функция, распознающая число

char\* extract(char\* str, int& index); //функция для извлечения подстроки

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

{

public:

static bool Euler = false;

static bool RungeKutta2 = false;

static bool RungeKutta4 = false;

static bool RungeKuttaFehlberg = false;

Book\* Data = xlCreateBook();

double accuracy = 0.1;

double x0 = 0.;

double y0 = 1.;

double result = y0;

public:

IntegralSolver(void)

{

InitializeComponent();

//

//TODO: добавьте код конструктора

//

}

protected:

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

~IntegralSolver()

{

if (components)

{

delete components;

}

}

protected:

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

private: System::Windows::Forms::DataGridView^ dataGridView1;

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

private: System::Windows::Forms::CheckBox^ checkBox1;

private: System::Windows::Forms::CheckBox^ checkBox2;

private: System::Windows::Forms::CheckBox^ checkBox3;

private: System::Windows::Forms::CheckBox^ checkBox4;

private: System::Windows::Forms::Panel^ panel1;

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

private: System::Windows::Forms::Label^ label2;

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

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

private: System::Windows::Forms::TextBox^ textBox2;

private: System::Windows::Forms::Label^ label4;

private: System::Windows::Forms::TextBox^ textBox3;

private: System::Windows::Forms::Label^ label5;

private: System::Windows::Forms::Label^ label7;

private: System::Windows::Forms::Label^ label8;

public: static System::Windows::Forms::ComboBox^ comboBox1;

private: System::Windows::Forms::Panel^ panel4;

private: System::Windows::Forms::Panel^ panel5;

private: System::Windows::Forms::Panel^ panel6;

private: System::Windows::Forms::Label^ label14;

private: System::Windows::Forms::Panel^ panel7;

private: System::Windows::Forms::Panel^ panel8;

private: System::Windows::Forms::Label^ label18;

private: System::Windows::Forms::Panel^ panel9;

private: System::Windows::Forms::Panel^ panel10;

private: System::Windows::Forms::Label^ label19;

private: System::Windows::Forms::Panel^ panel11;

private: System::Windows::Forms::Panel^ panel12;

private: System::Windows::Forms::Label^ label20;

private: System::Windows::Forms::PictureBox^ pictureBox1;

private: System::Windows::Forms::Panel^ panel13;

private: System::Windows::Forms::Panel^ panel14;

private: System::Windows::Forms::Panel^ panel15;

private: System::Windows::Forms::Panel^ panel16;

private: System::Windows::Forms::Panel^ panel17;

private: System::Windows::Forms::Label^ label21;

private: System::Windows::Forms::Panel^ panel18;

private: System::Windows::Forms::PictureBox^ pictureBox4;

private: System::Windows::Forms::PictureBox^ pictureBox3;

private: System::Windows::Forms::PictureBox^ pictureBox2;

private: System::Windows::Forms::Timer^ timer1;

private: System::Windows::Forms::Timer^ timer2;

private: System::Windows::Forms::Timer^ timer3;

private: System::Windows::Forms::Timer^ timer4;

private: System::Windows::Forms::ProgressBar^ progressBar1;

private: System::Windows::Forms::Label^ label9;

private: System::Windows::Forms::Panel^ panel2;

private: System::Windows::Forms::Label^ label10;

private: System::Windows::Forms::Panel^ panel3;

private: System::Windows::Forms::Label^ label11;

private: System::Windows::Forms::Label^ label12;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column1;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ Column2;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn3;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn4;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn5;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn6;

private: System::Windows::Forms::DataGridView^ dataGridView5;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn7;

private: System::Windows::Forms::DataGridViewTextBoxColumn^ dataGridViewTextBoxColumn8;

private: System::Windows::Forms::Panel^ panel19;

private: System::Windows::Forms::CheckBox^ checkBox5;

private: System::Windows::Forms::Label^ label6;

public: static System::Windows::Forms::TextBox^ textBox4;

private: System::Windows::Forms::CheckBox^ checkBox7;

private: System::Windows::Forms::CheckBox^ checkBox8;

private: System::Windows::Forms::CheckBox^ checkBox9;

private: System::Windows::Forms::CheckBox^ checkBox10;

private: System::Windows::Forms::Label^ label13;

private: System::Windows::Forms::Label^ label15;

private: System::Windows::Forms::DataGridView^ dataGridView3;

private: System::Windows::Forms::Label^ label16;

private: System::Windows::Forms::DataGridView^ dataGridView4;

private: System::Windows::Forms::Label^ label17;

private: System::ComponentModel::IContainer^ components;

private:

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

#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)

{

this->components = (gcnew System::ComponentModel::Container());

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

System::Windows::Forms::DataVisualization::Charting::Legend^ legend1 = (gcnew System::Windows::Forms::DataVisualization::Charting::Legend());

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::Windows::Forms::DataVisualization::Charting::Series^ series3 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

System::Windows::Forms::DataVisualization::Charting::Series^ series4 = (gcnew System::Windows::Forms::DataVisualization::Charting::Series());

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

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

this->dataGridView1 = (gcnew System::Windows::Forms::DataGridView());

this->Column1 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->Column2 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

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

this->checkBox1 = (gcnew System::Windows::Forms::CheckBox());

this->checkBox2 = (gcnew System::Windows::Forms::CheckBox());

this->checkBox3 = (gcnew System::Windows::Forms::CheckBox());

this->checkBox4 = (gcnew System::Windows::Forms::CheckBox());

this->panel1 = (gcnew System::Windows::Forms::Panel());

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

this->label2 = (gcnew System::Windows::Forms::Label());

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

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

this->textBox2 = (gcnew System::Windows::Forms::TextBox());

this->label4 = (gcnew System::Windows::Forms::Label());

this->textBox3 = (gcnew System::Windows::Forms::TextBox());

this->label5 = (gcnew System::Windows::Forms::Label());

this->label7 = (gcnew System::Windows::Forms::Label());

this->label8 = (gcnew System::Windows::Forms::Label());

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

this->progressBar1 = (gcnew System::Windows::Forms::ProgressBar());

this->label9 = (gcnew System::Windows::Forms::Label());

this->panel2 = (gcnew System::Windows::Forms::Panel());

this->label10 = (gcnew System::Windows::Forms::Label());

this->label12 = (gcnew System::Windows::Forms::Label());

this->checkBox7 = (gcnew System::Windows::Forms::CheckBox());

this->checkBox8 = (gcnew System::Windows::Forms::CheckBox());

this->checkBox9 = (gcnew System::Windows::Forms::CheckBox());

this->checkBox10 = (gcnew System::Windows::Forms::CheckBox());

this->panel3 = (gcnew System::Windows::Forms::Panel());

this->label11 = (gcnew System::Windows::Forms::Label());

this->label13 = (gcnew System::Windows::Forms::Label());

this->label15 = (gcnew System::Windows::Forms::Label());

this->dataGridView3 = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn3 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn4 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->label16 = (gcnew System::Windows::Forms::Label());

this->dataGridView4 = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn5 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn6 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->label17 = (gcnew System::Windows::Forms::Label());

this->checkBox5 = (gcnew System::Windows::Forms::CheckBox());

this->label6 = (gcnew System::Windows::Forms::Label());

this->textBox4 = (gcnew System::Windows::Forms::TextBox());

this->panel4 = (gcnew System::Windows::Forms::Panel());

this->pictureBox4 = (gcnew System::Windows::Forms::PictureBox());

this->pictureBox3 = (gcnew System::Windows::Forms::PictureBox());

this->pictureBox2 = (gcnew System::Windows::Forms::PictureBox());

this->panel5 = (gcnew System::Windows::Forms::Panel());

this->panel6 = (gcnew System::Windows::Forms::Panel());

this->label14 = (gcnew System::Windows::Forms::Label());

this->panel7 = (gcnew System::Windows::Forms::Panel());

this->panel8 = (gcnew System::Windows::Forms::Panel());

this->label18 = (gcnew System::Windows::Forms::Label());

this->panel9 = (gcnew System::Windows::Forms::Panel());

this->panel10 = (gcnew System::Windows::Forms::Panel());

this->label19 = (gcnew System::Windows::Forms::Label());

this->panel11 = (gcnew System::Windows::Forms::Panel());

this->panel12 = (gcnew System::Windows::Forms::Panel());

this->label20 = (gcnew System::Windows::Forms::Label());

this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());

this->panel13 = (gcnew System::Windows::Forms::Panel());

this->panel19 = (gcnew System::Windows::Forms::Panel());

this->panel14 = (gcnew System::Windows::Forms::Panel());

this->panel15 = (gcnew System::Windows::Forms::Panel());

this->panel16 = (gcnew System::Windows::Forms::Panel());

this->dataGridView5 = (gcnew System::Windows::Forms::DataGridView());

this->dataGridViewTextBoxColumn7 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->dataGridViewTextBoxColumn8 = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());

this->panel17 = (gcnew System::Windows::Forms::Panel());

this->label21 = (gcnew System::Windows::Forms::Label());

this->panel18 = (gcnew System::Windows::Forms::Panel());

this->timer1 = (gcnew System::Windows::Forms::Timer(this->components));

this->timer2 = (gcnew System::Windows::Forms::Timer(this->components));

this->timer3 = (gcnew System::Windows::Forms::Timer(this->components));

this->timer4 = (gcnew System::Windows::Forms::Timer(this->components));

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

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();

this->panel1->SuspendLayout();

this->panel2->SuspendLayout();

this->panel3->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView3))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView4))->BeginInit();

this->panel4->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox4))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox3))->BeginInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox2))->BeginInit();

this->panel5->SuspendLayout();

this->panel7->SuspendLayout();

this->panel9->SuspendLayout();

this->panel11->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();

this->panel13->SuspendLayout();

this->panel19->SuspendLayout();

this->panel14->SuspendLayout();

this->panel15->SuspendLayout();

this->panel16->SuspendLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView5))->BeginInit();

this->panel17->SuspendLayout();

this->panel18->SuspendLayout();

this->SuspendLayout();

//

// chart1

//

this->chart1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(17)), static\_cast<System::Int32>(static\_cast<System::Byte>(23)),

static\_cast<System::Int32>(static\_cast<System::Byte>(46)));

this->chart1->BorderlineColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)), static\_cast<System::Int32>(static\_cast<System::Byte>(15)),

static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

chartArea1->Area3DStyle->LightStyle = System::Windows::Forms::DataVisualization::Charting::LightStyle::Realistic;

chartArea1->Area3DStyle->WallWidth = 6;

chartArea1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(17)), static\_cast<System::Int32>(static\_cast<System::Byte>(23)),

static\_cast<System::Int32>(static\_cast<System::Byte>(46)));

chartArea1->BackGradientStyle = System::Windows::Forms::DataVisualization::Charting::GradientStyle::VerticalCenter;

chartArea1->BackSecondaryColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(17)), static\_cast<System::Int32>(static\_cast<System::Byte>(23)),

static\_cast<System::Int32>(static\_cast<System::Byte>(46)));

chartArea1->Name = L"ChartArea1";

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

legend1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(17)), static\_cast<System::Int32>(static\_cast<System::Byte>(23)),

static\_cast<System::Int32>(static\_cast<System::Byte>(46)));

legend1->BackImageTransparentColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)),

static\_cast<System::Int32>(static\_cast<System::Byte>(15)), static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

legend1->BackSecondaryColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)), static\_cast<System::Int32>(static\_cast<System::Byte>(15)),

static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

legend1->BorderColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)), static\_cast<System::Int32>(static\_cast<System::Byte>(15)),

static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

legend1->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 9.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

legend1->ForeColor = System::Drawing::Color::White;

legend1->IsTextAutoFit = false;

legend1->Name = L"Legend1";

legend1->Title = L"Обозначения";

legend1->TitleFont = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 9.75F, System::Drawing::FontStyle::Bold));

legend1->TitleForeColor = System::Drawing::Color::White;

this->chart1->Legends->Add(legend1);

this->chart1->Location = System::Drawing::Point(17, 15);

