МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М.Є. Жуковського
“Харківський авіаційний інститут”

Кафедра комп’ютерних систем та мереж

Лабораторна робота № 5

“Разработка и исследование программы построения графиков”

По дисциплине “Технологии программирования”

ХАІ.503.525В.123. 1705067.180

Виконав студент гр. 525В Пеналоза Г.
(№ групи) (П.І.Б.)

15-03-19
(підпис, дата)

Перевірив ст. викладач каф. 503
(науковий ступінь, вчене звання, посада)

(підпис, дата) (П.І.Б.)

2019

Задание:

Разработать программу для построения на форме:

* гистограммы (столбиковой вертикальной и горизонтальной диаграмм);
* круговой диаграммы.

Диаграммы нарисовать с помощью примитивов в соответствии с вариантом для N значений (N=1..15).

Проект должен содержать три формы:

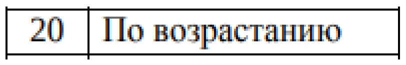
* 1-я форма – для задания и редактирования значений и параметров диаграммы и выбора вида диаграммы,

* 2-я форма – для визуализации диаграммы (в работающем приложении 2-я форма должна быть развернута на весь экран),

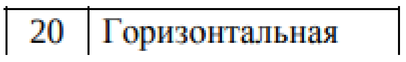
* 3-я форма – для сведений об авторе проекта (модальная форма, которая может вызываться из первых двух форм).

Разработать модульные тесты (unit-тесты) для методов класса.
UML диаграмма вариантов использования проекта приведена на рисунке 1.

Вариант 20



20	По возрастанию



20	Горизонтальная

Текст программы

Form.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;
```

```
namespace Lab2._2._5
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        public Diagrama diagrama = new Diagrama();//Instance of Graph window

        private void button3_Click(object sender, EventArgs e)//Show about autor
        {
            About about = new About();
            about.ShowDialog();
        }

        private void button2_Click(object sender, EventArgs e)//Show Graphs
        {
            diagrama.Refresh();
            diagrama.Show();
            diagrama.Focus();
        }

        private void numericUpDown2_ValueChanged(object sender, EventArgs e)//
Check min
        {
            if (numericUpDown2.Value > numericUpDown3.Value)
            {
                MessageBox.Show("This number can be higher than Max!");
                numericUpDown2.Value = numericUpDown3.Value - 1;
            }
            else
            {
                if (numericUpDown3.Value == 0 || numericUpDown3.Value <=
numericUpDown2.Value)
                {
                    numericUpDown3.Value = numericUpDown2.Value + 1;
                }
            }
        }

        private void numericUpDown3_ValueChanged(object sender, EventArgs e)//
Check max
        {
            if (numericUpDown3.Value <= numericUpDown2.Value)
            {
                MessageBox.Show("This number can be less than Min!");
                numericUpDown3.Value = numericUpDown2.Value + 1;
            }
        }

        private void button1_Click(object sender, EventArgs e)//Generate random
numbers according to the type of graph.
        {
            Metodos.GenerarArrayNumeros((int)numericUpDown1.Value,
(int)numericUpDown3.Value, (int)numericUpDown2.Value);
            FillData();
            button2.Enabled = true;
            button4.Enabled = true;
        }

        public void FillData()//Fill the values into the dataview
        {
            dataGridView1.Rows.Clear();
            dataGridView1.Columns.Clear();
            switch (Metodos.diagrama)
            {
                case 0:
                case 1:
                case 2:
                    dataGridView1.Columns.Add("Значение", "Значение");
                    for (int c = 0; c < (int)numericUpDown1.Value; c++)
                    {
                        dataGridView1.Rows.Add();
                        dataGridView1.Rows[c].Cells[0].Value =
Metodos._arrayNumbers012[c];
                    }
                    break;
                case 3:
                    dataGridView1.Columns.Add("Значение Graph 1", "Значение Graph
1");
                    dataGridView1.Columns.Add("Значение Graph 2", "Значение Graph
2");
                    for (int c = 0; c < (int)numericUpDown1.Value; c++)
                    {
                        dataGridView1.Rows.Add();
                        dataGridView1.Rows[c].Cells[0].Value =
Metodos._arrayNumbers3[c, 0];
```

```
                    dataGridView1.Rows[c].Cells[1].Value =
Metodos._arrayNumbers3[c, 1];
                }
                break;
            case 4:
                dataGridView1.Columns.Add("Значение Graph 1 - serie 1",
"Значение Graph 1  - serie 1");
                dataGridView1.Columns.Add("Значение Graph 1 - serie 2",
"Значение Graph 1  - serie 2");
                dataGridView1.Columns.Add("Значение Graph 2 - serie 1",
"Значение Graph 2  - serie 1");
                dataGridView1.Columns.Add("Значение Graph 2 - serie 2",
"Значение Graph 2  - serie 2");
                for (int c = 0; c < (int)numericUpDown1.Value; c++)
                {
                    dataGridView1.Rows.Add();
                    dataGridView1.Rows[c].Cells[0].Value =
Metodos._arrayNumbers4[c, 0];
                    dataGridView1.Rows[c].Cells[1].Value =
Metodos._arrayNumbers4[c, 1];
                    dataGridView1.Rows[c].Cells[2].Value =
Metodos._arrayNumbers4[c, 2];
                    dataGridView1.Rows[c].Cells[3].Value =
Metodos._arrayNumbers4[c, 3];
                }
                break;
        }
    }

    private void Form1_Load(object sender, EventArgs e)//Load the necessary
values when the program start
    {
        dataGridView1.AutoSizeColumnsMode =
DataGridViewAutoSizeColumnsMode.Fill;
        dataGridView1.Rows.Clear();
        Metodos.color = Color.Black;
        comboBox1.SelectedIndex = 2;
        button5.ForeColor = Color.Black;
        Metodos.diagrama = 2;
    }

    private void checkBox1_CheckedChanged(object sender, EventArgs
e)//Select the color for the graph
    {
    }

    private void button4_Click(object sender, EventArgs e)//Button to sort
    {
        try
        {
            Metodos.Sort();
            FillData();
        }
        catch { MessageBox.Show("Error while sorting the array, try to generate
again!"); return; }
    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs
e)//Select the graph type
    {
        Metodos.diagrama = comboBox1.SelectedIndex;
        button1_Click(sender,e);//Call to the generate numbers button
    }

    private void button5_Click(object sender, EventArgs e)//Select the color for
the graph
    {
        if (colorDialog1.ShowDialog() == DialogResult.OK)
        {
            button5.ForeColor = colorDialog1.Color;Metodos.color =
colorDialog1.Color;
        }
    }

    private void numericUpDown1_ValueChanged(object sender, EventArgs e)
    {
        button1_Click(sender, e);
    }
}
}
```

About.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;

namespace Lab2._2._5
{
   public partial class About : Form
   {
      public About()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
         this.Close();
      }
   }
}
```

