Velkommen til INF2100

Jeg er *Dag Langmyhr* (dag@ifi.uio.no).

Dagens tema:

- Hva går kurset ut på?
  - Bakgrunn for kurset
  - Hvordan gjennomføres kurset?
  - Hvordan får man det godkjent?
- Programmeringsspråket **Pascal2016**
  - En kort oversikt
  - Syntaks
  - Et par eksempler

Bakgrunnen for INF2100

I begynnerkursene har dere lært å programmere, men bare ganske små programmer ($\leq$ 500 linjer).

Hensikten med INF2100 er

- å gi mer programmeringstrening
- forstå mekanismene man trenger i større programmer (som objektorientering og moduler)

Prosjektet

Prosjektet er valgt fordi det har en nytteverdi i seg selv:

Prosjekt:

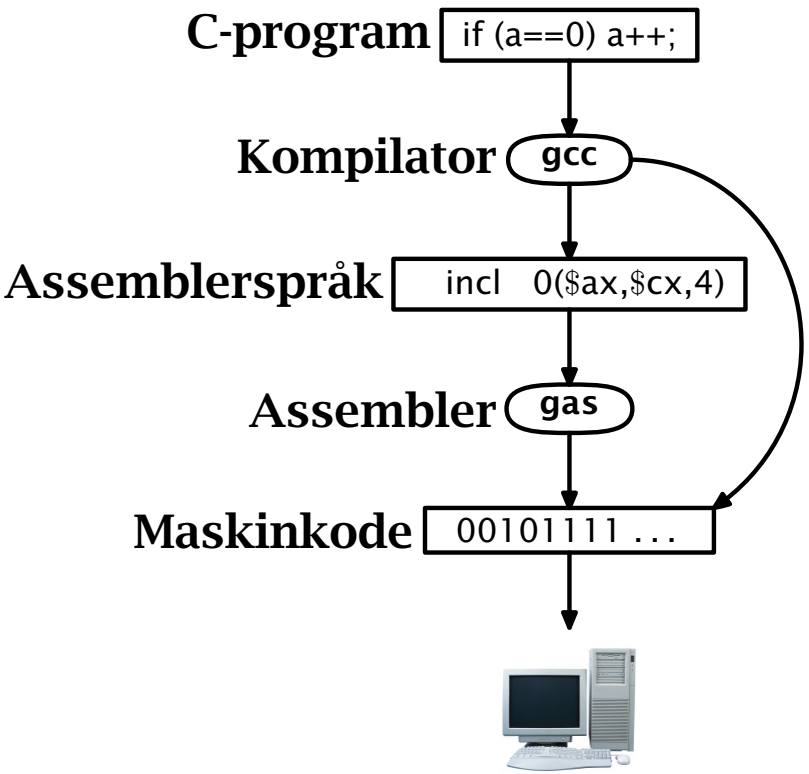
*Skriv en kompilator for programmeringsspråket **Pascal2016**.*

Dette gir

- eksempel på hvorledes et programmeringsspråk er definert og bygget opp
- forståelse for hvordan en kompilator fungerer
- kjennskap til maskin- og assemblerspråk

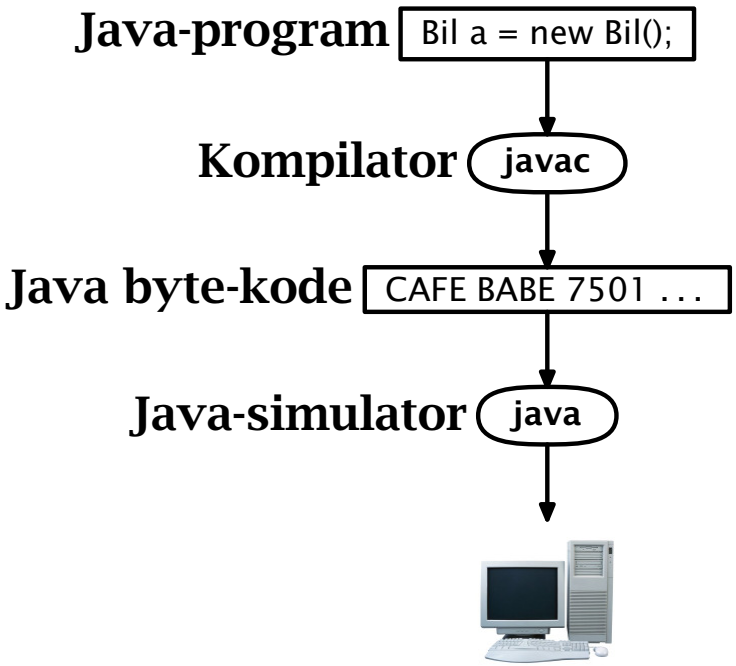
Hva gjør en kompilator?

