МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи «Графіка в Delphi»
з навчальної дисципліни «Інформаційні технології»

Харків
2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторної роботи «Графіка в Delphi»
з навчальної дисципліни «Інформаційні технології»
для студентів усіх форм навчання
за спеціальністю 131 «Прикладна механіка»

Затверджено
редакційно-видавничою
радою університету,
протокол №3 від 24.10.2024 р.

Харків
НТУ «ХПІ»
2024

Методичні вказівки до виконання лабораторної роботи «Графіка в Delphi» з навчальної дисципліни «Інформаційні технології» для студентів усіх форм навчання за спеціальністю 131 «Прикладна механіка» / уклад.: Т. Є. Третяк, О. Л. Мироненко, С. О. Мироненко. – Харків: НТУ «ХПІ», 2024. – 33 с.

Укладачі: Т. Є. Третяк
О. Л. Мироненко
С. О. Мироненко

Рецензент О. В. Кобець

Кафедра інтегрованих технологій машинобудування» ім. М. Ф. Семка

ЗМІСТ

ВСТУП

Навчальна дисципліна «Інформаційні технології» є вибірковим компонентом освітньо-професійної програми «Технологічні та логістичні системи у машинобудуванні» для підготовки здобувачів вищої освіти першого (бакалаврського) рівня за спеціальністю 131 «Прикладна механіка».

Метою викладання навчальної дисципліни є ознайомлення здобувачів з призначенням, можливостями та сучасним станом розвитку інформаційних технологій, формування знань, умінь та навичок ефективного застосування сучасних інформаційних технологій та програмного забезпечення для автоматизації інженерної діяльності, розв'язання різноманітних практичних задач за фахом.

У результаті вивчення навчальної дисципліни студент отримує відомості про архітектуру персонального комп'ютера IBM PC, принципи функціонування інформаційної системи та програмного забезпечення ЕОМ, основи роботи у середовищі об'єктно-орієнтованого та візуального програмування Delphi, сучасні засоби, прийоми і методи розробки програмного забезпечення, основні концепції та методологію об'єктно-орієнтованого програмування (ООП), принципи побудування графічних зображень, засоби обробки та аналізу інформації із застосуванням сучасних програмних засобів.

При виконанні лабораторної роботи «Графіка в Delphi» з навчальної дисципліни «Інформаційні технології» студенти набувають практичні навички розробки, налагодження та виконання в середовищі Delphi програм, що реалізують побудову графічних зображень.

Методичні вказівки до виконання лабораторної роботи містять формулювання завдання, вихідні дані за варіантами, теоретичні відомості за даною темою, докладний опис ходу виконання завдання на конкретному прикладі та правила оформлення звіту до лабораторної роботи. Також у документ включено список питань для самоперевірки знань студентів та список літератури, рекомендованої для вивчення.

МЕТА ВИКОНАННЯ РОБОТИ

Метою виконання лабораторної роботи на тему «Графіка в Delphi» є знайомство студентів із середовищем об’єктно-орієнтованого програмування Delphi, мовою програмування Object Pascal, технологіями візуального та подієвого програмування, графічними можливостями середовища Delphi, засобами та методами побудови графічних зображень на екрані монітора.

1. ФОРМУЛЮВАННЯ ЗАВДАННЯ

Задано типи геометричних фігур та їх взаємне розташування.

Вихідні дані за номерами варіантів наведено у розділі 2. Номер варіанта завдання відповідає номеру прізвища студента у журналі групи.

У середовищі Delphi необхідно розробити програму, що здійснює креслення та заливання кольором заданих геометричних фігур на екрані монітора. Рекомендованою версією середовища програмування є версія Delphi 7.

Передбачити можливість введення вихідних даних на основній формі програми, а виведення графіки – на додаткову форму.

2. ВИХІДНІ ДАНІ ЗА ВАРІАНТАМИ

Таблиця 2.1 – Вихідні дані відповідно до заданих варіантів

Номер варіанта	Завдання
1	Виконати креслення двох трикутників, одного всередині іншого, за заданими координатами їх вершин. Залити кольором внутрішній трикутник.
2	Виконати креслення двох трикутників, що перетинаються, за заданими координатам їх вершин. Залити кольором область перетину трикутників.
3	Виконати креслення двох кіл з загальним центром, одного всередині іншого, за їх заданими радіусами. Залити кольором зовнішнє коло.
4	Виконати креслення двох кіл, що перетинаються, за їх заданими радіусами. Залити кольором область, що покривається колами.
5	Виконати креслення двох прямокутників, одного всередині іншого, за заданими довжинами їх сторін. Залити різним кольором зовнішній та внутрішній прямокутники.
6	Виконати креслення двох прямокутників, що перетинаються, за заданими довжинами їх сторін. Залити кольором область перетину прямокутників.
7	Виконати креслення двох п’ятикутників, одного всередині іншого, за заданими координатами їх вершин. Залити кольором зовнішній п’ятикутник.
8	Виконати креслення двох п’ятикутників, що перетинаються, за заданими координатами їх вершин. Залити кольором область, що покривається п’ятикутниками.
9	Виконати креслення двох еліпсів, одного всередині іншого, за їхніми заданими півосями. Залити кольором внутрішній еліпс.
10	Виконати креслення двох еліпсів, що перетинаються, за їх заданими півосями. Залити кольором область перетину еліпсів.
11	Виконати креслення квадрата за заданою довжиною його боку. Розділити квадрат діагональною лінією на два трикутники і залити кольором один із них.
12	Виконати креслення квадрата за заданою довжиною його сторони. Розділити квадрат вертикальною лінією на два рівні прямокутники та залити їх різними кольорами.
13	Виконати креслення квадрата за заданою довжиною його сторони. Розділити квадрат двома діагональними лініями на чотири трикутники та залити кольором два протилежні трикутники.

