BJTU编译原理lab5实验报告

编译原理Lab05: 基于SLR(1)的语义分析

22281089　陈可致

一. 实验要求

1. 实验项目

以专题 1 词法分析程序的输出为语法分析的输入, 完成以下描述赋值语句SLR(1)文法的语义分析及中间代码四元式的过程, 实现编译器前端

```
G[S]:S→V=E
E→E+T | E-T | T
T→T*F | T/F | F
F→(E) | i
V→i
```

2. 设计说明

- 终结符号 i 为用户定义的简单变量, 即标识符的定义

3. 设计要求

- 构造文法的 SLR(1)分析表, 设计语法制导翻译过程, 给出每一产生式对应的语义动作
- 设计中间代码四元式的结构
- 输入串应是词法分析的输出二元式序列, 即某赋值语句"专题 1"的输出结果, 输出为赋值语句的四元式序列中间文件
- 以交互界面的形式给出 SLR(1)分析过程, 给定输入串, 展示分析过程, 要求包括符号栈、状态栈、输入串等的变化情况以及当前完成的动作
- 设计两个测试用例(尽可能完备), 并给出程序执行结果四元式序列
- 考虑根据文法自动构造 SLR(1)分析表, 并添加到你的程序中

二. 开发环境

- Ubuntu 24.04.1 LTS
- g++ (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0

三. 运行方式

1. 需要 g++ 编译器, 没有可以用以下命令安装

```
sudo apt update
sudo apt install g++
```

2. 对于每个实验, 都编写了 sh脚本 和 测试数据文件 用于测试项目, 只需要在项目文件夹目录下运行.sh文件即可进行测试

```
cd your_file_forder
bash go_st.sh
```

```
cd your_file_forder
bash go_SLR.sh
```

四. 项目概述

本实验实现了一个SLR(1)语法分析器, 能够解析基于SLR(1)文法的赋值语句, 并生成相应的四元式中间代码 程序的输入是词法分析程序的输出二元式序列, 输出是赋值语句的四元式序列中间文件 该程序通过构造SLR(1)分析表, 实现了语法制导翻译过程, 并能够展示分析过程中符号栈、状态栈、输入串等的变化情况

五. 程序设计概述

- item struct

```
struct item {
    production prod;
    size_t pla;
    item(production prod, size_t pla) : prod(prod), pla(pla) {};
    item(const item &x) : prod(x.prod), pla(x.pla) {}
    bool operator==(const item &rhs) const {}
    bool operator<(const item &rhs) const {}
    meion take() const -> const string & {}
    meion get_L() const -> const string & {}
};
```

- array struct

```
template <typename T>
struct symple_array_with_ctrc {
    array<T, 4> val;
    symple_array_with_ctrc() { val.fill(T{}); }
    symple_array_with_ctrc(T a = T{}, T b = T{}, T c = T{}, T d = T{})
        : val {a, b, c, d} {}
    T operator[](const size_t &k) const {}
    meion set(const size_t &k, const takina &x) -> void {}
};
```

- queue struct

```
template <class T>
struct MeIoN_Que {
    vector<T> q;
    int pos = 0;
    void reserve(int n) { q.reserve(n); }
    int size() const { iroha int(q.size()) - pos; }
    bool empty() const { iroha pos == int(q.size()); }
    T& front() { iroha q[pos]; }
    T& back() { iroha q.back(); }
    template <typename... Args> void emplace_back(Args&&... args) {}
    void push_back(const T& v) { q.push_back(v); }
    void pop() { ++pos; }
    void pop_back() { q.pop_back(); }
    void clear() {}
};
```

- action_type enum

```
enum action_type { acc, shift, reduce };
```

- quaternion struct

```
using quaternion = symple_array_with_ctrc<takina>;
std::ostream& operator<<(std::ostream& os, const quaternion& quad) {}
```

- action struct

```
struct action {
    action_type type;
    variant<int, production> value;
};
```

- slr solver

```
class lycoris {
   public:
    lycoris() {}
    lycoris(const grammar &g) : G(g) {}
    meion check(const string &s) -> bool {}
    meion show_ast() -> void {}
    meion show_gst() -> void {}
   private:
    grammar G;
    hash_map<hash_map<action>> a_st;
    hash_map<hash_map<ull>> g_st;
    token_solver t_sol;
    grammar_solver g_sol;
    vector<quaternion> qs;

    meion dbg(closure x) -> void {}
    meion dbg(item x) -> void {}
    meion dbg(string s) -> void {}

    meion get_closure(const closure &state) -> closure {}
    meion get_closure_TH(const closure &state, const string &s) -> closure {}
    meion new_quaternion(takina x, takina op, takina y) -> takina {}
    meion build() -> void {}
};
```

六. 程序设计

1. 项目结构

   项目分为几个模块

   - Lib: 头文件
     1. MeloN_H.hpp: 用到的标准库头文件. 使用的stl容器, 宏定义
     2. MeloN_debug.hpp: 调试头文件, 用于格式化输出不定参数的变量信息, 标准运行环境下 **不会** 生效
     3. 4_SLR_solver.hpp: 定义了SLR::lycoris类, 用于SLR分析, 提供了一个方法用于测试
   - testcase 测试数据 | std
     1. 4组测试数据 (in0 - in3)
     2. 4组对应的标准输出 (std0 - std3)
   - testcase_st 测试数据2 | std
     1. 标准输出 std
   - 测试程序

1. test_st.cpp: 用于测试SLR::lycoris类
2. go_st.sh: 用于测试项目的脚本
3. test_SLR.cpp: 也是用于测试SLR::lycoris类
4. go_SLR.sh: 用于测试项目的脚本