Diagrama.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;

namespace Lab2._2._5
{
   public partial class Diagrama : Form
   {
      public Diagrama()
      {
         InitializeComponent();
      }
```

```
        private void Diagrama_Load(object sender, EventArgs e)
        {
            if (Metodos.diagrama==0||Metodos.diagrama==1)
            {
                FillArrayPrimitive();
            }
        }

        private Panel panel=new Panel();//Graph
        private Panel title=new Panel();//Title
        private string m_name="";//Title for the primitives
        public Graphics gr;//Graphs primitives
        private Metodos.Item[] arr;//Array with values of the primitive chart

            private void Gtitle(object sender, System.Windows.Forms.PaintEventArgs
e)//Draw tittle 0-1
        {
            Graphics gtitle = e.Graphics;
            gtitle.SmoothingMode = SmoothingMode.HighQuality;
            Font font = new Font("Times New Roman", 15f);
            SizeF sizeF = gtitle.MeasureString(m_name, font);
                        gtitle.DrawString(m_name, font, Brushes.Blue, title.Location.X,
title.Location.Y);
        }

         private void Ggraphs(object sender, System.Windows.Forms.PaintEventArgs
e)//Draw Graphs
        {
            FillArrayPrimitive();
            gr = e.Graphics;
            switch (Metodos.diagrama)
            {
                case 0:
                    Description(true);
                    Circle();
                    break;
                case 1:
                    Description(false);
                    HorizontalScale();
                    BuildHorizontalChart();
                    break;
            }

        }
        public void FillArrayPrimitive()//Fill the array with primitive chart
        {
            arr = new Metodos.Item[Metodos._arrayNumbers012.Length];
            for (int i = 0; i < arr.Length; i++)
            {
                arr[i].value = Metodos._arrayNumbers012[i];
                arr[i].color = Metodos.GetColor(i);
                arr[i].name = "Val ";
                                        arr[i].name += (i + 1).ToString() + " = " +
Metodos._arrayNumbers012[i].ToString();
            }
        }

            private void Diagrama_Activated(object sender, EventArgs e)//Draw with
chart or primitive
        {
            chart1.BackGradientStyle = GradientStyle.DiagonalLeft;
            switch (Metodos.diagrama)
            {
                case 0:

                    //FillArrayPrimitive();
                    this.Refresh();

                    chart1.Visible = false;
                    label1.Visible = false;

                    //Draw the круговая
                    panel.Dock = DockStyle.Fill;
                    panel.BackColor = Color.LightGray;
                    panel.Paint += new PaintEventHandler(this.Ggraphs);
                    tableLayoutPanel1.Controls.Add(panel,0,0);

                    //Write the tittle
                    m_name = "круговая - примитивов Visual C#";
                    title.Dock = DockStyle.Fill;
                    title.BackColor = Color.LightGray;
                    title.Paint += new PaintEventHandler(this.Gtitle);
                    tableLayoutPanel2.Controls.Add(title, 0, 0);
                    break;
                case 1:
                    //FillArrayPrimitive();
                    this.Refresh();
                    chart1.Visible = false;
                    label1.Visible = false;

                    //Draw the линейная BAR
                    panel.Dock = DockStyle.Fill;
                    panel.BackColor = Color.LightGray;
                    panel.Paint += new PaintEventHandler(this.Ggraphs);
                    tableLayoutPanel1.Controls.Add(panel, 0, 0);