En kompilator oversetter et program til en annen kode, oftest maskin- eller assemblerkode.

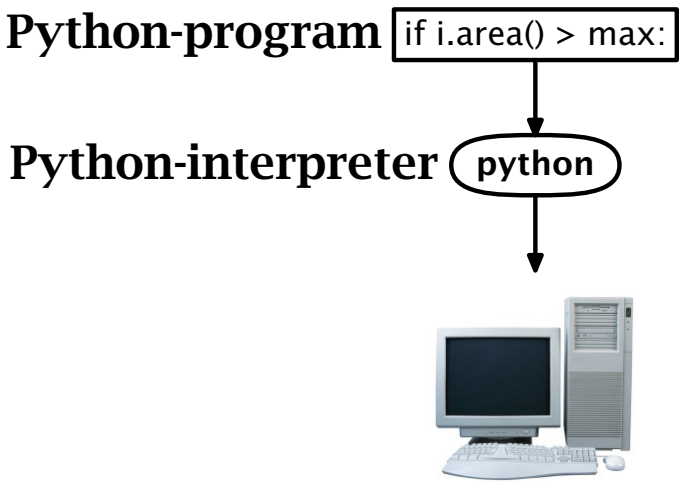


Noen kompilatorer oversetter til en egen kode som *tolkes* av en interpreter.

(Det finnes også Java-kompilatorer som lager maskinkode, men de er ikke så vanlige.)

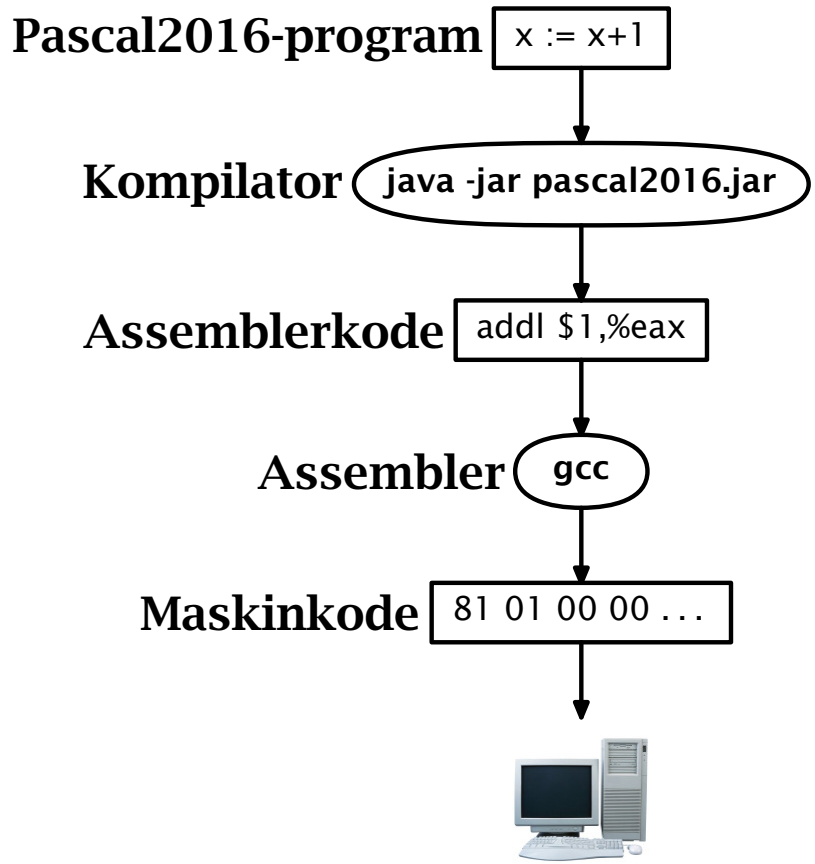


Noen interpretere leser kildekoden direkte (men vil oversette den til en passende form internt).