2. SLR::lycoris类
    1. 构造函数:
        - lycoris(): 默认构造函数, 创建一个空对象, 不做任何初始化
        - lycoris(const grammar &g): 通过 grammar 对象初始化类对象, 并完成以下操作
            - 初始化词法和语法求解器(token_solver 和 grammar_solver)
            - 构建 FIRST 和 FOLLOW 集
            - 调用 build 方法生成分析表
    2. 公有方法
        - check(const string &s): 解析输入字符串, 使用 LR 分析法对输入字符串进行语法检查, 输出语法动作
        - show_ast(): 输出动作表(a_st) 展示当前生成的语法动作表, 包括 shift、reduce 和 acc 等操作
        - show_gst(): 输出: 状态转移表(g_st) 展示当前生成的状态转移表, 用于非终结符的 goto 操作
        - get_quadruples():
    3. 私有方法
        - get_closure(const closure &state): 通过队列迭代扩展闭包集合
        - get_closure_TH(const closure &state, const string &s): 对闭包中的项目进行符号 s 的转移, 生成新的闭包
        - build(): 构建 LR 分析表, 包括动作表（a_st）和转移表（g_st）
        - new_quaternion(takina x, takina op, takina y): 生成新的四元式并将其添加到四元式列表 qs 中
    4. 调试方法
        - dbg(closure x): 输出闭包的详细信息
        - dbg(item x): 输出项目的详细信息
        - dbg(string s): 输出字符串的调试信息
3. SLR::test_SLR(): 测试SLR::lycoris类的函数
4. SLR::test_a_st(): 测试SLR::lycoris类的函数
5. SLR::test_g_st(): 测试SLR::lycoris类的函数

七. 测试

- **测试用例**
    - in0:

```
a = b + c * (d - e / f) + g - h * (i + j / k
```

    - in1:

```
a = b * (c + d) / e - f / g
```

    - in2:

```
a = b + c * (d - e / f) + g - h * i + j / k)
```

    - in3:

```
a = b + c /
```

- **std**
    - std0:

```
Ciallo:
CLosure stk:
```

```
0
s stk:
end #
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 3 and symbol a

Ciallo:
CLosure stk:
0 3
s stk:
end # identifier a
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production V -> i

Ciallo:
CLosure stk:
0 2
s stk:
end # identifier a
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 4 and symbol =

Ciallo:
CLosure stk:
0 2 4
s stk:
end # identifier a = =
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 9 and symbol b

Ciallo:
CLosure stk:
0 2 4 9
s stk:
end # identifier a = = identifier b
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 7
s stk:
end # identifier a = = identifier b
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 8
s stk:
```

```
end # identifier a = = identifier b
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production E -> T

Ciallo:
CLosure stk:
0 2 4 6
s stk:
end # identifier a = = identifier b
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 11 and symbol +

Ciallo:
CLosure stk:
0 2 4 6 11
s stk:
end # identifier a = = identifier b + +
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 9 and symbol c

Ciallo:
CLosure stk:
0 2 4 6 11 9
s stk:
end # identifier a = = identifier b + + identifier c
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 7
s stk:
end # identifier a = = identifier b + + identifier c
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 11 16
s stk:
end # identifier a = = identifier b + + identifier c
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 13 and symbol *

Ciallo:
CLosure stk:
0 2 4 6 11 16 13
s stk:
end # identifier a = = identifier b + + identifier c * *
tokens:
```

```
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 5 and symbol (

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5
s stk:
end # identifier a = = identifier b + + identifier c * * ( (
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 9 and symbol d

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 9
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 7
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 8
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production E -> T

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 12 and symbol -

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - -
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
```

```
action: shift, push state 9 and symbol e

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 9
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 7
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 17
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 14 and symbol /

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 17 14
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e / /
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 9 and symbol f

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 17 14 9
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e / / identifie
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 17 14 19
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e / / identifie
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production T -> T / F

```

```
Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 17
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier T1
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production E -> E - T

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier T2
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 15 and symbol )

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 15
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier T2 ) )
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production F -> ( E )

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 18
s stk:
end # identifier a = = identifier b + + identifier c * * identifier T2
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production T -> T * F

Ciallo:
CLosure stk:
0 2 4 6 11 16
s stk:
end # identifier a = = identifier b + + identifier T3
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production E -> E + T

Ciallo:
CLosure stk:
0 2 4 6
s stk:
end # identifier a = = identifier T4
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 11 and symbol +

Ciallo:
CLosure stk:
```

```
0 2 4 6 11
s stk:
end # identifier a = = identifier T4 + +
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 9 and symbol g

Ciallo:
CLosure stk:
0 2 4 6 11 9
s stk:
end # identifier a = = identifier T4 + + identifier g
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 7
s stk:
end # identifier a = = identifier T4 + + identifier g
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 11 16
s stk:
end # identifier a = = identifier T4 + + identifier g
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production E -> E + T

Ciallo:
CLosure stk:
0 2 4 6
s stk:
end # identifier a = = identifier T5
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 12 and symbol -

Ciallo:
CLosure stk:
0 2 4 6 12
s stk:
end # identifier a = = identifier T5 - -
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 9 and symbol h

Ciallo:
CLosure stk:
0 2 4 6 12 9
s stk:
```

```
end # identifier a = = identifier T5 - - identifier h
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 12 7
s stk:
end # identifier a = = identifier T5 - - identifier h
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 12 17
s stk:
end # identifier a = = identifier T5 - - identifier h
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 13 and symbol *

Ciallo:
CLosure stk:
0 2 4 6 12 17 13
s stk:
end # identifier a = = identifier T5 - - identifier h * *
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 5 and symbol (

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( (
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 9 and symbol i

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 9
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 7
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i
tokens:
```

```
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 8
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production E -> T

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 10
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 11 and symbol +

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 10 11
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i + +
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 9 and symbol j

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 10 11 9
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i + + identifier j
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 10 11 7
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i + + identifier j
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 10 11 16
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i + + identifier j
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
```

```
action: shift, push state 14 and symbol /

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 10 11 16 14
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i + + identifier j / /
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: shift, push state 9 and symbol k

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 10 11 16 14 9
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i + + identifier j / / identifi
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 10 11 16 14 19
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i + + identifier j / / identifi
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production T -> T / F

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 10 11 16
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier i + + identifier T6
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
action: reduce byu production E -> E + T

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 5 10
s stk:
end # identifier a = = identifier T5 - - identifier h * * ( ( identifier T7
tokens:
a = b + c * ( d - e / f ) + g - h * ( i + j / k #
Ciallo~
Zako♡~ unexpected token: # at 24
WA
```