                    //Write the tittle
                    m_name = "линейная - примитивов Visual C#";
                    title.Dock = DockStyle.Fill;
                    title.BackColor = Color.LightGray;
                    title.Paint += new PaintEventHandler(this.Gtitle);
                    tableLayoutPanel2.Controls.Add(title, 0, 0);
                    break;
                case 2:
                    if (tableLayoutPanel1.Controls.Count==3)
                    {
                        tableLayoutPanel1.Controls.Remove(panel);
                        tableLayoutPanel2.Controls.Remove(title);
                    }
                    chart1.Visible = true;
                    label1.Visible = true;
                    chart1.ChartAreas.Clear();
                    chart1.ChartAreas.Add("Горизонтальная линейная - TChart");
                    chart1.ChartAreas[0].AxisX.Title = "Ось X";
                    chart1.ChartAreas[0].AxisY.Title = "Ось Y";
                    chart1.Series.Clear();
                    chart1.Series.Add("Горизонтальная линейная - TChart");
                    chart1.Series[0].ChartType = SeriesChartType.Bar;
                    label1.Text = "Горизонтальная линейная - TChart";
                    chart1.Series[0].Color = Metodos.color;
                    foreach (var t in Metodos._arrayNumbers012)
                        chart1.Series[0].Points.Add(t);
                    break;
                case 3:
                    if (tableLayoutPanel1.Controls.Count == 3)
                    {
                        tableLayoutPanel1.Controls.Remove(panel);
                        tableLayoutPanel2.Controls.Remove(title);
                    }
                    chart1.Visible = true;
                    label1.Visible = true;
                    chart1.ChartAreas.Clear();
                    chart1.ChartAreas.Add("2 горизонтальная линейная - TChart");
                    chart1.ChartAreas[0].AxisX.Title = "Ось X";
                    chart1.ChartAreas[0].AxisY.Title = "Ось Y";
                    chart1.Series.Clear();
                    chart1.Series.Add("1 горизонтальная линейная - TChart");
                    chart1.Series.Add("2 горизонтальная линейная - TChart");
                    chart1.Series[0].ChartType= SeriesChartType.Bar;
                    chart1.Series[0].Color = Metodos.color;
                    chart1.Series[1].ChartType = SeriesChartType.Bar;
                    label1.Text = "2 горизонтальная линейная - TChart";
                    for (int c=0;c<Metodos._arrayNumbers3.GetLength(0);c++)
                    {
                       chart1.Series[0].Points.Add(Metodos._arrayNumbers3[c,0]);
                       chart1.Series[1].Points.Add(Metodos._arrayNumbers3[c,1]);
                    }
                    break;
                 case 4:
                    if (tableLayoutPanel1.Controls.Count == 3)
                    {
                       tableLayoutPanel1.Controls.Remove(panel);
                       tableLayoutPanel2.Controls.Remove(title);
                    }
                    chart1.Visible = true;
                    label1.Visible = true;
                    chart1.ChartAreas.Clear();
                    chart1.Series.Clear();
                    chart1.ChartAreas.Add("Chart_Area_1");
                    chart1.ChartAreas.Add("Chart_Area_2");
                    chart1.ChartAreas[0].AxisX.Title = "Ось X";
                    chart1.ChartAreas[0].AxisY.Title = "Ось Y";
                    chart1.ChartAreas[1].AxisX.Title = "Ось X";
                    chart1.ChartAreas[1].AxisY.Title = "Ось Y";
```

chart1.Series.Add("1 горизонтальная линейная - 1 TChart"); chart1.Series[0].ChartType = SeriesChartType.Bar; chart1.Series[0].ChartArea = "Chart_Area_1";

chart1.Series.Add("2 горизонтальная линейная - 1 TChart"); chart1.Series[1].ChartType = SeriesChartType.Bar; chart1.Series[1].ChartArea = "Chart_Area_1";

chart1.Series.Add("1 горизонтальная линейная - 2 TChart"); chart1.Series[2].ChartType = SeriesChartType.Bar; chart1.Series[2].ChartArea = "Chart_Area_2";

chart1.Series.Add("2 горизонтальная линейная - 2 TChart"); chart1.Series[3].ChartType = SeriesChartType.Bar; chart1.Series[3].ChartArea = "Chart_Area_2";

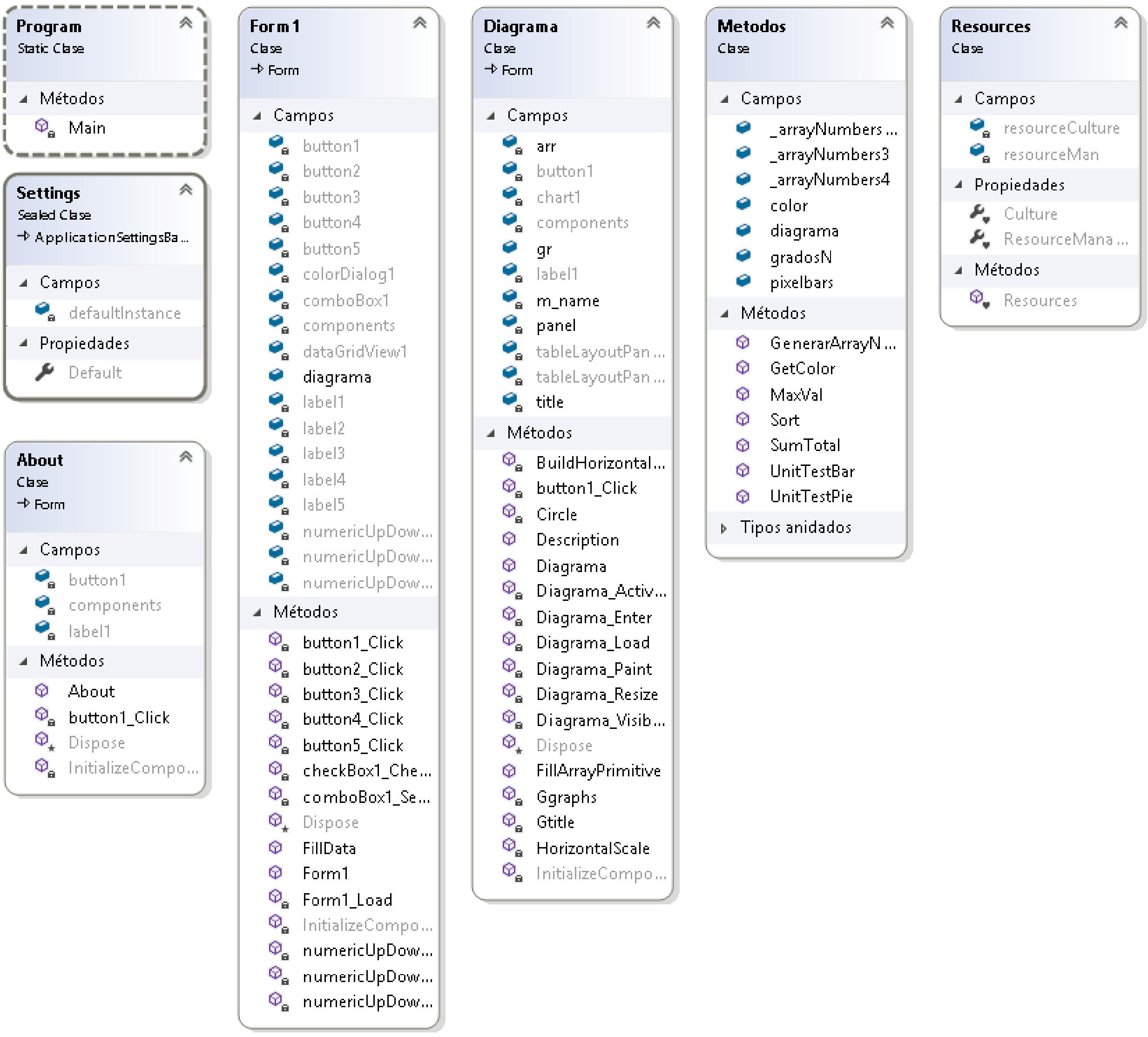
```
            for (int i=0;i<Metodos._arrayNumbers4.GetLength(0);i++)
            {
                chart1.Series[0].Points.Add(Metodos._arrayNumbers4[i, 0]);
                chart1.Series[1].Points.Add(Metodos._arrayNumbers4[i, 1]);
                chart1.Series[2].Points.Add(Metodos._arrayNumbers4[i, 2]);
                chart1.Series[3].Points.Add(Metodos._arrayNumbers4[i, 3]);
            }

            label1.Text = "2 горизонтальная линейная и две графы - TChart";
            break;
    }
}

private void button1_Click(object sender, EventArgs e)
{
    this.Visible = false;
}

private void HorizontalScale()//Draw scale for the horizontal bars
{
    int max = Metodos.MaxVal();
    Pen pen = new Pen(Color.Black, 2);
    float y = 60;
    float x1 = 20, x2 = panel.Width - 200;
    gr.DrawLine(pen, new PointF(x1, y), new PointF(x2, y));
    float interval = (x2 - x1) / 10;
    Font font = new Font("Arial", 12);
    SolidBrush brush = new SolidBrush(Color.Black);
    for (int i = 0; i < 11; i++)
    {
        float tmpX = x2 - i * interval;
        gr.DrawLine(pen, new PointF(tmpX, y - 5), new PointF(tmpX, y + 5));
        string str = ((double)(max * (10 - i)) / 10).ToString();
        gr.DrawString(str, font, brush, new PointF(tmpX - 10, 37));
    }
}
private void BuildHorizontalChart()//Draw horizontal bars
{
    int startY = 70;
    int finishY = panel.Height - 20;
    int addY = (finishY - startY) / (int)Metodos._arrayNumbers012.Length;
    int leftX = 20;
    int rightX = panel.Width - 200;
    int maxW = (rightX - leftX);
    int curH = addY * 3 / 4;
```

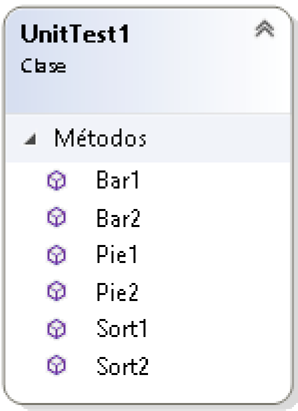
```
            int curW;
            Pen pen = new Pen(Color.Black, 3);
            for (int i = 0; i < Metodos._arrayNumbers012.Length; i++)
            {
                curW = maxW * arr[i].value / Metodos.MaxVal();
                Rectangle rect = new Rectangle(new Point(leftX, startY + i * addY),
                    new Size(curW, curH));
                gr.FillRectangle(new SolidBrush(arr[i].color), rect);
                gr.DrawRectangle(pen, rect);
            }
                gr.DrawLine(pen, new Point(leftX, startY), new Point(leftX, finishY -
addY));
        }
        private void Circle()//Draw Pie diagram
        {
            Pen pen = new Pen(Color.Red, 1);
            int y = 100;
            int a = panel.Height - 50 - y;
            int x = (panel.Width - a) / 2 - 100;
            Point pt = new Point(x, y);
            Size sz = new Size(a, a);
            Rectangle rect = new Rectangle(pt, sz);
            float curAngle = 0;
            for (int i = 0; i < Metodos._arrayNumbers012.Length; i++)
            {
                float swAngle = 360.0F * arr[i].value / Metodos.SumTotal();
                gr.FillPie(new SolidBrush(arr[i].color), rect, curAngle, swAngle);
                gr.DrawPie(pen, rect, curAngle, swAngle);
                curAngle += swAngle;
            }
        }
        public void Description(bool isRound)//Show the description of graphs
        {

