CS323 Compiler Project Phase 1

Group: 12110529 CAO Zhezhen, 12110804 FANG Jiawei, 12110817 ZHANG Zhanwei.

Sorted in alphabetical order.

Test Platform

Name	Value
OS	Ubuntu 22.04.2 LTS on Windows 10 x86_64
Bison	bison (GNU Bison) 3.8.2
Flex	flex 2.6.4
libbison-dev	2:3.8.2+dfsg-1build1
gcc	gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Make	GNU Make 4.3. Built for x86_64-pc-linux-gnu

Compile

Before: `make clean`

Then: `make` or `make splc`.

Basic Feature List

All implemented.

Extended Feature List

Error Detection

- Hanging else
- Invalid function definition inside functions
- Various errors about missing parenthesis/square bracket:

  Run parser on `test/test_12110804_2.spl` for details on missing square brackets.
- Various errors about missing operands:

```
iskxcr@ISK-WKST:~/CS323-Compiler-Project$ bin/spl_parser test-ex/extra_test_3.spl
test-ex/extra_test_3.spl:4:15: error: too many decimal points or exponential indicators [A]
       4 |     float y = .3e-13e7;
         |               ^~~~~~~
test-ex/extra_test_3.spl:6:5: error: expected expression before '>' [B]
       6 |     >7;
         |     ^
```

- Errors about invalid constant form:

```
iskxcr@ISK-WKST:~/CS323-Compiler-Project$ bin/spl_parser test/test_5.spl
test/test_5.spl:3:13: warning: overflow in integer constant [WARNING]
       3 |     int x = 0x123456789;
         |             ^~~~~~~~~~
test/test_5.spl:5:13: warning: overflow in integer constant [WARNING]
       5 |     int z = 4294967308;
         |             ^~~~~~~~~~
test/test_5.spl:6:13: warning: overflow in integer constant [WARNING]
       6 |     int k = 2147483649;
         |             ^~~~~~~~~~
```

Other SPL Parser Features

- Optimized error/warning output

```
iskxcr@ISK-WKST:~/CS323-Compiler-Project$ bin/spl_parser tests/phase1/basic/test_1_r04.spl
tests/phase1/basic/test_1_r04.spl:9:16: error: missing closing parenthesis ')' [B]
       9 |int test_1_r04(
         |               ^
tests/phase1/basic/test_1_r04.spl:12:48: error: missing right parenthesis ')' [B]
      12 |   c = func_2(func_1(func_1(a+b*func_1(b)), 1.7);
         |                                               ^
iskxcr@ISK-WKST:~/CS323-Compiler-Project$ bin/spl_parser tests/phase1/basic/test_1_r03.spl
tests/phase1/basic/test_1_r03.spl:4:13: error: unknown lexeme [A]
       4 |   float i = $;
         |             ^
tests/phase1/basic/test_1_r03.spl:6:13: error: missing semicolon ';' [B]
       6 |     return 1
         |            ^
tests/phase1/basic/test_1_r03.spl:8:10: error: unknown lexeme [A]
       8 |   return @;
         |          ^
iskxcr@ISK-WKST:~/CS323-Compiler-Project$ bin/spl_parser tests/phase1/extended/test_6.spl
tests/phase1/extended/test_6.spl:2:5: error: hanging else is not allowed. [B]
       2 |     else
         |     ^~~~
iskxcr@ISK-WKST:~/CS323-Compiler-Project$ bin/spl_parser tests/phase1/extended/test_4.spl
tests/phase1/extended/test_4.spl:3:9: error: function definition not allowed here. [B]
       3 |     int test_p()
         |         ^~~~~~~~
tests/phase1/extended/test_4.spl:3:17: error: missing semicolon ';' [B]
       3 |     int test_p()
         |                 ^
iskxcr@ISK-WKST:~/CS323-Compiler-Project$ bin/spl_parser tests/phase1/bonus_test/extra_test_2.spl
tests/phase1/bonus_test/extra_test_2.spl:3:9: error: function definition not allowed here. [B]
       3 |     int foo();
         |         ^~~~~
iskxcr@ISK-WKST:~/CS323-Compiler-Project$
```

This may cause the `diff` utility not to work when doing batch verifying, as ASCII control sequences are used to color the output and they will not be recognized by `diff`.

The output parsing tree will not get colored.

This is done by using ASCII control sequences and reusable file descriptors. ***Some of the SPL grammar structure is modified, but the target language does not change.***

- `StmtList` *and* `DefList` *now* ***cannot*** *be empty.*

- Allowing unary operators: `+/-`, unary prefix/postfix operator: `++/--`

  Run parser on `test/test_5.spl` for details.

- Allowing `for` loop declaration

  - Allow any type of for loop combination:

    ```
    for ([optional definition]; [optional expression]; optional expression)
        stmt
    ```

  *This is done by decomposing the for loop into **loop body** and **statement**.*

  Run parser on `test_ex/test_6.spl` for details.

- Allowing the following floating-point declaration, given by `[0-9]*\.[0-9]+([eE][-+]?[0-9]+)?`:

  ```
  float y = .3e-13;
  ```

  Run parser on `test-ex/test_3.spl` for details.

- Allowing single-line/cross-line comments

  ```
  int main()
  {
      int a = 0; // This is a single comment.
      int b = 0; // This is a cross-line comment \
                    as you can tell by changing the language server interpreting this
  comment to match the C language.
      int c = 233; // Hi! C
      /*
      This is a cross-line comment.
      */
      int d = c;
      /* Hi!
      */

      */ // This is a hanging comment, which should not be recognized
  }
  ```

  ```
  iskxcr@ISK-WKST:~/CS323-Compiler-Project$ bin/spl_parser test-ex/extra_test_4.spl
  syntax error, unexpected RC at line 15
  test-ex/extra_test_4.spl:14:6: error: expected expression after '/' [B]
        14 |     */ // This is a hanging comment, which should not be recognized
           |      ^
  test-ex/extra_test_4.spl:14:5: error: expected expression before '*' [B]
        14 |     */ // This is a hanging comment, which should not be recognized
           |     ^
  syntax error, unexpected RC at line 15
  test-ex/extra_test_4.spl:15:2: error: missing semicolon ';' [B]
        15 |}
           | ^
  ```

  This is done by assigning `yylex` with specific states.

  Run parser on `test-ex/test_4.spl` for details.

- Allow single-line/cross-line strings:

  - Allowed escape characters: `\[abefnrtv\'"?]`

```
int main()
{
    char msg1 = "Hi Just want to test!";

    char msg2 = "Hi \
                    Just want to test!";

    char msg3 = "Hi"
                "Thanks!";
    printf("[%s]", msg);
    return 0;
}
```

This is done by assigning `yylex` *with specific states*.

Run parser on `test-ex/test_5.spl` for details.

There are other tests that does hybrid testing. You may check them out individually.