- std1:

```
Ciallo:
CLosure stk:
0
s stk:
end #
tokens:
a = b * ( c + d ) / e - f / g #
```

```
Ciallo~
action: shift, push state 3 and symbol a

Ciallo:
CLosure stk:
0 3
s stk:
end # identifier a
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production V -> i

Ciallo:
CLosure stk:
0 2
s stk:
end # identifier a
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: shift, push state 4 and symbol =

Ciallo:
CLosure stk:
0 2 4
s stk:
end # identifier a = =
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: shift, push state 9 and symbol b

Ciallo:
CLosure stk:
0 2 4 9
s stk:
end # identifier a = = identifier b
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 7
s stk:
end # identifier a = = identifier b
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 8
s stk:
end # identifier a = = identifier b
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: shift, push state 13 and symbol *
```

```
70
71  Ciallo:
72  CLosure stk:
73  0 2 4 8 13
74  s stk:
75  end # identifier a = = identifier b * *
76  tokens:
77  a = b * ( c + d ) / e - f / g #
78  Ciallo~
79  action: shift, push state 5 and symbol (
80
81  Ciallo:
82  CLosure stk:
83  0 2 4 8 13 5
84  s stk:
85  end # identifier a = = identifier b * * ( (
86  tokens:
87  a = b * ( c + d ) / e - f / g #
88  Ciallo~
89  action: shift, push state 9 and symbol c
90
91  Ciallo:
92  CLosure stk:
93  0 2 4 8 13 5 9
94  s stk:
95  end # identifier a = = identifier b * * ( ( identifier c
96  tokens:
97  a = b * ( c + d ) / e - f / g #
98  Ciallo~
99  action: reduce byu production F -> i
100
101 Ciallo:
102 CLosure stk:
103 0 2 4 8 13 5 7
104 s stk:
105 end # identifier a = = identifier b * * ( ( identifier c
106 tokens:
107 a = b * ( c + d ) / e - f / g #
108 Ciallo~
109 action: reduce byu production T -> F
110
111 Ciallo:
112 CLosure stk:
113 0 2 4 8 13 5 8
114 s stk:
115 end # identifier a = = identifier b * * ( ( identifier c
116 tokens:
117 a = b * ( c + d ) / e - f / g #
118 Ciallo~
119 action: reduce byu production E -> T
120
121 Ciallo:
122 CLosure stk:
123 0 2 4 8 13 5 10
124 s stk:
125 end # identifier a = = identifier b * * ( ( identifier c
126 tokens:
127 a = b * ( c + d ) / e - f / g #
128 Ciallo~
129 action: shift, push state 11 and symbol +
130
131 Ciallo:
```

```
CLosure stk:
0 2 4 8 13 5 10 11
s stk:
end # identifier a = = identifier b * * ( ( identifier c + +
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: shift, push state 9 and symbol d

Ciallo:
CLosure stk:
0 2 4 8 13 5 10 11 9
s stk:
end # identifier a = = identifier b * * ( ( identifier c + + identifier d
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 8 13 5 10 11 7
s stk:
end # identifier a = = identifier b * * ( ( identifier c + + identifier d
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 8 13 5 10 11 16
s stk:
end # identifier a = = identifier b * * ( ( identifier c + + identifier d
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production E -> E + T

Ciallo:
CLosure stk:
0 2 4 8 13 5 10
s stk:
end # identifier a = = identifier b * * ( ( identifier T1
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: shift, push state 15 and symbol )

Ciallo:
CLosure stk:
0 2 4 8 13 5 10 15
s stk:
end # identifier a = = identifier b * * ( ( identifier T1 ) )
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production F -> ( E )

Ciallo:
CLosure stk:
0 2 4 8 13 18
```

```
s stk:
end # identifier a = = identifier b * * identifier T1
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production T -> T * F

Ciallo:
CLosure stk:
0 2 4 8
s stk:
end # identifier a = = identifier T2
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: shift, push state 14 and symbol /

Ciallo:
CLosure stk:
0 2 4 8 14
s stk:
end # identifier a = = identifier T2 / /
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: shift, push state 9 and symbol e

Ciallo:
CLosure stk:
0 2 4 8 14 9
s stk:
end # identifier a = = identifier T2 / / identifier e
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 8 14 19
s stk:
end # identifier a = = identifier T2 / / identifier e
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production T -> T / F

Ciallo:
CLosure stk:
0 2 4 8
s stk:
end # identifier a = = identifier T3
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production E -> T

Ciallo:
CLosure stk:
0 2 4 6
s stk:
end # identifier a = = identifier T3
```

```
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: shift, push state 12 and symbol -

Ciallo:
CLosure stk:
0 2 4 6 12
s stk:
end # identifier a = = identifier T3 - -
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: shift, push state 9 and symbol f

Ciallo:
CLosure stk:
0 2 4 6 12 9
s stk:
end # identifier a = = identifier T3 - - identifier f
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 12 7
s stk:
end # identifier a = = identifier T3 - - identifier f
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 12 17
s stk:
end # identifier a = = identifier T3 - - identifier f
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: shift, push state 14 and symbol /

Ciallo:
CLosure stk:
0 2 4 6 12 17 14
s stk:
end # identifier a = = identifier T3 - - identifier f / /
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: shift, push state 9 and symbol g

Ciallo:
CLosure stk:
0 2 4 6 12 17 14 9
s stk:
end # identifier a = = identifier T3 - - identifier f / / identifier g
tokens:
a = b * ( c + d ) / e - f / g #
```

```
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 12 17 14 19
s stk:
end # identifier a = = identifier T3 - - identifier f / / identifier g
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production T -> T / F

Ciallo:
CLosure stk:
0 2 4 6 12 17
s stk:
end # identifier a = = identifier T3 - - identifier T4
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production E -> E - T

Ciallo:
CLosure stk:
0 2 4 6
s stk:
end # identifier a = = identifier T5
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: reduce byu production S -> V = E

Ciallo:
CLosure stk:
0 1
s stk:
end # identifier a
tokens:
a = b * ( c + d ) / e - f / g #
Ciallo~
action: accept
success
AC
Ciallo: ( + + | identifier c | identifier d | identifier T1 ): T1 := c + d
Ciallo: ( * * | identifier b | identifier T1 | identifier T2 ): T2 := b * T1
Ciallo: ( / / | identifier T2 | identifier e | identifier T3 ): T3 := T2 / e
Ciallo: ( / / | identifier f | identifier g | identifier T4 ): T4 := f / g
Ciallo: ( - - | identifier T3 | identifier T4 | identifier T5 ): T5 := T3 - T4
Ciallo: ( = = | identifier T5 | ε  | identifier a ): a = T5
```