Opplegget for Pascal2016 er det samme som for C og det vanligste for kompilatorer.

Oppgaven deres er å skrive denne kompilatoren.



Ulike programmeringsspråk

I dette kurset skal vi innom flere språk:

- **Java**, som benyttes til implementasjonen.
- **Pascal2016**, som er målet for prosjektet.
- **x86-maskinkode** til datamaskinen vår
- **assemblerspråket** til x86.

Oppbyggingen av kurset

Kurset har seks hovedkomponenter:

- **Forelesningene** holdes i bolker; se http://www.uio.no/studier/emner/matnat/ifi/INF2100/h16/timeplan/. Forelesningene blir lagt ut som screencast etterpå.
- **Gruppeøvelser** 2 timer hver uke.
- **Gruppearbeid** for å løse oppgavene. Man jobber to og to (eller alene om man vil).
- **Kompendiet** gir en innføring i Pascal2016 og prosjektet.
- **Basiskoden** gir starthjelp til programmeringen.
- **Nettsidene** er også en viktig informasjonskanal.

Godkjenning av kurset

Kurset har ikke karakterer, bare bestått/ikke bestått.

Det er fire obligatoriske oppgaver. Når de er godkjent, er kurset godkjent.

Men . . .

Mot slutten av semesteret vil noen bli plukket ut til en samtale om programmet de har laget. Dette kan man stryke på!

Siden man normalt jobber i lag, forventes at

- begge har kjennskap til hovedstrukturen i programmet
- begge kan identifisere sin del av programmet (som skal være rundt halvparten) der de kan forklare nøye hvorfor koden er blitt slik den er.

Samarbeid og fusk

Samarbeid og utveksling av ideer er bra!

Kopiering og fusk er ikke bra!

Gode råd for samarbeid

- Snakk gjerne med andre om ideer.
- Kopier aldri andres kode.

Ulike programmeringshjelpemidler

Når man skal programmere et større prosjekt, trenger man hjelpemidler:

- et kompileringssystem (à la make eller ant)
- et dokumentasjonsopplegg (à la JavaDoc)
- et versjonskontrollsystem (à la Subversion)
- ...

Dette blir tatt opp i kurset (når dere er motivert for å høre om det ☺).

En siste ting

Det er ingen krav til utviklingsverktøy. Du kan bruke Emacs, Atom, Eclipse eller hva du vil.

Det finnes åpne kodebanker med gode utviklingsverktøy, som GitHub. Det er ikke lov å bruke disse hvis kildekoden deres blir tilgjengelig for alle.

En snarvei for INF1000-studenter

Førstegangs INF1000-studenter med programmeringserfaring i Java kan få ta INF2100 om de består en opptaksprøve.

Opptaksprøven foregår i **Modula** fredag kl 14.15–16.00.

Programmeringsspråket *Pascal*

- Laget 1970 av *Niklaus Wirth* ved ETH i Zürich.
- Formål:
  1. Begynnerundervisning i programmering
  2. Oppdra til å strukturere både data og setninger
  3. Kunne kjøre på en liten maskin og likevel kompilere raskt
- Var kanskje det mest populære programmeringsspråket 1973–1985.
- I dag husket med glede, men ikke brukt (unntatt i Delphi).

Hvorfor ble Pascal så populært?

Fra 1970 begynte det å komme personlige datamaskiner:

- Billige (typisk 20 000–50 000 kr)
- Små (typisk 16–64KB)

I starten var det eneste tilgjengelige programmeringsspråket **Basic**.