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

this->chart1->Palette = System::Windows::Forms::DataVisualization::Charting::ChartColorPalette::Excel;

series1->BorderColor = System::Drawing::Color::White;

series1->BorderWidth = 6;

series1->ChartArea = L"ChartArea1";

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

series1->Color = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(214)), static\_cast<System::Int32>(static\_cast<System::Byte>(69)),

static\_cast<System::Int32>(static\_cast<System::Byte>(27)));

series1->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

series1->LabelForeColor = System::Drawing::Color::White;

series1->Legend = L"Legend1";

series1->Name = L"Метод Эйлера";

series2->BorderWidth = 5;

series2->ChartArea = L"ChartArea1";

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

series2->Color = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(40)), static\_cast<System::Int32>(static\_cast<System::Byte>(157)),

static\_cast<System::Int32>(static\_cast<System::Byte>(19)));

series2->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

series2->LabelForeColor = System::Drawing::Color::White;

series2->Legend = L"Legend1";

series2->Name = L"Рунге-Кутта 2 порядка";

series3->BorderWidth = 4;

series3->ChartArea = L"ChartArea1";

series3->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Line;

series3->Color = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(12)), static\_cast<System::Int32>(static\_cast<System::Byte>(113)),

static\_cast<System::Int32>(static\_cast<System::Byte>(203)));

series3->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

series3->LabelForeColor = System::Drawing::Color::White;

series3->Legend = L"Legend1";

series3->Name = L"Рунге-Кутта 4 порядка";

series4->BorderWidth = 3;

series4->ChartArea = L"ChartArea1";

series4->ChartType = System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Line;

series4->Color = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(175)), static\_cast<System::Int32>(static\_cast<System::Byte>(48)),

static\_cast<System::Int32>(static\_cast<System::Byte>(163)));

series4->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

series4->LabelForeColor = System::Drawing::Color::White;

series4->Legend = L"Legend1";

series4->Name = L"Рунге-Кутта-Фельберг";

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

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

this->chart1->Series->Add(series3);

this->chart1->Series->Add(series4);

this->chart1->Size = System::Drawing::Size(787, 416);

this->chart1->TabIndex = 0;

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

//

// dataGridView1

//

this->dataGridView1->BackgroundColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)),

static\_cast<System::Int32>(static\_cast<System::Byte>(15)), static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView1->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(2) {

this->Column1,

this->Column2

});

this->dataGridView1->Location = System::Drawing::Point(21, 60);

this->dataGridView1->Name = L"dataGridView1";

this->dataGridView1->ReadOnly = true;

this->dataGridView1->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->dataGridView1->Size = System::Drawing::Size(354, 165);

this->dataGridView1->TabIndex = 1;

//

// Column1

//

this->Column1->HeaderText = L"X";

this->Column1->Name = L"Column1";

this->Column1->ReadOnly = true;

this->Column1->Width = 150;

//

// Column2

//

this->Column2->HeaderText = L"Y";

this->Column2->Name = L"Column2";

this->Column2->ReadOnly = true;

this->Column2->Width = 200;

//

// richTextBox1

//

this->richTextBox1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)), static\_cast<System::Int32>(static\_cast<System::Byte>(15)),

static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

this->richTextBox1->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->richTextBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->richTextBox1->ForeColor = System::Drawing::Color::White;

this->richTextBox1->Location = System::Drawing::Point(3, 3);

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

this->richTextBox1->ReadOnly = true;

this->richTextBox1->ScrollBars = System::Windows::Forms::RichTextBoxScrollBars::Vertical;

this->richTextBox1->Size = System::Drawing::Size(250, 425);

this->richTextBox1->TabIndex = 2;

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

//

// checkBox1

//

this->checkBox1->AutoSize = true;

this->checkBox1->BackColor = System::Drawing::Color::Transparent;

this->checkBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->checkBox1->ForeColor = System::Drawing::Color::White;

this->checkBox1->Location = System::Drawing::Point(24, 62);

this->checkBox1->Name = L"checkBox1";

this->checkBox1->Size = System::Drawing::Size(98, 27);

this->checkBox1->TabIndex = 3;

this->checkBox1->Text = L"Эйлера";

this->checkBox1->UseVisualStyleBackColor = false;

this->checkBox1->CheckedChanged += gcnew System::EventHandler(this, &IntegralSolver::checkBox1\_CheckedChanged);

//

// checkBox2

//

this->checkBox2->AutoSize = true;

this->checkBox2->BackColor = System::Drawing::Color::Transparent;

this->checkBox2->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->checkBox2->ForeColor = System::Drawing::Color::White;

this->checkBox2->Location = System::Drawing::Point(24, 99);

this->checkBox2->Name = L"checkBox2";

this->checkBox2->Size = System::Drawing::Size(242, 27);

this->checkBox2->TabIndex = 4;

this->checkBox2->Text = L"Рунге-Кутта 2 порядка";

this->checkBox2->UseVisualStyleBackColor = false;

this->checkBox2->CheckedChanged += gcnew System::EventHandler(this, &IntegralSolver::checkBox2\_CheckedChanged);

//

// checkBox3

//

this->checkBox3->AutoSize = true;

this->checkBox3->BackColor = System::Drawing::Color::Transparent;

this->checkBox3->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->checkBox3->ForeColor = System::Drawing::Color::White;

this->checkBox3->Location = System::Drawing::Point(24, 137);

this->checkBox3->Name = L"checkBox3";

this->checkBox3->Size = System::Drawing::Size(242, 27);

this->checkBox3->TabIndex = 5;

this->checkBox3->Text = L"Рунге-Кутта 4 порядка";

this->checkBox3->UseVisualStyleBackColor = false;

this->checkBox3->CheckedChanged += gcnew System::EventHandler(this, &IntegralSolver::checkBox3\_CheckedChanged);

//

// checkBox4

//

this->checkBox4->AutoSize = true;

this->checkBox4->BackColor = System::Drawing::Color::Transparent;

this->checkBox4->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->checkBox4->ForeColor = System::Drawing::Color::White;

this->checkBox4->Location = System::Drawing::Point(24, 177);

this->checkBox4->Name = L"checkBox4";

this->checkBox4->Size = System::Drawing::Size(243, 27);

this->checkBox4->TabIndex = 6;

this->checkBox4->Text = L"Рунге-Кутта-Фельберг";

this->checkBox4->UseVisualStyleBackColor = false;

this->checkBox4->CheckedChanged += gcnew System::EventHandler(this, &IntegralSolver::checkBox4\_CheckedChanged);

//

// panel1

//

this->panel1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(3)), static\_cast<System::Int32>(static\_cast<System::Byte>(93)),

static\_cast<System::Int32>(static\_cast<System::Byte>(223)));

this->panel1->Controls->Add(this->label1);

this->panel1->Location = System::Drawing::Point(591, 376);

this->panel1->Name = L"panel1";

this->panel1->Size = System::Drawing::Size(213, 47);

this->panel1->TabIndex = 8;

this->panel1->Click += gcnew System::EventHandler(this, &IntegralSolver::panel1\_Click);

this->panel1->MouseEnter += gcnew System::EventHandler(this, &IntegralSolver::panel1\_MouseEnter);

this->panel1->MouseLeave += gcnew System::EventHandler(this, &IntegralSolver::panel1\_MouseLeave);

//

// label1

//

this->label1->AutoSize = true;

this->label1->BackColor = System::Drawing::Color::Transparent;

this->label1->Enabled = false;

this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 15.75F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label1->ForeColor = System::Drawing::Color::White;

this->label1->Location = System::Drawing::Point(37, 10);

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

this->label1->Size = System::Drawing::Size(139, 27);

this->label1->TabIndex = 0;

this->label1->Text = L"Рассчитать";

//

// label2

//

this->label2->AutoSize = true;

this->label2->BackColor = System::Drawing::Color::Transparent;

this->label2->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label2->ForeColor = System::Drawing::Color::White;

this->label2->Location = System::Drawing::Point(64, 259);

this->label2->Name = L"label2";

this->label2->Size = System::Drawing::Size(102, 23);

this->label2->TabIndex = 9;

this->label2->Text = L"Точность:";

//

// label3

//

this->label3->AutoSize = true;

this->label3->BackColor = System::Drawing::Color::Transparent;

this->label3->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label3->ForeColor = System::Drawing::Color::White;

this->label3->Location = System::Drawing::Point(64, 296);

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

this->label3->Size = System::Drawing::Size(40, 23);

this->label3->TabIndex = 10;

this->label3->Text = L"От:";

//

// textBox1

//

this->textBox1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)), static\_cast<System::Int32>(static\_cast<System::Byte>(15)),

static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

this->textBox1->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->textBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->textBox1->ForeColor = System::Drawing::Color::White;

this->textBox1->Location = System::Drawing::Point(125, 296);

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

this->textBox1->Size = System::Drawing::Size(100, 25);

this->textBox1->TabIndex = 11;

//

// textBox2

//

this->textBox2->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)), static\_cast<System::Int32>(static\_cast<System::Byte>(15)),

static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

this->textBox2->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->textBox2->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->textBox2->ForeColor = System::Drawing::Color::White;

this->textBox2->Location = System::Drawing::Point(125, 336);

this->textBox2->Name = L"textBox2";

this->textBox2->Size = System::Drawing::Size(100, 25);

this->textBox2->TabIndex = 13;

//

// label4

//

this->label4->AutoSize = true;

this->label4->BackColor = System::Drawing::Color::Transparent;

this->label4->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label4->ForeColor = System::Drawing::Color::White;

this->label4->Location = System::Drawing::Point(64, 336);

this->label4->Name = L"label4";

this->label4->Size = System::Drawing::Size(40, 23);

this->label4->TabIndex = 12;

this->label4->Text = L"До:";

//

// textBox3

//

this->textBox3->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)), static\_cast<System::Int32>(static\_cast<System::Byte>(15)),

static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

this->textBox3->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->textBox3->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->textBox3->ForeColor = System::Drawing::Color::White;

this->textBox3->Location = System::Drawing::Point(125, 382);

this->textBox3->Name = L"textBox3";

this->textBox3->Size = System::Drawing::Size(100, 25);

this->textBox3->TabIndex = 15;

//

// label5

//

this->label5->AutoSize = true;

this->label5->BackColor = System::Drawing::Color::Transparent;

this->label5->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label5->ForeColor = System::Drawing::Color::White;

this->label5->Location = System::Drawing::Point(64, 382);

this->label5->Name = L"label5";

this->label5->Size = System::Drawing::Size(50, 23);

this->label5->TabIndex = 14;

this->label5->Text = L"Шаг:";

//

// label7

//

this->label7->AutoSize = true;

this->label7->BackColor = System::Drawing::Color::Transparent;

this->label7->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label7->ForeColor = System::Drawing::Color::White;

this->label7->Location = System::Drawing::Point(20, 24);

this->label7->Name = L"label7";

this->label7->Size = System::Drawing::Size(76, 23);

this->label7->TabIndex = 18;

this->label7->Text = L"Метод:";

//

// label8

//

this->label8->AutoSize = true;

this->label8->BackColor = System::Drawing::Color::Transparent;

this->label8->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label8->ForeColor = System::Drawing::Color::White;

this->label8->Location = System::Drawing::Point(401, 81);

this->label8->Name = L"label8";

this->label8->Size = System::Drawing::Size(174, 23);

this->label8->TabIndex = 19;

this->label8->Text = L"Готовые функции";

