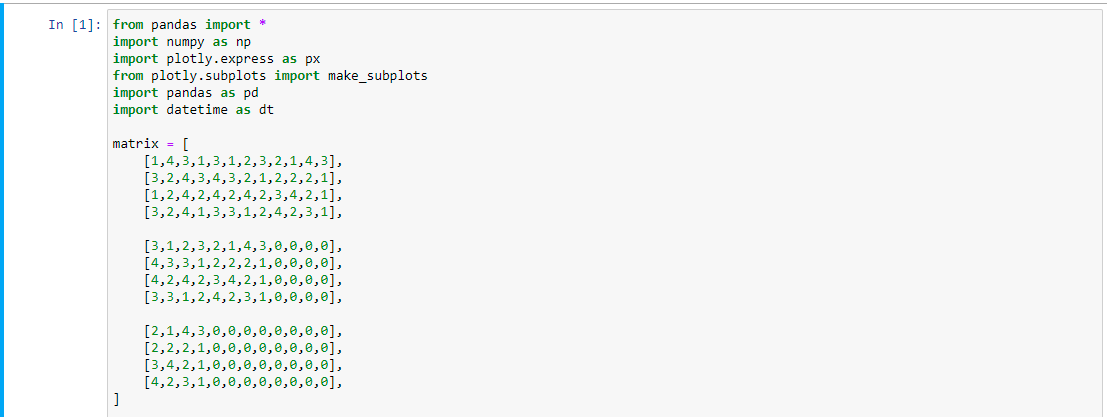
Задача

Представить с помощью линейных диаграмм процесс выполнения 5 процессов на 5 процессорах в асинхронном и двух синхронных режимах, если матрица времен выполнения блоков имеет вид:

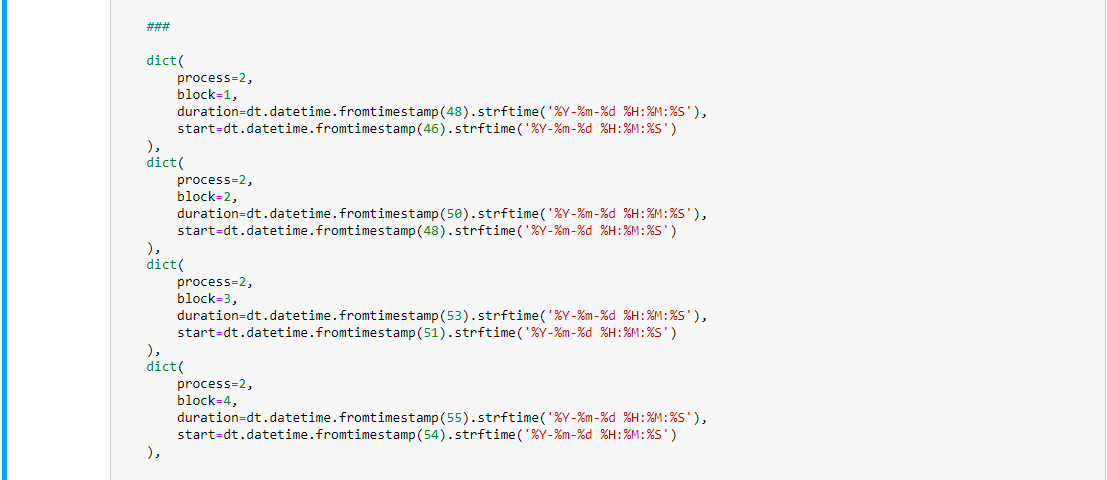
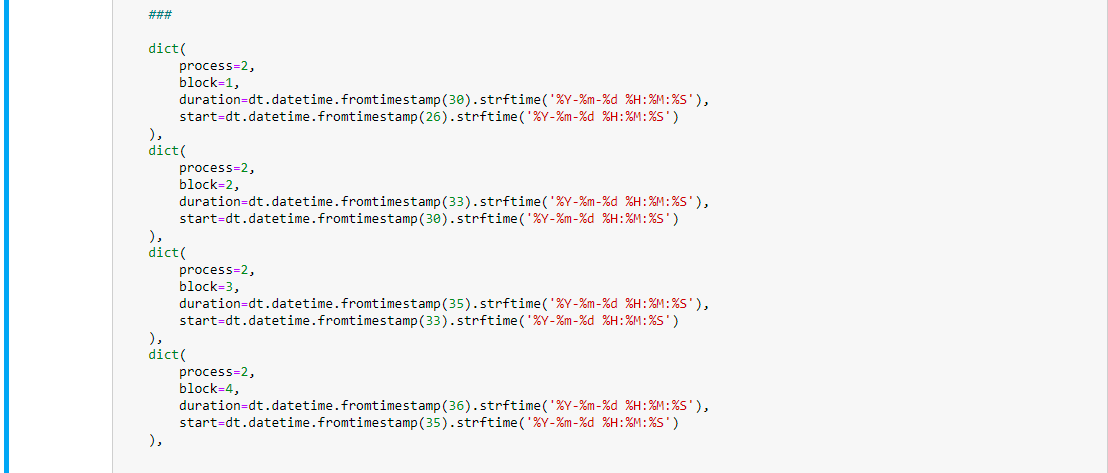
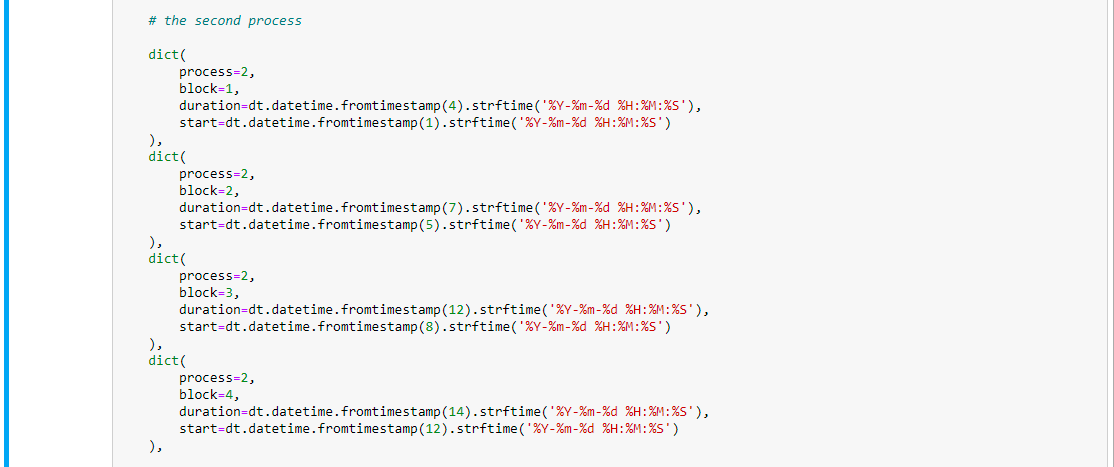
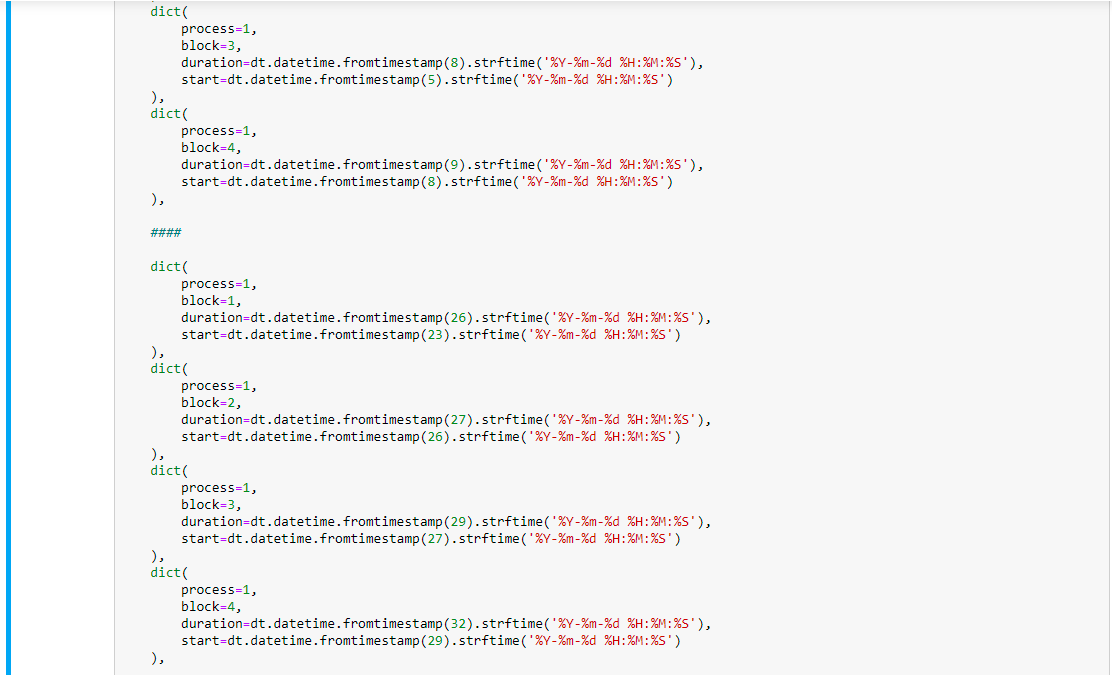


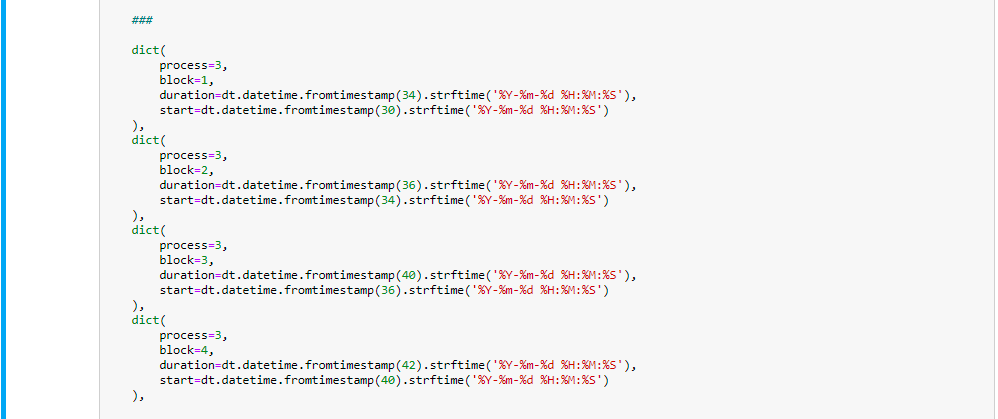
Найти величину общего времени выполнения заданного числа процессов в каждом из режимов (асинхронный и два синхронных). Провести сравнительный анализ общего времени в каждом из режимов. Построить несовмещённые и совмещенную диаграммы.

