
CS 241

Control Structures

Christopher A. Gantz

SPS Undergraduate Program
Regis University
cgantz@regis.edu

Lecture 2a

Overview of CS 241: Program Format

Christopher A. Gantz
School of Professional Studies
Regis University
cgantz@regis.edu

Overview

- Reading
 - Nance textbook Pages 28-34
- Primary Pascal language Primitives
- Program Format
 - Construct Purpose/Usage Overview
 - Syntactical Definition & Representation
 - Construct Semantics
 - Examples

Primary Pascal Language Primitives

- Valid Identifiers
 - Reserved Words
 - Standard Words
 - User Defined Words
- Valid Sentences (i.e. Language statements)
 - Declaration Statements
 - Executable Statements (i.e. Processing statements)

Pascal Reserved Words

- Reserved words are identifiers with a specific semantic/meaning in Pascal.
- Their meaning can not be changed or altered in any way.
- The following is a list of Pascal reserved words

–	and	end	library	shl
–	array	exports	mod	shr
–	asm	file	nil	string
–	begin	for	not	then

Pascal Reserved Words (Cont.)

- **case** **function** **object** **to**
- **const** **goto** **of** **type**
- **constructor** **if** **or** **unit**
- **declare** **implementation** **packed** **until**
- **destructor** **in** **procedure** **uses**
- **div** **inherited** **program** **var**
- **do** **inline** **record** **while**
- **downto** **interface** **repeat** **with**
- **else** **label** **set** **xor**

Standard Pascal identifiers

- Standard identifiers have a predefined meaning, but a programmer could override that definition with one of her or his own.
- While this is possible, it is not advisable
- Directives are an example of standard Pascal identifiers
- Directives are used only in contexts where user-defined identifiers can't occur.
- Again, unlike reserved words, you can redefine standard directives, but it is advise that you don't.

Standard Pascal identifiers

- The following shows a list of Pascal's standard directives.
 - `__reader` `code` `index` `postlude`
 - `__writer` `conv` `name` `private`
 - `absolute` `export` `out` `public`
 - `assembler` `external` `overload` `stdcall`
 - `cdecl` `forward` `pascal` `virtual`

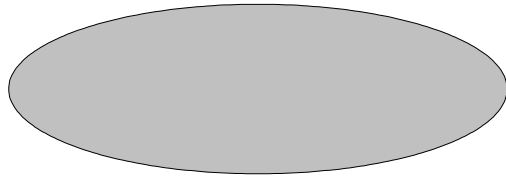
Pascal Syntax Diagrams

- Diagrams that graphically describe the syntax of a language construct or element
- There are three primary symbols utilized in syntax diagrams
- There are also flow arrows that define how the primary symbols are connected
- Syntax diagrams are a good shorthand for remembering the syntactical structures of the Pascal language.

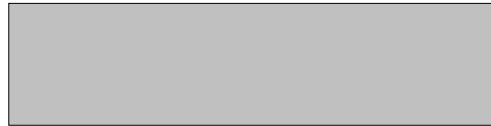
Pascal Syntax Diagrams

- The three primary graphical symbols are:
 - Named elements enclosed in an ellipse are: Reserved or Fully Defined
 - Named elements enclosed in a rectangle are: Defined Elsewhere
 - Named Separators are enclosed in a circle

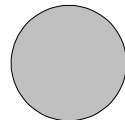
Pascal Syntax Diagrams (Cont.)



