Agenda

- Lexer with richer vocabulary
- Vectors
- Stacks
- Well-ballanced expressions
- Infix and postfix expressions

Improved Lexer

New Tokens

- INTEGER: is a consecutive sequence of digits
- OPERATOR: is one of five operators +-*/=
- DELIM: bracket delimiters such as {}[]()
- COMMENT: all characters that follow a # character to the end of the source file/string or until the end of line '\n'character is reached

- Unrecognized tokens are considered to be syntax error

New Member Function

- Returns a vector of remaining tokens
- vector<Token> Lexer::tokenize()

Vector in C++

- vector<int> myvec;
- myvec.pushback(123);
- myvec.pushback(456);
- Access using myvec[0], myvec[1]
- myvec.front() // first element
- myvec.back() // last element
- myvec.insert(position)
- myvec.size()
- myvec.pop_back()
- …

Stacks and Applications

- WELL-FORMED EXPRESSIONS
- STACKS
- INFIX, POSTFIX

HTML file


```
<div id="navigation">
 <div class="inner">
  <div id="searcher">
   <form method="get" action="http://www.gnu.org/cgi-bin/estseek.cgi">
    <div><label class="netscape4" for="phrase">Search:</label>
    <input name="phrase" id="phrase" type="text" size="18" accesskey="s"
           value="Why GNU/Linux?" onfocus="this.value=''" />
    <input type="submit" value="Search" /></div><!-- unnamed label -->
   </form>
  </div><!-- /searcher -->
  <ul>
   <li id="tabPhilosophy"><a href=
                          "/philosophy/philosophy.html">Philosophy</a></li>
   <li id="tabLicenses"><a href="/licenses/licenses.html">Licenses</a></li>
   <li id="tabEducation"><a href="/education/education.html">Education</a></li>
   <li id="tabSoftware"><a href="/software/software.html">Downloads</a></li>
   <li id="tabDoc"><a href="/doc/doc.html">Documentation</a></li>
   <li id="tabHelp"><a href="/help/help.html">Help GNU</a></li>
   <li id="joinfsftab"><a
href="https://www.fsf.org/associate/support_freedom?referrer=4052">Join the 
FSF!</a></li>
  </ul>

 </div><!-- /inner -->
</div><!--  /navigation -->
```

Well-formed expressions

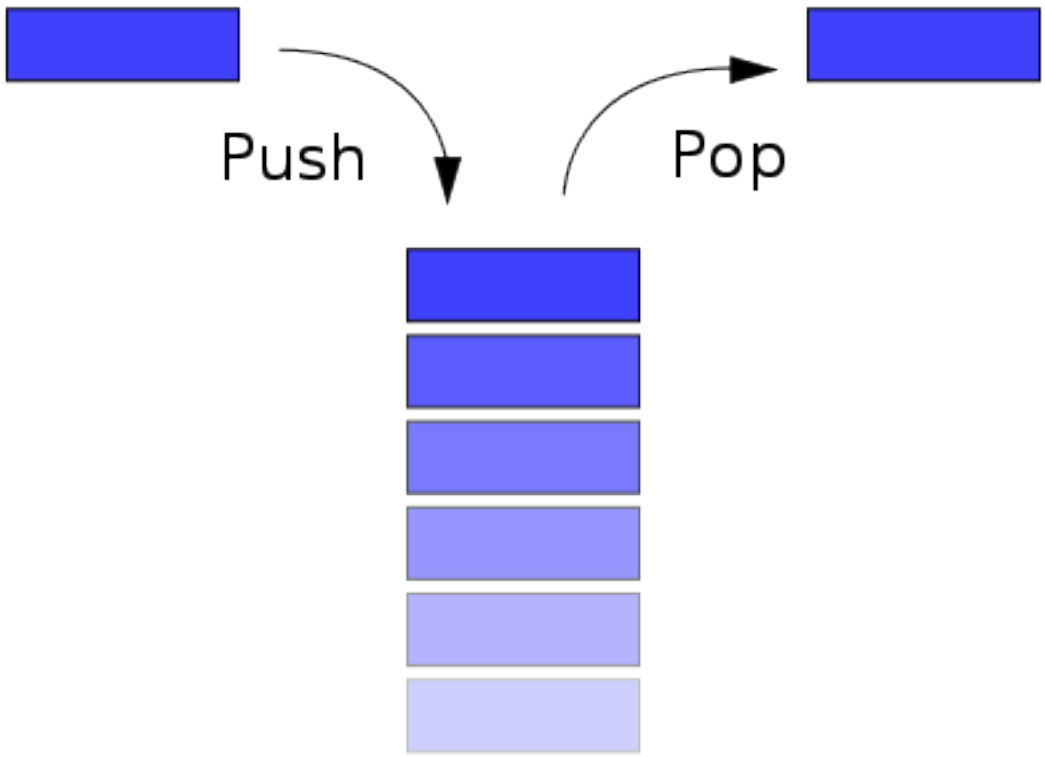
- Or “balanced expressions”:

- ([this is] { a number } 12345) # this is well-formed
- ([this is] { a number) 12345} # that is not
- {[(34+4)/5] + 7}/4 # this is well-formed
- {[(34+4)/5} + 7]/4 # that is not

Definition of WFE

- The empty sequence is well-formed.
- If A and B are well-formed, then the concatenation AB is well-formed
- If A is well-formed, then [A], {A}, and (A) are well-formed.

Stack



Algorithm for recognizing WFE

- Read the next delimiter token.
- If it is an open delimeter (i.e. [({), then we push it in the stack.
- If it is a close delimiter (i.e.])}), then we match it with a corresponding open delimiter in the stack ([with] and so on). If there is no match then the sequence is not well-formed. If there is a match, then we pop the stack and discard both the tokens.
- When there is no more token left and the stack is empty, then we have a well-formed sequence. Otherwise the sequence is not well-formed.

Infix vs Postfix Expressions

- 5+4*5/2-3 in postfix is written as 5 4 5 * 2 / + 3 -

- (5+4)*5/2-3 in postfix is written as 5 4 + 5 * 2 / 3 –

Postfix Expression Evaluation Algorithm

- Initialize an empty stack
- While (there is still a token to read)
  - read the token t
  - if t is an operand, push it onto the stack
  - if t is an operator,
    - pop two operands from the stack, compute the result (using t)
      // if there is division by zero, scream foul
    - push the result back onto the stack
      // if there is less than two operands, scream foul
- In the end, if there is one number in the stack, output it.
  // If there is more than one number in the stack, scream foul.