# Föreläsning 2

## Grundläggande imperativ programmering i Modula-2

- Övergripande programstruktur
  - Importlista
  - Deklarationer
  - Kommandon
- Grundkommandon
  - In- och utmatning (proceduranrop)
  - Variabler och tilldelning
- Sammansättning av kommandon i
  - Sekvens
  - Val
  - Upprepning
- Procedurer och funktioner
  - Funktionsprocedurer
  - Rena procedurer
- Biblioteksmoduler
  - InOut



# Exempel 1

## Hello-programmet

```
MODULE hello;

FROM InOut IMPORT WriteString, WriteLn;

BEGIN
  WriteString("Hello, world!");
  WriteLn
END hello.
```

### Exempel på

- hur moduler ska se ut
- import av procedurer
- procedurandrop med 0 eller 1 argument.
- utskrift på terminalen
- sekvens av kommandon
- diverse semikolon
- nyckelord: skrivs med stora bokstäver



# Exempel 2

## Ett artigare program

```
MODULE hello2;

FROM InOut IMPORT ReadString, WriteString,
  WriteLn;
FROM Strings IMPORT String;

VAR namn : String;

BEGIN
  WriteString("Vad heter du? ");
  ReadString(namn);
  WriteString("Trevligt att träffas, ");
  WriteString(namn);
  WriteString("!");
  WriteLn
END hello2.
```

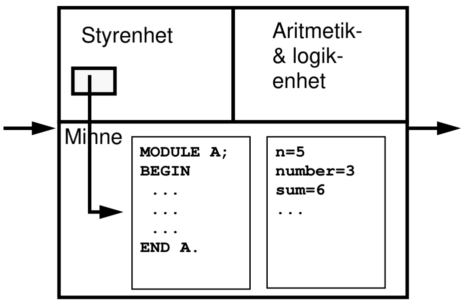
### Exempel på

- deklaration av variabler
- inläsning från terminalen
- användning av variabel för lagring av indata



# Hur datorn kör program

## von Neumann-datorn




- Programmet ändras inte under körningen.
- Kommandona i programmet utförs i tur och ordning.
- Instruktionspekaren pekar ut nästa instruktion som ska utföras.

  (Anm: satser=instruktioner=kommandon)
- Variabler är platser i minnet där ett värde av variabelns typ kan lagras.
- Variablernas värden varierar vanligen!

## Exempel 3

### Skottår

```
MODULE skott;

FROM InOut IMPORT ReadInt, WriteString, WriteLn;

VAR year : INTEGER;

BEGIN
  WriteString("Programmet testar om ett år är ett skottår.");
  WriteLn;
  WriteString("Vilket år ska undersökas? ");
  ReadInt(year);
  IF year MOD 4 = 0 THEN
    WriteString("Året är ett skottår.")
  ELSE
    WriteString("Året är inte ett skottår.")
  END;
  WriteLn
END skott.
```

Exempel på

- val med IF-sats
- enkla uttryck



## Exempel 4

```
MODULE skott2;

FROM InOut IMPORT ReadInt, WriteString, WriteLn;

VAR year : INTEGER;

BEGIN
  WriteString("Programmet testar om ett år är ett skottår.");
  WriteLn;
  WriteString("Vilket år ska undersökas? ");
  ReadInt(year);
  WriteString("Året är ");
  IF year MOD 4 # 0 THEN
    WriteString("inte ")
  END;
  WriteString("ett skottår.");
  WriteLn
END skott2.
```

Exempel på

- IF-sats utan ELSE-gren
- # (ej likamed)



## Exempel 5

### Åldersgrupper

```
MODULE age;

FROM InOut IMPORT ReadCard, WriteString, WriteLn;

VAR age : CARDINAL;

BEGIN
  WriteString("Programmet undersöker vilken ");
  WriteString("åldersgrupp en person tillhör.");
  WriteLn;
  WriteString("Hur gammal är personen? ");
  ReadCard(age);
  WriteString("Åldersgrupp: ");
  IF age < 13 THEN
    WriteString("barn.")
  ELSIF age < 20 THEN
    WriteString("tonåring.")
  ELSIF age < 65 THEN
    WriteString("vuxen.")
  ELSE
    WriteString("pensionär.")
  END;
  WriteLn
END age.
```