Programmeringsspråket *Basic*

```
10 i = 0
20 i = i + 1
30 PRINT i; " squared = "; i * i
40 IF i >= 10 THEN GOTO 60
50 GOTO 20
60 PRINT "Program Completed."
70 END
```

- Laget 1964 ved Dartmouth college i New Hampshire.
- Formål:
  1. Undervisning i programmering
  2. Benytte *time-sharing* så flere kunne bruke maskinen samtidig.
- Et meget enkelt språk som krevde svært få ressurser.

Basic var ikke populært hos informatikere

Professor *Bjørn Kirkerud* (Ifi)

Studenter som skal studere informatikk og som har programmert Basic, stiller betydelig svakere enn de som aldri har programmert!

Professor *Edsger Dijkstra*

It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration.

Hvorfor ikke?

Basic var veldig ustrukturert og lærte programmererne uvaner:

```
10 i = 0
20 i = i + 1
30 PRINT i; " squared = "; i * i
40 IF i >= 10 THEN GOTO 60
50 GOTO 20
60 PRINT "Program Completed."
70 END
```

- All struktur var basert på linjenummer.
- Det var lettest å rette feil ved å legge inn noen ekstra hopp.
- Variabelnavn kun A og A1.
- Tidlige versjoner gikk langsommere om du la inn kommentarer.

Betegnelsen **spagettiprogrammering** stammer fra Basic.

Pascal

Pascal ble oppfattet som redningen:

- Det var velstrukturert.
- Det var lett å implementere og krevde få ressurser.

Nå ville «alle» bruke Pascal (i hvert fall på småmaskiner):

- **CCITT** laget sin egen Pascal for telefonsentraler: **Chill**.
- Det amerikanske forsvaret laget sin egen versjon **Ada** for alle sine innebygde systemer.
- LaTeX-kjernen er skrevet i Pascal.

Men så forsvant Pascal fra arenaen mot slutten av 1980-tallet.

- Pascal er et undervisningsspråk; det er for eksempel ikke et systemprogrammeringsspråk.
- Pascal er ikke objektorientert (men noen har laget ulike oo-varianter).
- Småmaskinene ble større og kraftigere.
- Pascal har også sine særegenheter som irriterte programmerere.

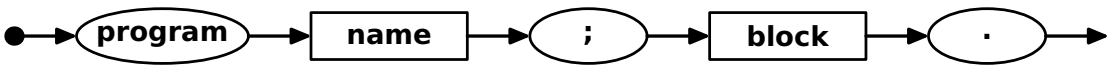
Programmeringsspråket Pascal2016

Vårt språk **Pascal2016** er en litt redusert utgave av Pascal.

Hovedprogrammet

Hovedprogrammet er rett og slett en **blokk** med et navn, som definert av dette **jernbanediagrammet**:

program



Terminaler (runde) er faste symboler.

Ikke-terminaler (firkanter) angir definisjon et annet sted.

Syntaks

Språkets syntaks (= grammatikk) defineres med jernbanediagrammer som gir en helt presis definisjon.

Men de forteller ikke alt om et språk.

Semantikk

Språkets betydning (= semantikk) forteller hva som skjer under kjøringen. Noen bruker en formell notasjon; vi bruker norsk.

Eksempel: En funksjons returverdi angis ved å tilordne den til funksjonen.