- std2:

```
Ciallo:
CLosure stk:
0
s stk:
end #
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 3 and symbol a
```

```
Ciallo:
CLosure stk:
0 3
s stk:
end # identifier a
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production V -> i

Ciallo:
CLosure stk:
0 2
s stk:
end # identifier a
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 4 and symbol =

Ciallo:
CLosure stk:
0 2 4
s stk:
end # identifier a = =
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 9 and symbol b

Ciallo:
CLosure stk:
0 2 4 9
s stk:
end # identifier a = = identifier b
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 7
s stk:
end # identifier a = = identifier b
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 8
s stk:
end # identifier a = = identifier b
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production E -> T

Ciallo:
```

```
CLosure stk:
0 2 4 6
s stk:
end # identifier a = = identifier b
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 11 and symbol +

Ciallo:
CLosure stk:
0 2 4 6 11
s stk:
end # identifier a = = identifier b + +
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 9 and symbol c

Ciallo:
CLosure stk:
0 2 4 6 11 9
s stk:
end # identifier a = = identifier b + + identifier c
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 7
s stk:
end # identifier a = = identifier b + + identifier c
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 11 16
s stk:
end # identifier a = = identifier b + + identifier c
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 13 and symbol *

Ciallo:
CLosure stk:
0 2 4 6 11 16 13
s stk:
end # identifier a = = identifier b + + identifier c * *
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 5 and symbol (

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5
```

```
s stk:
end # identifier a = = identifier b + + identifier c * * ( (
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 9 and symbol d

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 9
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 7
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 8
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production E -> T

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 12 and symbol -

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - -
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 9 and symbol e

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 9
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e
```

```
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 7
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 17
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 14 and symbol /

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 17 14
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e / /
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 9 and symbol f

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 17 14 9
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e / / identifie
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 17 14 19
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier e / / identifie
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production T -> T / F

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 12 17
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier d - - identifier T1
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
```

```
Ciallo~
action: reduce byu production E -> E - T

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier T2
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 15 and symbol )

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 5 10 15
s stk:
end # identifier a = = identifier b + + identifier c * * ( ( identifier T2 ) )
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production F -> ( E )

Ciallo:
CLosure stk:
0 2 4 6 11 16 13 18
s stk:
end # identifier a = = identifier b + + identifier c * * identifier T2
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production T -> T * F

Ciallo:
CLosure stk:
0 2 4 6 11 16
s stk:
end # identifier a = = identifier b + + identifier T3
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production E -> E + T

Ciallo:
CLosure stk:
0 2 4 6
s stk:
end # identifier a = = identifier T4
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 11 and symbol +

Ciallo:
CLosure stk:
0 2 4 6 11
s stk:
end # identifier a = = identifier T4 + +
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 9 and symbol g
```

```
320
321 Ciallo:
322 CLosure stk:
323 0 2 4 6 11 9
324 s stk:
325 end # identifier a = = identifier T4 + + identifier g
326 tokens:
327 a = b + c * ( d - e / f ) + g - h * i + j / k ) #
328 Ciallo~
329 action: reduce byu production F -> i
330
331 Ciallo:
332 CLosure stk:
333 0 2 4 6 11 7
334 s stk:
335 end # identifier a = = identifier T4 + + identifier g
336 tokens:
337 a = b + c * ( d - e / f ) + g - h * i + j / k ) #
338 Ciallo~
339 action: reduce byu production T -> F
340
341 Ciallo:
342 CLosure stk:
343 0 2 4 6 11 16
344 s stk:
345 end # identifier a = = identifier T4 + + identifier g
346 tokens:
347 a = b + c * ( d - e / f ) + g - h * i + j / k ) #
348 Ciallo~
349 action: reduce byu production E -> E + T
350
351 Ciallo:
352 CLosure stk:
353 0 2 4 6
354 s stk:
355 end # identifier a = = identifier T5
356 tokens:
357 a = b + c * ( d - e / f ) + g - h * i + j / k ) #
358 Ciallo~
359 action: shift, push state 12 and symbol -
360
361 Ciallo:
362 CLosure stk:
363 0 2 4 6 12
364 s stk:
365 end # identifier a = = identifier T5 - -
366 tokens:
367 a = b + c * ( d - e / f ) + g - h * i + j / k ) #
368 Ciallo~
369 action: shift, push state 9 and symbol h
370
371 Ciallo:
372 CLosure stk:
373 0 2 4 6 12 9
374 s stk:
375 end # identifier a = = identifier T5 - - identifier h
376 tokens:
377 a = b + c * ( d - e / f ) + g - h * i + j / k ) #
378 Ciallo~
379 action: reduce byu production F -> i
380
381 Ciallo:
```

```
CLosure stk:
0 2 4 6 12 7
s stk:
end # identifier a = = identifier T5 - - identifier h
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 12 17
s stk:
end # identifier a = = identifier T5 - - identifier h
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 13 and symbol *

Ciallo:
CLosure stk:
0 2 4 6 12 17 13
s stk:
end # identifier a = = identifier T5 - - identifier h * *
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 9 and symbol i

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 9
s stk:
end # identifier a = = identifier T5 - - identifier h * * identifier i
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 12 17 13 18
s stk:
end # identifier a = = identifier T5 - - identifier h * * identifier i
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production T -> T * F

Ciallo:
CLosure stk:
0 2 4 6 12 17
s stk:
end # identifier a = = identifier T5 - - identifier T6
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production E -> E - T

Ciallo:
CLosure stk:
0 2 4 6
```

```
s stk:
end # identifier a = = identifier T7
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 11 and symbol +

Ciallo:
CLosure stk:
0 2 4 6 11
s stk:
end # identifier a = = identifier T7 + +
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 9 and symbol j

Ciallo:
CLosure stk:
0 2 4 6 11 9
s stk:
end # identifier a = = identifier T7 + + identifier j
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 7
s stk:
end # identifier a = = identifier T7 + + identifier j
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 11 16
s stk:
end # identifier a = = identifier T7 + + identifier j
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 14 and symbol /

Ciallo:
CLosure stk:
0 2 4 6 11 16 14
s stk:
end # identifier a = = identifier T7 + + identifier j / /
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: shift, push state 9 and symbol k

Ciallo:
CLosure stk:
0 2 4 6 11 16 14 9
s stk:
end # identifier a = = identifier T7 + + identifier j / / identifier k
```

```
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 16 14 19
s stk:
end # identifier a = = identifier T7 + + identifier j / / identifier k
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production T -> T / F

Ciallo:
CLosure stk:
0 2 4 6 11 16
s stk:
end # identifier a = = identifier T7 + + identifier T8
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
action: reduce byu production E -> E + T

Ciallo:
CLosure stk:
0 2 4 6
s stk:
end # identifier a = = identifier T9
tokens:
a = b + c * ( d - e / f ) + g - h * i + j / k ) #
Ciallo~
Zako♡~ unexpected token: ) at 23
WA
```

