Project 3

Wprowadzenie do programowania

Język C#

Denys Savchenko

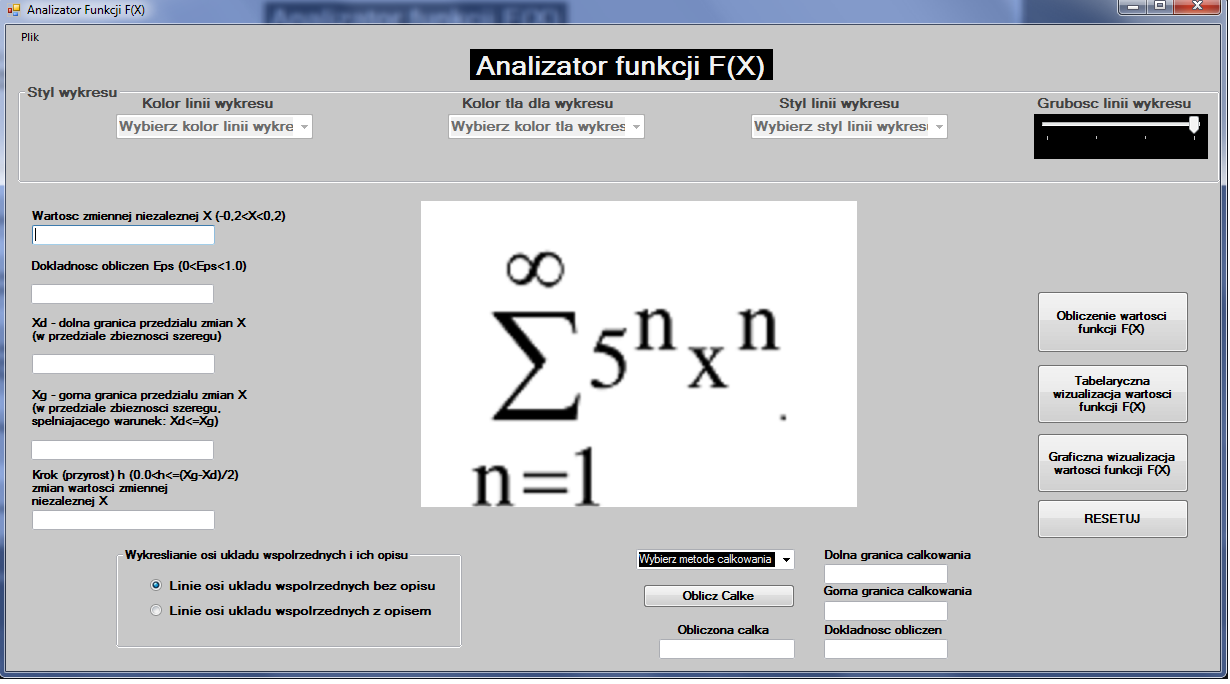
Nr.Albumu - 51908

Student 1-ego roku na kierunku ‘Informatyka’

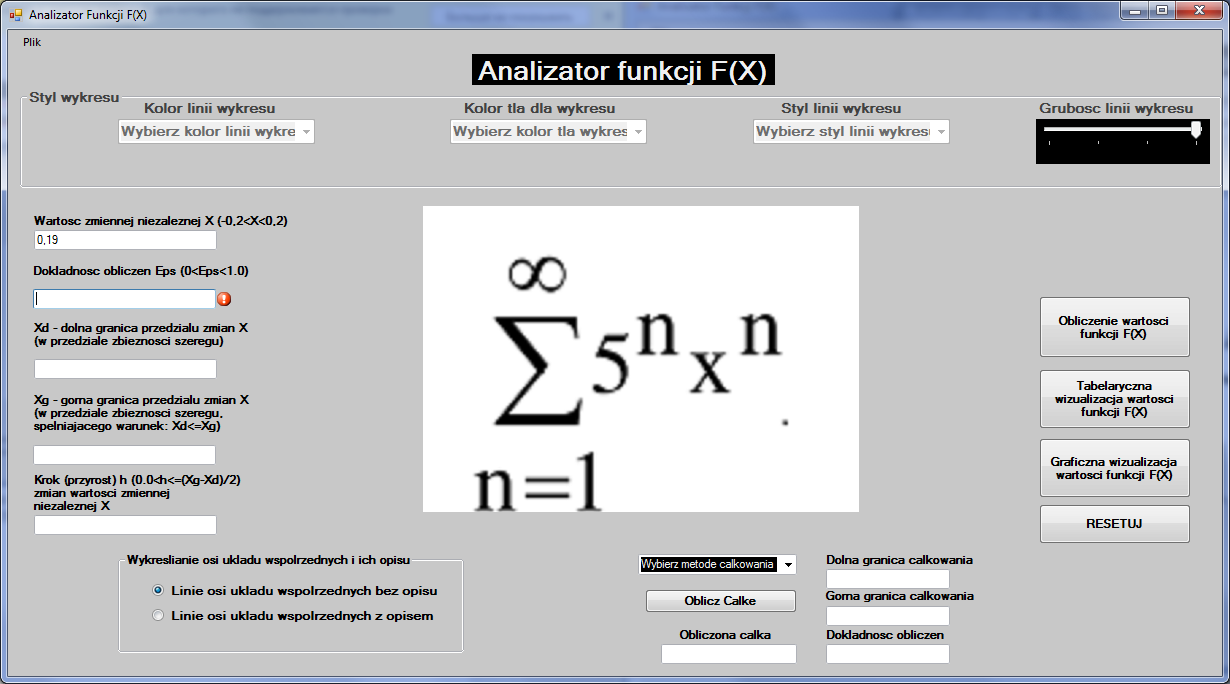
Grupa 3

Analizator Funkcji

* Otworzony fomularz oraz funkcja, którą będziemy analizować.



* Przy nacisnięciu na przycisk Obliczenie Wartości Funkcji, będzie wyłołana funkcja dla Pobrania Dannych: Warości X(-0,2<X<0,2) oraz Dokładność obliczeń Eps(0<Eps<1.0).



* Metoda do Pobrania Dannych Wejściowych opisana w Visual Studio:

private bool dsPobranieDannychObliczenie(out double dsx, out double dseps)

{

//errorProvider1.Dispose();

dsx = 0;

dseps = 0;

if (string.IsNullOrEmpty(dstbX.Text) || !double.TryParse(dstbX.Text, out dsx) || Math.Abs(dsx)>=0.2)

{

if (string.IsNullOrEmpty(dstbX.Text))

{

errorProvider1.SetError(dstbX, "ERROR: musisz wpisac wartosc!");

return false;

}

else if (!double.TryParse(dstbX.Text, out dsx))

{

errorProvider1.SetError(dstbX, "Error: podana wartosc jest nieprawidlowa!");

return false;

}

else if(Math.Abs(dsx) >= 0.2)

{

errorProvider1.SetError(dstbX, "Error: podana wartosc musi spelniacz warunek (-0.2 < X < 0.2)");

return false;

}

}

else errorProvider1.Dispose();

if (string.IsNullOrEmpty(dstbEps.Text) || !double.TryParse(dstbEps.Text, out dseps) || (Math.Abs(dseps) > 0.0 && Math.Abs(dseps) >= 1.0) || dseps == 0.0)

{

if (string.IsNullOrEmpty(dstbEps.Text))

{

errorProvider1.SetError(dstbEps, "ERROR: musisz wpisac wartosc!");

return false;

}

else if (!double.TryParse(dstbEps.Text, out dseps))

{

errorProvider1.SetError(dstbEps, "Error: podana wartosc jest nieprawidlowa!");

return false;

}

else

{

errorProvider1.SetError(dstbEps, "Error: podana wartosc musi spelniacz warunek (0 < Eps < 1.0)");

return false;

}

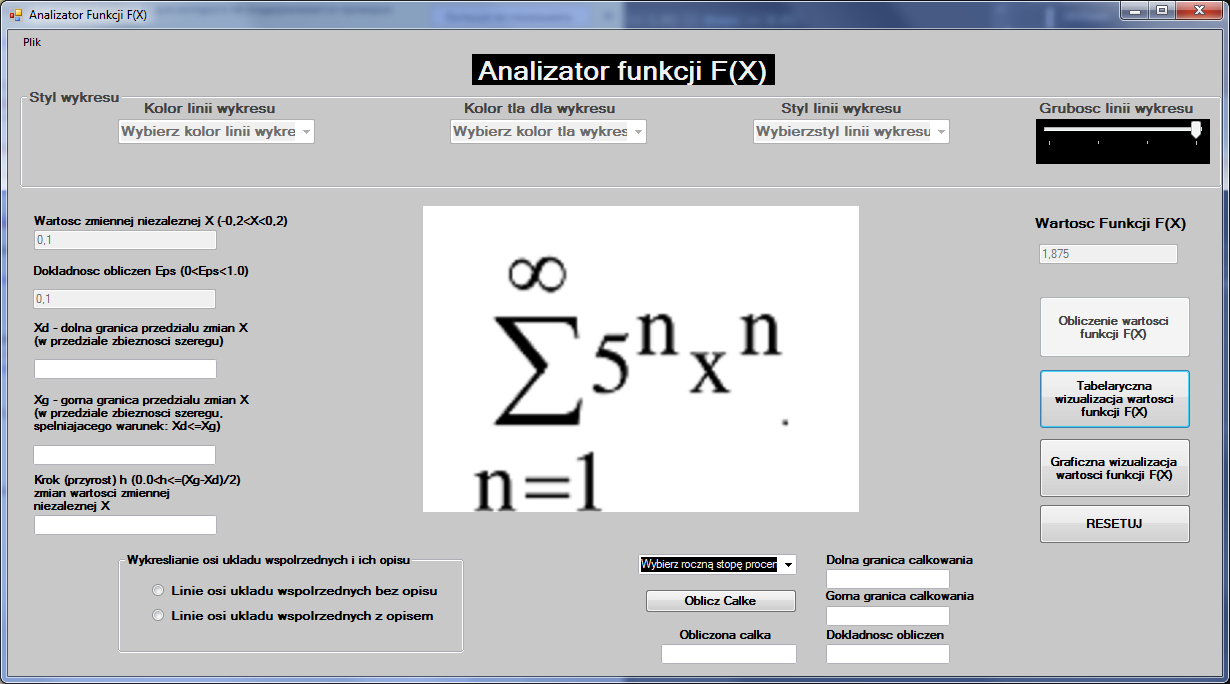
}

else errorProvider1.Dispose();

return true;