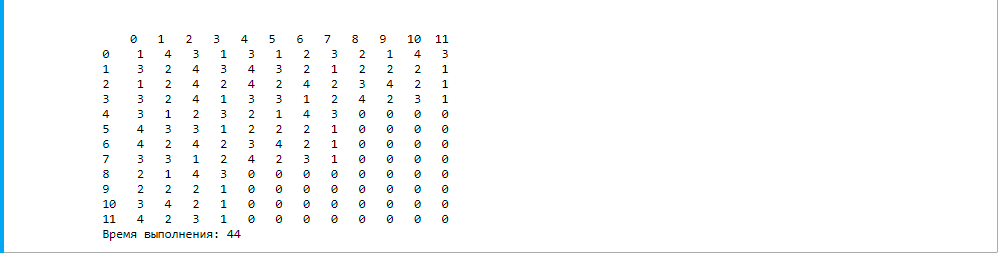
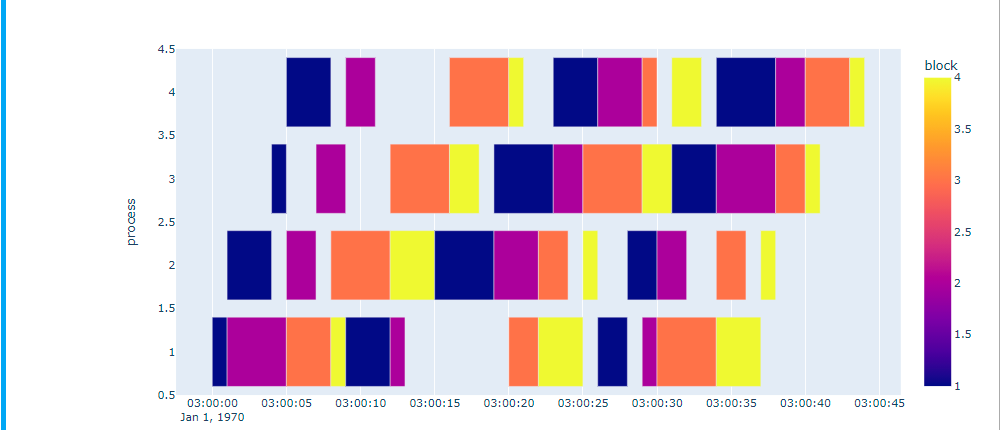
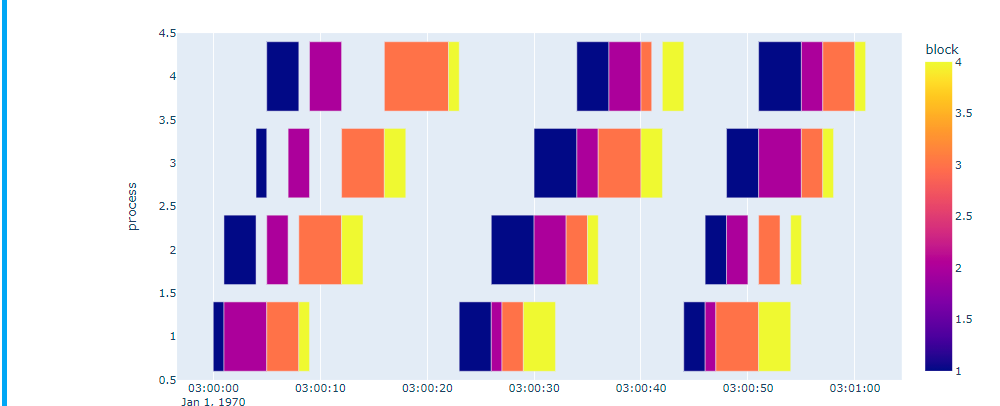
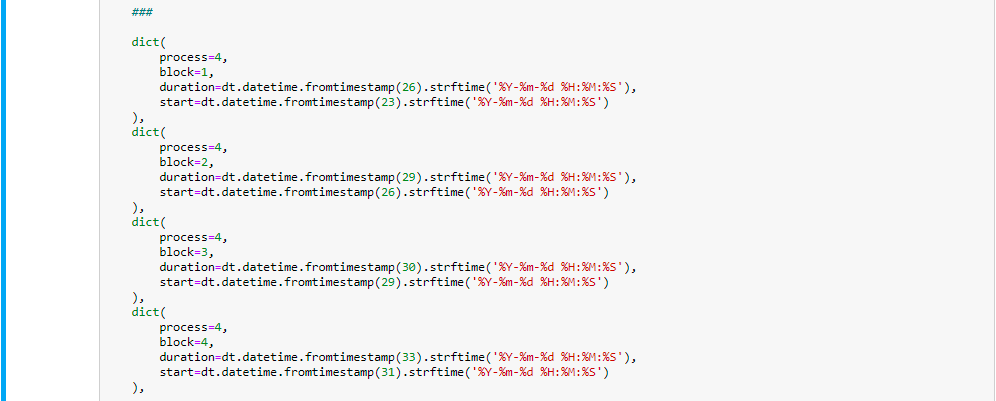
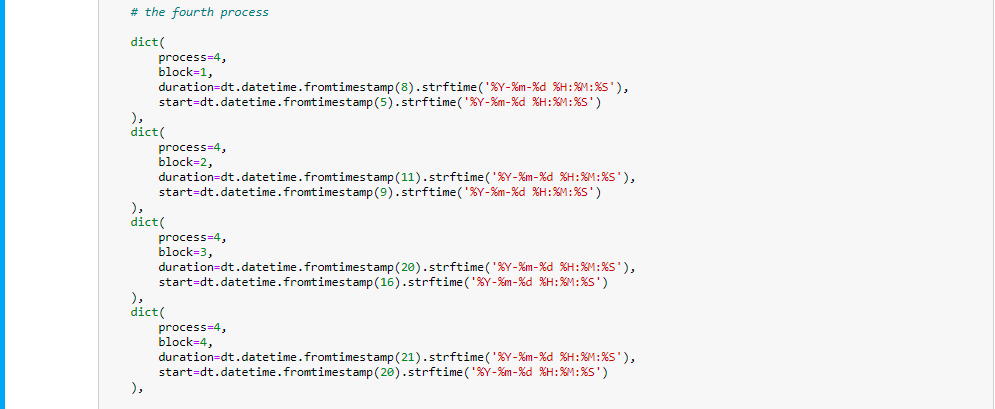
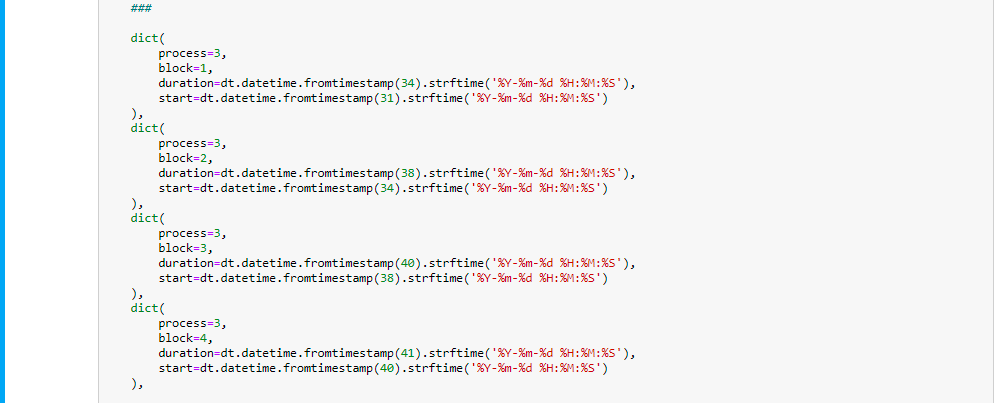
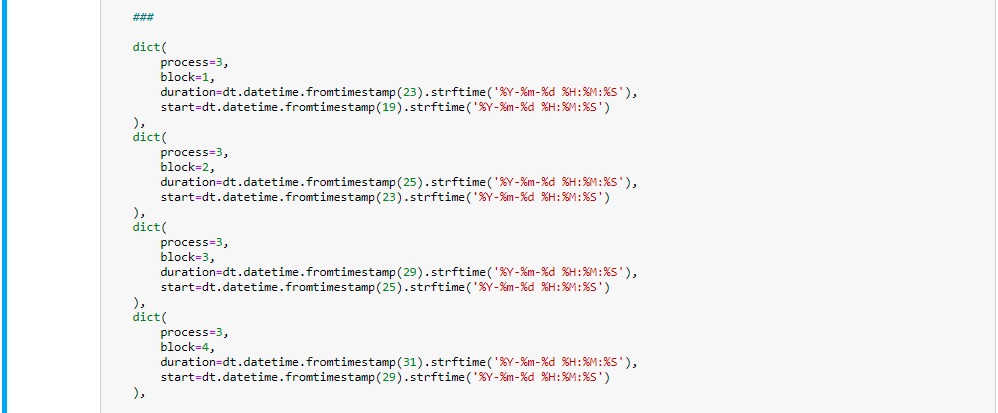
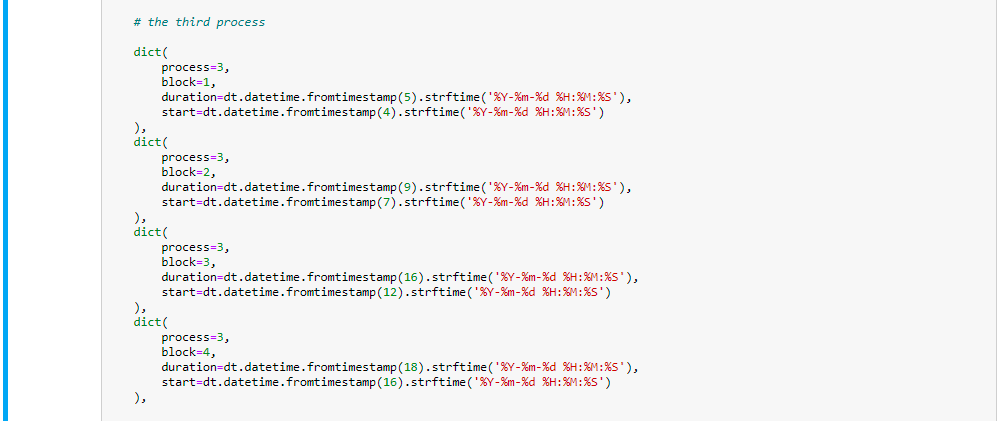
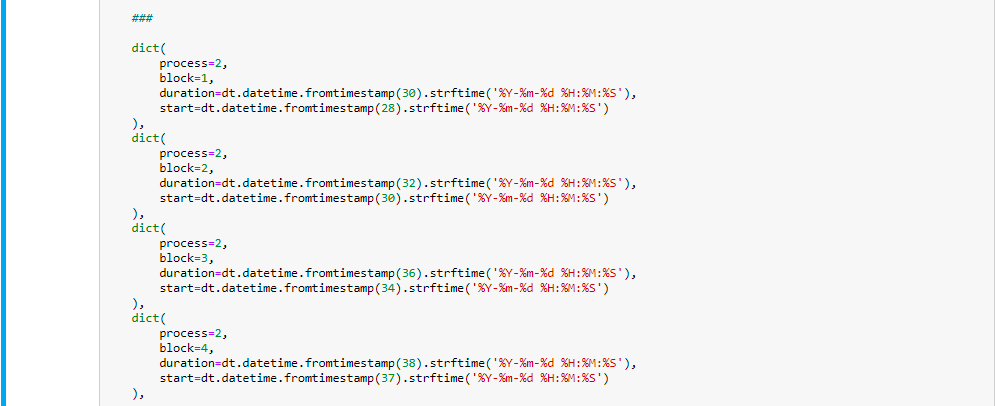
