Pascal in EBNF

Resources on Pascal
- Jim Welsh & John Elder, Introduction to Pascal
- Any other book you know
- gpc documentation
- Internet resources – links in the course site

EBNF
Meta-notation for describing the grammar of a language

- **Nonterminals** = concepts of the language, written in our notation in normal font or with underscore.
- **Terminals** = actual legal strings, written in bold font, or between “ ”.
- | is choice among several possibilities
- [ ] enclose optional constructs
- { } encloses zero or more repetitions

- One nonterminal is designated as the **start** of any derivation.
- A sequence of terminals not derivable from the **start symbol** by rules of the grammar is illegal.
- **Example:**
  
  if-stat = if expression then statement [ else statement ]
### EBNF Example

```
if-stat = if expression then statement [ else statement ]
```

### Pascal Program Structure

The base Pascal nonterminal:

```
program = program-heading block "."
```

```
program ProgramName(input,output);
const n = 100;
type Color = (Red, Green, Blue, Yellow);
  Index = 1..100;
var c : Color;
  i,j : Index; { comment - indexes }

function func(m,n : Integer; c : Color) : boolean;
var count : integer;
begin
  if m = 42 then
    begin
      ...
    end;
  func := true
end;
begin
  writeln( func(n*n,n-2,Red) )
end.
```
program ProgramName(input,output);
const n = 100;
type Color = (Red, Green, Blue, Yellow);
  Index = 1..100;
var c : Color;
  i,j : Index; { comment – indexes }
function func(m,n : Integer; c : Color) : boolean;
  var count : integer;
begin
  if m = 42 then
    begin...
    end;
  func := true
  end;
begin
  writeln( func(n*n,n-2,Red) )
  end.
Pascal Program Structure

declaration-part = [ label-declaration-part ]
[ constant-definition-part ]
[ type-definition-part ]
[ variable-declaration-part ]
procedure-and-function-declaration-part

const n = 100;
type Color = (Red, Green, Blue, Yellow);
   Index = 1..100;
var c : Color;
   i, j : Index; 
      { comment - indexes }

function func(m, n : Integer; c : Color) : boolean;
var count : integer;
begin
   if m = 42 then
      begin
         ...
      end;
      func := true
end;
Pascal Program Structure

type-definition-part =
  type type-definition ";" { type-definition ";" }

type-definition =
  identifier ";=" type

const n = 100;
type Color = (Red, Green, Blue, Yellow);
  Index = 1..100;
var c : Color;
  i, j : Index; { comment - indexes }

function func(m,n : Integer; c : Color) : boolean;
  var count : integer;
  begin
    if m = 42 then
      begin
        ...
      end;
    func := true
  end;

variable-declaration-part =
  var variable-declaration ";" { variable-declaration ";" }

variable-declaration =
  identifier-list ";" type

const n = 100;
type Color = (Red, Green, Blue, Yellow);
  Index = 1..100;
var c : Color;
  i, j : Index; { comment - indexes }

function func(m,n : Integer; c : Color) : boolean;
  var count : integer;
  begin
    if m = 42 then
      begin
        ...
      end;
    func := true
  end;
function func(m,n : Integer; c : Color) : boolean;
var count : integer;
begin if m = 42 then
begin ... end;
func := true
end;

Pascal Program Structure

statement-part = begin statement-sequence end

statement-sequence = statement { ; " statement }

statement = ... | procedure statement | ...

procedure-statement = procedure-identifier [ actual-parameter-list ]

begin writeln( func(n*n,n-2,Red) )
end.

Pascal Program Structure

procedure-and-function-declaration-part =
{ (procedure-declaration | function-declaration) ;; }

function-declaration = function-heading ;; function-body

function-heading = function identifier [ formal-parameter-list ] ;; result-type

function-body = block

function func(m,n : Integer; c : Color) : boolean;
var count : integer;
begin if m = 42 then
begin ... end;
func := true
end;
EBNF Tree

- We can see the EBNF derivation as a syntax tree of the program:

```
program
  program-heading block ".."
    Program identifier "(" identifier-list ")" ";" declaration-part statement-part
```

Data Types

- Pascal has 4 primitive types:
  integer, real, char, boolean

```pascal
var i : integer;
  hasPassed : boolean;
```

- We can also create our own types:
  - Enumerated types
    ```pascal
type Color = (Red, Green, Blue, Yellow);
```

Enumerated types are comparable:

```pascal
  Red < Blue = true,
  succ(Red) = Green,
  pred(Blue) = Green,
  ord(Yellow) = 3
```
Data Types

- Pascal has 4 primitive types:
  integer, real, char, boolean

```pascal
var i : integer
hasPassed : boolean
```

- We can also create our own types:
  - Subrange types
    ```pascal
type Letter = 'A' .. 'Z'
    Index = 3 .. 8
    Colorlist = Red .. Blue
```

- We can also create records which are complex types, like C structs

```pascal
record-type = record field-list end
```

```pascal
type date = record
    day : 1 .. 31;
    month : January .. December;
    year : 1900 .. 2000
end;
```
Arrays in Pascal

array-type = array "[ " index-type { "," index-type } " ]" of element-type.

- var A : array [1 .. 5] of real;
- var pens : array [Red .. Yellow] of record
  width : 1..3;
  kind : (Regular, Bold)
end;

for color := Red to Yellow do
  writeln( pens[color].width );