//

// comboBox1

//

this->comboBox1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)), static\_cast<System::Int32>(static\_cast<System::Byte>(15)),

static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

this->comboBox1->DisplayMember = L"(нет)";

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

this->comboBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->comboBox1->ForeColor = System::Drawing::Color::White;

this->comboBox1->FormattingEnabled = true;

this->comboBox1->Items->AddRange(gcnew cli::array< System::Object^ >(5) {

L"x \* y", L"y \* cos(x)", L"sin(y) \* cos(x)", L"1 / (x \* y - 1)",

L"x \* x / (y \* y)"

});

this->comboBox1->Location = System::Drawing::Point(623, 81);

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

this->comboBox1->Size = System::Drawing::Size(121, 31);

this->comboBox1->TabIndex = 20;

this->comboBox1->Text = L"x\*y";

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

//

// progressBar1

//

this->progressBar1->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(3)), static\_cast<System::Int32>(static\_cast<System::Byte>(93)),

static\_cast<System::Int32>(static\_cast<System::Byte>(223)));

this->progressBar1->Location = System::Drawing::Point(270, 391);

this->progressBar1->Name = L"progressBar1";

this->progressBar1->Size = System::Drawing::Size(305, 32);

this->progressBar1->TabIndex = 21;

//

// label9

//

this->label9->AutoSize = true;

this->label9->BackColor = System::Drawing::Color::Transparent;

this->label9->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label9->ForeColor = System::Drawing::Color::White;

this->label9->Location = System::Drawing::Point(403, 353);

this->label9->Name = L"label9";

this->label9->Size = System::Drawing::Size(37, 23);

this->label9->TabIndex = 22;

this->label9->Text = L"0%";

//

// panel2

//

this->panel2->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(3)), static\_cast<System::Int32>(static\_cast<System::Byte>(93)),

static\_cast<System::Int32>(static\_cast<System::Byte>(223)));

this->panel2->Controls->Add(this->label10);

this->panel2->Location = System::Drawing::Point(74, 251);

this->panel2->Name = L"panel2";

this->panel2->Size = System::Drawing::Size(213, 51);

this->panel2->TabIndex = 9;

this->panel2->Click += gcnew System::EventHandler(this, &IntegralSolver::panel2\_Click);

this->panel2->MouseEnter += gcnew System::EventHandler(this, &IntegralSolver::panel2\_MouseEnter);

this->panel2->MouseLeave += gcnew System::EventHandler(this, &IntegralSolver::panel2\_MouseLeave);

//

// label10

//

this->label10->AutoSize = true;

this->label10->BackColor = System::Drawing::Color::Transparent;

this->label10->Enabled = false;

this->label10->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label10->ForeColor = System::Drawing::Color::White;

this->label10->Location = System::Drawing::Point(13, 15);

this->label10->Name = L"label10";

this->label10->Size = System::Drawing::Size(183, 23);

this->label10->TabIndex = 0;

this->label10->Text = L"Выгрузить данные";

//

// label12

//

this->label12->AutoSize = true;

this->label12->BackColor = System::Drawing::Color::Transparent;

this->label12->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 18, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label12->ForeColor = System::Drawing::Color::White;

this->label12->Location = System::Drawing::Point(115, 18);

this->label12->Name = L"label12";

this->label12->Size = System::Drawing::Size(93, 31);

this->label12->TabIndex = 30;

this->label12->Text = L"Метод";

//

// checkBox7

//

this->checkBox7->AutoSize = true;

this->checkBox7->BackColor = System::Drawing::Color::Transparent;

this->checkBox7->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->checkBox7->ForeColor = System::Drawing::Color::White;

this->checkBox7->Location = System::Drawing::Point(75, 194);

this->checkBox7->Name = L"checkBox7";

this->checkBox7->Size = System::Drawing::Size(243, 27);

this->checkBox7->TabIndex = 28;

this->checkBox7->Text = L"Рунге-Кутта-Фельберг";

this->checkBox7->UseVisualStyleBackColor = false;

//

// checkBox8

//

this->checkBox8->AutoSize = true;

this->checkBox8->BackColor = System::Drawing::Color::Transparent;

this->checkBox8->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->checkBox8->ForeColor = System::Drawing::Color::White;

this->checkBox8->Location = System::Drawing::Point(74, 161);

this->checkBox8->Name = L"checkBox8";

this->checkBox8->Size = System::Drawing::Size(242, 27);

this->checkBox8->TabIndex = 27;

this->checkBox8->Text = L"Рунге-Кутта 4 порядка";

this->checkBox8->UseVisualStyleBackColor = false;

//

// checkBox9

//

this->checkBox9->AutoSize = true;

this->checkBox9->BackColor = System::Drawing::Color::Transparent;

this->checkBox9->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->checkBox9->ForeColor = System::Drawing::Color::White;

this->checkBox9->Location = System::Drawing::Point(74, 128);

this->checkBox9->Name = L"checkBox9";

this->checkBox9->Size = System::Drawing::Size(242, 27);

this->checkBox9->TabIndex = 26;

this->checkBox9->Text = L"Рунге-Кутта 2 порядка";

this->checkBox9->UseVisualStyleBackColor = false;

//

// checkBox10

//

this->checkBox10->AutoSize = true;

this->checkBox10->BackColor = System::Drawing::Color::Transparent;

this->checkBox10->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->checkBox10->ForeColor = System::Drawing::Color::White;

this->checkBox10->Location = System::Drawing::Point(75, 95);

this->checkBox10->Name = L"checkBox10";

this->checkBox10->Size = System::Drawing::Size(98, 27);

this->checkBox10->TabIndex = 25;

this->checkBox10->Text = L"Эйлера";

this->checkBox10->UseVisualStyleBackColor = false;

//

// panel3

//

this->panel3->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(43)), static\_cast<System::Int32>(static\_cast<System::Byte>(60)),

static\_cast<System::Int32>(static\_cast<System::Byte>(115)));

this->panel3->Controls->Add(this->label11);

this->panel3->Location = System::Drawing::Point(24, 444);

this->panel3->Name = L"panel3";

this->panel3->Size = System::Drawing::Size(213, 47);

this->panel3->TabIndex = 31;

this->panel3->Click += gcnew System::EventHandler(this, &IntegralSolver::panel3\_Click);

this->panel3->MouseEnter += gcnew System::EventHandler(this, &IntegralSolver::panel3\_MouseEnter);

this->panel3->MouseLeave += gcnew System::EventHandler(this, &IntegralSolver::panel3\_MouseLeave);

//

// label11

//

this->label11->AutoSize = true;

this->label11->BackColor = System::Drawing::Color::Transparent;

this->label11->Enabled = false;

this->label11->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 15.75F, System::Drawing::FontStyle::Bold));

this->label11->ForeColor = System::Drawing::Color::White;

this->label11->Location = System::Drawing::Point(20, 10);

this->label11->Name = L"label11";

this->label11->Size = System::Drawing::Size(174, 27);

this->label11->TabIndex = 0;

this->label11->Text = L"Очистить Логи";

//

// label13

//

this->label13->AutoSize = true;

this->label13->BackColor = System::Drawing::Color::Transparent;

this->label13->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label13->ForeColor = System::Drawing::Color::White;

this->label13->Location = System::Drawing::Point(17, 26);

this->label13->Name = L"label13";

this->label13->Size = System::Drawing::Size(68, 23);

this->label13->TabIndex = 32;

this->label13->Text = L"Эйлер";

//

// label15

//

this->label15->AutoSize = true;

this->label15->BackColor = System::Drawing::Color::Transparent;

this->label15->FlatStyle = System::Windows::Forms::FlatStyle::Flat;

this->label15->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label15->ForeColor = System::Drawing::Color::White;

this->label15->Location = System::Drawing::Point(435, 26);

this->label15->Name = L"label15";

this->label15->Size = System::Drawing::Size(261, 23);

this->label15->TabIndex = 36;

this->label15->Text = L"Рунге-Кутта с 2 параметра";

//

// dataGridView3

//

this->dataGridView3->BackgroundColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)),

static\_cast<System::Int32>(static\_cast<System::Byte>(15)), static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

this->dataGridView3->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView3->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(2) {

this->dataGridViewTextBoxColumn3,

this->dataGridViewTextBoxColumn4

});

this->dataGridView3->Location = System::Drawing::Point(438, 60);

this->dataGridView3->Name = L"dataGridView3";

this->dataGridView3->ReadOnly = true;

this->dataGridView3->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->dataGridView3->Size = System::Drawing::Size(354, 165);

this->dataGridView3->TabIndex = 35;

//

// dataGridViewTextBoxColumn3

//

this->dataGridViewTextBoxColumn3->HeaderText = L"X";

this->dataGridViewTextBoxColumn3->Name = L"dataGridViewTextBoxColumn3";

this->dataGridViewTextBoxColumn3->ReadOnly = true;

this->dataGridViewTextBoxColumn3->Width = 150;

//

// dataGridViewTextBoxColumn4

//

this->dataGridViewTextBoxColumn4->HeaderText = L"Y";

this->dataGridViewTextBoxColumn4->Name = L"dataGridViewTextBoxColumn4";

this->dataGridViewTextBoxColumn4->ReadOnly = true;

this->dataGridViewTextBoxColumn4->Width = 200;

//

// label16

//

this->label16->AutoSize = true;

this->label16->BackColor = System::Drawing::Color::Transparent;

this->label16->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label16->ForeColor = System::Drawing::Color::White;

this->label16->Location = System::Drawing::Point(435, 234);

this->label16->Name = L"label16";

this->label16->Size = System::Drawing::Size(261, 23);

this->label16->TabIndex = 38;

this->label16->Text = L"Рунге-Кутта с 4 параметра";

//

// dataGridView4

//

this->dataGridView4->BackgroundColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)),

static\_cast<System::Int32>(static\_cast<System::Byte>(15)), static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

this->dataGridView4->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView4->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(2) {

this->dataGridViewTextBoxColumn5,

this->dataGridViewTextBoxColumn6

});

this->dataGridView4->Location = System::Drawing::Point(439, 266);

this->dataGridView4->Name = L"dataGridView4";

this->dataGridView4->ReadOnly = true;

this->dataGridView4->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->dataGridView4->Size = System::Drawing::Size(353, 165);

this->dataGridView4->TabIndex = 37;

//

// dataGridViewTextBoxColumn5

//

this->dataGridViewTextBoxColumn5->HeaderText = L"X";

this->dataGridViewTextBoxColumn5->Name = L"dataGridViewTextBoxColumn5";

this->dataGridViewTextBoxColumn5->ReadOnly = true;

this->dataGridViewTextBoxColumn5->Width = 150;

//

// dataGridViewTextBoxColumn6

//

this->dataGridViewTextBoxColumn6->HeaderText = L"Y";

this->dataGridViewTextBoxColumn6->Name = L"dataGridViewTextBoxColumn6";

this->dataGridViewTextBoxColumn6->ReadOnly = true;

this->dataGridViewTextBoxColumn6->Width = 200;

//

// label17

//

this->label17->AutoSize = true;

this->label17->BackColor = System::Drawing::Color::Transparent;

this->label17->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label17->ForeColor = System::Drawing::Color::White;

this->label17->Location = System::Drawing::Point(17, 234);

this->label17->Name = L"label17";

this->label17->Size = System::Drawing::Size(224, 23);

this->label17->TabIndex = 40;

this->label17->Text = L"Рунге-Кутта-Фельберг";