Продовження таблиці 2.1

Номер варіанта	Завдання
14	Виконати креслення кола за його заданим радіусом. Розділити коло горизонтальною лінією на два півкола та залити один із них.
15	Виконати креслення кола за його заданим радіусом. Розділити коло вертикальною та горизонтальною лініями на чотири рівні сектори та залити їх різними кольорами.
16	Виконати креслення півкола за його заданим радіусом. Розділити півколо вертикальною лінією на два рівні сектори і залити їх різними кольорами.
17	Виконати креслення квадрата, вписаного в коло, за заданими довжиною сторони квадрата та радіусом кола. Залити кольором квадрат.
18	Виконати креслення квадрата, вписаного в інший квадрат, за заданими довжинами їх сторін. Залити кольором зовнішній квадрат.
19	Виконати креслення кола, вписаного в квадрат, за заданим радіусом кола та довжиною сторони квадрата. Залити кольором коло.
20	Виконати креслення двох прямокутників із загальною вершиною при внутрішньому торканні за заданими довжинами їх сторін. Залити кольором внутрішній прямокутник.
21	Виконати креслення двох трикутників із загальною вершиною при зовнішньому торканні за заданими координатами їх вершин. Залити кольором обидва трикутники.
22	Виконати креслення двох кіл з загальною точкою при зовнішньому торканні за їх заданими радіусами. Залити кола різними кольорами.
23	Виконати креслення двох прямокутників із загальною вершиною при зовнішньому торканні за заданими довжинами їх сторін. Залити прямокутники різними кольорами.
24	Виконати креслення двох кіл з загальною точкою при внутрішньому торканні за їх заданими радіусами. Залити кольором зовнішнє коло.
25	Виконати креслення півкола та квадрата, що мають загальну лінію зовнішнього торкання, за заданими радіусом півкола та довжиною сторони квадрата. Залити півколо та квадрат різними кольорами.

3. ТЕОРЕТИЧНІ ВІДОМОСТІ

Графічні можливості Delphi:

1. Особливості викреслювання на полотні Delphi ліній та контурів.

Середовище Delphi дозволяє розробляти програми, що виводять графіку на поверхню об’єкта (форми ***Form*** або області ілюстрації ***Image***). ***Поверхні об’єкта*** відповідає властивість ***Canvas*** – абстрактне ***полотно***, що складається з ***пікселів***.

Положення пікселя характеризується його горизонтальною (x) і вертикальною (y) координатами, що є значеннями цілого типу. Ліва верхня точка полотна має координати (0,0). Координати зростають зверху вниз і зліва направо. Значення координат правої нижньої точки полотна залежать від розміру полотна (рис. 3.1).

Розмір полотна можна отримати, звернувшись до властивостей ***Height*** і ***Width*** області ілюстрації ***Image*** або до властивостей ***ClientHeight*** і ***Clientwidth*** форми ***Form***.

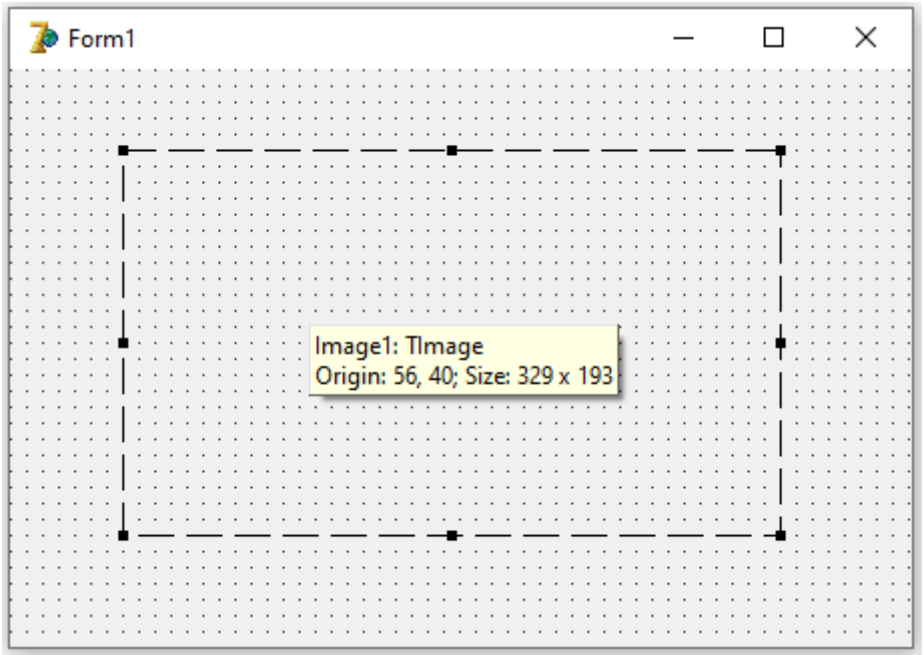


Рисунок 3.1 – Координати точок полотна

Методи полотна забезпечують виведення графічних примітивів на його поверхню. Ці методи використовують внутрішні ***властивості полотна*** – ***Pen (Олівець)*** і ***Brush (Пензель)***, які дозволяють задавати характеристики графічних примітивів, що виводяться.

Вид ліній та контурів, що викреслюються на полотні, визначають такі внутрішні ***властивості олівця***:

- ***Color (Колір лінії)*** **–** може приймати значення однієї з іменованих констант, визначених у Delphi для позначення кольорів:

  clblack – чорний;

  clmaroon – каштановий;

  clgreen – зелений;

  clolive – оливковий;

  clnavy – темно-синій;

  clpurple – рожевий;

  clteal – зелено-голубий;

  clgray – сірий;

  clsilver – сріблястий;

  clred – червоний;

  cllime – салатний;

  clblue – синій;

  clfuchsia – яскраво-рожевий;

  claqua – бірюзовий;

  clwhite – білий;
- ***Width (Товщина лінії)*** – задається кількістю пікселів;
- ***Style (Стиль лінії)*** – може приймати значення однієї з іменованих констант, визначених у Delphi для позначення стилів лінії:

  psSolid – суцільна лінія;

  psDash – пунктирна лінія з довгих штрихів;

  psDot – пунктирна лінія з коротких штрихів;

  psDashDot – пунктирна лінія з довгого і короткого штрихів, що чергуються;

psDashDotDot – пунктирна лінія з одного довгого і двох коротких штрихів, що чергуються;

psClear – лінія не відображається;

- ***Mode (Режим відображення лінії)*** **–** може приймати значення однієї з іменованих констант, визначених у Delphi для позначення режимів відображення лінії:

  pmBlack – чорний колір лінії незалежно від значення властивості ***Color***;

  pmWhite – білий колір лінії незалежно від значення властивості ***Color***;

  pmCopy – колір лінії визначається значенням властивості ***Color***;

  pmNotCopy – колір лінії є інверсним по відношенню до кольору, що визначається значенням властивості ***Color***;

  pmNot – колір точки лінії є інверсним по відношенню до кольору точки полотна, в яку виводиться точка лінії.