}

* Wynnik po nacisnięciu na przycisk Obliczenie Wartości funkcji:



* Obsługa przycisku Obliczenie Wartości Funkcji:

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

{

double dsx, dseps,wynnik;

if(!dsPobranieDannychObliczenie(out dsx, out dseps))

{

return;

}

wynnik=Obliczenie(dsx,dseps);

dslbWartosc.Visible = true;

dstbWynikFunkcji.Visible = true;

dstbWynikFunkcji.Text = wynnik.ToString();

dstbWynikFunkcji.Enabled = false;

dstbX.Enabled = false;

dstbEps.Enabled = false;

dsbtnObliczWartosc.Enabled = false;

}

private double Obliczenie(double dsx, double dseps)

{

double dsSum=0;

double dsn = 0;

double dsw = 1;

while (Math.Abs(dsw) > dseps)

{

dsSum += dsw;

dsn++;

dsw \*= Math.Pow(5,dsn)\*Math.Pow(dsx,dsn)/dsw;

};

return dsSum;

}

private double Obliczenie(double dsx, double dseps, out ushort dsn)

{

double dsSum = 0;

dsn = 0;

double dsw = 1;

while (Math.Abs(dsw) > dseps)

{

dsSum += dsw;

dsn++;

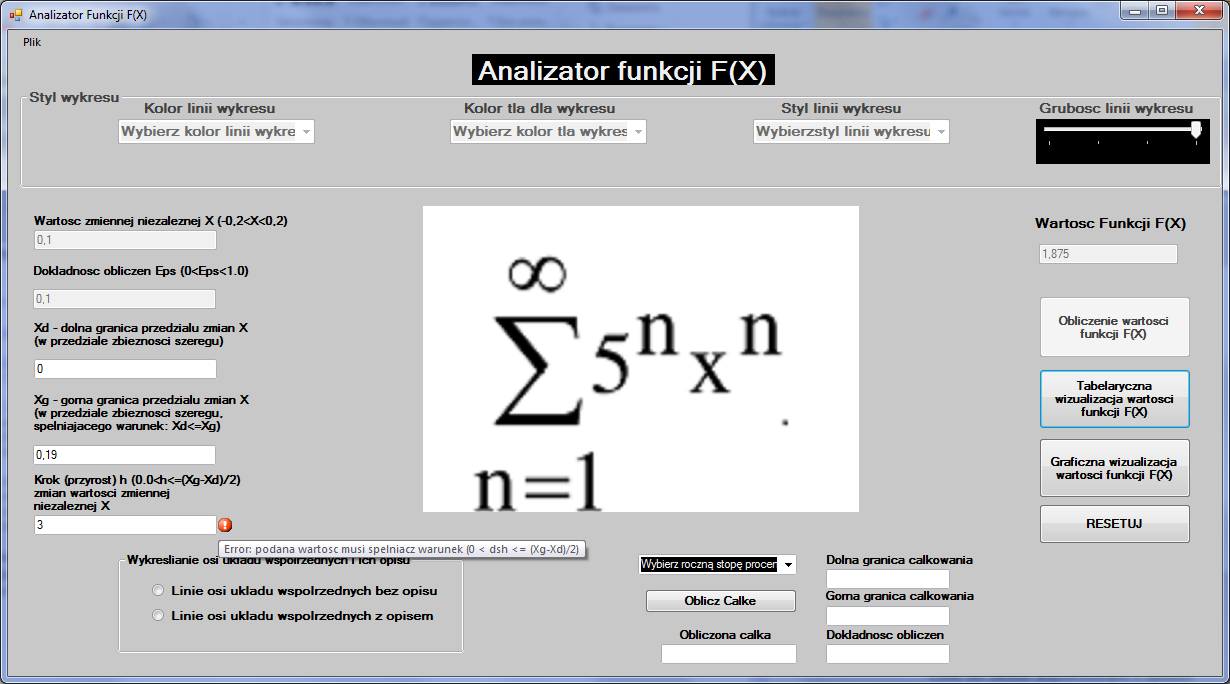
dsw \*= Math.Pow(5, dsn) \* Math.Pow(dsx, dsn) / dsw;

};

return dsSum;

}

* Przy naciśnięciu na przycisk Tabelaryczna Wizualizacja Wartości Funkcji, będzie wyłołana inna funkcja dla Pobrania Dannych: Dokładność obliczeń Eps(0<Eps<1.0), Dolna granica(-0,2<X<0,2), Górna Granica (-0,2<X<0,2 oraz xd<=xg), Krok przyrost (0.0<h<(xg-xd)/2);



* Obsługa Pobrania Dannych dla tego przycisku:

private bool dsPobranieDanychTablaAndGragic(out double dseps, out double dsxd, out double dsxg, out double dsh ){

dseps = 0; dsxd = 0; dsxg = 0; dsh = 0;

if (string.IsNullOrEmpty(dstbEps.Text) || !double.TryParse(dstbEps.Text, out dseps) || (Math.Abs(dseps) > 0.0 && Math.Abs(dseps) >= 1.0) || dseps == 0.0)

{

if (string.IsNullOrEmpty(dstbEps.Text))

{

errorProvider1.SetError(dstbEps, "ERROR: musisz wpisac wartosc!");

return false;

}

else if (!double.TryParse(dstbEps.Text, out dseps))

{

errorProvider1.SetError(dstbEps, "Error: podana wartosc jest nieprawidlowa!");

return false;

}

else

{

errorProvider1.SetError(dstbEps, "Error: podana wartosc musi spelniacz warunek (0 < Eps < 1.0)");

return false;

}

}

else errorProvider1.Dispose();

if (string.IsNullOrEmpty(dstbXd.Text) || !double.TryParse(dstbXd.Text, out dsxd) || Math.Abs(dsxd) >= 0.2)

{

if (string.IsNullOrEmpty(dstbXd.Text))

{

errorProvider1.SetError(dstbXd, "ERROR: musisz wpisac wartosc!");

return false;

}

else if (!double.TryParse(dstbXd.Text, out dsxd))

{

errorProvider1.SetError(dstbXd, "Error: podana wartosc jest nieprawidlowa!");

return false;

}

else if (Math.Abs(dsxd) >= 0.2)

{

errorProvider1.SetError(dstbXd, "Error: podana wartosc musi spelniacz warunek (-0.2 < Xd < 0.2)");

return false;

}

}

else errorProvider1.Dispose();

if (string.IsNullOrEmpty(dstbXg.Text) || !double.TryParse(dstbXg.Text, out dsxg) || dsxg<dsxd || dsxg >= 0.2)

{

if (string.IsNullOrEmpty(dstbXg.Text))

{

errorProvider1.SetError(dstbXg, "ERROR: musisz wpisac wartosc!");

return false;

}

else if (!double.TryParse(dstbXg.Text, out dsxg))

{

errorProvider1.SetError(dstbXg, "Error: podana wartosc jest nieprawidlowa!");

return false;

}

else if (dsxg >= 0.2)

{

errorProvider1.SetError(dstbXg, "Error: podana wartosc musi spelniacz warunek (Xd <= Xg < 0.2)");

return false;

}

else

{

errorProvider1.SetError(dstbXg, "Error: podana wartosc musi spelniacz warunek (Xd <= Xg)");

return false;

}

}

else errorProvider1.Dispose();

if (string.IsNullOrEmpty(dstbH.Text) || !double.TryParse(dstbH.Text, out dsh) || (Math.Abs(dsh) > 0.0 && Math.Abs(dsh) > ((dsxg-dsxd)/2)))

{

if (string.IsNullOrEmpty(dstbH.Text))

{

errorProvider1.SetError(dstbH, "ERROR: musisz wpisac wartosc!");

return false;

}

else if (!double.TryParse(dstbH.Text, out dsh))

{

errorProvider1.SetError(dstbH, "Error: podana wartosc jest nieprawidlowa!");

return false;

}

else

{

errorProvider1.SetError(dstbH, "Error: podana wartosc musi spelniacz warunek (0 < dsh <= (Xg-Xd)/2)");

return false;

}

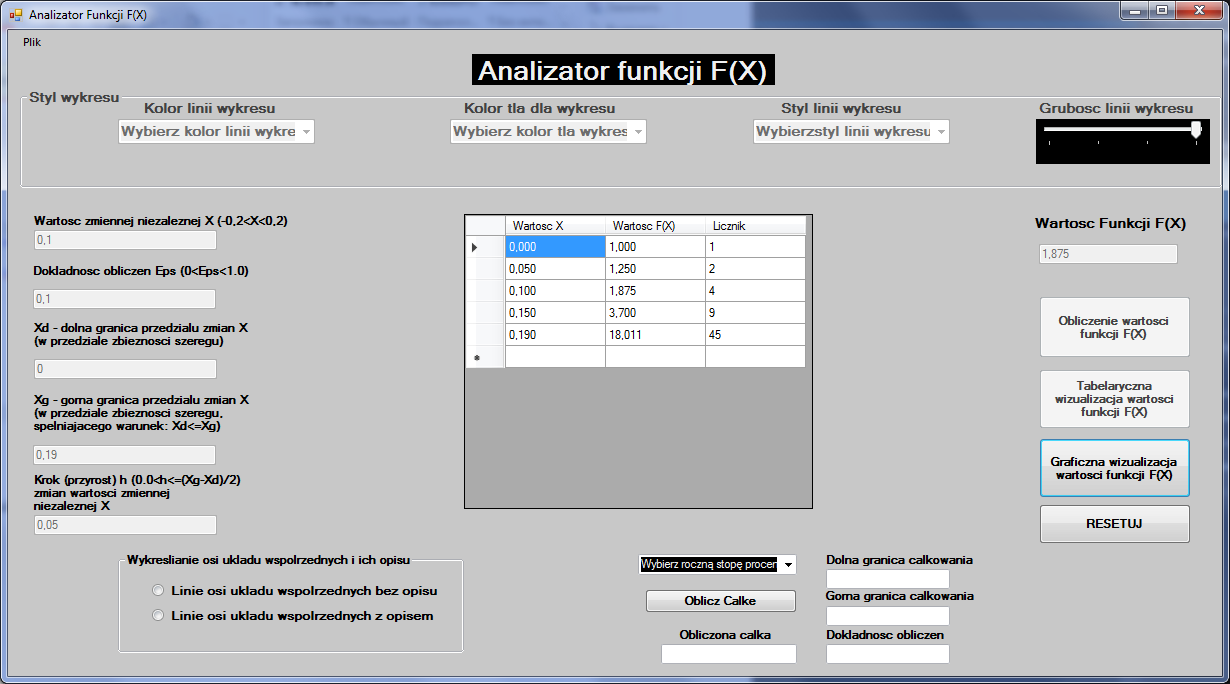
}

else errorProvider1.Dispose();

return true;