//

// checkBox5

//

this->checkBox5->AutoSize = true;

this->checkBox5->BackColor = System::Drawing::Color::Transparent;

this->checkBox5->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->checkBox5->ForeColor = System::Drawing::Color::White;

this->checkBox5->Location = System::Drawing::Point(405, 125);

this->checkBox5->Name = L"checkBox5";

this->checkBox5->Size = System::Drawing::Size(290, 27);

this->checkBox5->TabIndex = 41;

this->checkBox5->Text = L"Пользовательская функция";

this->checkBox5->UseVisualStyleBackColor = false;

this->checkBox5->CheckedChanged += gcnew System::EventHandler(this, &IntegralSolver::checkBox5\_CheckedChanged);

//

// label6

//

this->label6->AutoSize = true;

this->label6->BackColor = System::Drawing::Color::Transparent;

this->label6->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->label6->ForeColor = System::Drawing::Color::White;

this->label6->Location = System::Drawing::Point(535, 42);

this->label6->Name = L"label6";

this->label6->Size = System::Drawing::Size(84, 23);

this->label6->TabIndex = 43;

this->label6->Text = L"Функци:";

//

// textBox4

//

this->textBox4->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)), static\_cast<System::Int32>(static\_cast<System::Byte>(15)),

static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

this->textBox4->BorderStyle = System::Windows::Forms::BorderStyle::None;

this->textBox4->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold));

this->textBox4->ForeColor = System::Drawing::Color::White;

this->textBox4->Location = System::Drawing::Point(405, 174);

this->textBox4->Name = L"textBox4";

this->textBox4->Size = System::Drawing::Size(346, 25);

this->textBox4->TabIndex = 42;

this->textBox4->TextChanged += gcnew System::EventHandler(this, &IntegralSolver::textBox4\_TextChanged);

//

// panel4

//

this->panel4->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(13)), static\_cast<System::Int32>(static\_cast<System::Byte>(19)),

static\_cast<System::Int32>(static\_cast<System::Byte>(37)));

this->panel4->Controls->Add(this->pictureBox4);

this->panel4->Controls->Add(this->pictureBox3);

this->panel4->Controls->Add(this->pictureBox2);

this->panel4->Location = System::Drawing::Point(0, 0);

this->panel4->Name = L"panel4";

this->panel4->Size = System::Drawing::Size(1094, 23);

this->panel4->TabIndex = 44;

this->panel4->MouseDown += gcnew System::Windows::Forms::MouseEventHandler(this, &IntegralSolver::Panel\_MouseDown);

this->panel4->MouseMove += gcnew System::Windows::Forms::MouseEventHandler(this, &IntegralSolver::Panel\_MouseMove);

this->panel4->MouseUp += gcnew System::Windows::Forms::MouseEventHandler(this, &IntegralSolver::Panel\_MouseUp);

//

// pictureBox4

//

this->pictureBox4->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox4.BackgroundImage")));

this->pictureBox4->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;

this->pictureBox4->Location = System::Drawing::Point(1034, 5);

this->pictureBox4->Name = L"pictureBox4";

this->pictureBox4->Size = System::Drawing::Size(15, 15);

this->pictureBox4->TabIndex = 55;

this->pictureBox4->TabStop = false;

this->pictureBox4->Click += gcnew System::EventHandler(this, &IntegralSolver::pictureBox4\_Click);

//

// pictureBox3

//

this->pictureBox3->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox3.BackgroundImage")));

this->pictureBox3->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;

this->pictureBox3->Location = System::Drawing::Point(1055, 5);

this->pictureBox3->Name = L"pictureBox3";

this->pictureBox3->Size = System::Drawing::Size(15, 15);

this->pictureBox3->TabIndex = 45;

this->pictureBox3->TabStop = false;

this->pictureBox3->Click += gcnew System::EventHandler(this, &IntegralSolver::pictureBox3\_Click);

//

// pictureBox2

//

this->pictureBox2->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox2.BackgroundImage")));

this->pictureBox2->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;

this->pictureBox2->Location = System::Drawing::Point(1076, 5);

this->pictureBox2->Name = L"pictureBox2";

this->pictureBox2->Size = System::Drawing::Size(15, 15);

this->pictureBox2->TabIndex = 44;

this->pictureBox2->TabStop = false;

this->pictureBox2->Click += gcnew System::EventHandler(this, &IntegralSolver::pictureBox2\_Click);

//

// panel5

//

this->panel5->Controls->Add(this->panel6);

this->panel5->Controls->Add(this->label14);

this->panel5->Location = System::Drawing::Point(114, 52);

this->panel5->Name = L"panel5";

this->panel5->Size = System::Drawing::Size(181, 57);

this->panel5->TabIndex = 45;

this->panel5->Click += gcnew System::EventHandler(this, &IntegralSolver::panel5\_Click);

this->panel5->MouseEnter += gcnew System::EventHandler(this, &IntegralSolver::panel5\_MouseEnter);

this->panel5->MouseLeave += gcnew System::EventHandler(this, &IntegralSolver::panel5\_MouseLeave);

//

// panel6

//

this->panel6->Location = System::Drawing::Point(0, 47);

this->panel6->Name = L"panel6";

this->panel6->Size = System::Drawing::Size(181, 10);

this->panel6->TabIndex = 46;

//

// label14

//

this->label14->AutoSize = true;

this->label14->BackColor = System::Drawing::Color::Transparent;

this->label14->Enabled = false;

this->label14->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label14->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(208)), static\_cast<System::Int32>(static\_cast<System::Byte>(219)),

static\_cast<System::Int32>(static\_cast<System::Byte>(250)));

this->label14->Location = System::Drawing::Point(33, 17);

this->label14->Name = L"label14";

this->label14->Size = System::Drawing::Size(110, 23);

this->label14->TabIndex = 0;

this->label14->Text = L"Настройки";

//

// panel7

//

this->panel7->Controls->Add(this->panel8);

this->panel7->Controls->Add(this->label18);

this->panel7->Location = System::Drawing::Point(292, 52);

this->panel7->Name = L"panel7";

this->panel7->Size = System::Drawing::Size(181, 57);

this->panel7->TabIndex = 46;

this->panel7->Click += gcnew System::EventHandler(this, &IntegralSolver::panel7\_Click);

this->panel7->MouseEnter += gcnew System::EventHandler(this, &IntegralSolver::panel7\_MouseEnter);

this->panel7->MouseLeave += gcnew System::EventHandler(this, &IntegralSolver::panel7\_MouseLeave);

//

// panel8

//

this->panel8->Location = System::Drawing::Point(0, 47);

this->panel8->Name = L"panel8";

this->panel8->Size = System::Drawing::Size(181, 10);

this->panel8->TabIndex = 46;

//

// label18

//

this->label18->AutoSize = true;

this->label18->BackColor = System::Drawing::Color::Transparent;

this->label18->Enabled = false;

this->label18->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label18->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(208)), static\_cast<System::Int32>(static\_cast<System::Byte>(219)),

static\_cast<System::Int32>(static\_cast<System::Byte>(250)));

this->label18->Location = System::Drawing::Point(51, 17);

this->label18->Name = L"label18";

this->label18->Size = System::Drawing::Size(81, 23);

this->label18->TabIndex = 0;

this->label18->Text = L"График";

//

// panel9

//

this->panel9->Controls->Add(this->panel10);

this->panel9->Controls->Add(this->label19);

this->panel9->Location = System::Drawing::Point(472, 52);

this->panel9->Name = L"panel9";

this->panel9->Size = System::Drawing::Size(181, 57);

this->panel9->TabIndex = 47;

this->panel9->Click += gcnew System::EventHandler(this, &IntegralSolver::panel9\_Click);

this->panel9->MouseEnter += gcnew System::EventHandler(this, &IntegralSolver::panel9\_MouseEnter);

this->panel9->MouseLeave += gcnew System::EventHandler(this, &IntegralSolver::panel9\_MouseLeave);

//

// panel10

//

this->panel10->Location = System::Drawing::Point(0, 47);

this->panel10->Name = L"panel10";

this->panel10->Size = System::Drawing::Size(181, 10);

this->panel10->TabIndex = 46;

//

// label19

//

this->label19->AutoSize = true;

this->label19->BackColor = System::Drawing::Color::Transparent;

this->label19->Enabled = false;

this->label19->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label19->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(208)), static\_cast<System::Int32>(static\_cast<System::Byte>(219)),

static\_cast<System::Int32>(static\_cast<System::Byte>(250)));

this->label19->Location = System::Drawing::Point(46, 17);

this->label19->Name = L"label19";

this->label19->Size = System::Drawing::Size(87, 23);

this->label19->TabIndex = 0;

this->label19->Text = L"Таблица";

//

// panel11

//

this->panel11->Controls->Add(this->panel12);

this->panel11->Controls->Add(this->label20);

this->panel11->Location = System::Drawing::Point(651, 52);

this->panel11->Name = L"panel11";

this->panel11->Size = System::Drawing::Size(181, 57);

this->panel11->TabIndex = 48;

this->panel11->Click += gcnew System::EventHandler(this, &IntegralSolver::panel11\_Click);

this->panel11->MouseEnter += gcnew System::EventHandler(this, &IntegralSolver::panel11\_MouseEnter);

this->panel11->MouseLeave += gcnew System::EventHandler(this, &IntegralSolver::panel11\_MouseLeave);

//

// panel12

//

this->panel12->Location = System::Drawing::Point(0, 47);

this->panel12->Name = L"panel12";

this->panel12->Size = System::Drawing::Size(181, 10);

this->panel12->TabIndex = 46;

//

// label20

//

this->label20->AutoSize = true;

this->label20->BackColor = System::Drawing::Color::Transparent;

this->label20->Enabled = false;

this->label20->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 14.25F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label20->ForeColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(208)), static\_cast<System::Int32>(static\_cast<System::Byte>(219)),

static\_cast<System::Int32>(static\_cast<System::Byte>(250)));

this->label20->Location = System::Drawing::Point(49, 17);

this->label20->Name = L"label20";

this->label20->Size = System::Drawing::Size(79, 23);

this->label20->TabIndex = 0;

this->label20->Text = L"Импорт";

//

// pictureBox1

//

this->pictureBox1->BackgroundImage = (cli::safe\_cast<System::Drawing::Image^>(resources->GetObject(L"pictureBox1.BackgroundImage")));

this->pictureBox1->BackgroundImageLayout = System::Windows::Forms::ImageLayout::Stretch;

this->pictureBox1->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->pictureBox1->Location = System::Drawing::Point(12, 29);

this->pictureBox1->Name = L"pictureBox1";

this->pictureBox1->Size = System::Drawing::Size(80, 80);

this->pictureBox1->TabIndex = 49;

this->pictureBox1->TabStop = false;

//

// panel13

//

this->panel13->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->panel13->Controls->Add(this->panel19);

this->panel13->Controls->Add(this->panel1);

this->panel13->Controls->Add(this->label2);

this->panel13->Controls->Add(this->label3);

this->panel13->Controls->Add(this->label6);

this->panel13->Controls->Add(this->textBox1);

this->panel13->Controls->Add(this->textBox4);

this->panel13->Controls->Add(this->label4);

this->panel13->Controls->Add(this->checkBox5);

this->panel13->Controls->Add(this->textBox2);

this->panel13->Controls->Add(this->label5);

this->panel13->Controls->Add(this->textBox3);

this->panel13->Controls->Add(this->label8);

this->panel13->Controls->Add(this->comboBox1);

this->panel13->Controls->Add(this->label9);

this->panel13->Controls->Add(this->progressBar1);

this->panel13->Location = System::Drawing::Point(12, 115);

this->panel13->Name = L"panel13";

this->panel13->Size = System::Drawing::Size(820, 445);

this->panel13->TabIndex = 50;

//

// panel19

//

this->panel19->Controls->Add(this->checkBox4);

this->panel19->Controls->Add(this->label7);

this->panel19->Controls->Add(this->checkBox1);

this->panel19->Controls->Add(this->checkBox3);

this->panel19->Controls->Add(this->checkBox2);

this->panel19->Location = System::Drawing::Point(44, 18);

this->panel19->Name = L"panel19";

this->panel19->Size = System::Drawing::Size(302, 220);

this->panel19->TabIndex = 44;