2. Особливості заливання на полотні Delphi областей, обмежених контурами.

Вид заливання областей, обмежених контурами, на полотні визначають такі внутрішні ***властивості пензля***:

- ***Color (Колір заливання)*** **–** може приймати значення однієї з іменованих констант, визначених у Delphi для позначення кольорів (див. раніше);
- ***Style (Стиль заливання)*** **–** може приймати значення однієї з іменованих констант, визначених у Delphi для позначення стилів заливання:

  bsSolid – суцільне заливання;

  bsClear– заливання не відображається;

  bsHorizontal – горизонтальне штрихування;

  bsVertical – вертикальне штрихування;

bsFDiagonal – діагональне штрихування з нахилом ліній вперед;

bsBDiagonal – діагональне штрихування з нахилом ліній назад;

bsCross – горизонтально-вертикальне штрихування у клітину;

bsDiagCross – діагональне штрихування у клітину.

3. Виведення тексту на полотно Delphi.

Для ***виведення тексту*** на полотно використовується метод

TextOut (*x, y, Текст*);

де *x*, y – координати точки полотна, від якої виконується виведення тексту;

Текст – змінна або константа рядкового типу, значення якої виводиться.

Цей метод використовує внутрішню ***властивість полотн***а – ***Font (Шрифт)***, що дозволяє задавати характеристики шрифту, що використовується під час виведення тексту.

Характеристики шрифту визначають такі внутрішні ***властивості шрифту***:

- ***Name (Назва шрифту) –*** може приймати значення однієї з назв шрифтів, визначених у Windows;
- ***Size (Розмір шрифту) –*** задається кількістю пунктів;
- ***Style (Стиль шрифту) –*** може приймати значення однієї з іменованих констант, визначених у Delphi для позначення стилів шрифту:

  fsBold – напівжирний шрифт;

  fsItalic– курсив;

  fsUnderline – підкреслений шрифт;

  fsStrikeOut – перекреслений шрифт;

ця властивість є множиною, може бути укладена у квадратні дужки і складатися з кількох розділених комами значень наведених вище іменованих констант, що дозволяє комбінувати необхідні стилі;

- ***Color (Колір символів)*** **–** може приймати значення однієї з іменованих констант, визначених у Delphi для позначення кольорів (див. раніше).

Область виведення тексту зафарбовується поточним кольором кисті. Тому перед виведенням тексту властивості ***Brush.Color*** потрібно привласнити значення ***bsClear*** або задати колір кисті, співпадаючий з кольором поверхні, на яку виводиться текст.

Наступний фрагмент програми демонструє використання методу ***Textout*** для виведення тексту на поверхню форми:

```
with Form1.Canvas do
 begin
 {Встановити характеристики шрифту}
  Font.Name := ‘Arial’;
  Font.Size := 20;
  Font.Style := [fsltalic, fsBold];
 {Область виведення тексту не зафарбовується}
  Brush.Style := bsClear;
  TextOut(0, 10, ‘Borland Delphi 7’);
 end;
```

Після виведення тексту методом ***Textout*** покажчик виводу (позиція олівця) переміщається в правий верхній кут області виведення тексту.

4. Викреслювання прямої та ламаної ліній на полотні Delphi.

Для ***викреслювання прямої лінії*** на полотні використовується метод

LineTo (*x, y*);

де *x, y* – координати точки полотна, до якої викреслюється лінія від позиції олівця.

Переміщення позиції олівця у необхідну точку виконує метод

MoveTo (*x, y*);

де *x, y* – координати нової позиції олівця.

Для ***викреслювання ламаної лінії*** на полотні використовується метод

PolyLine (*Масив координат точок*);

де *Масив координат точок* – змінна типу масив записів типу ***TPoint***, що містять координати *x* та *y* точок, які з'єднує ламана лінія.