            int x = Width - 175;
            int curY = 50;
            Pen pen = new Pen(Color.Black, 1);
            Font font = new Font("Times New Roman", 12);
            string str;
            for (int i = 0; i < Metodos._arrayNumbers012.Length; i++)
            {
                Rectangle rect = new Rectangle(x, curY, 10, 10);
                SolidBrush brush = new SolidBrush(arr[i].color);
                gr.FillRectangle(brush, rect);
                gr.DrawRectangle(pen, rect);
            str = arr[i].name;
            float perCent = 0;
            if (isRound)
            {
               perCent = (arr[i].value * 360 )/ Metodos.SumTotal();
               if (perCent == 0)
                  perCent = 1;
               str += " (" + perCent.ToString() + "°)";

            }
            gr.DrawString(str, font, brush, new PointF(x + 20, curY - 4));
            curY += 40;
         }
      }

      private void Diagrama_Resize(object sender, EventArgs e)//Re draw
      {
         this.Refresh();
      }

      private void Diagrama_Enter(object sender, EventArgs e)
      {
         if (Metodos.diagrama == 0 || Metodos.diagrama == 1)
         {
            FillArrayPrimitive();
         }
      }

      private void Diagrama_Paint(object sender, PaintEventArgs e)
      {
         if (Metodos.diagrama == 0 || Metodos.diagrama == 1)
         {
            FillArrayPrimitive();
         }
      }

      private void Diagrama_VisibleChanged(object sender, EventArgs e)
      {
         if (Metodos.diagrama == 0 || Metodos.diagrama == 1)
         {
            FillArrayPrimitive();
         }
      }
   }
}
```

Metodos.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Drawing;

namespace Lab2._2._5
{
    public class Metodos
    {
        public static float[] gradosN;//Unit test grades
        public static float[] pixelbars;//Unit test pixels
        public static int [] _arrayNumbers012;//Array to 0,1,2 graph one-dimensional
        public static int[,] _arrayNumbers3;//Array to 3 graph two-dimensional
        public static int[,] _arrayNumbers4;//Array to 4 graph 4-dimensional
        public static int diagrama = 0;//Type of the graph
        public static Color color;//Color for the graphs
        public struct Item//The structure that stores the properties of the chart element
        {
            public int value;
            public Color color;
            public string name;
        }
        public static Color GetColor(int n)//Get color to chart
        {
            switch (n)
            {
                case 0: return Color.DarkRed;
                case 1: return Color.Green;
                case 2: return Color.Blue;
                case 3: return Color.Orange;
                case 4: return Color.Cyan;
                case 5: return Color.Magenta;
                case 6: return color;
                case 7: return Color.Thistle;
                case 8: return Color.SteelBlue;
                case 9: return Color.DarkKhaki;
                case 10: return Color.DarkRed;
                case 11: return Color.Yellow;
                case 12: return Color.Purple;
                case 13: return Color.DarkBlue;
            case 14: return Color.Red;
        }
        return color;
    }
    public static void Sort()//Sort the array according the graph to show
    {
        switch (diagrama)
        {
            case 0:
            case 1:
            case 2:
                for (int i=0;i<_arrayNumbers012.Length-1;i++)
                {
                    for (int j=i+1;j>0;j--)
                    {
                        if (_arrayNumbers012[j-1]>_arrayNumbers012[j])
                        {
                            int temp = _arrayNumbers012[j - 1];
                            _arrayNumbers012[j - 1] = _arrayNumbers012[j];
                            _arrayNumbers012[j] = temp;
                        }
                    }
                }
                break;
            case 3:
                for (int i = 0; i < _arrayNumbers3.GetLength(0) - 1; i++)
                {
                    for (int j = i + 1; j > 0; j--)
                    {
                        if (_arrayNumbers3[j - 1,0] > _arrayNumbers3[j,0])
                        {
                            int temp = _arrayNumbers3[j - 1,0];
                            _arrayNumbers3[j - 1,0] = _arrayNumbers3[j,0];
                            _arrayNumbers3[j,0] = temp;
                        }
                        if (_arrayNumbers3[j - 1, 1] > _arrayNumbers3[j, 1])
                        {
                            int temp = _arrayNumbers3[j - 1, 1];
                            _arrayNumbers3[j - 1, 1] = _arrayNumbers3[j, 1];
                            _arrayNumbers3[j, 1] = temp;
                        }
                    }
                }
                break;
            case 4:
                for (int i = 0; i < _arrayNumbers4.GetLength(0) - 1; i++)
                {
                    for (int j = i + 1; j > 0; j--)
                    {
                        if (_arrayNumbers4[j - 1, 0] > _arrayNumbers4[j, 0])
                        {
                            int temp = _arrayNumbers4[j - 1, 0];
                            _arrayNumbers4[j - 1, 0] = _arrayNumbers4[j, 0];
                            _arrayNumbers4[j, 0] = temp;
                        }
                        if (_arrayNumbers4[j - 1, 1] > _arrayNumbers4[j, 1])
                        {
                            int temp = _arrayNumbers4[j - 1, 1];
                            _arrayNumbers4[j - 1, 1] = _arrayNumbers4[j, 1];
                            _arrayNumbers4[j, 1] = temp;
                        }
                        if (_arrayNumbers4[j - 1, 2] > _arrayNumbers4[j, 2])
                        {
                            int temp = _arrayNumbers4[j - 1, 2];
                            _arrayNumbers4[j - 1, 2] = _arrayNumbers4[j, 2];
                            _arrayNumbers4[j, 2] = temp;
                        }
                        if (_arrayNumbers4[j - 1, 3] > _arrayNumbers4[j, 3])
                        {
                            int temp = _arrayNumbers4[j - 1, 3];
                            _arrayNumbers4[j - 1, 3] = _arrayNumbers4[j, 3];
                            _arrayNumbers4[j, 3] = temp;
                        }
                    }
                }
                break;
            }
        }
         public static void GenerarArrayNumeros(int _n, int max, int min)//Fill array
according the graph
        {
            if (diagrama==0|| diagrama == 1|| diagrama == 2)
            {
                _arrayNumbers012 = new int[_n];
                Random rm = new Random();
                for (int c = 0; c < _n; c++)
                {
                    _arrayNumbers012[c] = rm.Next(min, max);
                }
            }
        else if (diagrama==3)
        {
            _arrayNumbers3 = new int[_n,2];
            Random rm = new Random();
            for (int c = 0; c < _n; c++)
            {
                _arrayNumbers3[c, 0] = rm.Next(min, max);
                _arrayNumbers3[c, 1] = rm.Next(min, max);
            }
        }
        else
        {
            _arrayNumbers4 = new int[_n, 4];
            Random rm = new Random();
            for (int c = 0; c < _n; c++)
            {
                _arrayNumbers4[c, 0] = rm.Next(min, max);
                _arrayNumbers4[c, 1] = rm.Next(min, max);
                _arrayNumbers4[c, 2] = rm.Next(min, max);
                _arrayNumbers4[c, 3] = rm.Next(min, max);
            }
        }
    }
    public static void UnitTestPie()//Logic Pie
    {
        gradosN = new float[_arrayNumbers012.Length];
        for (int i = 0; i < Metodos._arrayNumbers012.Length; i++)
        {
            gradosN[i] = (360 * _arrayNumbers012[i]) / SumTotal();
        }
    }
    public static void UnitTestBar()//Logit Bars
    {
        int x = 691, y = 404;
        int startY = 70;
        int finishY = y - 20;
        int addY = (finishY - startY) / (int)Metodos._arrayNumbers012.Length;
        int leftX = 20;
        int rightX = x- 200;
        int maxW = (rightX - leftX);
        int curH = addY * 3 / 4;
        int curW;
        pixelbars = new float[_arrayNumbers012.Length];
        for (int i=0;i<_arrayNumbers012.Length;i++)
        {
            curW = maxW * _arrayNumbers012[i] / MaxVal();
            pixelbars[i] = curH * curW;
         }
      }
      public static int SumTotal()//Get sum of the array values
      {
         int sum = 0;
         for (int i = 0; i < _arrayNumbers012.Length; i++)
            sum += _arrayNumbers012[i];
         return sum;
      }
      public static int MaxVal()//Get max val of the array
      {
         int max = 0;
         for (int i = 0; i < _arrayNumbers012.Length; i++)
         {
            if (_arrayNumbers012[i] > max)
            {
               max = _arrayNumbers012[i];
            }
         }
         return max;
      }
   }
}
```