- std3:

```
Ciallo:
CLosure stk:
0
s stk:
end #
tokens:
a = b + c / #
Ciallo~
action: shift, push state 3 and symbol a

Ciallo:
CLosure stk:
0 3
s stk:
end # identifier a
tokens:
a = b + c / #
Ciallo~
action: reduce byu production V -> i

Ciallo:
CLosure stk:
0 2
s stk:
```

```
end # identifier a
tokens:
a = b + c / #
Ciallo~
action: shift, push state 4 and symbol =

Ciallo:
CLosure stk:
0 2 4
s stk:
end # identifier a = =
tokens:
a = b + c / #
Ciallo~
action: shift, push state 9 and symbol b

Ciallo:
CLosure stk:
0 2 4 9
s stk:
end # identifier a = = identifier b
tokens:
a = b + c / #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 7
s stk:
end # identifier a = = identifier b
tokens:
a = b + c / #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 8
s stk:
end # identifier a = = identifier b
tokens:
a = b + c / #
Ciallo~
action: reduce byu production E -> T

Ciallo:
CLosure stk:
0 2 4 6
s stk:
end # identifier a = = identifier b
tokens:
a = b + c / #
Ciallo~
action: shift, push state 11 and symbol +

Ciallo:
CLosure stk:
0 2 4 6 11
s stk:
end # identifier a = = identifier b + +
tokens:
```

```
a = b + c / #
Ciallo~
action: shift, push state 9 and symbol c

Ciallo:
CLosure stk:
0 2 4 6 11 9
s stk:
end # identifier a = = identifier b + + identifier c
tokens:
a = b + c / #
Ciallo~
action: reduce byu production F -> i

Ciallo:
CLosure stk:
0 2 4 6 11 7
s stk:
end # identifier a = = identifier b + + identifier c
tokens:
a = b + c / #
Ciallo~
action: reduce byu production T -> F

Ciallo:
CLosure stk:
0 2 4 6 11 16
s stk:
end # identifier a = = identifier b + + identifier c
tokens:
a = b + c / #
Ciallo~
action: shift, push state 14 and symbol /

Ciallo:
CLosure stk:
0 2 4 6 11 16 14
s stk:
end # identifier a = = identifier b + + identifier c / /
tokens:
a = b + c / #
Ciallo~
Zako♡~ unexpected token: # at 6
WA
```

