中山大學 SUN YAT-SEN UNIVERSITY

计算机学院 COMPUTER SCIENCE AND ENGINEERING

国家超级计算广州中心 NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

编译器构造实验

实验一 词法分析器

Yat Compiler Construction with AI

yatcc-ai.com

中山大学 计算机学院

国家超级计算广州中心

2025.3

www.nscc-gz.cn

OUTLINE

目　　录

中山大学计算机学院
School of Computer Science & Engineering

实验内容：基于框架flex或antlr实现一个词法分析器

输入词法分析器的文件

build > test > task0 > functional-0 > C 000_main.sysu.c > ...

```
1   # 1 "/YatCC/test/cases/functional-0/000_main.sysu.c"
2   # 1 "<built-in>" 1
3   # 1 "<built-in>" 3
4   # 389 "<built-in>" 3
5   # 1 "<command line>" 1
6   # 1 "<built-in>" 2
7   # 1 "/YatCC/test/cases/functional-0/000_main.sysu.c" 2
8   int main(){
9       return 3;
10  }
```

初始的残缺词法分析器输出

build > test > task1 > functional-0 > 000_main.sysu.c > output.txt

```
1   int 'int'   [StartOfLine]   Loc=<0:0>
2   identifier 'main'   Loc=<0:0>
3   l_paren '(' Loc=<0:0>
4   r_paren ')' Loc=<0:0>
5   l_brace '{' Loc=<0:0>
6   return 'return' Loc=<0:0>
7   numeric_constant '3'    Loc=<0:0>
8   semi ';'    Loc=<0:0>
9   r_brace '}' Loc=<0:0>
10  eof ''  Loc=<0:0>
```