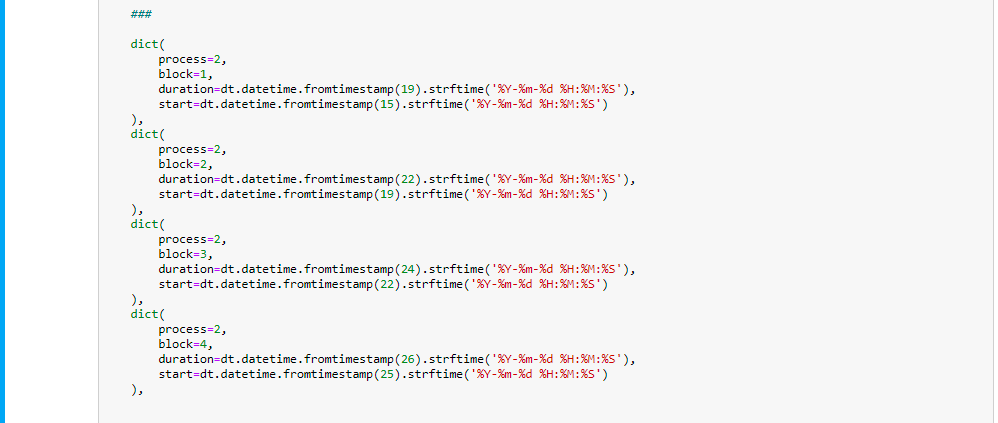
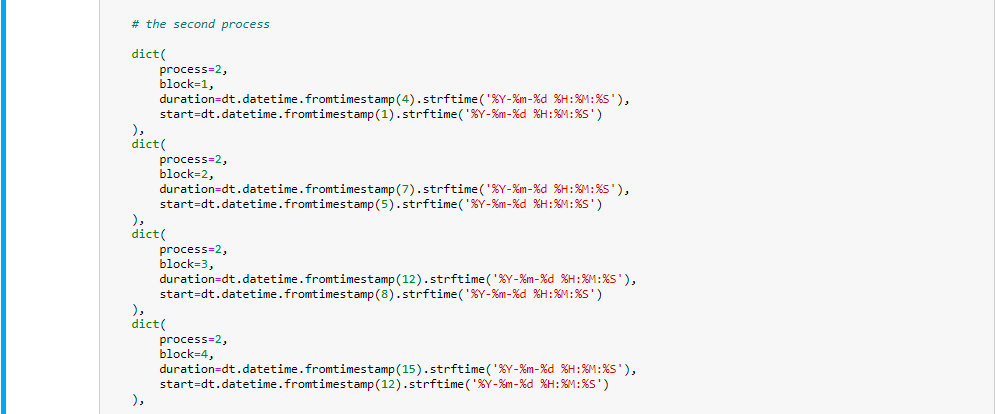
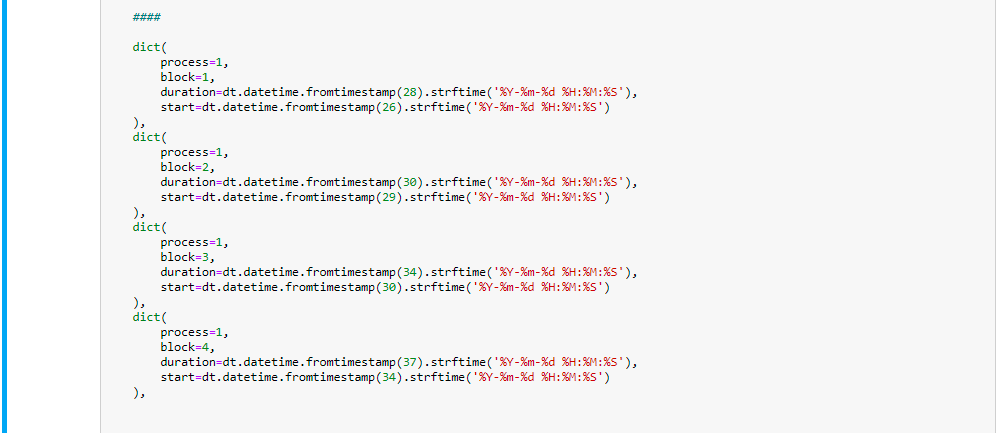
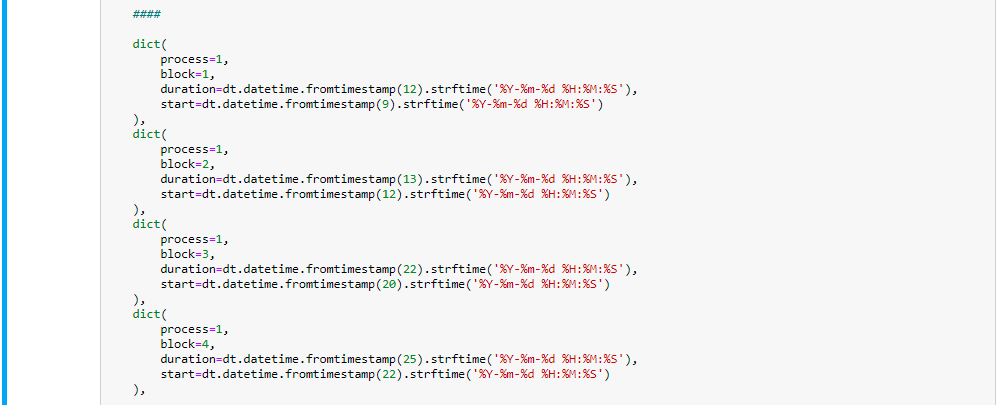
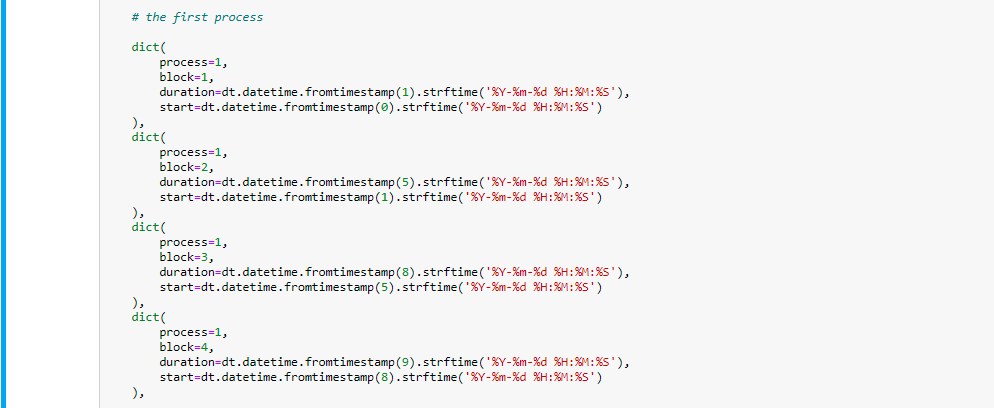
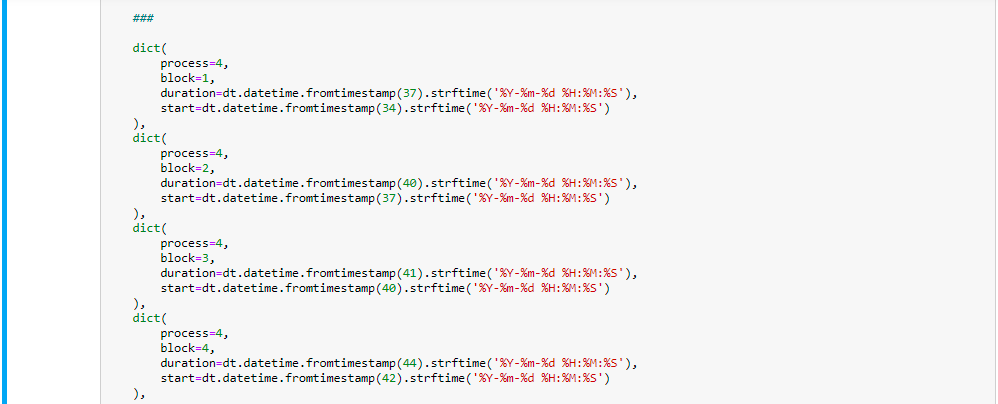
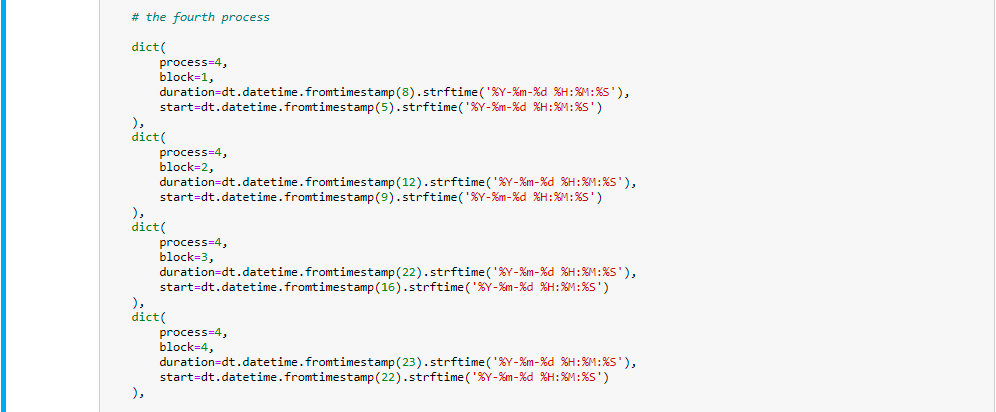
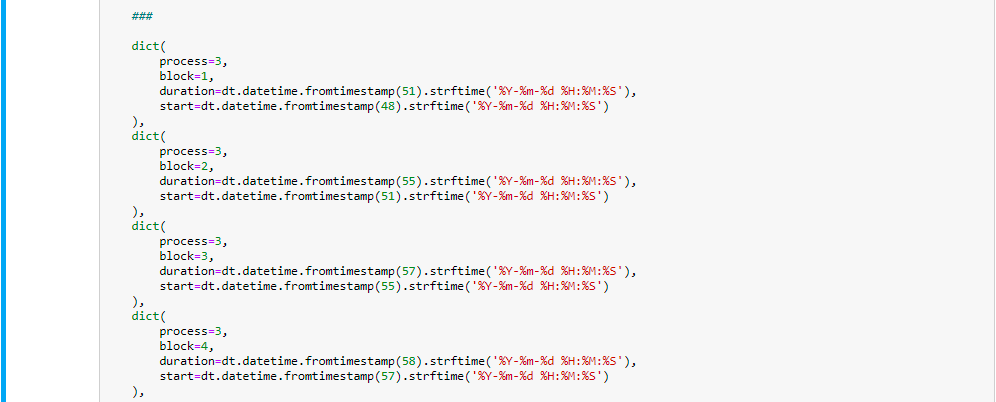
Программная реализация задач:











Задача

Пусть имеется программный ресурс, длительность выполнения которого составляет T=100 тактов. Предполагаемое число процессоров n=21. Накладные расходы на организацию параллельного выполнения составляют величину τ= 0,8 такта для каждого блока. Найти оптимальное число блоков s0 и величину выигрыша по времени в тактах E(s) если

а) n=p=21,

б) n= 21, p=7,

в) n= 21, p=3,

г) n= 21, p=5.

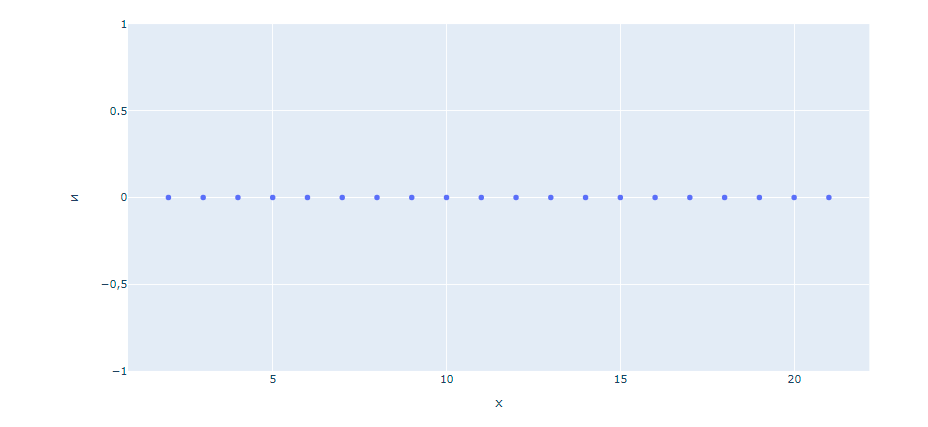
Задача 3.

Пусть θs=(3,5,2,5,7,6,4,8,6,5) n=10.

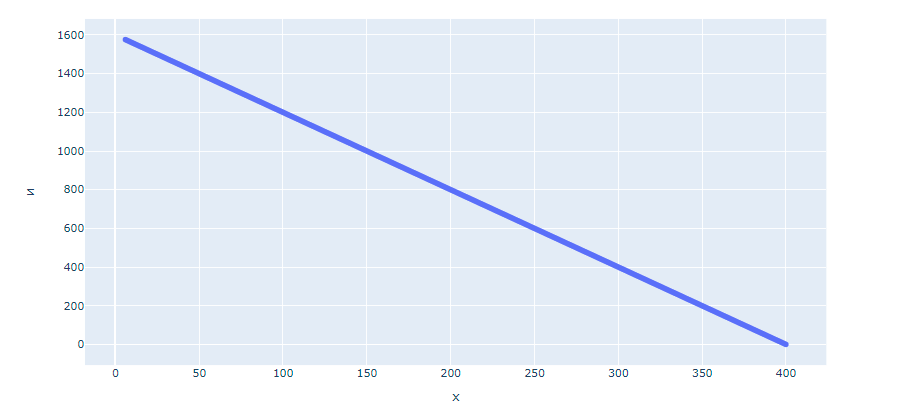
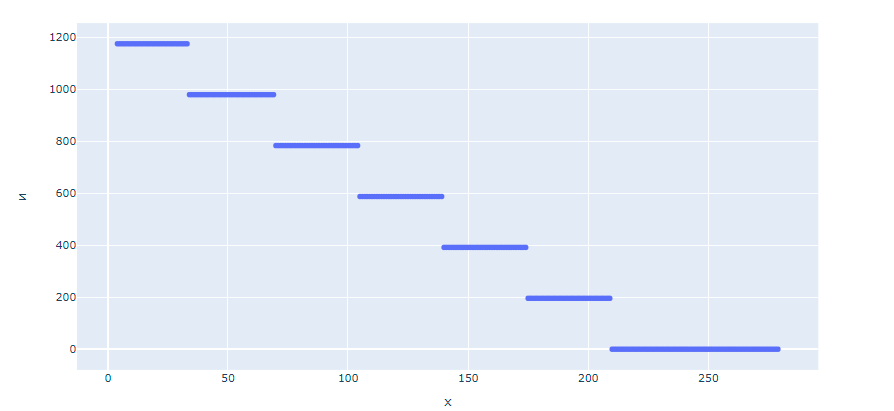
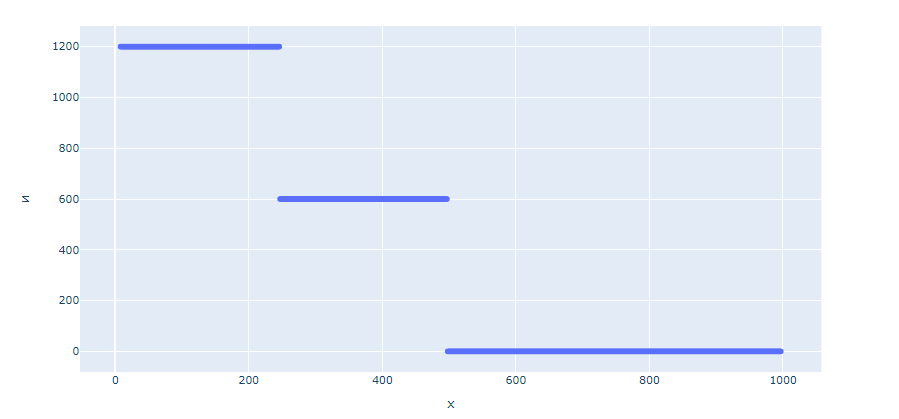
а) Найти минимальное число процессоров p\*, необходимых для выполнения заданного числа процессов за минимальное время.

