Практическая работа №16_2

Тема: Разработка приложений с графикой

Цель работы: изучение приемов создания приложений Windows Forms с использованием графических методов, построения графиков с помощью компонента отображения графической информации Chart

Задачи:

– изучение приемов создания приложений Windows Forms с графическими объектами;

– изучить возможности построения графиков с помощью компонента отображения графической информации Chart

– создание проектов с использованием графических методов.

Материально-техническое обеспечение:

Место проведения: Компьютерный класс.

Время на выполнение работы: 2 часа.

Оборудование: ПК

Средства обучения: операционная система, текстовый процессор MS Word, программные средства определенного вида

Исходные данные:

1. Конспект занятия.
2. Задание для практической работы.

Перечень справочной литературы:

1) Программирование на языке высокого уровня. Программирование на языке С++: учеб. пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев ; под ред. Л. Г. Гагариной. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 512 с. – (Среднее профессиональное образование).

Краткие теоретические сведения:

Класс Control определяет совокупность событий элементов управления. Для каждого компонента класса определен свой набор стандартных событий, на которые он может реагировать. К числу **типовых событий** относят:

Activated- получение формой фокуса ввода;

Click - одинарный щелчок мышкой по элементу управления. В некоторых случаях это событие происходит также и в других ситуациях, например при нажатии пользователем программы клавиши Enter;

DoubleClick - двойной щелчок мышки по элементу управления. Обработка события Click для некоторых элементов управления (например, для Button) полностью исключает возможность события DoubleClick;

MouseDown - нажатие кнопки мышки над элементом управления. Это событие происходит перед отпусканием кнопки мышки;

MouseUp - отпускание кнопки мышки над *элементом управления*;

MouseMove- перемещение мыши. Происходит непрерывно при *перемещении мышки*;

DragDrop – завершение операции *перетаскивания объекта управления* и освобождения кнопки мышки пользователем. Кроме этого события есть другие аналогичные – при размещении объекта внутри границ другого элемента, покидании границ другого элемента;

Closed - закрытие *формы*;

Load - загрузка *формы.* Происходит до первоначального отображения формы.

Событие Paint

Событие **Paint** возникает, если окно становится «грязным» (dirty) – то есть, если изменяется его размер, если оно перестает закрывать (частично или полностью) другое окно или если оно было свернуто а потом развернуто. Во всех этих случаях – то есть если форму необходимо перерисовать, платформа .NET автоматически вызовет событие Paint.

Метод **Invalidate()** имеет несколько перегруженных вариантов. Например, если нужно обновить заданный прямоугольник, то нужно создать такой код:

```
Rectangle r = new Rectangle(0, 0, 50, 50);
Invalidate(r);
```

Ход работы:

Требования к содержанию отчета:

- Номер и название практической работы.
- Цель работы.
- По каждой заданию (задаче/примеру) экранные формы (при наличии) и листинг программного кода, показывающие порядок выполнения практической работы, и результаты, полученные в ходе её выполнения.
- Ответы на контрольные вопросы в тетради.

Порядок выполнения работы:

Все проекты практической работы размещать в своей сетевой в новой папке **Пр16_2_ФИО**

В начале каждого файла проекта установить комментарии: пр.р.№_____ (указать номер), свою Фамилию. Формулировку задания

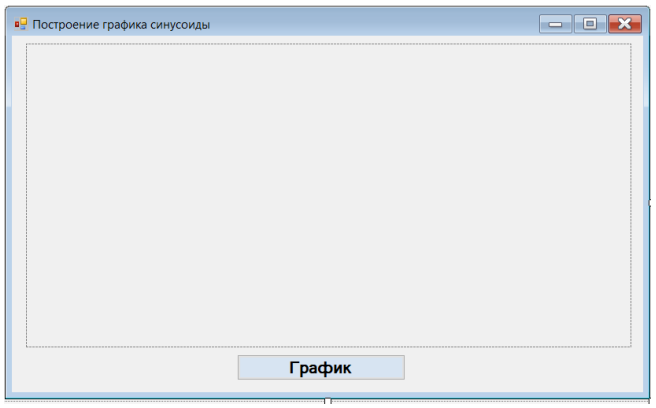
Задание 1. Построение графиков функций с использованием Graphics

Рассмотрим еще один способ построения графика синусоиды по точкам:

✓ Создайте новый проект **pr16_2.1_Фамилия** типа Windows Forms.

