Практична робота №12

Використання вбудованих процедур і функцій

Мета роботи: освоїти можливості Delphi в рисуванні найпростіших графічних зображень.

Завдання. Побудувати на формі зображення координатних осей, конверта і піраміди, графіка функції, використовуючи графічні можливості Delphi.

Теоретичні відомості

У Windows для позначення канви, на якій виконується рисунок, використовується *термін контекст* пристрою (DC). Взаємодія з контекстами пристроїв на рівні викликів функцій API може виявитися доволі складною. Тому спочатку потрібно одержати в системі Windows дескриптор контексту пристрою (посилання на формальний опис можливостей пристрою), після чого вибрати в нього потрібні вам об'єкти (пера, пензлі або шрифти). Тільки потім можна на ньому щось рисувати. Після закінчення рисування, перш ніж видалити контекст пристрою, потрібно звільнити обрані в нього об'єкти. В іншому випадку пам'ять, яка використовується у вашому додатку, ніколи не буде повернута системі.

Наявний клас TCanvas із бібліотеки візуальних компонентів значно полегшує роботу з контекстами пристроїв. Як властивість *(property)* він включений у багато компонентів.

Розглянемо основні властивості та методи класу TCanvas (табл. 1 і 2).

Основні властивості класу TCanvas

Таблиця 1

Властивість	Опис
Brush	Задає колір заливки та шаблон у ході зафарбовування фігур
ClipRect	Являє собою поточну прямокутну область відсікання канви. Усі операції графічного виведення обмежені заданим прямокутником. Властивість тільки для читання
CopyMode	Визначає, як відбуватиметься копіювання (метод CopyRect) на дану канву зображення з іншого місця: один до одного, з інверсією зображення тощо

Font	Містить шрифт, який клас TCanvas використовує у виведенні тексту (метод TextOut)
Handle	Дескриптор (HDC) канви. Ця властивість дає можливість прямо викликати функції API Windows
Pen	Визначає стиль і колір проведених ліній
PenPos	Задає координати *x* і *y* для чергової операції рисування
Pixels	Являє собою двовимірний масив пікселів об'єкта Canvas

Основні методи класу TCanvas

Таблиця 2

Властивість	Опис
Arc	Рисує на канві поточним пером сегмент еліпса
BrushCopy	Рисує бітову матрицю з прозорим тлом
CopyRect	Копіює на канву частину зображення
Draw	Копіює зображення з пам'яті на канву
El 1 ipse	Рисує поточним пером еліпс і зафарбовує його поточним пензлем
FloodFi11	Здійснює заливку області канви поточним пензлем
LineTo	Проводить поточним пером лінію з даної точки в точку з координатами *x* і *y*
MoveTo	Встановлює поточне положення пера
Pie	Рисує на канві сектор еліпса
Polygon	За допомогою масиву об'єктів типу TPoint будує многокутник і зафарбовує його поточним пензлем
Polyline	За допомогою масиву об'єктів типу TPoint рисує поточним пером ламану лінію. Автоматичного замикання контуру при цьому не відбувається
Rectangle	Рисує на канві поточним пером прямокутник і зафарбовує його поточним пензлем
RoundRect	Рисує зафарбований прямокутник із закругленими кутами
StretchDraw	Копіює бітову матрицю з пам'яті на канву. При цьому відповідно до цільової прямокутної області відбувається масштабування (розтягнення або стиснення) растрових зображень
TextExtent	Повертає висоту й ширину рядка Text у пікселах. Ураховується поточний шрифт канви
TextHight	Повертає висоту рядка Text у пікселах. Ураховується поточний шрифт канви
TextOut	Виводить рядок Text за допомогою поточного шрифту в заданому місці канви
TextRect	Виводить текст із відсіканням. При цьому частина тексту, що потрапила за межі прямокутної області відсікання, стає невидимою

Перелічені властивості й методи становлять лише малу частину можливостей контекстів пристроїв Windows. Але навіть ця мала частина може задовольнити 80 % потреб, що виникають зазвичай у ході роботи з графікою.

