Министерство науки и высшего образования Российской Федерации

Российский химико-технологический университет имени Д. И. Менделеева

Факультет цифровых технологий и химического инжиниринга

Кафедра информационных компьютерных технологий

**ОТЧЕТ**

**ПО ТЕХНОЛОГИЯМ ПРОГРАММИРОВАНИЯ**

КУРСОВАЯ РАБОТА

**«Реализация алгоритма игры “Ипподром”»**

**ВЫПОЛНИЛ:** студент группы КС-20 Колесников Артем Максимович

**ПРОВЕРИЛ:** ст. преп. Васецкий А. М.

**Москва**

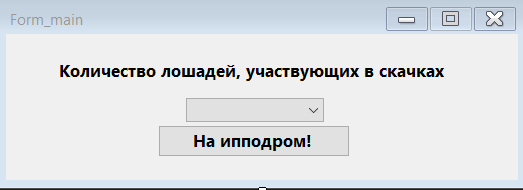
**2021**

**Задание №36**

Реализовать алгоритм игры «Ипподром». Играющий выбирает одну из заданного количества лошадей (2-10), состязающихся на бегах, и выигрывает, если его лошадь приходит первой. Скорость передвижения лошадей на разных этапах (не менее 10) выбирается программой с помощью генератора псевдослучайных чисел. Вывести точечную диаграмму скоростей лошадей с форматированием линий данных. Протоколировать ход игры. Стирать старый файл или дополнять его по выбору пользователя.

Запуская программу, мы попадаем в меню с выбором числа лошадей. После нажатия кнопки **«На ипподром!»** открывается форма арены. Сразу же – генерируется сам ипподром, в зависимости от выбранного числа лошадей. На этой форме мы можем выбрать лошадь, способ записи истории игр, а также – сохранить или загрузить пользовательские настройки. При нажатии кнопки **«Старт»** – запускается сам алгоритм игры. Во время забега мы можем приостановить игру (нажав кнопку **«Пауза»**), чтобы посмотреть диаграмму скоростей лошадей или же сделать это после того, как все лошади пересекут финишную черту. Нажав кнопку **«График»,** откроется ещё 1 форма, на этот раз с двумя вкладками – «График» и «Форматирование». На первой – находится сама диаграмма скоростей, на второй же, в зависимости от числа лошадей, сгенерированы элементы управления, позволяющие форматировать линии графика (тип, толщина, цвет), также там находится кнопка **«Сбросить»**, при нажатии на которую – все настройки сбрасываются «по умолчанию». После завершения гонки на основной форме мы увидим общие итоги забега и непосредственно результат нашей ставки (все результаты также можно увидеть в файле «Record.txt»). Теперь мы можем сохранить все выбранные настройки или нажать кнопку **«Вернуться в меню»**, где нам вновь предстоит выбрать число лошадей и быть может снова очутиться в такой захватывающей игре, как «Ипподром».

**Код программы**



public partial class Form\_menu : Form

{

public int max\_number\_of\_horses = 10;

public Form\_menu()

{

InitializeComponent();

for (int i = 2; i <= max\_number\_of\_horses; i++)

comboBox\_number\_horses.Items.Add(i);

comboBox\_number\_horses.SelectedIndex = 0;

\_Form\_menu = this;

}

public static Form\_menu \_Form\_menu;

//Нажатие кнопки "На ипподром"

private void button\_start\_Click(object sender, EventArgs e)

{

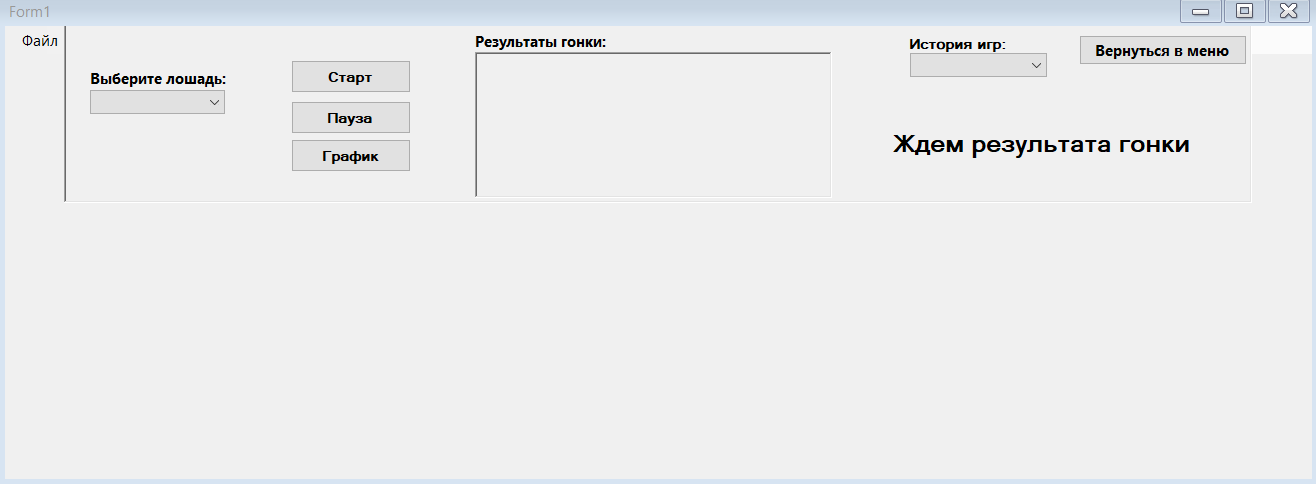
Form1 f1 = new Form1(\_Form\_menu);

f1.Show();

this.Hide();

}

}



public partial class Form1 : Form

{

public int number\_of\_horses; //количество лошадей, учавствующих в гонке

int max\_number\_of\_horses; //макимально возможное количество лошадей

public int ind\_selected\_horse; //выбранная лошадь

int ind\_previous\_horse; //индекс предыдущей выбр. лошади (для возвращения дефолтной картинки дорожки)

int number\_of\_finishers; //счетчик лошадей пришедших к финишу

int number\_of\_stages; //количество этапов

public List<int> list\_stages = new List<int>(); //список этапов (для графика)

int finish\_line = 640; //число пикселей до финиша

List<PictureBox> list\_pictureBoxes = new List<PictureBox>(); //коллекция picturebox-ов лошадей

List<Panel> list\_road = new List<Panel>(); //коллекция дорожек

List<double> list\_average\_speed = new List<double>(); //список средней скорости каждой лошади

public List<List<int>> list\_of\_speeds\_horses = new List<List<int>>(); //двумерный список скоростей каждой лошади на каждом этапе

int place\_select\_horse; //место выбранной лошади

int index\_win\_horse; //индекс лошади-победителя

List<FileMode> list\_filemode = new List<FileMode>() { FileMode.Create, FileMode.Append };//список способов открытия файла

public int ind\_select\_filemode;//индекс выбранной лошади

Random rnd = new Random();

public Form1(Form\_menu form\_menu)

