Lecture 8

COP3402 FALL 2015 – DR. MATTHEW GERBER – 9/23/2015

FROM EURIPIDES MONTAGNE, FALL 2014

Tonight

First Steps toward Compiling PL/0
(Or Compiling Anything at All)

PL/0 Symbols

Consider this PL/0 program.

As with any language, we need to identify the vocabulary – what names and special symbols we accept as valid.

```
const m = 7, n = 85;
var i, x, y, z, q, r;
procedure mult;
  var a, b;
  begin
    a := x;  b := y; z := 0;
    while b > 0 do
    begin
      if odd x then z := z + a;
      a := 2 * a;
      b := b / 2;
    end
  end;
begin
  x := m;
  y := n;
  call mult;
end.
```

PL/0 Symbols

Consider this PL/0 program.

As with any language, we need to identify the vocabulary – what names and special symbols we accept as valid.

- For instance, we notice many reserved words in this example.

```
const m = 7, n = 85;
var i, x, y, z, q, r;
procedure mult;
  var a, b;
  begin
    a := x;  b := y; z := 0;
    while b > 0 do
    begin
      if odd x then z := z + a;
      a := 2 * a;
      b := b / 2;
    end
  end;
begin
  x := m;
  y := n;
  call mult;
end.
```

PL/0 Symbols

Consider this PL/0 program.

As with any language, we need to identify the vocabulary – what names and special symbols we accept as valid.

- For instance, we notice many reserved words in this example.
- There are also operators and special symbols.
  - *Operators* are the mathematical operators, along with := for assignment and = for equality.
  - *Special symbols* are the semicolon and comma.

```
const m = 7, n = 85;
var i, x, y, z, q, r;
procedure mult;
  var a, b;
  begin
    a := x;  b := y; z := 0;
    while b > 0 do
    begin
      if odd x then z := z + a;
      a := 2 * a;
      b := b / 2;
    end
  end;
begin
  x := m;
  y := n;
  call mult;
end.
```

PL/0 Symbols

Consider this PL/0 program.

As with any language, we need to identify the vocabulary – what names and special symbols we accept as valid.

- For instance, we notice many reserved words in this example.
- There are also operators and special symbols.
- Finally, there are numerals.
- (We'll leave names in black.)

```
const m = 7, n = 85;
var i, x, y, z, q, r;
procedure mult;
  var a, b;
  begin
    a := x;  b := y; z := 0;
    while b > 0 do
    begin
      if odd x then z := z + a;
      a := 2 * a;
      b := b / 2;
    end
  end;
begin
  x := m;
  y := n;
  call mult;
end.
```

Other Constructs

We also need to think about…

- Comments
  - /* C-style */
  - { Pascal-style }
  - // C++ style
  - # Scripting language style
- Separators
  - When is white space important?
  - When is white space *not* important?
  - Do we differentiate between types of white space? Spaces versus tabs versus new lines?

The Lexical Analyzer

The purpose of the scanner is to decompose the source program into all these tokens. It needs to:

1. Read the input characters of the source program
2. Group these input characters into *lexemes* – character sequences that match the patterns for tokens
3. Produce a token for each lexeme

The general order of operations is:

- Remove whitespace
- Identify tokens
- Create the symbol table
- Insert tokens into the symbol table
- Generate any errors
- Send the tokens to the parser

Some Definitions

To get much farther, we need to think about what a language is.

- Every language has an *alphabet* – a finite set of characters that are allowed in that language.
- Symbols are composed one or more characters; if more than one, the symbol is composed of multiple *concatenated* characters.
- In fact, the alphabet is actually the senior definition, so let's get more precise:

Definition (Alphabet): An ***alphabet*** is *any defined set of characters.*

Definition (Language): A ***language*** is *any set of strings over a fixed alphabet.*

Example Alphabets and Languages

Alphabet	Languages
{0, 1}	{0, 10, 100, 1000, 10000, …} {0, 1, 00, 11, 000, 111, …}
{a, b, c}	{abc, aabbcc, aaabbbccc, …}
{A...Z}	{TEE, FORE, BALL, …} {FOR, WHILE, GOTO, …}
{The ASCII Character Set}	{All legal Pascal programs}
{The Roman alphabet, with standard English punctuation}	{All grammatically correct English sentences}

Over any alphabet, Ø is the *empty language* – the language that contains no strings. {ε} is the language that only contains the *empty string* (denoted by ε). They'll be important later.

ASCII, Part I

Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII
0	00	NUL (null)	16	10	DLE (data link escape)	32	20	SP (space)	48	30	0
1	01	SOH (start of heading)	17	11	DC1 (device control 1)	33	21	!	49	31	1
2	02	STX (start of text)	18	12	DC2 (device control 2)	34	22	"	50	32	2
3	03	ETX (end of text)	19	13	DC3 (device control 3)	35	23	#	51	33	3
4	04	EOT (end of transmission)	20	14	DC4 (device control 4)	36	24	$	52	34	4
5	05	ENQ (enquiry)	21	15	NAK (negative acknowledge)	37	25	%	53	35	5
6	06	ACK (acknowledge)	22	16	SYN (synchronous idle)	38	26	&	54	36	6
7	07	BEL (bell)	23	17	ETB (end of transmission block)	39	27	'	55	37	7
8	08	BS (backspace)	24	18	CAN (cancel)	40	28	(	56	38	8
9	09	HT (horizontal tab)	25	19	EM (end of medium)	41	29	)	57	39	9
10	0A	LF (line feed)	26	1A	SUB (substitute)	42	2A	*	58	3A	:
11	0B	VT (vertical tab)	27	1B	ESC (escape)	43	2B	+	59	3B	;
12	0C	FF (form feed)	28	1C	FS (file separator)	44	2C	,	60	3C	<
13	0D	CR (carriage return)	29	1D	GS (group separator)	45	2D	-	61	3D	=
14	0E	SO (shift out)	30	1E	RS (record separator)	46	2E	.	62	3E	>
15	0F	SI (shift in)	31	1F	US (unit separator)	47	2F	/	63	3F	?

ASCII, Part II

Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII	Dec	Hex	ASCII
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	DEL

The ASCII Table

The ordinal number of a character *ch* is computed from its coordinates (*X*, *Y*) in the table by ord(*ch*) = 16*X* + *Y*.

	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
10(A)	LF	SUB	*	:	J	Z	j	z
11(B)	VT	ESC	+	;	K	[	k	{
12(C)	FF	FS	,	<	L	\	l	|
13(D)	CR	GS	-	=	M	]	m	}
14(E)	SO	RS	.	>	N	^	n	~
15(F)	SI	US	/	?	O	_	o	DEL

Next Time: Exam Review