В інтерфейсі графічних пристроїв (GDI) Windows є велика кількість типів об'єктів, які впливають на роботу контекстів пристроїв. Найчастіше з об'єктів Graphics Device Interface використовуються пера, пензлі та шрифти. Менш популярні графічні об'єкти — палітри, бітові матриці й області відсікання.

Розглянемо можливості вкладеного об'єкта «перо» *(pen)*.

За допомогою пер можна рисувати різноманітні лінії: як окремі, що сполучають одну точку з іншою, так і ті, що обмежують різні геометричні фігури: прямокутники, еліпси й многокутники.

Для доступу до пера використовується властивість Pen класу TCanvas, що являє собою об'єкт типу TPen. У табл. 3 перелічено властивості класу TPen. Методів і подій, які заслуговують на згадування, у цьому класі немає.

Властивості класу TPen

Властивість	Опис
Color	Задавання кольору лінії
Handle	Являє собою дескриптор пера (HPEN). Використовується у випадках прямих викликів відповідних функцій API
Mode	Визначає спосіб прорисовки ліній (нормальний, інверсний, з виключним або **(хог)** тощо)
Style	Визначає стиль лінії (суцільний, пунктирний, штриховий, штрихпунктирний, прозорий і т. д.)
Width	Задавання товщини лінії в пікселах

Здебільшого використання цих власти востей не викликає ніяких труднощів.

Рис. 1. Виведення ліній на формі

Хід роботи

1. Запустимо середовище Delphi, створимо новий додаток, розмістимо на формі компонент «кнопка». Напишемо невелику програму, що виводить на екрані дві лінії — горизонтальну й вертикальну. При цьому лінії мають бути пунктирними (стилі psDash і psDot), а їхній колір — синім. Позначимо координати початку й кінця ліній (рис. 1).
2. Використовуючи роботу з об'єктом Canvas, задамо стилі, колір ліній і проведемо їх. Оформимо виведення значень координат.
3. В обробник події OnClick помістимо код:

```
procedure TForml.ButtonlClick(Sender: TObject);
begin
    Canvas.Pen.Color := clBlue;
    Canvas.Pen.Style := psDash;  // вибір стилю лінії
    Canvas.MoveTo(50, 50);        // перехід робочої точки
    Canvas.LineTo(200, 50);       // провести лінію в точку
                                  //(200,50)
    Canvas.Pen.Style := psDot;   // вибір стилю лінії
    Canvas.MoveTo(50, 50);       // перехід робочої точки
    Canvas.LineTo(50, 200);      // провести лінію в точку
                                  // (50, 200)
    canvas.TextOut(30,30,'50, 50'); // виведення тексту
                                  //в зазначених координатах

    canvas.TextOut(180, 30/200, 50');

    canvas.TextOut(30, 200,'50, 200');

end;
```

У попередніх завданнях ми виводили текст і рисунок у вікні програми. Однак якщо нам необхідно сховати вікно, а потім знову його активізувати, ми виявимо, що рисунок на формі зник. Це відбувається тому, що наш рисунок поки що має тимчасовий характер. Щоб зробити його постійним, слід помістити наведений вище код в обробник події OnPaint форми. Тепер

у разі потреби перерисувати вікно генеруватиметься подія OnPaint, і рисунок буде відновлено.

Штриховий і пунктирний стилі можна використовувати тільки в роботі з пером завтовшки 1 піксел. За допомогою стилю psClear ви можете усунути лінії, що їх Windows рисує по зовнішній межі таких об'єктів, як, наприклад, прямокутники, еліпси й зафарбовані многокутники.

4. Збережемо проект на диску.
5. Створимо новий додаток для роботи з пензлями (brush).

