Державний вищий навчальний заклад

«Прикарпатський національний університет імені Василя Стефаника» Кафедра комп’ютерних наук та інформаційних систем

**ЛАБОРАТОРНА РОБОТА №3**

з предмету «Програмування С#»

Тема: «Вивчення абстрактних класів»

Виконав:

студент групи КН-2

Гриньків В.І.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2020р.

к.т.н., доц. Ровінський В.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2020р.

Івано-Франківськ – 2020

**Мета:** Написати класи LineInterpolation та HermiteInterpolation, які реалізують абстрактний клас СInterpolation і здійснюють лінійну та ермітову інтерполяцію масиву даних.

**Код програми:**

using System;

using System.Windows.Forms;

namespace Laboratorna\_3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

Hermit h = new Hermit();

LineInterpoliation k = new LineInterpoliation();

}

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

{

Hermit h = new Hermit();

LineInterpoliation k = new LineInterpoliation();

}

private void button1\_Click(object sender, EventArgs e)

{

Cinterpoliation.PeriodPoints = int.Parse(textBox1.Text);

Cinterpoliation.StepI = int.Parse(textBox2.Text);

Cinterpoliation.NumberOP = int.Parse(textBox3.Text);

Cinterpoliation.length = Cinterpoliation.PeriodPoints \* Cinterpoliation.NumberOP;

chart1.Series[0].Points.Clear();

chart1.Series[1].Points.Clear();

chart1.Series[2].Points.Clear();

vyvid1();

vyvid2();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{}

private void textBox2\_TextChanged(object sender, EventArgs e)

{}

private void textBox3\_TextChanged(object sender, EventArgs e)

{ }

public void vyvid1()

{

chart1.Series[0].Points.Clear();

if (checkBox1.CheckState == CheckState.Checked)

{

for (int i = 0; i < Cinterpoliation.length + 1; i++)

{

chart1.Series[0].Points.AddXY(Cinterpoliation.h2[i], Cinterpoliation.h1[i]);

}

}

else

{

chart1.Series[1].Points.Clear();

}

}

public void vyvid2()

{

chart1.Series[1].Points.Clear();

LineInterpoliation A = new LineInterpoliation(Cinterpoliation.PeriodPoints, Cinterpoliation.StepI, Cinterpoliation.NumberOP);

if (checkBox2.CheckState == CheckState.Checked)

{

for (int i = 0; i < LineInterpoliation.length; i++ )

{

chart1.Series[1].Points.AddXY(LineInterpoliation.h3[i], LineInterpoliation.h4[i]);

}

}

else

{

chart1.Series[1].Points.Clear();

}

}

public void vyvid3()

{

chart1.Series[2].Points.Clear();

Hermit A = new Hermit(Hermit.PeriodPoints, Hermit.StepI, Hermit.NumberOP);

if (checkBox3.CheckState == CheckState.Checked)

{

double l = 0;

for (double i = 1, x = 0; i < Cinterpoliation.length+1;i++, x =0)

{

for (int h = 0; h < Cinterpoliation.StepI; h++, x += 1.0 / Cinterpoliation.StepI, l += (2 \* Math.PI) / ((Cinterpoliation.StepI) \* Cinterpoliation.length))

{

chart1.Series[2].Points.AddXY(l, Hermit.h4[h]);

}

}

}

else

{

chart1.Series[2].Points.Clear();

}

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

vyvid1();

}

private void checkBox2\_CheckedChanged(object sender, EventArgs e)

{

vyvid2();

}

private void checkBox3\_CheckedChanged(object sender, EventArgs e)

{

vyvid3();

}

}

public abstract class Cinterpoliation

{

public static int PeriodPoints=10;

public static int StepI=10;

public static int NumberOP=4;

public static int length = PeriodPoints \* NumberOP ;

public static double start\_num;

public static double[] h1 = new double[PeriodPoints\*1000];

public static double[] h2 = new double[PeriodPoints\*1000];

public static double[] h3 = new double[StepI\*1000];

public static double[] h4 = new double[StepI\*1000];

public Cinterpoliation() { defaul(10, 10, 4); }

public Cinterpoliation(int PP, int SI, int NO)

{

defaul(PP, SI, NO);

}

public virtual void defaul(int PP, int SI, int NO)

{

PeriodPoints = PP;

StepI = SI;

NumberOP = NO;

double x = 0;

for (int i = 0; i < length; i++, x += (2 \* Math.PI) / PeriodPoints)

{

h2[i] = x;

h1[i] = Math.Sin(h2[i]);

}

}

public abstract void Interpoliation();

}

public class LineInterpoliation : Cinterpoliation

{

public static int Number;

public LineInterpoliation() : base()

{

defaull();

}

public LineInterpoliation(int PP, int SI, int NO) : base(PeriodPoints, StepI, NumberOP)

{

defaull();

Interpoliation();

}

public virtual void defaull()

{

}

public override void Interpoliation()

{

double a, b, k;

a = 0;

b = StepI ;

k = StepI + 10;

double m = (b - a) / k;

double x = 0;

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

{

for (int i = 0; a < b; a += m, x += ((2 \* Math.PI) / Cinterpoliation.PeriodPoints) )

{

h3[i] = x;

h4[i] = Math.Sin(a);

}

a = a + 1;

}

}

}

public class Hermit : Cinterpoliation

{

public static double[] ar = new double[length+5];

public Hermit() : base()

{

Array.Resize(ref ar, length + 5);

}

public Hermit(int PP, int SI, int NO) : base(PeriodPoints, StepI, NumberOP)

{

Array.Resize(ref ar, length + 5);

}

public override void Interpoliation()

{

double j = 0;

double x = 0;

ar[0] = 0;

for(int p = 1; p<Hermit.length+1; p++)

{

ar[p] = Math.Sin(p - 1);

}

int i = Convert.ToInt32(j);

double c0 = Hermit.ar[i];

double c1 = Hermit.ar[i + 1] - 1 / 3.0 \* Hermit.ar[i - 1] - 1 / 2.0 \* Hermit.ar[i] - 1 / 6.0 \* Hermit.ar[i + 2];

double c2 = 1 / 2.0 \* (Hermit.ar[i - 1] + Hermit.ar[i + 1]) - Hermit.ar[i];

double c3 = 1 / 6.0 \* (Hermit.ar[i + 2] - Hermit.ar[i - 1]) + 1 / 2.0 \* (Hermit.ar[i] - Hermit.ar[i + 1]);

for (int w = 0; w < length + 1; w++)

h4[i] = ((((c3 \* x + c2)) \* x) + c1) \* x + c0;

}

}

}

**Результат виконання:**





