МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

ордена Трудового Красного Знамени федеральное государственное

бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

Учебная практика

|  |  |
| --- | --- |
|  | Выполнил: студент гр. БЭИ2202  Кулешов А. С.  Вариант 130  Проверил: ст. преп, Юсков И. О. |

Москва, 2024 г.

**Оглавление**

[**Индивидуальное задание на разработку проекта** **3.**](#_Toc139350186)

[**Схемы алгоритмов** **5.**](#_Toc139350187)

[**Программный код** **10.**](#_Toc139350188)

[**Результаты работы программы** **15.**](#_Toc139350189)

1. Индивидуальное задание на разработку проекта

Цепь, висящая на гладком крюке, соскальзывает вниз. В начале движения по одну сторону крюка свисает 10м цепи, по другую - 8м.

Определить, за сколько времени с крюка соскользнёт вся цепь, какова будет скорость цепи в начальный момент её свободного падения

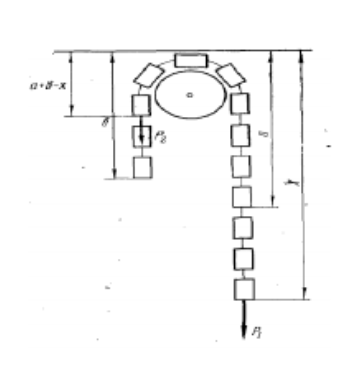


Рисунок 1 – схема цепи

Рассмотрим силу, применяемую к участку цепи s0, когда длина цепи равна s, . Таким образом можно прийти к дифференциальному уравнению, которое нам предстоит решить:

Решив данное уравнение, мы получаем функцию с двумя неизвестными константами:

Найти значение констант и можно, если использовать информацию о изначальных данных системы:

* + - 1. 0

Таким образом, мы и находим следующие константы:

Решить первую часть задания можно используя алгоритм двоичного поиска по известной нам монотонной функции s(t), чтобы найти такое время t, что s(t) = s0 + s1. Найдя соответствующее время t, в которое цепь находится в свободном падении нам лишь необходимо найти значение производной данной функции в найденный момент времени ().

Также перепроверим себя, решив задачу методом Рунге-Куута, найдя значения функции в некоторых точках функции.

И наконец, выведем результат вычислений в текстовый файл.

1. Схемы алгоритмов

На рисунке 2 изображена схема алгоритма бинарного поиска по функции.

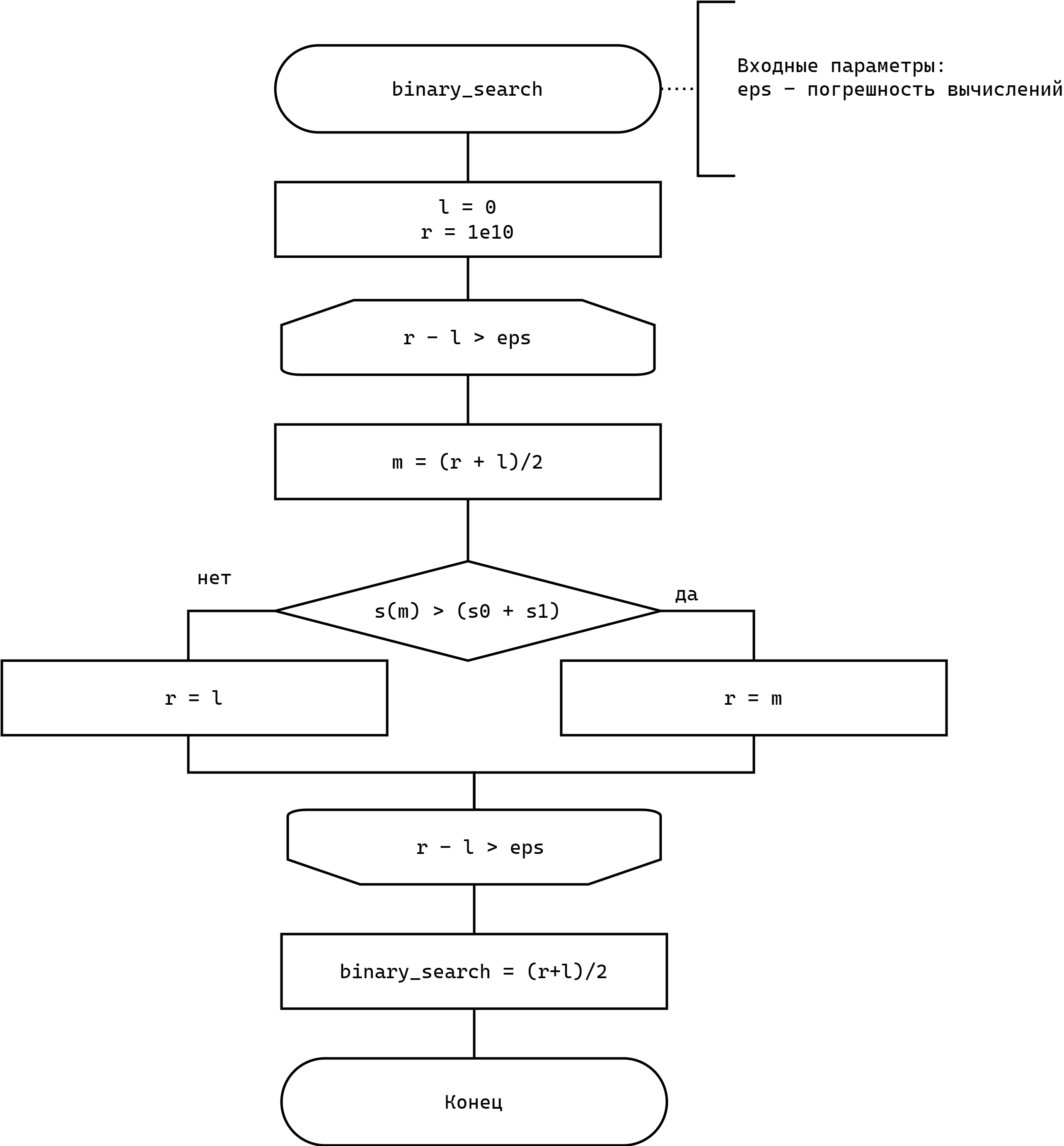


Рисунок 2 – схема алгоритма бинарного поиска

На рисунке 3 изображена схема алгоритма функции, являющейся решением дифференциального уравнения.