Наступний фрагмент програми демонструє використання методу ***PolyLine*** для викреслювання ламаної лінії через три точки на поверхні форми:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  pol: array [1..3] of TPoint; // координати точок
begin
  {Заповнюємо елементи масиву записів}
  pol[1].x := 10;
  pol[1].y := 50;
  pol[2].x := 40;
  pol[2].y := 10;
  pol[3].x := 70;
  pol[3].y := 50;
  Form1.Canvas.PolyLine(pol);
end;
```

5. Викреслювання еліпса, кола, їх дуг або секторів на полотні Delphi.

Для ***викреслювання еліпса або кола*** на полотні використовується метод

Ellipse (*x1, y1, x2, y2*);

де *x1, y1* – координати лівого верхнього кута прямокутника (або квадрата), усередині якого викреслюється еліпс (або коло);

x2, y2 – координати правого нижнього кута прямокутника (або квадрата), усередині якого викреслюється еліпс (або коло) (рис. 3.2).

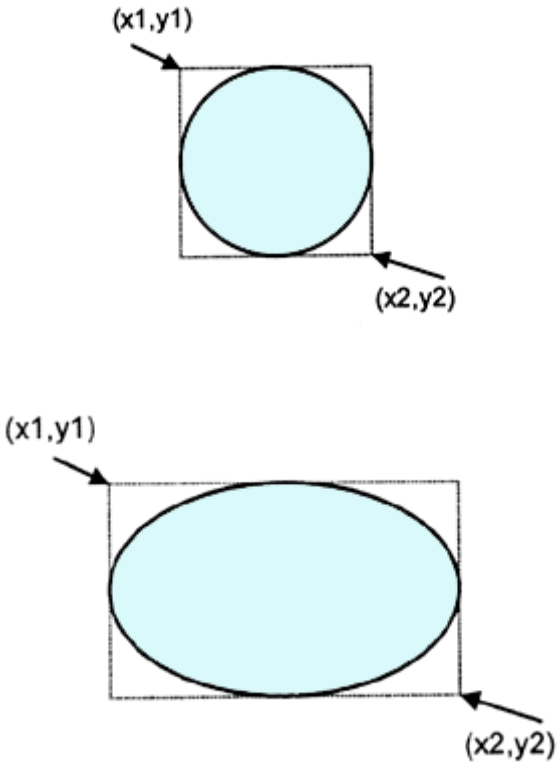


Рисунок 3.2 – Параметри методу *Ellipse*

Для ***викреслювання дуг еліпса або кола*** на полотні використовується метод

Arc (*x1*, *y1*, *x2*, *y2*, *x3*, *y3*, *x4*, *y4*);

де *x1*, *y1* – координати лівого верхнього кута прямокутника (або квадрата), усередині якого викреслюється дуга еліпса (або кола);

x2, *y2* – координати правого нижнього кута прямокутника (або квадрата), усередині якого викреслюється дуга еліпса (або кола);

x3, *y3* – координати початкової точки дуги еліпса (або кола);

x4, *y4* – координати кінцевої точки дуги еліпса (або кола) (рис. 3.3).

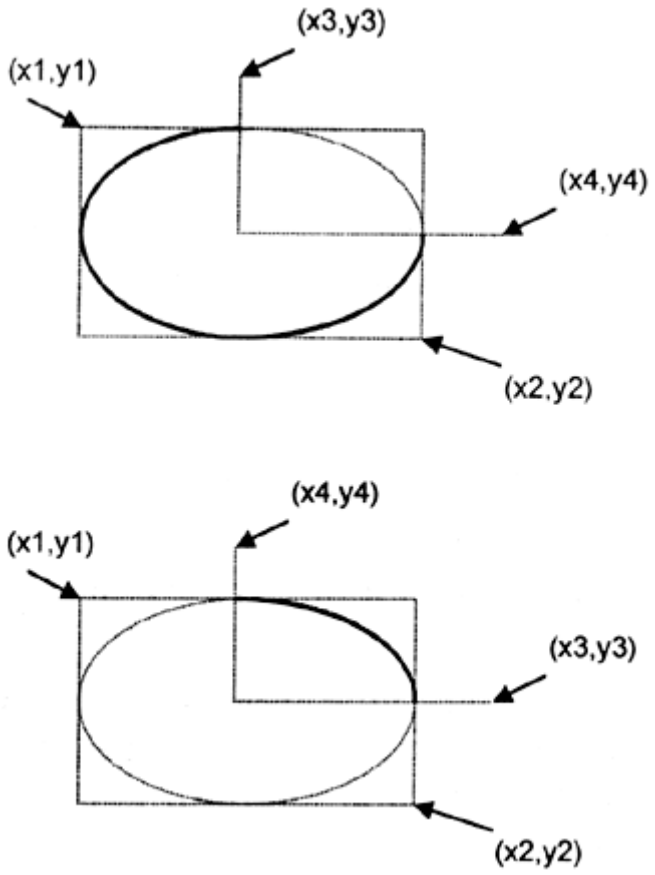


Рисунок 3.3 – Параметри методу *Arc*

Для ***викреслювання секторів еліпса або кола*** на полотні використовується метод

Pie (*x1, y1, x2, y2, x3, y3, x4, y4*);

де *x1, y1* – координати лівого верхнього кута прямокутника (або квадрата), усередині якого викреслюється сектор еліпса (або кола);

x2, y2 – координати правого нижнього кута прямокутника (або квадрата), усередині якого викреслюється сектор еліпса (або кола);

x3, y3, x4, y4 – координати кінцевих точок прямих, що є межами сектора еліпса (або кола), початкові точки прямих збігаються з центром еліпса (або кола) (рис. 3.4).

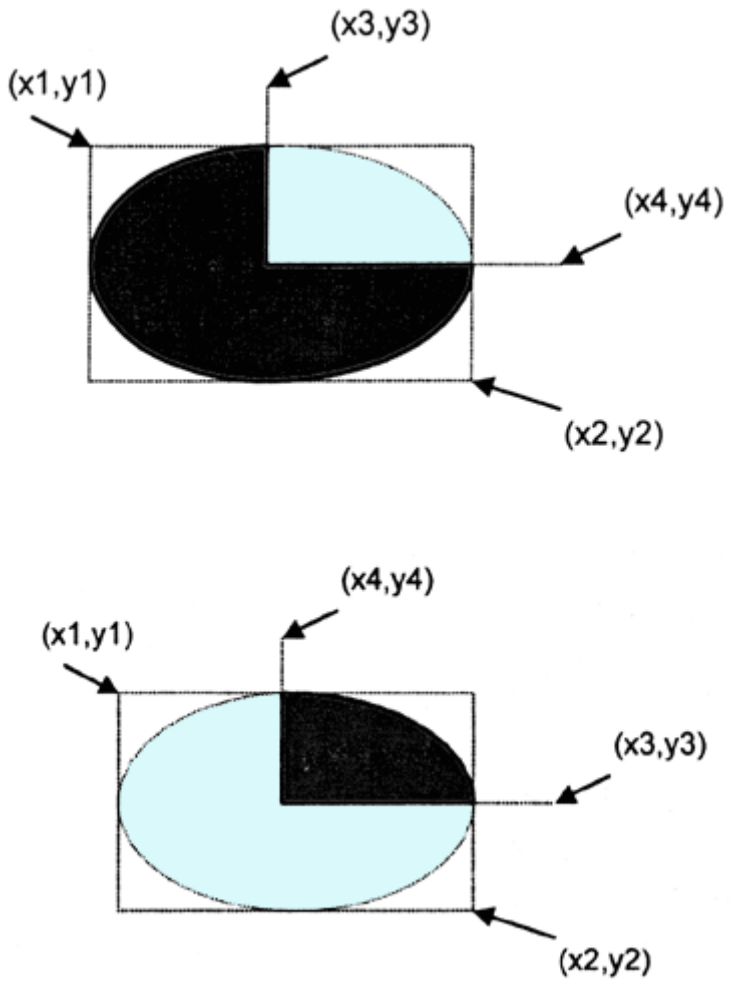


Рисунок 3.4 – Параметри методу *Pie*

6. Викреслювання прямокутника та багатокутника на полотні Delphi.

Для ***викреслювання прямокутника*** на полотні використовується метод

Rectangle (*x1*, *y1*, *x2*, *y2*);

де *x1*, y1 – координати лівого верхнього кута прямокутника;

x2, y2 – координати правого нижнього кута прямокутника.

Для ***викреслювання прямокутника із заокругленими кутами*** на полотні використовується метод

RoundRec (*x1*, *y1*, *x2*, *y2*, *x3*, *y3*);

де *x1*, *y1* – координати лівого верхнього кута прямокутника;

x2, y2 – координати правого нижнього кута прямокутника;

x3, y3 – розміри еліпса, одна чверть якого використовується для викреслювання округленого кута прямокутника (рис. 3.5).

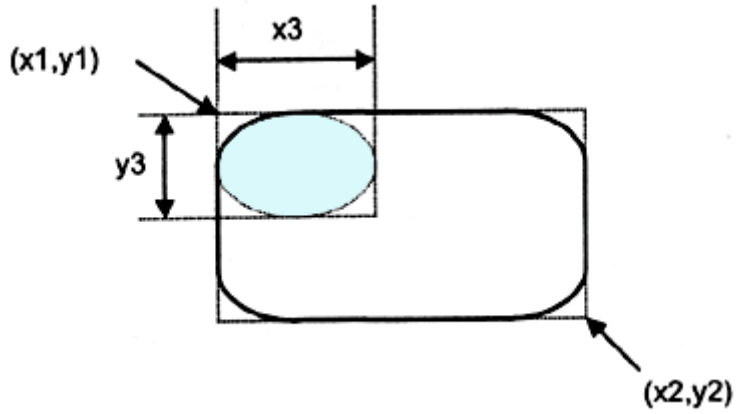


Рисунок 3.5 – Параметри методу *RoundRec*

Існує два методи, які викреслюють прямокутник не олівцем, а пензлем.

Для ***викреслювання зафарбованого прямокутника*** пензлем використовується метод

FillRect (*Координати кутів прямокутника*);

де *Координати кутів прямокутника* – структура типу ***TRect***, що містить координати *x1* та *y1* лівого верхнього кута прямокутника та координати *x2* та *y2* правого нижнього кутів прямокутника.

Для ***викреслювання контуру прямокутника*** пензлем використовується метод

FrameRect (*Координати кутів прямокутника*);

де *Координати кутів прямокутника* – структура типу ***TRect***, що містить координати *x1* та *y1* лівого верхнього кута прямокутника та координати *x2* та *y2* правого нижнього кутів прямокутника.

Структура типу ***TRect*** може бути заповнена за допомогою функції

Rect (*x1, y1, x2, y2*).

Наступний фрагмент програми демонструє використання методів ***FillRect*** і ***FrameRect*** для викреслювання прямокутника з червоною заливкою і прямокутника з зеленим контуром на поверхні форми:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  r1, r2: TRect; // координати кутів прямокутників
begin
  {Заповнюємо поля структури}
  r1 := Rect(20,20,60,40);
  r2 := Rect(10,10,40,50);
  with form1.Canvas do
    begin
      Brush.Color := clRed;
      FillRect(r1); // зафарбовуємо прямокутник
      Brush.Color := clGreen;
      FrameRect(r2); // зафарбовуємо тільки контур прямокутника
    end;
end;
```