//

// panel14

//

this->panel14->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->panel14->Controls->Add(this->richTextBox1);

this->panel14->Controls->Add(this->panel3);

this->panel14->Location = System::Drawing::Point(838, 52);

this->panel14->Name = L"panel14";

this->panel14->Size = System::Drawing::Size(256, 508);

this->panel14->TabIndex = 51;

//

// panel15

//

this->panel15->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->panel15->Controls->Add(this->chart1);

this->panel15->Location = System::Drawing::Point(12, 574);

this->panel15->Name = L"panel15";

this->panel15->Size = System::Drawing::Size(820, 445);

this->panel15->TabIndex = 52;

//

// panel16

//

this->panel16->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->panel16->Controls->Add(this->dataGridView5);

this->panel16->Controls->Add(this->dataGridView1);

this->panel16->Controls->Add(this->label13);

this->panel16->Controls->Add(this->dataGridView3);

this->panel16->Controls->Add(this->label15);

this->panel16->Controls->Add(this->dataGridView4);

this->panel16->Controls->Add(this->label16);

this->panel16->Controls->Add(this->label17);

this->panel16->Location = System::Drawing::Point(841, 574);

this->panel16->Name = L"panel16";

this->panel16->Size = System::Drawing::Size(820, 445);

this->panel16->TabIndex = 53;

//

// dataGridView5

//

this->dataGridView5->BackgroundColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)),

static\_cast<System::Int32>(static\_cast<System::Byte>(15)), static\_cast<System::Int32>(static\_cast<System::Byte>(32)));

this->dataGridView5->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;

this->dataGridView5->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ >(2) {

this->dataGridViewTextBoxColumn7,

this->dataGridViewTextBoxColumn8

});

this->dataGridView5->Location = System::Drawing::Point(21, 266);

this->dataGridView5->Name = L"dataGridView5";

this->dataGridView5->ReadOnly = true;

this->dataGridView5->ScrollBars = System::Windows::Forms::ScrollBars::Vertical;

this->dataGridView5->Size = System::Drawing::Size(353, 165);

this->dataGridView5->TabIndex = 42;

//

// dataGridViewTextBoxColumn7

//

this->dataGridViewTextBoxColumn7->HeaderText = L"X";

this->dataGridViewTextBoxColumn7->Name = L"dataGridViewTextBoxColumn7";

this->dataGridViewTextBoxColumn7->ReadOnly = true;

this->dataGridViewTextBoxColumn7->Width = 150;

//

// dataGridViewTextBoxColumn8

//

this->dataGridViewTextBoxColumn8->HeaderText = L"Y";

this->dataGridViewTextBoxColumn8->Name = L"dataGridViewTextBoxColumn8";

this->dataGridViewTextBoxColumn8->ReadOnly = true;

this->dataGridViewTextBoxColumn8->Width = 200;

//

// panel17

//

this->panel17->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->panel17->Controls->Add(this->label21);

this->panel17->Controls->Add(this->panel18);

this->panel17->Location = System::Drawing::Point(1100, 115);

this->panel17->Name = L"panel17";

this->panel17->Size = System::Drawing::Size(820, 445);

this->panel17->TabIndex = 54;

//

// label21

//

this->label21->AutoSize = true;

this->label21->BackColor = System::Drawing::Color::Transparent;

this->label21->Font = (gcnew System::Drawing::Font(L"Microsoft Tai Le", 18, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,

static\_cast<System::Byte>(0)));

this->label21->ForeColor = System::Drawing::Color::White;

this->label21->Location = System::Drawing::Point(80, 18);

this->label21->Name = L"label21";

this->label21->Size = System::Drawing::Size(228, 31);

this->label21->TabIndex = 32;

this->label21->Text = L"Импорт в Эксель";

//

// panel18

//

this->panel18->BorderStyle = System::Windows::Forms::BorderStyle::FixedSingle;

this->panel18->Controls->Add(this->label12);

this->panel18->Controls->Add(this->checkBox7);

this->panel18->Controls->Add(this->panel2);

this->panel18->Controls->Add(this->checkBox8);

this->panel18->Controls->Add(this->checkBox10);

this->panel18->Controls->Add(this->checkBox9);

this->panel18->Location = System::Drawing::Point(21, 74);

this->panel18->Name = L"panel18";

this->panel18->Size = System::Drawing::Size(354, 339);

this->panel18->TabIndex = 31;

//

// timer1

//

this->timer1->Interval = 1;

this->timer1->Tick += gcnew System::EventHandler(this, &IntegralSolver::timer1\_Tick);

//

// timer2

//

this->timer2->Interval = 1;

this->timer2->Tick += gcnew System::EventHandler(this, &IntegralSolver::timer2\_Tick);

//

// timer3

//

this->timer3->Interval = 1;

this->timer3->Tick += gcnew System::EventHandler(this, &IntegralSolver::timer3\_Tick);

//

// timer4

//

this->timer4->Interval = 1;

this->timer4->Tick += gcnew System::EventHandler(this, &IntegralSolver::timer4\_Tick);

//

// IntegralSolver

//

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

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

this->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(17)), static\_cast<System::Int32>(static\_cast<System::Byte>(23)),

static\_cast<System::Int32>(static\_cast<System::Byte>(46)));

this->ClientSize = System::Drawing::Size(1921, 1028);

this->ControlBox = false;

this->Controls->Add(this->panel17);

this->Controls->Add(this->panel16);

this->Controls->Add(this->panel15);

this->Controls->Add(this->panel14);

this->Controls->Add(this->panel13);

this->Controls->Add(this->pictureBox1);

this->Controls->Add(this->panel11);

this->Controls->Add(this->panel9);

this->Controls->Add(this->panel7);

this->Controls->Add(this->panel5);

this->Controls->Add(this->panel4);

this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::None;

this->Name = L"IntegralSolver";

this->Text = L"IntegralSolver";

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

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

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->EndInit();

this->panel1->ResumeLayout(false);

this->panel1->PerformLayout();

this->panel2->ResumeLayout(false);

this->panel2->PerformLayout();

this->panel3->ResumeLayout(false);

this->panel3->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView3))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView4))->EndInit();

this->panel4->ResumeLayout(false);

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox4))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox3))->EndInit();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox2))->EndInit();

this->panel5->ResumeLayout(false);

this->panel5->PerformLayout();

this->panel7->ResumeLayout(false);

this->panel7->PerformLayout();

this->panel9->ResumeLayout(false);

this->panel9->PerformLayout();

this->panel11->ResumeLayout(false);

this->panel11->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->EndInit();

this->panel13->ResumeLayout(false);

this->panel13->PerformLayout();

this->panel19->ResumeLayout(false);

this->panel19->PerformLayout();

this->panel14->ResumeLayout(false);

this->panel15->ResumeLayout(false);

this->panel16->ResumeLayout(false);

this->panel16->PerformLayout();

(cli::safe\_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView5))->EndInit();

this->panel17->ResumeLayout(false);

this->panel17->PerformLayout();

this->panel18->ResumeLayout(false);

this->panel18->PerformLayout();

this->ResumeLayout(false);

}

#pragma endregion

private: System::Void checkBox1\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (checkBox1->Checked) {

Euler = true;

richTextBox1->Text += L"Выбран метод Эйлера\n";

}

else {

Euler = false;

richTextBox1->Text += L"Убран метод Эйлера\n";

}

}

private: System::Void checkBox2\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (checkBox2->Checked) {

RungeKutta2 = true;

richTextBox1->Text += L"Выбран метод Рунге-Кутта 2 порядка\n";

}

else {

RungeKutta2 = false;

richTextBox1->Text += L"Убран метод Рунге-Кутта 2 порядка\n";

}

}

private: System::Void checkBox3\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (checkBox3->Checked) {

RungeKutta4 = true;

richTextBox1->Text += L"Выбран метод Рунге-Кутта 4 порядка\n";

}

else {

RungeKutta4 = false;

richTextBox1->Text += L"Убран метод Рунге-Кутта 4 порядка \n";

}

}

private: System::Void checkBox4\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

if (checkBox4->Checked) {

RungeKuttaFehlberg = true;

richTextBox1->Text += L"Выбран метод Рунге-Кутта-Фельберга\n";

}

else {

RungeKuttaFehlberg = false;

richTextBox1->Text += L"Убран метод Рунге-Кутта-Фельберга\n";

}

}

private: System::Void panel3\_Click(System::Object^ sender, System::EventArgs^ e) {

richTextBox1->Text = L"";

}

private: System::Void panel1\_Click(System::Object^ sender, System::EventArgs^ e) {

bool f = true;

richTextBox1->Text += L"Рассчёт начался\n";

for (int i = 0; i < 4; i++) {

chart1->Series[i]->Points->Clear(); //Очистка графика

}

dataGridView1->Rows->Clear(); //Очистка таблиц

dataGridView3->Rows->Clear();

dataGridView4->Rows->Clear();

dataGridView5->Rows->Clear();

progressBar1->Value = 0; //Очистка прогресса

marshal\_context^ marshal = gcnew marshal\_context();

if (textBox3->Text != "") accuracy = atof(marshal->marshal\_as<const char\*>(textBox3->Text));

else f = false;

if (textBox1->Text != "") x0 = atof(marshal->marshal\_as<const char\*>(textBox1->Text));

else f = false;

if (textBox2->Text != "") y0 = atof(marshal->marshal\_as<const char\*>(textBox2->Text));

else f = false;

int maximum\_progress = 0;

if (Euler) maximum\_progress += 25;

if (RungeKutta2) maximum\_progress += 25;

if (RungeKutta4) maximum\_progress += 25;

if (RungeKuttaFehlberg) maximum\_progress += 25;

progressBar1->Maximum = maximum\_progress;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

if (f) {

result = y0;

if (Euler) {

for (int i = 0; i < 11; i++)

{

double x = 0.1 \* i;

if (checkBox5->Checked)

result = ODE\_Euler(UserFunc, x0, result, accuracy, x);

else

result = ODE\_Euler(Func, x0, result, accuracy, x);

if (i % 5 == 0) {

chart1->Series[0]->Points->Add(x, result);

}

dataGridView1->Rows->Add();

dataGridView1->Rows[i]->Cells[0]->Value = x;

dataGridView1->Rows[i]->Cells[1]->Value = result;

x0 = x;

progressBar1->Value += 1;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

progressBar1->Value += 14;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

result = y0;

if (RungeKutta2) {

for (int i = 0; i < 11; i++)

{

double x = 0.1 \* i;

if (checkBox5->Checked)

result = ODE\_RungeKutta2(UserFunc, x0, result, accuracy, x);

else

result = ODE\_RungeKutta2(Func, x0, result, accuracy, x);

if (i % 5 == 0) {

chart1->Series[1]->Points->Add(x, result);

}

dataGridView3->Rows->Add();

dataGridView3->Rows[i]->Cells[0]->Value = x;

dataGridView3->Rows[i]->Cells[1]->Value = result;

x0 = x;

progressBar1->Value += 1;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

progressBar1->Value += 14;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

result = y0;

if (RungeKutta4) {

for (int i = 0; i < 11; i++)

{

double x = 0.1 \* i;

if (checkBox5->Checked)

result = ODE\_RungeKutta4(UserFunc, x0, result, accuracy, x);

else

result = ODE\_RungeKutta4(Func, x0, result, accuracy, x);

if (i % 5 == 0) {

chart1->Series[2]->Points->Add(x, result);