}

* Wynnik po naciśnięciu na przycisk Tabelaryczna Wizualizacja Wartości Funkcji:



* Obsługa przycisku Tabelaryczna Wizualizacja Wartości Funkcji:

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

{

double dsx, dseps, dsxg, dsxd,dsh;

if (!dsPobranieDanychTablaAndGragic(out dseps,out dsxd,out dsxg,out dsh))

{

return;

}

dsdgv.Rows.Clear();

ushort licznik;

dsx = dsxd;

int dsi;

for( dsi = 0; dsx <= dsxg; dsi++, dsx = dsxd + dsi \* dsh)

{

if (dsx == dsxg)

break;

dsdgv.Rows.Add();

dsdgv.Rows[dsi].Cells[0].Value =string.Format("{0:0.000}",dsx);

dsdgv.Rows[dsi].Cells[1].Value = string.Format("{0:0.000}", Obliczenie(dsx,dseps,out licznik));

dsdgv.Rows[dsi].Cells[2].Value = licznik;

}

dsdgv.Rows.Add();

dsdgv.Rows[dsi].Cells[0].Value = string.Format("{0:0.000}", dsxg);

dsdgv.Rows[dsi].Cells[1].Value = string.Format("{0:0.000}", Obliczenie(dsxg, dseps, out licznik));

dsdgv.Rows[dsi].Cells[2].Value = licznik;

pictureBox1.Visible = false;

dsch.Visible = false;

dsdgv.Visible = true;

dstbEps.Enabled = false;

dstbXd.Enabled = false;

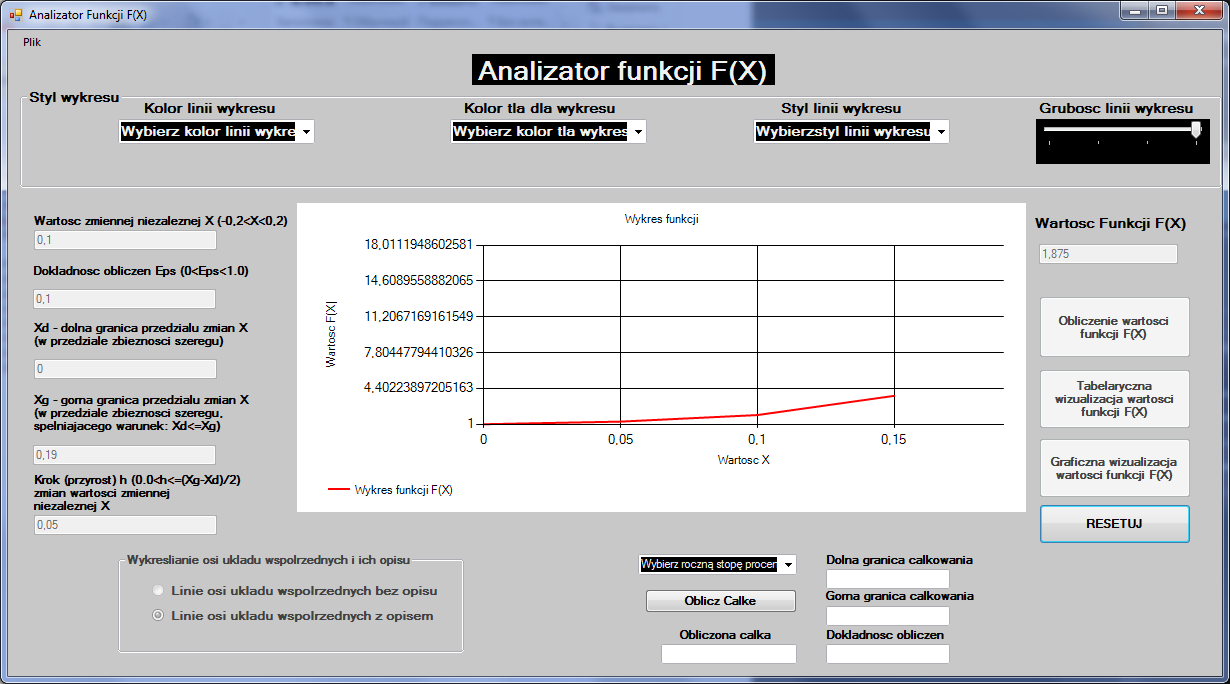
dstbXg.Enabled = false;

dstbH.Enabled = false;

dsbtnTabela.Enabled = false;

}

* Wynnik po nacisnięciu na przycisk Graficzna Wizualizacja Wartości funkcji(Metoda Pobrania dannych taka sama jak i była do przycisku Tabelaryczna Wizualizacja Wartości funkcji, oprócz kontrolek radiobox Linii osi układu z opisem czy bez – jest zrealizowana w obsłudze przycisku Graficzna Wizualizacja Wartości funkcji ):



* Obsługa przycisku Graficzna Wizualizacja Wartości funkcji:

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

{

double dsx, dseps, dsxg, dsxd, dsh;

if (!dsPobranieDanychTablaAndGragic(out dseps, out dsxd, out dsxg, out dsh))

{

return;

}

dsx = dsxd;

dsch.Titles.Add("Wykres funkcji");

dsch.Series.Clear();

dsch.Series.Add("Wykres funkcji F(X)");

if (dsrbZOpisem.Checked == true)

dsch.ChartAreas[0].AxisX.Title = "Wartosc X";

dsch.ChartAreas[0].AxisX.Minimum = dsxd;

dsch.ChartAreas[0].AxisX.Maximum = dsxg;

dsch.ChartAreas[0].AxisX.Interval = dsh;

if (dsrbZOpisem.Checked == true)

dsch.ChartAreas[0].AxisY.Title = "Wartosc F(X)";

dsch.ChartAreas[0].AxisY.Minimum = Obliczenie(dsxd,dseps);

dsch.ChartAreas[0].AxisY.Maximum = Obliczenie(dsxg, dseps);

dsch.Series[0].IsVisibleInLegend = true;

dsch.Legends.FindByName("Legend1").Docking = System.Windows.Forms.DataVisualization.Charting.Docking.Bottom;

dsch.Series[0].Name = "Wykres funkcji F(X)";

dsch.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;

dsch.Series[0].Color = Color.Red;

//dsch.BackColor = Color.Green;

dsch.Series[0].BorderDashStyle = System.Windows.Forms.DataVisualization.Charting.ChartDashStyle.Solid;

dsch.Series[0].BorderWidth = 2;

int dsi = 0;

for ( dsi = 0; dsx<=dsxg; dsi++, dsx=dsxd+dsi\*dsh)

{

dsch.Series[0].Points.AddXY(dsx, Obliczenie(dsx, dseps));

}

pictureBox1.Visible = false;

dsdgv.Visible = false;

dsch.Visible = true;

dsgbWykress.Enabled = true;

dstbEps.Enabled = false;

dstbXd.Enabled = false;

dstbXg.Enabled = false;

dstbH.Enabled = false;

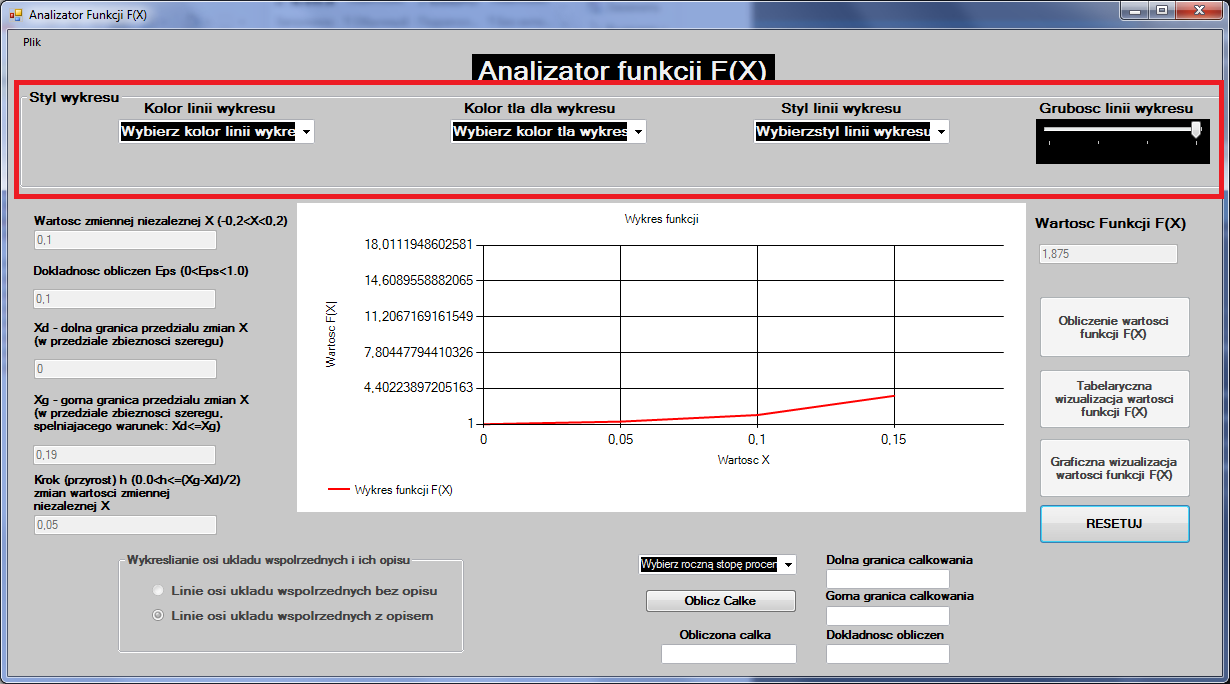
dsbtnTabela.Enabled = false;

dsgbOs.Enabled = false;