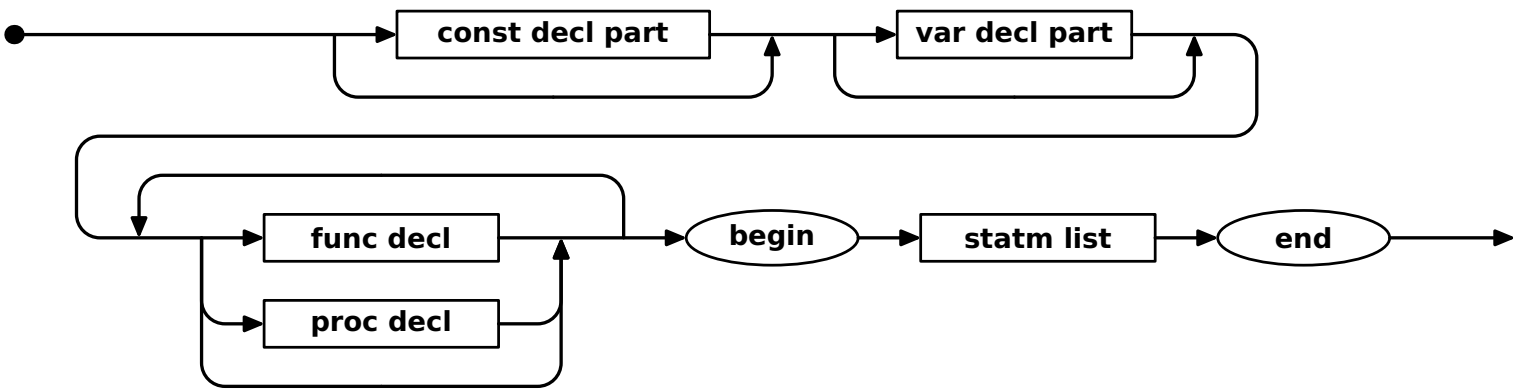
Annen informasjon

I tillegg finnes andre opplysninger som programmereren må vite om, som biblioteket, hvordan kommentarer ser ut etc.

Blokker

Pascal er et **blokkstrukturert** språk der alle definisjoner (av konstanter, variabler, funksjoner og prosedyrer) ligger i en blokk, Blokker inneholder også setninger.

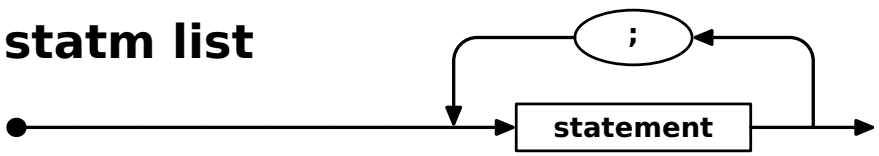
block



Forskjell fra Java

I Pascal kan vi ha blokker inni blokker inni . . .

Setninger

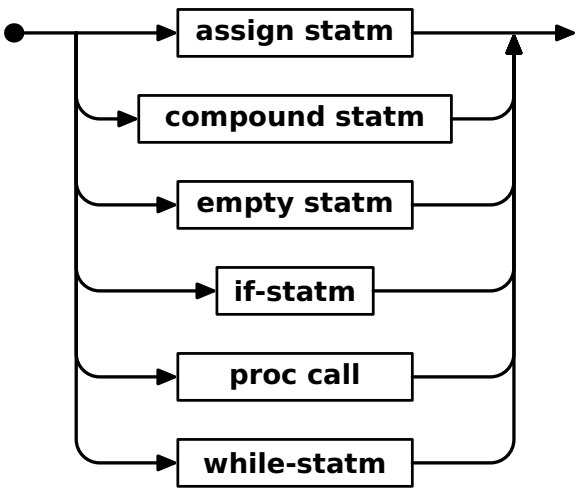


Forskjell fra Java

I Pascal brukes semikolon som **skilletegn** mellom setninger. (I Java er det avslutningstegn.)

Noen setninger er ganske enkle.