}

dataGridView4->Rows->Add();

dataGridView4->Rows[i]->Cells[0]->Value = x;

dataGridView4->Rows[i]->Cells[1]->Value = result;

x0 = x;

progressBar1->Value += 1;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

progressBar1->Value += 14;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

result = y0;

if (RungeKuttaFehlberg) {

for (int i = 0; i < 11; i++)

{

double x = 0.1 \* i;

if (checkBox5->Checked)

result = ODE\_RungeKuttaFehlberg(UserFunc, x0, result, x, accuracy, 1e-8);

else

result = ODE\_RungeKuttaFehlberg(Func, x0, result, x, accuracy, 1e-8);

if (i % 5 == 0) {

chart1->Series[3]->Points->Add(x, result);

}

dataGridView5->Rows->Add();

dataGridView5->Rows[i]->Cells[0]->Value = x;

dataGridView5->Rows[i]->Cells[1]->Value = result;

x0 = x;

progressBar1->Value += 1;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

progressBar1->Value += 14;

std::string val = std::to\_string(progressBar1->Value);

String^ val1 = gcnew String(val.c\_str());

label9->Text = val1;

}

label9->Text = "100%";

}

else {

richTextBox1->Text += L"Ошибка, не заполненый все формы!\n";

}

richTextBox1->Text += L"Рассчёт окончен\n";

}

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

richTextBox1->Text += L"Формула изменена на " + comboBox1->Text + "\n";

}

private: System::Void textBox4\_TextChanged(System::Object^ sender, System::EventArgs^ e) {

richTextBox1->Text += L"Формула изменена на " + textBox4->Text + "\n";

}

private: System::Void checkBox5\_CheckedChanged(System::Object^ sender, System::EventArgs^ e) {

richTextBox1->Text += L"Выбрана пользовательская функция\n";

if (checkBox5->Checked) {

MessageBox::Show("Пользовательская функция недоступна для метода Рунге-Кутта-Фельберга", "Внимание!", MessageBoxButtons::OK, MessageBoxIcon::Warning);

richTextBox1->Text += L"!!!Пользовательская функция недоступна для метода Рунге-Кутта-Фельберга!!!\n";

checkBox4->Checked = false;

checkBox4->Enabled = false;

}

else {

checkBox4->Checked = true;

checkBox4->Enabled = true;

}

}

private: System::Void panel2\_Click(System::Object^ sender, System::EventArgs^ e) {

richTextBox1->Text += L"Началась выгрузка данных\n";

marshal\_context^ marshal = gcnew marshal\_context();

if (checkBox10->Checked) {

Sheet\* sheet1 = Data->addSheet(L"Euler");

sheet1->writeStr(1, 0, L"X");

sheet1->writeStr(1, 1, L"Y");

for (int i = 0; i < dataGridView1->Rows->Count - 1; i++) {

std::string buffer;

buffer = marshal->marshal\_as<const char\*>(dataGridView1->Rows[i]->Cells[0]->Value->ToString());

std::wstring buffer1;

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 0, buffer1.c\_str());

buffer1 = L"";

buffer = marshal->marshal\_as<const char\*>(dataGridView1->Rows[i]->Cells[1]->Value->ToString());

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 1, buffer1.c\_str());

}

richTextBox1->Text += L"Данные Метода Эйлера успешно выгружены!\n";

}

if (checkBox9->Checked) {

Sheet\* sheet1 = Data->addSheet(L"RungeKutta2");

sheet1->writeStr(1, 0, L"X");

sheet1->writeStr(1, 1, L"Y");

for (int i = 0; i < dataGridView3->Rows->Count - 1; i++) {

std::string buffer;

buffer = marshal->marshal\_as<const char\*>(dataGridView3->Rows[i]->Cells[0]->Value->ToString());

std::wstring buffer1;

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 0, buffer1.c\_str());

buffer1 = L"";

buffer = marshal->marshal\_as<const char\*>(dataGridView3->Rows[i]->Cells[1]->Value->ToString());

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 1, buffer1.c\_str());

}

richTextBox1->Text += L"Данные Метода Рунге-Кутта 2 порядка успешно выгружены!\n";

}

if (checkBox8->Checked) {

Sheet\* sheet1 = Data->addSheet(L"RungeKutta4");

sheet1->writeStr(1, 0, L"X");

sheet1->writeStr(1, 1, L"Y");

for (int i = 0; i < dataGridView4->Rows->Count - 1; i++) {

std::string buffer;

buffer = marshal->marshal\_as<const char\*>(dataGridView4->Rows[i]->Cells[0]->Value->ToString());

std::wstring buffer1;

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 0, buffer1.c\_str());

buffer1 = L"";

buffer = marshal->marshal\_as<const char\*>(dataGridView4->Rows[i]->Cells[1]->Value->ToString());

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 1, buffer1.c\_str());

}

richTextBox1->Text += L"Данные Метода Рунге-Кутта 4 порядка успешно выгружены!\n";

}

if (checkBox7->Checked) {

Sheet\* sheet1 = Data->addSheet(L"RungeKuttaFehlberg");

sheet1->writeStr(1, 0, L"X");

sheet1->writeStr(1, 1, L"Y");

for (int i = 0; i < dataGridView5->Rows->Count - 1; i++) {

std::string buffer;

buffer = marshal->marshal\_as<const char\*>(dataGridView5->Rows[i]->Cells[0]->Value->ToString());

std::wstring buffer1;

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 0, buffer1.c\_str());

buffer1 = L"";

buffer = marshal->marshal\_as<const char\*>(dataGridView5->Rows[i]->Cells[1]->Value->ToString());

for (int i = 0; i < buffer.length(); ++i)

buffer1 += wchar\_t(buffer[i]);

sheet1->writeStr(i + 2, 1, buffer1.c\_str());

}

richTextBox1->Text += L"Данные Метода Рунге-Кутта-Фельберга порядка успешно выгружены!\n";

}

Data->save(L"Data.xls");

Data->release();

richTextBox1->Text += L"Выгрузка данных завершена!\n";

}

private: System::Void panel5\_Click(System::Object^ sender, System::EventArgs^ e) { //Настройки

panel13->Location = System::Drawing::Point(12, 115);

panel15->Location = System::Drawing::Point(12, 574);

panel16->Location = System::Drawing::Point(841, 574);

panel17->Location = System::Drawing::Point(1100, 115);

}

private: System::Void panel7\_Click(System::Object^ sender, System::EventArgs^ e) { //График

panel15->Location = System::Drawing::Point(12, 115);

panel13->Location = System::Drawing::Point(12, 574);

panel16->Location = System::Drawing::Point(841, 574);

panel17->Location = System::Drawing::Point(1100, 115);

}

private: System::Void panel9\_Click(System::Object^ sender, System::EventArgs^ e) { //Таблица

panel16->Location = System::Drawing::Point(12, 115);

panel13->Location = System::Drawing::Point(841, 574);

panel15->Location = System::Drawing::Point(12, 574);

panel17->Location = System::Drawing::Point(1100, 115);

}

private: System::Void panel11\_Click(System::Object^ sender, System::EventArgs^ e) { //Импорт

panel17->Location = System::Drawing::Point(12, 115);

panel15->Location = System::Drawing::Point(12, 574);

panel16->Location = System::Drawing::Point(841, 574);

panel13->Location = System::Drawing::Point(1100, 115);

}

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

this->Size = System::Drawing::Size(1100, 570);

}

private: System::Void pictureBox2\_Click(System::Object^ sender, System::EventArgs^ e) {

Close();

}

private: System::Void pictureBox3\_Click(System::Object^ sender, System::EventArgs^ e) {

MessageBox::Show("Разворот в полноэкранный режим недоступен, эта кнопка просто декоративная.", "Извиняемся!", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

private: System::Void pictureBox4\_Click(System::Object^ sender, System::EventArgs^ e) {

this->WindowState = FormWindowState::Minimized;

}

private: System::Void Panel\_MouseDown(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

this->dragging = true;

this->offset = Point(e->X, e->Y);

}

private: bool dragging;

private: Point offset;

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

this->dragging = false;

}

private: System::Void Panel\_MouseMove(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

if (this->dragging) {

Point currentScreenPos = PointToScreen(e->Location);

Location = Point(currentScreenPos.X - this->offset.X, currentScreenPos.Y - this->offset.Y);

}

};

private: System::Void Panel\_MouseUp(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e) {

this->dragging = false;

}

private: System::Void panel5\_MouseEnter(System::Object^ sender, System::EventArgs^ e) {

panel5->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(25)), static\_cast<System::Int32>(static\_cast<System::Byte>(31)),

static\_cast<System::Int32>(static\_cast<System::Byte>(54)));

back1 = false;

timer1->Enabled = true;

}

private: System::Void panel5\_MouseLeave(System::Object^ sender, System::EventArgs^ e) {

panel5->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(17)), static\_cast<System::Int32>(static\_cast<System::Byte>(23)),

static\_cast<System::Int32>(static\_cast<System::Byte>(46)));

back1 = true;

timer1->Enabled = true;

}

private: System::Void timer1\_Tick(System::Object^ sender, System::EventArgs^ e) {

if (back1) {

if (r1 != 17) r1+=2;

if (g1 != 23) g1-=19;

if (b1 != 46) b1-=15;

}

else {

if (r1 != 15) r1-=2;

if (g1 != 99) g1+=19;

if (b1 != 196) b1+=15;

}

panel6->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(r1)), static\_cast<System::Int32>(static\_cast<System::Byte>(g1)),

static\_cast<System::Int32>(static\_cast<System::Byte>(b1)));

if (r1 == 15 && g1 == 99 && b1 == 196) timer1->Enabled = false;

if (r1 == 17 && g1 == 23 && b1 == 46) timer1->Enabled = false;

}

private: System::Void timer2\_Tick(System::Object^ sender, System::EventArgs^ e) {

if (back2) {

if (r2 != 17) r2 += 2;

if (g2 != 23) g2 -= 19;

if (b2 != 46) b2 -= 15;

}

else {

if (r2 != 15) r2 -= 2;

if (g2 != 99) g2 += 19;

if (b2 != 196) b2 += 15;

}

panel8->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(r2)), static\_cast<System::Int32>(static\_cast<System::Byte>(g2)),

static\_cast<System::Int32>(static\_cast<System::Byte>(b2)));

if (r2 == 15 && g2 == 99 && b2 == 196) timer2->Enabled = false;

if (r2 == 17 && g2 == 23 && b2 == 46) timer2->Enabled = false;

}

private: System::Void timer3\_Tick(System::Object^ sender, System::EventArgs^ e) {

if (back3) {

if (r3 != 17) r3 += 2;

if (g3 != 23) g3 -= 19;

if (b3 != 46) b3 -= 15;

}

else {

if (r3 != 15) r3 -= 2;

if (g3 != 99) g3 += 19;

if (b3 != 196) b3 += 15;

}

panel10->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(r3)), static\_cast<System::Int32>(static\_cast<System::Byte>(g3)),

static\_cast<System::Int32>(static\_cast<System::Byte>(b3)));

if (r3 == 15 && g3 == 99 && b3 == 196) timer3->Enabled = false;

if (r3 == 17 && g3 == 23 && b3 == 46) timer3->Enabled = false;

}

private: System::Void timer4\_Tick(System::Object^ sender, System::EventArgs^ e) {

if (back4) {

if (r4 != 17) r4 += 2;

if (g4 != 23) g4 -= 19;

if (b4 != 46) b4 -= 15;

}

else {

if (r4 != 15) r4 -= 2;

if (g4 != 99) g4 += 19;

if (b4 != 196) b4 += 15;

}

panel12->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(r4)), static\_cast<System::Int32>(static\_cast<System::Byte>(g4)),

static\_cast<System::Int32>(static\_cast<System::Byte>(b4)));

if (r4 == 15 && g4 == 99 && b4 == 196) timer4->Enabled = false;

if (r4 == 17 && g4 == 23 && b4 == 46) timer4->Enabled = false;