✓ Разместите на форме размером **820*500** контейнер **Panel** и кнопку **Button**:

✓ Зарегистрируйте событие кнопки и опишите методы построения:




```
private void button1_Click(object sender, EventArgs e)
{
    GraphicsFunk(panel1, (float)(Math.PI * 2), (0));
}

ссылка: 1
void GraphicsFunk(Panel panel1, float Diapason, int tip)
{
    GraphicSetting setting = new GraphicSetting(panel1);
    Graphics grapnics = Graphics.FromHwnd(panel1.Handle);
    if (tip == 0)
        grapnics.Clear(Color.White);
    else
        grapnics.Clear(Color.Violet);

    float step = Diapason / setting.width;
    float dispstep = setting.width / Diapason;
    Pen pen = new Pen(Brushes.Silver, 0.01F);

    //вертильные линии - шкала деления по X
    for (int f = 0; f <= panel1.Width; f += 5)
    {
        grapnics.DrawLine(pen, f, panel1.Height, f, 0);
    }

    // вычисление значений и построение синусоиды по точкам
    for (float f = 0; f <= Diapason; f += 0.001F)
    {
        float fun = 0;

        fun = (float)Math.Sin(f) * 150;

        float amp = setting.center + fun;
        float ds = dispstep * f;
        grapnics.DrawLine(Pens.Blue, ds, amp, ds, amp - 2);
    }
}
Ссылок: 3
class GraphicSetting
{
    public float center;
    public float width;
    public float heigth;
    ссылка: 1
    public GraphicSetting(Panel panel)
    {
        center = ((float)(panel.ClientRectangle.Height)) / 2;
        width = (float)panel.ClientRectangle.Width;
        heigth = (float)panel.ClientRectangle.Height;
    }
}
```

✓ Протестируем программу. При необходимости откорректируйте код. Установите поясняющие

комментарии в программном коде.

Задание 2. Построение графика функции двух переменных ***z = f(x, y)***

Условие задачи

Задана формула функции двух переменных ***z = sin(x) + cos(y).*** Разработать приложение, которое рисует график этой функции в отдельной форме.

Дополнительно реализовать поворот графика влево, вправо, вверх, вниз. Также нужно выводить оси ***OX, OY, OZ.***

В качестве примера, выбрана функция

$$z = sin(x) + cos(y)$$

Используя данный пример, можно создавать собственные программы для построения графиков других функций. По желанию можно модернизировать работу программы по своему усмотрению.

Математическая постановка задачи

Построение графика функции двух переменных есть математически решаемой задачей, в которой используются известные формулы вычисления.

График функции двух переменных *z*(*x*,*y*) строится в параллелепипеде с размерами (xx1, xx2), (yy1, yy2), (zz1, zz2).

Для использования поворота системы в 3-мерном пространстве возникает понятие точки (x0, y0, z0), относительно которой происходит поворот системы координат.

Также возникает понятие углов:

- α (альфа) – поворот системы относительно оси *OZ*;
- β (бета) – поворот системы относительно оси *OX*.

Сдвиг в точку (x0, y0, z0) с учетом поворота на углы α и β описывается известными соотношениями

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{pmatrix}$$

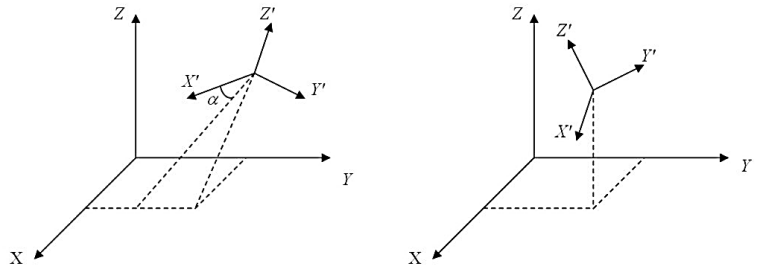
После перемножения матриц получаем формулу для вычисления:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{pmatrix} \times \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$x = (x - x_0) \cdot \cos\alpha - (y - y_0)$$
$$y = ((x - x_0) \cdot \sin\alpha + (y - y_0) \cdot \cos\alpha) \cdot \cos\alpha - (z - z_0) \cdot \sin\alpha$$
$$z = ((x - x_0) \cdot \sin\alpha + (y - y_0) \cdot \cos\alpha) \cdot \sin\alpha + (z - z_0) \cdot \cos\alpha$$

По этой формуле будет происходить преобразование системы координат и масштабирование (рисунок 1).

