Белгородский Государственный Технологический Университет им. В. Г. Шухова

Кафедра программного обеспечения вычислительной техники  
и автоматизированных систем

## Лабораторная работа №2 по теме: «Построение графиков функций»

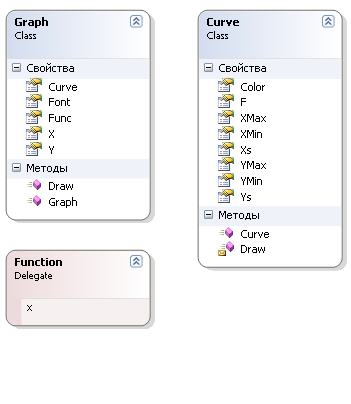
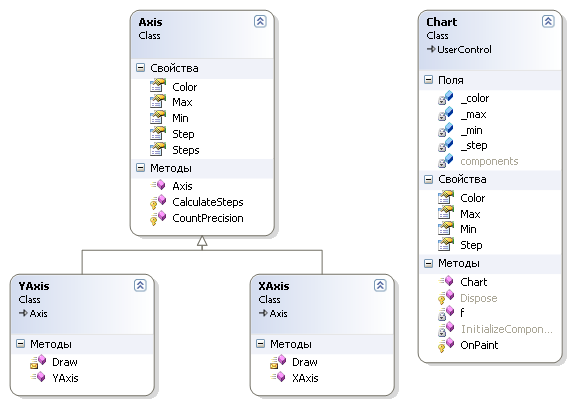
**Выполнил:**  
студент группы ПВ-41  
Адаменко И. И.

**Проверил:**  
старший преподаватель  
Осипов О. В.

Белгород  
2015

**Цель работы:** разработка модуля для построения графиков функций на декартовой плоскости в среде Microsoft Visual Studio.

