TOPIC G: RECORD

Vrije Universiteit Brussel

```
MODULE D1;
<* NOOPTIMIZE + *>
  FROM IO IMPORT RdChar, WrChar, WrStr, WrLn,WrFixReal;
  VAR
   c1, c2, c3: RECORD
                 re, img: REAL;
               END;
BEGIN
  WrLn;
  c1.re := 3.5;
  c1.img := -4.76;
  c2.re := 4.0;
  c2.img := 14.6;
  (* sum *)
  c3.re := c1.re + c2.re;
  c3.img := c1.img + c2.img;
  (* print *)
  WrStr("The complex sum of ");WrFixReal(c1.re, 1, 0);WrChar('+'); WrFixReal(c1.img, 1, 0);WrChar('i');
  WrStr(" and ");WrFixReal(c2.re, 1, 0);WrChar('+'); WrFixReal(c2.img, 1, 0);WrChar('i');
  WrStr(" is: ");WrFixReal(c3.re, 2, 0);WrChar('+'); WrFixReal(c3.img, 2, 0);WrChar('i');WrLn;
END D1.
```


TOPIC G: RECORD

Vrije Universiteit Brussel

```
MODULE D1;
<* NOOPTIMIZE + *>
  FROM IO IMPORT RdChar, WrChar, WrStr, WrLn,WrFixReal;
  VAR
   c1, c2, c3: RECORD
                 re, img: REAL;
               END;
BEGIN
  WrLn;
  c1.re := 3.5;
  c1.img := -4.76;
  c2.re := 4.0;
  c2.img := 14.6;
  (* sum *)
  c3.re := c1.re + c2.re;
  c3.img := c1.img + c2.img;
  (* print *)
  WrStr("The complex sum of ");WrFixReal(c1.re, 1, 0);WrChar('+'); WrFixReal(c1.img, 1, 0);WrChar('i');
  WrStr(" and ");WrFixReal(c2.re, 1, 0);WrChar('+'); WrFixReal(c2.img, 1, 0);WrChar('i');
  WrStr(" is: ");WrFixReal(c3.re, 2, 0);WrChar('+'); WrFixReal(c3.img, 2, 0);WrChar('i');WrLn;
END D1.
```


TOPIC G: RECORD

Vrije Universiteit Brussel

```
MODULE D1;
<* NOOPTIMIZE + *>
  FROM IO IMPORT RdChar, WrChar, WrStr, WrLn,WrFixReal;
  VAR
   c1, c2, c3: RECORD
                 re, img: REAL;
               END;
BEGIN
  WrLn;
  c1.re := 3.5;
  c1.img := -4.76;
  c2.re := 4.0;
  c2.img := 14.6;
  (* sum *)
  c3.re := c1.re + c2.re;
  c3.img := c1.img + c2.img;
  (* print *)
  WrStr("The complex sum of ");WrFixReal(c1.re, 1, 0);WrChar('+'); WrFixReal(c1.img, 1, 0);WrChar('i');
  WrStr(" and ");WrFixReal(c2.re, 1, 0);WrChar('+'); WrFixReal(c2.img, 1, 0);WrChar('i');
  WrStr(" is: ");WrFixReal(c3.re, 2, 0);WrChar('+'); WrFixReal(c3.img, 2, 0);WrChar('i');WrLn;
END D1.
```

```
WITH c1 DO
  re := 3.5;
  img := -4.76;
END;
```


TOPIC G: RECORD

Vrije Universiteit Brussel

```
MODULE D1;
<* NOOPTIMIZE + *>
1   FROM IO IMPORT RdChar, WrChar, WrStr, WrLn,WrFixReal;
2   VAR
3   c1, c2, c3: RECORD
4                 re, img: REAL;
5               END;
6  BEGIN
7   WrLn;
8   c1.re := 3.5;
9   c1.img := -4.76;
10  c2.re := 4.0;
11  c2.img := 14.6;
12  (* sum *)
13   c3.re := c1.re + c2.re;
14  c3.img := c1.img + c2.img;
15   (* print *)
16   WrStr("The complex sum of ");WrFixReal(c1.re, 1, 0);WrChar('+'); WrFixReal(c1.img, 1, 0);WrChar('i');
17  WrStr(" and ");WrFixReal(c2.re, 1, 0);WrChar('+'); WrFixReal(c2.img, 1, 0);WrChar('i');
18   WrStr(" is: ");WrFixReal(c3.re, 2, 0);WrChar('+'); WrFixReal(c3.img, 2, 0);WrChar('i');WrLn;
END D1.
```

Things to remember!!!