Диаграмму классов

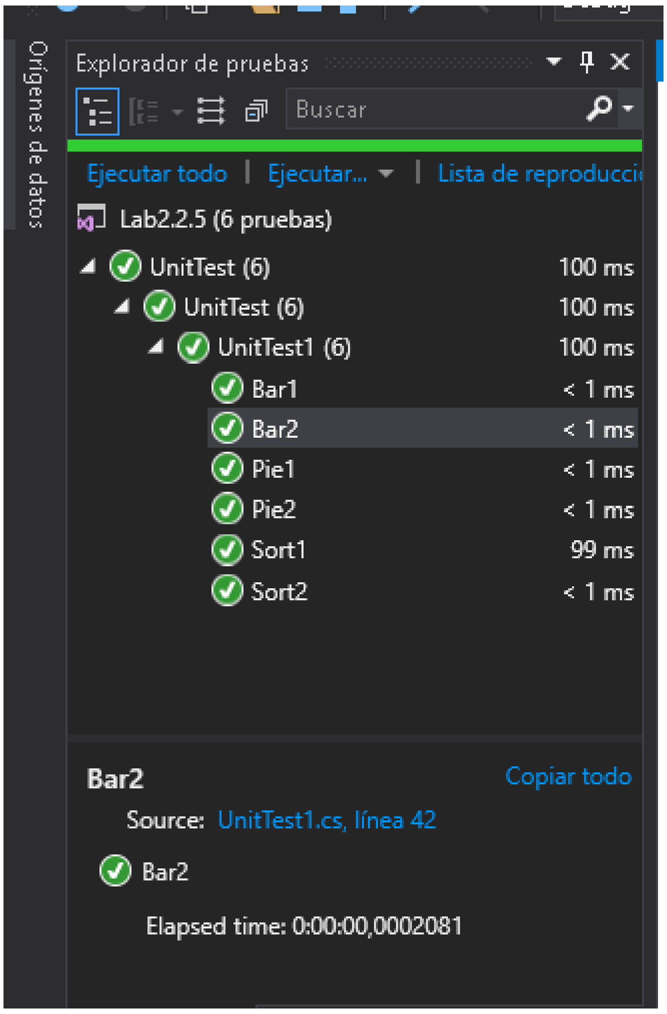




UnitTests

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using Lab2._2._5;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Windows.Forms;

namespace UnitTest
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Sort1()
        {
            int[] arr = new int[] {1,3,2,7,5 };
            int[] arrEX = new int[] {1,2,3,5,7 };
            Metodos._arrayNumbers012 = (int[])arr.Clone();
            Metodos.Sort();
            CollectionAssert.AreEqual(arrEX,Metodos._arrayNumbers012);
        }
        [TestMethod]
        public void Sort2()
        {
            int[] arr = new int[] { 5, 11, 2, 0, 5 };
            int[] arrEX = new int[] { 0,2,5,5,11 };
            Metodos._arrayNumbers012 = (int[])arr.Clone();
            Metodos.Sort();
            CollectionAssert.AreEqual(arrEX, Metodos._arrayNumbers012);
        }

        [TestMethod]
        public void Bar1()
        {
            int[] arr = new int[] { 20, 30, 40, 10 };
            float[] arrGRADOSE = new float[] { 13630,20474,27318,6786 };
            Metodos._arrayNumbers012 = (int[])arr.Clone();
            Metodos.UnitTestBar();
            CollectionAssert.AreEqual(arrGRADOSE, Metodos.pixelbars);
        }
        [TestMethod]
        public void Bar2()
        {
            int[] arr = new int[] { 2, 3, 4, 10 };
            float[] arrGRADOSE = new float[] { 5452, 8178, 10904, 27318 };
            Metodos._arrayNumbers012 = (int[])arr.Clone();
            Metodos.UnitTestBar();
            CollectionAssert.AreEqual(arrGRADOSE, Metodos.pixelbars);
        }