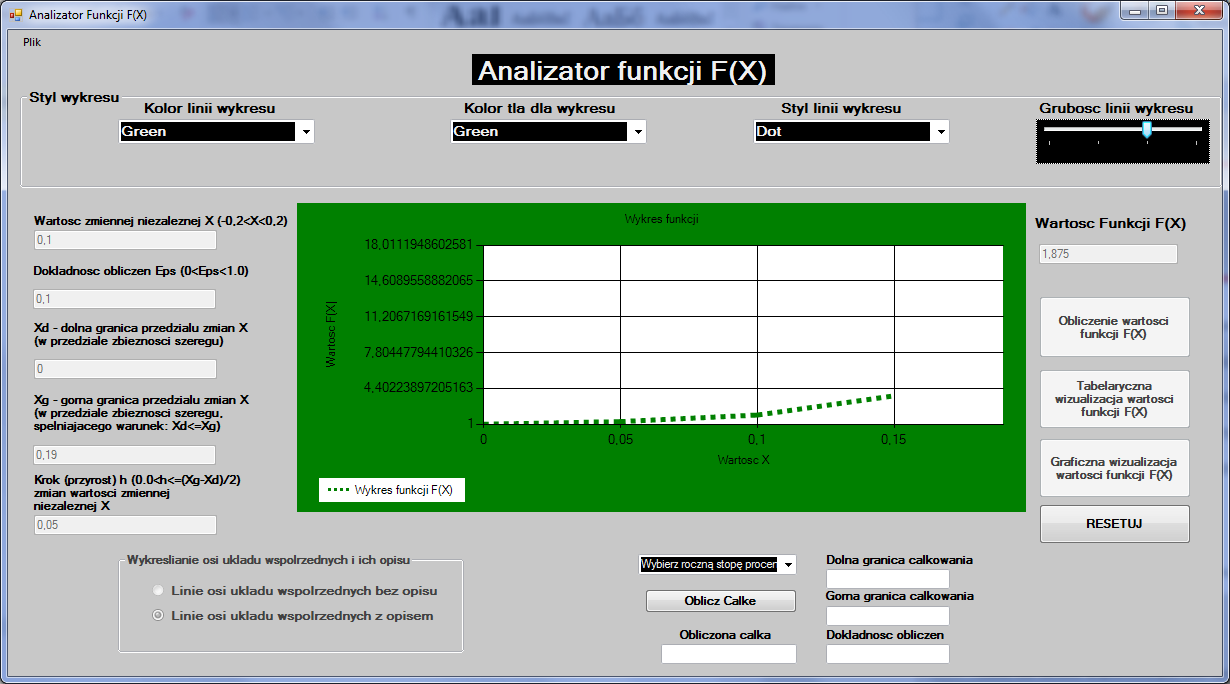
dsbtnGraphic.Enabled = false;

}

* Po naciśnięciu przycisku Graficzna Wizualizacja Wartości Funkcji zostaje odblokowany rozdział (Group Box) ‘Styl Wykresu’



* Ustalamy sobie wygląd wykresu funkcji taki, jaki chcemy:



* Obsługa wyboru koloru linii wykresu:

private void dscbKolorLinii\_SelectedIndexChanged(object sender, EventArgs e)

{

if (dscbKolorLinii.SelectedIndex == 0)

{

dsch.Series[0].Color = Color.Green;

}

else if (dscbKolorLinii.SelectedIndex == 1)

{

dsch.Series[0].Color = Color.Red;

}

else if (dscbKolorLinii.SelectedIndex == 2)

{

dsch.Series[0].Color = Color.Blue;

}

else if (dscbKolorLinii.SelectedIndex == 3)

{

dsch.Series[0].Color = Color.Yellow;

}

}

* Obsługa wyboru koloru tła wykresu:

private void dscmKolorTla\_SelectedIndexChanged(object sender, EventArgs e)

{

if (dscmKolorTla.SelectedIndex == 0)

{

dsch.BackColor = Color.Green;

}

else if (dscmKolorTla.SelectedIndex == 1)

{

dsch.BackColor = Color.Red;

}

else if (dscmKolorTla.SelectedIndex == 2)

{

dsch.BackColor = Color.Blue;

}

else if (dscmKolorTla.SelectedIndex == 3)

{

dsch.BackColor = Color.Yellow;

}

}

* Obsługa wyboru stylu linii wykresu:

private void dscbStylLinii\_SelectedIndexChanged(object sender, EventArgs e)

{

if (dscbStylLinii.SelectedIndex == 0)

{

dsch.Series[0].BorderDashStyle = System.Windows.Forms.DataVisualization.Charting.ChartDashStyle.Dash;

}

else if (dscbStylLinii.SelectedIndex == 1)

{

dsch.Series[0].BorderDashStyle = System.Windows.Forms.DataVisualization.Charting.ChartDashStyle.Dot;

}

else if (dscbStylLinii.SelectedIndex == 2)

{

dsch.Series[0].BorderDashStyle = System.Windows.Forms.DataVisualization.Charting.ChartDashStyle.DashDot;

}

else if (dscbStylLinii.SelectedIndex == 3)

{

dsch.Series[0].BorderDashStyle = System.Windows.Forms.DataVisualization.Charting.ChartDashStyle.Solid;

}

}

* Obsługa wyboru grubości linii wykresu:

private void dstbGruboscLinii\_Scroll(object sender, EventArgs e)

{

if (dstbGruboscLinii.Value == 1)

{

dsch.Series[0].BorderWidth = 3;

}

else if (dstbGruboscLinii.Value == 2)

{

dsch.Series[0].BorderWidth = 4;

}

else if (dstbGruboscLinii.Value == 3)

{

dsch.Series[0].BorderWidth = 5;

}

else if (dstbGruboscLinii.Value == 4)

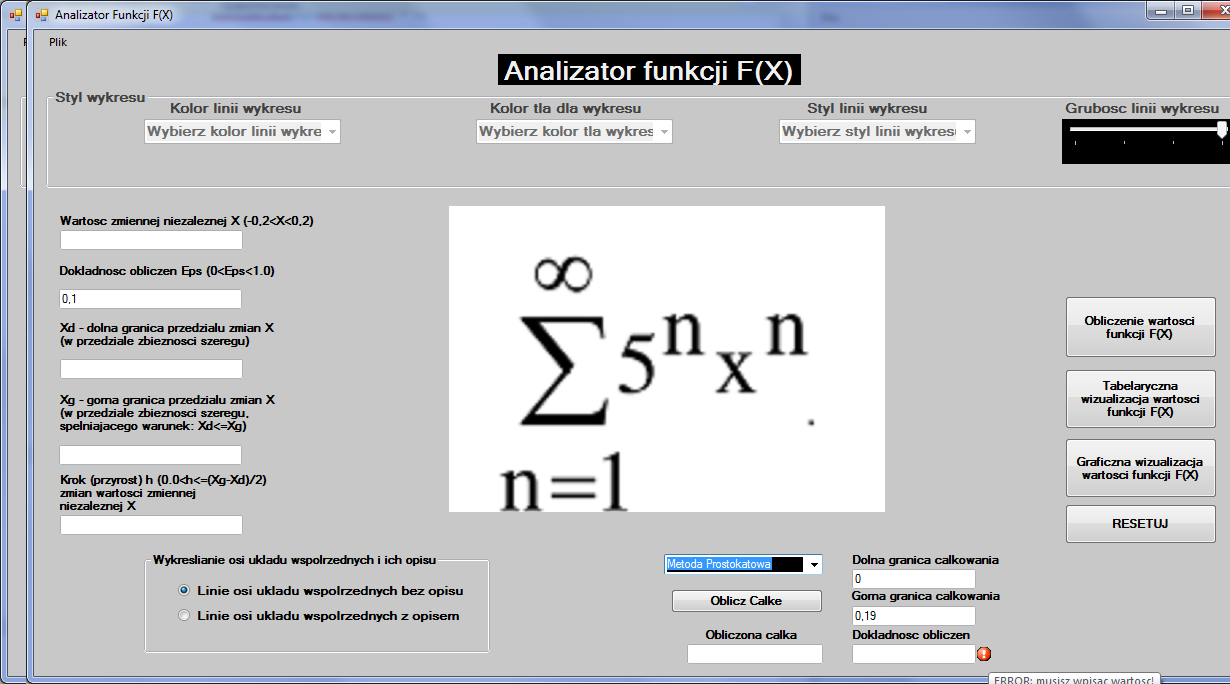
{

dsch.Series[0].BorderWidth = 6;

}

}

* Przy naciśnięciu na przycisk Oblicz Całkę, wyłołuje się metoda Pobrania Danych Wejściowych:( Dokładność Obliczen Szeregu 0<eps<1, Dolna Granica(-0,2<xd<0,2), Górna granica(-0,2<xd<0,2 oraz xg>=xd), Dokładność Obliczen Całki 0<eps<1 oraz Metoda Całkowania(Prostokątowa lub Trapezowa))



* Obsługa metody Pobrania Danych :

private bool PobranieDannychCalkowanie(out double dseps,out double dsxd, out double dsxg, out double dsd, out byte dsm)