Для ***викреслювання багатокутника*** на полотні використовується метод

Polygon (*Масив координат точок*);

де *Масив координат точок* – змінна типу масив записів типу ***TPoint***, що містять координати *x* та *y* точок, які є вершинами багатокутника.

* На відміну від методу ***PolyLine***, що використовується для викреслювання ламаної лінії через задані точки (див. раніше), метод ***Polygon*** викреслює ламану лінію та з’єднує першу та останню точки ламаної.

Наступний фрагмент програми демонструє використання методу ***Polygon*** для викреслювання трикутника на поверхні форми:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  pol: array [1..3] of TPoint; // координати точок трикутника
begin
  {Заповнюємо елементи масиву записів}
  pol[1].x := 10;
  pol[1].y := 50;
  pol[2].x := 40;
  pol[2].y := 10;
  pol[3].x := 70;
  pol[3].y := 50;
  Form1.Canvas.Polygon(pol);
end;
```

7. Заливання замкнутих областей на полотні Delphi.

Для ***заливання замкнутих областей*** на полотні використовується метод

FloodFill (*x, y, Колір, Спосіб заливання*);

де *x, y* – координати точки полотна всередині замкнутої області, від якої виконується заливання;

Колір – колір, який використовується при визначенні границі області, що заливається;

Спосіб заливання – вказує, як саме по цьому кольору визначається границя, він може приймати значення однієї з іменованих констант, визначених у Delphi для позначення способів заливання:

fsSurface – заливання всіх точок області, в яких колір дорівнює кольору, заданому третім параметром методу;

fsBorder – заливання всіх точок області, у яких колір не дорівнює кольору, заданому третім параметром методу.

8. Фарбування точки на полотні Delphi.

Інформація про колір кожної точки полотна містить внутрішня ***властивість полотна Pixels (Пікселі)***, яка є двовимірним масивом елементів типу ***TColor***.

Для ***фарбування точки*** на полотні використовується виклик

Pixels[*x, y*]:=*Колір*;

де *x, y* – координати точки полотна, що фарбується;

Колір – новий колір точки полотна, що фарбується, він може приймати значення однієї з іменованих констант, визначених у Delphi для позначення кольорів (див. раніше).

Підключення до проекту Delphi додаткової форми:

Для ***підключення до проекту додаткової форми*** необхідно:

1) викликати команду меню ***File/New/Form*** для додавання на екрані вікна додаткової форми та відкриття у вікні коду додаткової вкладки файлу ***unit2.pas***;
2) підключити модуль додаткової форми ***unit2.pas*** до модуля форми ***unit1.pas***, вказавши його ім’я ***unit2*** у блоці ***uses*** розділу оголошень ***interface*** в тексті файлу ***unit1.pas***.

Під час роботи додатку для виведення на передній план вікна додаткової форми використовується метод форми ***Show***, який робить форму видимою.

Для закриття вікна додаткової форми використовується метод ***Close***. При цьому слід зважати на те, що основна форма проекту є батьківською, а додаткова форма – дочірньою. Тому закриття додаткової форми виконує повернення до основної форми, а закриття основної форми призводить до закриття додатку.