        [TestMethod]
        public void Pie1()
        {
            int[] arr = new int[] { 20, 30, 40, 10 };
            float[] arrGRADOSE = new float[] {72,108,144,36 };
            Metodos._arrayNumbers012= (int[])arr.Clone();
            Metodos.UnitTestPie();
            CollectionAssert.AreEqual(arrGRADOSE, Metodos.gradosN);
        }
        [TestMethod]
        public void Pie2()
        {
            int[] arr = new int[] { 208, 90, 140, 10 };
            float[] arrGRADOSE = new float[] { 167,72,112,8 };
            Metodos._arrayNumbers012 = (int[])arr.Clone();
            Metodos.UnitTestPie();
            CollectionAssert.AreEqual(arrGRADOSE, Metodos.gradosN);
        }
    }
}
```

Скриншоты



Orígenes de datos

Explorador de pruebas

Buscar

Ejecutar todo | Ejecutar... | Lista de reproducci

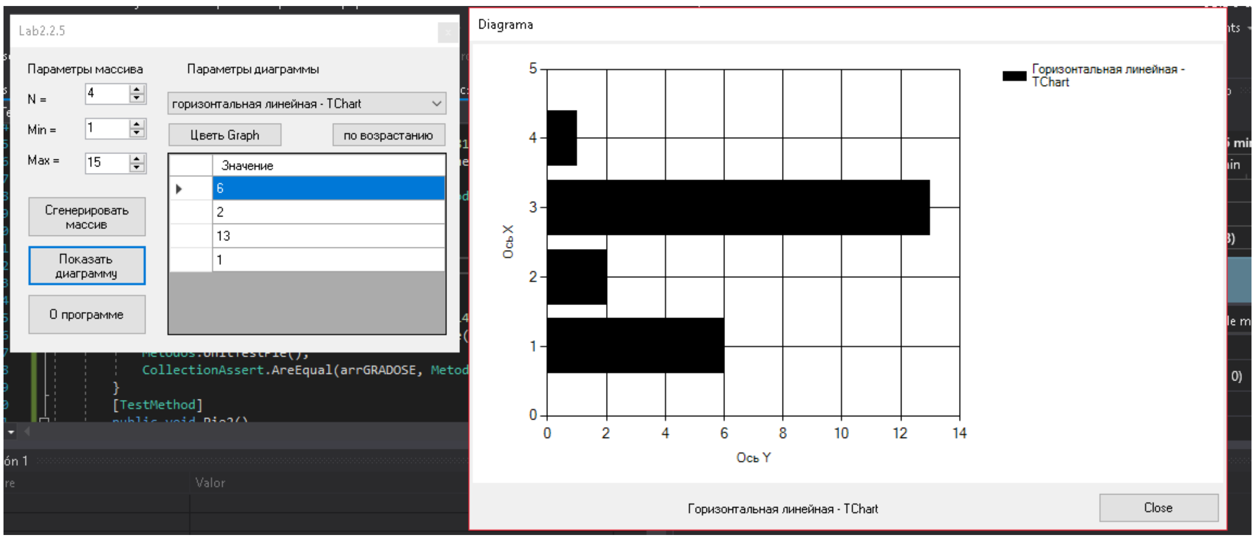
```
Lab2.2.5 (6 pruebas)
◢ ✓ UnitTest (6)            100 ms
  ◢ ✓ UnitTest (6)          100 ms
    ◢ ✓ UnitTest1 (6)       100 ms
        ✓ Bar1              < 1 ms
        ✓ Bar2              < 1 ms
        ✓ Pie1              < 1 ms
        ✓ Pie2              < 1 ms
        ✓ Sort1              99 ms
        ✓ Sort2             < 1 ms
```

Bar2 Copiar todo

Source: UnitTest1.cs, línea 42

✓ Bar2

Elapsed time: 0:00:00,0002081



Lab2.2.5

Параметры массива

N = 4

Min = 1

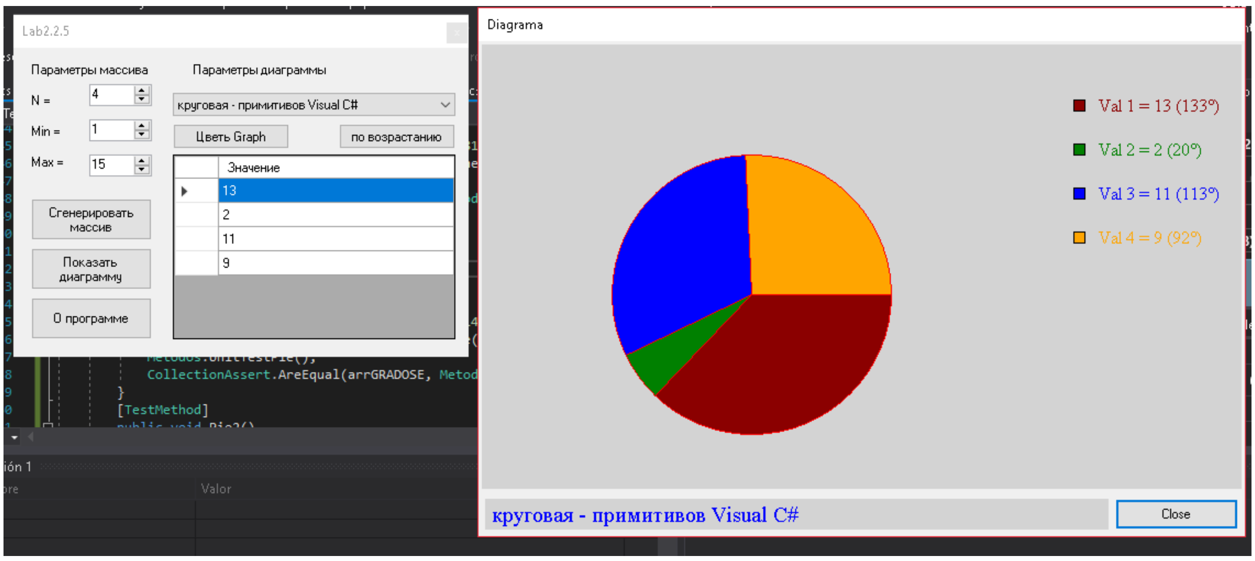
Max = 15

Параметры диаграммы

горизонтальная линейная - TChart

	Значение
▶	6
	2
	13
	1