# Схема структуры классов



# Код программы

1. **using** System;
2. **using** System.Collections.Generic;
3. **using** System.Drawing;
4. **using** System.Linq;
5. **using** System.Windows.Forms;
7. **namespace** Chart
8. {
9. **public** partial **class** Chart : UserControl
10. {
11. **private** **float** \_min = -9;
12. **private** **float** \_max = 11;
13. **private** **int** \_step = 50;
14. **private** Color \_color = Color.Black;
16. **public** Chart()
17. {
18. InitializeComponent();
20. **this**.SetStyle(
21. // вкл. двойную буферизацию == боремся с мерцанием
22. ControlStyles.DoubleBuffer |
23. // контрол отрисовывает себя сам, вместо ОС
24. ControlStyles.UserPaint |
25. // игнорирование события WM\_ERASEBKGND для уменьшения мерцания
26. ControlStyles.AllPaintingInWmPaint |
27. // перерисовывание при ресайзе окна
28. ControlStyles.ResizeRedraw,
29. **true**
30. );
31. **this**.UpdateStyles();
32. }
34. **protected** **override** **void** OnPaint(PaintEventArgs e)
35. {
36. **if** (**this**.DesignMode)
37. {
38. **base**.OnPaint(e);
39. **return**;
40. }
42. **base**.OnPaint(e);
44. var graph = **new** Graph();
46. graph.Func = f;
48. graph.X = **new** XAxis(Color.Black, Min, Max, Step);
50. graph.Y = **new** YAxis(Color.Black, Step);
52. graph.Curve = **new** Curve(**new** Function(f), Color);
54. graph.Draw(e.Graphics);
55. }
57. **private** **float** f(**float** x)
58. {
59. // x + sin(x)
60. **return** x + (**float**)Math.Sin(x);
61. }
63. **public** **float** Min
64. {
65. **get**
66. {
67. **return** \_min;
68. }
69. **set**
70. {
71. \_min = value;
72. }
73. }
75. **public** **float** Max
76. {
77. **get**
78. {
79. **return** \_max;
80. }
81. **set**
82. {
83. \_max = value;
84. }
85. }
87. **public** **int** Step
88. {
89. **get**
90. {
91. **return** \_step;
92. }
93. **set**
94. {
95. \_step = value;
96. }
97. }
99. **public** Color Color
100. {
101. **get**
102. {
103. **return** \_color;
104. }
105. **set**
106. {
107. \_color = value;
108. }
109. }
110. }
112. **public** **delegate** **float** Function(**float** x);
114. **public** **class** Graph
115. {
116. **public** XAxis X { **get**; **set**; }
117. **public** YAxis Y { **get**; **set**; }
118. **public** Curve Curve { **get**; **set**; }
119. **public** Font Font { **get**; **set**; }
120. **public** Function Func { **get**; **set**; }
122. **public** Graph()
123. {
124. }
126. **public** **void** Draw(Graphics g)
127. {
128. **int** axisWidth = 1;
129. **int** curveWidth = 2;
131. **if** (Font == **null**)
132. {
133. Font = **new** Font("Consolas", 10);
134. }
136. var format = **new** StringFormat();
137. format.Alignment = StringAlignment.Center;
138. format.LineAlignment = StringAlignment.Near;
140. SizeF stringMeasure = g.MeasureString("99.9999", Font);
142. var clip = g.VisibleClipBounds;
143. // отступ снизу для оси ординат
144. var paddingBottom = stringMeasure.Height + 4;
145. // отступ слева для оси абсцисс
146. var paddingLeft = clip.Left + stringMeasure.Width + 3;
148. var tmp = g.VisibleClipBounds;
149. tmp.Offset(paddingLeft, 0);
150. g.Clip = **new** Region(tmp);
152. X.Steps = **new** List<**float**>();
153. X.Draw(g, axisWidth, Font);
155. tmp = clip;
156. tmp.Offset(0, -paddingBottom);
157. g.Clip = **new** Region(tmp);
159. Curve.Xs = X.Steps;
160. Curve.XMin = X.Min;
161. Curve.XMax = X.Max;
163. var values = X.Steps;
164. values = values.Select(x => Func(x)).ToList();
166. Curve.Ys = values;
168. Y.Steps = **new** List<**float**>();
169. Y.Min = values.Min();
170. Y.Max = values.Max();
172. Y.Draw(g, axisWidth, Font);
174. tmp = clip;
175. tmp.Offset(paddingLeft, -paddingBottom);
177. g.Clip = **new** Region(tmp);
179. Curve.YMin = Y.Min;
180. Curve.YMax = Y.Max;
182. Curve.Draw(g, curveWidth, stringMeasure);
183. }
184. }
186. **public** **class** Curve
187. {