{

dsxd = 0;

dseps = 0;

dsxg = 0;

dsd = 0;

dsm = 0;

if (string.IsNullOrEmpty(dstbEps.Text) || !double.TryParse(dstbEps.Text, out dseps) || (Math.Abs(dseps) > 0.0 && Math.Abs(dseps) >= 1.0) || dseps == 0.0)

{

if (string.IsNullOrEmpty(dstbEps.Text))

{

errorProvider2.SetError(dstbEps, "ERROR: musisz wpisac wartosc!");

return false;

}

else if (!double.TryParse(dstbEps.Text, out dseps))

{

errorProvider2.SetError(dstbEps, "Error: podana wartosc jest nieprawidlowa!");

return false;

}

else

{

errorProvider2.SetError(dstbEps, "Error: podana wartosc musi spelniacz warunek (0 < Eps < 1.0)");

return false;

}

}

if (string.IsNullOrEmpty(dstbXdCalk.Text) || !double.TryParse(dstbXdCalk.Text, out dsxd)|| Math.Abs(dsxd) >= 0.2)

{

if (string.IsNullOrEmpty(dstbXdCalk.Text))

{

errorProvider2.SetError(dstbXdCalk, "ERROR: musisz wpisac wartosc!");

return false;

}

else if (!double.TryParse(dstbXdCalk.Text, out dsxd))

{

errorProvider2.SetError(dstbXdCalk, "Error: podana wartosc jest nieprawidlowa!");

return false;

}

else if (Math.Abs(dsxd) >= 0.2)

{

errorProvider2.SetError(dstbXdCalk, "Error: podana wartosc musi spelniacz warunek (-0.2 < Xd < 0.2)");

return false;

}

}

else errorProvider2.Dispose();

if (string.IsNullOrEmpty(dstbXgCalk.Text) || !double.TryParse(dstbXgCalk.Text, out dsxg) || dsxg < dsxd || dsxg >= 0.2)

{

if (string.IsNullOrEmpty(dstbXgCalk.Text))

{

errorProvider2.SetError(dstbXgCalk, "ERROR: musisz wpisac wartosc!");

return false;

}

else if (!double.TryParse(dstbXgCalk.Text, out dsxg))

{

errorProvider2.SetError(dstbXgCalk, "Error: podana wartosc jest nieprawidlowa!");

return false;

}

else if (dsxg >= 0.2)

{

errorProvider2.SetError(dstbXgCalk, "Error: podana wartosc musi spelniacz warunek (Xd <= Xg < 0.2)");

return false;

}

else

{

errorProvider2.SetError(dstbXgCalk, "Error: podana wartosc musi spelniacz warunek (Xd <= Xg)");

return false;

}

}

else errorProvider2.Dispose();

if (string.IsNullOrEmpty(dstbEpsCalk.Text) || !double.TryParse(dstbEpsCalk.Text, out dsd) || (Math.Abs(dsd) > 0.0 && Math.Abs(dsd) >= 1.0) || dsd == 0.0)

{

if (string.IsNullOrEmpty(dstbEpsCalk.Text))

{

errorProvider2.SetError(dstbEpsCalk, "ERROR: musisz wpisac wartosc!");

return false;

}

else if (!double.TryParse(dstbEpsCalk.Text, out dsd))

{

errorProvider2.SetError(dstbEpsCalk, "Error: podana wartosc jest nieprawidlowa!");

return false;

}

else

{

errorProvider2.SetError(dstbEpsCalk, "Error: podana wartosc musi spelniacz warunek (0 < dsd < 1.0)");

return false;

}

}

else errorProvider2.Dispose();

if (dscbMetodaCalk.SelectedIndex == 0) dsm = 1;

else if (dscbMetodaCalk.SelectedIndex == 1) dsm = 2;

else if(dscbMetodaCalk.SelectedIndex==-1)

{

errorProvider2.SetError(dscbMetodaCalk, "Error: musisz wybrac metode calkowania");

return false;

}

else errorProvider2.Dispose();

return true;

}

* Obsługa przycisku Oblicz Całkę:

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

{

double dsxd, dsxg, dsd,dseps,dsh,dsci,dsci\_1,dssumaFx,dsx,dsszerokosc;

ushort dslicznik;

int dslicznikprzed;

byte dsm;

if (!PobranieDannychCalkowanie(out dseps, out dsxd, out dsxg, out dsd, out dsm))

{

return;

}

if(dsm==1)

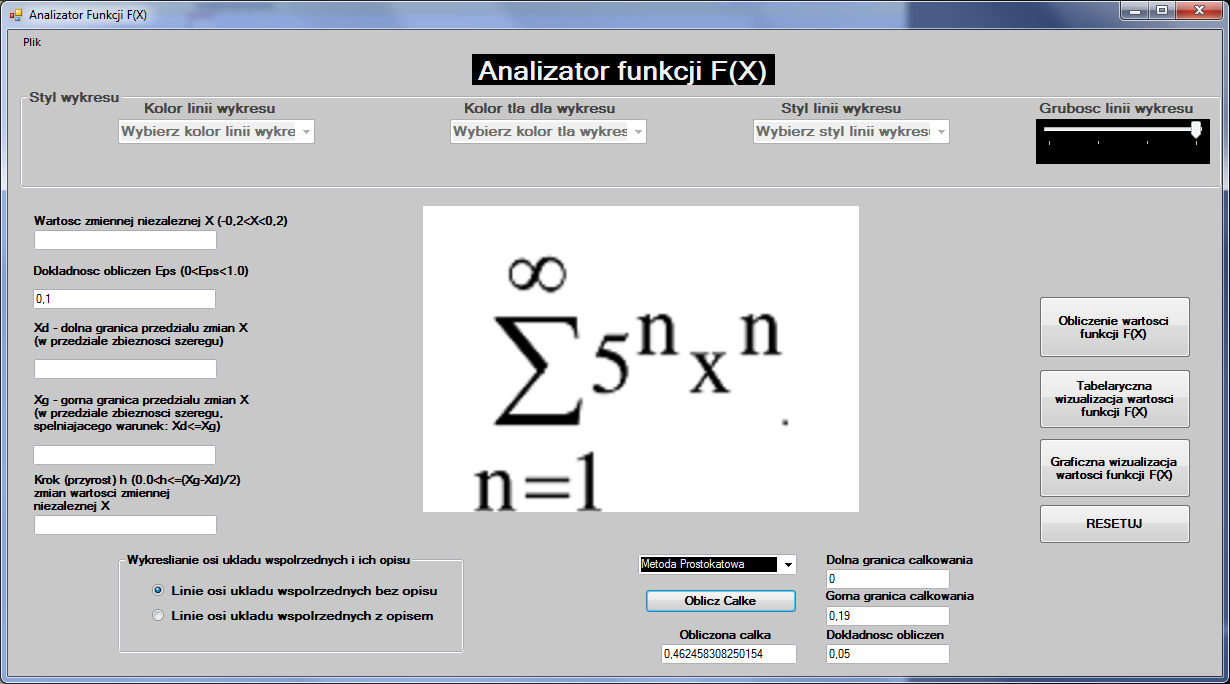
dstbObliczonaCalka.Text = dsMetodaProstoK(dseps,dsxd,dsxg,dsd,out dslicznikprzed,out dsszerokosc).ToString();

else

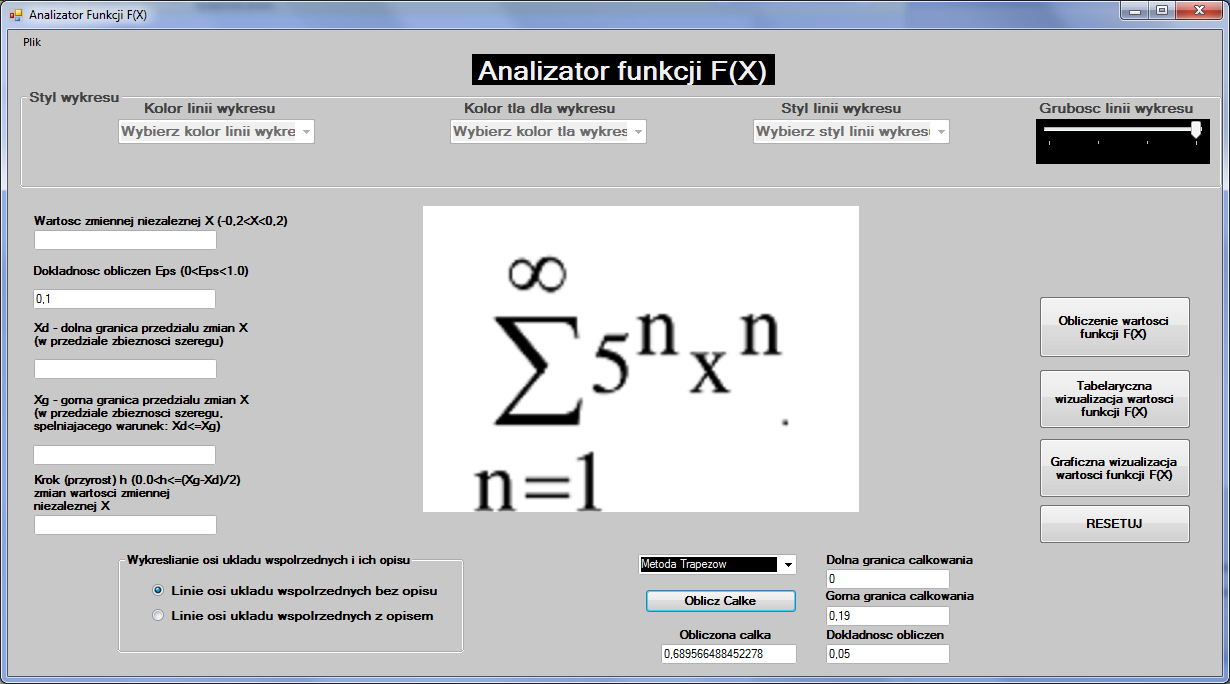
dstbObliczonaCalka.Text = dsMetodaTrapezow(dseps, dsxd, dsxg, dsd, out dslicznikprzed, out dsszerokosc).ToString();

