ЗАДАНИЕ 4

Създайте графически интерфейс и подходящ код за:

1. приближение на данни чрез интерполация по Лагранж

2. Изчисляване на определен интеграл по метода на Симпсън.

3. Решаване на нелинейно уравнение по методите на Нютон или на хордите.

4. Решаване на системи от обикновени диференциални уравнения от 1 ред.

Където е нужно покажете графика на решенията

За целта ще използвайте компонент **TStringGrid** за входните данни, **TChart** за графиката

Тук ще даден примерен интерфейс и код за интерполация по метода на Лагранж

1. Подготовка на формата (Дизайн)

Поставете следните компоненти върху `Form1`:

- **TStringGrid** (от таб *Additional*): Настройте `ColCount = 2`, `RowCount = 6`. Това ще е таблицата за x и y.
- **TChart** (от таб *Additional*): Кликнете два пъти върху него, изберете **Add** -> **Line Series**.
- **TButton** (от таб *Standard*): Променете `Caption` на "Изчертай".
- **TEdit**: За броя на точките (опционално).

  В C++ Builder трябва да се създаде функция, която приема масиви от данни и стойност `x`, за която търсим `y`.

Примерен програмен код

В `Unit1.cpp` добавете следния код за бутона:

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"

// --- Помощна функция за Лагранж ---
double Lagrange(double x, double *X_arr, double *Y_arr, int n)
```

```
{
   double result = 0;
   for (int i = 0; i < n; i++) {
      double p = Y_arr[i];
      for (int j = 0; j < n; j++) {
         if (i != j) {
            p = p * (x - X_arr[j]) / (X_arr[i] - X_arr[j]);
         }
      }
      result += p;
   }
   return result;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button1Click(TObject *Sender)
{
   int n = StringGrid1->RowCount - 1; // Брой точки
   double *x_vals = new double[n];
   double *y_vals = new double[n];

   // 1. Четене на данни от таблицата
   for (int i = 0; i < n; i++) {
      x_vals[i] = StringGrid1->Cells[0][i+1].ToDouble();
      y_vals[i] = StringGrid1->Cells[1][i+1].ToDouble();
   }

   // 2. Подготовка на графиката
   Series1->Clear(); // Изчистване на старата графика

   // Настройка на "звездичките" за реалните точки
   Series1->Pointer->Visible = true;
   Series1->Pointer->Style = psStar;
   Series1->Pointer->InflateMargins = true;

   // 3. Изчертаване на полинома (през малки стъпки)
   double minX = x_vals[0], maxX = x_vals[n-1];
   // Намираме мин и макс за обхвата на графиката
   for(int i=0; i<n; i++) {
      if(x_vals[i] < minX) minX = x_vals[i];
      if(x_vals[i] > maxX) maxX = x_vals[i];
   }

   double step = (maxX - minX) / 100.0; // 100 точки за плавна линия
   for (double x = minX; x <= maxX; x += step) {
      double y = Lagrange(x, x_vals, y_vals, n);
      Series1->AddXY(x, y, "", clRed);
   }

   // Добавяне на оригиналните точки, за да сме сигурни, че са маркирани
   for(int i = 0; i < n; i++) {
      Series2->AddXY(x_vals[i], y_vals[i], "", clBlue);
   }
```

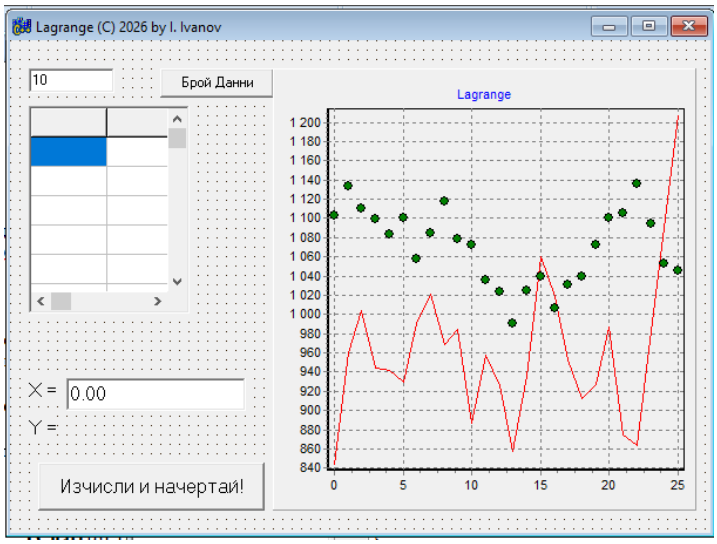
```
  xx=Edit2->Text.ToDouble();
  yy=Lagrange(xx, x_vals, y_vals, n);          //пресмята Лагранж за избраната точка X
  Label2->Caption="Y = "+FloatToStr(yy);

   delete[] x_vals;
   delete[] y_vals;
}
```

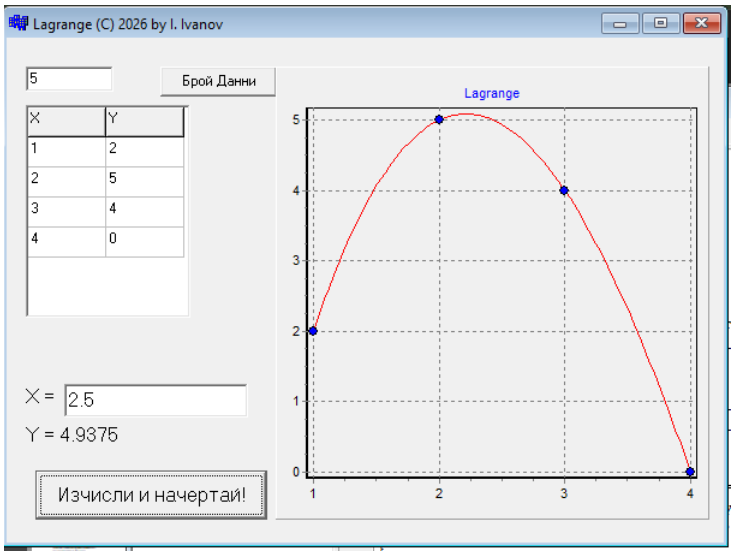
Обяснение на детайлите:

1. **Динамични масиви:** Използваме `new double[n]`, за да заделим памет спрямо броя редове в таблицата. Не забравяйте `delete[]` накрая, за да няма теч на памет.
2. **Series1->Pointer:** В компонента `TChart`, всяка серия (`Series`) има под-обект `Pointer`. Когато `Visible = true` и `Style = psStar`, всяка изчертана точка ще бъде отбелязана със звезда.
3. **В Series2** са входните данни. Дава се Series2->Format->Line->Border->Unvisible
4. **Точност:** Тъй като Лагранж преминава точно през зададените точки, в кода добавяме 100 междинни точки, за да изглежда графиката като крива линия, а не като начупена.
5. **StringGrid1:** Първият ред (индекс 0) обикновено е за заглавия, затова започваме четенето от `i+1`.

Примерен дизайн на формата и контролите:



Ако сте работили вярно се получава:


Lagrange (C) 2026 by I. Ivanov
5
Брой Данни
X
Y
1
2
2
5
3
4
4
0
X = 2.5
Y = 4.9375
Изчисли и начертай!
Lagrange
5
4
3
2
1
0
1
2
3
4