Рис. 1. Сдвиг и поворот системы координат



Необходимо определиться, в какой плоскости монитора будут лежать оси координат *OX*, *OY*, *OZ*. Принимаем, что в плоскости монитора лежат оси *OX* и *OY*. А ось *OZ* перпендикулярна экрану.

Координаты расчетной точки (x, y) прижимаются к точке (0, 0) по формулам:

$$x_n = \frac{x}{\frac{z}{A} + 1} \qquad y_n = \frac{y}{\frac{z}{a} + 1}$$

где *A*, *a* – коэффициенты перспективы, которые подбираются экспериментально в зависимости от функции.

Выполнение задания:

1) Создайте новый проект **pr16_2.2_Фамилия** типа Windows Forms.
2) Создайте форму размером **500*250** по образцу и разместите на форме кнопку **Button**.

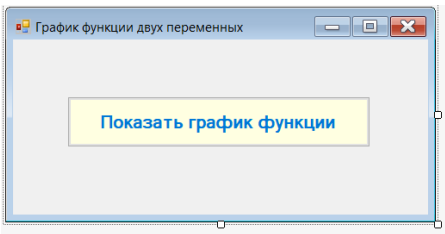


Рис. 2. Вид основной формы программы

✓ Настроить следующие свойства компонент и формы:

- в форме **Form1** свойство **Text** = **График функции двух переменных**;

- в форме **Form1** свойство **MaximizeBox** = **False**;
- в форме **Form1** свойство **StartPosition** = **CenterScreen**;
- в компоненте **button1** свойство **Text** = Показать график функции …

3) Добавьте в проект новую форму **Form2** размером **630*430**.

✓ Разместить на форме четыре компонента типа **Button**. Автоматически создается четыре объекта с именами **button1, button2, button3, button4.**

✓ Настроить свойства компонент и формы следующим образом:

- в форме **Form2** свойство **StartPosition** = **CenterScreen**;
- в форме **Form2** свойство **Text** = **График функции z = f(x,y)**;
- в компоненте **button1** свойство **Text** = ^;
- в компоненте **button2** свойство **Text** = **v**;
- в компоненте **button3** свойство **Text** = <;
- в компоненте **button4** свойство **Text** = >.

Приблизительный вид формы **Form2** изображен на рисунке 3.

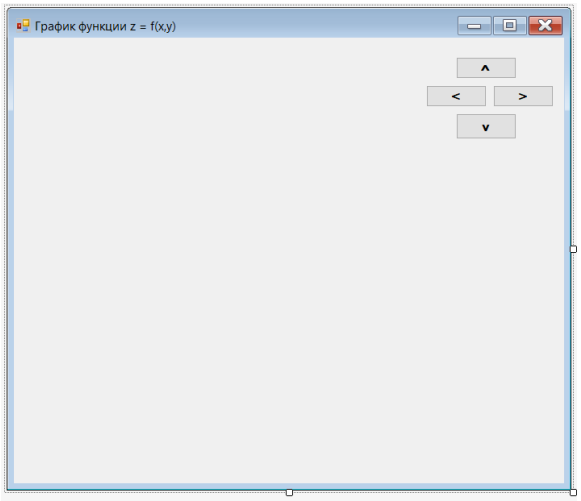


Рис. 3. Форма Form2 приложения

4) Ввод внутренних переменных в форму **Form2**.

Все внутренние переменные, использующиеся для организации вывода графика, размещаются в классе формы **Form2**. Поэтому, сначала надо активизировать модуль «**Form2**».

В модуль формы Form2 вводятся следующие внутренние переменные с классом видимости **private**:

- **xx1, xx2, yy1, yy2** – соответствуют координатам точек, которые отображаются на экране монитора;
- массивы ***xx*** и ***yy*** предназначены для вывода плоскости из 4-х точек. Область определения функции z = f(x, y) разбивается на прямоугольники, на любом из которых функция экстраполируется ребрами четырехугольника.

