编译小组作业文档-第一次提交

高英杰 2022012068

查益 2022012107

叶隽希 2022012093

狄奕辰 2022012072

一、简介

本小组作业计划完成一个简单的,**从C语言到LLVM语言**的编译器、翻译器。其中作业分为两个阶段提交:第一阶段需要完成词法分析和语法分析部分,第二阶段需要提交完整作业,即实现完整的编译器翻译器。

本次提交及本文档为第一阶段提交和文档。

二、分工

查益:组长、语法分析、报告

高英杰:管理仓库、语法分析、词法分析

叶隽希:词法分析、提供C语言语法、提供测例

狄奕辰: 词法分析、C语言语法、调试并修改

三、代码说明

本作业提供了三个python文件:

• src/lexer.py : C语言词法分析器的实现

• src/parser.py : C语言语法分析器的实现

• src/syntax.py : C语言语法规则定义

1. 语法规则定义

在 syntax.py 文件中,使用BNF形式定义了C语言的简化文法规则如下:

```
1 rules = {
       "PROGRAM": {
 2
            "name": "PROGRAM",
 4
             "rules": [
 5
                 Ε
 6
                      "OUTER_STMT",
 7
                      "OUTER_SUB_PROGRAM"
 8
                 ],
9
                 Ε
10
                      "OUTER_STMT"
                 1
11
12
13
        },
14
        . . . . . .
15
    }
```

其中,规则保存在rules字典当中,每个键是一个非终结符,值是一个字典,包含name和rules (name 是非终结符的名称,存在空名称,用于简化语法树的生成; rules是一个二维列表,每个元素是一个产生式,产生式是一个列表,每个元素是一个非终结符或终结符)。

2. 词法分析

在lexer.py中实现了词法分析功能,可以将原始的C代码转换为一系列的Token。其中,定义了Token类和Lexer类如下:

- Token类用于表示词法分析后的基本元素(Token),每个Token包含类型,值,行号和列号。
- Lexer类负责将输入的C语言源代码进行词法分析,转换为Token序列。包含C语言的关键字集合,C 语言的标点和操作符集合,已经生成的Tokens列表,当前处理位置的行号和列号,以及当前待分析的源代码字符串。

词法分析流程如下:

- 1. 初始化Lexer对象,清空Token列表,设置行号和列号初始值。
- 2. 设置待分析的源代码字符串 _code 。
- 3. 循环遍历源代码字符串,根据当前字符,调用相应的处理方法。
 - 。 跳过空白字符和注释。
 - 识别并处理关键字、标识符、数字、字符串、字符常量、操作符和标点符号。
- 4. 将识别到的Token添加到Token列表中,记录其类型、值、行号和列号。
- 5. 如果遇到无法识别的字符或非法的Token, 抛出异常, 提示错误位置和原因。

3. 语法分析

Parser类实现了一个LR(1)语法分析器,用于将Token序列解析为语法树。

与编译小作业内容类似:首先构建LR(1)分析表,初始化状态栈、符号栈、语法树栈。接着读取Token序列,在每一步中读取当前状态和下一个Token的类型,并根据Action表,进行移进、规约、接收、错误的操作。

四、运行实例

程序 parser.py 接受的参数形式如下:

```
usage: parser.py [-h] (-t | -i INPUT) [-o OUTPUT]

-t, --test 使用样例程序测试语法分析器

-i INPUT, --input INPUT

输入文件路径

-o OUTPUT, --output OUTPUT

输出文件路径
```