}

* Wynik całkowania (Metoda Prostokątowa):

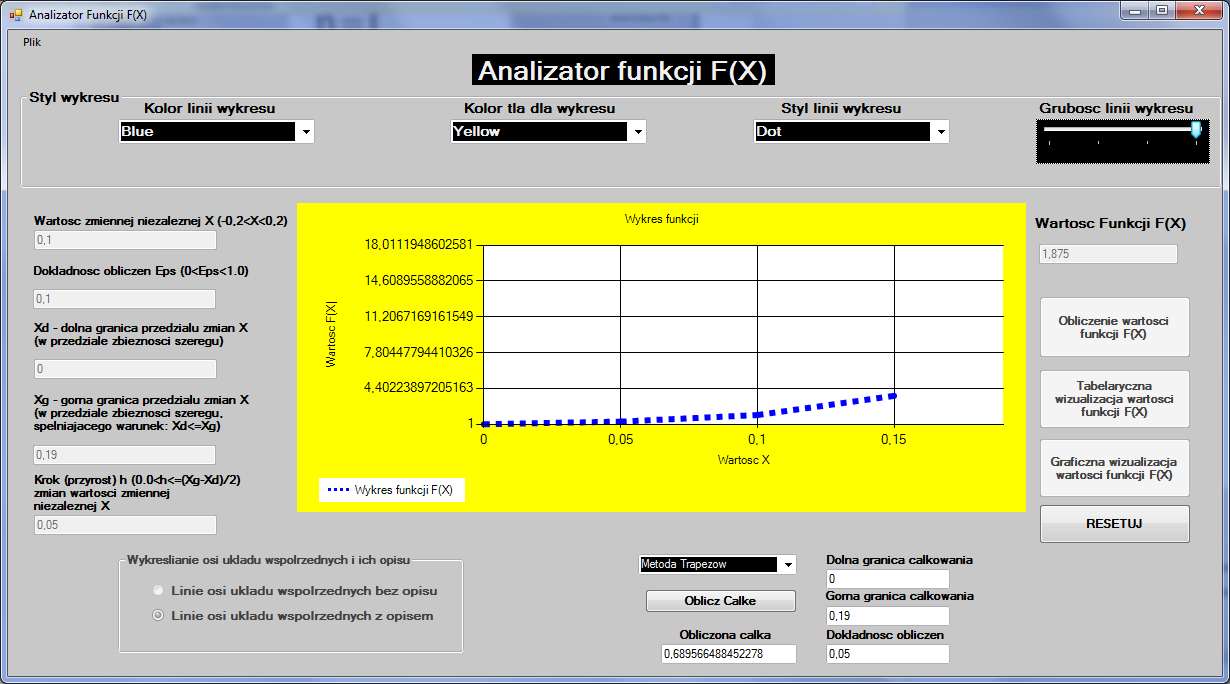


* Wynnik całkowania (Metoda Trapezów):

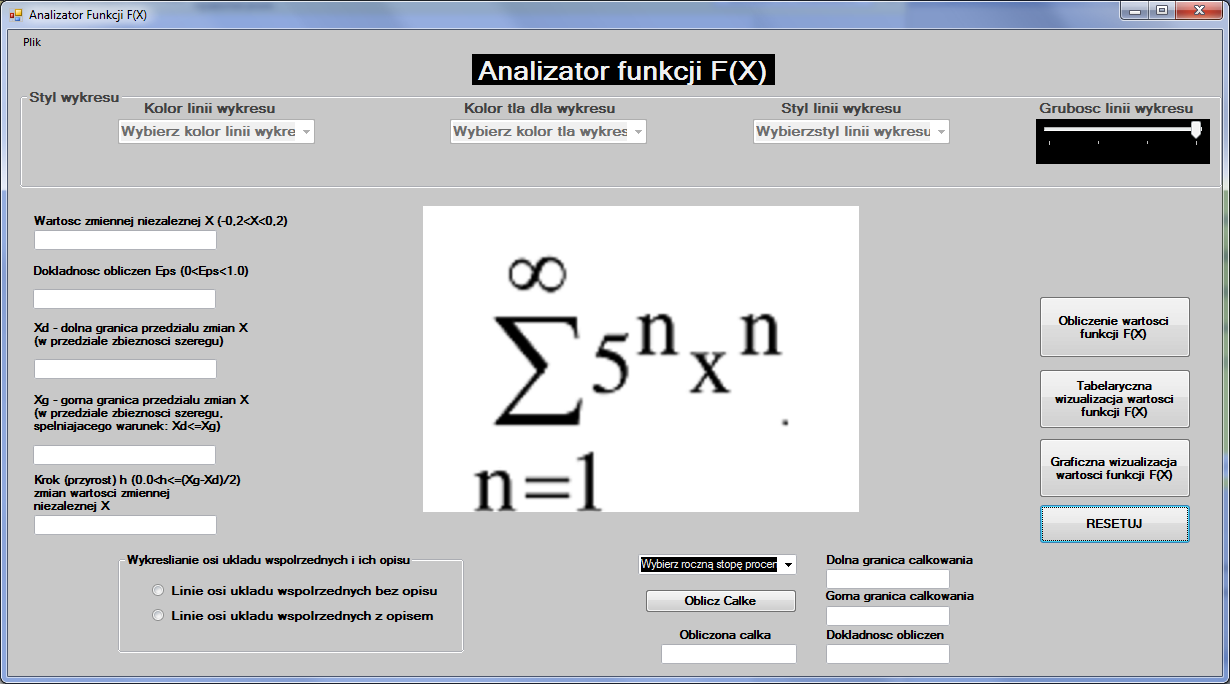


* Przycisk Resetuj

Do naciśnięcia:



Po naciśnięciu:



* Obsługa przycisku Resetuj:

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

{

errorProvider1.Dispose();

errorProvider2.Dispose();

dslbWartosc.Visible = false;

dstbWynikFunkcji.Enabled = true;

dstbWynikFunkcji.Clear();

dstbWynikFunkcji.Visible = false;

dstbX.Enabled = true ;

dstbX.Clear();

dstbEps.Enabled = true;

dstbEps.Clear();

dstbXd.Enabled = true;

dstbXd.Clear();

dstbXg.Enabled = true;

dstbXg.Clear();

dstbH.Enabled = true;

dstbH.Clear();

if (dsrbBezOpisu.Checked == true)

dsrbBezOpisu.Checked = false;

if (dsrbZOpisem.Checked == true)

dsrbZOpisem.Checked = false;

dscbMetodaCalk.SelectedIndex = 0;

dscbMetodaCalk.Text = "Wybierz roczną stopę procentową";

dscbMetodaCalk.Enabled = true;

dscbKolorLinii.SelectedIndex = 0;

dscbKolorLinii.Text = "Wybierz kolor linii wykresu";

dscmKolorTla.SelectedIndex = 0;

dscmKolorTla.Text = "Wybierz kolor tla wykresu";

dscbStylLinii.SelectedIndex = 0;

dscbStylLinii.Text = "Wybierz styl linii wykresu";

dstbObliczonaCalka.Clear();

dstbObliczonaCalka.Enabled = true;

dstbXdCalk.Enabled = true;

dstbXdCalk.Clear();

dstbXgCalk.Enabled = true;

dstbXgCalk.Clear();

dstbEpsCalk.Enabled = true;

dstbEpsCalk.Clear();

dsbtnCalka.Enabled = true;

dsbtnObliczWartosc.Enabled = true;

dsbtnGraphic.Enabled = true;

dsbtnTabela.Enabled = true;

dsdgv.Rows.Clear();

dsdgv.Visible = false;

dsch.Visible = false;

pictureBox1.Visible = true;

dsch.Series.Clear();

dsch.BackColor = Color.White;

dsch.Titles.Clear();

dscbKolorLinii.SelectedIndex = 0;

dscbKolorLinii.Text = "Wybierz kolor linii wykresu";

dscmKolorTla.SelectedIndex = 0;

dscmKolorTla.Text = "Wybierz kolor tla wykresu";

dscbStylLinii.SelectedIndex = 0;

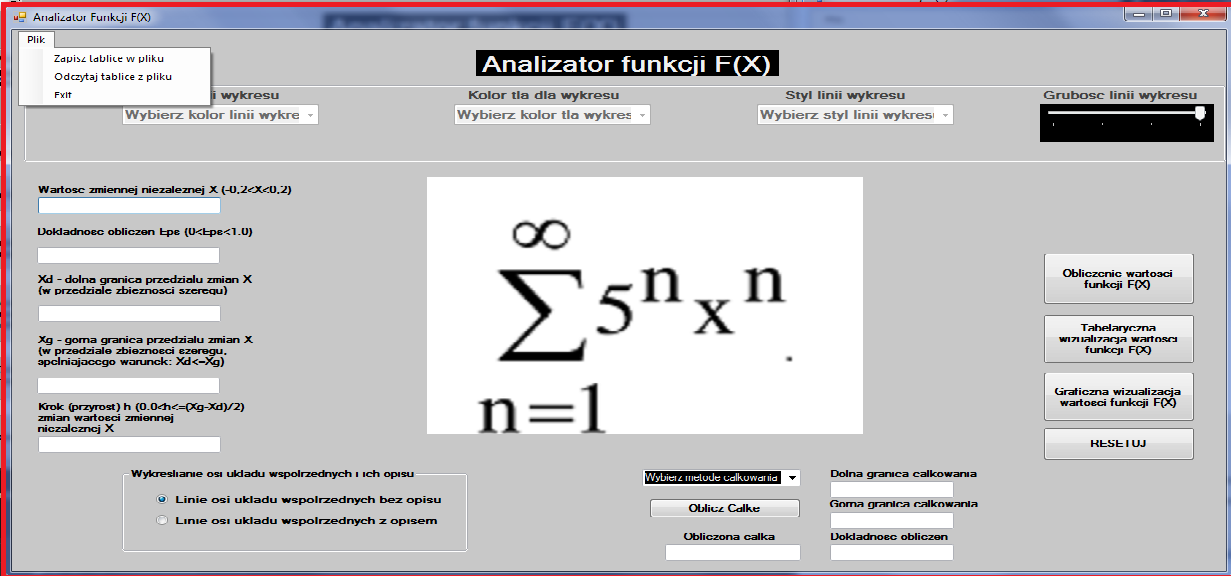
dscbStylLinii.Text = "Wybierzstyl linii wykresu";

dsgbWykress.Enabled = false;