Exempel på

- IF-sats med många grenar



## Exempel 6

### Antal dagar i en månad

```
MODULE month;

FROM InOut IMPORT ReadCard, WriteString, WriteLn;

VAR month : CARDINAL;

BEGIN
  WriteString("Programmet beräknar antal dagar i en månad.");
  WriteLn;
  WriteString("Mata in månadens nummer: ");
  ReadCard(month);
  WriteString("Månaden har ");
  CASE month OF
    30, 4, 6, 9: WriteString("30");
  | 2: WriteString("28");
  ELSE WriteString("31")
  END;
  WriteString(" dagar."); WriteLn
END month.
```

Exempel på

- CASE-sats

# Exempel 7

## En evighetsmaskin?

```
MODULE evigt;

FROM InOut IMPORT WriteString, WriteLn;

BEGIN
  LOOP
    WriteString("En dator blir aldrig trött...")
  END
END evigt.
```

### Exempel på

- LOOP - upprepning, iteration



# Exempel 8

## Meningsfull upprepning

```
MODULE evigt2;

FROM InOut IMPORT ReadCard, WriteString, WriteLn;

VAR month : CARDINAL;

BEGIN
  WriteString("Programmet beräknar antal dagar i en månad.");
  WriteLn;
  LOOP
    WriteString("Mata in månadens nummer: ");
    ReadCard(month);
    WriteString("Månaden har ");
    CASE month OF
      30, 4, 6, 9: WriteString("30");
    | 2: WriteString("28");
    ELSE WriteString("31")
    END;
    WriteString(" dagar."); WriteLn;
    WriteLn
  END
END evigt2.
```

### Exempel på

- LOOP - evig men inte meningslös



# Exempel 9

## Upprepning ett bestämt antal gånger

```
MODULE for;

FROM InOut IMPORT WriteString, WriteLn;

VAR n : CARDINAL;

BEGIN
  FOR n := 1 TO 10 DO
    WriteString("10 kopior av denna sträng");
    WriteLn
  END
END for.
```

### Exempel på

- FOR - ger ett bestämt antal upprepningar
- (Inte bara) ett sätt att slippa skriva samma sak många gånger



# Exempel 9 (forts)

## Om inte FOR fanns

```
MODULE utanfor;

FROM InOut IMPORT WriteString, WriteLn;

BEGIN
  WriteString("10 kopior av denna sträng"); WriteLn;
  WriteString("10 kopior av denna sträng"); WriteLn;
  WriteString("10 kopior av denna sträng"); WriteLn;
  WriteString("10 kopior av denna sträng"); WriteLn;
  WriteString("10 kopior av denna sträng"); WriteLn;
  WriteString("10 kopior av denna sträng"); WriteLn;
  WriteString("10 kopior av denna sträng"); WriteLn;
  WriteString("10 kopior av denna sträng"); WriteLn;
  WriteString("10 kopior av denna sträng"); WriteLn;
  WriteString("10 kopior av denna sträng"); WriteLn
END utanfor.
```

### Exempel på

- FOR - kan man använda istället för att göra så här.
- Man kan se de som att detta är vad FOR-satsen i förra exemplet betyder.

## Exempel 10

### Upprepning med variation

```
MODULE for2;

FROM InOut IMPORT WriteString, WriteCard,
  WriteLn;

VAR n : CARDINAL;

BEGIN
  WriteString("Kvadrattabell"); WriteLn;
  WriteLn;
  WriteString(" n   n*n"); WriteLn;
  WriteString("==   ==="); WriteLn;
  FOR n := 5 TO 15 DO
    WriteCard(n,2);
    WriteCard(n*n,6);
    WriteLn
  END
END for2.
```

### Exempel på

- FOR - användning av loop-variabeln
- WriteCard
- procedurer med mer än ett argument: "," mellan argumenten.



