Compilers 2023 SE3355 2023

Lab 3: Parsing

Description

Use Bisonc++ to implement a parser for the Tiger language. Appendix A describes, among other things, the syntax of Tiger.

Bisonc++ is a general-purpose parser generator converting grammar descriptions for LALR(1) context-free grammars into C++ classes whose members can parse such grammars. So your task is to write the correct rules in *tiger.y* that describes a parser, which will generate a proper abstract syntax tree for every given tiger source file.

Related files for this lab are:

- **src/tiger/absyn/absync.*** The declarations and definitions of all abstract syntax classes
- **src/tiger/parse/tiger.y** The grammar file you must fill in.
- **src/tiger/parse/parser.*** The Parser class declaration.
- **src/tiger/errormsg/errormsg.*** The ErrorMsg class, useful for producing error messages with file names and line numbers.
- **src/tiger/symbol/symbol.*** The Symbol class and Symbol table.

You will only need to fill in the file *tiger.y*. And your grammar should have as few shift-reduce conflicts as possible, and no reduce-reduce conflicts.

Notice: Before you start this lab, you should carefully read the chapter 3 of the textbook. And if you have any question about this lab, feel free to contact teaching assistants.

Important Notes

1. Some tokens such as ID, STRING, and INT will have semantic values. We have already set their semantic values for you in the function *Parser::lex()* in *src/tiger/parse/parser.ih*.

2. Normally we will use *%token* directives to define tokens. However, when you want to define some precedence rules in directives like *%left*, you need to remove the corresponding token definition in *%token* directive because the precedence directives will also defining tokens, or there will be duplicate token definition error.

3. Different from the textbook, we add a new expression class called *VoidExp* to refer to the empty expression, which is just wrapped by parentheses.

4. We use std::list instead of traditional C-list from now on, you are supposed to learn some basic usage of std::list and use it to generate correct AST in this lab.

Environment

You will use the same code framework that you had set up when you worked on lab2 and use the code you written in lab2. What you need to do now is to fetch the latest update of the code framework. You may have to do some code merging jobs. If you have any difficulties in merging codes, you can ask TAs in our Wechat group.

```
shell% git feth upstream
shell% git checkout -b lab3 upstream/lab3
shell% git push -u origin
shell% git merge lab2

You may have to do some code merging jobs here.

After merging, you are supposed to push your update to your remote repo
shell% git add files
shell% git commit -m "[lab3] merge lab2"
```

```
shell% git push origin lab3
```

`Now you can check your pipeline on gitlab, if you successfully merge, you should see your previous labs passed.`

Grade Test

The lab environment contains a grading script named as grade.sh, you can use it to evaluate your code, and that's how we grade your code, too. If you pass all the tests, the script will print a successful hint, otherwise, it will output some error messages. You can execute the script with the following commands.

```
shell% make gradelab3
shell% ...
shell% [^_^]: Pass Lab3 #If you pass all the tests, you will see these messages.
shell% TOTAL SCORE: 100
```

Handin

The deadline of this lab is on Sunday 12:00 AT NOON, Oct 22, 2023, and if you miss the deadline, points will be deducted based on the number of days you are late!

After you have passed the grade test, you need first commit your modification, then push it to your remote repository on gitlab.You can use the following commands to finish this step.

```
shell% git add src/tiger/parse/tiger.y
shell% git commit -m "[lab3] finish lab3"
shell% git push origin lab3
```

In the end, you need to check your pipeline on gitlab. Make sure you see all the test passed, otherwise you may not get your score.

Go to Top // *Compilers* Home Page
Questions or comments regarding *Compilers* course? Send e-mail to the course Staffs or TAs.
Last updated: Sun Oct. 8 2023