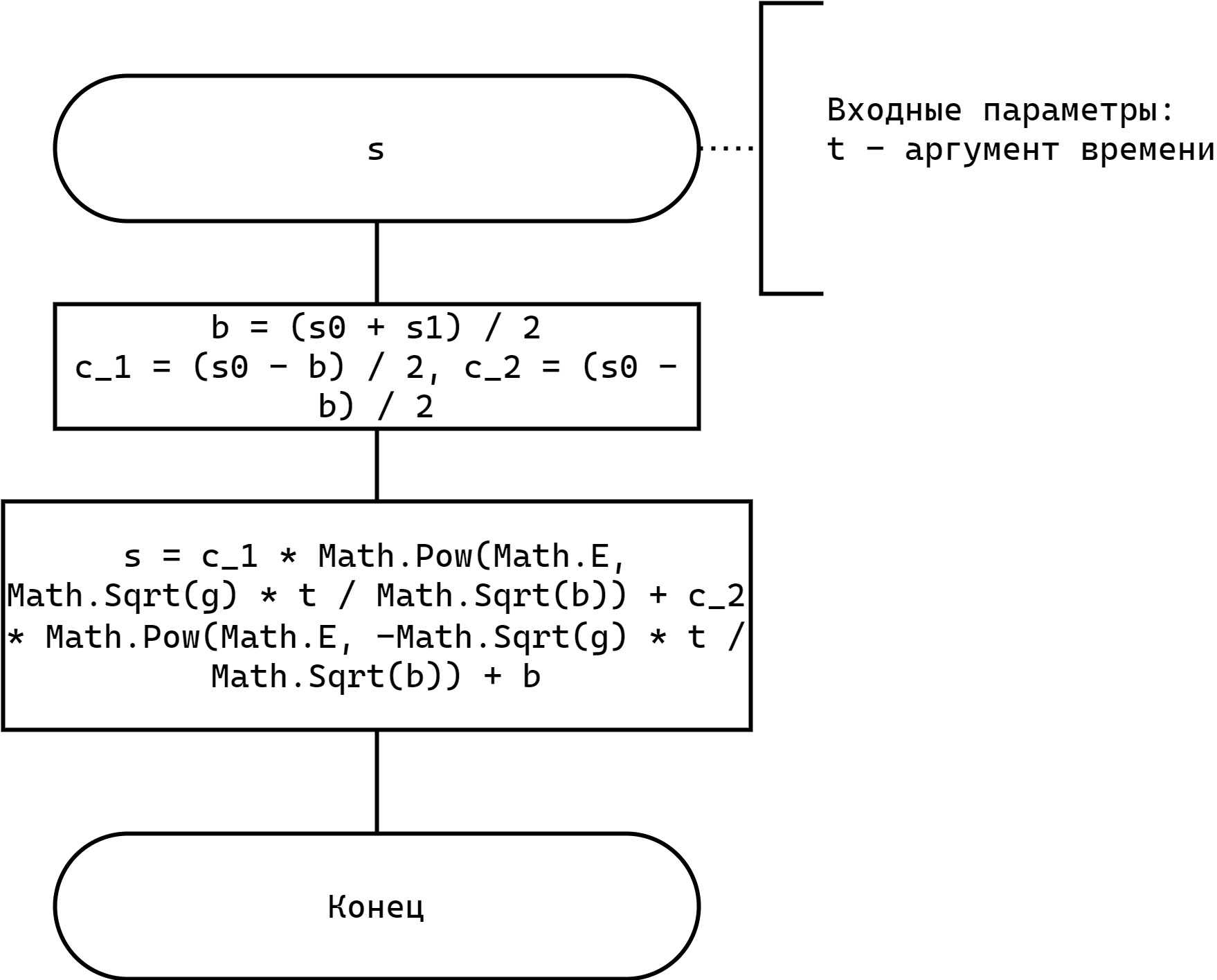


Рисунок 3 – схема алгоритма функции решения дифф. Уравнения

На рисунке 4 изображена схема алгоритма функции, вычисляющей производную функции, изображённой на рисунке 3.

