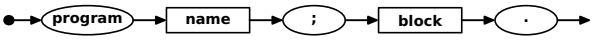
primes.pas

```
1
2   /* This program prints all primes less than 1000
3      using a technique called "The sieve of Eratosthenes". */
4
5   program Primes;
6
7   const Limit = 1000;
8
9   var prime : array [2..Limit] of Boolean;
10      i     : integer;
11
12  procedure FindPrimes;
13  var i1 : integer;
14      I2 : Integer;
15  begin
16     i1 := 2;
17     while i1 <= Limit do
18     begin
19        i2 := 2*i1;
20        while i2 <= Limit do
21        begin
22           prime[i2] := false;
23           i2 := i2+i1
24        end;
25        i1 := i1 + 1
26     end
27  end; {FindPrimes}
28
29  procedure P4 (x : integer);
30  begin
31     if x < 1000 then write(' ');
32     if x <  100 then write(' ');
33     if x <   10 then write(' ');
34     write(x);
35  end; {P4}
36
37  procedure PrintPrimes;
38  var i        : integer;
39      NPrinted : integer;
40  begin
41     i := 2;  NPrinted := 0;
42     while i <= Limit do
43     begin
44        if prime[i] then
45        begin
46           if (NPrinted > 0) and (NPrinted mod 10 = 0) then write(eol);
47           P4(i);  NPrinted := NPrinted + 1;
48        end;
49        i := i + 1;
50     end;
51     write(eol)
52  end; {PrintPrimes}
53
54  begin {main program}
55     i := 2;
56     while i <= Limit do begin prime[i] := true;  i := i+1 end;
57
58     /* Find and print the primes: */
59     FindPrimes;  PrintPrimes;
60  end. {main program}
```

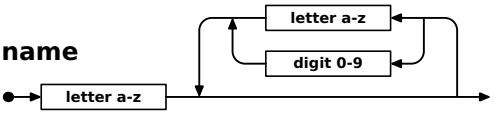
```
$ ~inf2100/pascal2016 primes.pas
This is the Ifi Pascal2016 compiler (2016-05-14)
Parsing... checking... generating code...OK
Running gcc -m32 -o primes primes.s -L. -L/hom/
inf2100 -lpas2016

$ ./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
```