б) Найти минимальное число процессоров p\*, необходимых для выполнения заданного числа процессов за минимальное время, если структурирование оптимальное.

Программная реализация задач:







Программная реализация задач:

Задача

Пусть число процессоров , число процессов , и число блоков , а времена выполнения блоков каждым из процессов заданы матрицей размерности 7x7



Используя формулы (2.13-2.15) найти минимальное общее время выполнения однородных процессов в условиях асинхронного и двух синхронных режимов. Сравнить результаты. Построить несовмещённые и совмещенную диаграммы.

Програмная реализация задачи:

Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using ZedGraph;

namespace Lab1

{

public partial class Рисование : Form

{

public int sprocessors = 0;

public int nprocessors = 0;

public int[,] Tmatrix = new int[7,7];

public Рисование()

{

InitializeComponent();

this.MouseDown += new MouseEventHandler(Рисование\_MouseDown);

sprocessors = Convert.ToInt32(textBox1.Text);

nprocessors = Convert.ToInt32(textBox2.Text);

}

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

{

int[,] Tmatrix = new int[nprocessors, sprocessors];

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

{

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

{

Tmatrix[i, j] = Convert.ToInt32(dataGridView1.Rows[i].Cells[j].Value);

}

}

zedGraphControl1.Visible = true;

GraphPane pane = zedGraphControl1.GraphPane;

pane.Title.Text = "Асинхронный режим распределенных конкурирующих процессов";

pane.XAxis.Title.Text = "T (p,n,s,e)";

pane.YAxis.Title.Text = "P";

pane.CurveList.Clear();

int[] last\_koef\_array = new int[nprocessors];

int last\_dot = 0;

LineItem drawing = new LineItem("0");

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

{

int x= 0;

int x1 = last\_koef\_array[0];

for (int y = 0; y < sprocessors; y++)

{

PointPairList temp\_list\_of\_process = new PointPairList();

for (int i = x1; i <= Tmatrix[j, y] + x1; i++)

{

temp\_list\_of\_process.Add(i, y + 1);

x = i;

}

last\_koef\_array[y] = x;

if (y != (sprocessors - 1))

{

if (last\_koef\_array[y + 1] < x)

{

x1 = x;

}

else

{

x1 = last\_koef\_array[y + 1];

}

}

else

{

last\_dot = x;

}

string name = "t" + (j+1)+","+(y+1);

if (y == 0)

{

switch (j)

{

case 0: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Blue, SymbolType.Diamond); break;

case 1: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Red, SymbolType.Diamond); break;

case 2: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.LimeGreen, SymbolType.Diamond); break;

case 3: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Green, SymbolType.Diamond); break;

case 4: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Black, SymbolType.Diamond); break;

case 5: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Orange, SymbolType.Diamond); break;

case 6: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Purple, SymbolType.Diamond); break;

default: break;

}

}

if (y == 1)

{

switch (j)

{

case 0: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Blue, SymbolType.TriangleDown); break;

case 1: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Red, SymbolType.TriangleDown); break;

case 2: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.LimeGreen, SymbolType.TriangleDown); break;

case 3: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Green, SymbolType.TriangleDown); break;

case 4: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Black, SymbolType.TriangleDown); break;

case 5: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Orange, SymbolType.TriangleDown); break;

case 6: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Purple, SymbolType.TriangleDown); break;

default: break;

}

}

else

{

switch (j)

{

case 0: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Blue, SymbolType.Triangle); break;

case 1: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Red, SymbolType.Triangle); break;

case 2: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.LimeGreen, SymbolType.Triangle); break;

case 3: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Green, SymbolType.Triangle); break;

case 4: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Black, SymbolType.Triangle); break;

case 5: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Orange, SymbolType.Triangle); break;

case 6: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Purple, SymbolType.Triangle); break;

default: break;

}

}

}

}

PointPairList end\_list\_of\_process = new PointPairList();

end\_list\_of\_process.Add(last\_dot, 0);

string end\_text = "T end:" + last\_dot.ToString();

textBox3.Text = last\_dot.ToString();

drawing = pane.AddCurve(end\_text, end\_list\_of\_process, Color.Red, SymbolType.None);

zedGraphControl1.GraphPane.AxisChange();

zedGraphControl1.Invalidate();

}

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

{

zedGraphControl1.GraphPane.CurveList.Clear();

}

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

{

this.Close();

}

private void Рисование\_MouseDown(object sender, MouseEventArgs e)

{

this.Capture = false;

Win32.SendMessage(this.Handle, Win32.WM\_NCLBUTTONDOWN, Win32.HTCAPTION, 0);

}

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

{

sprocessors = Convert.ToInt32(textBox1.Text);

nprocessors = Convert.ToInt32(textBox2.Text);

DrawTableForMatrix(sprocessors, nprocessors);

}

private void DrawTableForMatrix(int s,int n)

{

dataGridView1.Columns.Clear();

dataGridView1.ColumnCount = s;

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

{

dataGridView1.Columns[i].Width = 15;

}

dataGridView1.RowCount = n;

}

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

{

DrawTableForMatrix(7, 7);

int[,] DefaultMatrix = {

{ 1,3,4,5,3,2,1 },

{ 3,4,5,3,2,1,1 },

{ 4,5,3,2,1,1,3 },

{ 5,3,2,1,1,3,4 },

{ 3,2,1,1,3,4,5 },

{ 2,1,1,3,4,5,3 },

{ 1,1,3,4,5,3,2 },

};

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

{

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

{

dataGridView1.Rows[i].Cells[j].Value = DefaultMatrix[i, j];

}

}

Tmatrix = DefaultMatrix;

}

private void Draw2\_Click(object sender, EventArgs e)

{

zedGraphControl1.Visible = true;

GraphPane pane = zedGraphControl1.GraphPane;

pane.Title.Text = "Первый синхронный режим \n распределенных конкурирующих процессов";

pane.XAxis.Title.Text = "T (p,n,s,e)";

pane.YAxis.Title.Text = "P";

pane.CurveList.Clear();

int[] last\_koef\_array = new int[sprocessors];

int last\_dot = 0;

LineItem drawing = new LineItem("0");

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

{

int x = 0;

for (int y = 0; y < sprocessors-1; y++)

{

if(last\_koef\_array[y] + Tmatrix[j, y]< last\_koef\_array[y+1])

{

for(int y1=0;y1<=y;y1++)

{

last\_koef\_array[y1] += last\_koef\_array[y + 1] - Tmatrix[j, y]- last\_koef\_array[y];

}

}

}

int x1 = last\_koef\_array[0];

for (int y = 0; y < sprocessors; y++)

{

PointPairList temp\_list\_of\_process = new PointPairList();

for (int i = x1; i <= Tmatrix[j, y] + x1; i++)

{

temp\_list\_of\_process.Add(i, y + 1);

x = i;

}

last\_koef\_array[y] = x;

if (y != (sprocessors - 1))

{

if (last\_koef\_array[y + 1] < x)

{

x1 = x;

}

else

{

x1 = last\_koef\_array[y + 1];

}

}

else

{

last\_dot = x;

}

string name = "t" + (j + 1) + "," + (y + 1);

if (y == 0)

{

switch (j)

{

case 0: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Blue, SymbolType.Diamond); break;

case 1: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Red, SymbolType.Diamond); break;

case 2: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.LimeGreen, SymbolType.Diamond); break;

case 3: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Green, SymbolType.Diamond); break;

case 4: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Black, SymbolType.Diamond); break;

case 5: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Orange, SymbolType.Diamond); break;

case 6: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Purple, SymbolType.Diamond); break;

default: break;

}

}

if (y == 1)

{

switch (j)

{

case 0: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Blue, SymbolType.TriangleDown); break;

case 1: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Red, SymbolType.TriangleDown); break;

case 2: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.LimeGreen, SymbolType.TriangleDown); break;

case 3: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Green, SymbolType.TriangleDown); break;

case 4: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Black, SymbolType.TriangleDown); break;

case 5: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Orange, SymbolType.TriangleDown); break;

case 6: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Purple, SymbolType.TriangleDown); break;

default: break;

}

}

else

{

switch (j)

{

case 0: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Blue, SymbolType.Triangle); break;

case 1: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Red, SymbolType.Triangle); break;

case 2: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.LimeGreen, SymbolType.Triangle); break;

case 3: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Green, SymbolType.Triangle); break;

case 4: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Black, SymbolType.Triangle); break;

case 5: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Orange, SymbolType.Triangle); break;

case 6: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Purple, SymbolType.Triangle); break;

default: break;

}

}

}

}

PointPairList end\_list\_of\_process = new PointPairList();

end\_list\_of\_process.Add(last\_dot, 0);

string end\_text = "T end:" + last\_dot.ToString();

textBox4.Text = last\_dot.ToString();

drawing = pane.AddCurve(end\_text, end\_list\_of\_process, Color.Red, SymbolType.None);

zedGraphControl1.GraphPane.AxisChange();

zedGraphControl1.Invalidate();

}

private void Draw3\_Click(object sender, EventArgs e)

{

zedGraphControl1.Visible = true;

GraphPane pane = zedGraphControl1.GraphPane;

pane.Title.Text = "Второй синхронный режим \n распределенных конкурирующих процессов";

pane.XAxis.Title.Text = "T (p,n,s,e)";

pane.YAxis.Title.Text = "P";

pane.CurveList.Clear();

int[] last\_koef\_array = new int[nprocessors];

int last\_dot = 0;