188. **public** Function F { **get**; **set**; }
189. **public** Color Color { **get**; **set**; }
190. **public** List<**float**> Xs { **get**; **set**; }
191. **public** List<**float**> Ys { **get**; **set**; }
192. **public** **float** XMin { **get**; **set**; }
193. **public** **float** XMax { **get**; **set**; }
194. **public** **float** YMin { **get**; **set**; }
195. **public** **float** YMax { **get**; **set**; }
197. **public** Curve(Function func, Color c)
198. {
199. F = func;
200. Color = c;
201. }
203. **internal** **void** Draw(Graphics g, **int** width, SizeF offsets)
204. {
205. var pen = **new** Pen(Color, width);
206. var clip = g.VisibleClipBounds;
208. var xReal = (XMax - XMin) / clip.Width;
209. var yReal = (YMax - YMin) / clip.Height;
211. var points = Xs.Zip(Ys, (x, y) => **new** PointF(
212. // clip.X -- отступ для оси ординат
213. (x - XMin) / xReal + clip.X,
214. (YMax - y) / yReal
215. ));
217. g.DrawCurve(pen, points.ToArray<PointF>());
218. }
219. }
221. **public** **class** Axis
222. {
223. **public** Color Color { **get**; **set**; }
224. **public** **float** Min { **get**; **set**; }
225. **public** **float** Max { **get**; **set**; }
226. **public** **int** Step { **get**; **set**; }
227. **public** List<**float**> Steps { **get**; **set**; }
229. **public** Axis(Color color, **float** min, **float** max, **int** h)
230. {
231. Color = color;
232. Min = min;
233. Max = max;
234. Step = h;
235. }
237. **protected** **uint** CountPrecision(**float** f)
238. {
239. **uint** zeros = 0;
240. var numStr = (f - Math.Truncate(f)).ToString();
241. **string** num;
243. **if** (numStr.Length > 2)
244. {
245. num = numStr.Substring(2);
246. }
247. **else**
248. {
249. num = **null**;
250. }
252. **while** (num != **null** && num[0] == '0')
253. {
254. zeros++;
255. num = num.Substring(1);
256. }
258. **return** zeros > 0 ? ++zeros : zeros;
259. }
261. **protected** **void** CalculateSteps(**ref** **float** k, **ref** **float** h)
262. {
263. **int** m = 1, n = 0, step;
264. **float** prev = k;
265. var ms = **new** **int**[] { 1, 2, 5 };
266. var found = **false**;
268. **if** (k > h)
269. {
270. step = -1;
271. ms = ms.Reverse().ToArray<**int**>();
273. n += step;
275. **while** (!found)
276. {
277. **float** new1 = k \* (**float**)Math.Pow((**double**)10, (**double**)n);
278. **float**[] arr = ms.Select(x => x \* new1).ToArray();
280. **for** (**int** i = 0; i < arr.Length; i++)
281. {
282. **if** (arr[i] < h)
283. {
284. found = **true**;
285. prev = arr[i];
286. m = ms[i];
288. **break**;
289. }
290. }
292. **if** (!found)
293. {
294. prev = arr[arr.Length - 1];
295. m = ms[arr.Length - 1];
296. n += step;
297. }
298. }
300. k = m \* (**float**)Math.Pow((**double**)10, (**double**)n);
301. }
302. **else**
303. {
304. step = 1;
306. **while** (!found)
307. {
308. **float** new1 = k \* (**float**)Math.Pow((**double**)10, (**double**)n);
309. **float**[] arr = ms.Select(x => x \* new1).ToArray();
311. **for** (**int** i = 0; i < arr.Length; i++)
312. {
313. **if** (arr[i] > h)
314. {
315. found = **true**;
317. **if** (i != 0)
318. {
319. prev = arr[i - 1];
320. m = ms[i - 1];
321. n++;
322. }
324. **break**;
325. }
326. }
328. **if** (!found)
329. {
330. prev = arr.Last();
331. m = ms.Last();
332. n += step;
333. }
334. }
336. k = m \* (**float**)Math.Pow((**double**)10, (**double**)(n - 1));
337. }
339. h = prev;
340. }
341. }
343. **public** **class** XAxis : Axis
344. {
345. **public** XAxis(Color color, **float** min, **float** max, **int** h)
346. : **base**(color, min, max, h)
347. {
349. }
351. **internal** **void** Draw(Graphics g, **int** width, Font font)
352. {
353. var clip = g.VisibleClipBounds;
354. var pen = **new** Pen(Color, width);
356. var format = **new** StringFormat();
357. format.Alignment = StringAlignment.Center;