4. ПРИКЛАД ВИКОНАННЯ ЗАВДАННЯ

Вихідні дані:

Виконати креслення квадрата за заданою довжиною його сторони та трикутника за заданими координатами його вершин. Залити кольорами квадрат та трикутник.

Хід виконання завдання:

1. Малюємо на папері ескіз заданих квадрата та трикутника (рис. 4.1). На ескізі вказуємо всі необхідні розміри та параметри геометричних фігур – для квадрата це координати його вершини (x0, y0) та довжина сторони *a*, для трикутника це координати його вершин (x1, y1), (x2, y2) та (x3, y3).

Початок системи координат та напрямки осей координат вибираємо такими, що співпадають з прийнятими в комп'ютерній графіці. Виділяємо всі характерні точки профілів.

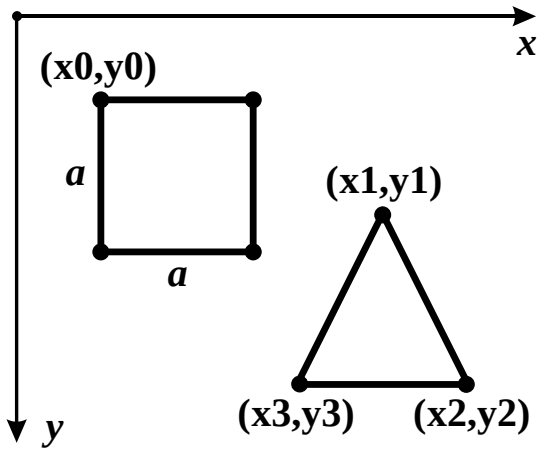


Рисунок 4.1 – Ескіз квадрата та трикутника

2. У середовищі Delphi розробляємо програму, що здійснює креслення та заливання кольором заданих геометричних фігур на екрані монітора:

● Згідно з завданням реалізуємо введення вихідних даних на основній формі програми, а виведення графіки – на додаткову форму.

Підключаємо до проекту додаткову форму програми. Для цього:

– викликаємо команду меню ***File/New/Form*** для додавання на екрані вікна додаткової форми та відкриття у вікні коду додаткової вкладки файлу ***unit2.pas***;

– підключаємо модуль додаткової форми ***unit2.pas*** до модуля форми ***unit1.pas***, вказавши його ім’я ***unit2*** у блоці ***uses*** розділу оголошень ***interface*** в тексті файлу ***unit1.pas***.

● Налаштовуємо стартову форму – заготовку вікна програми (рис. 4.2). На етапі налаштування форми вводимо початкові значення розмірів та параметрів геометричних фігур, змінюючи значення властивостей ***Text*** об’єктів ***Edit***.

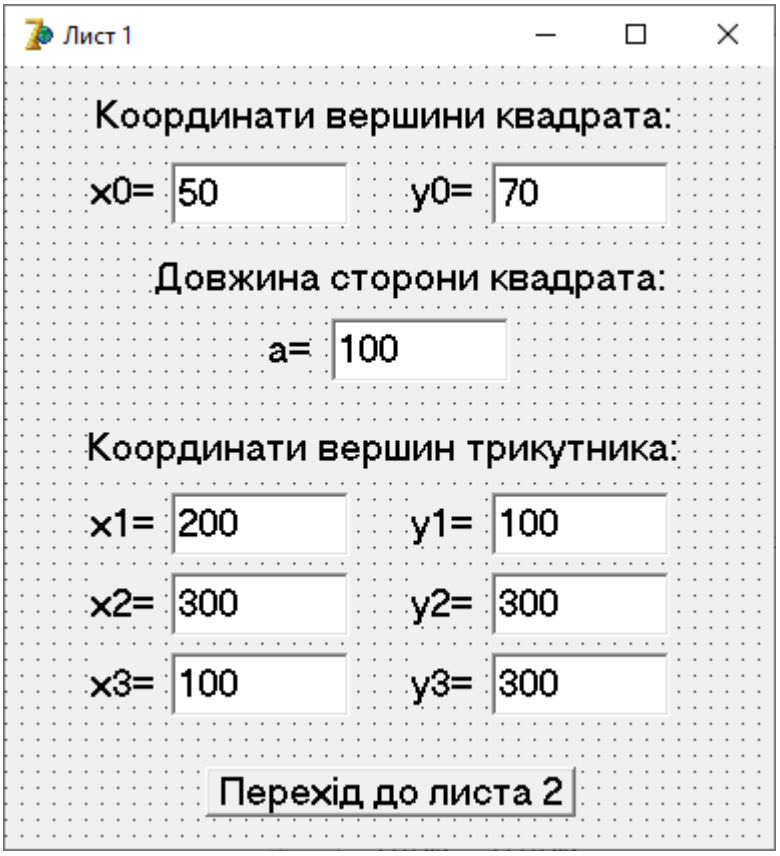


Рисунок 4.2 – Налаштування стартової форми програми

● Налаштовуємо додаткову форму програми (рис. 4.3). При цьому для пояснювальних написів замість об’єктів ***Label*** краще використовувати об’єкти ***StaticText*** (вкладка палітри компонентів ***Additional***). При очищенні поверхні форми методом ***FillRect*** об’єкти ***Label*** заливаються, тоді як об’єкти ***StaticText*** та інші об’єкти – ні.

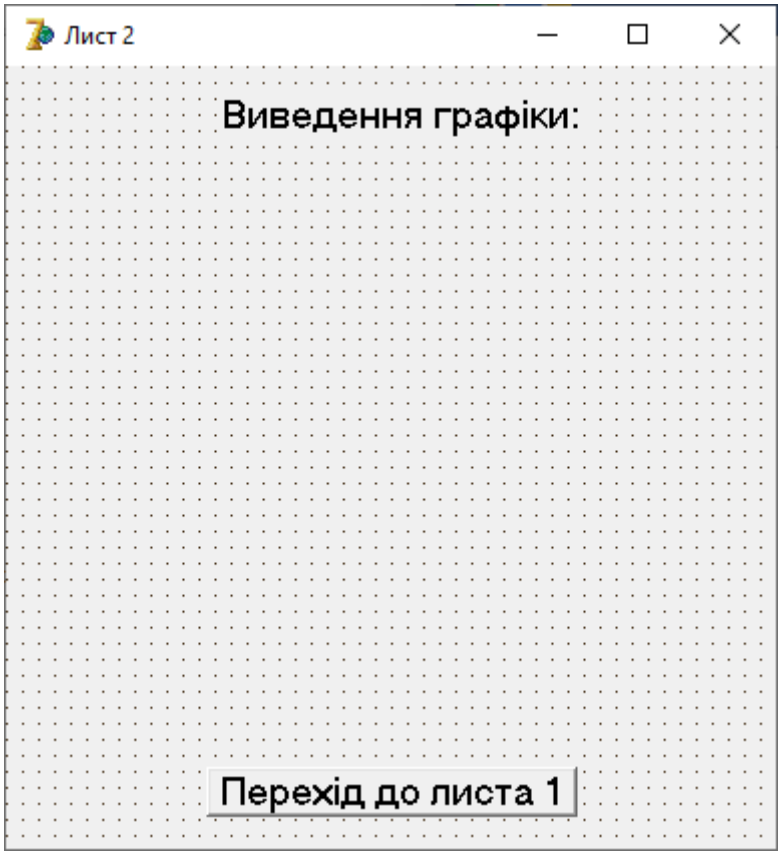


Рисунок 4.3 – Налаштування додаткової форми програми

● Доповнюємо шаблони файлів ***Unit1.pas*** та ***Unit2.pas***.

Доповнений шаблон файлу ***Unit1.pas*** має наступний вигляд:

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Unit2;

type
  TForm1 = class(TForm)
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Edit8: TEdit;
    Edit9: TEdit;
    Button1: TButton;
    Edit7: TEdit;
```

```
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var
  a,x0,y0:integer;
  x,y:array [1..3] of integer;
  p:array [1..3] of TPoint;
begin
  {Координати вершини квадрата}
  x0:=StrToInt(Edit1.Text);
  y0:=StrToInt(Edit2.Text);
  {Довжина сторони квадрата}
  a:=StrToInt(Edit3.Text);

  {Координати вершин трикутника}
  x[1]:=StrToInt(Edit4.Text); y[1]:=StrToInt(Edit5.Text);
  x[2]:=StrToInt(Edit6.Text); y[2]:=StrToInt(Edit7.Text);
  x[3]:=StrToInt(Edit8.Text); y[3]:=StrToInt(Edit9.Text);

  Form2.Show;
  with Form2.Canvas do
```