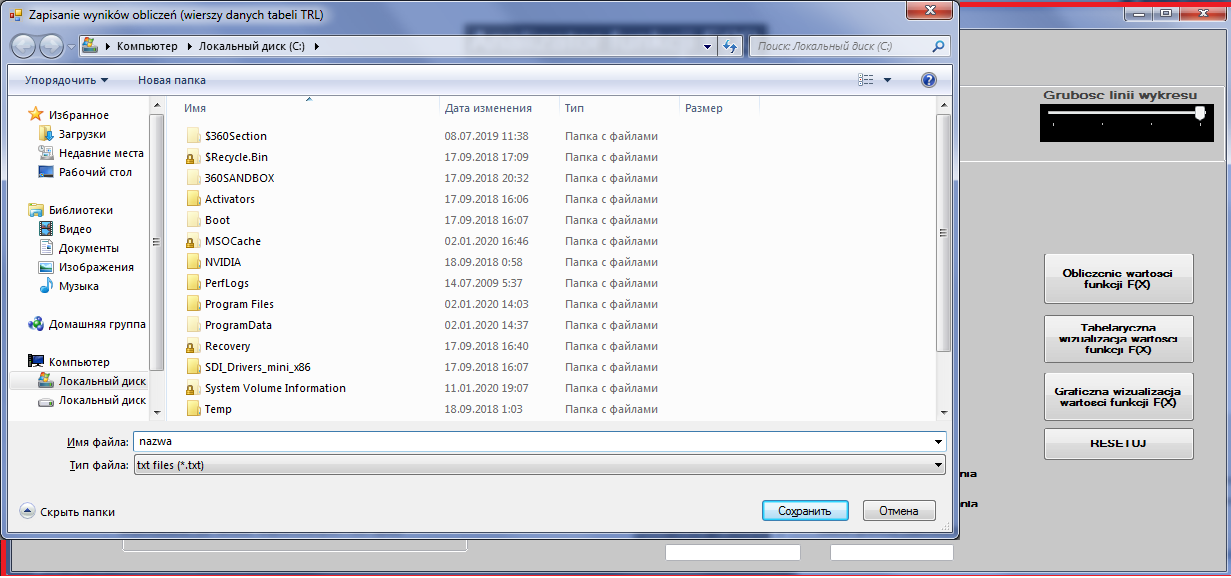
dsgbOs.Enabled = true;

}

* Przy naciśnięciu na przycisk Plik (na górze) będzie można wybrać punkty(Zapisać w plik, Odczytać z pliku, Exit):



* Obsługa Przycisku Zapisz tablicę w pliku:



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

{

if(dsbtnTabela.Enabled == false)

{

double dsx, dseps, dsxd, dsxg, dsh;

if (!dsPobranieDanychTablaAndGragic(out dseps, out dsxd, out dsxg, out dsh))

return;

dsx = dsxd;

ushort licznik;

SaveFileDialog OknoZapisu = new SaveFileDialog();

OknoZapisu.Filter = "txt files (\*.txt)|\*.txt|All Files (\*.\*)|\*.\*";

OknoZapisu.FilterIndex = 1;

OknoZapisu.RestoreDirectory = true;

OknoZapisu.InitialDirectory = "C:\\";

OknoZapisu.Title = "Zapisanie wyników obliczeń (wierszy danych tabeli TRL)";

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

{

StreamWriter Plik = new StreamWriter(OknoZapisu.OpenFile());

int dsi = 0;

for (dsi = 0; dsx <= dsxg; dsi++, dsx = dsxd + dsi \* dsh)

{

if (dsx == dsxg)

break;

Plik.Write(string.Format("{0,8:0.000} \t", dsx));

Plik.Write(string.Format("{0,8:0.000} \t", Obliczenie(dsx, dseps, out licznik)));

Plik.Write(string.Format("{0,8:0} \t", licznik));

Plik.WriteLine("\t");

}

Plik.Write(string.Format("{0,8:0.000} \t", dsx));

Plik.Write(string.Format("{0,8:0.000} \t", Obliczenie(dsx, dseps, out licznik)));

Plik.Write(string.Format("{0,8:0} \t", licznik));

Plik.WriteLine("\t");

Plik.Dispose();

Plik.Close();

}

}

else

{

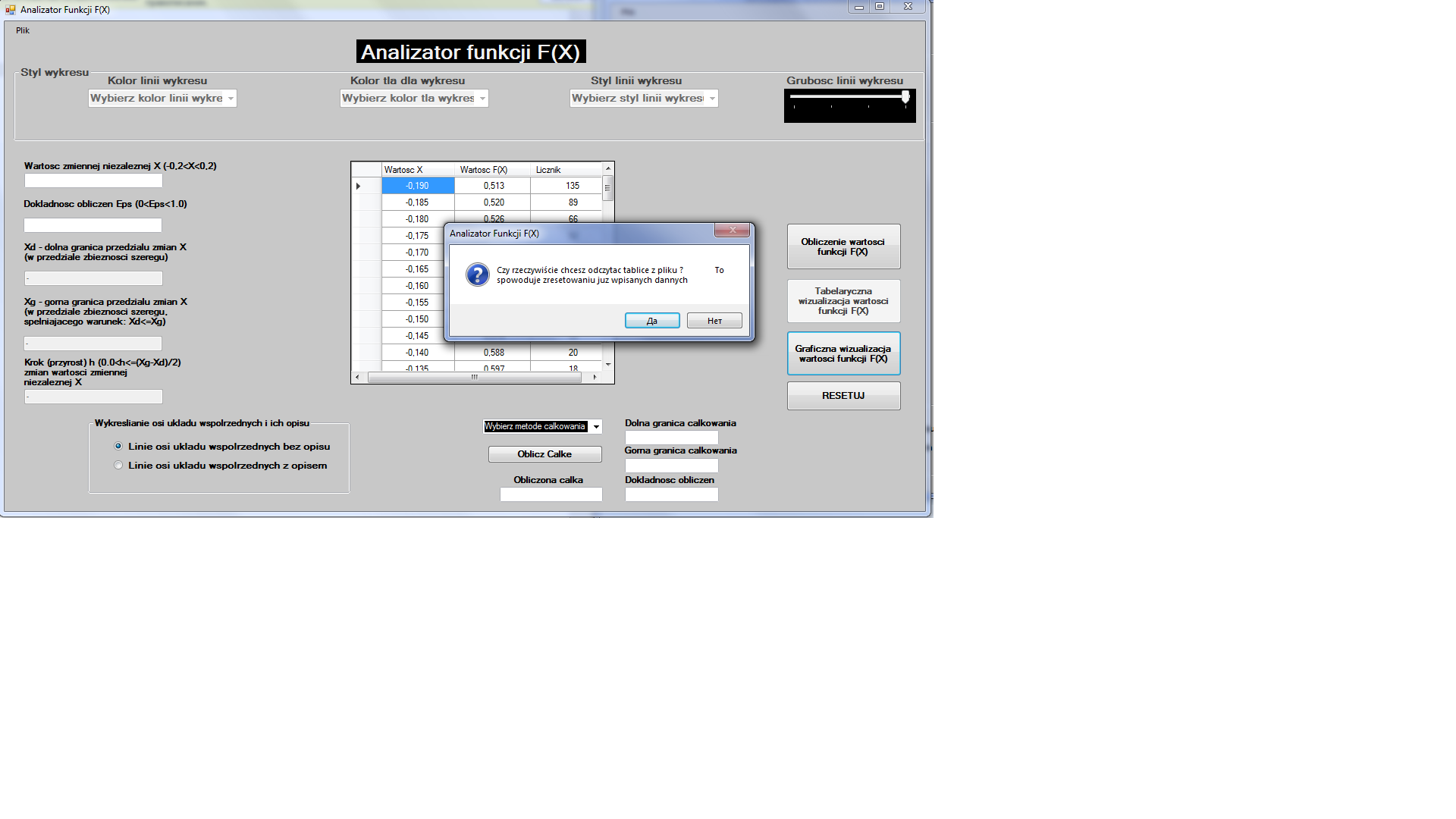
DialogResult Pytanie = MessageBox.Show("Nie mozesz zapisac tablice do pliku, bo nie masz ja!",

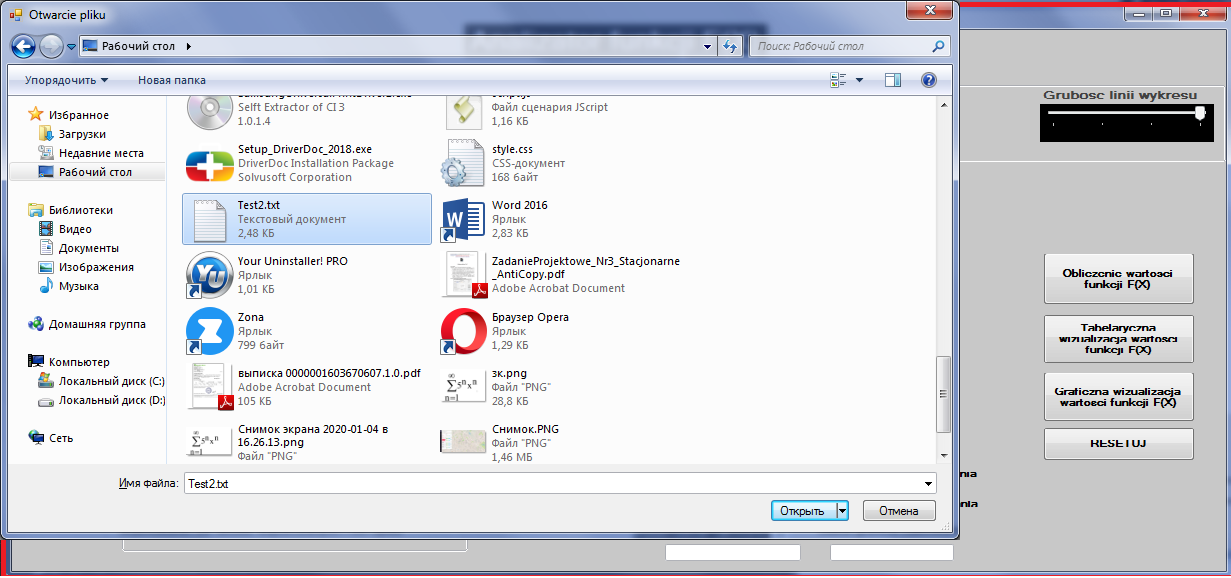
this.Text, MessageBoxButtons.OK, MessageBoxIcon.Question, MessageBoxDefaultButton.Button3);