- Geen spaties in de naam van bestanden!
- Kopieer nodige libraries in je working folder
  - Wimdows, StandardLib
- Maak geen nodeloze procedures!
- Vermijd het gebruik van globale variabelen in procedures.
- Gebruik zinvolle namen voor variabelen, constanten, procedures.
- Indenteer je code!

Oefening S1

1. Begin van de vb. Oefening D1, maar maak een type ComplexRc, en gebruik het om c1, c2 en c3 te definieren.
2. Maak een procedure die een (1!!!) complex getal (met formele parameter ComplexRc) print.
3. Maak een procedure die 2 complexe getallen vermenigvuldigt.

Oefening S1

1. Begin van de vb. Oefening D1, maar maak een **type** ComplexRc, en gebruik het om c1, c2 en c3 te definieren.
   → **TYPE** ComplexRC = ..
2. Maak een procedure die een complex getal (met formele parameter ComplexRc) print.
3. Maak een procedure die 2 complexe getallen vermenigvuldigt.

Oefening S1

1. Begin van de vb. Oefening D1, maar maak een type ComplexRc, en gebruik het om c1, c2 en c3 te definieren.
2. Maak een procedure die een complex getal (met formele parameter ComplexRc) print.
   → **PROCEDURE** *name(formele parameters);*
   BEGIN

   END …;
3. Maak een procedure die 2 complexe getallen vermenigvuldigt.

Oefening S1

1. Begin van de vb. Oefening D1, maar maak een type ComplexRc, en gebruik het om c1, c2 en c3 te definieren.
2. Maak een procedure die een complex getal (gebruik ComplexRc).
   → **PROCEDURE** *name(formele parameters)<returntype>;*
   …
   BEGIN

   END …;
3. Maak een procedure die

- Hoeveel parameters?
- Welk type parameters?
- Return waarde? Zo ja, welk type? (Let op! De symbolen "<" en ">" worden hier enkel gebruikt om aan te geven dat je ***eventueel*** een returntype moet geven, dit hoort dus niet thuis in je Modula code)

Oefening S1

1. Begin van de vb. Oefening D1, maar maak een type ComplexRc, en gebruik het om c1, c2 en c3 te definieren.
2. Maak een procedure die een complex getal (met formele parameter ComplexRc) print.
3. Maak een procedure die 2 complexe getallen vermenigvuldigt.
   → Zelfde principe als in 2
   → Formule: (a+bi)*(c+di) = (ac-bd)+(bc+ad)i
4. Gebruik in de body je nieuwe procedures!

OEFENING S6: DEEL 1

1. Definieer een RECORD
2. Definieer een ARRAY van het type record gemaakt in 1
3. Kopieer initialisatie van "a"

OEFENING S6: DEEL 1

1. Definieer een RECORD
   ➢ **TYPE** CirclRc = **RECORD**
   …
   END;
2. Definieer een ARRAY van het type record gemaakt in 1
3. Kopieer initialisatie van "a"

OEFENING S6: DEEL 1

1. Definieer een RECORD
   - **TYPE** CircleRc = **RECORD**
     …
     END;
2. Definieer een ARRAY van het type record gemaakt in 1
   - **VAR** a: **ARRAY** [1..A_SIZE] **OF** CircleRc;
3. Kopieer initialisatie van "a"

OEFENING S6: DEEL 2

Teken cirkels en snelheidsvectoren, laat ze bewegen (in deze stap is het voldoende om ze gewoon uit het scherm te laten gaan). Zie oefening S6 van Topic D, bewegende cirkels.

1. Teken cirkels door het **Disc** command
   - Disc(x_center,y_center,straal,kleur)
2. Teken arrows door het **Arrow** command
   - Arrow(x_0,y_0,x_1,y_1,size,kleur);
     - Zorg dat je ze kan zien!

