

Flex (Objectives)

- Given a regular expression the student will be able to write a flex specification of that regular expression.

Flex

- Flex is a scanner generator.
- It takes a regular expression, R , as input and emits C code to implement a DFA that recognizes $L(R)$.
- It uses the technology covered in computation theory

Flex File Layout

```
// Flex macro definitions
WS    [ \t\n]
%{
    // C++ definitions needed by the flex actions
    #include "nonsense.tab.h"
}%

%%
    // Flex Rules

{WS}      {;}
%%

    // User Functions
```

Flex Regular Expressions

- characters stand for themselves except
 `\,[.,^,-,?,*,+,|,(,),$,{},%,<,>`
 - put a `\` in front of those characters to denote the character itself
- classes of characters are denoted with `[]`
 - `[abc]` denotes 'a', 'b', or 'c'
 - `[a-zA-Z]` denotes the English alphabet
 - `[0-9]` denotes any digit
 - `[^a]` denotes any character but 'a'
- repetition is denoted with `*` and `+`
 - `1*` denotes 0 or more 1's
 - `1+` denotes 1 or more 1's

Flex Regular Expressions

- grouping is denoted with ()
 - $(123)^*$ denotes 0 or more "123"s
- alternation is done with |
 - $(12)|3^*$ denotes "12" or 0 or more 3's
- the wildcard character is .
 - it denotes anything but newline
- newline is '\n'
- tab is '\t'

Excerpts from a Flex File

```
WS  [ \t\n]
%{
#include "void.tab.h"
}%
%%
WHILE          {return O_WHILE;}
[0-9]+         {return O_INT;}
{WS}          {;}
%%
```

Flex Variables

- Variables that hold important information that can be accessed by the parser.
 - `yytext` - contains the lexeme
 - `yytext` - length of `yytext`
 - `yylex()` - function called by parser generated by bison; returns a token
 - this is called automatically by the parser. You do not need to invoke this function.

Flex Notes

- flex matches the longest string that matches one of the possible regular expressions
- Putting the following rule at the end, will catch all characters not specified.

. {ERROR(yytext);}