358. format.LineAlignment = StringAlignment.Near;
359. SizeF stringMeasure = g.MeasureString("99.9999", font);
361. // 3 здесь отступ от текста до риски
362. var bottom = clip.Bottom - stringMeasure.Height - 3;
364. PointF pStart = **new** PointF(clip.Left, bottom),
365. pEnd = **new** PointF(clip.Right, bottom);
367. // отрисовываем саму ось
368. g.DrawLine(pen, pStart, pEnd);
370. **float** stepWorld = clip.Width / (Max - Min),
371. stepReal = Step;
373. CalculateSteps(**ref** stepWorld, **ref** stepReal);
375. Steps.Add(Min);
376. **float** tmpWorld = (**float**)Math.Round(Min / stepWorld) \* stepWorld,
377. // здесь -1, т. к. иначе последняя риска не влезает
378. tmpReal = clip.Left + stepReal - 1,
379. stepHalfSize = 5,
380. stepWidth = 1;
382. **if** (tmpWorld < Min)
383. {
384. Steps.Add(tmpWorld);
385. tmpWorld += stepWorld;
386. }
388. // просто какая-то магия
389. tmpReal -= (1 - (tmpWorld - Min) / stepWorld) \* stepReal;
391. **while** (tmpWorld <= Max)
392. {
393. Steps.Add(tmpWorld);
395. PointF pStepTop = **new** PointF(tmpReal,
396. bottom - stepHalfSize),
397. pStepBottom = **new** PointF(tmpReal,
398. bottom + stepHalfSize),
399. pGridTop = **new** PointF(tmpReal, 0);
401. Pen penStep = **new** Pen(Color, stepWidth),
402. penGrid = **new** Pen(ControlPaint.LightLight(Color),
403. stepWidth);
405. g.DrawLine(penGrid, pGridTop, pStepTop);
406. g.DrawLine(penStep, pStepTop, pStepBottom);
408. var num = tmpWorld.ToString("G" + CountPrecision(stepWorld));
409. g.DrawString(num,
410. font,
411. **new** SolidBrush(ControlPaint.Light(Color)),
412. pStepBottom,
413. format);
415. tmpWorld += stepWorld;
416. tmpReal += stepReal;
417. }
419. Steps.Add(Max);
420. }
421. }
423. **public** **class** YAxis : Axis
424. {
425. **public** YAxis(Color color, **int** h, **float** min = 0, **float** max = 0)
426. : **base**(color, min, max, h)
427. {
429. }
431. **internal** **void** Draw(Graphics g, **int** width, Font font)
432. {
433. var clip = g.VisibleClipBounds;
434. var pen = **new** Pen(Color, width);
436. var format = **new** StringFormat();
437. format.Alignment = StringAlignment.Far;
438. format.LineAlignment = StringAlignment.Center;
439. SizeF stringMeasure = g.MeasureString("99.9999", font);
441. // 3 здесь отступ от текста до риски
442. var left = clip.Left + stringMeasure.Width + 3;
444. PointF pStart = **new** PointF(left, 0),
445. pEnd = **new** PointF(left, clip.Bottom);
447. g.DrawLine(pen, pStart, pEnd);
449. **float** stepWorld = clip.Height / (Max - Min),
450. stepReal = Step;
452. CalculateSteps(**ref** stepWorld, **ref** stepReal);
454. Steps.Add(Min);
455. **float** tmpWorld = (**float**)Math.Round(Min / stepWorld) \* stepWorld,
456. // здесь -1, т. к. иначе последняя риска не влезает
457. tmpReal = clip.Top + stepReal - 1,
458. stepHalfSize = 5,
459. stepWidth = 1;
461. **if** (tmpWorld < Min) tmpWorld += stepWorld;
463. Steps.Add(tmpWorld);
464. tmpWorld += stepWorld;
466. var bottom = clip.Bottom - tmpReal;
468. **while** (tmpWorld <= Max)
469. {
470. Steps.Add(tmpWorld);
472. PointF pStepLeft = **new** PointF(left - stepHalfSize,
473. bottom),
474. pStepRight = **new** PointF(left + stepHalfSize,
475. bottom),
476. pGridRight = **new** PointF(clip.Right, bottom);
478. Pen penStep = **new** Pen(Color, stepWidth),
479. penGrid = **new** Pen(ControlPaint.LightLight(Color),
480. stepWidth);
482. g.DrawLine(penGrid, pStepRight, pGridRight);
483. g.DrawLine(penStep, pStepLeft, pStepRight);
485. var num = tmpWorld.ToString("G" + CountPrecision(stepWorld));
486. g.DrawString(num,
487. font,
488. **new** SolidBrush(ControlPaint.Light(Color)),
489. pStepLeft,
490. format);
492. tmpWorld += stepWorld;
493. tmpReal += stepReal;
494. bottom = clip.Bottom - tmpReal;
495. }
497. Steps.Add(Max);
498. }
499. }
500. }

# Примеры работы программы