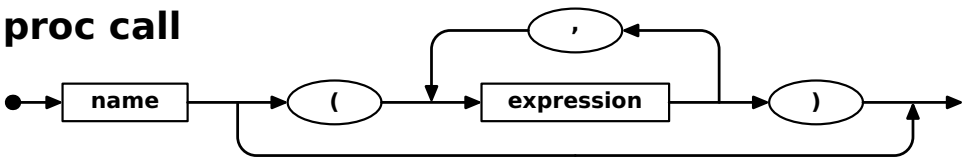
statement



empty statm

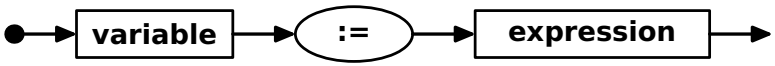
compound statm

proc call

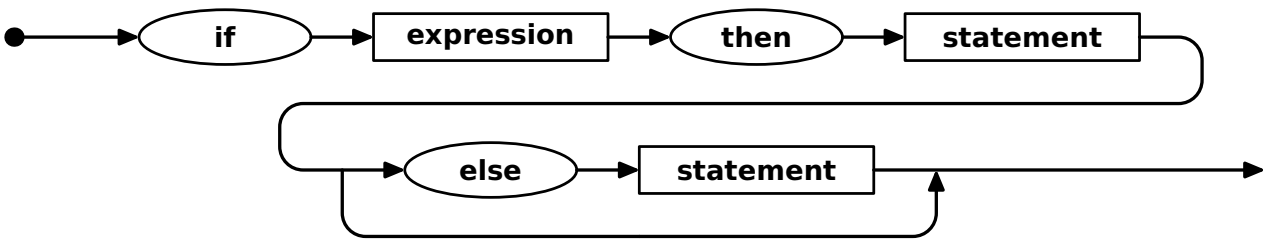


De andre setningene kjenner vi fra andre språk, selv om syntaksen er litt anderledes:

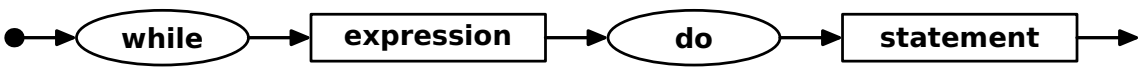
assign statm



if-statm



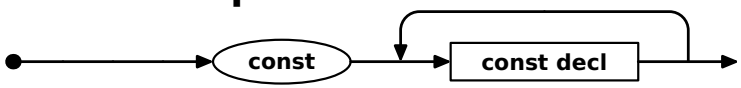
while-statm



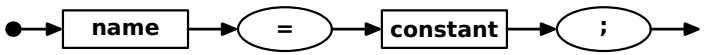
Konstanter

Det er lett å definere konstanter:

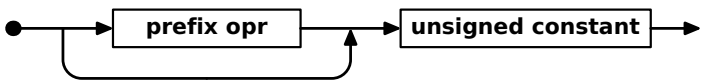
const decl part



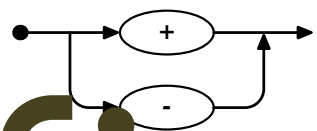
const decl



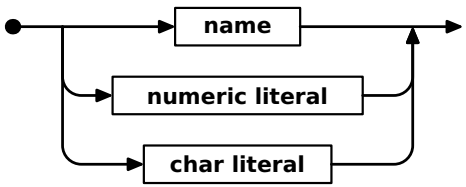
constant



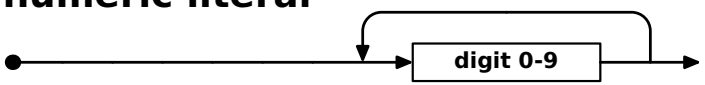
prefix opr



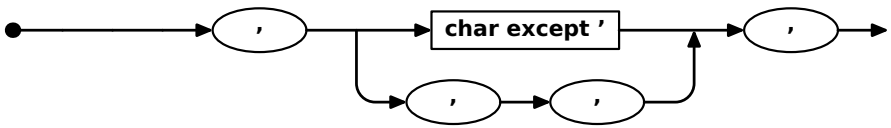
unsigned constant



numeric literal



char literal



Prosedyrer og funksjoner

I Pascal skiller man tydelig mellom prosedyrer (som tilsvarer `void`-funksjoner i Java) og funksjoner.

- Prosedyrer returnerer aldri en verdi, så de kan bare kalles som setninger.
- Funksjoner returnerer alltid en verdi, så de kan bare kalles som del av et uttrykk.
- Funksjonens returverdi angis ved å tilordne den til «funksjonen selv» (se eksempel senere).
- Prosedyrer og funksjoner er definert med en blokk, så de kan inneholde lokale prosedyrer og funksjoner.
- Rekursive prosedyrer og funksjoner (som kaller seg selv) er lov.

Er dette lov?

```
if (A = 15) then begin
   A := A + 1;
end
```

```
while A <> 0 do
   A = A/2;
```

```
var v1: integer, v2: char;
```

```
const C1 = 1;
      C2 = C1 + 1;
```

Jernbanediagrammene gir oss svaret.