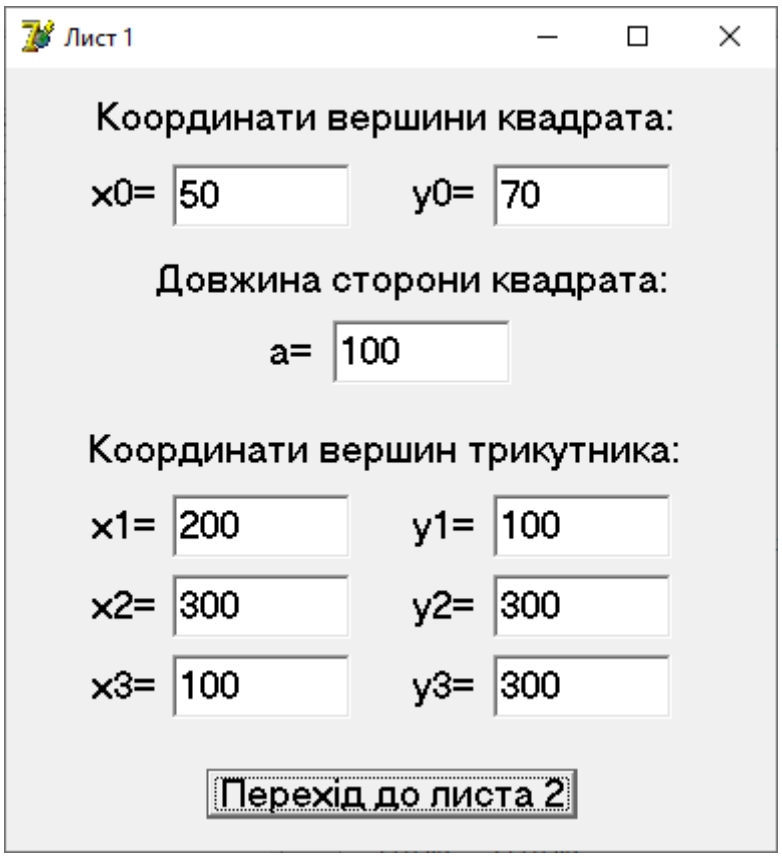
```
  begin
    {Очищення поверхні форми}
    Brush.Color:=clBtnFace;
    FillRect(Rect(0,0,Form2.ClientWidth,Form2.ClientHeight));

    Pen.Color:=clRed;

    {Малювання квадрата}
    Rectangle(x0,y0,x0+a,y0+a);

    {Малювання трикутника}
    p[1].X:=x[1];  p[1].Y:=y[1];
    p[2].X:=x[2];  p[2].Y:=y[2];
    p[3].X:=x[3];  p[3].Y:=y[3];
    Polygon(p);

    {Заливання квадрата}
    Brush.Style:=bsSolid;
    Brush.Color:=clBlue;
    FloodFill(x0+3,y0+3,clRed,fsBorder);

    {Заливання трикутника}
    Brush.Color:=clGreen;
    FloodFill(x[1],y[1]+3,clRed,fsBorder);
  end
 end;

end.
```

Доповнений шаблон файлу ***Unit2.pas*** має наступний вигляд:

```
unit Unit2;

interface
```

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

type
  TForm2 = class(TForm)
    Button1: TButton;
    StaticText1: TStaticText;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
  Form2.Close
end;