}

private: System::Void panel7\_MouseEnter(System::Object^ sender, System::EventArgs^ e) {

panel7->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(25)), static\_cast<System::Int32>(static\_cast<System::Byte>(31)),

static\_cast<System::Int32>(static\_cast<System::Byte>(54)));

back2 = false;

timer2->Enabled = true;

}

private: System::Void panel7\_MouseLeave(System::Object^ sender, System::EventArgs^ e) {

panel7->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(17)), static\_cast<System::Int32>(static\_cast<System::Byte>(23)),

static\_cast<System::Int32>(static\_cast<System::Byte>(46)));

back2 = true;

timer2->Enabled = true;

}

private: System::Void panel9\_MouseEnter(System::Object^ sender, System::EventArgs^ e) {

panel9->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(25)), static\_cast<System::Int32>(static\_cast<System::Byte>(31)),

static\_cast<System::Int32>(static\_cast<System::Byte>(54)));

back3 = false;

timer3->Enabled = true;

}

private: System::Void panel9\_MouseLeave(System::Object^ sender, System::EventArgs^ e) {

panel9->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(17)), static\_cast<System::Int32>(static\_cast<System::Byte>(23)),

static\_cast<System::Int32>(static\_cast<System::Byte>(46)));

back3 = true;

timer3->Enabled = true;

}

private: System::Void panel11\_MouseEnter(System::Object^ sender, System::EventArgs^ e) {

panel11->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(25)), static\_cast<System::Int32>(static\_cast<System::Byte>(31)),

static\_cast<System::Int32>(static\_cast<System::Byte>(54)));

back4 = false;

timer4->Enabled = true;

}

private: System::Void panel11\_MouseLeave(System::Object^ sender, System::EventArgs^ e) {

panel11->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(17)), static\_cast<System::Int32>(static\_cast<System::Byte>(23)),

static\_cast<System::Int32>(static\_cast<System::Byte>(46)));

back4 = true;

timer4->Enabled = true;

}

private: System::Void panel1\_MouseEnter(System::Object^ sender, System::EventArgs^ e) {

panel1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)), static\_cast<System::Int32>(static\_cast<System::Byte>(100)),

static\_cast<System::Int32>(static\_cast<System::Byte>(230)));

}

private: System::Void panel1\_MouseLeave(System::Object^ sender, System::EventArgs^ e) {

panel1->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(3)), static\_cast<System::Int32>(static\_cast<System::Byte>(93)),

static\_cast<System::Int32>(static\_cast<System::Byte>(223)));

}

private: System::Void panel2\_MouseEnter(System::Object^ sender, System::EventArgs^ e) {

panel2->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(10)), static\_cast<System::Int32>(static\_cast<System::Byte>(100)),

static\_cast<System::Int32>(static\_cast<System::Byte>(230)));

}

private: System::Void panel2\_MouseLeave(System::Object^ sender, System::EventArgs^ e) {

panel2->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(3)), static\_cast<System::Int32>(static\_cast<System::Byte>(93)),

static\_cast<System::Int32>(static\_cast<System::Byte>(223)));

}

private: System::Void panel3\_MouseEnter(System::Object^ sender, System::EventArgs^ e) {

panel3->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(50)), static\_cast<System::Int32>(static\_cast<System::Byte>(67)),

static\_cast<System::Int32>(static\_cast<System::Byte>(122)));

}

private: System::Void panel3\_MouseLeave(System::Object^ sender, System::EventArgs^ e) {

panel3->BackColor = System::Drawing::Color::FromArgb(static\_cast<System::Int32>(static\_cast<System::Byte>(43)), static\_cast<System::Int32>(static\_cast<System::Byte>(60)),

static\_cast<System::Int32>(static\_cast<System::Byte>(115)));

}

};

double Func(double x, double y)

{

if (IntegralSolver::comboBox1->Text == "x \* y")

return x \* y;

if (IntegralSolver::comboBox1->Text == "y \* cos(x)")

return y \* cos(x);

if (IntegralSolver::comboBox1->Text == "sin(y) \* cos(x)")

return sin(y) \* cos(x);

if (IntegralSolver::comboBox1->Text == "1 / (x \* y - 1)")

return 1 / (x \* y - 1);

if (IntegralSolver::comboBox1->Text == "x \* x / (y \* y)")

return x \* x / (y \* y);

}

double dx(double x, double y, double z)

{

return 10.0 \* (y - x);

}

double dy(double x, double y, double z)

{

return x \* (28.0 - z) - y;

}

double dz(double x, double y, double z)

{

return x \* y - 8.0 \* z / 3.0;

}

double ODE\_Euler(std::function<double(double, double)> f, double x0, double y0, double h, double x)

{

double xnew, ynew, result;

if (x <= x0)

result = y0;

else if (x > x0)

{

do

{

if (h > x - x0) h = x - x0;

ynew = y0 + f(x0, y0) \* h;

xnew = x0 + h;

x0 = xnew;

y0 = ynew;

} while (x0 < x);

result = ynew;

}

return result;

}

double ODE\_RungeKutta2(std::function<double(double, double)> f, double x0, double y0, double h, double x)

{

double xnew, ynew, k1, k2, result = y0;

if (x == x0)

result = y0;

else if (x > x0)

{

do

{

if (h > x - x0) h = x - x0;

k1 = h \* f(x0, y0);

k2 = h \* f(x0 + 0.5 \* h, y0 + 0.5 \* k1);

ynew = y0 + k2;

xnew = x0 + h;

x0 = xnew;

y0 = ynew;

} while (x0 < x);

result = ynew;

}

return result;

}

double ODE\_RungeKutta4(std::function<double(double, double)> f, double x0, double y0, double h, double x)

{

double xnew, ynew, k1, k2, k3, k4, result = y0;

if (x == x0)

result = y0;

else if (x > x0)

{

do

{

if (h > x - x0) h = x - x0;

k1 = h \* f(x0, y0);

k2 = h \* f(x0 + 0.5 \* h, y0 + 0.5 \* k1);

k3 = h \* f(x0 + 0.5 \* h, y0 + 0.5 \* k2);

k4 = h \* f(x0 + h, y0 + k3);

ynew = y0 + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6;

xnew = x0 + h;

x0 = xnew;

y0 = ynew;

} while (x0 < x);

result = ynew;

}

return result;

}

double ODE\_RungeKuttaFehlberg(std::function<double(double, double)> f, double x0, double y0, double x, double h, double tolerance)

{

double xnew, ynew, hnew, k1, k2, k3, k4, k5, k6;

double hmin = 0.0001;

double hmax = 0.5;

if (h > hmax) h = hmax;

if (h < hmin) h = hmin;

while (x0 < x)

{

k1 = h \* f(x0, y0);

k2 = h \* f(x0 + 0.25 \* h, y0 + 0.25 \* k1);

k3 = h \* f(x0 + 3 \* h / 8, y0 + 3 \* k1 / 32 + 9 \* k2 / 32);

k4 = h \* f(x0 + 12 \* h / 13, y0 + 1932 \* k1 / 2197 - 7200 \* k2 / 2197 + 7296 \* k3 / 2197);

k5 = h \* f(x0 + h, y0 + 439 \* k1 / 216 - 8 \* k2 + 3680 \* k3 / 513 - 845 \* k4 / 4104);

k6 = h \* f(x0 + 0.5 \* h, y0 - 8 \* k1 / 27 + 2 \* k2 - 3544 \* k3 / 2565 + 1859 \* k4 / 4104 - 11 \* k5 / 40);

double error = abs(k1 / 360 - 128 \* k3 / 4275 - 2197 \* k4 / 75240 + k5 / 50 + 2 \* k6 / 55) / h;

double s = pow(0.5 \* tolerance / error, 0.25);

if (error < tolerance)

{

ynew = y0 + 25 \* k1 / 216 + 1408 \* k3 / 2565 + 2197 \* k4 / 4104 - 0.2 \* k5;

xnew = x0 + h;

x0 = xnew;

y0 = ynew;

}

if (s < 0.1) s = 0.1;

if (s > 4) s = 4;

hnew = h \* s;

h = hnew;

if (h > hmax) h = hmax;

if (h < hmin) h = hmin;

if (h > x - x0) h = x - x0;

} return y0;

}

char\* insert(char\* str, size\_t capacity, char\* subs, char c, int\* uerr)

{

int dummy, \* err = uerr ? uerr : &dummy;

\*err = 0;

char\* p = c ? strchr(str, c) : str - 1;

if (p) {

p++;

size\_t l2 = strlen(subs);

if (l2) {

size\_t l1 = strlen(str);

if (l1 + l2 > capacity)

return (\*err = 1, (char\*)0);

memmove(p + l2, p, l1 - (p - str) + 1);

memcpy(p, subs, l2);

}

}

return p;

}

double UserFunc(double x, double y) {

marshal\_context^ marshal = gcnew marshal\_context();

std::string buffer;

buffer = marshal->marshal\_as<const char\*>(IntegralSolver::textBox4->Text);

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

if (buffer[i] == 'x') {

buffer.erase(i, 1);

buffer.insert(i, std::to\_string(x));

}

if (buffer[i] == 'y') {

buffer.erase(i, 1);

buffer.insert(i, std::to\_string(y));

}

}

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

if (buffer[i] == ',') buffer[i] = '.';

}

char\* cstr = new char[buffer.length() + 1];

strcpy(cstr, buffer.c\_str());

eatspaces(cstr); //удаляем все пробелы из строки

return expr(cstr);

}

void eatspaces(char\* str)

{

int i = 0; //индекс места в строке "куда копировать"

int j = 0; //индекс места в строке "откуда копировать"

while ((\*(str + i) = \*(str + j++)) != '\0') //цикл, пока очередной символ не '\0'

if (\*(str + i) != ' ') //увеличиваем i, если символ не пробел

i++;

return;

}

double expr(char\* str)

{

double value = 0.0; //здесь сохраняем результат

int index = 0; //текущая позиция символа

value = term(str, index); //получить первый элемент

for (; ; ) //бесконечный цикл, выход внутри

{

switch (\*(str + index++)) //выбрать действие на основе текущего символа

{

case '\0': //конец строки, возвращаем значение

return value;

case '+': //знак плюс, прибавляем элемент к value

value += term(str, index);

break;

case '-': //знак минус, вычитаем элемент из value

value -= term(str, index);

break;

default: //все остальное не котируется

int i = index;

while (--i > 0)

MessageBox::Show("Неправильно задана формула.", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

}

}

double term(char\* str, int& index)

{

double value = 0.0; //здесь накапливается значение результата

value = power(str, index); //получить первое число элемента

//выполняем цикл до тех пор, пока имеем допустимую операцию

while ((\*(str + index) == '\*') || (\*(str + index) == '/'))

{

if (\*(str + index) == '\*')

value \*= power(str, ++index);

if (\*(str + index) == '/')

value /= power(str, ++index);

}

return value;

}

double power(char\* str, int& index)

{

double value = 0.0;

value = trigon(str, index);

while (\*(str + index) == '^')

{

value = pow(value, trigon(str, ++index)); //возводим в степень

}

return value;

}

double trigon(char\* str, int& index)

{

int buf\_index = 0;

int temp\_index = index; //переменная для хранения индекса (чтобы если что вернуть индекс без изменений)

char\* p\_str = 0; //временный указатель для сравнения символов

double value = 0;

while (isalpha(\*(str + temp\_index)))

{

buf\_index++; //сколько букв

temp\_index++; //текущий индекс

}

if (!buf\_index) //если нет ни одной буквы, то возвращаем число

{

value = number(str, index);

return value;

