Föreläsning 3

- Ett programmeringsexempel som illustrerar
  - Problemlösning & programmeringsmetodik
  - In- & utmatning
  - Variabler
  - IF & CASE
  - Typer

(Lite repetition och lite nytt)

- Typdeklarationer och egna typer
  - Delintervall
  - Poster (record-typer)
  - Uppräkningstyper
  - Fördefinierade typer

Ett programmeringsexempel

Uppgift

- Skriv ett program som
  - frågar användaren efter ett datum,
  - beräknar nästa dags datum,
  - skriver ut det uträknade datumet.

Körningsexempel

```
muppet52% tomorrow
Mata in ett datum (år mån dag): 1994 2 28
Nästa dags datum är: 1994  3  1
muppet52% tomorrow
Mata in ett datum (år mån dag): 1992 2 28
Nästa dags datum är: 1992  2 29
muppet52% tomorrow
Mata in ett datum (år mån dag): 1994 12 31
Nästa dags datum är: 1995  1  1
```


Problemlösning

Steg 1

- Förstå problemet.

Steg 2

- Dela upp problemet i delproblem.

  Illustrera med ett strukturdiagram.

  ...

Steg 3

- Lös delproblemen.

Viktigt!

Programmeringsmetodik

Konsten att organisera programmet så att det blir lätt att skriva, läsa, förstå, förändra, ...

Programstruktur

- Låt programmets struktur avspegla problemets struktur!
  - Lösningen på hela problemet = ett program.
  - Lösningen på ett delproblem = en funktion eller en procedur.

Datatyper

- Definiera datatyper som direkt motsvarar de storheter (den slags information) som förekommer i problemet.
  - I tomorrow-programmet: en typ för datum!

(I större program: lägg en datatyp med tillhörande funktioner och procedurer i en egen modul.)

Typer

Nya namn på gamla typer

Exempel

```
MODULE typexempel;

FROM ... IMPORT ...;

TYPE Year  = INTEGER;
     Month = CARDINAL;
     Day   = CARDINAL;

VAR day:Day; month:Month;

BEGIN
    ...
END typexempel.
```

Värt att notera

- Typdeklarationer börjar med nyckelordet **TYPE**.
- Program blir mer självdokumenterande om man namnger typer
- Namngivna typer fungerar som typsynonymer i SML

```
...
month:=day; (* kompilatorn klagar ej *)
```


Typer

Delintervallstyper

Exempel

```
TYPE Month = [1..12];
     Day   = [1..31];

VAR day:Day; month:Month;
```

Värt att notera

- Program blir mer självdokumenterande om man använder mer informativa typer.
- Man får hjälp med felkontroll: kompilatorn kontrollerar att en variabel inte får värden utanför gränserna.
- Man kan blanda olika delintervallstyper i samma uttryck:

```
WriteCard(day+month,1);
```

Inte fel, tycker kompilatorn.

```
day:=month;
```

Inte heller fel.

```
month:=day;
```

Också tillåtet, men programmet stoppas om day>12.

Typer

Record-typer (poster)

Exempel

```
TYPE Date = RECORD
              year:  Year;
              month: Month;
              day:   Day
            END;

VAR date:Date;
```

Värt att notera

- Närmaste motsvarigheten till tupler i SML.
- Delarna väljs ut med "punktnotation"

```
date.year  date.month  date.day
```

- Det finns ingen notation för record-konstanter...

Exempel: lagra dagens datum i variabeln date:

```
date := ???
```

Man måste ge värde till en del i taget.

```
date.year:=1995;
date.month:=4;
date.day:=20;
```


Typer

Uppräkningstyper

Exempel

```
TYPE Month = (Jan,Feb,Mar,Apr,Maj,Jun,
              Jul,Aug,Sep,Oct,Nov,Dec);
```

Pragmatik

- Program blir mer självdokumenterande...
- En uppräkningstyp är en helt ny typ, som inte kan blandas med andra typer

Konverteringsfunktioner

- Om *typ* är en uppräkningstyp, x har typen typ, n har typen CARDINAL:

ORD(x): CARDINAL
VAL(*typ*,n): *typ*

Exempel:

```
ORD(Jan)  = 0
VAL(Month,0)  = Jan
```

Fördefinierade typer i Modula-2

Vad är en datatyp?