```
CollectionAssert.AreEqual(arrGRADOSE, Metod
}
[TestMethod]
```



Lab2.2.5

Параметры массива

N = 4

Min = 1

Max = 15

Параметры диаграммы: круговая - примитивов Visual C#

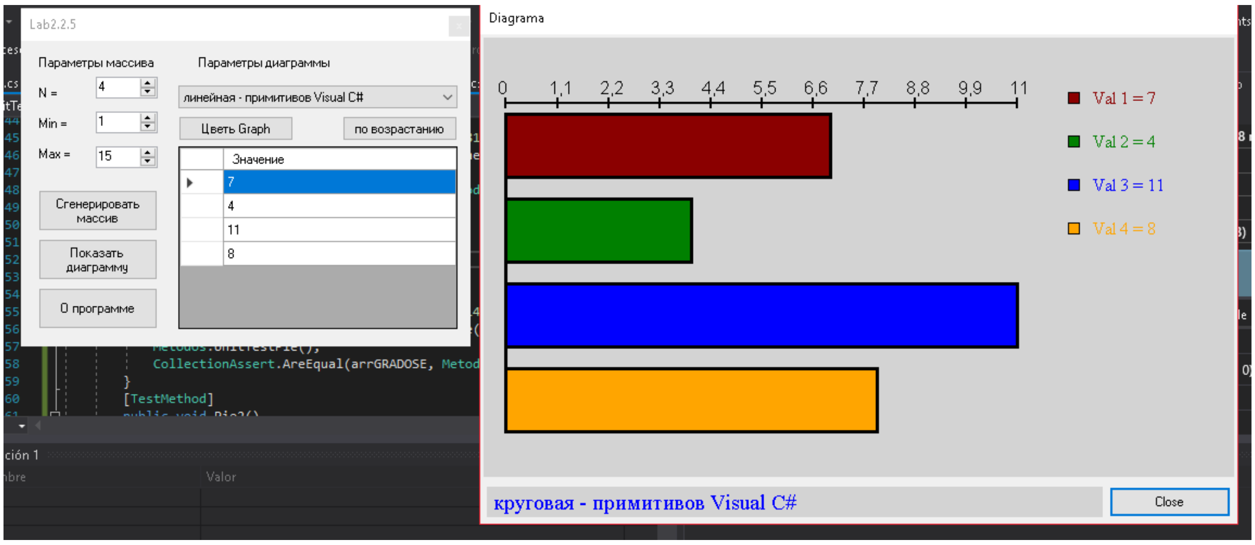
	Значение
▸	13
	2
	11
	9

Diagrama

- 🟥 Val 1 = 13 (133º)
- 🟩 Val 2 = 2 (20º)
- 🟦 Val 3 = 11 (113º)
- 🟧 Val 4 = 9 (92º)

круговая - примитивов Visual C#

```
CollectionAssert.AreEqual(arrGRADOSE, Metod
}
[TestMethod]
```



Lab2.2.5

Параметры массива

N = 4

Min = 1

Max = 15

Параметры диаграммы: линейная - примитивов Visual C#

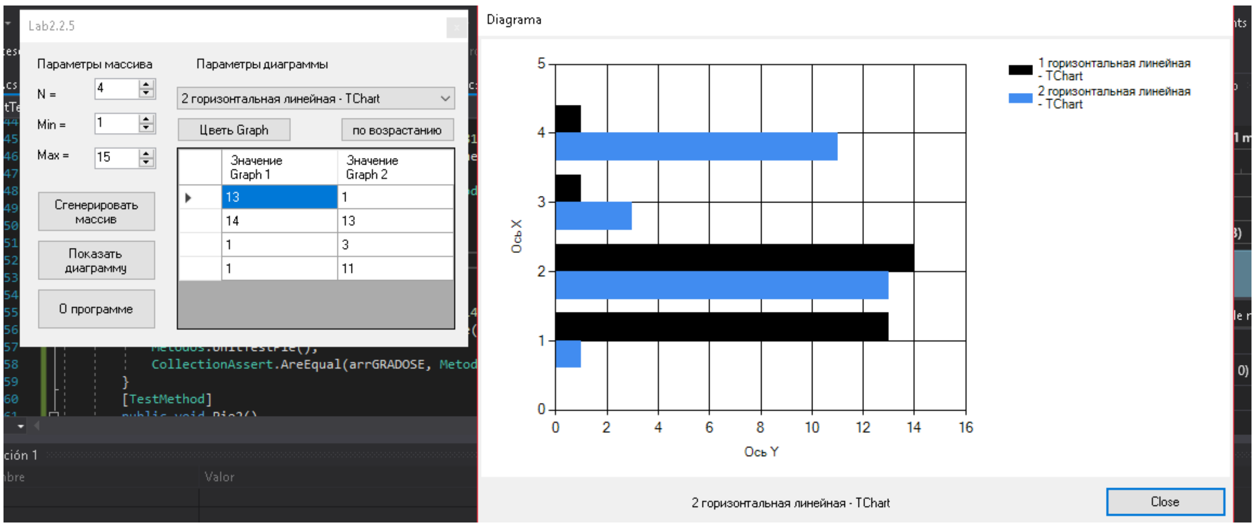
	Значение
▸	7
	4
	11
	8

Diagrama

- 🟥 Val 1 = 7
- 🟩 Val 2 = 4
- 🟦 Val 3 = 11
- 🟧 Val 4 = 8

круговая - примитивов Visual C#

```
CollectionAssert.AreEqual(arrGRADOSE, Metod
}
[TestMethod]
```



Lab2.2.5

Параметры массива

N = 4

Min = 1

Max = 15

Параметры диаграммы: 2 горизонтальная линейная - TChart

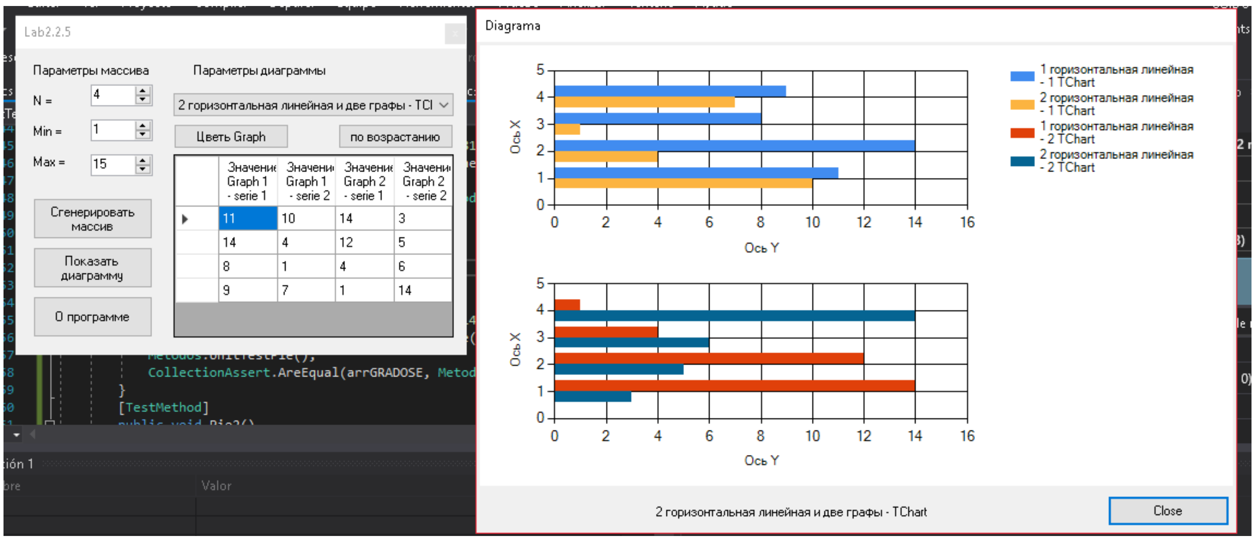
	Значение Graph 1	Значение Graph 2
▸	13	1
	14	13
	1	3
	1	11

Diagrama

- ■ 1 горизонтальная линейная - TChart
- 🟦 2 горизонтальная линейная - TChart

2 горизонтальная линейная - TChart

