# Errata



Då man skriver en bok verkar det som att man inte alltid får allt rätt på en gång. Även om intentionerna var goda har jag missat ett antal saker. På några av dessa fel märks det tydligt att jag inte sovit tillräckligt inför skrivandet.

Utan att försöka skylla ifrån mig följer därför här ett antal rättelser till boken.

# Sida 30, största värdet i det binära talet

Matematiken stämmer inte riktigt. Antalet värden är 256, men det största talet som kan representeras är 255.

******* Det står:**

Om man för enkelhets skull antar att ett heltal består av 8 bitar och att dessa kan representera talen 0, 1, 2, ..., 256.

******* Det bör stå:**

Om man för enkelhets skull antar att ett heltal består av 8 bitar och att dessa kan representera talen 0, 1, 2, ..., 255.

# Sida 33, i listan av attribut

Ordet ”föregående” skall ersättas med ”nästföljande” när det gäller attributet ”Succ”.

******* Det står:**

*Type*’Succ(*T*) - det föregående värdet i datatypen (*Friday* i exemplet ger *Saturday*)

******* Det bör stå:**

*Type*’Succ(*T*) - det efterföljande värdet i datatypen (*Friday* i exemplet ger *Saturday*)

# Sida 75, sista stycket

En lite förhastad slutsats på sidan 75 vad det gäller innehållet i tabellen.

******* Det står:**

Den enda gång operatorerna får samma resultat är för positiva "T" och "N".

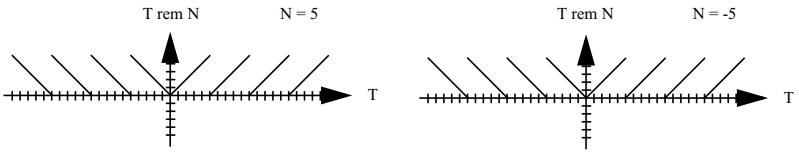
******* Det bör stå:**

De enda gånger operatorerna får samma resultat är då både "T" och "N" är positiva eller både "T" och "N" är negativa.

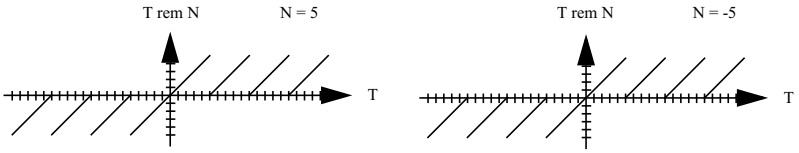
# Sida 76, graferna för ”rem” och stycket efter graferna

Den negativa delen av ”T”-axeln har råkat få positiva “T mod N”-värden i graferna för ”rem”. Dessutom konstig formulering i stycket efter graferna.

******* Grafernas utseende nu:**




******* De bör se ut enligt följande:**




******* Det står:**

Observera att positivt och negativt N-värde ger olika resultat ”mod”, medan ”rem” får samma resultat.

******* Det bör stå:**

Observera att ett positivt och negativt N-värde ger olika resultat för ”mod”, medan ”rem” ger samma resultat.

# Sida 91, andra stycket, tredje raden

Det saknas ett ”till” och ”anropande” ska bytas mot ”anropade”.

******* Det står:**

tillbaka programmet som anropande ”Get_Line”

******* Det bör stå:**

tillbaka till programmet som anropade ”Get_Line”

# Sida 119, första stycket, sista meningen

Ett felstavat “Length”.

******* Det står:**

Detta medför att ”Lenght” ger värdet 9 medan ”Last” ger värdet 10.

******* Det bör stå:**

Detta medför att ”Length” ger värdet 9 medan ”Last” ger värdet 10.

# Sida 131, i faktarutan och i exemplet

Det har smugit sig in ett semikolon (;) istället för ett komma (,) i parameterlistorna i både faktarutan och i exemplet på samma sida. Man skall inte använda ”klipp och klistra”. :-(

******* I faktarutan står det:**

```
procedure My_Procedure_Name is
  new Ada.Unchecked_Deallocation(Object => Object_Type;
                                 Name   => Access_Type);
```

******* Det borde stå:**

```
procedure My_Procedure_Name is
  new Ada.Unchecked_Deallocation(Object => Object_Type,
                                 Name   => Access_Type);
```

******* I exemplet står det:**

```
procedure Free is
  new Ada.Unchecked_Deallocation(Object => Integer;
                                 Name   => Integer_Ptr);
```

******* Det bör stå:**

```
procedure Free is
  new Ada.Unchecked_Deallocation(Object => Integer,
                                 Name   => Integer_Ptr);
```

# Sida 136, i exemplet

Man kan tänka sig att det här med ”klipp och klistra” inte alltid är bra som sagt.

******* I exemplet står det:**

```
procedure Free is
  new Ada.Unchecked_Deallocation(Object => Person_Type;
                                 Name   => Person_Ptr);
```

******* Det bör stå:**

```
procedure Free is
  new Ada.Unchecked_Deallocation(Object => Person_Type,
                                 Name   => Person_Ptr);
```

# Sida 197, första faktarutan

Instansiering av en function kan inte göras som en procedur. Ersätt ”procedure” med ”function”.

******* I faktarutan står det:**

```
procedure New_Function_Name is
  new Generic_Function_Name[(Actual_Parameter_List)];
```

******* Det borde stå:**

```
function New_Function_Name is
  new Generic_Function_Name[(Actual_Parameter_List)];
```

# Sida 208, sista stycket, och sida 209

******* Det står:**

Det finns två varianter av definitioner av ingångar i skyddade typer. Den ena varianten har ett villkor eller en ”barriär” att ta sig förbi. Detta ger att man kan se till att ingången är stängd i vissa fall. I den andra varianten har man inte detta villkor vilket gör att ingången alltid är öppen.

När man definierar en ingång, utan villkor, i en skyddad typ använder man följande kontruktion.

```
entry Entry_Name[(parameter_list)] is
  -- Declarations
begin
  -- Statements
end Entry_Name;
```

Lägger man till villkoret ändras endast första raden till följande:

```
entry Entry_Name[(parameter_list)]
  when condition is
```

******* Det bör stå:**

Definitionen av ingångar i skyddade typer innehåller alltid en så kallad ”barriär” att ta sig förbi. Detta ger att man kan se till att ingången är stängd i vissa fall.

När man definierar en ingång i en skyddad typ använder man följande konstruktion.

```
entry Entry_Name[(parameter_list)]
  when condition is
  -- Declarations
begin
  -- Statements
end Entry_Name;
```

# Sida 295, i exemplet

Variabeln ”Ch” som används i programmet är ej definierad.

******* I exemplet står det:**

```
procedure Find_A is

  File  : File_Type;
  Count : Integer;

begin
```

******* Det bör stå:**

```
procedure Find_A is

  File  : File_Type;
  Count : Integer;
  Ch    : Character;

begin
```