Теоретичні відомості

Пензлі визначають спосіб зафарбовування фігур. За допомогою поточного пензля заповнюється внутрішня область кожної з нарисованих вами фігур: еліпсів, прямокутників, многокутників тощо. Пензель може

рисувати штрихові лінії або бітову матрицю. Його зовнішній вигляд можна контролювати за допомогою властивості Brush класу TCanvas, що є, у свою чергу, об'єктом класу TBrush. Як і TPen, TBrush не містить методів і подій для програмування.

Властивості класу TBrush перелічено в табл. 4.

Властивості класу TBrush

Таблиця 4

Властивість	Опис
Bitmap	Ідентифікує бітову матрицю, що використовується як тло пензля. У Windows 95 бітова матриця повинна була мати розміри 8x8
Color	Встановлює колір пензля
Handle	Являє собою дескриптор пензля (HBRUSH). Використовується у випадках прямих викликів функцій API
Style	Визначає стиль пензля (суцільний, прозорий або одна з можливих штриховок)

Властивість Style за замовчуванням має значення bsSolid (суцільна заливка). Якщо в зафарбовуванні геометричних фігур ви хочете використати штриховку, то властивості Style слід присвоїти один із таких стилів: bsHorizontal, bsVertical, bsFDiagonal, bsBDiagonal, bsCross.

6. Продемонструємо перелічені способи заливки (у зазначеному порядку) за допомогою невеликої програми, що рисує прямокутники. Винесемо на форму кнопку. (Інтерфейс завдання подано на рис. 2.)

Рис.2. Способи заливки

7. В обробник події OnClick цієї кнопки впишемо програмний код:

```
procedure TForml.ButtonlClick(Sender: TObject);
begin
    canvas.pen.Width:=2; canvas.Brush.Color:=clblack;
    canvas.Brush.Style:=bsHorizontal;
    canvas.rectangle(10,10,80,120); canvas.Brush.Style:=bsVertical;
    canvas.rectangle(90,10,160,120);
    canvas.Brush.Style:=bsFDiagonal;
    canvas.rectang1e(170,10,240,120);
     canvas.Brush.Style:=bsBDiagonal;
     canvas.rectang1e(250,10,320,120);
     canvas.Brush.Style:=bsCross;
     canvas.rectang1e(330,10,400,120);
     canvas.Brush.Style:=bsDiagCross;
     canvas.rectang1e(410,10,480,120);
end;
```

Коли ви використовуєте пензель-штриховку, його властивість Color визначає колір штрихових ліній. У процесі зафарбовування графічних об'єктів за допомогою такого пензля VCL автоматично встановлює режим фона в Transparent (наприклад прозорий). Це означає, що колір фона пензля буде збігатися з кольором вікна, на якому нарисований графічний об'єкт.

8. Збережемо проект на диску.

9. Створимо новий додаток. Трохи ускладнимо завдання. Намалюємо на формі конверт і піраміду. Приклад того, як мають виглядати конверт і піраміда у вікні чинної програми, показано на рис 3.

Рис. 3. Побудова конверта й піраміди

10. За аналогією з попереднім завданням впишемо в кнопку обробник події:

```
procedure TForml.ButtonlClick(Sender: TObject);
var
   i:integer;
begin
  Canvas.Pen.width := 2;
   Canvas.Rectang1e(30,30,180,130);
   Canvas.MoveTo(30,30);
   Canvas.LineTo(105,80);
   Canvas.LineTo(180,30);
   for i:=0 to 4 do
     Canvas.Rectang1e(300-20* i,30+20* i,400+20* i,50+20* i);
end;
```

11. Збережемо проект на диску.