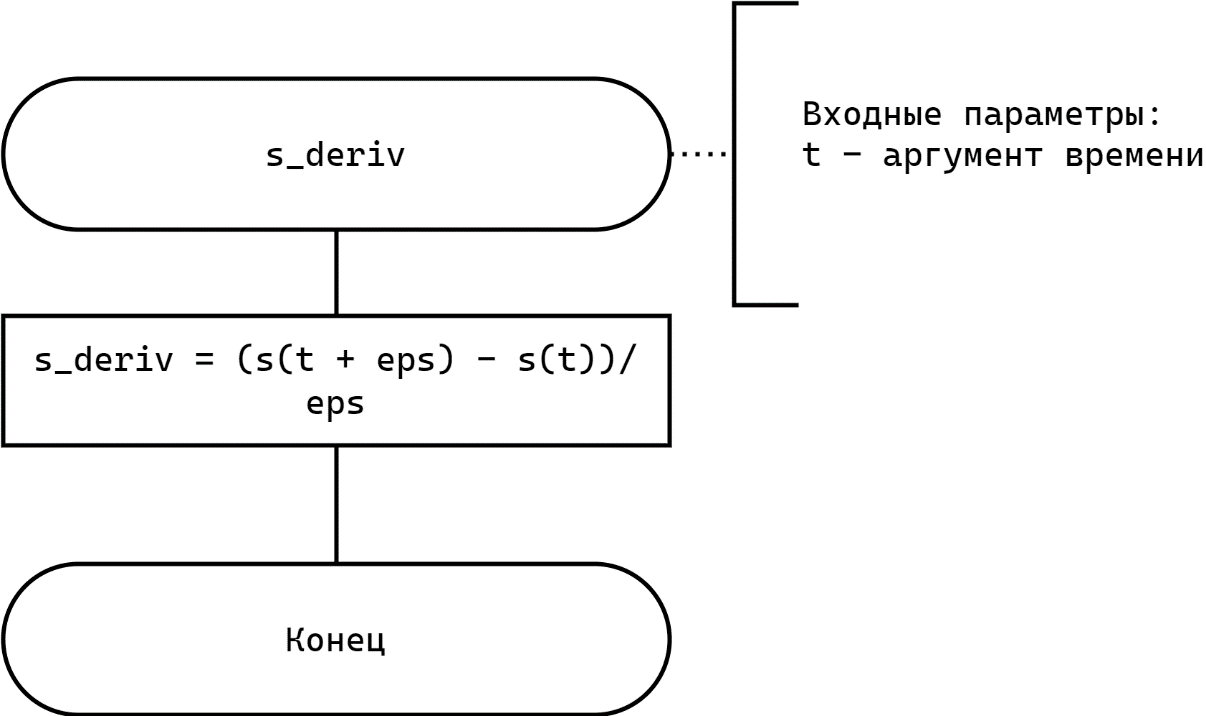


Рисунок 4 – схема алгоритма функции взятия производной

На рисунке 5 изображена схема алгоритма функции, выполняющейся при загрузки каждой из форм, данная функция устанавливает нужный фоновый цвет на соответствующей форме.

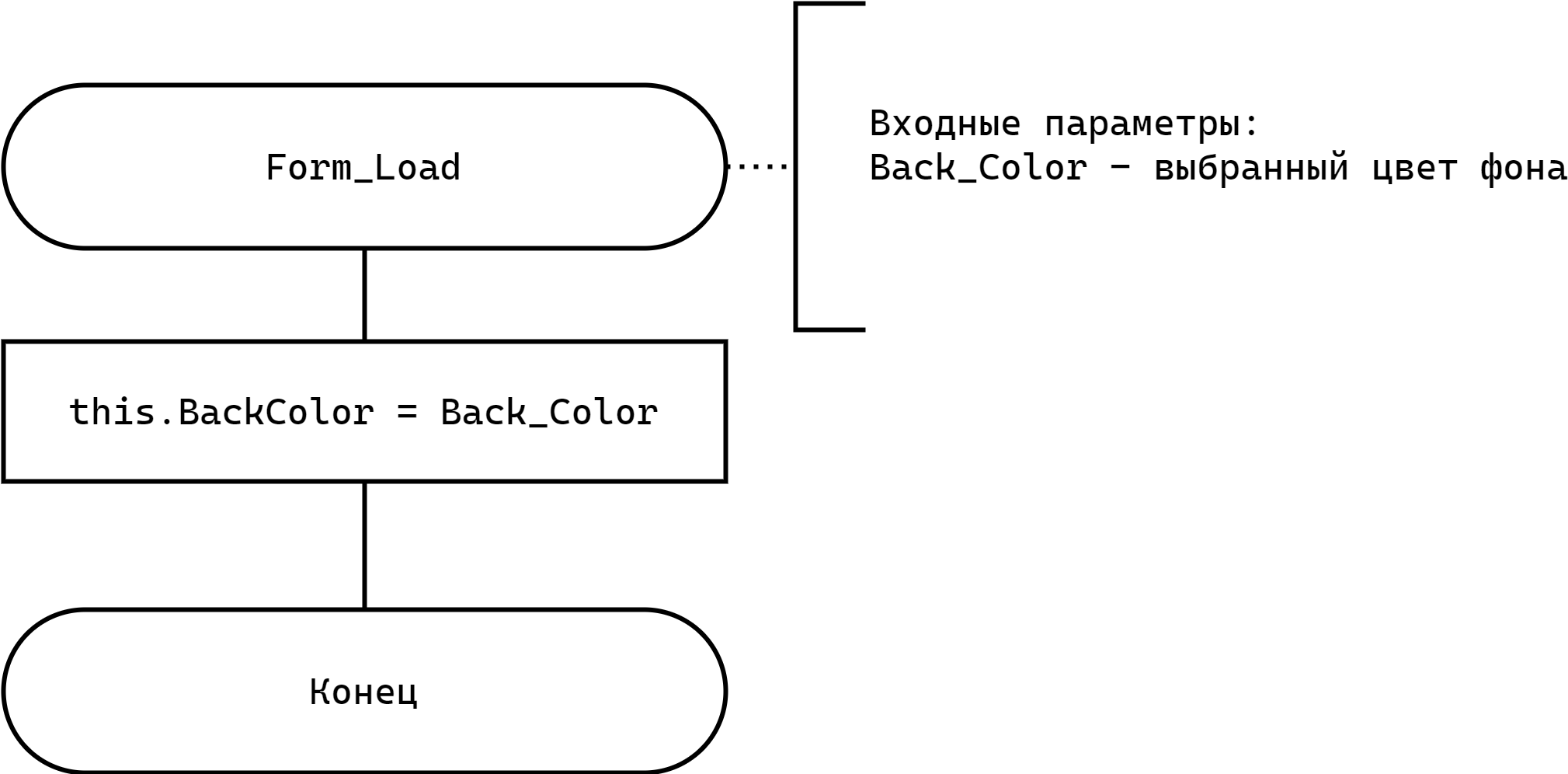


Рисунок 5 – схема алгоритма функции изменения фонового цвета

На рисунке 6 изображена схема алгоритма функции, включающий воспроизведение музыки в программе.