## Exempel 10 (forts)

### Om inte FOR fanns

```
MODULE utanfor2;

FROM InOut IMPORT WriteString, WriteCard, WriteLn;

BEGIN
  WriteString("Kvadrattabell"); WriteLn;
  WriteLn;
  WriteString(" n   n*n"); WriteLn;
  WriteString("==   ==="); WriteLn;
  WriteCard(5,2); WriteCard(5*5,6); WriteLn;
  WriteCard(6,2); WriteCard(6*6,6); WriteLn;
  WriteCard(7,2); WriteCard(7*7,6); WriteLn;
  WriteCard(8,2); WriteCard(8*8,6); WriteLn;
  WriteCard(9,2); WriteCard(9*9,6); WriteLn;
  WriteCard(10,2); WriteCard(10*10,6); WriteLn;
  WriteCard(11,2); WriteCard(11*11,6); WriteLn;
  WriteCard(12,2); WriteCard(12*12,6); WriteLn;
  WriteCard(13,2); WriteCard(13*13,6); WriteLn;
  WriteCard(14,2); WriteCard(14*14,6); WriteLn;
  WriteCard(15,2); WriteCard(15*15,6); WriteLn;
END utanfor2.
```

### Exempel på

- korkad programmerare?!



## Exempel 11

### Ränta på ränta

```
MODULE bank;

FROM InOut IMPORT WriteString, WriteCard, WriteReal, WriteLn;

CONST startkapital = 1000.0;
      ranta = 7.0; (* ränta *)

VAR n : CARDINAL;
    saldo : REAL;

BEGIN
  WriteString("Ränta på ränta"); WriteLn;
  WriteLn;
  WriteString("år   saldo   "); WriteLn;
  WriteString("==   ========"); WriteLn;
  saldo := startkapital;
  FOR n := 1 TO 10 DO
    saldo := saldo * (1.0+ranta/100.0);
    WriteCard(n,2);
    WriteReal(saldo,11,2);
    WriteLn
  END
END bank.
```

### Exempel på

- CONST - namngivna konstanter
- VAR - flera variabler, av olika typ
- användning av variabler i loopar
- WriteReal



## Exempel 12

### Variabelt antal upprepningar

```
MODULE bank2;

FROM InOut IMPORT WriteString, ReadCard, WriteReal, WriteLn;

CONST startkapital = 1000.0;
      ranta = 7.0; (* ränta *)

VAR n, antal : CARDINAL;
    saldo : REAL;

BEGIN
  WriteString("Ränta på ränta"); WriteLn;
  WriteLn;
  WriteString("Hur många år ska kapitalet stå inne? ");
  ReadCard(antal);
  saldo := startkapital;
  FOR n := 1 TO antal DO
    saldo := saldo * (1.0+ranta/100.0);
  END;
  WriteString("Efter så många år är saldot ");
  WriteReal(saldo,1,2);
  WriteLn
END bank2.
```

### Exempel på

- FOR - antalet varv bestäms av en variabel
- VAR - flera variabler, av samma typ

# Exempel 13

## Okänt antal upprepningar

```
MODULE bank3;
FROM InOut IMPORT WriteString, WriteCard, WriteLn;

CONST startkapital =  1000.0;
      malkapital   = 10000.0; (* målkapital *)
      ranta = 7.0;  (* ränta *)

VAR n : CARDINAL;
    saldo : REAL;

BEGIN
  WriteString("Ränta på ränta"); WriteLn;
  WriteLn;
  saldo := startkapital;
  n := 0;
  WHILE saldo < malkapital DO
    (* n     = antal år kapitalet har stått inne,
     * saldo = saldo efter n år. *)
    n := n + 1;
    saldo := saldo * (1.0+ranta/100.0);
  END;
  WriteString("Det önskade målkapitalet uppnås efter ");
  WriteCard(n,1);
  WriteString(" år."); WriteLn
END bank3.
```