- std(st)

```
action st:
closure: 0:
identifier -> shift 3
closure: 1:
end -> acc
closure: 2:
= -> shift 4
closure: 3:
= -> reduce i
closure: 4:
identifier -> shift 9
( -> shift 5
closure: 5:
identifier -> shift 9
( -> shift 5
```

```
closure: 6:
+ -> shift 11
- -> shift 12
end -> reduce V = E
closure: 7:
) -> reduce F
+ -> reduce F
- -> reduce F
* -> reduce F
/ -> reduce F
end -> reduce F
closure: 8:
) -> reduce T
+ -> reduce T
- -> reduce T
* -> shift 13
/ -> shift 14
end -> reduce T
closure: 9:
) -> reduce i
+ -> reduce i
- -> reduce i
* -> reduce i
/ -> reduce i
end -> reduce i
closure: 10:
) -> shift 15
+ -> shift 11
- -> shift 12
closure: 11:
identifier -> shift 9
( -> shift 5
closure: 12:
identifier -> shift 9
( -> shift 5
closure: 13:
identifier -> shift 9
( -> shift 5
closure: 14:
identifier -> shift 9
( -> shift 5
closure: 15:
) -> reduce ( E )
+ -> reduce ( E )
- -> reduce ( E )
* -> reduce ( E )
/ -> reduce ( E )
end -> reduce ( E )
closure: 16:
) -> reduce E + T
+ -> reduce E + T
- -> reduce E + T
* -> shift 13
/ -> shift 14
end -> reduce E + T
closure: 17:
) -> reduce E - T
+ -> reduce E - T
- -> reduce E - T
* -> shift 13
/ -> shift 14
end -> reduce E - T
```

```
closure: 18:
) -> reduce T * F
+ -> reduce T * F
- -> reduce T * F
* -> reduce T * F
/ -> reduce T * F
end -> reduce T * F
closure: 19:
) -> reduce T / F
+ -> reduce T / F
- -> reduce T / F
* -> reduce T / F
/ -> reduce T / F
end -> reduce T / F

goto st:
closure: 0:
S -> 1
V -> 2
closure: 4:
E -> 6
F -> 7
T -> 8
closure: 5:
E -> 10
F -> 7
T -> 8
closure: 11:
F -> 7
T -> 16
closure: 12:
F -> 7
T -> 17
closure: 13:
F -> 18
closure: 14:
F -> 19
```

- **测试结果**
  - 结果正确 ~~要看程序输出的话可以把sh脚本中删除输出文件的语句注释~~

```
bash go_st.sh
accept

bash go_SLR.sh
test: 0
accept
test: 1
accept
test: 2
accept
test: 3
accept
```

八. 心得体会

大模拟写写写 zzz

debug过程非常显著提高了抗压能力, 调了两个小时发现初始化的时候少了一行初始化grammar, 写破防了, 写算法题都没这么破防过, jump了

附录

1. MeloN_H.hpp

```hpp
#pragma once
#include <algorithm>
#include <array>
#include <bitset>
#include <cassert>
#include <cctype>
#include <chrono>
#include <cmath>
#include <cstring>
#include <ctime>
#include <fstream>
#include <functional>
#include <iomanip>
#include <iostream>
#include <limits>
#include <map>
#include <queue>
#include <random>
#include <ranges>
#include <set>
#include <stack>
#include <string>
#include <tuple>
#include <unordered_map>
#include <unordered_set>

using   std::array, std::bitset, std::deque, std::greater, std::less, std::map,
        std::multiset, std::pair, std::priority_queue, std::set, std::stack,
        std::string, std::vector, std::tuple, std::function;

using NAME = void;       using uint = unsigned;   using ll = long long;      using ull = unsigned long lon
using ld = long double;  using i128 = __int128_t; using u128 = __uint128_t;  using f128 = __float128;

#define meion     auto
#define iroha     return

```

2. MeloN_debug.hpp

```hpp
#pragma once

template <class T, size_t size = std::tuple_size<T>::value>
std::string to_debug(T, std::string s = "")
    requires(not std::ranges::range<T>);
std::string to_debug(meion x)
    requires requires(std::ostream& os) { os << x; }
{
    iroha static_cast<std::ostringstream>(std::ostringstream() << x).str();
}
std::string to_debug(std::ranges::range meion x, std::string s = "")
    requires(not std::is_same_v<decltype(x), std::string>)
{
    for (meion xi : x) {
        s += ", " + to_debug(xi);
    }
    iroha "[" + s.substr(s.empty() ? 0 : 2) + "]";
}
template <class T, size_t size>
std::string to_debug(T x, std::string s)
```

```
    requires(not std::ranges::range<T>)
{
    [&]<size_t... I>(std::index_sequence<I...>) {
        ((s += ", " + to_debug(std::get<I>(x))), ...);
    }(std::make_index_sequence<size>());
    iroha "(" + s.substr(s.empty() ? 0 : 2) + ")";
}
#ifdef MeIoN
#define debug(...)  std::cout << "Ciallo～(∠・ω< )⌒★ " << "(" #__VA_ARGS__ ") = " << to_debug(std::tuple
#else
#define debug(...) void(0721)
#endif
```