- En mängd av möjliga värden
- Ett antal operationer som kan utföras på värdena

Hur lagras värden i minnet?

- Det behöver man oftast inte bekymra sig för!
- I Modula-2:
  - Alla värden lagras som bit-följder av en bestämd längd.
  - Typen på värdet behöver inte framgå av bitmönstret som lagras, eftersom man vet varje variabels typ.

Fördefinierade typer

```
INTEGER
CARDINAL
REAL
BOOLEAN
CHAR
```


Numeriska typer

Namn	Beskrivning	Exempel	Namn i SML
`INTEGER`	Heltal	`-123 0 345`	`int`
`CARDINAL`	Heltal >=0	`98 0`	`-`
`REAL`	Reella tal	`12.34 -3.14`	`real`

Operationer

Binära operatorer

`+ - *`

- Definierade för de tre numeriska typerna. Operanderna måste ha samma typ (jmf SML).

`/`

- Endast för REAL

`DIV MOD`

- Heltalsdivision & rest. Definierade för **`INTEGER`** och **`CARDINAL`**.

Numeriska typer

Övriga operationer

Typkonverteringsfunktioner

- INTEGER: CARDINAL -> INTEGER
- CARDINAL: INTEGER -> CARDINAL
- FLOAT: CARDINAL -> REAL
- TRUNC: REAL -> CARDINAL

Andra fördefinierade operationer

- ABS
- ODD

Modulen MathLib

- Diverse funktioner på reella tal.

  Modula-2-systemet på våra datorer:

  `sqrt exp ln sin cos arctan entier real`

- Måste importeras för att kunna användas:

```
FROM MathLib IMPORT sqrt, entier;
```


Sanningsvärden

Namn	Beskrivning	Värden	Namn i SML
`BOOLEAN`	Sanningsvärden	`FALSE TRUE`	`bool`

Logiska operatorer

Namn	Beskrivning	Namn i SML
`NOT`	Logisk negation	`not`
`AND`	Konjunktion	`andalso`
`OR`	Disjunktion	`orelse`

- **`AND`** och **`OR`** beräknar vänstra operanden först och högra operanden bara om det behövs för att veta resultatet.

Jämförelser

`= # < <= > >=`

- Definierade för de fördefínierade typerna (ej strängar)
- **`#`** är operatorn för "ej lika med"

Tecken

Namn	Beskrivning	Värden	Namn i SML
CHAR	Tecken	`"a" "7" 12C`	-

- Tecken lagras enligt ASCII-koden eller någon utökning av denna.
  - IsoLatin1 används på våra datorer (UNIX).
  - En annan utökning används i MSDOS på PC-datorer.
  - En annan utökning används på Macintosh.

    Å Ä Ö och andra tecken som inte finns i ASCII-koden har olika koder i de olika utökningarna.

Operationer

- ORD: CHAR -> CARDINAL
- CHR: CARDINAL -> CHAR
- Alla jämförelseoperatorer

Modulen InOut

- **Write**, **Read**
- ...

Fördefinierade typer, allmänt

Operatorprioritet

= # < <= > >=

+ - OR

*** / DIV MOD AND**

NOT

- **AND** och **OR** har "fel" prioritet!

  age>12 AND age<20

  betyder

  age>(12 AND age)<20

  så man måste sätta ut parenteser

  (age>12) AND (age<20)

Fördefinierade funktioner

- En lista finns i Appendix D i läroboken.
- MathLib och andra biblioteksmoduler beskrivs i Appendix E och F.
  - Olika Modula-2-system skiljer sig åt.

    Se Laboration 4 för info om vårt Modula-2-system!

Skalära typer, allmänt

Skalära typer

Typer med "enkla" värden kan placeras på en "skala"

- **CARDINAL INTEGER REAL**
- **BOOLEAN**
- **CHAR**
- Uppräkningstyper

Inte skalära: poster, strängar och andra sammasatta värden

Operationer på skalära typer utom **REAL**

- Bilda delintervall

  [1..12] ["A".."Z"] [Jun..Aug]
- Konvertera till och från **CARDINAL** med **VAL** och **ORD**
- Öka och minska variabler med **INC** och **DEC** ...
- **FOR**-loopar ...
- Göra val med **CASE** ... **OF** ...

Operationer på skalära typer inklusive **REAL**

- Jämföra med jämförelseoperatorerna **= # < <= > >=**