LineItem drawing = new LineItem("0");

for (int y = 0; y < sprocessors; y++)

{

int x = 0;

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

{

if(last\_koef\_array[j]+Tmatrix[j, y]< last\_koef\_array[j+1])

{

for (int j1 = 0; j1 <= j; j1++)

{

last\_koef\_array[j1] += last\_koef\_array[j + 1] - last\_koef\_array[j] - Tmatrix[j, y];

}

}

}

int x1 = last\_koef\_array[0];

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

{

PointPairList temp\_list\_of\_process = new PointPairList();

for (int i = x1; i <= Tmatrix[j, y] + x1; i++)

{

temp\_list\_of\_process.Add(i, y + 1);

x = i;

}

x1 = x;

last\_koef\_array[j] = x1;

last\_dot = x;

string name = "t" + (j + 1) + "," + (y + 1);

switch (j)

{

case 0: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Blue, SymbolType.TriangleDown); break;

case 1: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Red, SymbolType.TriangleDown); break;

case 2: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.LimeGreen, SymbolType.TriangleDown); break;

case 3: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Green, SymbolType.TriangleDown); break;

case 4: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Black, SymbolType.TriangleDown); break;

case 5: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Orange, SymbolType.TriangleDown); break;

case 6: drawing = pane.AddCurve(name, temp\_list\_of\_process, Color.Purple, SymbolType.TriangleDown); break;

default: break;

}

}

}

PointPairList end\_list\_of\_process = new PointPairList();

end\_list\_of\_process.Add(last\_dot, 0);

string end\_text = "T end:" + last\_dot.ToString();

textBox5.Text = last\_dot.ToString();

drawing = pane.AddCurve(end\_text, end\_list\_of\_process, Color.Red, SymbolType.None);

zedGraphControl1.GraphPane.AxisChange();

zedGraphControl1.Invalidate();

}

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

{

int[,] Tmatrix = new int[nprocessors,sprocessors];

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

{

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

{

Tmatrix[i,j] = Convert.ToInt32(dataGridView1.Rows[i].Cells[j].Value);

}

}

}

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

{

DrawTableForMatrix(3, 4);

int[,] DefaultMatrix = {

{ 4,2,3 },

{ 1,4,1 },

{ 3,3,2 },

{ 3,1,2 },

};

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

{

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

{

dataGridView1.Rows[i].Cells[j].Value = DefaultMatrix[i, j];

}

}

Tmatrix = DefaultMatrix;

}

}

}

App.config

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

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />

</startup>

</configuration>

Form1.Designer.cs

namespace Lab1

{

partial class Рисование

{

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

this.Draw = new System.Windows.Forms.Button();

this.zedGraphControl1 = new ZedGraph.ZedGraphControl();

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.textBox1 = new System.Windows.Forms.TextBox();

this.textBox2 = new System.Windows.Forms.TextBox();

this.dataGridView1 = new System.Windows.Forms.DataGridView();

this.CloseButton = new System.Windows.Forms.Button();

this.GetMatrix = new System.Windows.Forms.Button();

this.DefaultValue = new System.Windows.Forms.Button();

this.Draw2 = new System.Windows.Forms.Button();

this.Draw3 = new System.Windows.Forms.Button();

this.NewValue = new System.Windows.Forms.Button();

this.textBox3 = new System.Windows.Forms.TextBox();

this.textBox4 = new System.Windows.Forms.TextBox();

this.textBox5 = new System.Windows.Forms.TextBox();

this.label3 = new System.Windows.Forms.Label();

this.SecondDefaultValue = new System.Windows.Forms.Button();

((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).BeginInit();

this.SuspendLayout();

//

// Draw

//

this.Draw.Location = new System.Drawing.Point(571, 343);

this.Draw.Name = "Draw";

this.Draw.Size = new System.Drawing.Size(75, 23);

this.Draw.TabIndex = 0;

this.Draw.Text = "Draw№1";

this.Draw.UseVisualStyleBackColor = true;

this.Draw.Click += new System.EventHandler(this.Draw\_Click);

//

// zedGraphControl1

//

this.zedGraphControl1.Location = new System.Drawing.Point(0, -1);

this.zedGraphControl1.Name = "zedGraphControl1";

this.zedGraphControl1.ScrollGrace = 0D;

this.zedGraphControl1.ScrollMaxX = 0D;

this.zedGraphControl1.ScrollMaxY = 0D;

this.zedGraphControl1.ScrollMaxY2 = 0D;

this.zedGraphControl1.ScrollMinX = 0D;

this.zedGraphControl1.ScrollMinY = 0D;

this.zedGraphControl1.ScrollMinY2 = 0D;

this.zedGraphControl1.Size = new System.Drawing.Size(565, 435);

this.zedGraphControl1.TabIndex = 2;

this.zedGraphControl1.Visible = false;

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(577, 12);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(125, 39);

this.label1.TabIndex = 3;

this.label1.Text = "(s) Число блоков\r\nструктурированного \r\nпрограммного ресурса";

this.label1.TextAlign = System.Drawing.ContentAlignment.TopRight;

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(612, 64);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(90, 39);

this.label2.TabIndex = 4;

this.label2.Text = "(n) Число \r\nконкурирующих \r\nпроцессов";

this.label2.TextAlign = System.Drawing.ContentAlignment.TopRight;

//

// textBox1

//

this.textBox1.Location = new System.Drawing.Point(708, 31);

this.textBox1.Name = "textBox1";

this.textBox1.Size = new System.Drawing.Size(65, 20);

this.textBox1.TabIndex = 5;

this.textBox1.Text = "7";

//

// textBox2

//

this.textBox2.Location = new System.Drawing.Point(708, 83);

this.textBox2.Name = "textBox2";

this.textBox2.Size = new System.Drawing.Size(65, 20);

this.textBox2.TabIndex = 6;

this.textBox2.Text = "7";

//

// dataGridView1

//

this.dataGridView1.AllowUserToAddRows = false;

this.dataGridView1.BackgroundColor = System.Drawing.SystemColors.Control;

this.dataGridView1.BorderStyle = System.Windows.Forms.BorderStyle.None;

this.dataGridView1.ColumnHeadersHeightSizeMode = System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize;

this.dataGridView1.ColumnHeadersVisible = false;

this.dataGridView1.Location = new System.Drawing.Point(571, 138);

this.dataGridView1.Name = "dataGridView1";

this.dataGridView1.RowHeadersVisible = false;

this.dataGridView1.Size = new System.Drawing.Size(244, 199);

this.dataGridView1.TabIndex = 7;

//

// CloseButton

//

this.CloseButton.Font = new System.Drawing.Font("Microsoft Sans Serif", 14F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.CloseButton.Location = new System.Drawing.Point(779, -3);

this.CloseButton.Name = "CloseButton";

this.CloseButton.Size = new System.Drawing.Size(45, 35);

this.CloseButton.TabIndex = 8;

this.CloseButton.Text = "X";

this.CloseButton.UseVisualStyleBackColor = true;

this.CloseButton.Click += new System.EventHandler(this.CloseButton\_Click);

//

// GetMatrix

//

this.GetMatrix.Location = new System.Drawing.Point(571, 109);

this.GetMatrix.Name = "GetMatrix";

this.GetMatrix.Size = new System.Drawing.Size(75, 23);

this.GetMatrix.TabIndex = 9;

this.GetMatrix.Text = "GetMatrix";

this.GetMatrix.UseVisualStyleBackColor = true;

this.GetMatrix.Click += new System.EventHandler(this.GetMatrix\_Click);

//

// DefaultValue

//

this.DefaultValue.Location = new System.Drawing.Point(652, 109);

this.DefaultValue.Name = "DefaultValue";

this.DefaultValue.Size = new System.Drawing.Size(85, 23);

this.DefaultValue.TabIndex = 10;

this.DefaultValue.Text = "DefaultValue";

this.DefaultValue.UseVisualStyleBackColor = true;

this.DefaultValue.Click += new System.EventHandler(this.DefaultValue\_Click);

//

// Draw2

//

this.Draw2.Location = new System.Drawing.Point(650, 343);

this.Draw2.Name = "Draw2";

this.Draw2.Size = new System.Drawing.Size(75, 23);

this.Draw2.TabIndex = 11;

this.Draw2.Text = "Draw№2";

this.Draw2.UseVisualStyleBackColor = true;

this.Draw2.Click += new System.EventHandler(this.Draw2\_Click);

//

// Draw3

//

this.Draw3.Location = new System.Drawing.Point(731, 343);

this.Draw3.Name = "Draw3";

this.Draw3.Size = new System.Drawing.Size(75, 23);

this.Draw3.TabIndex = 12;

this.Draw3.Text = "Draw№3";

this.Draw3.UseVisualStyleBackColor = true;

this.Draw3.Click += new System.EventHandler(this.Draw3\_Click);

//

// NewValue

//

this.NewValue.Location = new System.Drawing.Point(743, 109);

this.NewValue.Name = "NewValue";

this.NewValue.Size = new System.Drawing.Size(72, 23);

this.NewValue.TabIndex = 13;

this.NewValue.Text = "NewValue";

this.NewValue.UseVisualStyleBackColor = true;

this.NewValue.Click += new System.EventHandler(this.NewValue\_Click);

//

// textBox3

//

this.textBox3.Location = new System.Drawing.Point(571, 388);

this.textBox3.Name = "textBox3";

this.textBox3.ReadOnly = true;

this.textBox3.Size = new System.Drawing.Size(75, 20);

this.textBox3.TabIndex = 14;

//

// textBox4

//

this.textBox4.Location = new System.Drawing.Point(650, 388);

this.textBox4.Name = "textBox4";

this.textBox4.ReadOnly = true;

this.textBox4.Size = new System.Drawing.Size(75, 20);

this.textBox4.TabIndex = 15;

