Lab 1 (part 2) - 2019CS10399

1 `c.y` File

This is a bison file that is used to define the CFG used to define the grammar for the tokens that were lexed by the lexer.

2 Generating `c.tab.cpp` from `c.y`

`bison` command is used to generate `c.tab.cpp` file from `c.y` file. This also generates the `c.tab.hpp` file which is also required by `c.lex.cpp`.

3 Understanding `c.tab.cpp`

`c.tab.cpp` file is an auto-generated file based on the contents in the `c.y` file. The main parser function that does the generation of the parse tree is `yyparse()`.

3.1 Working of `yyparse`

1. On call to the function, multiple variables are initialised, some of which are `yyss`, `yyssa` (these two are used to store the base and the top of the stack), `yyvs`, `yyvsp` (these two are used to store the base and top of the semantic value stack), `yyresult` (return value of `yyparse`), `yytoken` (lookahead symbol)
2. The following `goto` labels are defined in `yyparse`:
   a. `yynewstate`: updates stack pointer after value is pushed in the stack
   b. `yysetstate`: updates the top of the stack to the current state (`yystate`)
   c. `yybackup`: performs processiong based on the current state and lookahead token
   d. `yydefault`: default action for current state
   e. `yyreduce`: perform reduction
   f. `yyerrlab`: label to go to on detecting error
   g. `yyerrorlab`: error raised by `YYERROR`
   h. `yyerrlab1`: common code for both syntax error and `YYERROR`
   i. `yyacceptlab`: on arriving at `YYACCEPT`
   j. `yyabortlab: on arriving at`YYABORT‘
   k. `yyexhaustedlab`: if memory is exceeded `YYNOMEM`

l. `yyreturnlab`: parsing is completed, clean up and return

3.2 Explanation of each `goto` State in Detail

3.2.1 `yynewstate`

This simply increments the top of the stack since a state has been pushed to the stack and then fall through to `yysetstate`

3.2.2 `yysetstate`

This sets the top of the stack to `yystate` and checks for overflow (stack being filled). After this, it checks if the state is the final state, then it calls `YYACCEPT`, otherwise jumps to `yybackup`.

3.2.3 `yybackup`

It first attempts to find the next action without using the lookahead token, it consults `yypact` table for this. If the proposed action is the default value, we jump to `yydefault`. Else it reads the lookahead token to find the action to perform. Here, it calls `yylex` if all the previously lexed characters have been consumed. It handles EOF and errors in lexing. It then finds the action to take based on the lookahead character read. If if finds a valid action, it reduces, else it shifts and jumps to `yynewstate`.

3.2.4 `yydefault`

Checks for the default rule to reduce the current state using a `yydefact` table (this value is stored in `yyn` variable). If the value is 0, then there is an error and we jump to `yyerrlab`, else we jump to `yyreduce`.

3.2.5 `yyreduce`

This reduces the top elements of the stack using the rule identified in `yyn` variable. It then makes a transition in the automaton based on the new top of the stack using tables `yytable` and `yydefgoto`. It then jumps to `yynewstate`.

3.2.6 `yyerrlab`

This checks the `yyerrstatus` variable to check error recovery status, so that the parsing is not completely stopped and the user can get more information about the parsing of their code. It then jumps to `yyerrlab1`.

3.2.7 `yyerrorlab`

If `YYERROR` was explicitly raised, simply pop the stack and jump to `yyerrlab1`

3.2.8 `yyerrlab1`

This pops the stack until a state is found that can shift on the error token. If we reach the bottom of the stack, we call `YYABORT`. Once we reach a state that can shift the error token, we jump to `yynewstate` to resume parsing from here.

3.2.9 `yyacceptlab`

On calling `YYACCEPT`, we jump to here, set `yyresult` to 0 and jump to `yyreturnlab`.

3.2.10 `yyabortlab`

On calling `YYABORT`, we jump to here, set `yyresult` to 1 and jump to `yyreturnlab`.

3.2.11 `yyexhaustedlab`

On calling `YYNOMEM`, we jump to here, set `yyresult` to 2 and jump to `yyreturnlab`.

3.2.12 `yyreturnlab`

This cleans and frees the stack and then returns `yyresult`, exiting from `yyparse`.