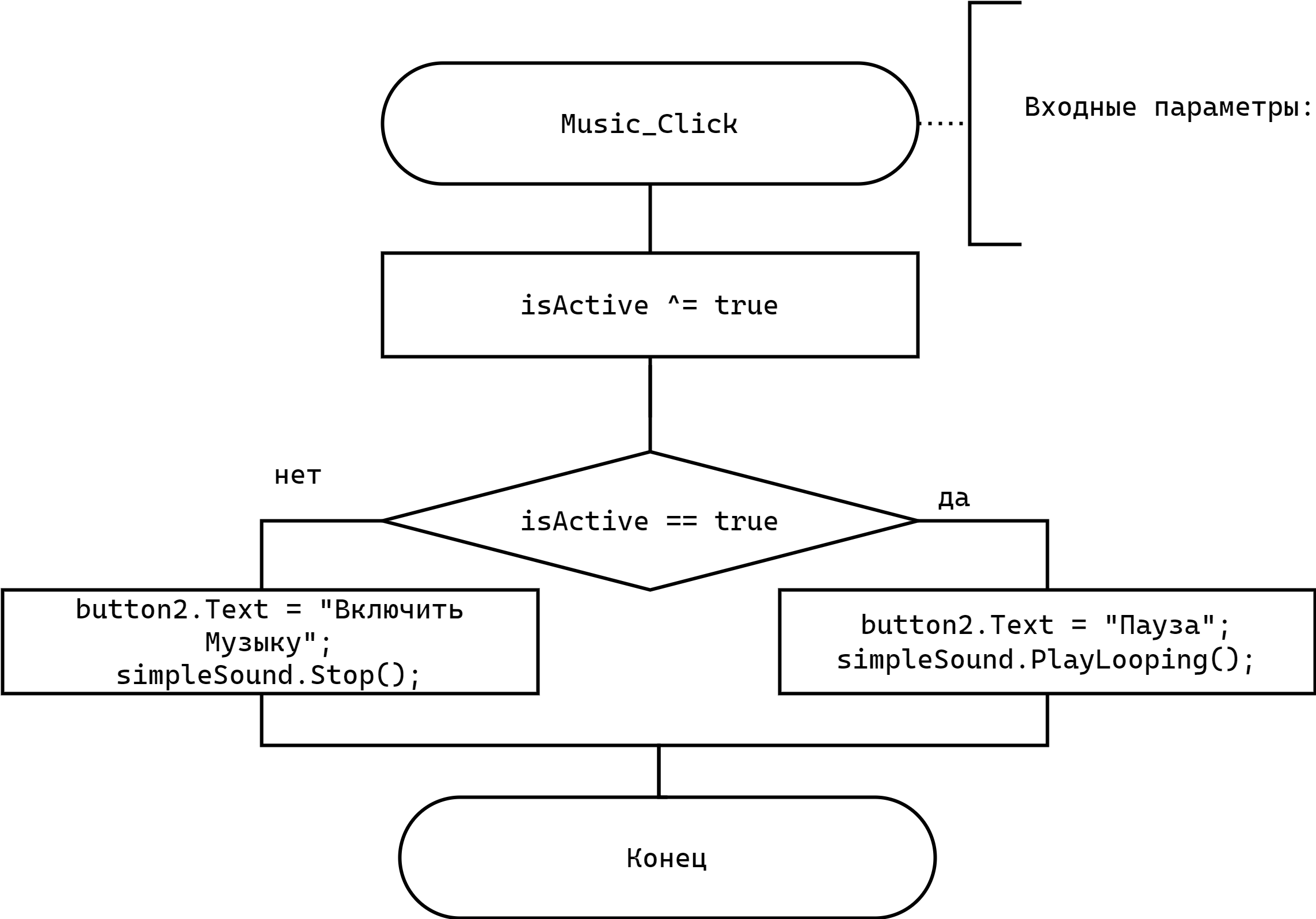


Рисунок 6 – схема алгоритма функции включения музыки

На рисунке 7 изображена схема алгоритма функции, меняющей текущую форму.