//

// textBox5

//

this.textBox5.Location = new System.Drawing.Point(731, 388);

this.textBox5.Name = "textBox5";

this.textBox5.ReadOnly = true;

this.textBox5.Size = new System.Drawing.Size(75, 20);

this.textBox5.TabIndex = 16;

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(612, 369);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(155, 13);

this.label3.TabIndex = 17;

this.label3.Text = "Итоговое время выполнения";

this.label3.TextAlign = System.Drawing.ContentAlignment.TopRight;

//

// SecondDefaultValue

//

this.SecondDefaultValue.Location = new System.Drawing.Point(787, 80);

this.SecondDefaultValue.Name = "SecondDefaultValue";

this.SecondDefaultValue.Size = new System.Drawing.Size(28, 23);

this.SecondDefaultValue.TabIndex = 18;

this.SecondDefaultValue.Text = "s";

this.SecondDefaultValue.UseVisualStyleBackColor = true;

this.SecondDefaultValue.Click += new System.EventHandler(this.SecondDefaultValue\_Click);

//

// Рисование

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(820, 443);

this.Controls.Add(this.SecondDefaultValue);

this.Controls.Add(this.label3);

this.Controls.Add(this.textBox5);

this.Controls.Add(this.textBox4);

this.Controls.Add(this.textBox3);

this.Controls.Add(this.NewValue);

this.Controls.Add(this.Draw3);

this.Controls.Add(this.Draw2);

this.Controls.Add(this.DefaultValue);

this.Controls.Add(this.GetMatrix);

this.Controls.Add(this.CloseButton);

this.Controls.Add(this.dataGridView1);

this.Controls.Add(this.textBox2);

this.Controls.Add(this.textBox1);

this.Controls.Add(this.label2);

this.Controls.Add(this.label1);

this.Controls.Add(this.zedGraphControl1);

this.Controls.Add(this.Draw);

this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;

this.Name = "Рисование";

this.Text = "Form1";

this.MouseDown += new System.Windows.Forms.MouseEventHandler(this.Рисование\_MouseDown);

((System.ComponentModel.ISupportInitialize)(this.dataGridView1)).EndInit();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private System.Windows.Forms.Button Draw;

private ZedGraph.ZedGraphControl zedGraphControl1;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.TextBox textBox2;

private System.Windows.Forms.DataGridView dataGridView1;

private System.Windows.Forms.Button CloseButton;

private System.Windows.Forms.Button GetMatrix;

private System.Windows.Forms.Button DefaultValue;

private System.Windows.Forms.Button Draw2;

private System.Windows.Forms.Button Draw3;

private System.Windows.Forms.Button NewValue;

private System.Windows.Forms.TextBox textBox3;

private System.Windows.Forms.TextBox textBox4;

private System.Windows.Forms.TextBox textBox5;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Button SecondDefaultValue;

}

}

Form1.resx

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

<root>

<!--

Microsoft ResX Schema

Version 2.0

The primary goals of this format is to allow a simple XML format

that is mostly human readable. The generation and parsing of the

various data types are done through the TypeConverter classes

associated with the data types.

Example:

... ado.net/XML headers & schema ...

<resheader name="resmimetype">text/microsoft-resx</resheader>

<resheader name="version">2.0</resheader>

<resheader name="reader">System.Resources.ResXResourceReader, System.Windows.Forms, ...</resheader>

<resheader name="writer">System.Resources.ResXResourceWriter, System.Windows.Forms, ...</resheader>

<data name="Name1"><value>this is my long string</value><comment>this is a comment</comment></data>

<data name="Color1" type="System.Drawing.Color, System.Drawing">Blue</data>

<data name="Bitmap1" mimetype="application/x-microsoft.net.object.binary.base64">

<value>[base64 mime encoded serialized .NET Framework object]</value>

</data>

<data name="Icon1" type="System.Drawing.Icon, System.Drawing" mimetype="application/x-microsoft.net.object.bytearray.base64">

<value>[base64 mime encoded string representing a byte array form of the .NET Framework object]</value>

<comment>This is a comment</comment>

</data>

There are any number of "resheader" rows that contain simple

name/value pairs.

Each data row contains a name, and value. The row also contains a

type or mimetype. Type corresponds to a .NET class that support

text/value conversion through the TypeConverter architecture.

Classes that don't support this are serialized and stored with the

mimetype set.

The mimetype is used for serialized objects, and tells the

ResXResourceReader how to depersist the object. This is currently not

extensible. For a given mimetype the value must be set accordingly:

Note - application/x-microsoft.net.object.binary.base64 is the format

that the ResXResourceWriter will generate, however the reader can

read any of the formats listed below.

mimetype: application/x-microsoft.net.object.binary.base64

value : The object must be serialized with

: System.Runtime.Serialization.Formatters.Binary.BinaryFormatter

: and then encoded with base64 encoding.

mimetype: application/x-microsoft.net.object.soap.base64

value : The object must be serialized with

: System.Runtime.Serialization.Formatters.Soap.SoapFormatter

: and then encoded with base64 encoding.

mimetype: application/x-microsoft.net.object.bytearray.base64

value : The object must be serialized into a byte array

: using a System.ComponentModel.TypeConverter

: and then encoded with base64 encoding.

-->

<xsd:schema id="root" xmlns="" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">

<xsd:import namespace="http://www.w3.org/XML/1998/namespace" />

<xsd:element name="root" msdata:IsDataSet="true">

<xsd:complexType>

<xsd:choice maxOccurs="unbounded">

<xsd:element name="metadata">

<xsd:complexType>

<xsd:sequence>

<xsd:element name="value" type="xsd:string" minOccurs="0" />

</xsd:sequence>

<xsd:attribute name="name" use="required" type="xsd:string" />

<xsd:attribute name="type" type="xsd:string" />

<xsd:attribute name="mimetype" type="xsd:string" />

<xsd:attribute ref="xml:space" />

</xsd:complexType>

</xsd:element>

<xsd:element name="assembly">

<xsd:complexType>

<xsd:attribute name="alias" type="xsd:string" />

<xsd:attribute name="name" type="xsd:string" />

</xsd:complexType>

</xsd:element>

<xsd:element name="data">

<xsd:complexType>

<xsd:sequence>

<xsd:element name="value" type="xsd:string" minOccurs="0" msdata:Ordinal="1" />

<xsd:element name="comment" type="xsd:string" minOccurs="0" msdata:Ordinal="2" />

</xsd:sequence>

<xsd:attribute name="name" type="xsd:string" use="required" msdata:Ordinal="1" />

<xsd:attribute name="type" type="xsd:string" msdata:Ordinal="3" />

<xsd:attribute name="mimetype" type="xsd:string" msdata:Ordinal="4" />

<xsd:attribute ref="xml:space" />

</xsd:complexType>

</xsd:element>

<xsd:element name="resheader">

<xsd:complexType>

<xsd:sequence>

<xsd:element name="value" type="xsd:string" minOccurs="0" msdata:Ordinal="1" />

</xsd:sequence>

<xsd:attribute name="name" type="xsd:string" use="required" />

</xsd:complexType>

</xsd:element>

</xsd:choice>

</xsd:complexType>

</xsd:element>

</xsd:schema>

<resheader name="resmimetype">

<value>text/microsoft-resx</value>

</resheader>

<resheader name="version">

<value>2.0</value>

</resheader>

<resheader name="reader">

<value>System.Resources.ResXResourceReader, System.Windows.Forms, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>

</resheader>

<resheader name="writer">

<value>System.Resources.ResXResourceWriter, System.Windows.Forms, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089</value>

</resheader>

</root>

Lab1.caproj

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

<Project ToolsVersion="15.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">

<Import Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props" Condition="Exists('$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props')" />

<PropertyGroup>

<Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>

<Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>

<ProjectGuid>{DD863E3D-DC7F-4434-9871-C41E94166C5A}</ProjectGuid>

<OutputType>WinExe</OutputType>

<RootNamespace>Lab1</RootNamespace>

<AssemblyName>Lab1</AssemblyName>

<TargetFrameworkVersion>v4.7.2</TargetFrameworkVersion>

<FileAlignment>512</FileAlignment>

<AutoGenerateBindingRedirects>true</AutoGenerateBindingRedirects>

<Deterministic>true</Deterministic>

</PropertyGroup>

<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">

<PlatformTarget>AnyCPU</PlatformTarget>

<DebugSymbols>true</DebugSymbols>

<DebugType>full</DebugType>

<Optimize>false</Optimize>

<OutputPath>bin\Debug\</OutputPath>

<DefineConstants>DEBUG;TRACE</DefineConstants>

<ErrorReport>prompt</ErrorReport>

<WarningLevel>4</WarningLevel>

</PropertyGroup>

<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">

<PlatformTarget>AnyCPU</PlatformTarget>

<DebugType>pdbonly</DebugType>

<Optimize>true</Optimize>

<OutputPath>bin\Release\</OutputPath>

<DefineConstants>TRACE</DefineConstants>

<ErrorReport>prompt</ErrorReport>

<WarningLevel>4</WarningLevel>

</PropertyGroup>

<ItemGroup>

<Reference Include="System" />

<Reference Include="System.Core" />

<Reference Include="System.Xml.Linq" />

<Reference Include="System.Data.DataSetExtensions" />

<Reference Include="Microsoft.CSharp" />

<Reference Include="System.Data" />

<Reference Include="System.Deployment" />

<Reference Include="System.Drawing" />

<Reference Include="System.Net.Http" />

<Reference Include="System.Windows.Forms" />

<Reference Include="System.Xml" />

<Reference Include="ZedGraph">

<HintPath>..\..\..\..\..\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\Common7\IDE\ZedGraph.dll</HintPath>

