**Arrays**

**Defining an array**

```
ARRAY array-name<number-of-array-elements> <$><length><array-elements>
<_TEMPORARY_><initial-values>;
```

**Referencing an array**

```
array-name[element-number];
```

The number of elements must be enclosed in either parentheses ( ), braces { }, or brackets [ ].

**Unknown Number of Elements**

Use an asterisk (*) within your brackets when defining an array.

Use the DIM function to return the number of elements in an array.

```
DIM(array-name);
```

**Two-Dimensional Arrays**

```
ARRAY array-name
[number-of-rows,number-of-columns];
array samplearray[3,2];
```

The example above creates an array named SampleArray which has 3 rows and 2 columns.

---

**Hash Objects**

A hash object is an in-memory table that contains key and data components.

**Hash Object and Iterator Process**

Declaring hash object or hash iterator object:

```
DECLARE hash object-name
(<argument_tag-1:value-1, ...>);
```

```
DECLARE hiter object-name
(‘hash-object-name’);
```

Defining a hash object:

```
object-name.ADD( );
object-name.DEFINEKEY(‘key-1’<,...’key-n’>);
object-name.DEFINEDATA(‘data-1’<,...’data-n’>);
object-name.DEFINEONE( );
object-name.OUTPUT( );
```

Using a hash object:

```
object-name.FIND( );
```

Retrieving a hash object with a hash iterator object:

```
object-name.FIRST( );
object-name.LAST( );
object-name.NEXT( );
object-name.PREV( );
```
Picture Formats

PROC FORMAT;
  PICTURE format-name <(format-options)>;
  <value-range-set-1= 'template-value' (template-options)>
  <value-range-set-n= 'template-value' (template-options)>;
RUN;

Options

Creating Custom Date, Time, Datetime Formats

DATATYPE=DATE | TIME | DATETIME enables the use of directives in the picture as a template to format date, time, or datetime values.

DEFAULT=length specifies the default length of the picture.

Creating Custom Numeric Formats

MULT|MULTIPLIER=n specifies a number to multiply the value by.

PREFIX='prefix' specifies a character prefix to place in front of the formatted value.

ROUND rounds the value to the nearest integer before formatting.

Creating Functions

PROC FCMP OUTLIB=libref.table.package;
  FUNCTION function-name(Arguments) <$><length>;
  . . . programming statements . . .
  RETURN(expression);
  ENDSUB;
QUIT;

Using Custom Functions

OPTIONS CMPLIB=libref.table | (libref.table-1...libref.table-n)

Perl Regular Expressions

Perl Regular Expressions Metacharacters

<table>
<thead>
<tr>
<th>Metacharacter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/.../</td>
<td>Starting and ending delimiter</td>
</tr>
<tr>
<td>(...)</td>
<td>Enables grouping</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>\d</td>
<td>Matches a digit (0-9)</td>
</tr>
<tr>
<td>\D</td>
<td>Matches a non-digit such as letter</td>
</tr>
<tr>
<td>\s</td>
<td>Matches a whitespace character</td>
</tr>
<tr>
<td>\w</td>
<td>Matches a group of characters</td>
</tr>
<tr>
<td>.</td>
<td>Matches any character</td>
</tr>
<tr>
<td>[...]</td>
<td>Matches a character in brackets</td>
</tr>
<tr>
<td>[^...]</td>
<td>Matches a character not in brackets</td>
</tr>
<tr>
<td>^</td>
<td>Matches the beginning of the string</td>
</tr>
<tr>
<td>$</td>
<td>Matches the end of the string</td>
</tr>
<tr>
<td>\b</td>
<td>Matches a word boundary</td>
</tr>
<tr>
<td>\B</td>
<td>Matches a non-word boundary</td>
</tr>
<tr>
<td>*</td>
<td>Matches the preceding character 0 or more times</td>
</tr>
<tr>
<td>+</td>
<td>Matches the preceding character 1 or more times</td>
</tr>
<tr>
<td>?</td>
<td>Matches the preceding character 0 or 1 times</td>
</tr>
<tr>
<td>{n}</td>
<td>Matches exactly n times</td>
</tr>
<tr>
<td>\</td>
<td>Overrides the next metacharacter such as a ( or ?)</td>
</tr>
</tbody>
</table>

PRXMATCH Function

PRXMATCH function searches for a pattern match and returns the position at which the pattern is found.

PRXMATCH (Perl-regular-expression, source);

PRXPARSE Function

PRXPARSE function returns a pattern identifier number that is used by other PRX functions and call routines.

pattern-ID-number=PRXPARSE (Perl-regular-expression);

PRXCHANGE Function

PRXCHANGE function performs a substitution for a pattern match

PRXCHANGE (Perl-regular-expression, times, source)

Advanced Functions

LAG<n>(column);
COUNT(string, substring<,modifiers>);
COUNTC(string, character-list<,modifiers>);
COUNTW(string, <delimiters> <,modifiers>);
FIND(string, substring<,modifiers><,start-position>);
FINDC(string, character-list<,modifiers> <,start-position>);
FINDW(string, word<,delimiters><,modifiers><,start-position>);