Store og små bokstaver

Vi skille ikke på store og små bokstaver i Pascal. `integer`, `Integer` og `INTEGER` er samme navnet.

Kommentarer

Kommentarer kan angis på to ulike måter:

```
{...}    /*...*/
```

Utskrift

Utskrift skjer med prosedyren `write` som kan ha vilkårlig mange parametre av ulik type. Konstanten `eol` angir linjeskift.

Standardeksemplet

Alle presentasjoner av programmeringsspråk starter med dette eksemplet:

```
program Hei;
begin
   write('H', 'e', 'i', '!', eol)
end.
```

Inntil dere har laget deres egen kompilator, kan dere bruke denne:

```
$ ~inf2100/pascal2016 hei.pas
This is the Ifi Pascal2016 compiler (2016-08-22)
Parsing... checking... generating code...OK
Running gcc -m32 -o hei hei.s -L. -L/hom/inf2100 -lpas2016
$ ./hei
Hei!
```

Et primtallprogram

```
/* This program prints all primes less than 1000
   using a technique called "The sieve of Eratosthenes". */

program Primes;

const Limit = 1000;

var prime : array [2..Limit] of Boolean;
    i     : integer;
```

Et mer realistisk eksempel

```
procedure FindPrimes;
var i1 : integer;
    I2 : Integer;
begin
   i1 := 2;
   while i1 <= Limit do
   begin
      i2 := 2*i1;
      while i2 <= Limit do
      begin
         prime[i2] := false;
         i2 := i2+i1
      end;
      i1 := i1 + 1
   end
end; {FindPrimes}
```

Et mer realistisk eksempel

```
procedure P4 (x : integer);
begin
   if x < 1000 then write(' ');
   if x <  100 then write(' ');
   if x <   10 then write(' ');
   write(x);
end; {P4}


procedure PrintPrimes;
var i        : integer;
    NPrinted : integer;
begin
   i := 2;  NPrinted := 0;
   while i <= Limit do
   begin
      if prime[i] then
      begin
         if (NPrinted > 0) and (NPrinted mod 10 = 0) then write(eol);
         P4(i);  NPrinted := NPrinted + 1;
      end;
      i := i + 1;
   end;
   write(eol)
end; {PrintPrimes}
```

Et mer realistisk eksempel

```
begin {main program}
   i := 2;
   while i <= Limit do begin prime[i] := true;  i := i+1 end;

   /* Find and print the primes: */
   FindPrimes;  PrintPrimes;
end. {main program}
```

Et mer realistisk eksempel

```
$ ~inf2100/pascal2016 primes.pas
This is the Ifi Pascal2016 compiler (2016-08-22)
Parsing... checking... generating code...OK
Running gcc -m32 -o primes primes.s -L. -L/hom/inf2100 -lpas2016
$ ./primes
   2   3   5   7  11  13  17  19  23  29
  31  37  41  43  47  53  59  61  67  71
  73  79  83  89  97 101 103 107 109 113
 127 131 137 139 149 151 157 163 167 173
 179 181 191 193 197 199 211 223 227 229
 233 239 241 251 257 263 269 271 277 281
 283 293 307 311 313 317 331 337 347 349
 353 359 367 373 379 383 389 397 401 409
 419 421 431 433 439 443 449 457 461 463
 467 479 487 491 499 503 509 521 523 541
 547 557 563 569 571 577 587 593 599 601
 607 613 617 619 631 641 643 647 653 659
 661 673 677 683 691 701 709 719 727 733
 739 743 751 757 761 769 773 787 797 809
 811 821 823 827 829 839 853 857 859 863
 877 881 883 887 907 911 919 929 937 941
 947 953 967 971 977 983 991 997
```

Oppsummering

- Pascal har mange språkelementer (variabler, funksjoner, setninger etc) felles med C og Java, men de ser oftest litt anderledes ut.
- Bruk jernbanediagrammene til å avgjøre hva som er lov.
- Bruk også kompilatoren for å se hva som er lov, og hva som skjer når man kjører koden.