{

InitializeComponent();

this.form\_menu = form\_menu;

number\_of\_horses = Convert.ToInt32(form\_menu.comboBox\_number\_horses.Text);

max\_number\_of\_horses = form\_menu.max\_number\_of\_horses;

}

Form\_menu form\_menu; //форма меню

public static Form1 \_Form1;

public Form\_chart form\_chart; //форма с графиком

//нажатие на кнопку "Старт"

private void button\_start\_Click(object sender, EventArgs e)

{

timer1.Enabled = true;

Start\_position();

Close\_chartForm();

File\_ToolStripMenuItem.Enabled = false;

button\_pause.Visible = true;

button\_start.Enabled = false;

comboBox\_select\_horse.Enabled = false;

comboBox\_record.Enabled = false;

}

//метод, срабатывающий при каждом тике таймера

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

{

int stage\_speed; //скорость на этапе

double average\_speed; //средняя скорость

number\_of\_stages += 1; //увеличиваем количество этапов

list\_stages.Add(number\_of\_stages); //добавляем номер этапа в список (для графика)

for (int i = 0; i < number\_of\_horses; i++)

{

//передвижение лошадей и запись скорости каждой лошади на каждом этапе в списки

stage\_speed = rnd.Next(5, 20);

list\_pictureBoxes[i].Left += stage\_speed;

list\_of\_speeds\_horses[i].Add(stage\_speed);

//блок, срабатывающий когда лошадь финиширует

if ((list\_pictureBoxes[i].Left > finish\_line) && ((int)list\_pictureBoxes[i].Tag == 0))

{

number\_of\_finishers += 1; //увелииваем число финиширующих лошадей

list\_pictureBoxes[i].Tag = number\_of\_finishers;

average\_speed = (double)list\_pictureBoxes[i].Left / (double)number\_of\_stages; //определяем среднюю скорость лошади

list\_average\_speed[i] = average\_speed; //добавляем среднюю скорость в список

}

}

//блок, срабаатывающий после того, как все лошади пересекут финишную черту

if (number\_of\_finishers == number\_of\_horses)

{

timer1.Enabled = false;

richTextBox\_result.Visible = true;

button\_pause.Visible = false;

File\_ToolStripMenuItem.Enabled = true;

//блок, который определяет место лошади в гонке

for (int i = 0; i < number\_of\_horses; i++)

{

//индекс максимального элемента в списке средних скоростей

int index\_max\_speed = list\_average\_speed.IndexOf(list\_average\_speed.Max());

//переменные для вывода лошади и места

int horse = index\_max\_speed + 1;

int place = i + 1;

//вывод результата гонки

richTextBox\_result.Text += horse + "-я лошадь - " + place + " место - V(ср) = " + Math.Round(list\_average\_speed.Max(), 4) + "\n";

if (index\_max\_speed == ind\_selected\_horse)

place\_select\_horse = place;

//сохраняем индекс лошади с максимальной скоростью

if (i == 0)

index\_win\_horse = index\_max\_speed;

list\_average\_speed[index\_max\_speed] = 0; //обнуление максимального элемента в списке средних скоростей

}

//блок, определяющий нашу победу или поражение в ставке

if (ind\_selected\_horse == index\_win\_horse)

Win();

else

Loose();

//вызываем метод, записывающий историю матча

Record\_game();

}

}

private void Form1\_Load(object sender, EventArgs e)

{

\_Form1 = this;

//вызываем метод отрисовки ипподрома

Load\_arena(number\_of\_horses);

comboBox\_select\_horse.SelectedIndex = 0;

comboBox\_record.SelectedIndex = 0;

ind\_select\_filemode = comboBox\_record.SelectedIndex;

form\_chart = new Form\_chart(\_Form1);

}

//метод, который приводит ипподром в исходное состояние

public void Start\_position()

{

number\_of\_finishers = 0; //обнуление числа финиширующих лошадей

number\_of\_stages = 0; //обнуление количества этапов

//обнуление и возвращение начальных значений списка мест и средней скорости

for (int i = 0; i < number\_of\_horses; i++)

{

//возвращаем лошадей на исходные позиции

list\_pictureBoxes[i].Location = new Point(0, 0);

list\_pictureBoxes[i].Tag = 0;

list\_average\_speed[i] = 0;

list\_of\_speeds\_horses[i].Clear();

}

list\_stages.Clear();

//обнуление табла результатов

richTextBox\_result.Text = "";

label\_result.Text = "Ждем результата гонки";

label\_result.ForeColor = Color.Black;

}

//метод генерирующий арену ипподрома в зависимости от выбранного количества лошадей

public void Load\_arena(int number\_of\_horses)

{

//размер дорожки

int width\_road = 974;

int height\_road = 65;

//размер картинки лошади

int width\_horse = 115;

int heigh\_horse = 60;

//настройка формы

int heigh\_form = panel1.Height + height\_road \* max\_number\_of\_horses + 30;

ClientSize = new Size(1000, heigh\_form);

//координаты дорожек относительно формы

int x\_road = (this.Width - width\_road) / 2;

int y\_road = panel1.Height + (this.Height - panel1.Height) / number\_of\_horses - height\_road;

for (int i = 0; i < number\_of\_horses; i++)

{

//заполняем комбобокс числом лошадей

comboBox\_select\_horse.Items.Add(i + 1);

//создаем элементы управления

PictureBox pic\_horse = new PictureBox();

Panel panel\_road = new Panel();

//устанавливаем размеры

pic\_horse.Size = new Size(width\_horse, heigh\_horse);

panel\_road.Size = new Size(width\_road, height\_road);

//загружаем картинки лошадей и дорожек

pic\_horse.Load("..\\picture\\horse" + i + ".png");

panel\_road.BackgroundImage = new Bitmap("..\\picture\\road\_defolt.png");

panel\_road.Anchor = AnchorStyles.Top;

//указываем способ отображения

panel\_road.BackgroundImageLayout = ImageLayout.Center;

pic\_horse.SizeMode = PictureBoxSizeMode.Zoom;

Controls.Add(panel\_road);

panel\_road.Controls.Add(pic\_horse); //устанавливаем родительский элемент (дорожка) для картинок лошадей

pic\_horse.BackColor = Color.Transparent;

//картинки лошадей на передний план

pic\_horse.BringToFront();

//устанавливаем расположение дорожек

panel\_road.Location = new Point(x\_road, y\_road);

y\_road = panel1.Height + (this.Height - panel1.Height) / number\_of\_horses + (i \* height\_road);