Reserved words or terms that cannot be further defined

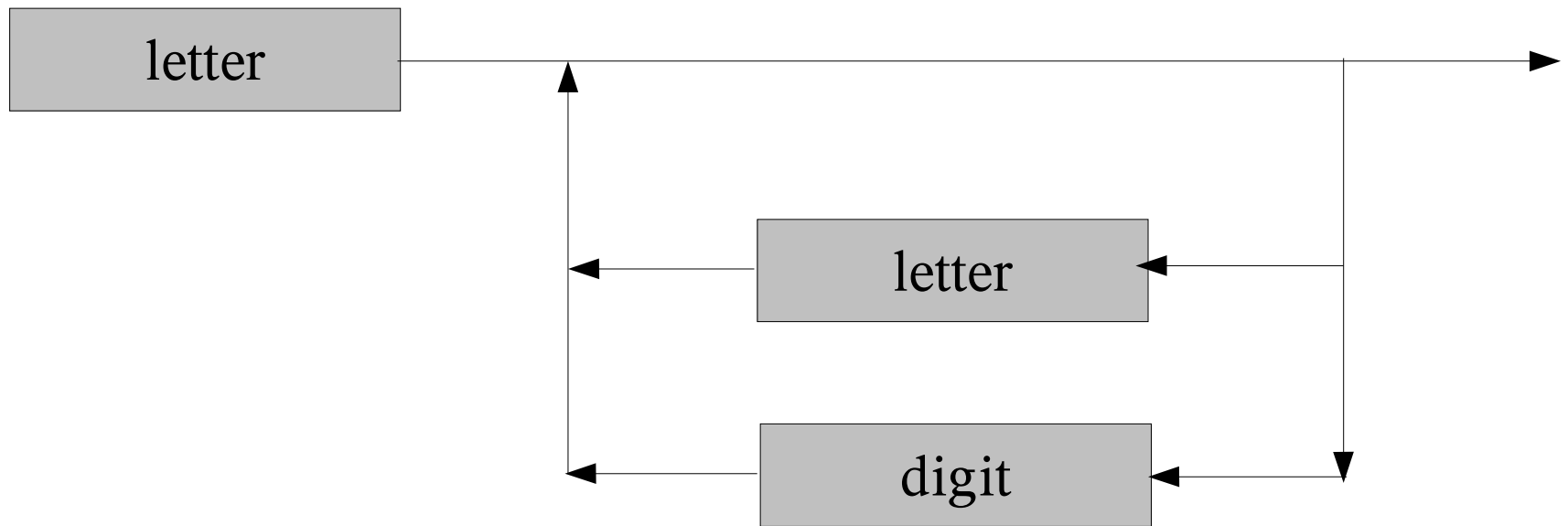


Items that are defined by another diagram



Any form of a separator

Identifier Syntax Diagram



User defined identifiers

- Alternative formal representations of syntactically correct user defined identifiers
 - BNF expression
 - $\langle \text{identifier} \rangle ::= \langle \text{letter} \rangle \langle \text{alphanumeric} \rangle^*$
 - $\langle \text{alphanumeric} \rangle ::= \langle \text{letter} \rangle \mid \langle \text{digit} \rangle$
 - $\langle \text{letter} \rangle ::= \text{a-z} \mid \text{A-Z}$
 - $\langle \text{digit} \rangle ::= \text{0-9}$
 - Regular Expression
 - Identifier = $[\text{a-zA-Z}][\text{a-zA-Z0-9}]^*$

Pascal Program Format

- The overall Pascal program format is comprised of the following three components:
 - Program heading
 - Declaration section which contains:
 - CONST definitions
 - TYPE definitions
 - VAR – variable declarations and subprogram definitions
 - Executable section
 - Contains a finite number of statements

Construct Purpose/Usage Overview (Cont.)

- The program heading is always the 1st statement of any Pascal program and contains the reserved word **PROGRAM**
 - E.g.
 - **PROGRAM <name> (<file list>)**

Pascal Program Format/Structure

```
{ Program Heading }  
Program ProgramStructure (input, output);  
{ PROGRAM <name> (<file list>); Declaration Statement}
```

```
{ Main Block }  
  { Declaration section (Optional) }  
  { constant definition section }  
CONST  
  { list of constants }  
  { Syntactical Format, Declaration Statements  
  <identifier1> = <value1>;  
  <identifier2> = <value2>;  
  
  . . .  
  
  <identifiern> = <valuen>;  
}
```

Pascal Program Format/Structure (Cont.)