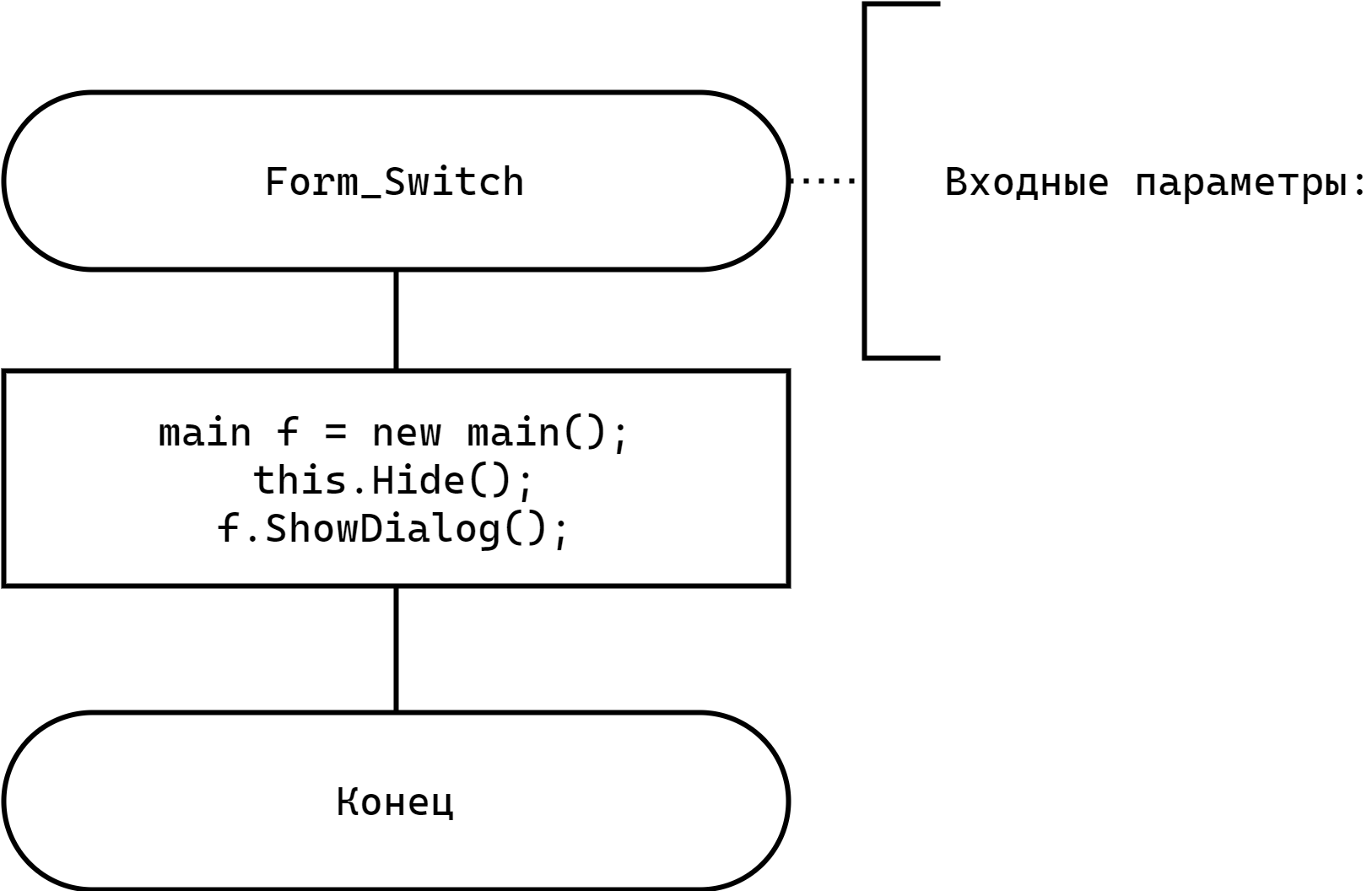


Рисунок 7 – схема алгоритма функции меняющей текущую форму

1. Программный код

Далее представлен программный код формы, выполняющей расчёты, **task.cs**.

**public** **void** **output\_mas**(double[] mas, int len, DataGridView grid, double h)

{

grid.RowCount = 2;

grid.ColumnCount = len;

**for** (int i = 0; i < len; i++)

{

grid.Rows[0].Cells[i].Value = (i \* h).ToString("F3");

grid.Rows[1].Cells[i].Value = mas[i].ToString("F3");

}

}

double g = 9.8;

double s1 = 8;

double s0 = 10;

double t\_last = 2.767;

double **s**(double t)

{

double b = (s0 + s1) / 2;

double c\_1 = (s0 - b) / 2, c\_2 = (s0 - b) / 2;

**return** c\_1 \* Math.Pow(Math.E, Math.Sqrt(g) \* t / Math.Sqrt(b)) + c\_2 \* Math.Pow(Math.E, -Math.Sqrt(g) \* t / Math.Sqrt(b)) + b;

}

double **s\_deriv**(double t)

{

double eps = 0.00001;

**return** (s(t + eps) - s(t))/ eps;

}

**public** **unsafe** double[] **diff\_solve\_arr**(double h, int it, double s0, double s1)

{

double t = 0;

double[] arr = **new** double[it];

**for**(int i =0; i < it; ++i)

{

arr[i] = s(t);

t += h;

}

**return** arr;

}

double **binary\_search**(double eps = 0.0001)

{

double l = 0; // f(l)=0

double r = 1e10; // f(r)=1

**while** (r - l > eps)

{

double m = (r + l) / 2;

**if** (s(m) > (s0 + s1)) // target = s0 + s1

r = m;

**else**

l = m;

}

**return** (r + l)/ 2;

}

String final\_content = "Не было произведено расчёта";

**private** **void** **Calculate\_Click**(object sender, EventArgs e)

{

double t0;

double h, it;

double ans1, ans2, ans3 = 0;

final\_content = "Задание №1\n\n";

**if** (Input(g\_txtBox, **out** g, errorProvider1, "В поля для ввода данных допускается лишь ввод числовых значений с плавающей точкой"))

final\_content += "g = " + g + '\n';

**else**

**return**;

**if** (Input(s0\_txtBox, **out** s0, errorProvider1, "В поля для ввода данных допускается лишь ввод числовых значений с плавающей точкой"))

final\_content += "s0 = " + s0 + '\n';

**else**

**return**;

**if** (Input(s1\_txtBox, **out** s1, errorProvider1, "В поля для ввода данных допускается лишь ввод числовых значений с плавающей точкой"))

final\_content += "s1 = " + s1 + '\n' + '\n';

**else**

**return**;

**if** (Input(h\_txtBox, **out** h, errorProvider1, "В поля для ввода данных допускается лишь ввод числовых значений с плавающей точкой"))

final\_content += "h = " + h + '\n';

**else**

**return**;

**if** (Input(it\_txtBox, **out** it, errorProvider1, "В поля для ввода данных допускается лишь ввод числовых значений с плавающей точкой"))

final\_content += "it = " + it + '\n' + '\n';

**else**

**return**;

animate();

output\_mas(diff\_solve\_arr(h, Int32.Parse(it.ToString()), s0, s1), Int32.Parse(it.ToString()), dataGridView2, h);

t\_last = binary\_search();

t\_fall\_txtBox.Text = t\_last.ToString("F3");

final\_content += "t\_падения = " + t\_fall\_txtBox.Text + '\n';

v\_fall\_txtBox.Text = s\_deriv(binary\_search(0.00001)).ToString("F3");

final\_content += "v\_падения = " + v\_fall\_txtBox.Text + '\n';

}

**private** **void** **task1\_Load**(object sender, EventArgs e)

{

**this**.BackColor = prac\_MTUCI.main.Back\_Color;

animate();

}

**private** **void** **to\_statement\_Click**(object sender, EventArgs e)

{

task1\_statement f = **new** task1\_statement();

**this**.Hide();

f.ShowDialog();

}

**private** **void** **textBox2\_KeyPress**(object sender, KeyPressEventArgs e)

{

//NumberLimiter(e, s0\_txtBox);

}

**private** **void** **textBox1\_KeyPress**(object sender, KeyPressEventArgs e)

{

//NumberLimiter(e, g\_txtBox);

}

**private** **void** **label1\_Click**(object sender, EventArgs e)

{

}

**private** **void** **button2\_Click**(object sender, EventArgs e)

{

**if** (animation\_tb.Value == 100)

{

animation\_tb.Value = 0;

timer1.Enabled = true;

**return**;

}

timer1.Enabled ^= true;

**if** (timer1.Enabled)

{

button2.Text = "Пауза";

}

**else**

{

button2.Text = "Анимировать";

}

}

**void** **animate**()

{

double cur\_time = (animation\_tb.Value / 100.0) \* t\_last;

double new\_s0 = 10 \* s(cur\_time);

double new\_s1 = 10 \* ((s1+s0)- s(cur\_time));

picture\_s1.Size = **new** Size(picture\_s1.Size.Width, Convert.ToInt32(Math.Ceiling(new\_s1)));

picture\_s0.Size = **new** Size(picture\_s0.Size.Width, Convert.ToInt32(Math.Ceiling(new\_s0)));