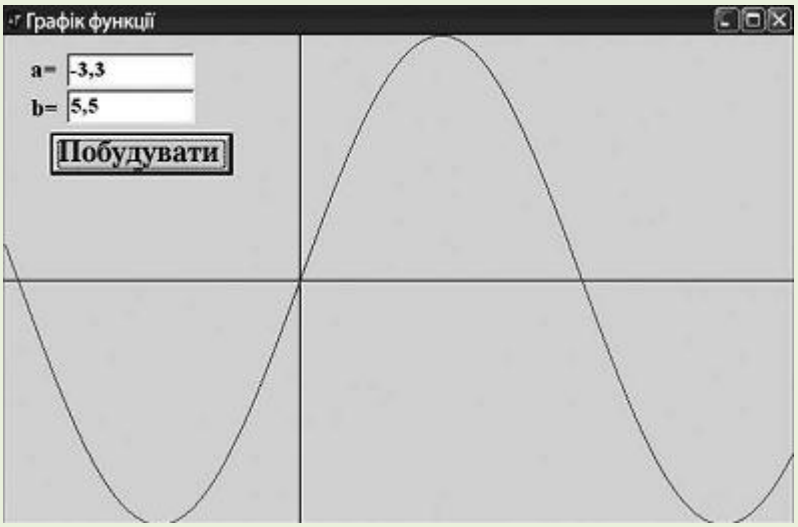

Рис.4. Побудова графіка функції y = sin(x). (рис. 4).

12. Створимо новий додаток. Іще раз ускладнимо завдання. Побудуємо на формі графік функції y = sin(x).

У ході побудови потрібно ввести ліву і праву межі інтервалу (дійсні числа a і b), виконати масш- табування графіка функції по осях *Ox* і *Oy*

Програма матиме такий вигляд:

```
unit Unitl; {Графік функції}

interface

uses Windows, Messages, Syslltils, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls;

type
   TForml = class(TForm)
     Editl: TEdit;
     Edit2: TEdit;
     Buttonl: TButton;
     Labell: TLabel;
     Label2: TLabel;
     procedure ButtonlClick(Sender: TObject);
private
   { Private declarations }
public
   { Public declarations }
end;


var
 Forml: TForml;
implementation
 {$R *.DFM}
function f(x:real):real;              // функція побудови
begin
  result:=sin(x);
end;


procedure TForml.ButtonlClick(Sender: TObject);
var
     xl,x2,max,min,shagx,shagy:real; i,sh:integer;
begin
     xl:=strtofloat(editl.text);                  // ліва межа
     x2:=strtofloat(edit2.text);                  // права межа
     shagx:=(x2-xl)/forml.CIientWidth;       // масштабування
```

```
                                        // по осі Ox
  min:=f(xl);                  // початкові значення для мінімального
  max:=f(xl);                  // і максимального значень функції
  for i:=1 to forml.CIientWidth do
  begin
    if f(xl+i*shagx)<min then     // пошук мінімального
      min:=f(xl+i*shagx);         // і максимального
                                  // значень функції
    if f(xl+i*shagx)>max then
     max:=f(xl+i*shagx);
 end;
 shagy:=form1.clientheight/(max-min);
                                        // масштабування по осі Oy
 if x1*x2 < 0 then                      // малювання осі Oy
 begin
 canvas.moveto(round(abs(xl)*forml.clientwidth/ (x2-xl)),0);
    canvas.1ineto(round(abs(xl)*forml.clientwidth/ (x2-xl)),
    forml.clientheight);
 end;
 if min*max<0 then                  // малювання осі Ox
 begin
    canvas.moveto(0,forml.clientheight -
    round(abs(min)*forml.clientheight/(max-min)));
    canvas.lineto(forml.clientwidth,forml.clientheight -
    round(abs(min)*forml.clientheight/(max-min)));
  end;
  canvas.pen.Color:=rgb(255,0,0);       // колір графіка функції
  canvas.moveto(l,round(forml.clientheight-(f(xl+l*shagx)- min)*shagy) );
  for i:=2 to forml.ClientWidth do     // побудова графіка
       canvas.1ineto(i,round(forml.clientheight- (f(xl+i*shagx)-
       min)*shagy) );
end;
end.
```

ПРАКТИЧНЕ ЗАВДАННЯ

Нарисувати на формі один із пропонованих прапорців (рис. 5). Кольори елементів прапорця узгодити з викладачем. У ході рисування використати умовні оператори та цикли, оскільки надалі можлива модифікація прапорця — збільшення кількості його елементів (кружків, квадратиків тощо).