</Reference>

</ItemGroup>

<ItemGroup>

<Compile Include="Form1.cs">

<SubType>Form</SubType>

</Compile>

<Compile Include="Form1.Designer.cs">

<DependentUpon>Form1.cs</DependentUpon>

</Compile>

<Compile Include="Program.cs" />

<Compile Include="Properties\AssemblyInfo.cs" />

<Compile Include="Win32.cs" />

<EmbeddedResource Include="Form1.resx">

<DependentUpon>Form1.cs</DependentUpon>

</EmbeddedResource>

<EmbeddedResource Include="Properties\Resources.resx">

<Generator>ResXFileCodeGenerator</Generator>

<LastGenOutput>Resources.Designer.cs</LastGenOutput>

<SubType>Designer</SubType>

</EmbeddedResource>

<Compile Include="Properties\Resources.Designer.cs">

<AutoGen>True</AutoGen>

<DependentUpon>Resources.resx</DependentUpon>

</Compile>

<None Include="Properties\Settings.settings">

<Generator>SettingsSingleFileGenerator</Generator>

<LastGenOutput>Settings.Designer.cs</LastGenOutput>

</None>

<Compile Include="Properties\Settings.Designer.cs">

<AutoGen>True</AutoGen>

<DependentUpon>Settings.settings</DependentUpon>

<DesignTimeSharedInput>True</DesignTimeSharedInput>

</Compile>

</ItemGroup>

<ItemGroup>

<None Include="App.config" />

</ItemGroup>

<Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />

</Project>

Lab1.sln

Microsoft Visual Studio Solution File, Format Version 12.00

# Visual Studio Version 16

VisualStudioVersion = 16.0.30413.136

MinimumVisualStudioVersion = 10.0.40219.1

Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "Lab1", "Lab1.csproj", "{DD863E3D-DC7F-4434-9871-C41E94166C5A}"

EndProject

Global

GlobalSection(SolutionConfigurationPlatforms) = preSolution

Debug|Any CPU = Debug|Any CPU

Release|Any CPU = Release|Any CPU

EndGlobalSection

GlobalSection(ProjectConfigurationPlatforms) = postSolution

{DD863E3D-DC7F-4434-9871-C41E94166C5A}.Debug|Any CPU.ActiveCfg = Debug|Any CPU

{DD863E3D-DC7F-4434-9871-C41E94166C5A}.Debug|Any CPU.Build.0 = Debug|Any CPU

{DD863E3D-DC7F-4434-9871-C41E94166C5A}.Release|Any CPU.ActiveCfg = Release|Any CPU

{DD863E3D-DC7F-4434-9871-C41E94166C5A}.Release|Any CPU.Build.0 = Release|Any CPU

EndGlobalSection

GlobalSection(SolutionProperties) = preSolution

HideSolutionNode = FALSE

EndGlobalSection

GlobalSection(ExtensibilityGlobals) = postSolution

SolutionGuid = {DF330728-1DFE-427C-82EE-A69BC37B2618}

EndGlobalSection

EndGlobal

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab1

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Рисование());

}

}

}

Win32.cs

using System;

using System.Runtime.InteropServices;

namespace Lab1

{

class Win32

{

public const int WM\_NCLBUTTONDOWN = 0xA1;

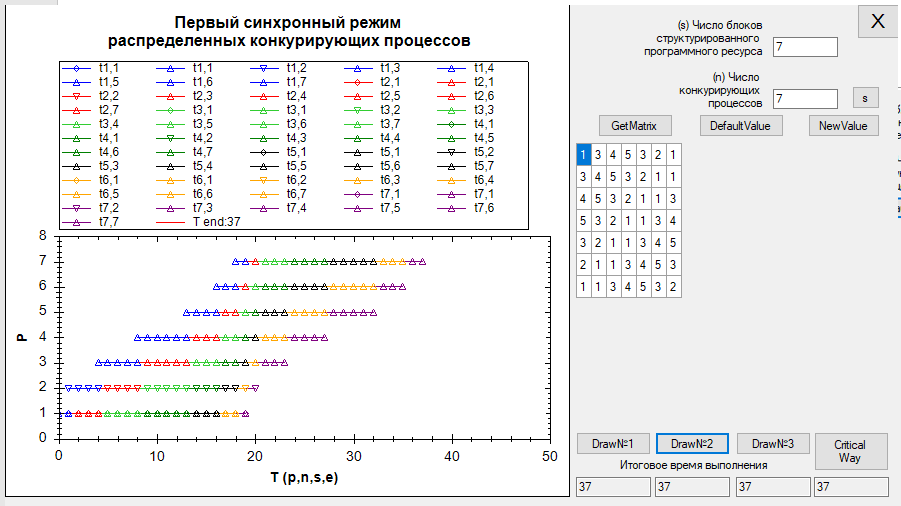
public const int HTCAPTION = 0x2;

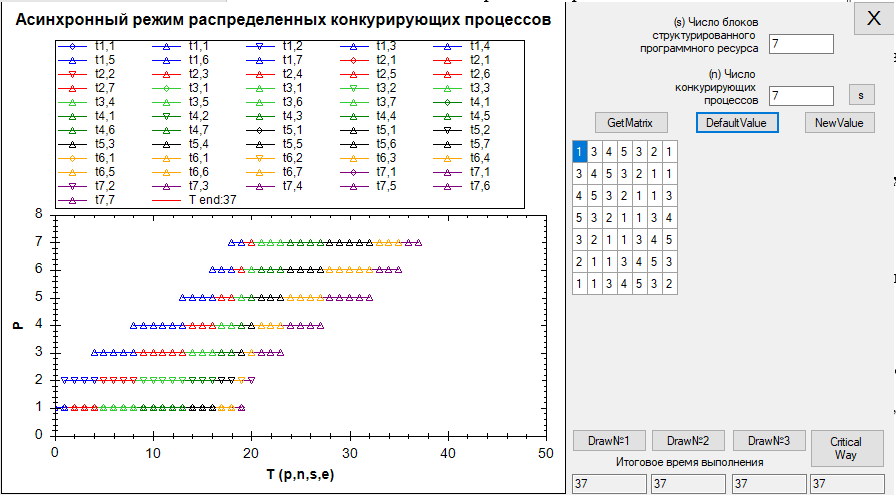
[DllImportAttribute("user32.dll")]

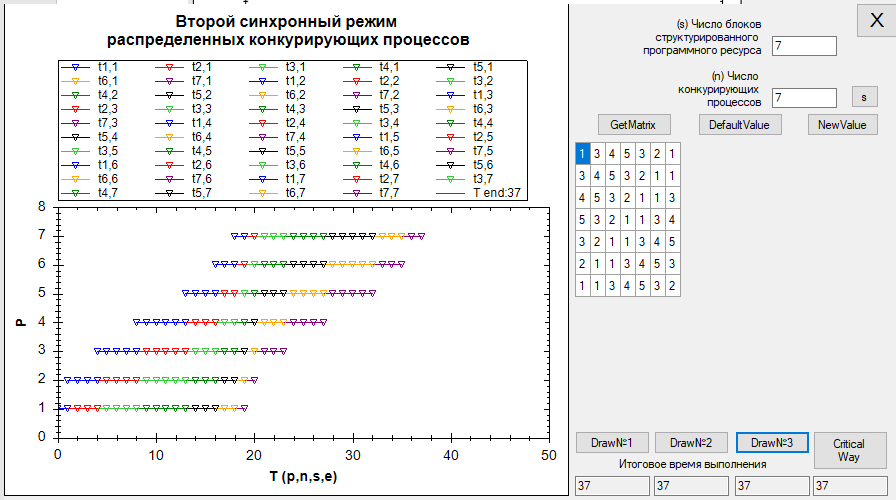
public static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);

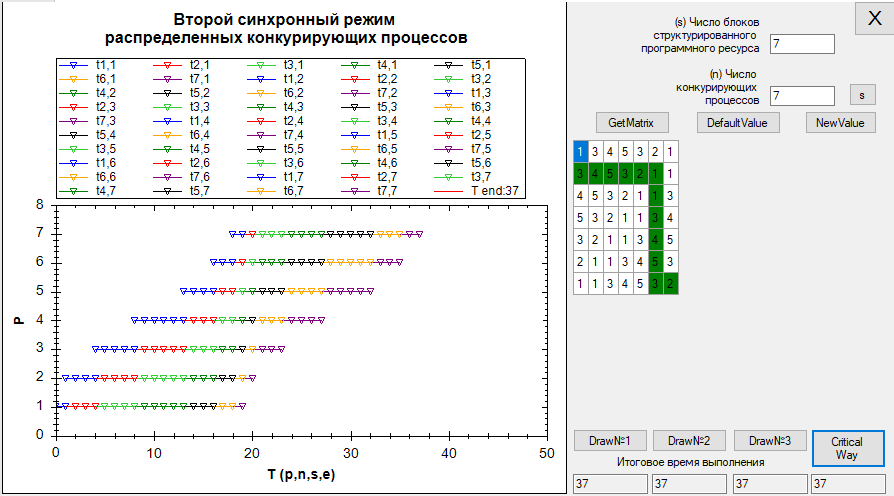
}

}









Задача

Пусть — последовательность времен выполнения блоков линейно структурированного программного ресурса , n = 6 - число конкурирующих процессов, р = 3 — число однородных процессоров,  = 4 — дополнительные системные накладные расходы на каждый блок, связанные со структурированием .

а) Построить оптимальную компоновку.

б) Провести сравнение числа блоков полученной оптимальной компоновки с теоретическим числом блоков согласно критерия оптимальности структурирования.

в) Посчитать коэффициент эффективности оптимальной компоновки по отношению к исходному структурированию в процентах.

Пример реализации программы:

