Прикарпатський національний університет імені Василя Стефаника
Факультет математики та інформатики
Кафедра інформатики

Методичні вказівки
до виконання лабораторних робіт
з дисципліни
«Обчислювальна геометрія і комп’ютерна графіка»

Власій О.О., Превисокова Н.В.

ЗМІСТ

ЛАБОРАТОРНА РОБОТА №1
ПОБУДОВА ГРАФІЧНИХ ОБ'ЄКТІВ ЗАСОБАМИ DELPHI

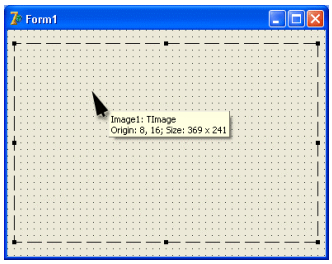
МЕТА: Ознайомитися з основними параметрами геометричних побудов. Оволодіти практичними навиками створення графічних об'єктів засобами Delphi

ТЕОРЕТИЧНІ ВІДОМОСТІ

Полотно

Поверхні, на яку програма може виводити графіку, відповідає властивість Canvas. У свою чергу, властивість Canvas — це об'єкт типу TCanvas. Методи цього типу забезпечують виведення графічних примітивів (точок, ліній, кіл, прямокутників і т. д.), а властивості дозволяють задати характеристики графічних примітивів, що виводяться: колір, товщину і стиль ліній; колір і вид заповнення областей; характеристики шрифту при виведенні текстової інформації.

Методи виведення графічних примітивів розглядають властивість Canvas як деяке абстрактне полотно, на якому вони можуть малювати (canvas переводиться як "поверхня", "полотно для малювання"). Полотно складається з окремих точок — пікселів. Положення пікселя характеризується його горизонтальною (X) і вертикальною (Y) координатами. Лівий верхній піксель має координати (0, 0). Координати зростають зверху вниз і зліва направо (мал. 1.1). Значення координат правої нижньої точки полотна залежать від розміру полотна.



Мал. 1.1. Координати точок полотна

Розмір полотна можна одержати, звернувшись до властивостей Height и Width області ілюстрації (image) або до властивостей форми: ClientHeight и ClientWidth.

Олівець

Олівець використовується для викреслювання точок, ліній, контурів геометричних фігур: прямокутників, кіл, еліпсів, дуг і ін. Вид лінії, яку залишає олівець на поверхні полотна, визначають властивості об'єкту Pen, які перераховані в табл. 1.1.

Таблиця 1.1. Властивості об'єкту Pen (Олівець)

Властивість	Визначає
Color	Колір лінії
Width	Товщину лінії
Style	Вид лінії
Mode	Режим відображення

Властивість Color задає колір лінії, що викреслюється олівцем. В табл. 1.2 перераховані іменовані константи (тип TColor), які можна використовувати як значення властивості Color.

Таблиця 1.2. Значення властивості Color визначає колір лінії

Константа	Колір	Константа	Колір
clBlack	Чорний	clSilver	Сріблястий
clMaroon	Каштановий	clRed	Червоний
clGreen	Зелений	clLime	Салатовий
clOlive	Оливковий	clBlue	Синій
clNavy	Темно-синій	clFuchsia	Яскраво-рожевий
clPurple	Рожевий	clAqua	Бірюзовий

clTeal	Зелено-голубий	clWhite	Білий
clGray	Сірий		

Властивість Width задає товщину лінії (в пікселях). Наприклад, інструкція Canvas. Pen. width: =2 встановлює товщину лінії в 2 пикселі.

Властивість style визначає вид (стиль) лінії, яка може бути неперервною або переривчастою, складається з штрихів різної довжини. В табл. 1.3 перераховані іменовані константи, що дозволяють задати стиль лінії. Товщина пунктирної лінії не може бути більше 1. Якщо значення властивості Pen.width більше за одиницю, то пунктирна лінія буде виведена як суцільна.

Таблиця 1.3. Значення властивості Pen.Type визначає вид лінії

Константа	Вид линії
psSolid	Суцільна лінія
psDash	Штрихова лінія, довгі штрихи
psDot	Штрихова лінія, короткі штрихи
psDashDot	Штрихова лінія, чергування довгого и короткого штрихов
psDashDotDot	Штрихова лінія, чергування одного довгого і двох коротких штрихів
psClear	Лінія не відображається (використовується, якщо не треба зображати межу області, наприклад, прямокутника)

Властивість Mode визначає, як формуватиметься колір точок лінії залежно від кольору точок полотна, через які ця лінія прокреслюється. По замовчуванню вся лінія викреслюється кольором, що визначений значенням властивості Pen.Color.

Проте програміст може задати інверсний колір лінії по відношенню до кольору фону. Це гарантує, що незалежно від кольору фону всі ділянки лінії будуть видимі, навіть в тому випадку, якщо колір лінії і колір фону співпадають.

В табл. 1.4 перераховані деякі константи, які можна використовувати як значення властивості Pen.Mode.

Таблиця 1.4. Значення властивості Pen. Mode впливає на колір лінії

Константа	Колір лінії
pmBlack	Чорний, не залежить від значення властивості Pen. Color
pmWhite	Білий, не залежить від значення властивості Pen. Color
pmCopy	Колір лінії визначається значенням властивості Pen . Color
pmNotCopy	Колір лінії є інверсним по відношенню до значення властивості Pen. Color
pmNot	Колір точки лінії визначається як інверсний по відношенню до кольору точки полотна, в яку виводиться точка лінії

Пензлик

Пензлик (canvas.Brush) використовується методами, що забезпечують викреслювання замкнутих областей, наприклад геометричних фігур, для заливки (замалювання) цих областей. Пензлик як об'єкт володіє двома властивостями, перерахованими в табл. 1.5.

Таблиця 1.5. Властивості об'єкту TBrush (пензлик)

Властивість	Визначає
Color	Колір замалювання замкнутої області
Style	Стиль (тип) заповнення області

Область усередині контура може бути замальована або заштрихована. В першому випадку область повністю перекриває фон, а в другому — крізь незаштриховані ділянки області буде видно фон.

Як значення властивості Color можна використовувати будь-яку з констант типу TColor (див. список констант для властивості Pen.color в табл. 1.2).

Константи, що дозволяють задати стиль заповнення області, наведені в табл. 1.6.

Таблиця 1.6. Значення властивості Brush. style визначають тип замалювання

Константа	Тип заповнення (заливки) області
bsSolid	Суцільна заливка
bsClear	Область не замальовується
bsHorizontal	Горизонтальне штрихування
bsVertical	Вертикальне штрихування
bsFDiagonal	Діагональне штрихування з нахилом ліній вперед
bsBDiagonal	Діагональне штрихування з нахилом ліній назад
bsCross	Горизонтально-вертикальне штрихування, в клітинку
bsDiagCross	Діагональне штрихування, в клітину

Лінія

Викреслювання прямої лінії здійснює метод LineTo, інструкція виклику якого в загальному вигляді виглядає таким чином:

Компонент.Canvas.LineTo(x,y)

Метод LineTo викреслює пряму лінію від поточної позиції олівця в точку з координатами, вказаними при виклику методу.

Початкову точку лінії можна задати, перемістивши олівець в потрібну точку графічної поверхні. Зробити це можна за допомогою методу MoveTo, вказавши як параметри координати нового положення олівця.

Вид лінії (колір, товщина і стиль) визначається значеннями властивостей об'єкту Реп графічної поверхні, на якій викреслюється лінія.

Коло і еліпс

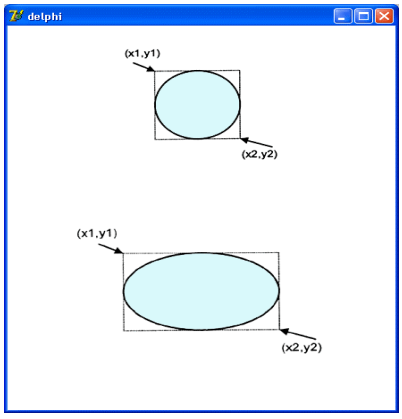
Метод Ellipse викреслює еліпс або коло, залежно від значень параметрів. Інструкція виклику методу в загальному вигляді виглядає таким чином:

Об'єкт.Canvas.Ellipse(x1,y1, x2,y2)

де:

об'єкт — ім'я об'єкту (компоненту), на поверхні якого виконується викреслювання;

x1, y1, x2, y2 — координати прямокутника, усередині якого викреслюється еліпс або, якщо прямокутник є квадратом, коло



Мал. 1.2. Значення параметрів методу Ellipse визначають вид геометричної фігури

Колір, товщина і стиль лінії еліпса визначаються значеннями властивості Pen, а колір і стиль заливки області усередині еліпса — значеннями властивості Brush поверхні (canvas), на яку виконується вивід малюнка.

Многокутник

Метод Polygon викреслює багатокутник. Як параметр метод одержує масив типу TPoint. Кожний елемент масиву є записом, полями (x,y) яку містять координати однієї вершини багатокутника. Метод Polygon викреслює многокутник, послідовно сполучаючи прямими лініями точки, координати яких знаходяться в масиві: першу з другою, другу з третьою, третиною з четвертою і т.д. Потім з'єднуються остання і перша точки.

Колір і стиль межі многокутника визначаються значеннями властивості Pen, а колір і стиль заливки області, обмеженою лінією межі, — значеннями властивості Brush, причому область замальовується з використанням поточного кольору і стилю кисті.

Нижче наведена процедура, яка, використовуючи метод polygon, викреслює трикутник:

```
procedure TForm1.Button2Click(Sender: TObject);
var
pol: array[1..3] of TPoint; // координати точок трикутника
begin
pol[1].x := 10;
polf1].y := 50;
pol[2].x := 40;
pol[2].y := 10;
pol[3].x := 70;
pol[3].y := 50;
Form1.Canvas.Polygon(pol);
end;
```

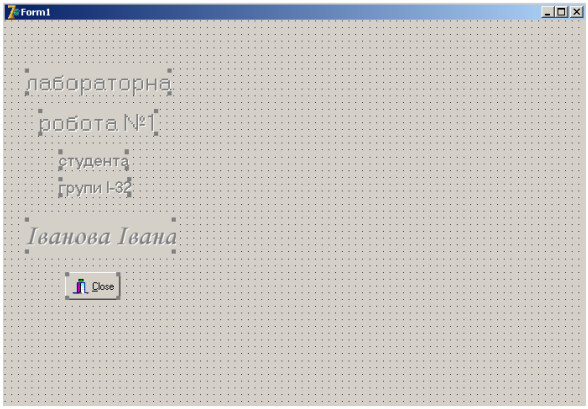
ХІД РОБОТИ

1) Завантажте середовище візуального програмування Delphi.
2) Створіть новий проект, назвою якому задайте власне прізвище.
3) Запрограмуйте подію OnCreate, створивши процедуру Param, в якій задайте початкове значення кольору і ширини олівця та кольору пензлика:

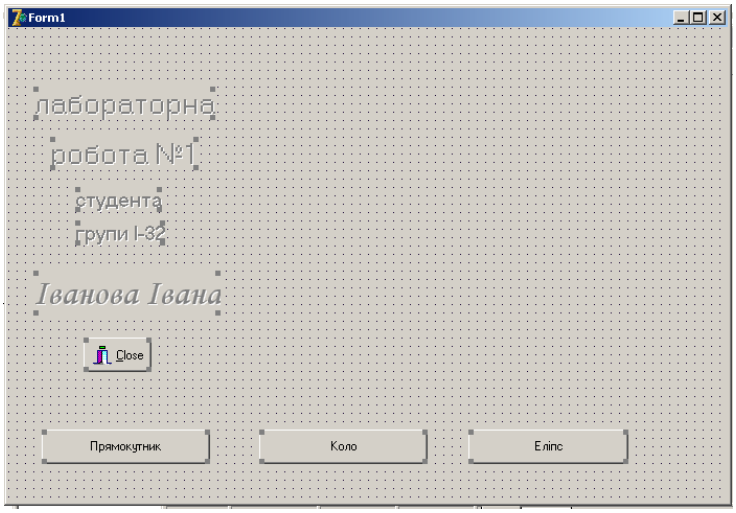
```
procedure TForm1.Param(Sender: TObject);
begin
Canvas.Pen.Color:=clBlue;
Canvas.Brush.Color:=clWhite;
Canvas.Pen.Width:=5;
end;
```

Задайте властивість BorderStyle форми –“bsNone”.

4) Створіть кнопку закривання форми – об’єкт типу BitBtn (закладка Additional), для якого вкажіть властивість Kind кнопки – “bkClose”. Задайте для властивості форми BorderStyle значення bsNone.
5) Створивши відповідні текстові поля (для них властивість Enabled встановіть “False”) підпишіть форму, як це вказано на малюнку:



6) Створіть три кнопки, які підпишіть відповідно «Прямокутник», «Коло», «Еліпс» (Використовуючи властивість Caption). Розташуйте їх рівновіддалено одна від одної в



нижній частині форми. Задайте однакові розміри кнопок.

7) Запрограмуйте кнопку «Прямокутник» так, щоб при її натисканні на екран виводилось зображення прямокутника із лівою верхньою вершиною (w/2-100, h/2-100) і правою нижньою вершиною (w/2+100, h/2), де w і h – видимі розміри форми

```
procedure TForm1.Button1Click(Sender: TObject);
begin
Canvas.Rectangle(ClientWidth div 2 -100, ClientHeight div 2 -100,ClientWidth div 2 +100,ClientHeight
div 2 );
end;
```

8) Запрограмуйте кнопку «Коло» так, щоб при її натисканні на екран виводилось зображення кола із центром в точці (w/2,h/2-50) і радіусом 100.

```
procedure TForm1.Button2Click(Sender: TObject);
begin
Canvas.Ellipse(ClientWidth div 2 -100, ClientHeight div 2 -150,ClientWidth div 2 +100,ClientHeight
div 2 +50);
end;
```

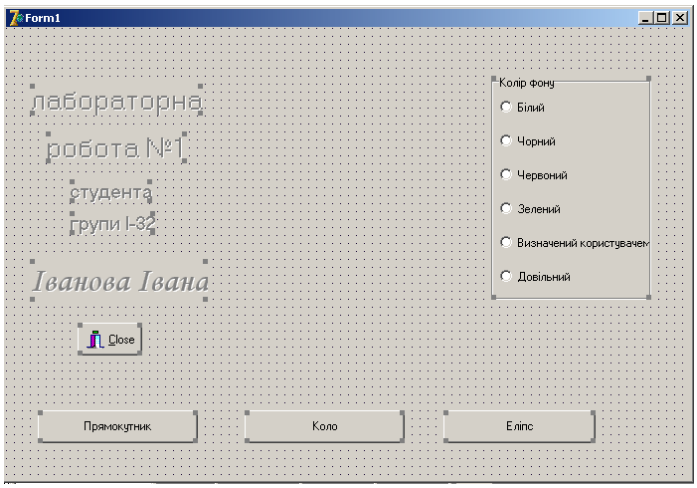
9) Запрограмуйте кнопку «Еліпс» так, щоб при її натисканні на екран виводилось зображення еліпса із центром в точці (w/2, h/2-50), великою піввіссю b=100 , і малою піввіссю 50.

```
procedure TForm1.Button3Click(Sender: TObject);
begin
Canvas.Ellipse(ClientWidth div 2 -100, ClientHeight div 2 -100,ClientWidth div 2 +100,ClientHeight div
2);
end;
```

10) Для того, щоб при кожному клацанні кнопок фігури, виведені на екран, не накладались, потрібно у кожну з процедур TForm1.ButtonClick додати процедуру «перемальовки» форми Refresh.

11) Створіть групу перемикачів (об’єктів типу RadioButton) під назвою «Колір фону» із такими елементами списку:

- білий;
- чорний;
- червоний;
- зелений;
- визначений користувачем;
- довільний



Для перших чотирьох елементів запрограмуйте кожен елемент списку, задавши відповідний колір пензлика:

```
procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
if RadioGroup1.ItemIndex=0 then
        Begin
        Canvas.Brush.Color:=clWhite;
        end;
if RadioGroup1.ItemIndex=1 then
        Begin
        Canvas.Brush.Color:=clBlack;
        end;
if RadioGroup1.ItemIndex=2 then
        Begin
       Canvas.Brush.Color:=clRed;
        end;
if RadioGroup1.ItemIndex=3 then
        Begin
        Canvas.Brush.Color:=clGreen;
        end;
```

12) В процедурі TForm1.Param опису події OnCreate задайте початкове положення перемикачів, встановивши перемикач «Білий». Вказана процедура набуде вигляду:

```
procedure TForm1.Param(Sender: TObject);
begin
RadioGroup1.ItemIndex:=0;
Canvas.Pen.Color:=clBlue;
Canvas.Brush.Color:=clWhite;
Canvas.Pen.Width:=5;
end;
```

13) Запрограмуйте перемикач «Довільний» в процедурі TForm1.RadioGroup1Click, задавши вибір кольору функцією довільного вибору кольору:

```
if RadioGroup1.ItemIndex=5 then
        Begin
        Canvas.Brush.Color:=rgb(random(255),random(255),random(255));
        end;
```

14) Для того, щоб перемикання радіокнопок призводило до перемальовки зображення, необхідно у кожній з відповідних процедур опису радіокнопок викликати процедуру

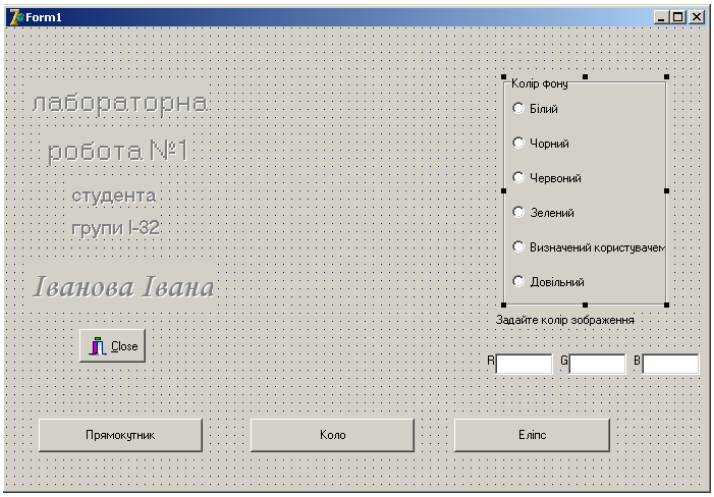
перемальовки активної фігури, тобто процедуру натискання відповідної кнопки. Для контролю за типом фігури введемо нову змінну (наприклад, p1), яка буде набувати значення 1, 2, 3 при натисканні відповідних кнопок «Прямокутник», «Коло», «Еліпс». Процедури опису даних кнопок матимуть вигляд:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Refresh;
p1:=1 ;
Canvas.Rectangle(ClientWidth div 2 -100, ClientHeight div 2 -100,ClientWidth div 2 +100,ClientHeight
div 2 );
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 Form1.Refresh;
 p1:=2;
Canvas.Ellipse(ClientWidth div 2 -100, ClientHeight div 2 -150,ClientWidth div 2 +100,ClientHeight div
2 +50);
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
Form1.Refresh;
p1:=3;
 Canvas.Ellipse(ClientWidth div 2 -100, ClientHeight div 2 -100,ClientWidth div 2 +100,ClientHeight
div 2);
end;
```

Тепер запрограмуємо «перемальовку» активної фігури вибраним кольором:

```
procedure TForm1.RadioGroup1Click(Sender: TObject);
begin
if RadioGroup1.ItemIndex=0 then
        Begin
        Canvas.Brush.Color:=clWhite;
        if p1=1 then Button1Click(sender);
        if p1=2 then Button2Click(sender);
        if p1=3 then Button3Click(sender);
        end;
  if RadioGroup1.ItemIndex=1 then
        Begin
        Canvas.Brush.Color:=clBlack;
        if p1=1 then Button1Click(sender);
        if p1=2 then Button2Click(sender);
        if p1=3 then Button3Click(sender);
        end;
  if RadioGroup1.ItemIndex=2 then
        Begin
        Canvas.Brush.Color:=clRed;
        if p1=1 then Button1Click(sender);
        if p1=2 then Button2Click(sender);
        if p1=3 then Button3Click(sender);
        end;

  if RadioGroup1.ItemIndex=3 then
         Begin
         Canvas.Brush.Color:=clGreen;
         if p1=1 then Button1Click(sender);
         if p1=2 then Button2Click(sender);
         if p1=3 then Button3Click(sender);
         end;
  if RadioGroup1.ItemIndex=5 then
        Begin
        Canvas.Brush.Color:=rgb(random(255),random(255),random(255));
        if p1=1 then Button1Click(sender);
        if p1=2 then Button2Click(sender);
        if p1=3 then Button3Click(sender);
        end;
end;
```

15) Запрограмуйте перемикач «Визначений користувачем» так, щоб при його включенні користувач міг вводити у відповідні поля редагування значення складових Red, Green, Blue (режим RGB) кольору фону. Для цього створіть на формі три текстові поля і три поля редагування так, як це вказано на малюнку:



16) Запрограмуйте перемикач «Визначений користувачем» так, щоб встановлювався колір фону в залежності від введених користувачем значень у поля редагування. Для цього в процедурі
procedure TForm1.RadioGroup1Click(Sender: TObject) допишіть фрагмент програми:

```
if RadioGroup1.ItemIndex=4 then
        Begin
        val(Edit1.Text,r,cod);
         val(Edit2.Text,g,cod);
          val(Edit3.Text,b,cod);
                Canvas.Brush.Color:=RGB(r,g,b);
                if p1=1 then Button1Click(sender);
                if p1=2 then Button2Click(sender);
                if p1=3 then Button3Click(sender);
                end;
```

Зауваження. *Опишіть змінні* r,g,b, cod як проміжні змінні вказаної процедури.
Для перемальовування активної фігури вказаним кольором (для «миттєвої» реакції на числа, що вводяться) задайте для всіх трьох полів редагування подію “On Change” – RadioGroup1Click.
17) Змініть код програми так, щоб всі поля (текстові і редагування, створені у пункті 12), необхідні при заданні кольору користувачем, відображалися тільки у випадку включення перемикача «Визначений користувачем».
18) Задайте можливість вибору кольору межі, аналогічно як для вибору кольору фону.
19) Замість виведення зображення кола передбачте виведення на екран трикутника .

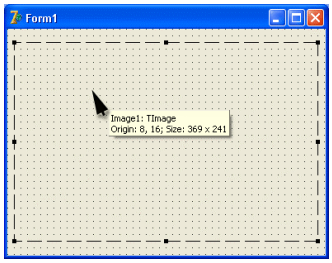
ЛАБОРАТОРНА РОБОТА №2
ФУНКЦІЇ ГРАФІЧНИХ ПОБУДОВ. ПОБУДОВА ГРАФІКІВ

МЕТА: Закріпити основні параметри графічних побудов. Оволодіти практичними навиками відображення тексту. Навчитись розробляти демонстраційні системи із використанням графіки

ТЕОРЕТИЧНІ ВІДОМОСТІ

Поверхні, на яку програма може здійснювати виведення графіки, відповідає об'єкт Canvas.

Положення пікселя характеризується його горизонтальною (X) і вертикальною (Y) координатами. Лівий верхній піксель має координати (0, 0). Координати зростають зверху вниз і зліва направо (мал. 1.1). Значення координат правої нижньої точки полотна залежать від розміру полотна.



Мал. 1.1. Координати точок полотна

Розмір полотна можна одержати, звернувшись до властивостей Height и Width області ілюстрації (image) або до властивостей форми: ClientHeight и ClientWidth.

Властивість pixels, що є двовимірним масивом типу TColor, містить інформацію про колір кожної точки графічної поверхні. Використовуючи властивість Pixels, можна задати потрібний колір для будь-якої точки графічної поверхні, тобто "намалювати" точку. Наприклад, інструкція

```
Form1.Canvas.Pixels[10,10]:=clRed
```

замальовує точку поверхні форми в червоний колір.

Розмірність масиву pixels визначається розміром графічної поверхні. Розмір графічної поверхні форми (робочої області, яку також називають клієнтською) задається значеннями властивостей clientwidth і ClientHeight, а розмір графічної поверхні компонента image — значеннями властивостей Width і Height. Лівій верхній точці робочої області форми відповідає елемент pixels [0,0], а правій нижній -Pixels[Ciientwidth - 1,ClientHeight - 1].

Властивість Pixels можна використовувати для побудови графіків. Графік будується, як правило, на основі обчислень за формулою. Межі діапазону зміни аргументу функції є вихідними даними. Діапазон зміни значення функції може бути обчислений. На підставі цих даних можна обчислити масштаб, що дозволяє побудувати графік так, щоб він займав всю область форми, призначену для виведення графіка.

Функція TextOut дозволяє малювати текст, використовуючи шрифт, заданий в канвасі:

ІМ'Я	ОПИС	ПРИКЛАД
TextOut	Малює даний рядок на канвасі починаючи з координат (x,y) - фон тексту заповнюється поточним кольором кисті.	Canvas.TextOut(10, 10, 'Some text');

Функція дозволяє малювати текст, не заповнюючи його фон. Якщо Вам необхідно змінити шрифт, використовуваний в TextOut, то необхідно змінити властивість Font канваса (ця властивість має тип TFont) - наприклад "Canvas.Font.Name := 'Verdana';", "Canvas.Font.Size := 24;" або "Canvas.Font.Color := clRed;".

Компонент ComboBox – створення списку

Основна властивість компонента - Items. Тип - TStrings. Вона визначає список рядків компонента. Додавання елементів, їх видалення, очистка вмісту ComboBox відбувається за допомогою властивості Items.

Операції з елементами списку можуть здійснюватись також програмно:

Код для додаваня нового елемента в список:
ComboBox1.Items.Add('Новий елемент');
Код для очистки всього списка:
ComboBox1.Items.Clear;
Код для удалення елемента списка з заданим номером, номери починаються з нуля:
ComboBox1.Items.Delete(1);
Параметр в дужках – це номер елемента, який удаляється
Код для додавання нового елемента в задане місце:
ComboBox1.Items.Insert(2,'Roman');

Вибраний користувачем елемент ComboBox можно визначити за допомогою властивості Text, а його номер - за допомогою властивості ItemIndex (нумерація починається з нуля), наприклад

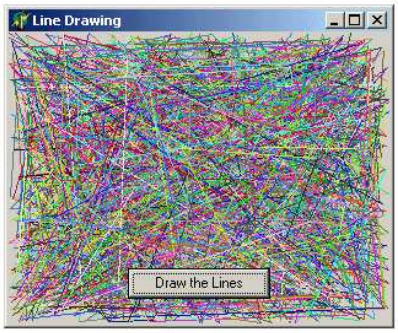
if ComboBox1.Text='Вибраний елемент' then begin…end,

де 'Вибраний елемент' – вибраний користувачем елемент зі списку ComboBox.

ХІД РОБОТИ.

1) Виконати приклад хаотичного малювання кольорових ліній

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Randomize;
end;

const NUM_LINES = 2000;

procedure TForm1.DrawLines;
var
  i: Integer;
begin
  for i := 0 to NUM_LINES - 1 do
  begin
    Canvas.Pen.Color :=
         RGB(Random(256),
             Random(256),
             Random(256)
             );
    Canvas.LineTo
       (Random(ClientWidth),
        Random(ClientHeight));
  end;
end;
```



Процедура DrawLines викликається з обробника кнопки OnClick. Кількість ліній задається в константі NUM_LINES. Функція RGB, утворює колір кожної лінії з трьох основних складових: червоного, зеленого і синього (значення від 0 до 255) і повертає колір у вигляді TColor.

2) Нарисувати різними кольорами десять концентричних кіл, які мають спільний центр посередині екрана, і описати навколо кіл червоний прямокутник.

3) Нарисувати емблему спеціальності: на чорному фоні нарисувати блакитний квадрат, а в ньому – чорне коло, зафарбоване довільним кольором. У центрі емблеми написати слово “Інформатика”

4) Створити демонстраційну систему, яка дозволяє:

1. здійснити вибір функції для побудови графіка зі списку, що містить не менше 5 функцій та побудувати графік вибраної зі списка функції. (Приклад побудови одного графіка функції – синусоїди наведено нижче)
2. здійснити вибір кольору для побудови графіка.
3. побудувати осі координат
4. підписати назви осей та графіка

Приклад побудови одного графіка функції – синусоїди

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    procedure FormPaint(Sender: TObject);
     private
    { Private declarations }
  public
    { Public declarations }
  end;
```

ЛАБОРАТОРНА РОБОТА №3
ОСНОВНІ СТРУКТУРИ ГЕОМЕТРИЧНИХ ДАНИХ: КЛАС ТОЧКА

МЕТА: Ознайомитися із типовими полями та методами класу Point (Точка). Опанувати створення відповідного класу на мові програмування (за вибором). Оволодіти практичними навиками при використанні даних типу TPoint.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Клас Point містить елементи даних – поля x і y для зберігання координат точки. Компонентні функції забезпечують основні операції, які виконуються над точками чи векторами. Додаткові функції забезпечують графічну реалізацію конкретних задач.

Операції < і > реалізують лексикографічний порядок відношень, коли вважається, що точка P1 менша за точку P2, якщо або P1.x < P2.x, або P1.x = P2.x і P1.y < P2.y. Для двох даних точок спочатку порівнюються їх x-координати, і, якщо вони рівні, то порівнюються y-координати. Такий порядок іноді називається словниковим порядком відношень, оскільки за таким же правилом розташовуються в словнику слова із двох букв.

ХІД РОБОТИ

1) Завантажте середовище візуального програмування Delphi. Створіть у папці власної групи папку і запишіть у неї новий проект.

2) Задайте властивості форми BorderStyle значення bsDialog. Створіть кнопку Close для закриття програми, яку розташуйте в правому нижньому куті майбутнього вікна.

3) Створіть робоче поле на формі для відображення запрограмованої в подальшому графіки. Для цього подію OnPaint запрограмуйте процедурою Pole:

```
procedure TForm1.Pole(Sender: TObject);
   begin
    Form1.Canvas.Pen.Color:=clBlue;
    Form1.Canvas.Brush.Color:=clWhite;
    Form1.Canvas.Rectangle(1,1,ClientWidth-1,ClientHeight-50);
   end;
```

4) Створіть клас Point, який буде описувати точку на площині, із полями, що вказують на координати точки – x, y та процедурами: Init, що ініціалізуватиме точку, і Draw, що малюватиме точку заданим кольором:

```
TPoint1 = object
    x,y : integer;
    Procedure Init(a,b:integer);
    Procedure Draw (C:TColor);
    end;
```

5) У розділі оголошення глобальних змінних оголосіть змінну p1 нового типу – p1: TPoint1.

6) Опишіть методи класу TPoint1:

```
Procedure TPoint1.Init(a,b:integer);
Begin
x:=a;y:=b;
end;
Procedure TPoint1.Draw(C:TColor);
Begin
Form1.Canvas.Pen.Color:=C;
Form1.Canvas.Brush.Color:=C;
Form1.Canvas.Ellipse(x-3,y-3,x+3,y+3);
end;
```

Увага! Після опису методів точки відображатись не будуть.

7) Створіть на формі два порожні поля редагування Edit1 і Edit2 і відповідні їм два текстові поля Label1 і Label2 (з підписами X1 і Y1 відповідно). Запрограмуйте подію OnMouseDown так, щоб при клацанні мишкою на робочому полі у відповідних полях редагування Edit1 і Edit2 відображалися координати цієї точки і на екран виводилося її зображення (як об'єкту класу TPoint1).

```
procedure TForm1.PutPoint(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
  var s1,s2 : string;
begin
 if (y< ClientHeight-50) then
      Begin
        str(x,s1);str(y,s2);
        Edit1.Text:= s1;
        Edit2.Text:=s2;
        p1.Init(x,y);
        p1.Draw(clGreen);
      end;
end;
```

Завдання для самостійної роботи.

8) Задайте та опишіть метод для класу TPoint1 (процедуру ReDraw), який би "замальовував" точку (малював її кольором фону – білим).

9) Видозмініть процедуру TForm1.PutPoint так, щоб на екрані відображалася тільки поточна точка та її координати (використайте процедуру ReDraw).

10) Запрограмуйте можливість задання на екрані з допомогою мишки двох точок та виведення їх координат у відповідні поля редагування, а також обчислення відстані між цими точками і виведення результату у вікно редагування.

11) Опишіть методи класу Point1 Menshe і Bilshe, які б реалізовували перевірку лексикографічного порядку відношень між поточною і вказаною точкою. Наприклад,

```
Function TPoint1.Menshe(p:TPoint1):boolean;
Begin
if (x<p.x) or ((x=p.x) and (y<p.y)) then Menshe:=True
    else  Menshe:=False;
end;
```

Запрограмуйте перевірку заданого порядку для двох точок, заданих клацанням мишки. Результат порівняння виведіть у поле редагування.

12) Додайте до методів класу Point1 дві функції Plus і Minus, які будуть реалізовувати векторну арифметику (ініціалізувати точку з координатами, які є сумою/різницею відповідних координат двох точок).

13) Запрограмуйте виведення на екран точки P3, яка одержана в результаті векторної суми точок P1 і P2 (які задаються клацанням мишки) і виведення координат точки P3 у поля редагування. Задайте колір двох вихідних точок зеленим, а колір точки, одержаної в результаті побудови – червоний.

ЛАБОРАТОРНА РОБОТА №4
СТВОРЕННЯ ТА ПЕРЕТВОРЕННЯ ГРАФІЧНИХ ПРИМІТИВІВ.

МЕТА: Ознайомитися із типовими полями та методами класів Point і Segment. Опанувати створення відповідних класів на мові програмування (за вибором). Оволодіти практичними навиками при використанні даних типу TPoint, TSegment.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Відрізок прямої лінії P1P2 складається з двох *кінцевих точок* P1 і P2 разом з точками, що лежать між ними. Коли важливий порядок розміщення точок P1 і P2, то задано *напрямлений відрізок прямої лінії* P1P2. Методи класу відрізок:

- Ініціалізація відрізка
- Відображення певним кольором
- Замальовування кольором фону
- Обчислення довжини відрізка
- Поворот відрізка
- Переміщення на задану відстань
- Обчислення відстані від точки до відрізка (прямої)
- інші

Під терміном ***поворот ребра*** розуміють обертання ребра на 90 градусів в напрямі за годинниковою стрілкою навколо його середньої точки.

Шляхом повороту ребра ab формується ребро cd. Якщо вектор b - a = (x, y), то вектор n, перпендикулярний вектору b - a, визначається як n = (y, -x). Середня точка m між точками a і b визначається як m = (a + b)/2. Тоді точки c і d визначаються як c = m - n/2 і d = m + n/2.

ХІД РОБОТИ

1) Завантажте середовище візуального програмування Delphi. Створіть у папці власної групи папку і запишіть у неї новий проект.
2) Задайте властивості форми BorderStyle значення bsDialog. Створіть кнопку Close для закриття програми, яку розташуйте в правому нижньому куті майбутнього вікна.
3) Створіть робоче поле на формі для відображення запрограмованої в подальшому графіки. Для цього подію OnPaint запрограмуйте процедурою Pole:

```
procedure TForm1.Pole(Sender: TObject);
   begin
    Form1.Canvas.Pen.Color:=clBlue;
    Form1.Canvas.Brush.Color:=clWhite;
    Form1.Canvas.Rectangle(1,1,ClientWidth-1,ClientHeight-50);
   end;
```

4) Створіть клас Point1, який буде описувати точку на площині, із наступними полями та процедурами:

```
TPoint1 = object
      x,y : integer;
      Procedure Init(a,b:integer);
      Procedure Draw(C:TColor);
     end;
```

4) Опишіть методи класу TPoint1: (методи ініціалізації та малювання описуються аналогічно, як описано в попередній роботі).

```
Procedure TPoint1.Init(a,b:integer);
Begin
x:=a;y:=b;
end;
Procedure TPoint1.Draw(C:TColor);
Begin
Form1.Canvas.Pen.Color:=C;
Form1.Canvas.Brush.Color:=C;
Form1.Canvas.Ellipse(x-3,y-3,x+3,y+3);
end;
```

5) Створіть клас TSegment (Відрізок) із полями pp, pk – точки типу TPoint1, які задають початок і кінець відрізка та методами Init і Draw, які ініціалізують відрізок та малюють його, аналогічно як для класу TPoint1. А також задайте метод Length, який обчислює довжину відрізка.

```
TSegment = object
      pp,pk : TPoint1;
      Procedure Init(p1,p2 : TPoint1);
      Procedure Draw(C:TColor);
      function Length(s:TSegment):real;
      end;
```

Процедури ініціалізації, малювання та знаходження довжини відрізка:

```
Procedure TSegment.Init(p1,p2:TPoint1);
Begin
pp:=p1; pk:=p2;
end;

Procedure TSegment.Draw(C:TColor);
Begin
Form1.Canvas.Pen.Color:=C;
Form1.Canvas.Brush.Color:=C;
Form1.Canvas.MoveTo(pp.x,pp.y);
Form1.Canvas.LineTo(pk.x,pk.y);
end;

function TSegment.Length(s:TSegment):real;
begin
Length:=sqrt(sqr(pp.x-pk.x)+sqr(pp.y-pk.y))
end;
```

Запрограмуйте подію OnMouseDown так, щоб за допомогою клацання мишки виводилося

на екран дві точки, координати яких виводилися б у відповідні поля редагування. Причому повинен малюватися відповідний відрізок (як об'єкт типу TSegment) та виводитись на екран його довжина. Наприклад, ініціалізація та малювання відрізка v1 типу TSegment, який сполучає точки P1 та P2, здійснюється за допомогою команд:

```
v1.Init(p1,p2);
v1.Draw(clRed);
l:=v1.Length(v1);
```

Зауваження 1: точки P1, P2 та відрізок v1 потрібно оголосити як глобальні змінні.

6) Запрограмуйте можливість виведення на екран за допомогою клацань мишки трьох точок P1, P2, P3 та малювання відрізків P1P2, P2P3, P1P3 (як об'єктів типу TSegment).

7) Створіть кнопку "Замалювати", за допомогою якої із використанням методу полігон побудований трикутник замальовується.

8) Створіть кнопку "Перемістити", вибір якої здійснює паралельне переміщення побудованого трикутника по горизонталі на задану в полі редагування відстань (у пікселях).

9) Створіть кнопку "Перемістити", вибір якої здійснює паралельне переміщення побудованого трикутника в довільно задану мишкою точку P4 (вершина P1 переміщається в P4).

10) Створіть кнопку "Повернути", вибір якої здійснює поворот відрізка P1P2 на 90º.

11) Побудувати прямокутник за заданими у полях редагування довжиною, висотою та координатами однієї вершини.

12) Запрограмуйте процедуру повороту на 90º побудованого прямокутника.

13) Запрограмуйте можливість виведення на екран за допомогою клацань мишки чотирьох точок та малювання відрізків – сторін чотирикутника, які їх з'єднують (як об'єктів типу TSegment).

ЛАБОРАТОРНА РОБОТА №5
СТВОРЕННЯ ІЛЮСТРАЦІЙ ТА АНІМАЦІЯ.

МЕТА: Закріпити основні параметри графічних побудов. Оволодіти практичними навиками відображення та імітації руху графічних об'єктів.

МЕТОДИЧНІ ВКАЗІВКИ

Імітація руху об'єкта на екрані.

Для імітації руху зображення об'єкта на екрані необхідно виконати такий алгоритм:

1. Нарисувати об'єкт у заданій точці.
2. Витерти об'єкт, зарисувавши його кольором тла.
3. Змінити координати об'єкта.
4. Перейти до пункту 1.

Функція Sleep(n) призначена для призупинки виконання програми на n мілісекунд.

ХІД РОБОТИ

1) разити горизонтальний рух кола за допомогою функції Sleep(n).
2) ористовуючи фоновий рисунок voda.bmp та рисунок із зображенням дельфіна delphin.bmp запрограмувати із використанням таймера:
   2.1) горизонтальний рух дельфіна на одній глибині, наприклад:

```
type
  TForm1 = class(TForm)
    Button1: TButton;
    Image1: TImage;
    Timer1: TTimer;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  Fon,Picture:TBitMap;
  W,H,WF,HF,x,y: integer;
implementation

{$R *.dfm}
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
Fon:= TBitmap.Create;
Picture := TBitmap.Create;
Fon.LoadFromFile('voda.bmp');
Picture.LoadFromFile('delphin.bmp');
  Image1.Width  := Fon.Width;
  Image1.Height := Fon.Height;
  Picture.Transparent := True;
  Image1.Canvas.Draw(0,0,fon);
  Image1.Canvas.Draw(0,0,picture);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
Close;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
x:=x+10;
if x > Image1.Width then x:=0;
y:=0;
Image1.Canvas.Draw(0,0,fon);
Image1.Canvas.Draw(x,y,picture);
end;


end.
```

2.2) горизонтальний рух дельфіна на іншій глибині після кожного проходження фонового рисунка

2.3) імітацію плавання – горизонтальний рух дельфіна по синусоїді на одній глибині.

2.4) імітацію плавання – горизонтальний рух дельфіна по синусоїді на іншій глибині після кожного проходження фонового рисунка.

2.5) Запрограмувати імітацію руху довільного об’єкта за довільною траєкторією.

ЛАБОРАТОРНА РОБОТА №6
РЕАЛІЗАЦІЯ ОСНОВНИХ СТРУКТУР ГЕОМЕТРИЧНИХ ДАНИХ ТА ВЕКТОРНИХ ОПЕРАЦІЙ.

МЕТА: Ознайомитися із типовими полями та методами класу Point (Точка). Опанувати створення відповідного класу на мові програмування (за вибором). Оволодіти практичними навиками при використанні даних типу TPoint.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Клас Point містить елементи даних – поля x і y для зберігання координат точки.

Компонентні функції забезпечують основні операції, які виконуються над точками чи векторами. Додаткові функції забезпечують графічну реалізацію конкретних задач.

Операції < і > реалізують лексикографічний порядок відношень, коли вважається, що точка P1 менша за точку P2, якщо або P1.x < P2.x, або P1.x = P2.x і P1.y < P2.y. Для двох даних точок спочатку порівнюються їх x-координати, і, якщо вони рівні, то порівнюються y-координати. Такий порядок іноді називається словниковим порядком відношень, оскільки за таким же правилом розташовуються в словнику слова із двох букв.

Впорядкована пара векторів називається додатньо орієнтованою, якщо найкоротший поворот від першого вектора до другого здійснюється проти годинникової стрілки. Якщо навпаки, то ця пара від'ємно орієнтована. Приклад додатньо орієнтованої пари: (1,0) і (0,1). Від'ємно орієнтована пара (0,1) і (1,0). Орієнтація пари векторів $a=(x_a, y_a)$ і $b=(x_b,y_b)$ може бути одержана як знак $sign(x_a*y_b-x_b*y_a)$. Вираз $x_a*y_b-x_b*y_a$ дорівнює площі (із знаком) паралелограма з вершинами 0, a, b, a+b.

Одна з найважливіших операцій полягає у визначенні положення точки відносно напрямленого відрізка прямої. Напрямлений відрізок прямої лінії чітко розділяє площину на сім областей, які не перетинаються, і ця операція визначає, якій з цих областей належить точка: 1 – ліворуч від прямої; 2 – праворуч від прямої; 3 – лежить на прямій до початку напрямленого відрізка прямої; 4 – лежить на прямій після кінця напрямленого відрізка прямої; 5 – співпадає з початком напрямленого відрізка прямої; 6 – співпадає з кінцем напрямленого відрізка прямої; 7 – лежить на напрямленому відрізку прямої.

Для визначення положення поточної точки P2 відносно відрізка прямої лінії P0P1, напрямленого від точки P0 до точки P1 потрібно виконати наступні дії.

Спочатку перевіряється орієнтація точок P0, P1 і P2, щоб визначити, чи розташовується точка P2 зліва або справа, або вона колінеарна з відрізком P0P1. В останньому випадку необхідні додаткові обчислення, якщо вектори a=Pl-P0 і b=P2-P0 мають протилежний напрям, то точка P2 лежить позаду напрямленого відрізка P0P1, якщо вектор a коротший за вектор b, то точка P2 розташована після відрізка P0P1. Інакше точка P2 порівнюється з точками P0 і P1 для визначення, чи співпадає з однією з цих кінцевих точок, чи лежить між ними.

ХІД РОБОТИ

1) Завантажте середовище візуального програмування Delphi. Створіть у папці власної групи папку і запишіть у неї новий проект.

2) Задайте властивості форми BorderStyle значення bsDialog. Створіть кнопку Close для закриття програми, яку розташуйте в правому нижньому куті майбутнього вікна.

3) Створіть робоче поле на формі для відображення запрограмованої в подальшому графіки. Для цього подію OnPaint запрограмуйте процедурою Pole:

```
procedure TForm1.Pole(Sender: TObject);
   begin
    Form1.Canvas.Pen.Color:=clBlue;
    Form1.Canvas.Brush.Color:=clWhite;
    Form1.Canvas.Rectangle(1,1,ClientWidth-1,ClientHeight-50);
   end;
```

4) Створіть клас Point, який буде описувати точку на площині, із полями, що вказують на координати точки – x, y та процедурами: Init, що ініціалізуватиме точку, і Draw, що малюватиме точку заданим кольором:

```
TPoint1 = object
    x,y : integer;
    Procedure Init(a,b:integer);
    Procedure Draw (C:TColor);
    end;
```

5) У розділі оголошення глобальних змінних оголосіть змінну p1 нового типу – p1: TPoint1.

6) Опишіть методи класу TPoint1:

```
Procedure TPointl.Init(a,b:integer);
Begin
x:=a;y:=b;
end;
Procedure TPointl.Draw(C:TColor);
Begin
Forml.Canvas.Pen.Color:=C;
Forml.Canvas.Brush.Color:=C;
Forml.Canvas.Ellipse(x-3,y-3,x+3,y+3);
end;
```

7) Створіть на формі два порожні поля редагування Edit1 і Edit2 і відповідні їм два текстові поля Label1 і Label2 (з підписами X1 і Y1 відповідно). Запрограмуйте подію OnMouseDown так, щоб при клацанні мишкою на робочому полі у відповідних полях редагування Edit1 і Edit2 відображалися координати цієї точки і на екран виводилося її зображення (як об'єкту класу TPoint1).

```
procedure TForml.PutPoint(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
  var sl,s2 : string;
begin
 if (y< ClientHeight-50) then
      Begin
        str(x,sl);str(y,s2);
        Editl.Text:= sl;
        Edit2.Text:=s2;
        pl.Init(x,y);
        pl.Draw(clGreen);
      end;
end;
```

Завдання для самостійної роботи.

8) Задайте та опишіть метод для класу TPoint1 (процедуру ReDraw), який би "замальовував" точку (малював її кольором фону – білим).

9) Видозмініть процедуру TForm1.PutPoint так, щоб на екрані відображалася тільки поточна точка та її координати (використайте процедуру ReDraw).

10) Запрограмуйте можливість задання на екрані з допомогою мишки двох точок та виведення їх координат у відповідні поля редагування, а також обчислення відстані між цими точками і виведення результату у вікно редагування.

11) Опишіть методи класу Point1 Menshe і Bilshe, які б реалізовували перевірку лексикографічного порядку відношень між поточною і вказаною точкою. Наприклад,

```
Function TPointl.Menshe(p:TPointl):boolean;
Begin
if (x<p.x) or ((x=p.x) and (y<p.y)) then Menshe:=True
    else  Menshe:=False;
end;
```

Запрограмуйте перевірку заданого порядку для двох точок, заданих клацанням мишки. Результат порівняння виведіть у поле редагування.

12) Додайте до методів класу Point1 дві функції Plus і Minus, які будуть реалізовувати векторну арифметику (ініціалізувати точку з координатами, які є сумою/різницею відповідних координат двох точок).

13) Запрограмуйте виведення на екран точки P3, яка одержана в результаті векторної суми точок P1 і P2 (які задаються клацанням мишки) і виведення координат точки P3 у поля редагування. Задайте колір двох вихідних точок зеленим, а колір точки, одержаної в результаті «побудови» – червоний.

14) Опишіть функцію Orintation, яка б для заданих трьох точок P0, P1, P2 повертала 1, якщо вони додатньо орієнтовані, -1 – якщо вони від'ємно орієнтовані і 0 – якщо вони колінеарні (лежать на одній прямій). Реалізуйте перевірку роботи даної функції за допомогою графіки.

15) Опишіть метод Classification класу Point1, який класифікує положення поточної точки відносно напрямленої прямої P0P1.

ЛАБОРАТОРНА РОБОТА №7
РЕАЛІЗАЦІЯ ОСНОВНИХ СТРУКТУР ГЕОМЕТРИЧНИХ ДАНИХ. ЗАДАЧА КЛАСИФІКАЦІЇ ПОЛОЖЕННЯ ОБ'ЄКТІВ

МЕТА: Ознайомитися із типовими полями та методами класів Point і Segment. Опанувати створення відповідних класів на мові програмування (за вибором). Оволодіти практичними навиками при використанні даних типу TPoint, TSegment.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Впорядкована пара векторів називається додатно орієнтованою, якщо найкоротший поворот від першого вектора до другого здійснюється проти годинникової стрілки. Якщо навпаки, то ця пара від'ємно орієнтована. Приклад додатньо орієнтованої пари: (1,0) і (0,1). Від'ємно орієнтована пара (0,1) і (1,0). Орієнтація пари векторів $a=(x_a, y_a)$ і $b=(x_b,y_b)$ може бути одержана як знак $sign(x_a*y_b-x_b*y_a)$. Вираз $x_a*y_b-x_b*y_a$ дорівнює площі із знаком паралеллограма з вершинами 0, a, b, a+b.

Одна з найважливіших операцій полягає у визначенні положення точки щодо напрямленого відрізка прямої лінії. Напрямлений відрізок прямої лінії чітко розділяє площину на сім областей, які не перетинаються, і ця операція визначає, якій з цих областей належить точка: 1 – ліворуч від прямої; 2 – праворуч від прямої; 3 – лежить на прямій до початку напрямленого відрізка прямої; 4 – лежить на прямій після кінця напрямленого відрізка прямої; 5 – співпадає з початком напрямленого відрізка прямої; 6 – співпадає з кінцем напрямленого відрізка прямої; 7 – лежить на напрямленому відрізку прямої.

Для визначення положення поточної точки P2 відносно відрізка прямої лінії P0P1, напрямленого від точки P0 до точки P1 потрібно виконати наступні дії:

Спочатку перевіряється орієнтація точок P0, P1 і P2, щоб визначити, чи розташовується точка P2 зліва або справа, або вона колінеарна з відрізком P0P1. В останньому випадку необхідні додаткові обчислення, якщо вектори a=Pl-P0 і b=P2-P0 мають протилежний напрям, то точка P2 лежить позаду напрямленого відрізка P0P1, якщо вектор a коротший за вектор b, то точка P2 розташована після відрізка P0P1. Інакше точка P2 порівнюється з точками P0 і P1 для визначення, чи співпадає з однією з цих кінцевих точок, чи лежить між ними.

ХІД РОБОТИ

1) Завантажте середовище візуального програмування Delphi. Створіть у папці власної групи папку і запишіть у неї новий проект.
2) Задайте властивості форми BorderStyle значення bsDialog. Створіть кнопку Close для закриття програми, яку розташуйте в правому нижньому куті майбутнього вікна.
3) Створіть робоче поле на формі для відображення запрограмованої в подальшому графіки. Для цього подію OnPaint запрограмуйте процедурою Pole:

```
procedure TForm1.Pole(Sender: TObject);
   begin
    Form1.Canvas.Pen.Color:=clBlue;
    Form1.Canvas.Brush.Color:=clWhite;
    Form1.Canvas.Rectangle(1,1,ClientWidth-1,ClientHeight-50);
   end;
```

4) Створіть клас Point1, який буде описувати точку на площині, із наступними полями та процедурами:

```
TPoint1 = object
      x,y : integer;
      Procedure Init(a,b:integer);
      Procedure Draw(C:TColor);
    end;
```

5) Опишіть методи класу TPoint1 як в попередній роботі.
6) Опишіть функцію Menshe, яка перевіряє впорядкованість точок згідно з лексикографічним порядком.

```
function TPointl.Menshe (p:TPointl): boolean;
 Begin
 if ((x<p.x) or ((x=p.x) and (y<p.y)) )then Menshe:=True
 else Menshe:=False;
 end;
```

7) Опишіть для класу TPoint1 методи – функції Plus, Minus, Dob, які б реалізовували векторну арифметику: додавання, віднімання, скалярне множення на число.

Наприклад,

```
function TPointl.plus(p:TPointl) : TPointl;
  Begin
  plus.x:=x+p.x;
  plus.y:=y+p.y;
  end;
```

Завдання для самостійної роботи.

8)Створіть клас TSegment (Відрізок) із полями pp, pk – точки типу TPoint1, які задають початок і кінець відрізка та методами Init і Draw, які ініціалізують відрізок та малюють його, аналогічно як для класу TPoint1. А також задайте метод Length, який обчислює довжину відрізка.

```
TSegment = object
      pp,pk : TPoint1;
      Procedure Init(p1,p2 : TPoint1);
      Procedure Draw(C:TColor);
      function Length(s:TSegment):real;
      end;
```

9) Запрограмуйте подію OnMouseDown так, щоб за допомогою клацання мишки виводилося на екран дві точки, координати яких виводилися б у відповідні поля редагування. Причому повинен малюватися відповідний відрізок (як об’єкт типу TSegment) та виводитись на екран його довжина.

Зауваження 1: точки P1, P2 потрібно оголосити як глобальні змінні.

10) Запрограмуйте можливість виведення на екран трьох точок P1, P2, P3 та малювання відрізків P1P2, P1P3 (як об’єктів типу TSegment) .

11) Опишіть функцію Orientation, яка б для заданих трьох точок P1, P2, P3 повертала 1, якщо вони додатньо орієнтовані, -1 – якщо вони від’ємно орієнтовані і 0 – якщо вони колінеарні (лежать на одній прямій). Реалізуйте перевірку роботи даної функції за допомогою графіки.

12) Опишіть метод Classification класу Point1, який класифікує положення поточної точки відносно напрямленої прямої P1P2. Запрограмуйте можливість визначення положення точки P3 відносно прямої P1P2 (P1P2 задайте як об’кт типу TSegment) .

13) Запрограмуйте вивід на екран вивід 20 довільних точок та визначення кількості точок, що лежать справа/зліва відносно заданої прямої P1P2.

14) Опишіть для класу TSegment метод, який би визначав точку перетину двох відрізків. Запрограмуйте можливість задання двох відрізків за допомогою клацання мишки та знаходження точки їх перетину.

ЛАБОРАТОРНА РОБОТА №8
РОЗВ'ЯЗУВАННЯ ЗАДАЧ БЛИЗЬКОСТІ ОБ'ЄКТІВ

МЕТА: Ознайомитися із типовими полями та методами класу Line (пряма). Опанувати створення відповідного класу на мові програмування (за вибором). Оволодіти практичними навиками при використанні даних типу TPoint, TSegment,TLine.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Рівняння прямої, що проходить через точки P1(x1,y1) та P2(x2,y2) має вигляд:

$$\frac{x-x1}{x2-x1}=\frac{y-y1}{y2-y1}.$$

Загальне рівняння прямої має вигляд $ax+by+c=0$.

Для прямої P1P2 маємо:

$a=y2-y1$, $b=-(x2-x1)$, $c=y1x2-x1y2$.

Якщо дві прямі задані загальними рівняннями:

$L1: a1x+b1y+c1=0$, $L2: a2x+b2y+c2=0$, то координати точки їх перетину визначаються із системи рівнянь:

$$\begin{cases} a1x+b1y=-c1 \\ a2x+b2y=-c2 \end{cases},$$

тобто $x=\frac{-(c1b2-c2b1)}{a1b2-b1a2}$, $y=\frac{-(a1c2-a2c1)}{a1b2-b1a2}$, якщо тільки $a1b2-a2b1\neq 0$.

Пряма P1P2 може задаватися параметрично:

$\begin{cases} x=x1+t(x2-x1) \\ y=y1+t(y2-y1) \end{cases}$, $t\in R$.

Якщо ж $t\in[0;1]$, то останнє рівняння задає відрізок P1P2.

ХІД РОБОТИ

1) Завантажте середовище візуального програмування Delphi. Створіть у папці власної групи папку і запишіть у неї новий проект.
2) Задайте властивості форми BorderStyle значення bsDialog. Створіть кнопку Close для закриття програми, яку розташуйте в правому нижньому куті майбутнього вікна.
3) Опишіть класи TPoint, TSegment із наступними полями та методами (можете скористатися результатами попередніх лабораторних робіт):

```
TPoint1 = object
      x,y : integer;
      Procedure Init(a,b:integer);
      Procedure Draw(C:TColor);
    end;

TSegment = object
      pp,pk : TPoint1;
      Procedure Init(p1,p2 : TPoint1);
      Procedure Draw(C:TColor);
      function Length(s:TSegment):real;
      end;
```

4) Створіть клас Line1 як нащадок класу TSegment:

```
TLine1 = object(TSegment)
     a,b,c : real;
     Procedure Init(p1,p2:TPoint1);
     Function Dist(p:TPoint1):real;
     end;
```

Опишіть відповідні методи:

```
Procedure TLine1.Init(p1,p2:TPoint1);
    Begin
    pp:=p1;pk:=p2;
    a:=pk.y-pp.y;
    b:=-(pk.x-pp.x);
    c:=pp.y*pk.x-pp.x*pk.y;
    end;
 Function TLine1.Dist(p:TPoint1):real;
     Begin
     Dist:=(a*p.x+b*p.y+c)/sqrt(a*a+b*b);
     end;
```

5) Додайте до методів класу TLine1 малювання прямої, заданої параметрично;

```
Procedure TLine1.Draw(P:TColor);
    Begin
    Form1.Canvas.Pen.Color:=P;
    Form1.Canvas.Brush.Color:=P;
    Form1.Canvas.MoveTo(pp.x-100*(pk.x-pp.x),pp.y-100*(pk.y-pp.y));
    Form1.Canvas.LineTo(pp.x+100*(pk.x-pp.x),pp.y+100*(pk.y-pp.y));
    end;
```

Завдання для самостійної роботи

1) Запрограмуйте можливість задання на екрані однієї прямої , що проходить через дві точки P1 і P2. Виведіть у відповідні поля редагування (із відповідними підписами) координати цих точок.
2) Запрограмуйте виведення у поле редагування відстані від точки до прямої – спочатку мишкою задаєте пряму, вказавши точки P1 і P2, а потім мишкою задаєте потрібну точку P3.

Примітка Відстань виводити із знаком, що буде вказувати на положення точки відносно прямої.

3) Запрограмуйте розв'язання наступної задачі:

Задано пряму P1P2. На екрані «кинуто» довільним чином 100 точок синього кольору. Знайти ту, яка найближче знаходиться до прямої P1P2 і «виділити» її більшим кругом (радіус 4 пікселі) червоного кольору.

Примітка. Довільно кинуті точки запам'ятайте у масиві і в ньому знайдіть точку з мінімальною відстанню до прямої P1P2.

ЛАБОРАТОРНА РОБОТА №9
ПЕРЕТИН ГЕОМЕТРИЧНИХ ОБ'ЄКТІВ

МЕТА: Ознайомитися із особливостями перетину геометричних об'єктів, прямих, променів, відрізків. Оволодіти практичними навиками роботи із різними формами запису геометричних об'єктів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Рівняння прямої, що проходить через точки P1(x1,y1) та P2(x2,y2) має вигляд:

$$\frac{x - x1}{x2 - x1} = \frac{y - y1}{y2 - y1}.$$

Загальне рівняння прямої має вигляд $ax + by + c = 0$.

Для прямої P1P2 маємо:

$a = y2 - y1$, $b = -(x2 - x1)$, $c = y1x2 - x1y2$.

Якщо дві прямі задані загальними рівняннями:

$L1: a1x + b1y + c1 = 0$, $L2: a2x + b2y + c2 = 0$, то координати точки їх перетину визначають із системи рівнянь: $\begin{cases} a1x + b1y = -c1 \\ a2x + b2y = -c2 \end{cases}$,

тобто $x = \frac{-(c1b2 - c2b1)}{a1b2 - b1a2}$, $y = \frac{-(a1c2 - a2c1)}{a1b2 - b1a2}$, якщо тільки $a1b2 - a2b1 \neq 0$.

Пряма P1P2 може задаватися параметрично: $\begin{cases} x = x1 + t(x2 - x1) \\ y = y1 + t(y2 - y1) \end{cases}$, $t \in R$.

Якщо ж $t \in [0;1]$, то останнє рівняння задає відрізок P1P2.

ХІД РОБОТИ

1) Завантажте середовище візуального програмування Delphi. Створіть у папці власної групи папку “FAM_2_1” (де FAM – Ваше прізвище) і запишіть у неї новий проект під назвою “Program_1”, назву pas-файлу задайте самостійно.

2) Задайте властивості форми BorderStyle значення bsDialog. Створіть кнопку Close для закриття програми, яку розташуйте в правому нижньому куті майбутнього вікна.

3) Скористайтеся описами класів TPoint, TSegment, TLine1 з попередньої лабораторної роботи:

4) Запрограмуйте можливість виведення на екран тільки двох прямих, які визначаються відрізками P1P2 та P3P4 за допомогою мишки.

5) Додайте до методів класу TLine1 функцію Function .Intersect(L:TLine1):TPoint1, яка визначає точку перетину прямих, наприклад:

```
Function TLine1.Intersect(L:TLine1):TPoint1;
   var x1,y1,x2,u1,u2:real;
  Begin
  y1:=a*L.b-b*L.a;
  x1:= -(c*L.b-b*L.c);
    x2:= -(a*L.c-c*L.a);
  u1:=x1/y1; u2:=x2/y1;
  Intersect.x:=trunc(u1);
  Intersect.y:=trunc(u2);
 end;
```

Визначте точку перетину прямих P1P2 та P3P4 та виділіть її на екрані.

6) Виведіть координати точки перетину на екран (якщо вона існує).

7) Запрограмуйте можливість виведення на екран трьох прямих L1, L2, L3, кожна з яких задається з допомогою мишки двома точками, що належать лише одній прямій

8) Визначте точки перетину прямих L1 і L2, L1 і L3, L2 і L3, запам’ятайте їх у масив. Виділіть точки перетину прямих на екрані та виведіть їх координати у відповідні поля редагування.

9) Замалюйте блакитним кольором трикутник, що утворився в результаті перетину прямих L1, L2, L3.

Зауваження. Стандартна процедура Polygon в якості аргументу використовує масив елементів типу TPoint, а не типу TPoint1 .

10) Запрограмуйте виведення на екран однієї прямої L та відрізка S. Перевірте, чи перетинає відрізок пряму та виведіть відповідне повідомлення на екран.

11) Визначте точку перетину відрізків P1P2 та P3P4 та виділіть її на екрані, якщо це можливо. Виведіть відповідне повідомлення про перетин відрізків на екран.

ЛАБОРАТОРНА РОБОТА №10
ЗАДАЧІ ПРИНАЛЕЖНОСТІ

МЕТА: Ознайомитися із особливостями розв'язання задач приналежності точки відрізку, прямій, опуклому та довільному многокутнику. Набути практичних навичок при реалізації алгоритмів розв'язування задач приналежності .

ТЕОРЕТИЧНІ ВІДОМОСТІ

Рівняння прямої, що проходить через точки P1(x1,y1) та P2(x2,y2) має вигляд:

$$\frac{x-x1}{x2-x1}=\frac{y-y1}{y2-y1}.$$

Загальне рівняння прямої має вигляд $ax+by+c=0$.

Для прямої P1P2 маємо:

$a=y2-y1$, $b=-(x2-x1)$, $c=y1x2-x1y2$.

Якщо дві прямі задані загальними рівняннями:

$L1: a1x+b1y+c1=0$, $L2: a2x+b2y+c2=0$, то координати точки їх перетину визначзок системи рівнянь:

$$\begin{cases} a1x+b1y=-c1 \\ a2x+b2y=-c2 \end{cases},$$

тобто $x=\frac{-(c1b2-c2b1)}{a1b2-b1a2}$, $y=\frac{-(a1c2-a2c1)}{a1b2-b1a2}$, якщо тільки $a1b2-a2b1\neq 0$.

Пряма P1P2 може задаватися параметрично:

$\begin{cases} x=x1+t(x2-x1) \\ y=y1+t(y2-y1) \end{cases}$, $t\in R$.

Якщо ж $t\in[0;1]$, то останнє рівняння задає відрізок P1P2.

Перевірка приналежності точки трикутнику

Якщо точка P розміщена одночасно зліва відносно всіх сторін трикутника, то вона знаходиться всередині трикутника.

Перевірка приналежності точки многокутнику

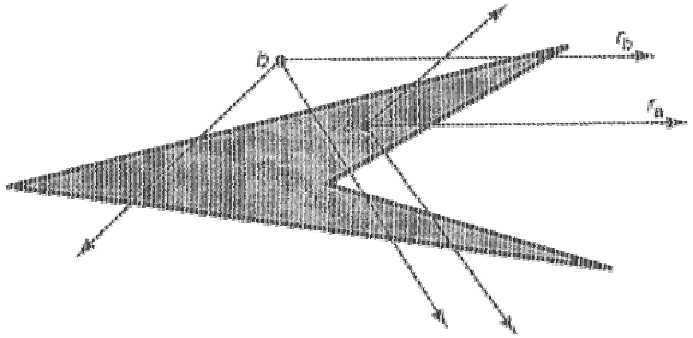
Випадок опуклого многокутника

Для довільного ***опуклого*** многокутника перевіряється той факт, що точка лежить по одну сторону від всіх ребер полігону. Якщо це не так, то вона зовні. У загальному випадку такий спосіб не працює.

Многокутник опуклий, якщо всі пари суміжних ребер додатно орієнтовані.

Випадок довільного многокутника.
Метод трасування променя

Припустимо, що нам необхідно визначити приналежність точки a полігону p.



Кожний промінь, що виходить із внутрішньої точки a, перетинає межу непарне число разів, а кожний промінь, із зовнішньої точки b, має парне число перетинів з межею полігону

ХІД РОБОТИ

1) Завантажте середовище візуального програмування Delphi. Створіть у папці власної групи папку "FAM_2_1" (де FAM – Ваше прізвище) і запишіть у неї новий проект під назвою "Program_1", назву pas-файлу задайте самостійно.

2) Задайте властивості форми BorderStyle значення bsDialog. Створіть кнопку Close для закриття програми, яку розташуйте в правому нижньому куті майбутнього вікна.

3) Скористайтеся описами класів TPoint, TSegment, TLine1 з попередніх лабораторних робіт.

4) Запрограмуйте можливість виведення на екран трикутника (потрібно малювати і сторони і вершини) заданням мишкою його вершин: вершини мають ініціалізуватися як об'єкти класу TPoint1, а сторони трикутника – як об'єкти класу TSegment.

5) Задайте метод класу TPoint, який би реалізовував визначення положення точки відносно прямої (Classify або Orientation). Наприклад:

```
Function TPoint1.Classify(p1,p2 : TPoint1):string;
var sa : real; a,b:TPoint1;
Begin
a:=p2.Minus(p1);
b.x:=x-p1.x; b.y:=y-p1.y,
sa:=a.x*b.y-b.x*a.y,
 if sa>0 then Classify:='Left'
  else
    if sa<0 then Classify:='Right'
     else
      if(a.x*b.x<0) or (a.y*b.y<0) then Classify:='Behind'
       else
        if a.Length<b.Length then Classify:='Beyond'
         else
          if (p1.x=x) and (p1.y=y) then Classify:='BeginPoint'
           else
            if (p2.x=x) and (p2.y=y) then Classify:='EndPoint'
             else Classify:='Between';
end;
```

6) Задайте на екрані довільну точку, виділіть її іншим кольором. Виведіть на екран повідомлення про приналежність точки заданому трикутникові.

7) Створіть на основі даного проекту новий проект, в якому задайте можливість виведення на екран довільного чотирикутника за допомогою мишки (ініціалізація вершин і сторін чотирикутника – аналогічна, як для трикутника).

8) Задайте довільним чином п'яту точку. Запрограмуйте алгоритм визначення приналежності точки опуклому чотирикутнику, задавши точку зовні/всередині чотирикутника. Перевірте, чи виконується алгоритм для чотирикутника із самоперетинами.

9) Запрограмуйте алгоритм визначення приналежності точки довільному чотирикутнику:

- Перш за все, проведіть горизонтальну пряму через задану точку.
- Обчисліть кількість точок перетину прямої і сторін, які (точки) лежать зліва від заданої точки.

Для цього необхідно скористатися методами визначення перетину прямої і відрізка! Можлива наступна реалізація:

```
Function TLine1.Peretyn(s:TSegment):boolean;
  Begin
   if s.pk.Classify(pp,pk)=s.pp.Classify(pp,pk) then
      Peretyn:=False else Peretyn:=True;
  End;
```

- Якщо кількість точок перетину непарна, то точка лежить всередині чотирикутника.

10) Запрограмуйте виведення на екран довільних 20 точок та визначте, скільки з них належить заданому чотирикутникові.

ЛІТЕРАТУРА

1. Анісімов А.В., Терещенко В. М, Кравченко І.В Основні алгоритми обчислювальної геометрії. Навчально-методичний посібник Київ, 2001 70 с.
2. Препарата Ф., Шеймос М. Вычислительная геометрия:Введение. М.: Мир, 1989. – 480с.
3. Графика в байтах:(Сборник статей/ Ред. Б.Васильева) М.: Знание, 1991. – 46с.
4. Машинная графика и геометрия: (Сборник статей/ Б.Васильева) М.: Знание, 1991. – 46 с.
5. Новейшая энциклопедия персонального компьютера. Москва, 2003
6. Програмное обеспечение персональных ЭВМ.: (Справочное пособие/ под ред. А.А.Стогния) К.: Наукова думка, 1989. – 367с.
7. Глинський Я.М. Практикум з інформатики: Навч. посібник, 6-е вид Львів: Деол, 2003. – 224с.
8. Грайс Д. Графические средства персонального комп’ютера.:пер. с англ.. М.: Мир, 1989. – 375 с.
9. Дудка О.М., Дудка В.В., Томич М.В. Машинна графіка. Графічні редактори.: Навчальний посібник. Івано-Франк., 1996. – 70 с.
10. Дудка О.М. Комп’ютерна графіка: Навчальний посібник. Івано-Франк., 2003. – 48с.
11. Загляднов И.Ю., Касаткин В.Н. Построение изображений на экране персональной ЭВМ. К.: Техника, 1990. – 117 с.
12. Павлидис С.Т. Алгоритмы машинной графики и обработки изображений.: пер. с англ. М.: Радио и связь, 1989. – 400 с.
13. Прокудин Г.С., Вольська С.Ю. Інженерна та комп’ютерна графіка: методичні вказівки до виконання практичних завдань та самостійної роботи студентів. Київ, вид. Європ.ун-ту, 2002. – 70 с.
14. Руденко В.Д., Макарчук О.М., Патланжоглу М.О. Практичний курс інформатики /За ред. Мадзігона В.М. К.: Фенікс, 1997. – 304 с.
15. Хирн Д.,Бейкер М. Микрокомпьютерная графика.: пер. с англ., под ред. Шаньгина. М.: Мир., 1987. – 351 с.