Concepts of Programming Languages

Lecture 5 - Syntax

Patrick Donnelly

Montana State University

Spring 2014

Administrivia

Assignment:

Programming #1 : due 02.10
Homework #2 : due 02.19

Reading:

Chapter 3

A language that is simple to parse for the compiler is also simple to parse for the human programmer.

N. Wirth (1974)

Thinking about Syntax

Definition

The ***syntax*** of a programming language is a precise description of all its grammatically correct programs.

Precise syntax was first used with Algol 60, and has been used ever since.

Three levels:

- Lexical syntax
- Concrete syntax
- Abstract syntax

Levels of Syntax

Lexical syntax = all the basic symbols of the language (names, values, operators, etc.)

Concrete syntax = rules for writing expressions, statements and programs.

Abstract syntax = internal representation of the program, favoring content over form.

- C: `if ( expr ) ...discard ( )`
- Ada: `if ( expr ) then discard then`

Syntax of a Small Language: *Clite*

Why examine only a subset of C?

Syntax of a Small Language: *Clite*

Why examine only a subset of C?

Grammar Size for Various Languages

Language	**Pages**	**Reference**
Pascal	5	[Jensen & Wirth, 1975]
C	6	[Kernighan & Richie, 1988]
C++	22	[Stroustrup, 1997]
Java	14	[Gosling *et. al.*, 1996]

The Clite grammar fits on one page (next few slides),
so it's a far better tool for studying language design.

Clite Grammar: Statements

```
      Program  →  int main ( ) { Declarations Statements }

 Declarations  →    { Declaration }

  Declaration  →  Type Identifier [ [ Integer ]  ]
                  { , Identifier [ [ Integer ] ] }

         Type  →  int | bool | float | char

   Statements  →  { Statement }

    Statement  →  ; | Block | Assignment |
                  IfStatement | WhileStatement
```

Clite Grammar: Statements

```
        Block  →  { Statements}

   Assignment  →  Identifier [ [ Expression ] ]
                  = Expression ;

  IfStatement  →  if ( Expression ) Statement
                  [ else Statement ]

WhileStatement →  while ( Expression ) Statement
```

Clite Grammar: Expressions

```
Expression  →  Conjunction { || Conjunction }

Conjunction →  Equality { && Equality }

Equality    →  Relation [ EquOp Relation ]

EquOp       →  == | !=

Relation    →  Addition [ RelOp Addition ]

RelOp       →  < | <= | > | >=
```

Clite Grammar: Expressions

```
Addition   →  Term { AddOp Term }

   AddOp   →  + | -

     Term  →  Factor { MulOp Factor }

    MulOp  →  * | / | %

   Factor  →  [ UnaryOp ] Primary

 UnaryOp   →  - | !

 Primary   →  Identifier [ [ Expression ] ] | Literal |
              ( Expression ) | Type ( Expression )
```

Clite Grammar: Lexical Level

```
Identifier  →  Letter { Letter | Digit }

    Letter  →  a | b | ...| z | A | B | ...| Z

     Digit  →  0 | 1 | ...| 9

   Literal  →  Integer | Boolean | Float | Char

   Integer  →  Digit { Digit }

   Boolean  →  true | false

     Float  →  Integer .  Integer

      Char  →  'ASCII Char'
```

Issues Not Addressed by this Grammar

1. Comments
2. Whitespace
3. Distinguishing one token <= from two tokens < =
4. Distinguishing identifiers from keywords like if

These issues are addressed by identifying two levels:

- lexical level
- syntactic level

Lexical Syntax

Input: a stream of characters from the ASCII set, keyed by a programmer.

Output: a stream of tokens or basic symbols, classified as follows:

Identifiers	e.g., Stack, x, i, push
Literals	e.g., 123, 'x', 3.25, true
Keywords	bool char else false float if int main true while
Operators	= || && == != < <= > >= + - * / !
Punctuation	; , { } ( )

Whitespace

Whitespace is any space, tab, end-of-line character (or characters), or character sequence inside a comment

No token may contain embedded whitespace
(unless it is a character or string literal)

Example:

- >= one token
- > = two tokens

Whitespace Examples in Pascal

Legal or Illegal?

```
while a < b do
```

Whitespace Examples in Pascal

Legal or Illegal?

`while a < b do` legal spacing between tokens

Whitespace Examples in Pascal

Legal or Illegal?

```
while a < b do
```
legal spacing between tokens

```
whilea < bdo
```

Whitespace Examples in Pascal

Legal or Illegal?

`while a < b do`	legal	spacing between tokens
`whilea < bdo`	illegal	cannot tell boundaries

Whitespace Examples in Pascal

Legal or Illegal?

`while a < b do`	legal	spacing between tokens
`whilea < bdo`	illegal	cannot tell boundaries
`while a<b do`		

Whitespace Examples in Pascal

Legal or Illegal?

`while a < b do`	legal	spacing between tokens
`whilea < bdo`	illegal	cannot tell boundaries
`while a<b do`	legal	spacing not needed for <

Whitespace Examples in Pascal

Legal or Illegal?

`while a < b do`	legal	spacing between tokens
`whilea < bdo`	illegal	cannot tell boundaries
`while a<b do`	legal	spacing not needed for <
`whilea<bdo`		

Whitespace Examples in Pascal

Legal or Illegal?

`while a < b do`	legal	spacing between tokens
`whilea < bdo`	illegal	cannot tell boundaries
`while a<b do`	legal	spacing not needed for <
`whilea<bdo`	illegal	cannot tell boundaries

Whitespace Examples in Pascal

Legal or Illegal?

`while a < b do`	legal	spacing between tokens
`whilea < bdo`	illegal	cannot tell boundaries
`while a<b do`	legal	spacing not needed for <
`whilea<bdo`	illegal	cannot tell boundaries

Comments

Not defined in grammar

Clite uses / / comment style of C++

Identifier

Sequence of letters and digits, starting with a letter

- **if** is both an identifier and a keyword
- Most languages require identifiers to be distinct from keywords

In some languages, identifiers are merely predefined
(and thus can be redefined by the programmer)

Identifier

Redefining Identifiers can be dangerous!

```
program confusing;
const true = false;
begin
    if (a<b) = true then
        f(a)
    else ...
```

Identifier

Should Identifiers be case-sensitive?

Pascal	no
Modula	yes
C, C++	yes
Java	yes
PHP	partly yes, partly no

Concrete Syntax

Based on a parse of its Tokens:

; is a statement terminator

(Algol-60, Pascal use ; as a separator)

Rule for `IfStatement` is ambiguous:

> The else ambiguity is resolved by connecting an else with the last encountered else-less if. Stroustrup, 1991

Expressions in *Clite*

13 grammar rules

Use of meta braces - operators are left associative

C++ expressions require 4 pages of grammar rules

[Stroustrup]

C uses an ambiguous expression grammar

[Kernighan and Ritchie]

Associativity and Precedence

Clite Operator	Associativity
Unary - !	none
* /	left
+ -	left
< <= > >=	none
== !=	none
&&	left
||	left

Clite Equality, Relational Operators

Clite Equality, Relational Operators are non-associative.

(an idea borrowed from Ada)

Why is this important?

Clite Equality, Relational Operators

Clite Equality, Relational Operators are non-associative.

(an idea borrowed from Ada)

Why is this important?

In C++, the expression:

```
if (a < x < b)
```

is **not** equivalent to

```
if (a < x && x < b)
```

But it is error-free!

So, what does it mean?