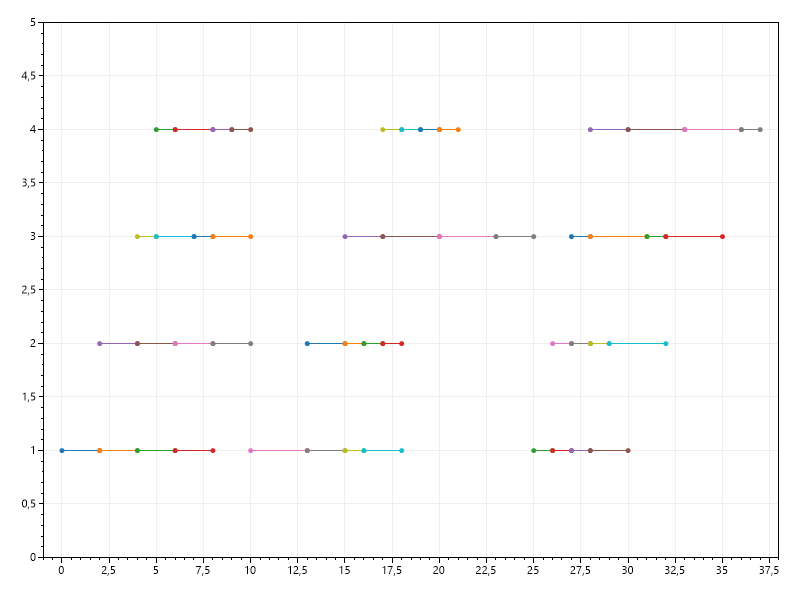
# Лабораторная работа 5

Программа находит величину выполнения заданного процесса в многопоточном режиме и строит соответствующую диаграмму.



На графике представлено время выполнения процессов (нижняя ось, в миллисекундах) и потоки, на которых эти процессы выполняются (ось слева, номер потока от 1 до 4).

**Program.cs**

const int matrixHeight = 4;

const int matrixWidth = 12;

const int matrixMinValue = 1;

const int matrixMaxValue = 4;

Solve();

static void Solve()

{

int[] processesIndexes = {1, 2, 3, 4};

var plot = new Plot();

var matrix = new int[matrixWidth, matrixHeight];

CreateMatrix(matrixWidth, matrixHeight, matrix);

Draw(matrixWidth, matrixHeight, plot, processesIndexes, matrix);

plot.SetAxisLimitsY(0, processesIndexes.Length + 1);

plot.SaveFig("../../../../Diagrams/FirstSynchronousMethod.png");

}

static void Draw(int width, int height, Plot plot, int[] processesIndexes, int[,] matrix)

{

var count = 0;

var begin = 0;

var newBegin = 0;

var max = 0;

for (var i = 0; i < width; i++)

{

if (count % processesIndexes.Length == 0)

{

count = 0;

newBegin = begin;

}

if (count == 0)

{

newBegin = max;

}

else

{

newBegin += matrix[i - 1, 0];

}

begin = newBegin;

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

{

begin = AddSegment(begin, begin + matrix[i, j], plot, processesIndexes[count], begin, matrix[i, j]);

if (max < begin)

{

max = begin;

}

}

count++;

}

}

static int AddSegment(int a, int b, Plot plot, double process, int begin, int c)

{

var x = new double[] {a, b};

var y = new[] {process, process};

plot.AddScatter(x, y);

begin += c;

return begin;

}

static void CreateMatrix(int width, int height, int[,] matrix)

{

var random = new Random();

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

{

for (var j = 0; j < width; j++)

{

matrix[j, i] = random.Next(matrixMinValue, matrixMaxValue); // рандомим по очереди каждый элемент матрицы

}

}

}