//добавляем картинку лошади в коллекцию

list\_pictureBoxes.Add(pic\_horse);

list\_road.Add(panel\_road);

list\_average\_speed.Add(0);

list\_of\_speeds\_horses.Add(new List<int>());

}

}

//метод, срабатывающий в случае, если наша ставка была верной

public void Win()

{

label\_result.Text = "Ваша лошадь победила!";

label\_result.ForeColor = Color.Green;

button\_start.Enabled = true;

comboBox\_select\_horse.Enabled = true;

comboBox\_record.Enabled = true;

}

//метод, срабатывающий в случае, если наша ставка прогорела

public void Loose()

{

label\_result.ForeColor = Color.Red;

label\_result.Text = "Ваша лошадь заняла:\n" + place\_select\_horse + " место";

button\_start.Enabled = true;

comboBox\_select\_horse.Enabled = true;

comboBox\_record.Enabled = true;

}

//возврат в главное меню по кнопке "Вернуться в меню" или при закрытии окна ипподрома

private void button\_main\_menu\_Click(object sender, EventArgs e)

{

//закрытие графика

Close\_chartForm();

Form menu = Application.OpenForms[0];

this.Close();

menu.Show();

}

private void Form1\_FormClosed(object sender, FormClosedEventArgs e)

{

//закрытие графика

Close\_chartForm();

Form menu = Application.OpenForms[0];

menu.Show();

}

//"Подсвечивание" выбранной дорожки, при выборе лошади

private void comboBox\_select\_horse\_SelectedIndexChanged(object sender, EventArgs e)

{

try

{

ind\_selected\_horse = comboBox\_select\_horse.SelectedIndex; //выбранная лошадь

//возвращаем картинку по умолчанию для предыдущей выбранной лошади

list\_road[ind\_previous\_horse].BackgroundImage = new Bitmap("..\\picture\\road\_defolt.png");

//загружаем "подсвеченную" дорожку для выбранной лошади

list\_road[ind\_selected\_horse].BackgroundImage = new Bitmap("..\\picture\\road\_select.png");

//запоминаем индекс измененной дорожки

ind\_previous\_horse = ind\_selected\_horse;

}

catch

{

list\_road[ind\_selected\_horse].BackgroundImage = new Bitmap("..\\picture\\road\_select.png");

ind\_previous\_horse = ind\_selected\_horse;

}

}

//нажатие на кнопку "График"

private void button\_chart\_Click(object sender, EventArgs e)

{

Form fc = Application.OpenForms["Form\_chart"];

//проверяем - открыта ли форма с графиком

if (fc == null)

{

form\_chart = new Form\_chart(\_Form1);

form\_chart.Show();

}

else

{

fc.Refresh();

fc.Show();

fc.BringToFront();

}

}

//нажатие на кнопку "Пауза"

private void button\_pause\_Click(object sender, EventArgs e)

{

bool check = timer1.Enabled;

//проверяем - включен ли таймер

if (check)

button\_pause.Text = "Продолжить";

else

{

Close\_chartForm();

button\_pause.Text = "Пауза";

}

timer1.Enabled = !check;

}

//метод, записывающий историю игр

public void Record\_game()

{

string text = $"Игра: {DateTime.Now}" +

$"\nКоличество лошадей: {number\_of\_horses}" +

$"\nВыбранная лошадь: {ind\_selected\_horse + 1}" +

$"\nРезультат: {place\_select\_horse} место" +

"\nИтоги гонки:\n" +

$"{richTextBox\_result.Text}\n\n";

//в зависимости от выбора пользователя выполняется "Запись" или "Перезапись" файла

try

{

using (FileStream filestream = new FileStream("Record.txt", list\_filemode[ind\_select\_filemode]))

{

byte[] array = System.Text.Encoding.Default.GetBytes(text);

filestream.Write(array, 0, array.Length);

}

}

catch (Exception ex)

{

MessageBox.Show($"Error.\n\nError message: {ex.Message}\n\n" +

$"Details:\n\n{ex.StackTrace}");

}

}

public List<int> list\_location\_settings = new List<int>(); //список расположения настроек в файле (для произвольного доступа)

string text\_save\_settings; //сохраненные настройки

//нажатие кнопки "Сохранить пользовательские настройки"

private void saveToolStripMenuItem\_Click(object sender, EventArgs e)

{

try

{

list\_location\_settings.Clear();

text\_save\_settings = "";

//вызываем метод, преобразующий входные данные к строковому формату и запоминающий их расположение в файле

Record\_text\_settings(number\_of\_horses, ind\_selected\_horse, ind\_select\_filemode);

int type\_line;

int width;

string color\_line;

Form fc = Application.OpenForms["Form\_chart"];

if (fc == null)

form\_chart = new Form\_chart(\_Form1);

//получаем данные из элементов форматирования графика

for (int i = 0; i < number\_of\_horses; i++)

{

type\_line = form\_chart.list\_combobox[i].SelectedIndex;

width = (int)form\_chart.list\_numericUpDOwn[i].Value;

color\_line = ColorTranslator.ToHtml(form\_chart.chart1.Series[$"Series{i}"].Color);

Record\_text\_settings(type\_line, width, color\_line);

}

//Записываем полученные настройки в файл

using (FileStream filestream = new FileStream("Settings.txt", FileMode.Create))

{

byte[] array = System.Text.Encoding.Default.GetBytes(text\_save\_settings);

filestream.Write(array, 0, array.Length);

}

string text = "";

//преобразуем информацию о расположении настроек в файле к строковому виду

foreach (int el in list\_location\_settings)

{

text += $"{el};";

}

//записываем расположения в отдельный файл

using (FileStream filestream = new FileStream("Location\_for\_settings.txt", FileMode.Create))

{

byte[] array = System.Text.Encoding.Default.GetBytes(text);

filestream.Write(array, 0, array.Length);

}

}

catch (Exception ex)

{

MessageBox.Show($"Error.\n\nError message: {ex.Message}\n\n" +

$"Details:\n\n{ex.StackTrace}");

}

}

//метод, преобразующий входные данные к строковому формату и запоминающий их расположение в файле

private void Record\_text\_settings(params object[] value)

{

foreach (var item in value)

{

text\_save\_settings += $"{item}";

list\_location\_settings.Add(text\_save\_settings.Length);

}

}

//метод, закрывающий форму с графиком

private void Close\_chartForm()

{

Form fc = Application.OpenForms["Form\_chart"];

if (fc != null)

fc.Close();

}

public List<List<string>> list\_formatting; //набор настроек элементов форматирования для каждой лошади

public bool download\_check = false; //индикатор нажатия кнопки "Загрузить пользовательские настройки"

//нажатие кнопки "Загрузить пользовательские настройки"

private void downloadToolStripMenuItem\_Click(object sender, EventArgs e)

{

try

{

string text;

//открываем "Маршрутную карту" с данными о расположении настроек в файле

using (FileStream fstream = File.OpenRead("Location\_for\_settings.txt"))

{

byte[] array = new byte[fstream.Length];

fstream.Read(array, 0, array.Length);

text = System.Text.Encoding.Default.GetString(array);

}

string[] splited = text.Split(new char[] { ';' }, StringSplitOptions.RemoveEmptyEntries);

list\_location\_settings.Clear();

//записываем данные о расположении в список

for (int i = 0; i < splited.Length; i++)

{

list\_location\_settings.Add(int.Parse(splited[i]));

}

list\_formatting = new List<List<string>>();

//открываем для чтения файл с пользовательскими настройками

using (FileStream fstream = File.OpenRead("Settings.txt"))

{

number\_of\_horses = int.Parse(Get\_setting(fstream, list\_location\_settings[0], 0));

int j = 0; //счетчик элементов форматирования

List<string> list\_for\_one\_horse = new List<string>(); //набор настроек для одной лошади

for (int i = 1; i < list\_location\_settings.Count; i++)

{

//длина настройки

int length = list\_location\_settings[i] - list\_location\_settings[i-1];

//число символов от начала файла

int location\_from\_start = list\_location\_settings[i-1];

//вызываем метод, который находит настройку в файле

string text\_settings = Get\_setting(fstream, length, location\_from\_start);

//распределяем полученные настройки

if (i == 1)

ind\_selected\_horse = int.Parse(text\_settings);

else if (i == 2)

ind\_select\_filemode = int.Parse(text\_settings);

else

{

list\_for\_one\_horse.Add(text\_settings);

j++;

//ограничение числа элементов форматирования для каждой лошади

if (j == 3)

{

list\_formatting.Add(list\_for\_one\_horse);

list\_for\_one\_horse = new List<string>();

j = 0;

}

}

}

}

download\_check = true;

//очищаем форму

foreach(Panel road in list\_road)

{

road.Controls.Clear();

Controls.Remove(road);

}

list\_pictureBoxes.Clear();

list\_road.Clear();

comboBox\_select\_horse.Items.Clear();

//генерируем загруженный ипподром

Load\_arena(number\_of\_horses);

//передаем элементам формы полученные настройки

form\_menu.comboBox\_number\_horses.SelectedIndex = number\_of\_horses - 2;

comboBox\_select\_horse.SelectedIndex = ind\_selected\_horse;

comboBox\_record.SelectedIndex = ind\_select\_filemode;

Start\_position();

}

catch (Exception ex)

{

MessageBox.Show($"Error.\n\nError message: {ex.Message}\n\n" +

$"Details:\n\n{ex.StackTrace}");

}

}

//метод, который отыскивает настройки в файле произвольного доступа

private string Get\_setting (FileStream fstream, int length, int location\_from\_start)

{

//перемещение указателя

fstream.Seek(location\_from\_start, SeekOrigin.Begin);

byte[] array = new byte[length];

fstream.Read(array, 0, array.Length);

return System.Text.Encoding.Default.GetString(array);

}

//выбор способа записи истории игр

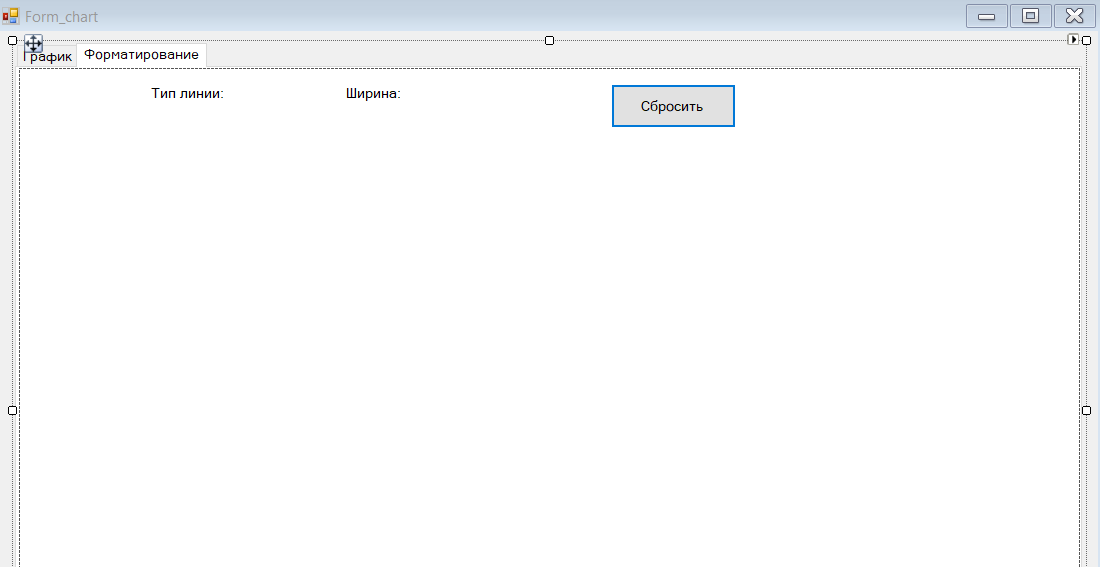
private void comboBox\_record\_SelectedIndexChanged(object sender, EventArgs e)

{

ind\_select\_filemode = comboBox\_record.SelectedIndex;

}

}



public partial class Form\_chart : Form

{

int number\_of\_horses; //количество лошадей, учавствующих в гонке

List<List<string>> list\_formatting = new List<List<string>>(); //набор настроек элементов форматирования для каждой лошади

Form1 fm1;

public List<ComboBox> list\_combobox = new List<ComboBox>(); //коллекция combobox-ов для типа линии

public List<NumericUpDown> list\_numericUpDOwn = new List<NumericUpDown>(); //коллекция numUpDown для толщины линий

public List<Button> list\_button = new List<Button>(); //коллекция кнопок цвета

List<ChartDashStyle> list\_type\_line = new List<ChartDashStyle>() { ChartDashStyle.NotSet, ChartDashStyle.Solid, ChartDashStyle.Dash, ChartDashStyle.Dot }; //список типов линий

public Form\_chart(Form1 form1)

{

InitializeComponent();

fm1 = form1;

number\_of\_horses = fm1.number\_of\_horses;

Chart(number\_of\_horses, fm1.list\_stages, fm1.list\_of\_speeds\_horses);

//Если нажата кнопка "Загрузить пользовательские настройки"

if (fm1.download\_check)

{

list\_formatting = fm1.list\_formatting;

Download\_formatting();

}

else

Restart();

}

//Отрисовка графиков и генерация элементов управления

private void Chart(int number\_of\_horses, List<int> list\_stages, List<List<int>> list\_of\_speeds\_horses)

{

chart1.Titles.Add("Диаграмма скоростей");

chart1.ChartAreas[0].AxisX.Title = "Этап";

chart1.ChartAreas[0].AxisY.Title = "Скорость";

chart1.ChartAreas[0].AxisX.Crossing = 0;

chart1.ChartAreas[0].AxisY.Crossing = 0;

chart1.ChartAreas[0].AxisX.Interval = 3;

chart1.ChartAreas[0].AxisY.Interval = 2;

int y\_start\_location = 52;

int y\_location = y\_start\_location;

string[] type\_line = new string[] { "Без линии", "Сплошная", "Пунктирная", "Точечная" };

for (int i = 0; i < number\_of\_horses; i++)

{

string name\_horse = $"Лошадь {i+1}";

chart1.Series.Add($"Series{i}");

chart1.Series[$"Series{i}"].LegendText = name\_horse;

chart1.Series[$"Series{i}"].ChartType = SeriesChartType.Spline;

chart1.Series[$"Series{i}"].MarkerStyle = MarkerStyle.Circle;

chart1.Series[i].Points.DataBindXY(list\_stages, list\_of\_speeds\_horses[i]);

Label label\_name\_horse = new Label();

ComboBox comboBox\_type\_line = new ComboBox();

NumericUpDown numericUp\_width = new NumericUpDown();

Button button\_color = new Button();

//размещение и настройка элементов управления

int x\_location = 23;

label\_name\_horse.Location = new Point(x\_location, y\_location);

label\_name\_horse.Text = name\_horse;

label\_name\_horse.Size = new Size(70, 17);

comboBox\_type\_line.Size = new Size(108, 24);

x\_location = label\_type\_line.Location.X;

comboBox\_type\_line.Location = new Point(x\_location, y\_location);

comboBox\_type\_line.DropDownStyle = ComboBoxStyle.DropDownList;

comboBox\_type\_line.Items.AddRange(type\_line);

comboBox\_type\_line.Tag = i;

comboBox\_type\_line.SelectedIndexChanged += new EventHandler(comboBox\_type\_lineSelectedIndexChanged);

x\_location = label\_width\_line.Location.X;

numericUp\_width.Location = new Point(x\_location, y\_location);

numericUp\_width.Size = new Size(48, 22);

numericUp\_width.Tag = i;

numericUp\_width.ValueChanged += new EventHandler(numericUpDown\_ValueChanged);

x\_location = label\_width\_line.Location.X + 85;

button\_color.Location = new Point(x\_location, y\_location);

button\_color.Text = "Цвет";

button\_color.Font = new Font("Microsoft Sans Serif", 9F, FontStyle.Bold, GraphicsUnit.Point, (byte)204);

button\_color.BackColor = chart1.Series[$"Series{i}"].Color;

button\_color.ForeColor = Color.White;

button\_color.Size = new Size(73, 24);

button\_color.Tag = i;

button\_color.Click += new EventHandler(btn\_color\_Click);

y\_location = y\_start\_location \* (i + 2);

tabPage2.Controls.Add(label\_name\_horse);

tabPage2.Controls.Add(comboBox\_type\_line);

tabPage2.Controls.Add(numericUp\_width);

tabPage2.Controls.Add(button\_color);

list\_combobox.Add(comboBox\_type\_line);

list\_numericUpDOwn.Add(numericUp\_width);

list\_button.Add(button\_color);

}

}

private void Form\_chart\_FormClosing(object sender, FormClosingEventArgs e)

{

chart1.Series.Clear();

}

//нажатие на кнопку "Цвет"

private void btn\_color\_Click(object sender, EventArgs e)

{

Button clickedButton = (Button)sender;

//меняем цвет линий и кнопок

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

chart1.Series[$"Series{clickedButton.Tag}"].Color = colorDialog1.Color;

clickedButton.BackColor = chart1.Series[$"Series{clickedButton.Tag}"].Color;

}

}

//метод, срабатывающий при изменении значения комбобокса типов линий

private void comboBox\_type\_lineSelectedIndexChanged(object sender, EventArgs e)

{

ComboBox comboBox = (ComboBox)sender;

int select\_ind = comboBox.SelectedIndex;

chart1.Series[$"Series{comboBox.Tag}"].BorderDashStyle = list\_type\_line[select\_ind];

}

//метод, срабатывающий при изменении значения ширины линии

private void numericUpDown\_ValueChanged(object sender, EventArgs e)

{

NumericUpDown upDown = (NumericUpDown)sender;

chart1.Series[$"Series{upDown.Tag}"].BorderWidth = (int)upDown.Value;

}

//метод, возвращающий настройки по умолчанию

private void Restart()

{

for (int i=0; i<number\_of\_horses;i++)

{

list\_combobox[i].SelectedIndex = 1;

list\_numericUpDOwn[i].Value = 1;

chart1.Series[$"Series{i}"].Color = default;

list\_button[i].BackColor = default;

}

}

//метод, преобразующий текстовые данные форматирования - в настоящие

private void Download\_formatting()

{

for (int i = 0; i < number\_of\_horses; i++)

{

for (int j = 0; j < 3; j++)

{

if (j == 2)

{

Color color = ColorTranslator.FromHtml(list\_formatting[i][j]);

chart1.Series[$"Series{i}"].Color = color;

list\_button[i].BackColor = color;

}

else

{

int value = int.Parse(list\_formatting[i][j]);

if (j == 0)

list\_combobox[i].SelectedIndex = value;

else

list\_numericUpDOwn[i].Value = value;

}

}

}

}

//нажатие кнопки "Сбросить"

private void button\_restart\_Click(object sender, EventArgs e)

{

Restart();

fm1.download\_check = false;

}

private void Form\_chart\_Load(object sender, EventArgs e)

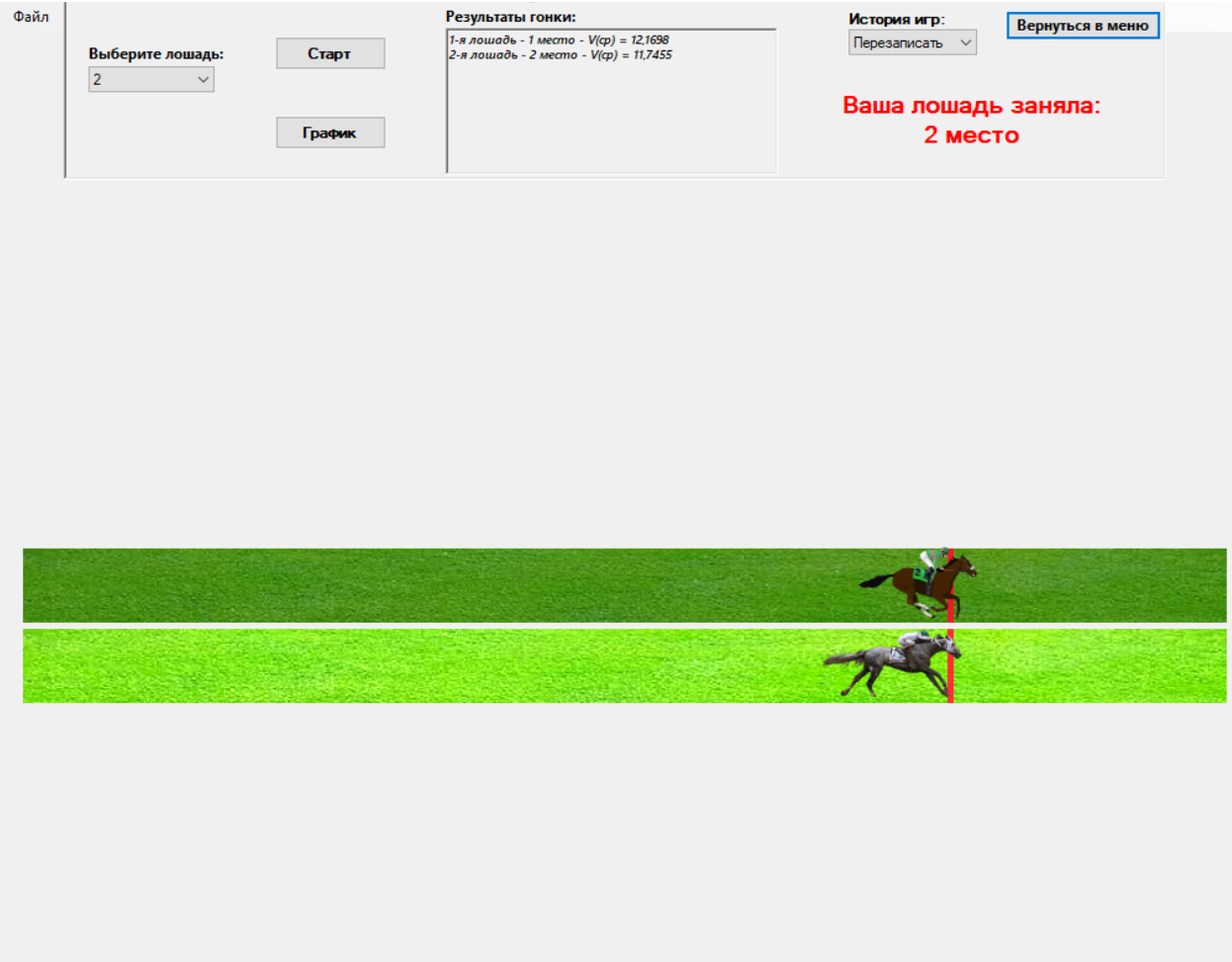
{

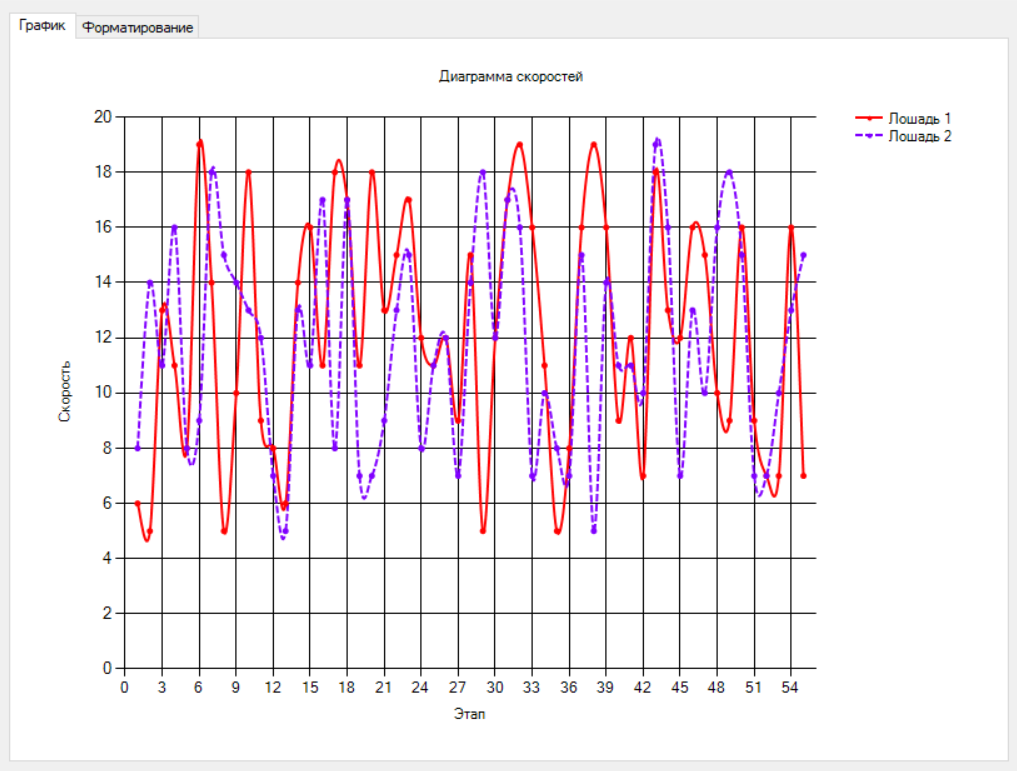
tabControl1.Height = 600;