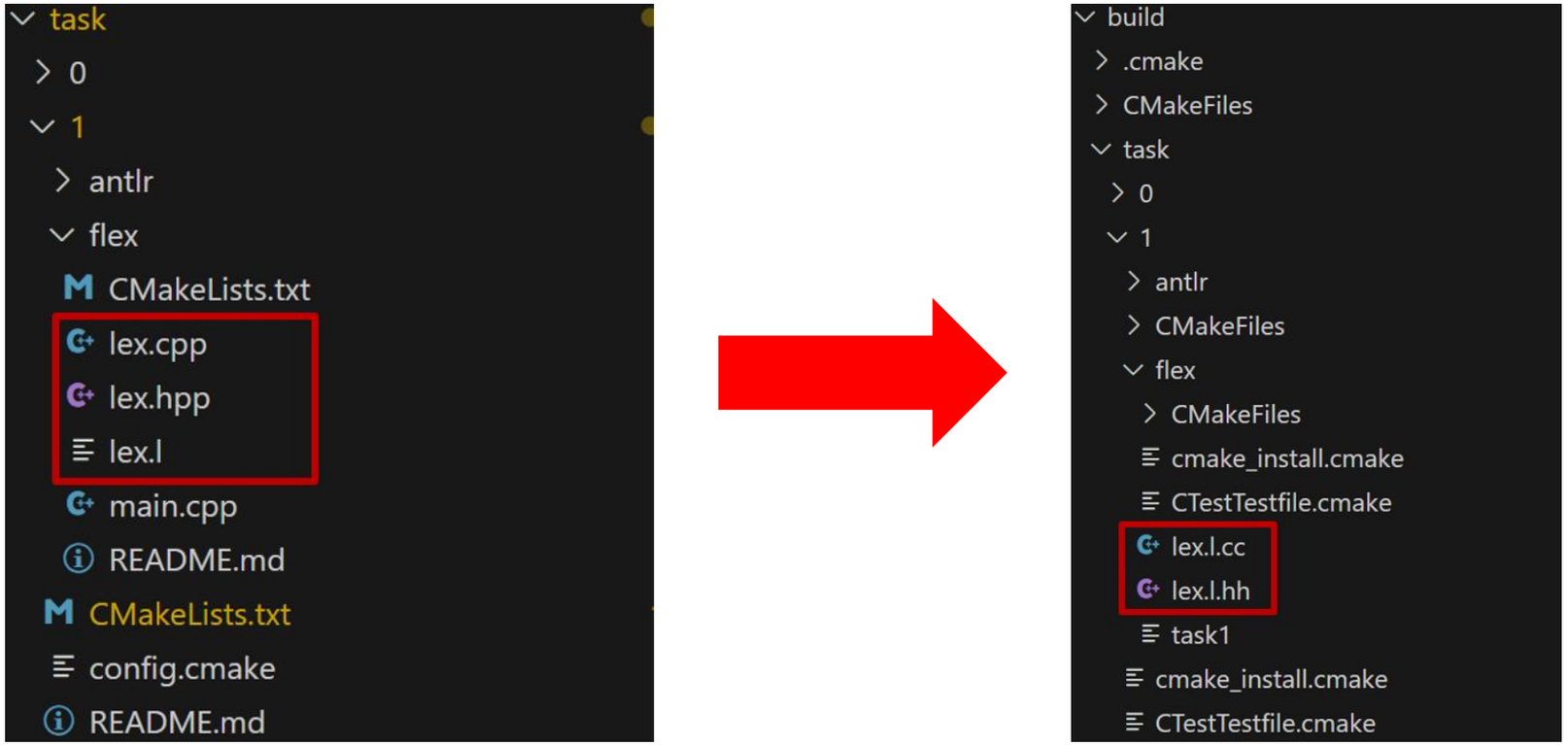
clang词法分析器输出的标准答案

build > test > task1 > functional-0 > 000_main.sysu.c > answer.txt

```
1   int 'int'    [StartOfLine]  Loc=</YatCC/test/cases/functional-0/000_main.sysu.c:1:1>
2   identifier 'main'    [LeadingSpace] Loc=</YatCC/test/cases/functional-0/000_main.sysu.c:1:5>
3   l_paren '('     Loc=</YatCC/test/cases/functional-0/000_main.sysu.c:1:9>
4   r_paren ')'     Loc=</YatCC/test/cases/functional-0/000_main.sysu.c:1:10>
5   l_brace '{'     Loc=</YatCC/test/cases/functional-0/000_main.sysu.c:1:11>
6   return 'return'  [StartOfLine] [LeadingSpace]   Loc=</YatCC/test/cases/functional-0/000_main.sysu.c:2:5>
7   numeric_constant '3'     [LeadingSpace] Loc=</YatCC/test/cases/functional-0/000_main.sysu.c:2:12>
8   semi ';'        Loc=</YatCC/test/cases/functional-0/000_main.sysu.c:2:13>
9   r_brace '}'  [StartOfLine]  Loc=</YatCC/test/cases/functional-0/000_main.sysu.c:3:1>
10  eof ''      Loc=</YatCC/test/cases/functional-0/000_main.sysu.c:3:2>
```

使用flex完成词法分析器实验

Flex: fast lexical analyzer generator（快速词法分析器生成器）



使用flex完成词法分析器实验

识别token

1. 简单token规则

```
"int"        { ADDCOL(); COME(INT); }
"return"     { ADDCOL(); COME(RETURN); }
```

2. 正则表达式定义

```
D       [0-9]
L       [a-zA-Z_]
IS      ((u|U)|(u|U)?(l|L|ll|LL)|(l|L|ll|LL)(u|U))
```

3. 用正则表达式定义复杂token规则

```
0[0-7]*{IS}?            { ADDCOL(); COME(CONSTANT); }
[1-9]{D}*{IS}?          { ADDCOL(); COME(CONSTANT); }
```