}

else //иначе смотрим, являются ли буквы чем-нибудь этим

{

p\_str = new char[buf\_index + 1]; //а для этого создаем временную строку, чтобы сравнить

p\_str[buf\_index] = '\0';

strncpy(p\_str, str + index, buf\_index);

}

if (strcmp(p\_str, "sin") == 0) //синус в градусах

{

value = sin(3.141592 / 180 \* number(str, temp\_index));

index = temp\_index;

delete[] p\_str; //не забываем удалить временную строку

return value;

}

else if (strcmp(p\_str, "cos") == 0) //косинус в градусах

{

value = cos(3.141592 / 180 \* number(str, temp\_index));

index = temp\_index;

delete[] p\_str; //не забываем удалить временную строку

return value;

}

else if (strcmp(p\_str, "tan") == 0) //тангенс в градусах

{

value = tan(3.141592 / 180 \* number(str, temp\_index));

index = temp\_index;

delete[] p\_str; //не забываем удалить временную строку

return value;

}

else

{

return value;

}

}

double number(char\* str, int& index)

{

double value = 0.0; //хранит результирующее значение

if (\*(str + index) == '(')

{

char\* p\_substr = 0;

p\_substr = extract(str, ++index);

value = expr(p\_substr);

delete[] p\_substr;

return value;

}

//продуманский цикл, превращает символы в число

while (isdigit(\*(str + index))) //цикл накапливает ведущие цифры

value = 10 \* value + (\*(str + index++) - '0');

if (\*(str + index) != '.') //если не цифра, проверяем на десятичную точку

return value;

double factor = 1.0; //множитель для десятичных разрядов

//еще один продуманский цикл, возвращает десятичную часть

while (isdigit(\*(str + (++index)))) //выполнять цикл, пока идут цифры

{

factor \*= 0.1;

value = value + (\*(str + index) - '0') \* factor;

}

return value;

}

char\* extract(char\* str, int& index)

{

char buffer[MAX]; //временное пространство для подстроки

char\* p\_str = 0; //указатель на новую строку для возврата

int numL = 0; //счетчик найденных левых скобок

int buf\_index = index; //сохранить начальное значение index

do

{

buffer[index - buf\_index] = \*(str + index); //копируем символ текущей строки в подстроку

switch (buffer[index - buf\_index]) //смотрим, чо это за символ

{

case ')':

if (numL == 0)

{

buffer[index - buf\_index] = '\0'; //если счетчик скобочек верный, ставим символ конца строки

++index; //устанавливаем индекс на следующий за скобочкой элемент

p\_str = new char[index - buf\_index];

if (!p\_str)

{

MessageBox::Show("Ошибка при выделении памяти.", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Warning);

}

strcpy\_s(p\_str, index - buf\_index, buffer); //и копируем подстроку в новую память

return p\_str;

}

else

numL--; //уменьшаем счетчик скобок

break;

case '(':

numL++; //соответственно увеличиваем

break;

}

} while (\*(str + index++) != '\0'); //устанавливаем индекс в следующий элемент

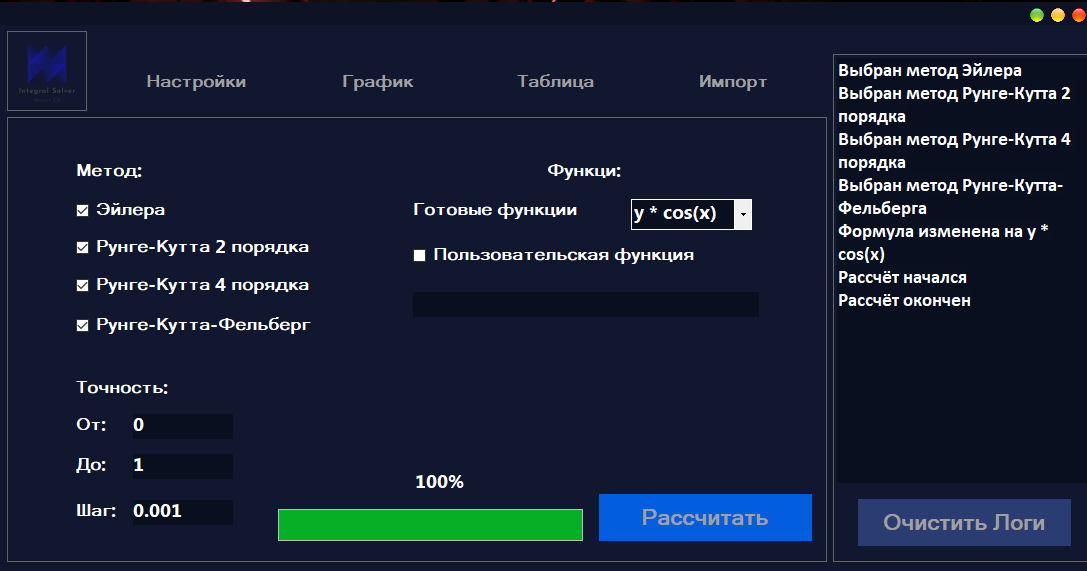
MessageBox::Show("Выход за пределы выражения, возможно, плохой ввод.", "Ошибка!", MessageBoxButtons::OK, MessageBoxIcon::Warning);

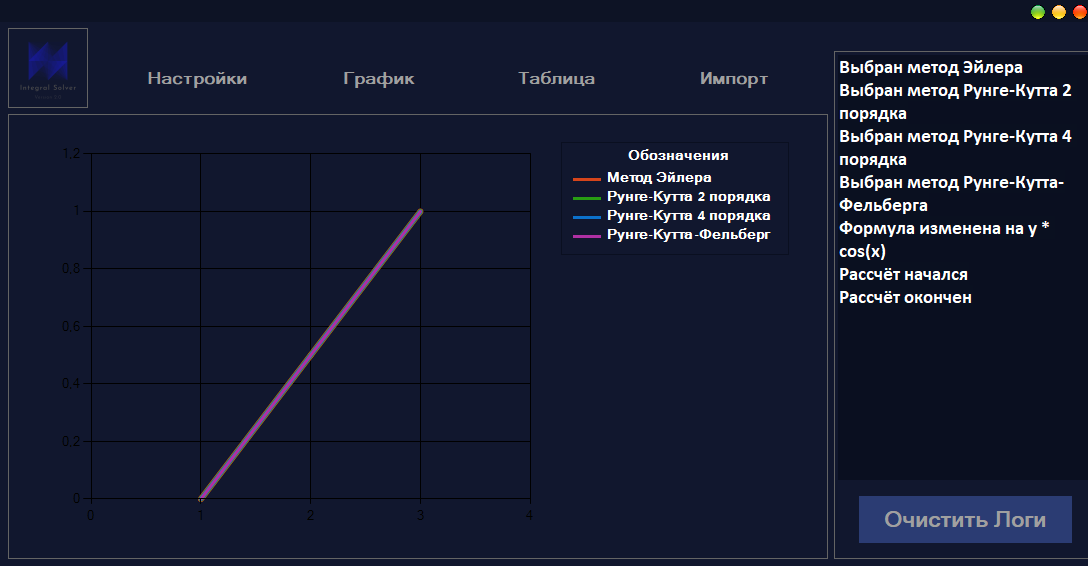
return p\_str;

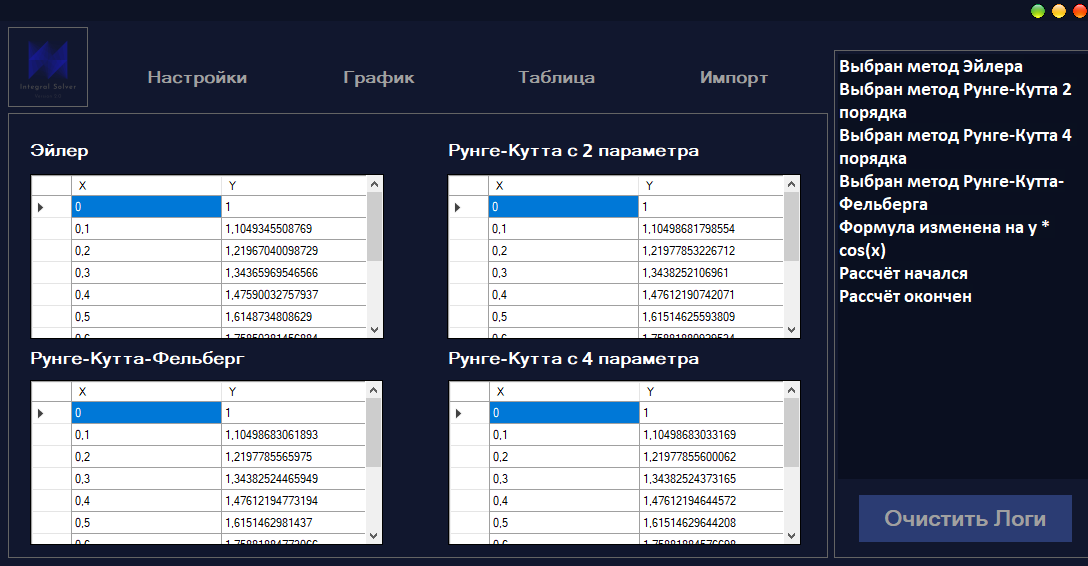
}

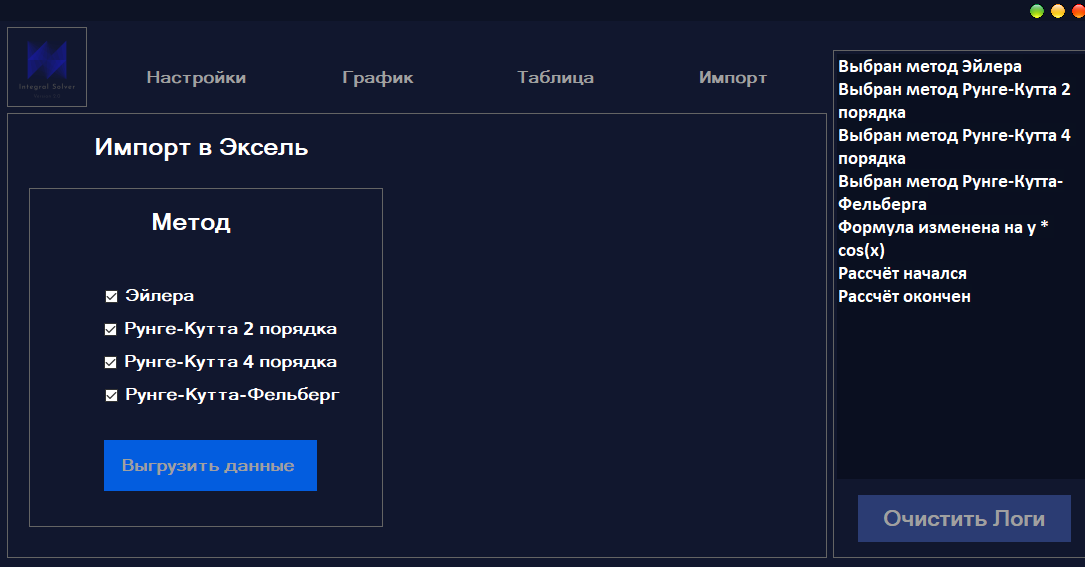
}

Работа программы









Заключение

Итогом нашей работы стала простая для понимания программа, позволяющая пользователю получить решение дифференциального уравнения любым из доступных методов. Так же представлен наглядный график зависимостей результата от методов решения дифференциальных уравнений и таблица с результатами каждого шага решения. Это позволит существенно сократить время вычислений и позволит подобрать наиболее подходящий способ вычислений. Мы видим наложение методов на графике, за счет этого мы можем сделать вывод, что результаты вычислений незначительно отличаются друг от друга.