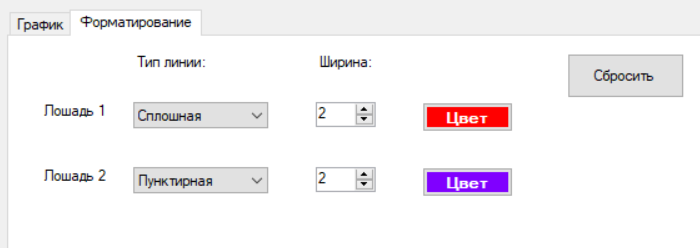
}

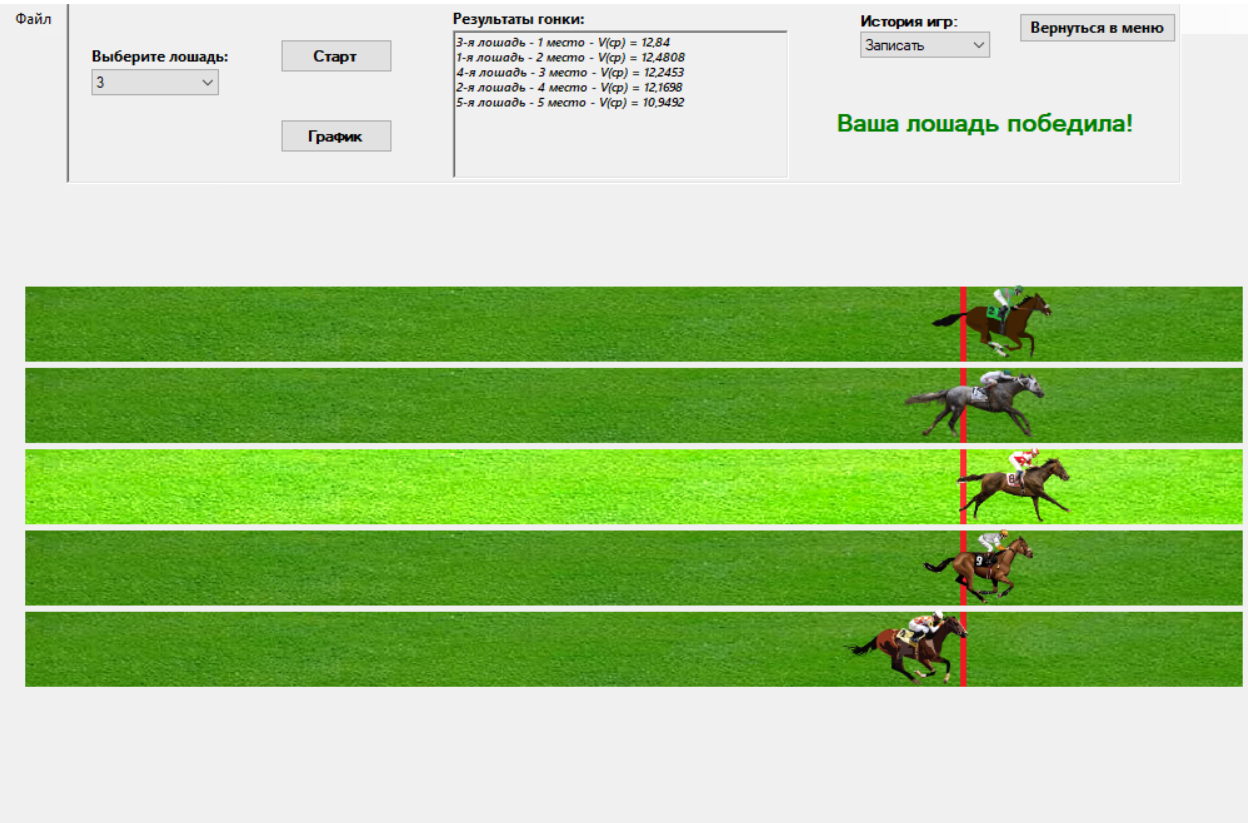
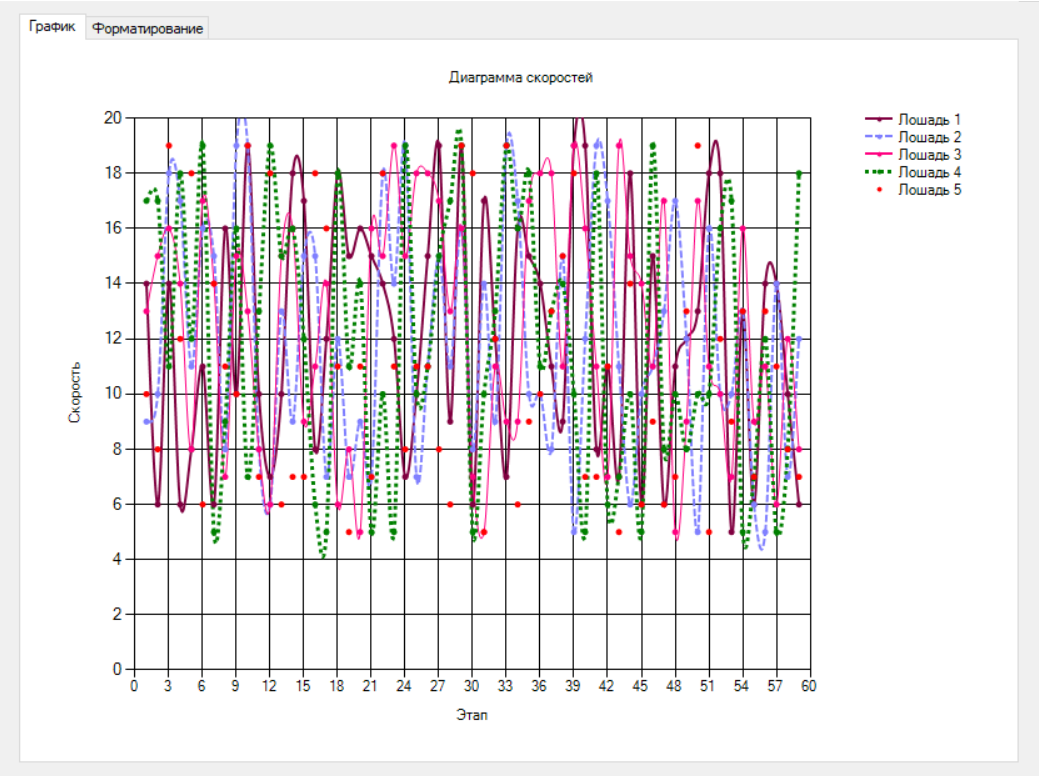
}