识别token的loc

```
#define ADDCOL() g.mColumn += yyleng;
```

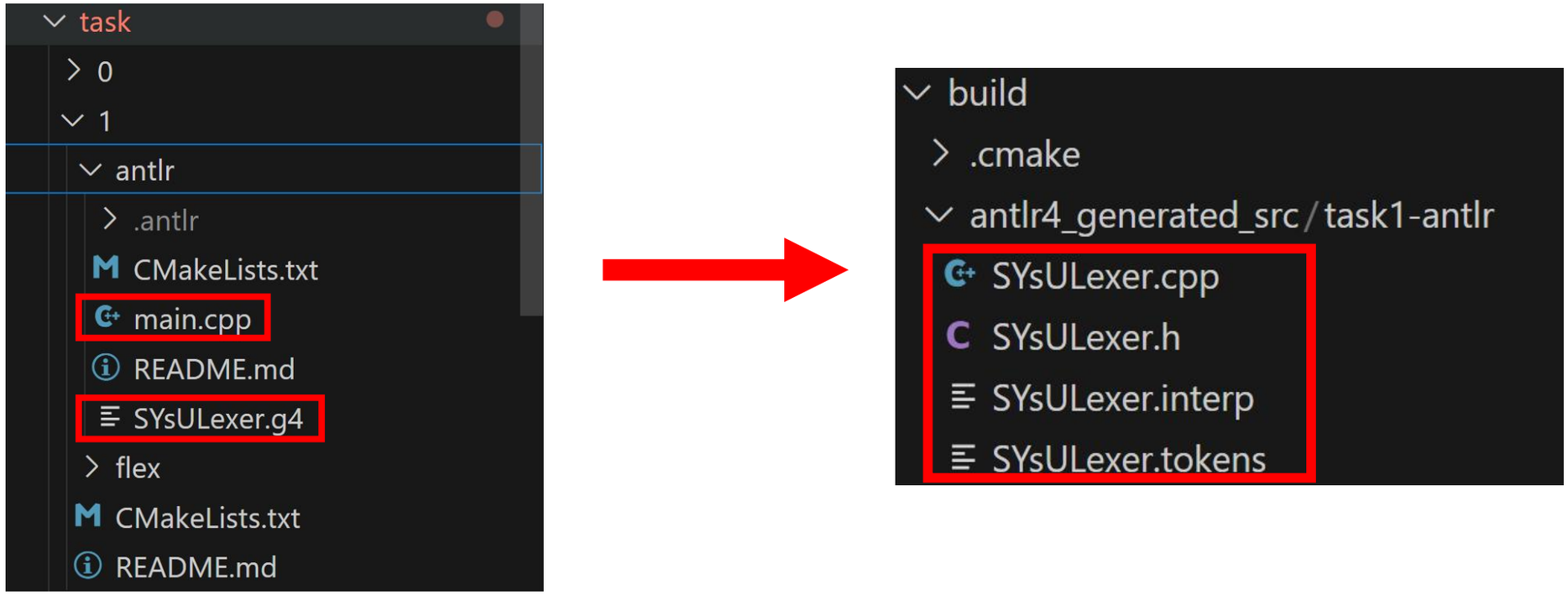
一些可能用到的函数与变量

- yylex(): 词法分析器的主要入口点，每次调用返回下一个词法单元。
- yy_scan_string(const char *str): 使词法分析器从一个字符串而不是标准输入或文件中读取输入。
- yy_switch_to_buffer(YY_BUFFER_STATE new_buffer): 切换当前的输入缓冲区。
- yy_create_buffer(FILE *file, int size): 为给定的文件创建一个新的输入缓冲区。
- yy_delete_buffer(YY_BUFFER_STATE b): 删除一个输入缓冲区。
- yyrestart(FILE *file): 重置词法分析器的状态并从新的文件开始读取输入。
- YY_BUFFER_STATE: 表示输入缓冲区的状态的类型。
- yylval: 在与 yacc/bison 配合使用时，用于传递词法单元的值。
- yytext: 包含当前匹配的文本。
- yyleng: 包含yytext的长度。
- yylineno: 跟踪当前的行号（如果%option yylineno被使用）。

使用antlr完成词法分析器实验

ANTLR : Another Tool for Language Recognition

是一个强大的工具，用于生成词法分析器、解析器以及遍历代码生成树的代码。



使用antlr完成词法分析器实验

识别token

1. 简单token规则

task > 1 > antlr > SYsULexer.g4 > ...

```
1   lexer grammar SYsULexer;
2
3   Int : 'int';
4   Return : 'return';
```

词法规则的定义需要首字母大写

2. 正则表达式定义

```
fragment
Nondigit
    :   [a-zA-Z_]
    ;
```

3. 用正则表达式定义复杂token规则

```
Identifier
    :   IdentifierNondigit
        (   IdentifierNondigit
        |   Digit
        )*
    ;
```

识别token的loc

一些你可能用到的词法单元接口

主要属性：

- getType()：获取词法单元的类型，类型通常由词法分析器的规则定义。
- getText()：获取词法单元的文本内容。
- getLine()：获取词法单元出现的行号。
- getCharPositionInLine()：获取词法单元在其所在行的位置（字符偏移量）。