In dit deel hoef je geen rekening te houden met het scherm of het programma einde.

OEFENING S6: DEEL 2

Teken cirkels en snelheidsvectoren, laat ze bewegen

1. Teken cirkels door het **Disc** command
   - Disc(x_center,y_center,straal,kleur)
2. Teken arrows door het **Arrow** command
   - Arrow(x_0,y_0,x_1,y_1,size,kleur);
     - Zorg dat je ze kan zien!
3. Laat ze bewegen: TIP: kleur → Delay → black →
   - Hoe verandert x en y?

In dit deel hoef je geen rekening te houden met het scherm of het programma einde.

OEFENING S6: DEEL 2

Teken cirkels en snelheidsvectoren, laat ze bewegen

1. Teken cirkels door het **Disc** command
   - Disc(x_center,y_center,straal,kleur)
2. Teken arrows door het **Arrow** command
   - Arrow(x_0,y_0,x_1,y_1,size,kleur);
3. Laat ze bewegen: TIP: kleur → Delay → black →
   - Hoe verandert x en y?
4. Verschillende kleuren van ballen → Maak een array van kleuren
   - **RgbColor(r_value,g_value,b_value)** met 0<r_value,g_value,b_value<255, geeft een cardinal waarde terug die een kleur voorstelt.
   - **RandomCard(min,max)** geeft een random cardinal tussen min en max

OEFENING S6: DEEL 3

Laat de ballen botsen tegen de rand → slechts 1 richtingsvector van teken veranderen.

Opties:
- → Het middelpunt van de bal botst tegen de rand
- → De rand van de bal botst
- → Extra kader maken, de ballen botsen hier tegen.

OEFENING S6: DEEL 3

Laat de ballen botsen tegen de rand → slechts 1 richtingsvector van teken veranderen.

IF x>.. **THEN**
IF x<.. **THEN**
IF y>.. **THEN**
IF y<.. **THEN**

OEFENING S6: DEEL 3

Laat de ballen botsen tegen de rand → slechts 1 richtingsvector van teken veranderen.

→ Stop het programma door op een toets te drukken, vb "s" van stop
- → KeyPressed();
- → RdKey();

OEFENING S6: DEEL 4

- Laat de circels botsen als ze elkaar raken, net als biljartballen.
- Gebruik de volgende functies uit StandardLib
  - ➢ Distance(x1, y1, x2, y2: INTEGER): CARDINAL;
  - ➢AngleWithXAx(x0, y0, x1, y1: INTEGER):REAL;
  - ➢ RadiansToDegrees(angle:REAL): INTEGER;
  - ➢RotatePointDegrees(VAR x, y:INTEGER; x0, y0: INTEGER; angle: INTEGER);

OEFENING S6: DEEL 4

1. Kijk voor elke bal, of hij tegen een andere bal botst

   Liggen bal_i en bal_k niet in elkaars veld??
   → Distance(x1, y1, x2, y2: INTEGER): CARDINAL;

→ Let op!!! Zorg dat je de huidige posities niet verliest!

OEFENING S6: DEEL 4

1. Kijk voor elke bal, of hij tegen een andere bal botst
   Liggen bal_i en bal_k niet in elkaars veld??
   → Distance(x1, y1, x2, y2: INTEGER): CARDINAL
1. Bereken!
   - Hoek b, hoek a:
     → AngleWithXAx(x0, y0, x1, y1: INTEGER):REAL

OEFENING S6: DEEL 4

1. Kijk voor elke bal, of hij tegen een andere bal botst
   Liggen bal_i en bal_k niet in elkaars veld??
   → Distance(x1, y1, x2, y2: INTEGER): CARDINAL
2. Bereken!
   - Hoek b, hoek a:
     → AngleWithXAx(x0, y0, x1, y1: INTEGER):REAL
   - c = b-a → zet om naar graden ipv radialen
     → RadiansToDegrees(angle:REAL): INTEGER;
   - v = 180-2*c
   - Nieuwe richtingsvector vx en vy
     → RotatePointDegrees(**VAR** x, y:INTEGER; x0, y0: INTEGER; angle: INTEGER);

Let op!! Deze functies zorgen voor grote afrondingsfouten, maak vx en vy tijdelijk groter (*100)