TYPE

{ list of data types }

{ variable declaration section }

VAR

{ list of variables }

{ Syntactical Format, Declaration Statements

<identifier1> : <data type 1>;

<identifier2> : <data type 2>;

. . .

<identifiem> : <data type n>;

}

{ list of subprograms }

Pascal Program Format/Structure (Cont.)

```
{ Executable Section}
Begin
{ body of program }
{ Syntactical Format, Processing Statements
  <statement 1>;
  <statement 1>;

. . .

  <statement n - 1>;
  <statement n>;
}

End.
```

The Basic Pascal Program

{ Basic Program Structure Example }

{ Program Header }

program StructureExample(input,output);

{ Declaration Section }

const

MyName = 'Prof. Christopher A. Gantz';

var

example1 : integer;

example2 : string;

{ Executable Section }

begin

example1 := 1; { An executable statement. Note the ; }

example2 := 'One';

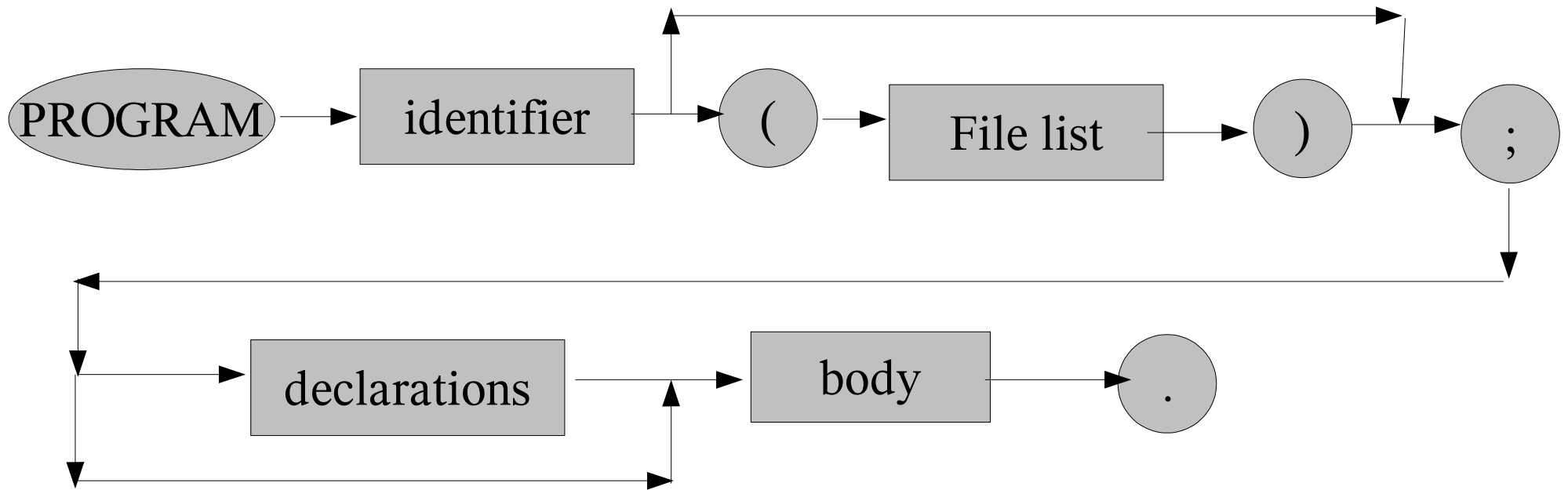
writeln('This is my name: ',MyName);

writeln('example1 value: ',Sample1);

writeln('example2 value: ',Sample2);

end.

Program Syntax Diagram



Program Syntax Diagram (Cont.)

- BNF representation
 - $\langle \text{program} \rangle ::= \text{PROGRAM } \langle \text{identifier} \rangle '(\langle \text{file list} \rangle)'; \langle \text{declarations} \rangle \langle \text{body} \rangle .$

Examples

Construct Summary