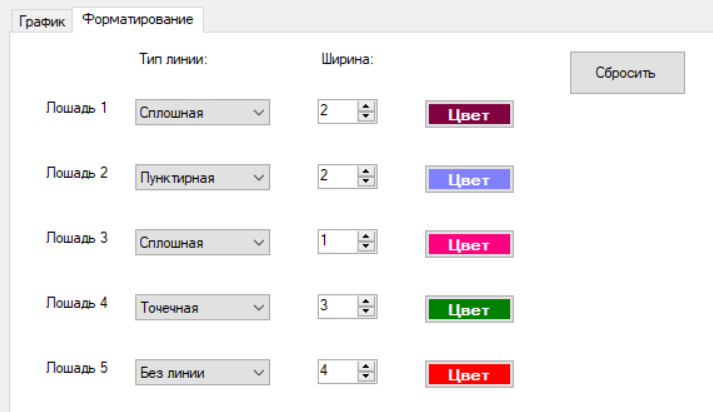
**Тестирование программы**

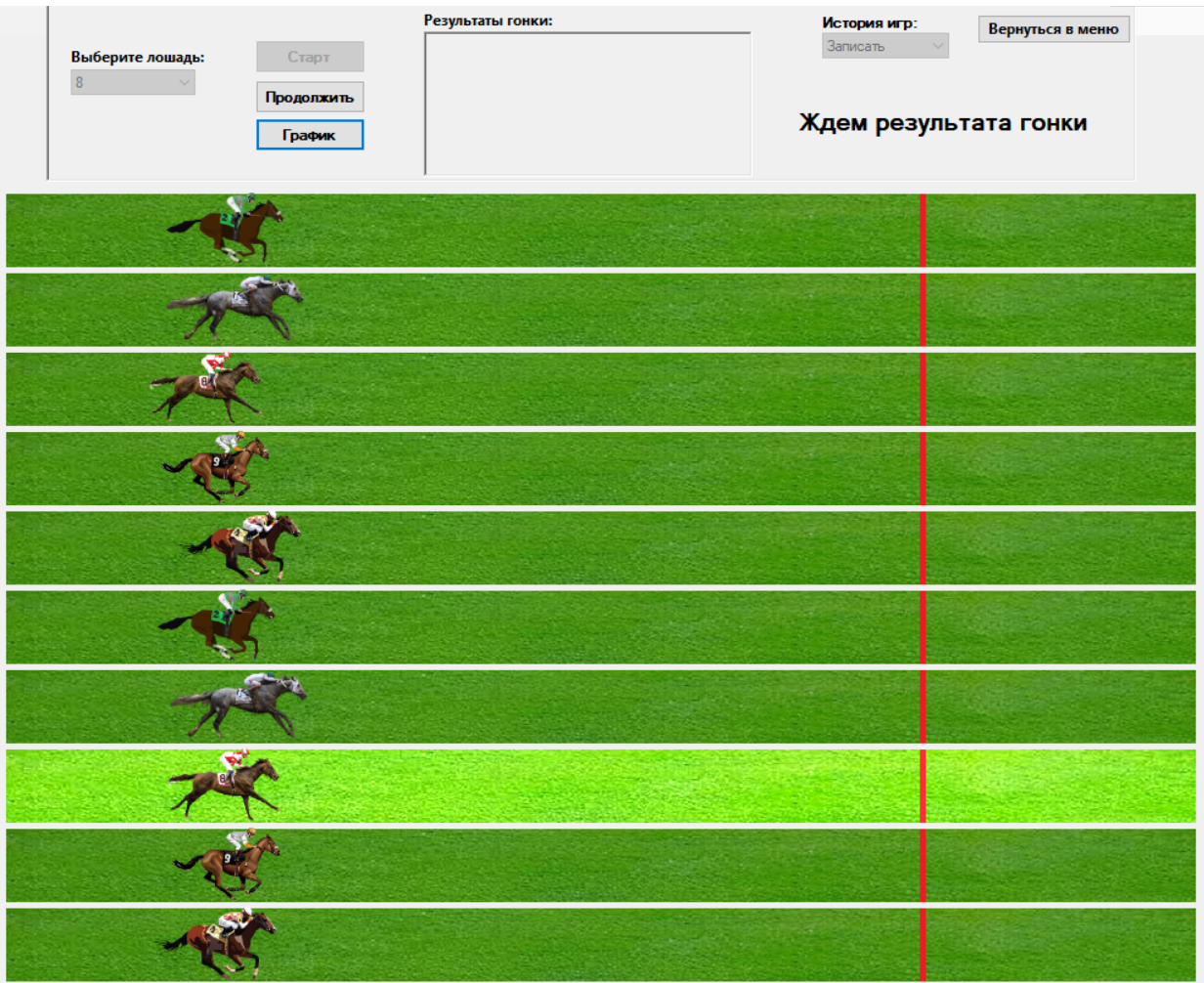
**Запуск 1:**

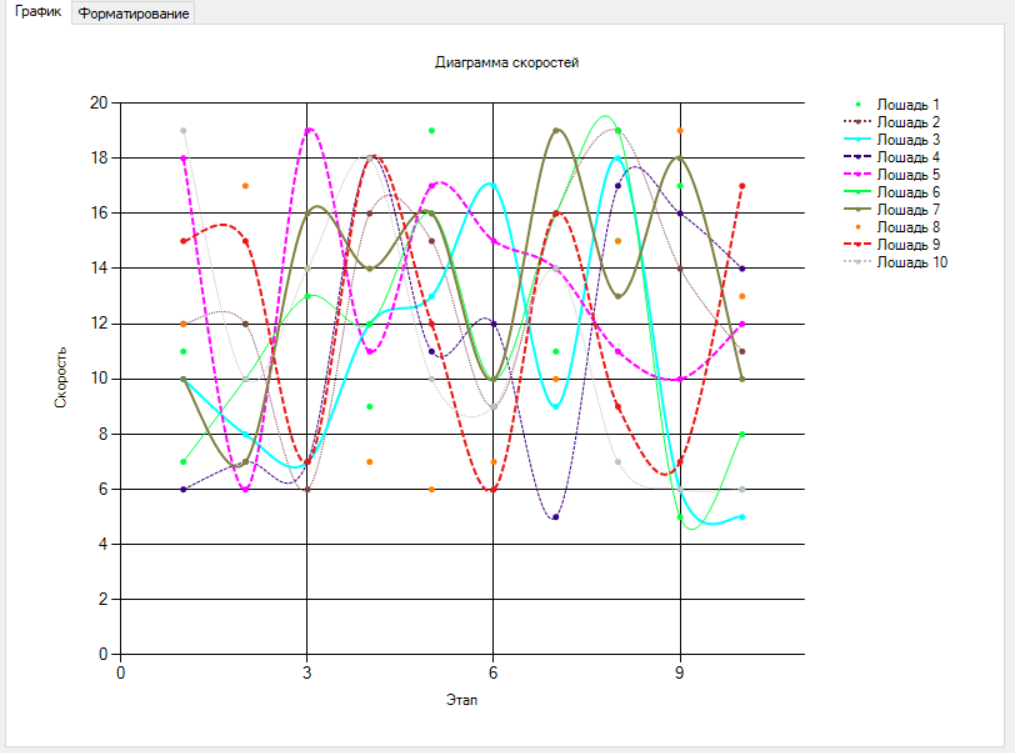




З**апуск 2:**



**Запуск 3 (на паузе):**





**Итоги**

Таким образом, мне удалось реализовать и визуализировать алгоритм игры «Ипподром» на платформе Windows Forms (.NET Framework) на языке C#. За время выполнения этой работы я узнал много нового и полезного, что постарался продемонстрировать на практике.