}

**private** **void** **timer1\_Tick**(object sender, EventArgs e)

{

**if** (animation\_tb.Value >= 99)

{

animation\_tb.Value = 100;

button2.Text = "Анимировать";

timer1.Enabled = false;

animate();

**return**;

}

animation\_tb.Value += 1;

animate();

}

**private** **void** **animation\_tb\_Scroll**(object sender, EventArgs e)

{

animate();

}

**private** **void** **button1\_Click**(object sender, EventArgs e)

{

System.IO.File.WriteAllText("task1.txt", final\_content);

}

**private** **void** **button3\_Click**(object sender, EventArgs e)

{

**var** reader = **new** StreamReader("input.txt");

string line;

int cou = 0;

**while** ((line = reader.ReadLine()) != null)

{

**if** (cou == 0) g\_txtBox.Text = line;

**if** (cou == 1) s0\_txtBox.Text = line;

**if** (cou == 2) s1\_txtBox.Text = line;

cou++;

}

}

Далее представлен программный код формы Настроек, **settings.cs**.

**private** **void** **button1\_Click**(object sender, EventArgs e)

{

**if** (comboBox1.Text == "Белый")

{

prac\_MTUCI.main.Back\_Color = SystemColors.Control;

}

**if** (comboBox1.Text == "Голубой")

{

prac\_MTUCI.main.Back\_Color = SystemColors.ActiveCaption;

}

**if** (comboBox1.Text == "Серый")

{

prac\_MTUCI.main.Back\_Color = SystemColors.InactiveCaption;

}

**this**.BackColor = prac\_MTUCI.main.Back\_Color;

}

**private** **void** **task2\_statement\_Load**(object sender, EventArgs e)

{

**this**.BackColor = prac\_MTUCI.main.Back\_Color;

**if** (isActive)

{

button2.Text = "Пауза";

}

**else**

{

button2.Text = "Включить Музыку";

}

}

**static** bool isActive = false;

**private** **void** **button2\_Click**(object sender, EventArgs e)

{

isActive ^= true;

**if** (isActive)

{

button2.Text = "Пауза";

simpleSound.PlayLooping();

}

**else**

{

button2.Text = "Включить Музыку";

simpleSound.Stop();

}

}

1. Результаты работы программы

На рисунке 8 представлен титульный лист программы

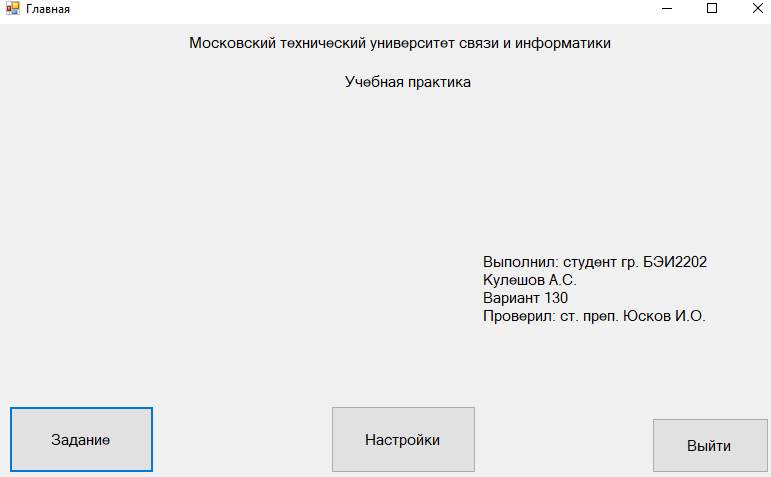


Рисунок 8 – Титульный лист

На рисунке 9 представлена форма с условием задания.

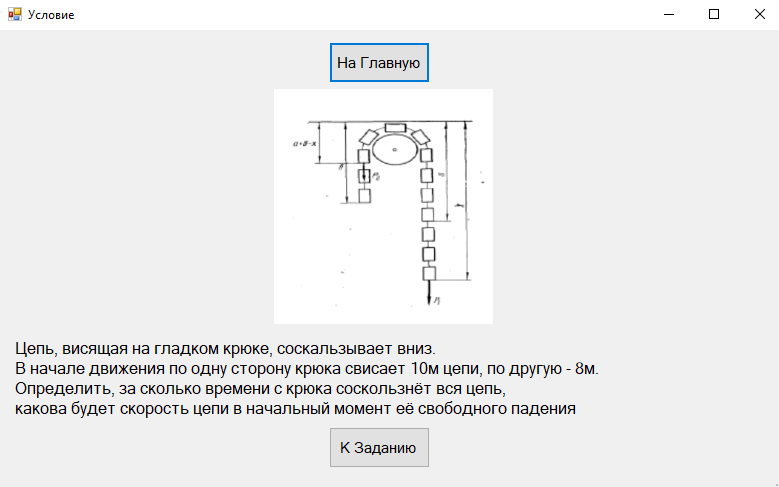


Рисунок 9 – Форма с условием задания

На рисунке 10 представлена форма с заданием.

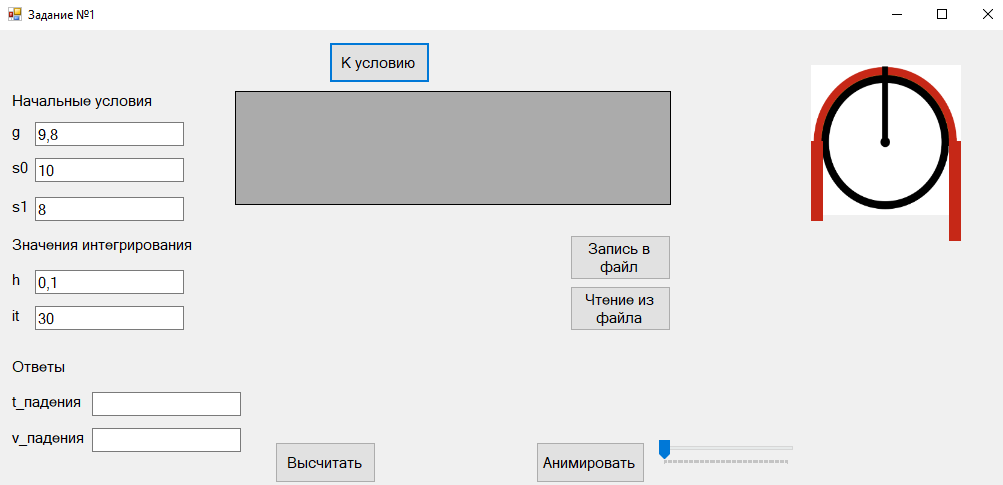


Рисунок 10 – Форма с заданием

При нажатии на кнопку «Высчитать», результаты вычислений будут отображены в соответствующих полях, это отображено на рисунке 11