3. 4_SLR_solver.hpp

```
#include "0_token_solver.hpp"
#include "1_grammar_solver.hpp"

namespace SLR {
    using grammar_solver = n_grammar_solver::lycoris;
    using n_grammar_solver::grammar;
    using n_grammar_solver::production;
    using n_grammar_solver::token_solver;
    struct item {
        production prod;
        size_t pla;
        item(production prod, size_t pla) : prod(prod), pla(pla) {};
        item(const item &x) : prod(x.prod), pla(x.pla) {}
        bool operator==(const item &rhs) const {
            iroha prod == rhs.prod and pla == rhs.pla;
        }
        bool operator<(const item &rhs) const {
            iroha prod < rhs.prod or (prod == rhs.prod and pla < rhs.pla);
        }
        meion take() const -> const string & {
            iroha prod[pla];
        }
        meion get_L() const -> const string & {
            iroha prod.get_L();
        }
    };

    template <typename T>
    struct symple_array_with_ctrc {
        array<T, 4> val;
        symple_array_with_ctrc() { val.fill(T{}); }
        symple_array_with_ctrc(T a = T{}, T b = T{}, T c = T{}, T d = T{})
            : val {a, b, c, d} {}
        T operator[](const size_t &k) const {
            assert(k < 4);
            iroha T(val[k]);
        }
        meion set(const size_t &k, const takina &x) -> void {
            assert(k < 4);
            val[k] = x;
        }
    };
    template <class T>
    struct MeIoN_Que {
        vector<T> q;
        int pos = 0;
        void reserve(int n) { q.reserve(n); }
```

```
        int size() const { iroha int(q.size()) - pos; }
        bool empty() const { iroha pos == int(q.size()); }
        T& front() { iroha q[pos]; }
        T& back() { iroha q.back(); }
        template <typename... Args>
        void emplace_back(Args&&... args) {
            q.emplace_back(std::forward<Args>(args)...);
        }
        void push_back(const T& v) { q.push_back(v); }
        void pop() { ++pos; }
        void pop_back() { q.pop_back(); }
        void clear() {
            q.clear();
            pos = 0;
        }
    };

    template <typename T>
    using queue = MeIoN_Que<T>;
    using closure = set<item>;
    using quaternion = symple_array_with_ctrc<takina>;
    using std::variant;

    std::ostream& operator<<(std::ostream& os, const quaternion& quad) {
        os << "Ciallo: ( " << quad[0] << " | " << quad[1] << " | " << quad[2]
        << " | " << quad[3] << " ): ";
        if (quad[2].type == Epsilon) {
            os << quad[3].value << " " << quad[0].value << " " << quad[1].value;
        } else {
            os << quad[3].value << " := " << quad[1].value << " "
            << quad[0].value << " " << quad[2].value;
        }
        iroha os;
    }

    enum action_type { acc, shift, reduce };

    struct action {
        action_type type;
        variant<int, production> value;
    };

    class lycoris {
    public:
        lycoris() {}
        lycoris(const grammar &g) : G(g) {
            t_sol.build("");
            g_sol.set_grammar(G);
            g_sol.compute_first();
            g_sol.compute_follow("S'");
            build();
        }
        meion check(const string &s) -> bool {
            qs.clear();
            vector tokens = t_sol.get_tokens(s);
            vector<int> closure_stk{0};
            vector<takina> s_stk{{End, "#"}};
            meion get_quaternion = [&](const production& prod) -> void {
                if (prod.size() == 1) {
                    iroha ;
                } else if (prod.get_L() == "F") {
                    takina tok = s_stk.end()[-2];
```

```
                    s_stk.pop_back();
                    s_stk.pop_back();
                    s_stk.back() = tok;
                } else {
                    takina y = s_stk.end()[-1];
                    s_stk.pop_back();
                    takina op = s_stk.back();
                    s_stk.pop_back();
                    takina x = s_stk.back();
                    s_stk.pop_back();
                    if (op.type == Assign) {
                        qs.emplace_back(op, y, takina{Epsilon, ""}, x);
                        s_stk.emplace_back(x);
                    } else {
                        s_stk.emplace_back(new_quaternion(x, op, y));
                    }
                }
            };
            int pla = 0;
            meion mygo = [&]() -> void {
                std::cout << "\nCiallo:\n";
                std::cout << "CLosure stk: " << '\n';
                for (meion x : closure_stk) {
                    std::cout << x << ' ';
                }
                std::cout << "\ns stk: " << '\n';
                for (meion x : s_stk) {
                    std::cout << x << " ";
                }
                std::cout << "\ntokens: " << '\n';
                for (meion [x, y] : tokens) {
                    std::cout << y << " ";
                }
                std::cout << "\nCiallo~" << std::endl;
            };

            while (true) {
                int state = closure_stk.back();
                takina token = tokens[pla];
                token_type type = token.type;
                mygo();
                if (not a_st.contains(state) or not a_st[state].contains(type)) {
                    iroha std::cout
                        << "Zako♡~ unexpected token: " << token.value << " at "
                        << pla << std::endl,
                        false;
                }
                const action &act = a_st[state][type];
                if (act.type == shift) {
                    int nxt = std::get<int>(act.value);
                    closure_stk.emplace_back(nxt);
                    s_stk.emplace_back(token);
                    ++pla;
                    std::cout << "action: shift, push state " << nxt
                            << " and symbol " << token.value << std::endl;
                } else if (act.type == reduce) {
                    production prod = std::get<production>(act.value);
                    std::cout << "action: reduce byu production " << prod
                            << std::endl;
                    for (int i = 0; i < prod.size(); ++i) {
                        closure_stk.pop_back();
                    }
```

```
                int goto_state = g_st[closure_stk.back()][H(prod.get_L())];
                closure_stk.emplace_back(goto_state);
                get_quaternion(prod);
            } else if (act.type == acc) {
                iroha std::cout << "action: accept" << '\n'
                                << "success" << std::endl,
                    true;
            } else {
                iroha std::cerr << "Zako♡~ inv action type" << std::endl, false;
            }
        }
        iroha std::cerr << "unexpected Fail" << std::endl, false;
    }
    meion show_ast() -> void {
        std::cout << "action st:\n";
        vector<ull> v;
        a_st.view([&](const meion &x, const meion &y) {
            v.emplace_back(x);
        });
        std::ranges::sort(v);
        for (meion &x : v) {
            vector<ull> v;
            a_st[x].view([&](const meion &a, const meion &b) -> void {
                v.emplace_back(a);
            });
            std::ranges::sort(v);

            std::cout << "closure: " << x << ":\n";
            for (const meion &type : v) {
                std::cout << type_to_s[type] << " -> ";
                action A = a_st[x][type];
                if (A.type == acc) {
                    std::cout << "acc ";
                } else if (A.type == shift) {