В разделе **public** вводятся:

- переменные **X_min, Y_min, X_max, Y_max** вещественного типа, которые представляют реальные координаты параллелепипеда, в котором выводится график функции. Эти переменные заполняются из основной формы Form1 экспериментальным путем:
- переменные **alfa, beta** вещественного типа, которые отображают углы наблюдения за графиком функции. Заполняются из главной формы **Form1**;
- переменные **x0, y0, z0** вещественного типа. Отображают величины из главной формулы вычисления (см. математическую постановку задачи);
- переменная A вещественного типа. Представляет коэффициент перспективы и подбирается экспериментально;
- переменная **f_show** логического типа используется для указания того, что нужно перерисовать график, в случае изменения положения углов alfa и beta.

После введения переменных в текст программы, фрагмент класса формы **Form2** имеет вид:

Переменные, имеющие идентификатор доступа public, заполняются из формы Form1.

```
public partial class Form2 : Form
{
    private int xx1, xx2, yy1, yy2;
    private int[] xx = new int[4];
    private int[] yy = new int[4];
    public int left;
    public int top;
    public int width;
    public int height;
    public double X_min, Y_min, X_max, Y_max;
    public double alfa, beta;


    public double x0, y0, z0;


    public double A;
    public bool f_show;

    ссылка: 1
    public Form2()
    {
        InitializeComponent();
    }
```

5) Программирование внутренних методов в форме **Form2**.
В текст класса **Form2** вводятся три дополнительных метода:

- функция преобразования системы координат и масштабирования **Zoom_XY();**
- функция **func()** для которой выводится график;
- функция рисования графика **Show_Graphic().**

Листинг метода преобразования системы координат следующий:

```
private void Zoom_XY(double x, double y, double z, out int xx, out int yy)
{
    double xn, yn;
    double tx, ty, tz;

    tx = (x - x0) * Math.Cos(alfa) - (y - y0) * Math.Sin(alfa);
    ty = ((x - x0) * Math.Sin(alfa) + (y - y0) * Math.Cos(alfa)) * Math.Cos(beta) - (z - z0) * Math.Sin(beta);
    tz = ((x - x0) * Math.Sin(alfa) + (y - y0) * Math.Cos(alfa)) * Math.Sin(beta) + (z - z0) * Math.Cos(beta);

    xn = tx / (tz / A + 1);
    yn = ty / (ty / A + 1);

    xx = (int)(width * (xn - X_min) / (X_max - X_min));
    yy = (int)(height * (yn - Y_max) / (Y_min - Y_max));
}
```

Листинг метода func() следующий.

```
private double func(double x, double y)
{
    double res;
    res = Math.Sin(x) + Math.Cos(y);
    return res;
}
```

В этом методе вместо строки
res = Math.Sin(x) + Math.Cos(y);
можно сделать вставку собственной функции.
Непосредственный вывод графика функции реализован в методе Show_Graphic(). Листинг метода Show_Graphic() следующий.

```
private void Show_Graphic(PaintEventArgs e)
{
    const double h = 0.1;
    const double h0 = 0;
    int i, j;

    Rectangle r1 = new Rectangle(left, top, left + width, top + height);
    Pen p = new Pen(Color.Black);
    e.Graphics.DrawRectangle(p, r1);

    // Создать шрифт
    Font font = new Font("Courier New", 12, FontStyle.Bold);
    SolidBrush b = new SolidBrush(Color.Blue);

    // рисование осей
    // ось X
    Zoom_XY(0, 0, 0, out xx1, out yy1);
    Zoom_XY(1.2, 0, 0, out xx2, out yy2);
    e.Graphics.DrawLine(p, xx1, yy1, xx2, yy2);
    e.Graphics.DrawString("X", font, b, xx2 + 3, yy2);

    // ось Y
    Zoom_XY(0, 0, 0, out xx1, out yy1);
    Zoom_XY(0, 1.2, 0, out xx2, out yy2);
    e.Graphics.DrawLine(p, xx1, yy1, xx2, yy2);
    e.Graphics.DrawString("Y", font, b, xx2 + 3, yy2);

    // ось Z
    Zoom_XY(0, 0, 0, out xx1, out yy1);
    Zoom_XY(0, 0, 1.2, out xx2, out yy2);
    e.Graphics.DrawLine(p, xx1, yy1, xx2, yy2);
    e.Graphics.DrawString("Z", font, b, xx2 + 3, yy2 - 3);

    // рисование поверхности
    p.Color = Color.Red;
    p.Width = 1;

    for (j = 0; j <= 9; j++)
        for (i = 0; i <= 9; i++)
        {
            Zoom_XY(h0 + h * i, h0 + h * j, func(h0 + h * i, h0 + h * j), out xx[0], out yy[0]);
            Zoom_XY(h0 + h * i, h + h * j, func(h0 + h * i, h + h * j), out xx[1], out yy[1]);
            Zoom_XY(h + h * i, h + h * j, func(h + h * i, h + h * j), out xx[2], out yy[2]);
            Zoom_XY(h + h * i, h0 + h * j, func(h + h * i, h0 + h * j), out xx[3], out yy[3]);

            e.Graphics.DrawLine(p, xx[0], yy[0], xx[1], yy[1]);
            e.Graphics.DrawLine(p, xx[1], yy[1], xx[2], yy[2]);
            e.Graphics.DrawLine(p, xx[2], yy[2], xx[3], yy[3]);
            e.Graphics.DrawLine(p, xx[3], yy[3], xx[0], yy[0]);
        }
}
```