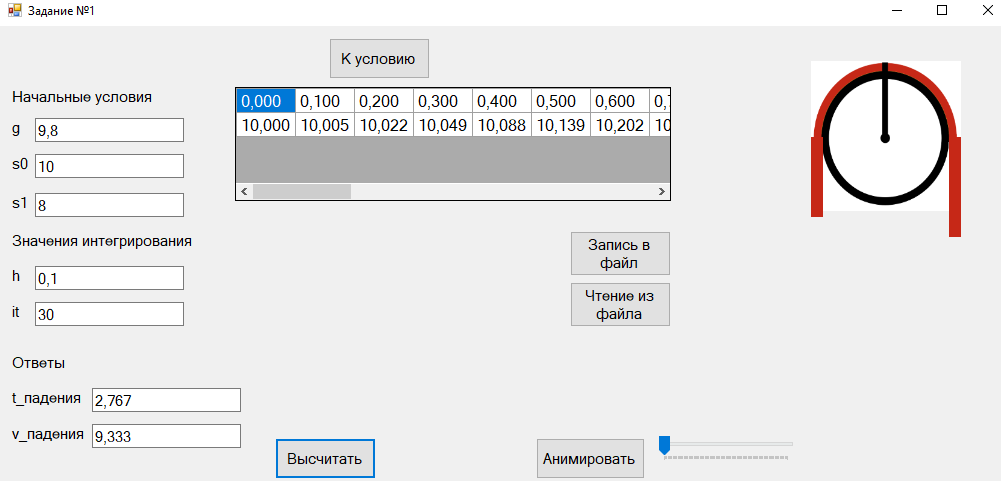


Рисунок 11 – Результат вычислений

После нажатия на кнопку «Запись в файл» результаты вычислений будут записаны в файл, отображено на рисунке 12.

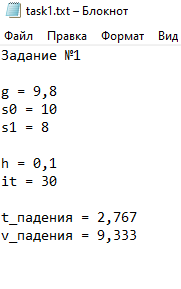


Рисунок 12 – Результат работы программы в текстовом файле

После нажатия на кнопку «Чтение из файла» данные в соответствующих входных полях программы будут изменены на данные, что находились во входном файле, показано на рисунках 13-14.

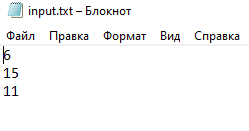


Рисунок 13 – текстовый файл с исходными данными

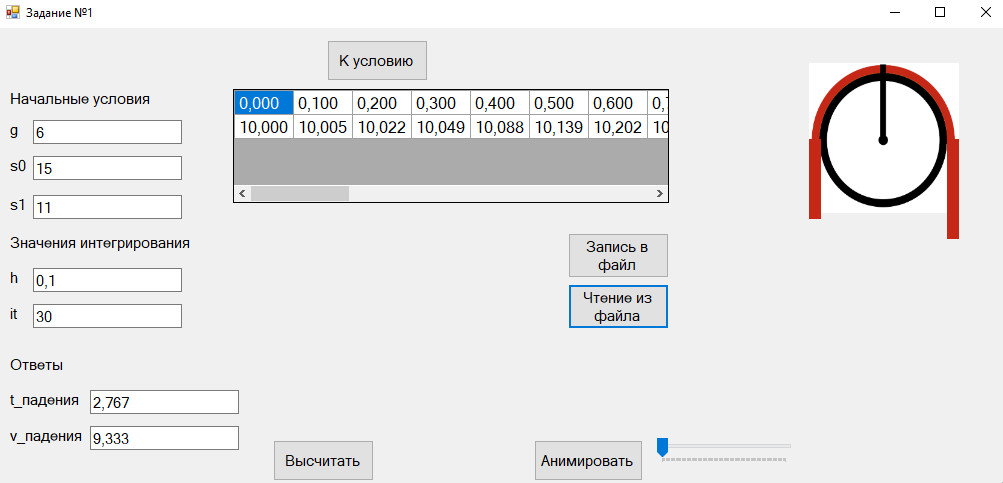


Рисунок 14 – данные были успешно перенесены в программу

После нажатия на кнопку «Анимировать» запуститься анимация, которую потом можно поставить на паузу, нажав на ту же кнопку, но уже с надписью «Пауза», также можно вручную двигать ползунок анимации. Отображено на рисунках 15-16.

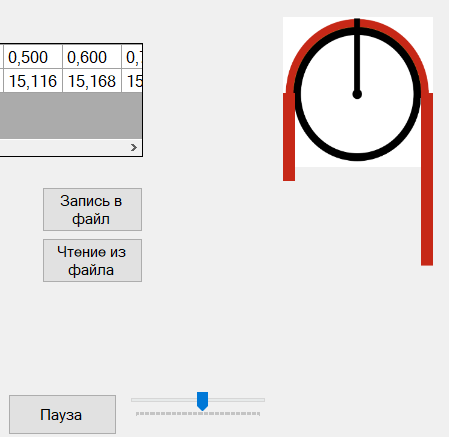


Рисунок 15 – Анимация в процессе

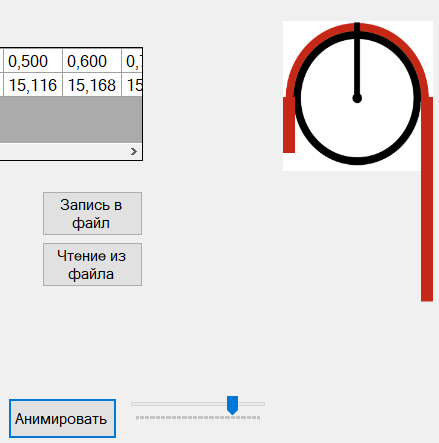


Рисунок 16 – анимация на паузе

На титульном листе, нажав на кнопку «Настройки» открывается форма с настройками, на которой можно поменять фон проекта и включить музыку. Отображено на рисунках 17-20.