```
CollectionAssert.AreEqual(arrGRADOSE, Metod
}
[TestMethod]
```



Lab2.2.5

Параметры массива

N = 4

Min = 1

Max = 15

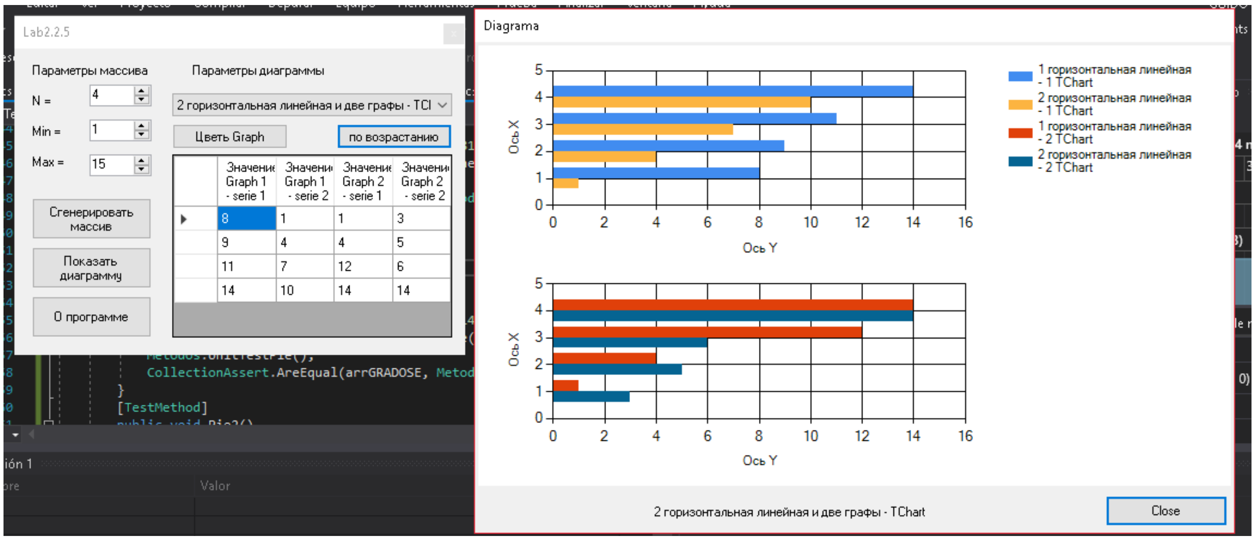
Параметры диаграммы

2 горизонтальная линейная и две графы - TCl

Значения Graph 1 - serie 1	Значения Graph 1 - serie 2	Значения Graph 2 - serie 1	Значения Graph 2 - serie 2
11	10	14	3
14	4	12	5
8	1	4	6
9	7	1	14

```
CollectionAssert.AreEqual(arrGRADOSE, Metod
}
[TestMethod]
```

Цветь Graph
по возрастанию
Сгенерировать массив
Показать диаграмму
О программе
Diagrama
Ось X
Ось Y
1 горизонтальная линейная - 1 TChart
2 горизонтальная линейная - 1 TChart
1 горизонтальная линейная - 2 TChart
2 горизонтальная линейная - 2 TChart
2 горизонтальная линейная и две графы - TChart
Close
Valor



Lab2.2.5

Параметры массива

N = 4

Min = 1

Max = 15

Параметры диаграммы

2 горизонтальная линейная и две графы - TCl

Значения Graph 1 - serie 1	Значения Graph 1 - serie 2	Значения Graph 2 - serie 1	Значения Graph 2 - serie 2
8	1	1	3
9	4	4	5
11	7	12	6
14	10	14	14

```
CollectionAssert.AreEqual(arrGRADOSE, Metod
}
[TestMethod]
```

Цветь Graph
по возрастанию
Сгенерировать массив
Показать диаграмму
О программе
Diagrama
Ось X
Ось Y
1 горизонтальная линейная - 1 TChart
2 горизонтальная линейная - 1 TChart
1 горизонтальная линейная - 2 TChart
2 горизонтальная линейная - 2 TChart
2 горизонтальная линейная и две графы - TChart
Close
Valor

Таблица 1 - Поля и методы класса А и их назначение

N°	**Поле**	**Назначение**
1	button1	Generate array
2	button2	Show graphs
3	button3	About program
4	button4	Sort
5	button5	Color grahs
6	comboBox1	Diagrams
7	colorDialog1	Color diagrams
8	dataGridView1	Show array
9	numericUpDown1	Array size
10	numericUpDown2	min
11	numericUpDown3	max
12	arr	Array with values of the primitive chart
13	button1	Exit
14	chart1	2-4 charts
15	gr	0-1 primitives
16	label1	Title for the charts
17	m_name	Title fot the primitives
18	panel	Panel to draw primitives
19	title	Panel to draw title pirmitive
20	_arrayNumber012	Array for 0,1,2 graphs
21	_arrayNumers3	Array for 3 graphs
22	_arrayNumbers4	Array for 4 graphs
23	color	Color graphs
24	diagrama	Type graphs
25	gradosN	Array pie unit test
26	pixelbars	Array bars unit test
	Метод	**Назначение**
27	FillData	Fill the values into dataview
28	Description	Show the description of graphs
29	HorizontalScale	Draw scale for the horizontal

		bars
30	BuildHorizontalChart	Draw horizontal bars
31	FillArrayPrimitive	Fill the array with primitive chart
32	GetColor	Get color to chart
33	Sort	Sort the array according the graph to show
34	GenerarArrayNumeros	Fill array according the graph
35	UnitTestPie	Logic Pie
36	UnitTestBar	Logit Bars
37	SumTotal	Get sum of the array values
38	MaxVal	Get max val of the array

Таблица 2 - Обработчики событий проекта и их назначение

N°	**Обработчик события**	**Назначение**
1	button3_Click	Show about autor
2	button2_Click	Show Graphs
3	numericUpDown2_Value Changed	Check min
4	numericUpDown3_Value Changed	Check max
5	button1_Click	Generate random numbers according to the type of graph
6	Form1_Load	Load the necessary values when the program star
7	checkBox1_CheckedChan ged	Select the color for the graph
8	button4_Click	Button to sort
9	comboBox1_SelectedInde xChanged	Select the graph type
10	button5_Click	Select the color for the graph
11	Gtitle	Draw tittle 0-1
12	Ggraphs	/Draw Graphs
13	Diagrama_Resize	Re draw

Выводы

В лабораторной практике изучалось "Разработка и исследование программы построения графиков”.
Данные были представлены в статистических графиках с использованием графических примитивов C # и функции «Диаграмма». Панель использовалась для рисования примитивов в макете.
Как улучшение: Реализуйте больше статистических характеристик для представленных данных.

Использованные источники

1. https://docs.microsoft.com/es-es/dotnet/