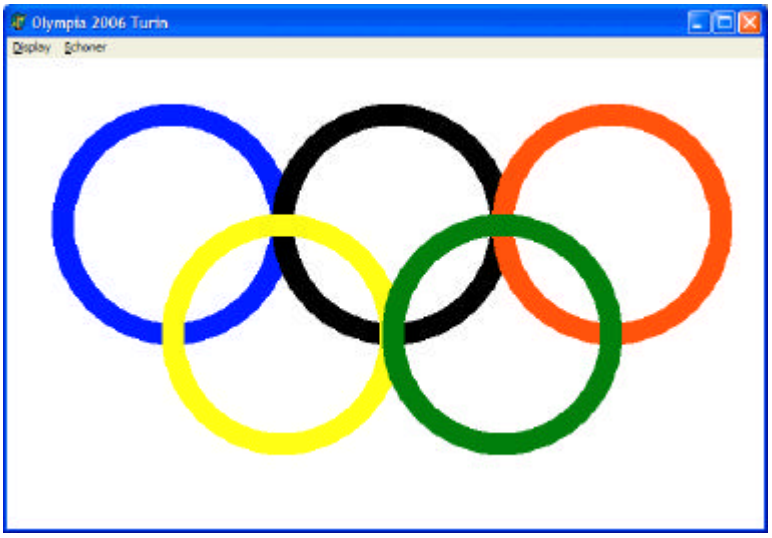
Projekt Olympische Ringe, Teil 1

Schreiben Sie ein Projekt, welches die Olympischen Ringe zeichnet.



- Die Menüauswahl **Ringe** soll die Olympischen Ringe zentriert auf das Fenster zeichnen (s.rechts):
- Die Menüauswahl **Vollbild** soll das Fenster voller Olympischer Ringe (ca. 100) zeichnen.
- Beim Menüpunkt **Bewegung** sollen die Ringe zufällig auf dem Bildschirm erscheinen und sich dann – immer größer werdend – über den Bildschirm bewegen.
- Beim Menüpunkt **Rotation** kommt zur Bewegung noch eine Rotation der Ringe hinzu.

Arbeitsauftrag:

Um die jeweils komplexer werdenden Aufgabenstellungen überschauen und damit besser bewältigen zu können (Prinzip „divide and conquer“ bzw. nach Caesar „divide et impera“) bietet es sich an, das Gesamtproblem in kleinere Teilprobleme (Prozeduren) zu zerlegen, die einfacher zu handhaben sind als das Problem als Ganzes.

Die Lösungen der Teilprobleme werden anschließend zur Lösung des Gesamtproblems verwendet. Dies kann erneut angewendet werden: Die Teilprobleme können ihrerseits in noch kleinere Teilprobleme zerlegt werden und so fort (Analyse und Synthese).

In der aktuellen Aufgabenstellung werden Sie zuerst eine Prozedur Kreis implementieren, welche Sie dann für den Aufbau der Ringe verwenden. Die Prozedur Ringe wird wiederum von den Prozeduren Vollbild, Bewegung und Rotation verwendet.

1. Kopieren Sie den **Ordner Olympia** vom Tauschverzeichnis in Ihr Arbeitsverzeichnis. Öffnen Sie die **Projektdatei olympia.dpr.**

2. **Implementieren** Sie eine eigene Prozedur **im Implementierungsteil der Unit**

```
procedure TFoOlympia.Kreis (Xm, Ym, R: Integer);
```

   welche einen **Kreisring** um den Mittelpunkt (**Xm,Ym**) mit dem Radius **R** zeichnet. Verwenden Sie die hierzu die
   - Canvas Methode **`Ellipse(Xo,Yo,Xu,Yu: Integer);`** und die
   - Canvas Eigenschaft **`Canvas.Brush.Style := bsClear`**;

3. **Deklarieren** Sie dieses neue Unterprogramm **in der Typdefinition der Schnittstelle**

```
private
  { Private declarations }
  procedure Kreis(Xm, Ym, R: integer);
```

4. **Testen Sie Ihre Prozedur**, indem Sie diese in der Ereignisbehandlungsroutine Ringe1Click (Auswahl des Menüpunktes Ringe) aufrufen:

```
procedure TFoOlympia.Ringe1Click(Sender: TObject);
begin
  Kreis(200, 200, 100);
end;
```

eventuell auch durch Verwendung innerhalb einer oder mehreren Zählschleifen:

```
for i := 1 to 10 do Kreis(100,  100, i*10);
for i := 1 to 20 do Kreis(i*20, 200, i*10);
```

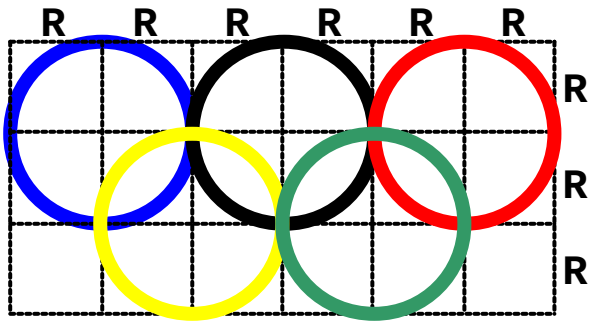
---

5. Implementieren **und** deklarieren (Private declarations) Sie eine weitere Prozedur

```
procedure TFoOlympia.Ringe (X, Y, R: Integer);
```

in der Sie die Prozedur **`Kreis (Xm, Ym, R: Integer);`** mehrmals aufrufen.

- Es sollen 5 Kreise gezeichnet werden,
  - mit jeweils anderen Mittelpunkten
  - und jeweils anderen Farben (Blau, Schwarz, Rot, Gelb, Grün):



- Die relative Lage der Kreismittelpunkte **`(Xm, Ym)`** muss bezüglich eines Referenzpunktes **`(X,Y)`** bestimmt werden. Wählen Sie hierzu den Drehpunkt, um den später die gesamte Figur rotieren soll.

- Testen Sie die neue Prozedur, indem Sie diese in der Ereignisbehandlungsroutine Ringe1Click aufrufen. Die Ringe sollen in der Mitte **`(X,Y)`** des Fensters erscheinen und der Radius R und die Stiftbreite sollen entsprechend angepasst werden:

```
procedure TFoOlympia.Ringe1Click(Sender: TObject);
var X,Y,R: integer;
begin
  X := ClientWidth  div 2;
  Y := ClientHeight div 2;
  R := ClientHeight div 4;
  Canvas.Pen.Width := R div 6;
  Ringe(X, Y, R);
end;
```

6. Implementieren **und** deklarieren (Private declarations) Sie nun eine weitere Prozedur:

```
procedure TFoOlympia.Vollbild1Click(Sender: TObject);
```

welche 100 Olympischen Ringen über das Fenster verteilt. Ort und Radius sollen dabei zufällig gewählt werden:

```
r := random(50);
x := random(ClientWidth);
y := random(ClientHeight);
```