Пояснения к некоторым фрагментам кода в методе Show_Graphic().
Область определения функции z = f(x,y) разбивается на прямоугольники, на любом из которых функция экстраполируется с ребрами четырехугольника. Построение четырехугольников на экране реализуется с помощью метода DrawLine().
После очистки канвы происходит рисование осей координат и методом DrawLine() выводятся фрагменты поверхности.
При рисовании поверхности, из метода Show_Graphic() вызывается метод Zoom_XY(), что осуществляет преобразование и масштабирование из реальных координат в экранные координаты.

6) Программирование события **Paint** формы **Form2**.

Чтобы получить объект Graphics, нужно запрограммировать событие Paint формы Form2.
Обработчик события ***Form2_Paint()*** *получает два параметра. Первый параметр типа* ***System.Object****, второй параметр типа* ***PaintEventArgs****.*
Параметр типа ***PaintEventArgs*** *содержит объект* ***Graphics****, необходимый для рисования на поверхности формы.*
Листинг обработчика события Form2_Paint() следующий.

```
private void Form2_Paint(object sender, PaintEventArgs e)
{
    Show_Graphic(e);
}
```

7) Программирование обработчиков событий клика на кнопках **button1, button2, button3, button4**. Поворот графика происходит в момент, когда пользователь делает клик на одной из кнопок, размещенных на форме **Form2** (элементы управления button1, button2, button3, button4).
Отображение графика зависит от внутренних переменных ***alfa*** *и* ***beta****. Переменная* ***alfa*** *содержит угол поворота относительно оси* OZ. *Переменная* ***beta*** *содержит значение угла поворота вокруг оси* OX.
Поэтому, в обработчиках событий происходит изменение значений alfa и beta на некоторую величину. По желанию, можно установить собственную величину изменения alfa и beta.
Листинг обработчиков событий приведен ниже.

```
private void Form2_Paint(object sender, PaintEventArgs e)
{
    Show_Graphic(e);
}

ссылка: 1
private void button1_Click(object sender, EventArgs e)
{
    beta = beta + 0.1;
    Invalidate();
}
ссылка: 1
private void button2_Click(object sender, EventArgs e)
{
    alfa = alfa - 0.1;
    Invalidate();
}
 ссылка: 1
 private void button3_Click(object sender, EventArgs e)
{
    beta = beta - 0.1;
    Invalidate();
}
ссылка: 1
private void button4_Click(object sender, EventArgs e)
{
    alfa = alfa + 0.1;
    Invalidate();
}
```

В вышеприведенных обработчиках событий, событие Paint генерируется явно с помощью унаследованного метода Invalidate(). Этот метод делает перерисовывание всей клиентской области программным путем.

8) Программирование обработчиков событий **MouseDown, MouseMove и MouseUp**.

Для осуществления поворота графика с помощью мышки нужно запрограммировать соответствующие обработчики событий.