program

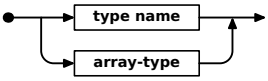


name



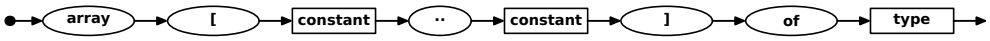
---

type



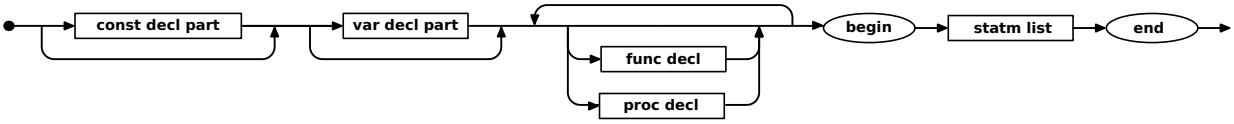
type name

array-type



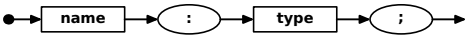
---

block

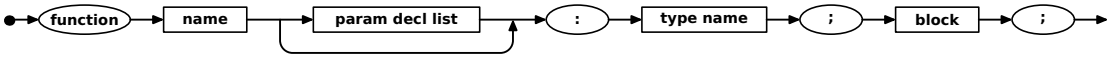


var decl part

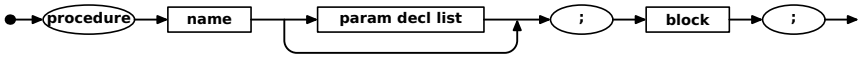
var decl



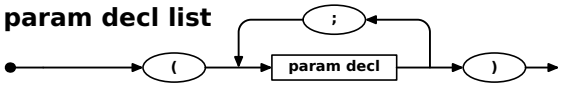
func decl



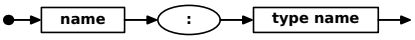
proc decl



param decl list

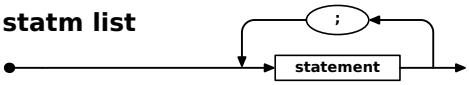


param decl

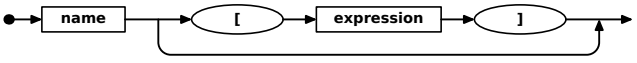


---

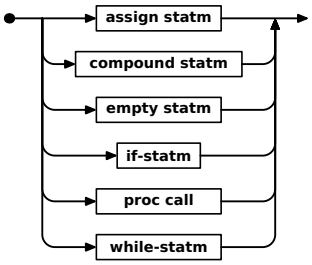
statm list



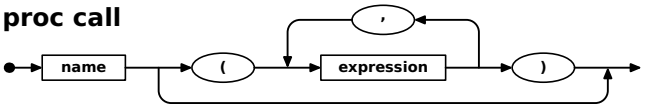
variable



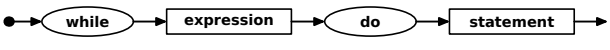
statement



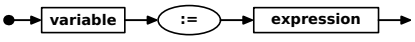
proc call



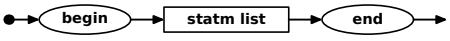
while-statm



assign statm



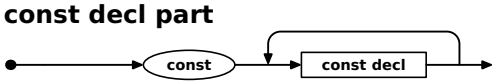
compound statm



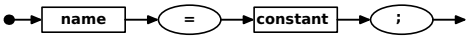
empty statm

if-statm

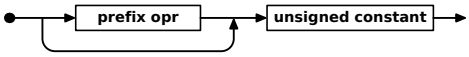
const decl part



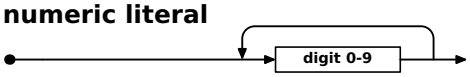
const decl



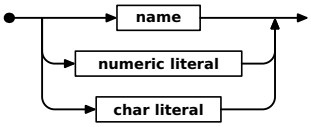
constant



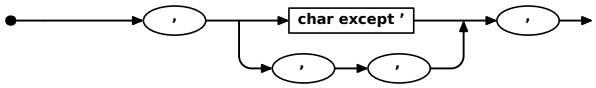
numeric literal



unsigned constant

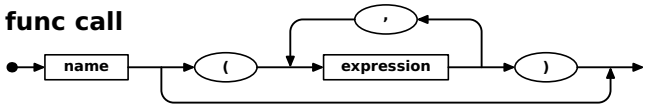


char literal



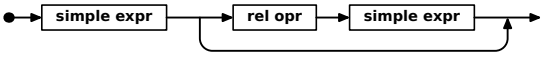
---

func call

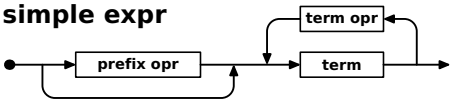


inner expr

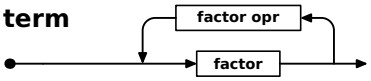
expression



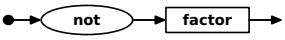
simple expr



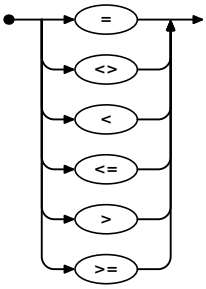
term



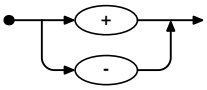
negation



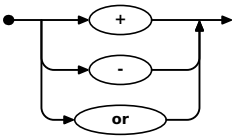
rel opr



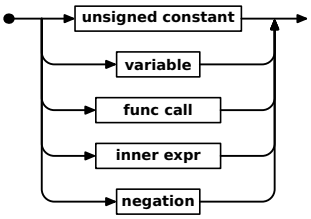
prefix opr



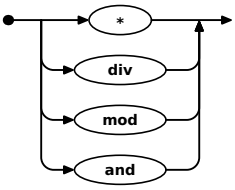
term opr



factor



factor opr



Forskjell fra Java: I Pascal2016, skiller man ikke på store og små bokstaver. Dette betyr at `integer`, `Integer` og `INTEGER` er tre ulike former av samme navn.