### Exempel på

- WHILE - loop där antalet varv beror på beräkningarna i loopen.
- variabler i loopar



# Sammanfattning

- I exemplen har vi har sett
  - Hur program ser ut

    Importlista, deklarationer, kommandon
  - Grundkommandon

    Proceduranrop för in- och utmatning

    Variabler och tilldelning
  - Sammansättning av kommandon i

    Sekvens

    Val

    Upprepning
- Vi har sett alla grundläggande programkonstruktioner för imperativ programmering
- Vi ska nu se på programkonstruktioner som har att göra med hur man *organiserar* program så att de blir överskådliga, lätta att förstå/förändra
  - Procedurer och funktioner (idag)
  - Moduler (senare)
  - Datatyper (senare)
  - Funktioner och procedurer



# Funktioner och procedurer

- De flesta språk har ett sätt att ge namn och parametrar åt programdelar

### I funktionella språk

- Program byggs upp av uttryck och funktioner
- Funktioner kan ses som namngivna uttryck med parametrar

### I imperativa språk

- Program byggs upp av kommandon och *procedurer*
- Procedurer kan ses som namngivna kommandon med parametrar
- Det finns uttryck också. Därför finns det även funktioner.

### I både funktionella och imperativa språk

- Datatyper - för organisation av data
- Moduler - för organisation of stora program



# Exempel 14

## Omorganisation av exempel 4

```
MODULE skott3;

FROM InOut IMPORT ReadInt, WriteString, WriteLn;

PROCEDURE isLeapYear(y: INTEGER): BOOLEAN;
BEGIN
  RETURN y MOD 4 = 0
END isLeapYear;

VAR year : INTEGER;

BEGIN
  WriteString("Programmet testar om ett år är ett skottår.");
  WriteLn;
  WriteString("Vilket år ska undersökas? ");
  ReadInt(year);
  WriteString("Året är ");
  IF NOT isLeapYear(year) THEN
    WriteString("inte ")
  END;
  WriteString("ett skottår.");
  WriteLn
END skott3.
```

### Exempel på

- Deklaration av funktionsprocedurer
- RETURN för att lämna resultatet från en funktion
- Jmf funktion i SML: fun isLeapYear y = y mod 4 = 0

# Exempel 15

## Mer omorganisation

```
MODULE skott4;
FROM InOut IMPORT ReadInt, WriteInt, WriteString, WriteLn;

PROCEDURE isLeapYear(y: INTEGER): BOOLEAN;
BEGIN
  RETURN y MOD 4 = 0
END isLeapYear;

PROCEDURE WriteBool(b: BOOLEAN);
BEGIN
  IF b THEN WriteString("sant")
  ELSE WriteString("falskt")
  END
END WriteBool;

VAR year : INTEGER;

BEGIN
  WriteString("Programmet testar om ett år är ett skottår.");
  WriteLn;
  WriteString("Vilket år ska undersökas? ");
  ReadInt(year);
  WriteString("Det är ");
  WriteBool(isLeapYear(year));
  WriteString(" att ");
  WriteInt(year,1);
  WriteString(" är ett skottår.");
  WriteLn
END skott4.
```

### Exempel på

- Deklaration av rena procedurer



# Exempel 16

## Omorganisation av exempel 6

```
MODULE month2;

FROM InOut IMPORT ReadCard, WriteCard, WriteString, WriteLn;

PROCEDURE monthLength(m: CARDINAL): CARDINAL;
BEGIN
  CASE m OF
    30, 4, 6, 9: RETURN 30
  | 2: RETURN 28
  ELSE RETURN 31
  END
END monthLength;

VAR month : CARDINAL;

BEGIN
  WriteString("Programmet beräknar antal dagar i en månad.");
  WriteLn;
  WriteString("Mata in månadens nummer: ");
  ReadCard(month);
  WriteString("Månaden har ");
  WriteCard(monthLength(month),1);
  WriteString(" dagar."); WriteLn
END month2.
```