Если нажать клавишу мыши и удерживать ее нажатой над формой Form2, а потом отпустить, то генерируются такие события (рисунок 4):

- *MouseDown – генерируется, если пользователь делает клик мышкой на форме Form2;*
- *MouseMove – генерируется, если пользователь перемещает мышку над формой Form2 (независимо, нажата ли одна из кнопок мышки);*
- *MouseUp – генерируется, если пользователь отпускает кнопку мышки после нажатия.*

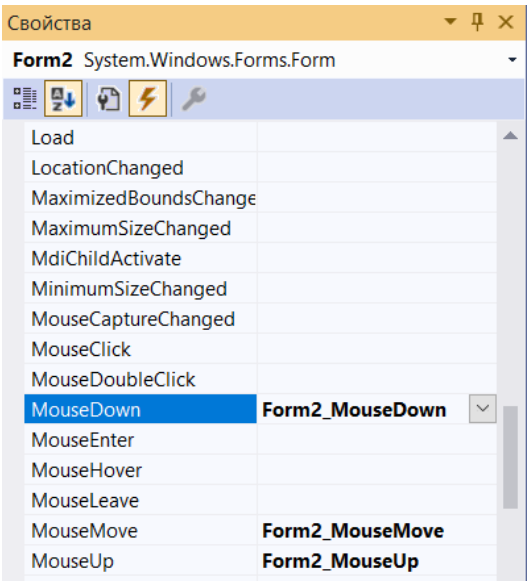


Рис. 4. События MouseDown, MouseMove, MouseUp

Запрограммируйте данные события (выбрав нужное событие в окне свойств, щелкните по нему дважды и напишите нужный программный код):

```
private void Form2_MouseDown(object sender, MouseEventArgs e)
{
    f_show = true;
}

ссылка: 1
private void Form2_MouseUp(object sender, MouseEventArgs e)
{
    f_show = false;
}

ссылка: 1
private void Form2_MouseMove(object sender, MouseEventArgs e)
{
    double a, b;
    if (f_show)
    {
        a = e.X - (int)(width / 2);
        b = e.Y - (int)(height / 2);
        if (a != 0)
            alfa = Math.Atan(b / a);
        else
            alfa = Math.PI / 2;
        beta = Math.Sqrt(Math.Pow(a / 10, 2) + Math.Pow(b / 10, 2));
        Invalidate();
    }
}
```

9) Программирование события клика на кнопке **button1** формы **Form1** (вызов формы рисования графика функции).

При клике на кнопке **button1** из формы **Form1** может выводиться график функции.

Обработчик события клика на кнопке Button1 имеет вид.

```
public partial class Form1 : Form
{
    ссылка: 1
    public Form1()
    {
        InitializeComponent();
    }

    ссылка: 1
    private void button1_Click(object sender, EventArgs e)
    {
        Form2 form2 = new Form2();

        // Прямоугольник, в котором будет выведен график функции
        form2.left = 20;
        form2.top = 20;
        form2.width = 300;
        form2.height = 300;

        form2.f_show = false;
        form2.x0 = 0;
        form2.y0 = 0;
        form2.z0 = 0;
        form2.A = -8;
        form2.alfa = 10;
        form2.beta = 12;
        form2.X_min = -3;
        form2.X_max = 3;
        form2.Y_min = -3;
        form2.Y_max = 3;
        form2.ShowDialog();
    }
}
```

10) Протестируем программу. При необходимости откорректируйте код. Установите поясняющие комментарии в программном коде.

Задания для самостоятельной работы (из пр.16_1)

Создать проект, содержащий форму. Расположить на форме 2 кнопки, при нажатии на которых отображаются **Герб** и **Флаг** государства. Флаг строить из прямоугольных элементов соответствующего цвета. Флаг и Герб отображаются на разных формах.

Номер задания определяется по номеру студента в списке группы.

1. Англия	2. Армения
3. Белоруссия	4. Болгария
5. Германия	6. Греция
7. Грузия	8. Дания
9. Индия	10. Испания
11. Италия	12. Казахстан
13. Китай	14. Латвия
15. Литва	16. Нидерланды
17. Норвегия	18. Польша
19. Россия	20. Румыния
21. Таджикистан	22. Туркменистан
23. Узбекистан	24. Украина
25. Финляндия	26. Франция
27. Чехия	28. Швейцария
29. Швеция	30. Япония

Контрольные вопросы:

1) Опишите технологию построения графика двух переменных.