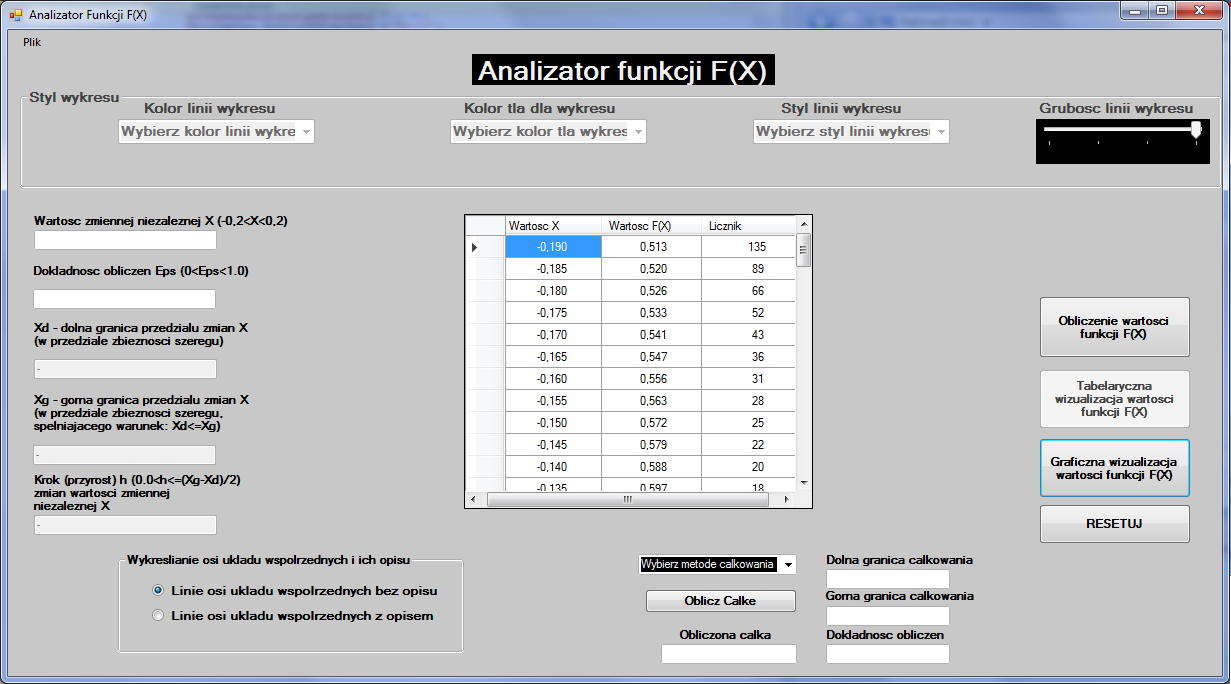
}

}

* Obsługa Przycisku Odczytaj tablicę z pliku:







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

{

if (dsbtnTabela.Enabled == false)

{

DialogResult Pytanie = MessageBox.Show("Czy rzeczywiście chcesz odczytac tablice z pliku ?\tTo spowoduje zresetowaniu juz wpisanych dannych",

this.Text, MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button3);

switch (Pytanie)

{

case DialogResult.Yes:

dsdgv.Rows.Clear();

pictureBox1.Visible = false;

dsch.Visible = false;

OpenFileDialog OknoOdczytu = new OpenFileDialog();

OknoOdczytu.Title = "Otwarcie pliku";

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

{

string nazwaPliku = OknoOdczytu.FileName;

string[] zawartoscPliku = File.ReadAllLines(nazwaPliku);

dsdgv.RowTemplate.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

foreach (string wiersz in zawartoscPliku)

{

dsdgv.Rows.Add(wiersz.Split('\t'));

dsdgv.Visible = true;

}

dstbXd.Enabled = false;

dstbXd.Text = "-";

dstbXg.Enabled = false;

dstbXg.Text = "-";

dstbH.Enabled = false;

dstbH.Text = "-";

}

dstbEps.Clear();

dstbEps.Enabled = true;

break;

case DialogResult.No:

break;

}

}

else

{

dsdgv.Rows.Clear();

pictureBox1.Visible = false;

dsch.Visible = false;

OpenFileDialog OknoOdczytu = new OpenFileDialog();

OknoOdczytu.Title = "Otwarcie pliku" ;

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

{

string nazwaPliku = OknoOdczytu.FileName;

string[] zawartoscPliku = File.ReadAllLines(nazwaPliku);

dsdgv.RowTemplate.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;

foreach (string wiersz in zawartoscPliku)

{

dsdgv.Rows.Add(wiersz.Split('\t'));

dsdgv.Visible = true;

}

dstbXd.Enabled = false;

dstbXd.Text = "-";

dstbXg.Enabled = false;

dstbXg.Text = "-";

dstbH.Enabled = false;

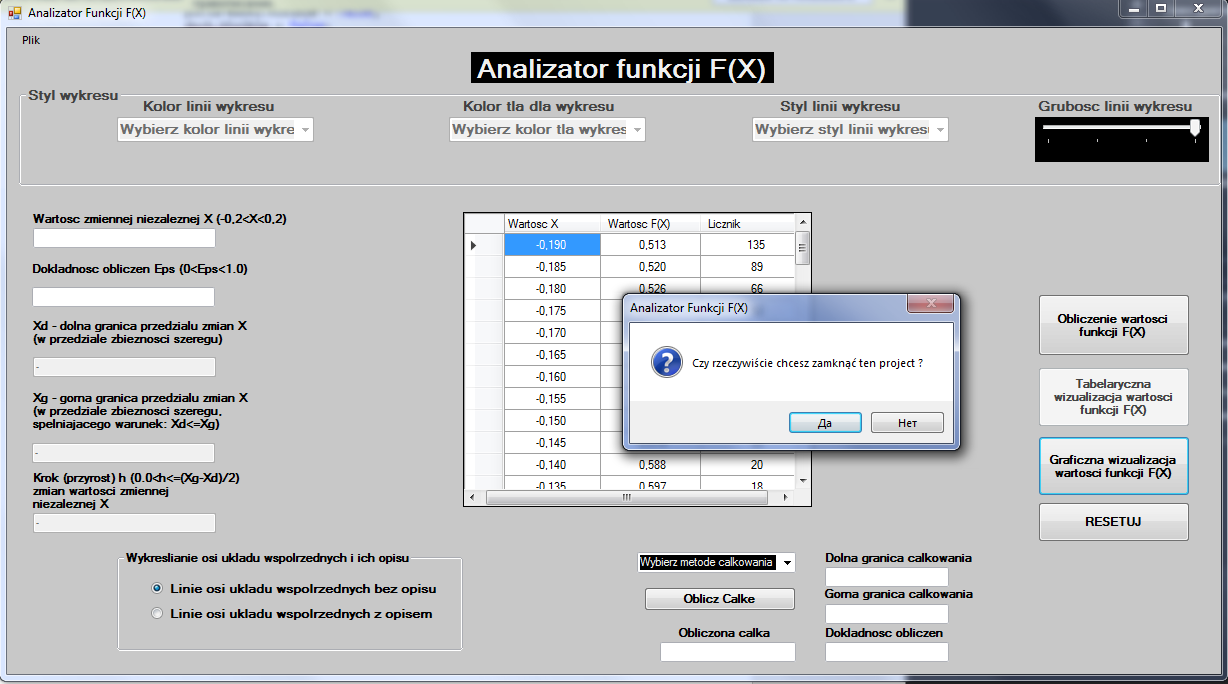
dstbH.Text = "-";

}

}

}

* Obsługa przycisku Exit, oraz zamknięcie aplikacji nacisnięciem czerwonego krzyżyka:



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

{

DialogResult Pytanie = MessageBox.Show("Czy rzeczywiście chcesz zamknąć ten project ?",

this.Text, MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button3);

switch (Pytanie)

{

case DialogResult.Yes:

MessageBox.Show("Teraz nastąpi zamknięcie projectu");

this.Close();

break;

case DialogResult.No:

break;

}

}

private void Zamkniecie\_FormClosing(object sender, FormClosingEventArgs e)

{

DialogResult Pytanie = MessageBox.Show("Czy rzeczywiście chcesz zamknąć ten project ?",

this.Text, MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button3);

switch (Pytanie)

{

case DialogResult.Yes:

break;

case DialogResult.No:

e.Cancel = true;

break;

case DialogResult.Cancel:

e.Cancel = true;

break;

}

}

SAMOOCENA

Sądzę, że za Project №3 z przedmiotu «Wprowadzenie do programowania», powinienem otrzymać ocenę - 5, również jak i za Projecty №1, №2.

Dziękuję

Project z przedmiotu «Wprowadzenie do programowania» został wykonany przez studenta, pierwszego semestru, na kierunku ‘Informatyka’,

Savchenka Denysa

nrAlbumu : 51908

Uczelnia: Akademia finansów i biznesu Vistula;

Rok : 2019;