OUTLINE

目　　录

中山大学计算机学院
School of Computer Science & Engineering

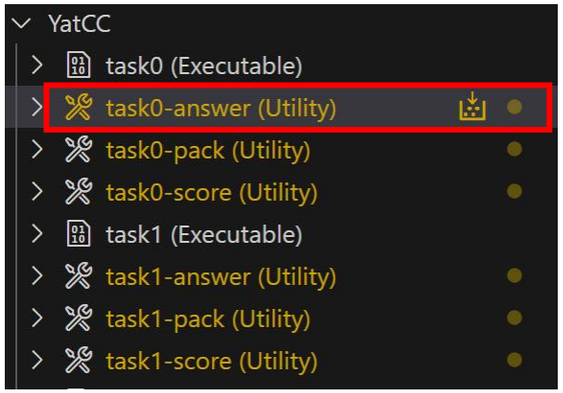
评分标准：1. 是否提取出正确的token（60 分）

2. 是否提取出正确的token location（30 分）

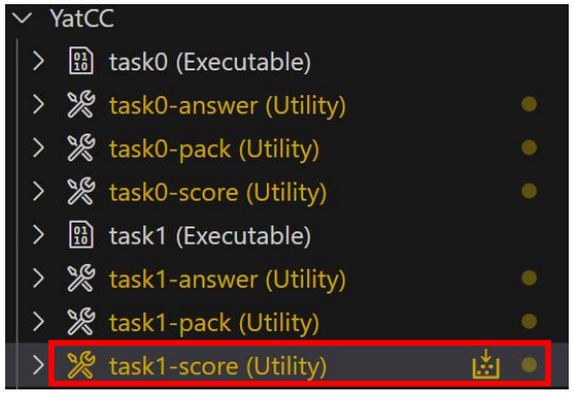
3. 是否识别其他无关字符（10 分）

ps： eof这个token 对应的行号和列号不计入评分

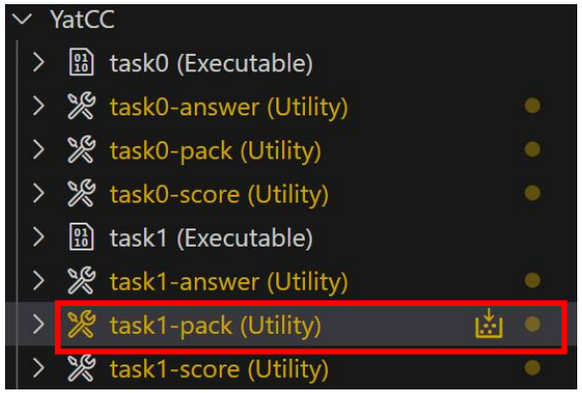
构建task0-answer



构建task1-score



打包代码



OUTLINE

目　　录

中山大学计算机学院
School of Computer Science & Engineering

手把手教你用ai辅助完成词法分析器实验

Prompt示例：
你是一位精通编译原理和词法分析器设计的专家助教，专门指导学生完成基于 Flex/ANTLR 的词法分析实验。
请根据以下角色设定和实验要求回答学生问题：

角色设定
- 身份：
  编译原理专家，熟悉 Flex/ANTLR 框架和 clang 词法规范
- 语气：
  耐心且严谨，用中文回答，必要时给出代码示例
- 任务：
  指导学生完成词法分析器实现，解决格式/位置输出问题
- 限制：
  不提供完整代码，仅给出关键思路或伪代码片段

实验核心要求

1. 输出必须包含以下字段（示例格式）：

   ```
   <别名> '<原始词素>' [属性标记]     Loc=<文件路径:起始行:起始列>
   ```

2. 必须处理的属性标记：
   - [StartOfLine]：词法单元位于行首
   - [LeadingSpace]：词法单元前有空格
3. 必须跳过的内容：
   - 预处理指令（以#开头的行）
   - 文件结束符`eof`的位置信息（只需输出别名）
4. 必须与 clang -cc1 -dump-tokens 的输出格式严格一致

示例问答

1. **框架配置问题**
   - 示例问题："如何选择 Flex 还是 ANTLR？"
   - 回答模式：
     ```
     比较框架差异：
     | 特性       | Flex                  | ANTLR                    |
     | -------- | --------------------- | ------------------------ |
     | 语法       | 正则表达式规则          | 类似BNF的语法规则          |
     | 输出控制   | 需手动处理位置计算      | 自动生成监听器接口         |
     | 适用场景   | 纯词法分析             | 词法+语法分析联合使用      |
     建议选择依据：______（根据问题上下文补充）
     ```

2. **词法规则问题**
   - 示例问题："为什么我的数字常量识别结果不正确？"
   - 回答模式：
     ```
     分步检查建议：
     1. 检查正则表达式是否覆盖边界情况（如十六进制/八进制）
     2. 验证规则优先级顺序（如确保数字规则在标识符之前）
     3. 使用测试用例调试：
        ```c
        123    // 应识别为numeric_constant
        0x1f   // 十六进制测试
     ```

token位置提取、其他字符识别、测试调试 .....
     ```

deepseek NSCC GZ Starlight

yatcc-ai.com

谢　谢！