在 src 目录下运行测试样例并获取结果:

```
1 | python parser.py -t
```

以下是两组输入和输出(输出文件太长,仅截取前几十行部分内容。输入详细见 in/<code>.c ,输出详细见 out/<code>.j son):

```
doubleBuddleSort.c
 3
        双端冒泡排序
    */
4
 5
    #include <stdio.h>
 6
7
    int main() {
8
        int arr[65535];
9
        int n = 0;
10
        char c;
11
        while (1) {
12
            scanf("%d", &arr[n]);
13
            n = n + 1;
14
            c = getchar();
15
            if (c != ',') {
                break;
16
17
            }
18
        }
19
20
        int 1 = 0;
21
        int r = n - 1;
22
        int p = 0;
23
        while (1 < r) {
            p = 1;
24
25
            while (p < r) {
26
                if (arr[p] > arr[p + 1]) {
27
                     int temp = arr[p];
                     arr[p] = arr[p + 1];
28
29
                     arr[p + 1] = temp;
30
                }
31
                p = p + 1;
32
            }
33
            r = r - 1;
34
35
            p = r;
36
            while (p > 1) {
37
                if (arr[p] < arr[p - 1]) {
38
                     int temp = arr[p];
39
                     arr[p] = arr[p - 1];
40
                     arr[p - 1] = temp;
41
                }
42
                p = p - 1;
43
            }
44
            1 = 1 + 1;
45
        }
46
47
        p = 0;
        while (p < n) {
48
49
            printf("%d", arr[p]);
            if (p != n - 1) {
50
51
                printf(",");
52
53
            p = p + 1;
54
        }
55
        return 0;
56
    }
```

```
1
        "type": "PROGRAM",
 2
        "children": [
 3
 4
            {
 5
                 "type": "OUTER_STMT",
                 "children": [
 6
 7
                     {
                         "type": "HEADER_STMT",
 8
9
                         "children": [
10
                             {
11
                                  "type": "include",
                                  "value": "#include",
12
                                  "line": 5,
13
                                  "column": 0
14
15
                             },
16
                             {
                                  "type": "1t",
17
                                  "value": "<",
18
19
                                  "line": 5,
20
                                  "column": 9
21
                             },
22
                             {
                                  "type": "header",
23
24
                                  "value": "stdio.h",
25
                                  "line": 5,
                                  "column": 10
26
27
                             },
28
29
                                  "type": "gt",
                                  "value": ">",
30
                                  "line": 5,
31
32
                                  "column": 17
33
                             }
                         ]
34
35
                     }
                 ]
36
37
            },
38
            . . .
39
        ]
40 }
```

```
2
        palindrome.c
3
       回文串检测
4
5
    #include <stdio.h>
 6
7
    int main() {
        char str[65535];
8
9
10
        gets(str);
        int len = 0;
11
12
        while (str[len] != '\0' && str[len] != '\n' && str[len] != '\r') {
13
```

```
14
      len = len + 1;
15
        }
16
17
        int p = 0;
18
        int flag = 1;
19
        while (p < len / 2 \&\& flag == 1) {
20
            if (str[p] != str[len - p - 1]) {
21
                flag = 0;
22
            }
23
            p = p + 1;
24
        }
25
26
        if (flag == 1) {
27
            printf("True");
28
        }
29
        else {
30
            printf("False");
31
        }
32
33
        return 0;
34
   }
```

```
1
2
        "type": "PROGRAM",
 3
        "children": [
 4
            {
                 "type": "OUTER_STMT",
 5
 6
                 "children": [
 7
                     {
 8
                          "type": "HEADER_STMT",
9
                          "children": [
10
                             {
                                  "type": "include",
11
12
                                  "value": "#include",
                                  "line": 5,
13
14
                                  "column": 0
15
                             },
16
                             {
                                  "type": "1t",
17
                                  "value": "<",
18
                                  "line": 5,
19
                                  "column": 9
20
21
                             },
22
                             {
                                  "type": "header",
23
24
                                  "value": "stdio.h",
                                  "line": 5,
25
                                  "column": 10
26
27
                             },
28
                             {
                                  "type": "gt",
29
                                  "value": ">",
30
31
                                  "line": 5,
                                  "column": 17
32
33
                             }
```

```
34 ]
35 }
36 ]
37 },
38 ...
39 ]
40 }
```