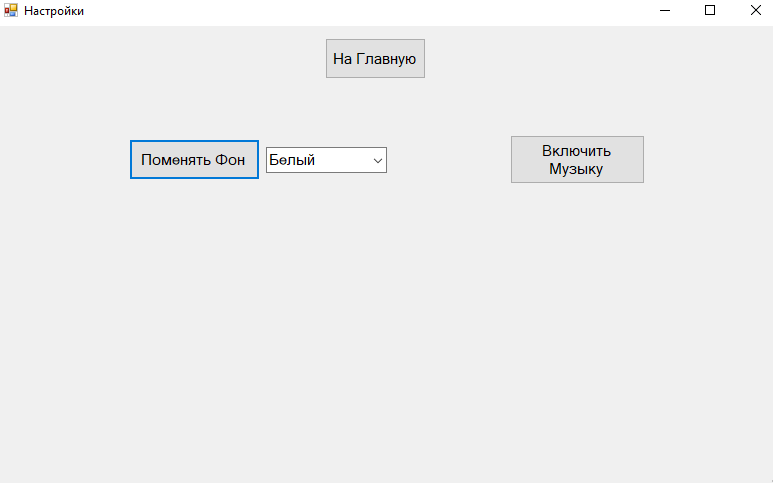


Рисунок 17 – Форма с настройками

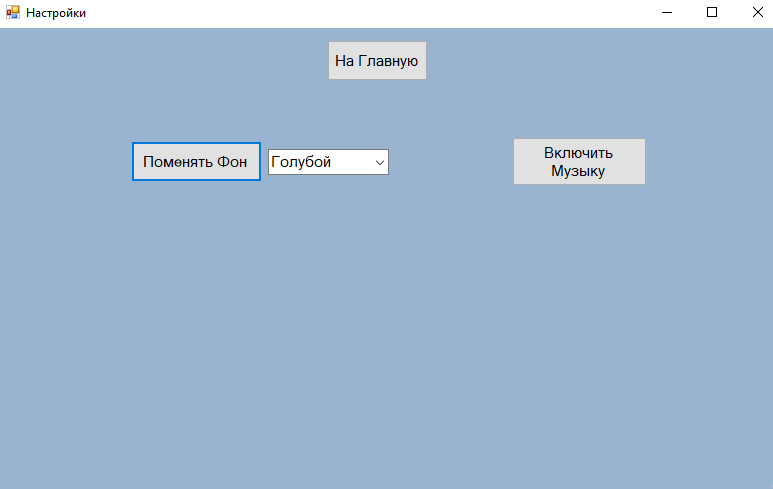


Рисунок 18 – изменённый цвет на форме

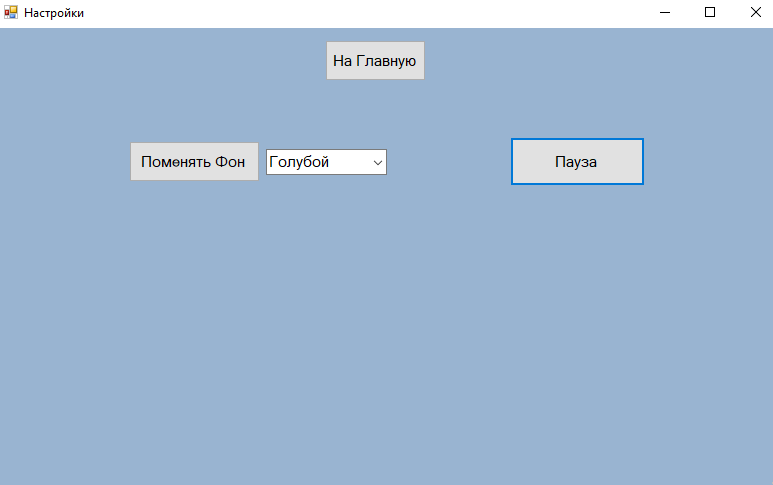


Рисунок 19 – изменение подписи на кнопке, после включения музыки

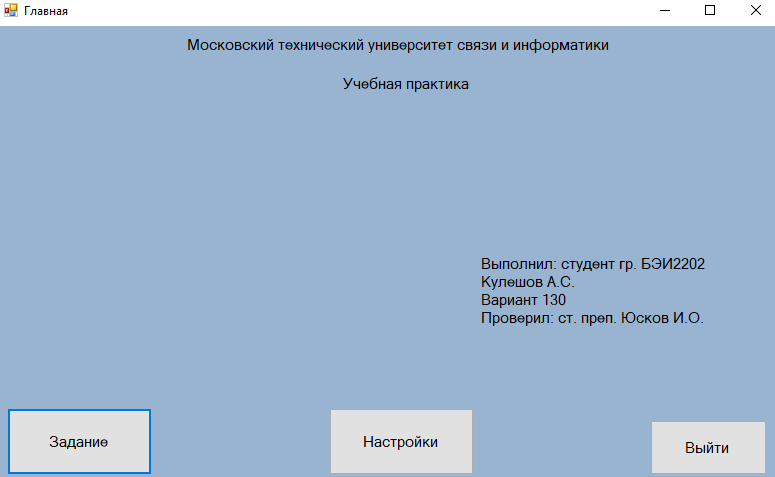


Рисунок 20 – изменение фона во всех формах приложения

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Гуриков С. Р. Основы алгоритмизации и программирования на Python : учебное пособие / С. Р. Гуриков. — Москва : ИНФРА-М, 2022. — 343 с. — (Высшее образование: Бакалавриат). - ISBN 978-5-16-017142-5. - Текст : электронный. - URL: https://znanium.com/catalog/product/1356003 (дата обращения: 26.02.2024). – Режим доступа: по подписке.

2 ГОСТ 7.32-2017. Межгосударственный стандарт. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления (введен в действие Приказом Росстандарта от 24.10.2017 N 1494-ст) — М., Стандартинформ, 2017.