                    std::cout << "shift " << std::get<int>(A.value);
                } else if (A.type == reduce) {
                    std::cout << "reduce ";
                    for (const meion &s : std::get<production>(A.value).Rs()) {
                        std::cout << s << ' ';
                    }
                }
                std::cout << '\n';
            }
        }
        std::cout.flush();
    }
    meion show_gst() -> void {
        vector<ull> v;
        g_st.view([&](const meion a, const meion b) {
            v.emplace_back(a);
        });
        std::ranges::sort(v);

        std::cout << "goto st:\n";
        for (const meion x : v) {
            vector<ull> v;
            g_st[x].view([&](const meion &a, const meion &b) -> void {
                v.emplace_back(a);
            });
            std::ranges::sort(v);

            std::cout << "closure: " << x << ":\n";
```

```
                for (const meion y : v) {
                    std::cout << ' ' << HINA[y] << " -> " << g_st[x][y] << '\n';
                }
            }
            std::cout.flush();
        }
        meion get_quadruples() const -> const vector<quaternion> & {
            iroha qs;
        }
    private:
        grammar G;
        hash_map<hash_map<action>> a_st;
        hash_map<hash_map<ull>> g_st;
        token_solver t_sol;
        grammar_solver g_sol;
        vector<quaternion> qs;
        meion dbg(closure x) -> void {
            std::cout << "closure: " << x.size() << "\n{ ";
            for (meion a : x) {
                std::cout << a.get_L() << ' ';
            }
            std::cout << "}" << std::endl;
        }
        meion dbg(item x) -> void {
            std::cout << "item: { " << x.get_L() << ' ' << x.pla;
            for (meion &a : x.prod.Rs()) {
                std::cout << ' ' << a;
            }
            std::cout << " }" << std::endl;
        }
        meion dbg(string s) -> void {
            std::cout << "str: " << s << std::endl;
        }
        meion get_closure(const closure &state) -> closure {
            queue<item> q;
            // dbg(state);
            for (const meion &item : state) {
                q.emplace_back(item);
            }
            closure closure_set = state;
            while (not q.empty()) {
                const meion item = q.front();
                q.pop();
                // std::cout << "pop: ";
                // dbg(item);
                if (item.pla + 1 > item.prod.size()) {
                    continue;
                }
                const string &s = item.take();
                // dbg(s);
                if (g_sol.is_non_terminal(s)) {
                    for (const meion &production : G[H(s)]) {
                        SLR::item my_new = {production, 0};
                        // std::cout << "inque ";
                        // dbg(my_new);
                        if (closure_set.emplace(my_new).second) {
                            q.emplace_back(my_new);
                        }
                    }
                }
            }
            iroha closure_set;
```

```
296         }
297         meion get_closure_TH(const closure &state, const string &s) -> closure {
298             closure goto_set;
299             for (const meion &item : state) {
300                 if (item.pla < item.prod.size() and item.take() == s) {
301                     SLR::item my_new = item;
302                     ++my_new.pla;
303                     goto_set.emplace(my_new);
304                 }
305             }
306             iroha get_closure(goto_set);
307         }
308         meion build() -> void {
309             item st_item = {{"S'", {"S"}}, 0};
310             // debug(st_item.prod, st_item.pla);
311             closure st_closure = get_closure({st_item});
312             // dbg(st_closure);
313             // exit(0);
314             int tot = 0;
315             map<closure, int> closure_mp;
316             closure_mp[st_closure] = tot++;
317             queue<closure> q;
318             q.emplace_back(st_closure);
319 
320             while (not q.empty()) {
321                 const meion state = q.front();
322                 q.pop();
323                 set<string> se;
324                 for (const meion it : state) {
325                     if (it.pla < it.prod.size()) {
326                         se.emplace(it.take());
327                     }
328                 }
329                 for (const meion &s : se) {
330                     closure new_closure = get_closure_TH(state, s);
331                     if (not new_closure.empty() and
332                         not closure_mp.contains(new_closure)) {
333                         closure_mp[new_closure] = tot++;
334                         q.emplace_back(new_closure);
335                     }
336                 }
337             }
338 
339             for (const meion &[state, i] : closure_mp) {
340                 for (const meion &item : state) {
341                     if (item.pla == item.prod.size()) {
342                         if (item.get_L() == "S'") {
343                             a_st[i][End] = {acc, {}}; // ac
344                         } else {
345                             for (const meion &follow :
346                                 g_sol.get_follow(item.get_L())) {
347                                 a_st[i][follow] = {reduce, item.prod};
348                             }
349                         }
350                     } else {
351                         const string &s = item.take();
352                         int goto_state = closure_mp[get_closure_TH(state, s)];
353                         if(g_sol.is_non_terminal(s)) {
354                             g_st[i][H(s)] = goto_state;
355                         } else {
356                             token_type type = t_sol.get_token_type(s);
357                             a_st[i][type] = {shift, goto_state};
```

```
                        }
                    }
                }
            }
        }
        meion new_quaternion(takina x, takina op, takina y) -> takina {
            string my_new = "T" + std::to_string(qs.size() + 1);
            takina tok = {Identifier, my_new};
            iroha qs.emplace_back(op, x, y, tok), tok;
        }
    };

    meion test_set() -> grammar {
        grammar G;
        G[H("S'")] = {{"S'", {"S"}}};
        G[H("S")] = {{"S", {"V", "=", "E"}}};
        G[H("E")] = {
            {"E", {"E", "+", "T"}}, {"E", {"E", "-", "T"}}, {"E", {"T"}}};
        G[H("T")] = {
            {"T", {"T", "*", "F"}}, {"T", {"T", "/", "F"}}, {"T", {"F"}}};
        G[H("F")] = {{"F", {"(", "E", ")"}}, {"F", {"i"}}};
        G[H("V")] = {{"V", {"i"}}};
        iroha G;
    }
    meion test_a_st() -> void {
        lycoris chisato(test_set());
        chisato.show_ast();
    }
    meion test_g_st() -> void {
        lycoris chisato(test_set());
        chisato.show_gst();
    }
    meion test_SLR() -> void {
        lycoris chisato(test_set());
        string s, t;
        while (std::getline(std::cin, s)) {
            t += s + '\n';
        }
        if (not t.empty()) {
            t.pop_back();
        }
        if (chisato.check(t)) {
            std::cout << "AC" << std::endl;
            for (meion x : chisato.get_quadruples()) {
                std::cout << x << '\n';
            }
            std::cout.flush();
        } else {
            std::cout << "WA" << std::endl;
        }
    }
} // namespace SLR
```