# ceng444 bison recitation

2023/24 spring

#### outline

- what is bison?
- structure of a .y file
- default actions and conflict resolution
- bison and flex with c++
- working with Eclipse IDE
- how to have bison on your system?
- recommended readings

### what is bison?

- a parser generation tool for syntax analysis
- works with c and c++
- open source (yay!)

#### what is bison?

- a parser generation tool for syntax analysis
- works with c and c++
- open source (yay!)
- named after yacc (yet another compiler compiler), which bison is a replacement of



p1, p2, and illustrations

#### what is bison?

- a parser generation tool for syntax analysis
- works with c and c++
- open source (yay!)
- named after yacc (yet another compiler compiler), which bison is a replacement of



### structure of a .y file

#### %{prologue%} and declarations

- code to be directly copied (%code or %{%})
- %token declarations to be used in rules
- other directives [1][2]

#### grammar rules

definition of the grammar to be parsed

#### epilogue

supporting c/c++ code

```
declarations>
%%

<grammar rules>
%%
<epilogue>
```

.y file

```
#include <ctype.h>
                                                                                            declarations
code to be directly
                                   %token DIGIT
  copied between
                                   %%
special parentheses
                                          : expr '\n' { printf("%d\n", $1); }
                                   line
                                          : expr '+' term { $$ = $1 + $3; }
                                   expr
                                           term
                                                                                            grammar rules
  %token declaration
                                          : term '*' factor { $$ = $1 * $3; }
                                   term
                                           factor
                                   factor : '(' expr ')' { $$ = $2; }
                                           DIGIT
                                   %%
                                   yylex() {
                                       int c;
                                       c = getchar();
                                                                                           epilogue
                                       if (isdigit(c)) {
         yylex()
                                          yylval = c-'0';
     lexer routine to
                                          return DIGIT;
    tokenize the input
                                       return c;
                                   Figure 4.58: Yacc specification of a simple desk calculator
                                                 (from Dragon Book)
```

## structure of a .y file - tokens

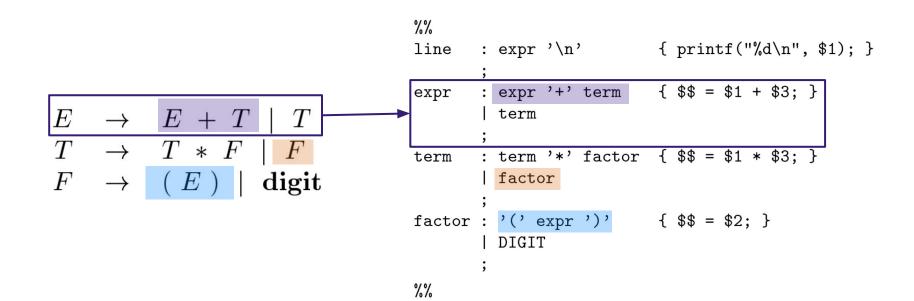
- %token keyword is used to declare tokens
- unquoted strings of letters and digits not declared to be %tokens are assumed to be nonterminals!
- single quoted strings are taken as terminals

```
%%
                                           : expr '\n'
                                    line
                                                             { printf("%d\n", $1); }
                                            expr '+' term
                                                             \{ \$\$ = \$1 + \$3; \}
%{
                                    expr
                                             term
#include <ctype.h>
%}
                                            term '*' factor { $$ = $1 * $3; }
                                    term
                                            factor
%token DIGIT
                                    factor : '(' expr ')' { $$ = $2; }
%%
                                            DIGIT
                                    %%
                               (from Dragon Book)
```

# structure of a .y file - grammar rules

a formal rule of the form:

is translated as:



# structure of a .y file - grammar rules

- \$\$: constructed value of the group, i.e. value of the <head>
- \$i: value of the i<sup>th</sup> component of a rule
- \$name or \$[name]: value of the component named name

(from Bison manual)

#### default actions and conflict resolution

- if no semantic action is specified for a rule, bison defaults to: \$\$ = \$1 [1]
- in case of parsing action conflicts, bison follows two rules:
  - for reduce/reduce conflicts, rule that appears first in the grammar is chosen [2]
  - for shift/reduce conflicts, shift is chosen [3]
- to override default conflict resolution for operation associativity and precedence, directives such as %right, %left, %nonassoc, and %precedence can be used [4]
- %start can be used to override the default starting symbol, which is the first nonterminal that appears in the rules [5]

## structure of a .y file – other directives

- %code and %code qualifier [1]
- %parse-param [2]
- %define, %language, and others [3][4]

#### bison & flex with c++

let's check the shared recitation example: sample02 compiling and running from terminal:

- flex -+ sample02.1
- bison -d sample02.y
- g++ -std=c++11 -o driver
   MyParser.cpp MyFlexLexer.cpp sample02.cpp
   sample02.tab.cc lex.yy.cc
- ./driver sample02.txt

# working with Eclipse IDE

- generate needed .1 and .tab.\* files using flex and bison from terminal
- create, build, and run project as described in the first recitation

# how to have bison on your system?

on a linux machine:

<your favourite package manager> install bison

- on macos:
  - existing version is old, use brew to install again for better compatibility brew tells you what to do to be able to use the newer version of bison
- on windows:
  - works when integrated in an IDE, tutorials are available online
  - → WSL may work, but using a virtual machine is a better idea

department labs have bison installed, you can always use them.

## recommended readings

- bison manual → best resource you can have!
  - o man bison in your terminal or online
  - many important sections are referenced in these slides (:
- chapter 4.9 from the *Dragon Book*
  - (+) easy-to-follow tutorial/explanation
  - (-) on c instead of c++
  - (-) lacks some features
- official example of bison + flex for c++

# thanks!