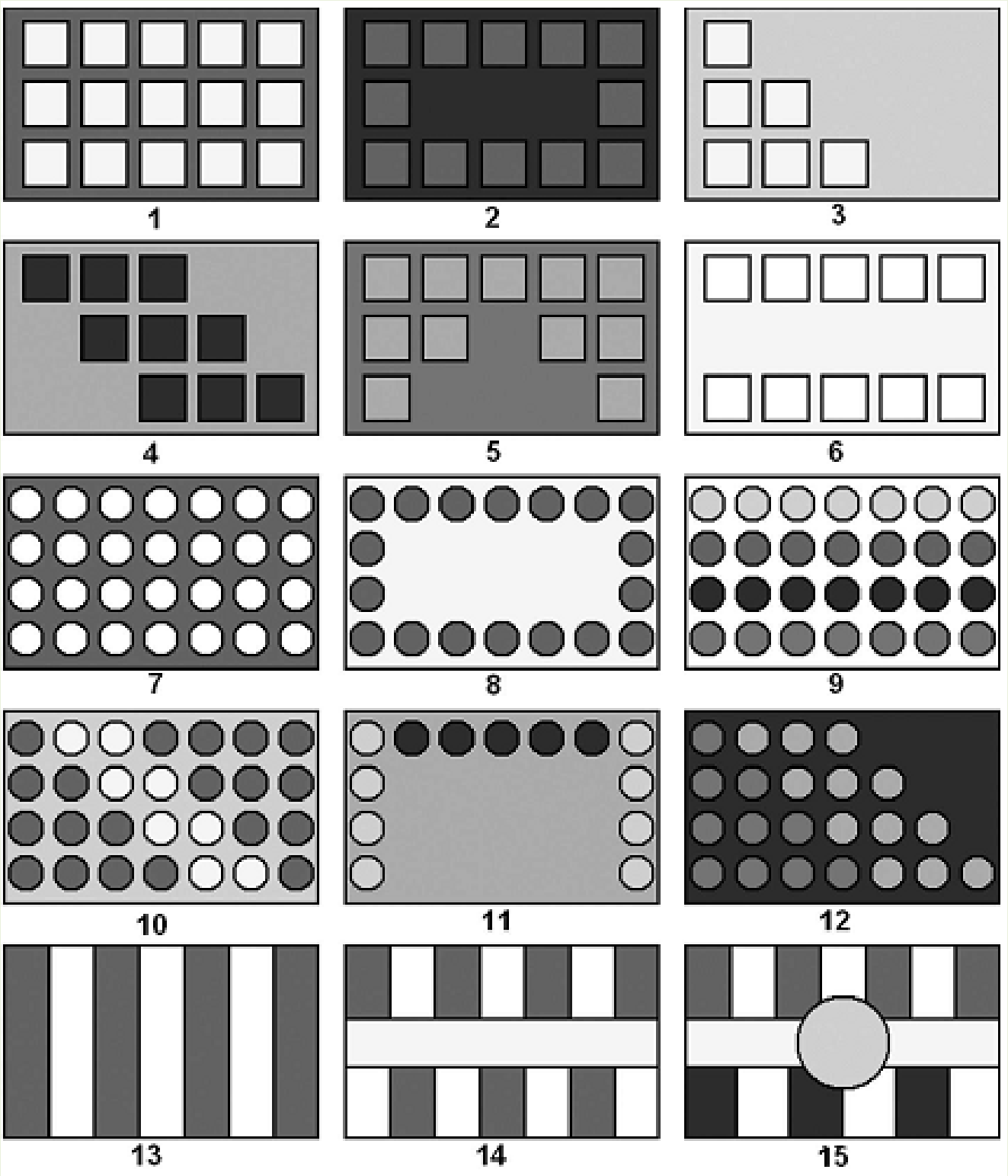

Рис. 5. Зразки прапорців для рисування за допомогою об'єкта Canv