Forskjell fra Java: I Pascal2016 kan man ha blokker utenpå hverandre, så man kan ha funksjoner som er lokale inni andre funksjoner.

Forskjell fra Java: Der man i Java skriver «`'\''`» for å få et enkelt anførselstegn, skriver vi i Pascal2016 «`''''`».

Forskjell fra Java: I Java kan man angi en initialverdi for variabelen; det kan man ikke i Pascal2016.

Forskjell fra Java: I Pascal2016 har man ingen **return**-setning; i stedet tilordner man en verdi til funksjonen selv, slik det er vist i funksjonen `GCD`.

Forskjell fra Java: I Pascal2016 benyttes semikolon som *skilletegn* mellom setninger, så siste setning før en `end` skal ikke ha noe semikolon. (Men om man legger inn et semikolon der allikevel, betyr det bare at det står en ekstra tom setning før `end`, og det betyr jo ingenting for kjøringen)

Forskjell fra Java: I Pascal2016 er det ikke lov å kalle en funksjon som om den var en prosedyre.

```
function GCD (m: integer; n: integer): integer;
begin
   if n = 0 then
      GCD := m
   else
      GCD := GCD(n, m mod n)
end; { GCD }
```

Pascal2016-program

Pascal2016 er et såkalt **blokkstrukturert** programmeringsspråk der alle deklarasjoner er plassert i blokker. Innmaten av program, funksjoner og prosedyrer er derfor blokker. I en blokk kan man deklarere **konstanter**, dvs verdier med et navn. En konstant kan definere som for eksempel, et navn på en annen konstant (muligens med skifte av fortegn)

I Pascal2016 skiller man mellom **funksjoner** og **prosedyrer**: de førstnevnte returnerer alltid en verdi, de sistnevnte kan ikke det.

Kommentarer kan skrives på to ulike former (begge formene kan strekke seg over flere linjer):

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

Predefinerte deklarasjoner

`Boolean` er en type som har to logiske verdier: `False` og `True`.

`Char` er typen for å lagre enkelttegn (som i Java). I dette kurset er det bare aktuelt å bruke ASCII-tegn.

`EoL` er en konstant av typen `Char`. Internrepresentasjonen er 10 og konstanten brukes ved utskrift for å angi linjeskift.

`False` er en konstant av typen `Boolean`; den representeres internt med verdien 0.

`Integer` er typen for heltall.

`True` er en konstant av typen `Boolean`; den representeres internt med verdien 1.

`Write` er standardprosedyren for utskrift; den er beskrevet i neste avsnitt.

Utskrift

Pascal2016-programmer kan skrive ut ved å benytte den helt spesielle prosedyren `write`. Denne prosedyren kan ha vilkårlig mange parametre, og parametrene kan være av typen `integer`, `char` eller `Boolean`. Her er noen eksempler:

```
write('H', 'e', 'i', '!', eol);
write(2, '+', 2, '=', 2+2, eol);
```

Forskjeller til standard Pascal

Som nevnt er Pascal2016 en forenklet utgave av Pascal, og følgende er utelatt:

- `case`-, `for`-, `repeat`- og `with`-setningene
- flyt-tal
- file og alt rundt lesing og skriving (unntatt `write`)
- dynamisk allokering og pekere
- record (som tilsvarer `struct` i C)
- var-, funksjons- og prosedyreparametre
- sjekking under kjøring av array-grenser og andre verdier