### Exempel på

- CASE-sats i funktion

  RETURN i alla grenarna



# Exempel 17

## Omorganisation av exempel 13

```
MODULE bank4;
FROM InOut IMPORT WriteString, WriteCard, WriteLn;
CONST startkapital =  1000.0;
      malkapital   = 10000.0;  (* målkapital *)
      ranta = 7.0;  (* ränta *)
PROCEDURE tid(start, mal: REAL) : INTEGER;
  VAR n : CARDINAL;
      saldo : REAL;
BEGIN
  saldo := start; n := 0;
  WHILE saldo < mal DO
    (* n     = antal år kapitalet har stått inne,
     * saldo = saldo efter n år. *)
    n := n + 1;
    saldo := saldo * (1.0+ranta/100.0);
  END;
  RETURN n
END tid;

BEGIN
  WriteString("Ränta på ränta"); WriteLn; WriteLn;
  WriteString("Det önskade målkapitalet uppnås efter ");
  WriteCard(tid(startkapital,malkapital),1);
  WriteString(" år."); WriteLn
END bank4.
```

### Exempel på

- Funktionsprocedur

  ser ut som en funktion utifrån

  är imperativt programmerad inuti



# Biblioteksmoduler

## InOut (se även lab 4)

```
DEFINITION MODULE InOut;

   PROCEDURE Read (VAR x : CHAR);
   (* Read the next character from std input into 'x' *)

   PROCEDURE ReadString (VAR x : ARRAY OF CHAR);
   (* Read the next string from std input into 'x'. *)
   (* Leading blanks are ignored.                   *)
   (* Input is terminated by any character <= ' '   *)

   PROCEDURE ReadCard (VAR x : CARDINAL);
   (* Read the next string from std input and *)
   (* convert it to cardinal 'x'.             *)
   (* Syntax : digit {digit}                  *)

   PROCEDURE ReadInt (VAR x : INTEGER);
   (* ... *)

   PROCEDURE ReadReal (VAR x : REAL);

   PROCEDURE Write (x : CHAR);
   PROCEDURE WriteString (x : ARRAY OF CHAR);

   PROCEDURE WriteCard (x : CARDINAL; n : CARDINAL);
   PROCEDURE WriteInt (x : INTEGER; n : CARDINAL);
   PROCEDURE WriteReal (x : REAL; n : CARDINAL; k : INTEGER);
   PROCEDURE WriteLn;

   PROCEDURE Done () : BOOLEAN;
   PROCEDURE EOF () : BOOLEAN;

END InOut.
```

# Biblioteksmoduler

## MathLib (se även lab 4)

```
DEFINITION MODULE MathLib;

   PROCEDURE sqrt (x : REAL) : REAL;
   PROCEDURE sqrtL (x : LONGREAL) : LONGREAL;
   PROCEDURE exp (x : REAL) : REAL;
   PROCEDURE expL (x : LONGREAL) : LONGREAL;
   PROCEDURE ln (x : REAL) : REAL;
   PROCEDURE lnL (x : LONGREAL) : LONGREAL;
   PROCEDURE sin (x : REAL) : REAL;
   PROCEDURE sinL (x : LONGREAL) : LONGREAL;
   PROCEDURE cos (x : REAL) : REAL;
   PROCEDURE cosL (x : LONGREAL) : LONGREAL;
   PROCEDURE arctan (x : REAL) : REAL;
   PROCEDURE arctanL (x : LONGREAL) : LONGREAL;
   PROCEDURE real (x : INTEGER) : REAL;
   PROCEDURE realL (x : INTEGER) : LONGREAL;
   PROCEDURE entier (x : REAL) : INTEGER;
   PROCEDURE entierL (x : LONGREAL) : INTEGER;

END MathLib.
```