پروژه کامپایلر فاز اول

اعضا گروه:

سام احمدی زاده

بردیا هاشمی

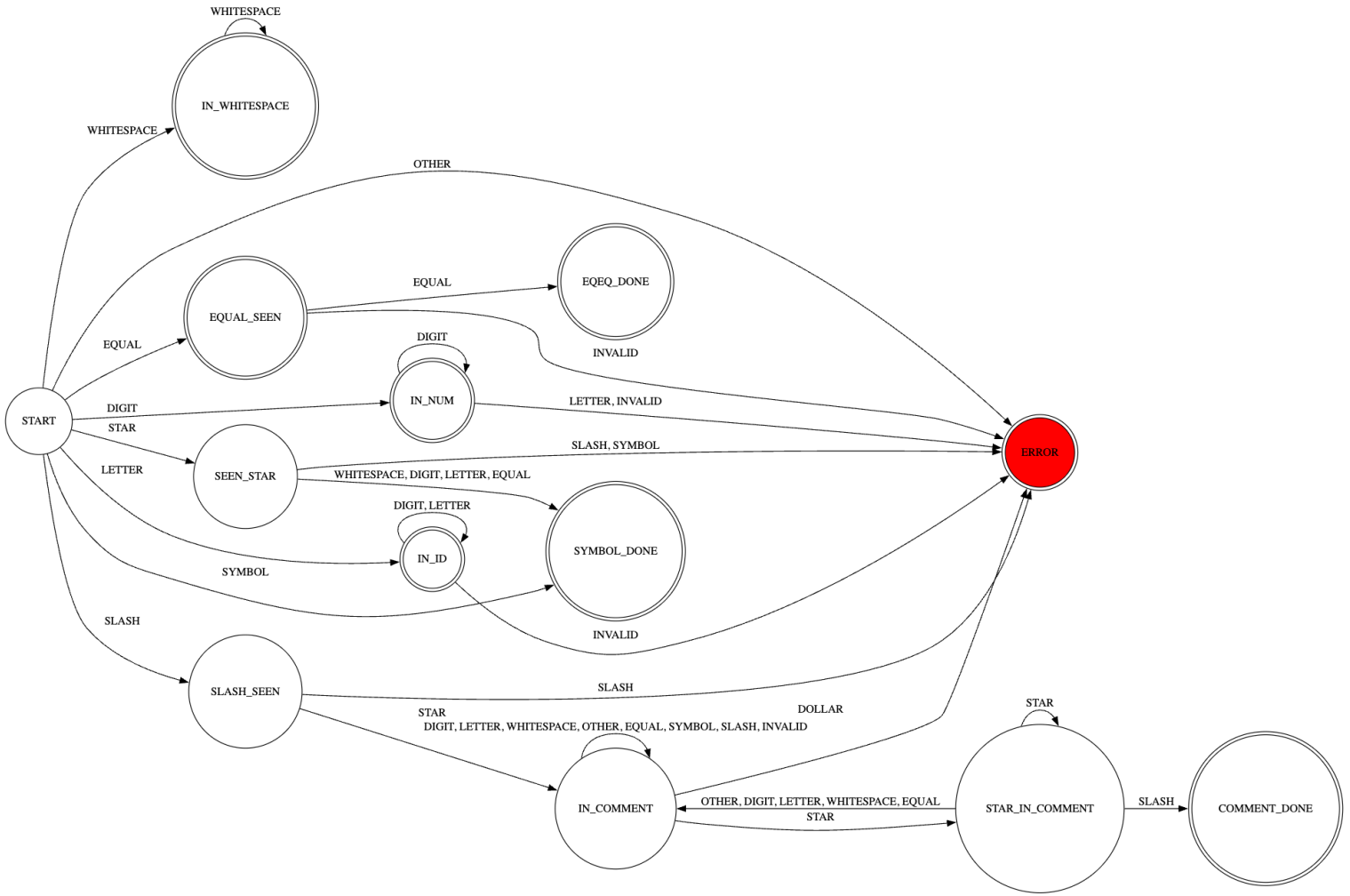
مهرانا خالقی

مقدمه:

این کد به زبان Python پیاده‌سازی شده و هدف آن پیاده‌سازی مرحله‌ی اسکنر (تحلیل واژگانی) برای زبان برنامه‌نویسی ساده‌شده‌ای به نام **C minus** است.

توضیحات DFA:

در این فاز ما از روش تشخیص توکن ها با DFA استفاده کردیم. DFA مربوطه به صورت زیر می‌باشد:



- DFA نشان داده شده در تصویر به صورت دو دیکشنری در هم پیاده شده است.

- ساختار DFA به این صورت است که طولانی‌ترین رشته انتخاب شده را پیدا کند.

توضیحات کد:

- همانطور که گفته شد تشخیص توکن ها به صورت Table Driven DFA می‌باشد و پیاده سازی آن به صورت دو دیکشنری در هم می‌باشد و کد تا زمانی ادامه پیدا می‌کند که State برا کاراکتر Lookahead وجود نداشته باشد. این پیاده سازی همچنین نیازمند یک کاراکتر lookahead است.

- با فراخوانی تابع get_next_token این کد توکن بعدی را می‌دهد. این ویژگی با استفاده از yield در پایتون پیاده سازی شده است.

```
def get_next_token(input_str):
    tokens = []
    i = 0
    n = len(input_str)

    while i < n:
        state = 'START'
        lexeme = ''
        last_final_state = None
        last_final_lexeme = ''

        while i < n:
            ch = input_str[i]
            cls = char_class(ch)
            next_state = DFA.get(state, {}).get(cls)
            if next_state:
                if state == 'EQUAL_SEEN' and cls != 'EQUAL' and cls !='INVALID':
                    break

                if state == 'SEEN_STAR' and cls != 'SLASH':
                    print(state, cls)
                    i -= 1
                    break

                state = next_state
                lexeme += ch
                i += 1
                if state in FINAL_STATES:
                    last_final_state = state
                    last_final_lexeme = lexeme

                if state == 'COMMENT_DONE':
                    break
            else:
                break
```

- تابع analyze_code که ارورها، توکن ها و ... را به صورت یکجا در یک فایل می‌ریزد، با فراخوانی چند باره تابع get_next_token پیاده سازی شده و خط مربوط به هر توکن را به عنوان کلید دیکشنری نگه می‌دارد.

```python
def analyze_code(input_code):
    tokens = {1:[]}
    errors = {1:[]}
    symbol_table =  set()
    comment_open = False
    lineno = 1

    for token in get_next_token(input_code):
        token_type, token_val = token
        if token_type == 'WHITESPACE' and "\n" in token_val:
            lineno += token_val.count("\n")
            tokens[lineno] = []
            errors[lineno] = []
            continue

        if token_type == 'WHITESPACE' or token_type == 'COMMENT':
            continue

        if token_type == 'ID' or token_type == 'KEYWORD':
            symbol_table.add(token_val)



        if token_type == 'ERROR':
            error_msg = (token_val, "Invalid input")
            if token_val.startswith('/*'):
                error_msg = (token_val[:7] + "...", 'Unclosed comment')
            if token_val[-1].isdigit() and any(c.isalpha() for c in token_val):
                error_msg = (token_val, "Invalid number")
            elif token_val == '*/':
                error_msg = (token_val, "Unmatched comment")
            errors[lineno].append(error_msg)
        else:
            tokens[lineno].append((token_type, token_val))

    return tokens, symbol_table, errors, lineno
```