end.
```

4. Запускаємо програму на виконання, вводимо вихідні дані та отримуємо результат роботи програми (рис. 4.4).



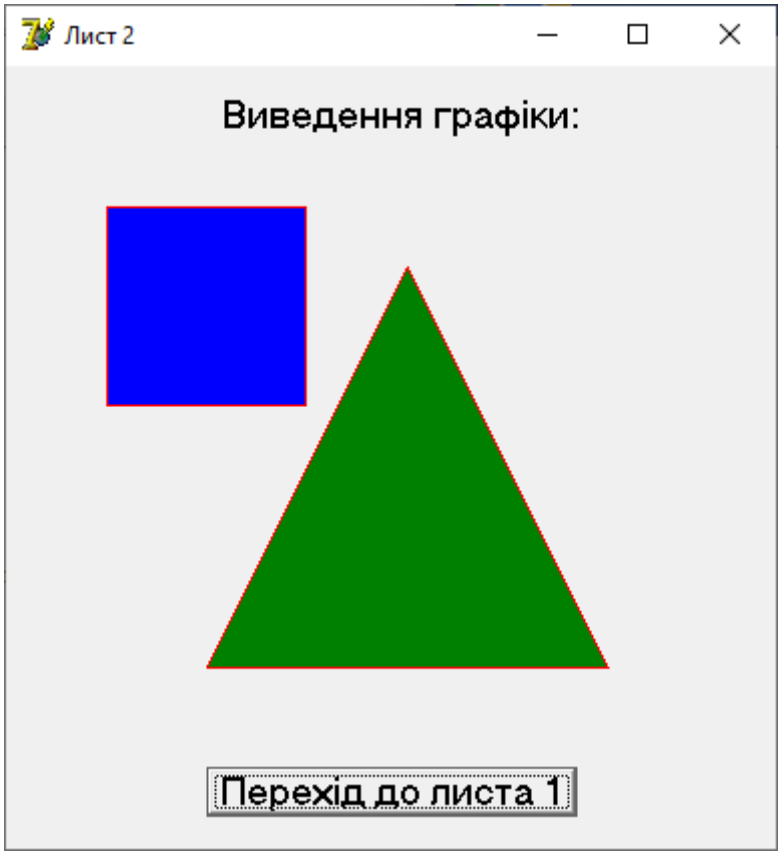


Рисунок 4.4 – Результат роботи програми побудови креслення квадрата та трикутника

5. ОФОРМЛЕННЯ ЗВІТУ ДО ЛАБОРАТОРНОЇ РОБОТИ

Звіт до лабораторної роботи необхідно оформити на аркушах формату А4.

Звіт повинен містити:

– титульний лист (приклад його оформлення показано на рис. 5.1);

– формулювання завдання (див. розділ 1);

– вихідні дані відповідно до заданого варіанту (див. розділ 2);

– хід виконання завдання (див. розділ 4):

1) ескіз заданих геометричних фігур;

2) текст розробленої програми побудови креслення заданих геометричних фігур на екрані монітора (вміст файлів ***Unit1.pas*** та ***Unit2.pas***);

4) результат роботи програми (зображення вікна програми для ведення розмірів та параметрів заданих геометричних фігур та вікна програми з виведеним кресленням геометричних фігур).

Міністерство освіти та науки України
Національний технічний університет
«Харківський політехнічний інститут»

Лабораторна робота
з навчальної дисципліни «Інформаційні технології»
на тему **«Графіка в Delphi»**

Варіант 1

Виконав
студент гр. МІТ-222а
Іванов І.І.

Перевірила
доц., к.т.н. Третяк Т.Є.

Харків
2024

Рисунок 5.1 – Приклад оформлення титульного листа звіту
до лабораторної роботи

ПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ ЗНАНЬ

1. Основні характеристики середовища програмування Delphi.
2. Елементи екрана середовища програмування Delphi.
3. Поняття об’єкта та події в середовищі програмування Delphi.
4. Створення стартової форми додатку Delphi, основні компоненти форми.
5. Базовий набір подій Delphi.
6. Поняття процедури обробки події в середовищі програмування Delphi.
7. Створення програмного коду додатка Delphi.
8. Сукупність модулів, що складають проект Delphi.
9. Призначення та структура головного модуля проекту Delphi.
10. Призначення та структура модуля форми проекту Delphi.
11. Особливості креслення на полотні Delphi ліній і контурів.
12. Особливості зафарбовування на полотні Delphi областей, обмежених контурами.
13. Вивід тексту на полотно Delphi.
14. Креслення прямої та ламаної ліній на полотні Delphi.
15. Креслення еліпса та окружності на полотні Delphi.
16. Креслення дуг еліпса та окружності на полотні Delphi.
17. Креслення секторів еліпса та окружності на полотні Delphi.
18. Креслення прямокутника або багатокутника на полотні Delphi.
19. Фарбування крапки на полотні Delphi.
20. Правила оголошення та використання в середовищі Delphi користувальницьких підпрограм.
21. Правила оголошення та використання в середовищі Delphi користувальницьких модулів.
22. Особливості взаємодії основної та додаткової форм проекту Delphi.
23. Поняття класу, об’єкта та метода Delphi.

СПИСОК ЛІТЕРАТУРИ

1. Ковалюк Т. В. Основи програмування: підручник. – К.: Видавнича група BHV, 2005. – 384 с.

2. Шаховська Н. Б., Голощук Р. О. Алгоритми і структури даних: посіб. – Львів: Магнолія, 2010. – 215 с.

3. Мироненко А. С. Імперативне та об'єктно-орієнтоване програмування на Turbo Pascal та Delphi. Глибоке занурення. – Одеса: ВМВ, 2007. – 408 с.

4. Безменов М. І. Основи програмування в середовищі Delphi: навч. посіб. – Харків: НТУ «ХПІ», 2010. – 608 с.

5. Кащеєв Л. Б., Коваленко С. В., Коваленко С. М. Інформатика. Основи візуального програмування [Електронний ресурс]: навч. посіб. – Харків: Веста, 2011. – 192 с.

6. Яременко Г. І., Копил Д. В. Програмування в середовищі Delphi: навч. посіб. – Черкаси: ЧДТУ, 2011. – 188 с.

7. Методичні вказівки до виконання Розрахунково-графічного завдання з навчальної дисципліни «Інформаційні технології» для студентів усіх форм навчання за спеціальністю 131 «Прикладна механіка» / уклад.: Т. Є. Третяк, О. Л. Мироненко, С. О. Мироненко. – Харків: НТУ «ХПІ», 2024. – 37 с.

Навчальне видання

Методичні вказівки
до виконання лабораторної роботи «Графіка в Delphi»
з навчальної дисципліни «Інформаційні технології»
для студентів усіх форм навчання
за спеціальністю 131 «Прикладна механіка»

Укладачі:
ТРЕТЯК Тетяна Євгенівна
МИРОНЕНКО Олександр Леонідович
МИРОНЕНКО Сергій Олександрович

Відповідальний за випуск Третяк Т. Є.
Роботу до видання рекомендував проф. Клочко О. О.

В авторській редакції

План 2024 р., поз. 961.

Підп. до друку 15.11.2024 р. Гарнітура Times New Roman. Ум. друк. арк. 1,5.

Видавничий центр НТУ «ХПІ», вул. Кирпичова, 2, м. Харків, 61002
Свідоцтво про державну реєстрацію ДК № 5478 від 21.08.2017 р.

Електронне видання