编译原理实践第1次课
正则表达式到NFA和DFA的转换

张昊　1927405160

概述

使用 C++ 实现了正则式转NFA（MYT），NFA 转 DFA（子集构造法）以及 DFA 最小化（Hopcroft 算法）。

> 其中 DFA 的构造没有消除死状态；构造的自动机状态标号尽量小，但不保证连续。

项目结构（`src` 目录）如下：

```
.
|  CMakeLists.txt    CMake 项目配置文件
|  lexer.h
|  minimize.cpp
|  myt.cpp
|  subset.cpp
|  tools.h
|
├─lexer
|      dfa.cpp
|      dfa.h
|      dfa_helper.hpp
|      fa.cpp
|      fa.h
|      nfa.cpp
|      nfa.h
|      properties.h
|      regex.cpp
|      regex.h
|
└─tools
       command_line.cpp
       command_line.h
       file_handler.cpp
       file_handler.h
```

编程环境说明

- **语言**：C++ 11
- **文件编码**：UTF-8 （目录、输入文件需要无中文等非ASCII字符）
- **开发环境**：
  - Windows 10, CLion, MinGW, 使用 CMake
- **测试环境**：
  - Windows 10, Visual Studio 2017：编译通过
  - Ubuntu 20.04, gcc 9.3.0, 使用 CMake：编译通过

编译

本项目编译后将生成**三个**可执行程序，分别为：

- `myt.exe` 或 `myt` ： 正则式转 NFA
- `subset-construction.exe` 或 `subset-construction` ： NFA 转 DFA
- `minimize.exe` 或 `minimize` ： DFA 最小化

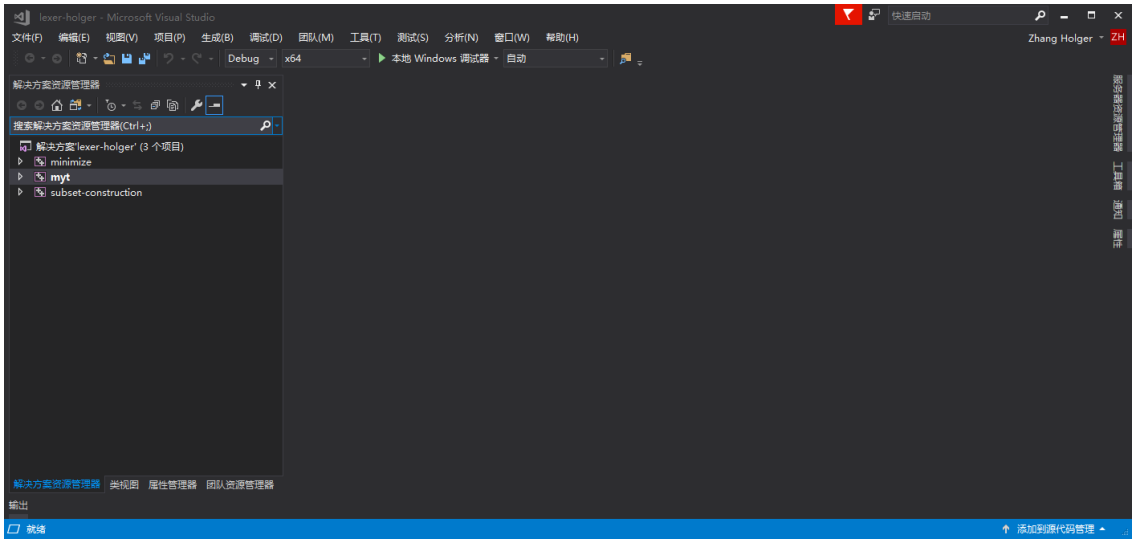
> 项目提供了两套配置（工程）文件，分别用于支持在 Linux 和 Windows 下编译。

注意：需使用**较为完整支持** C++ 11 特性的编译器编译本项目，推荐使用 Visual Studio 2017+、GCC 8.3.0+。

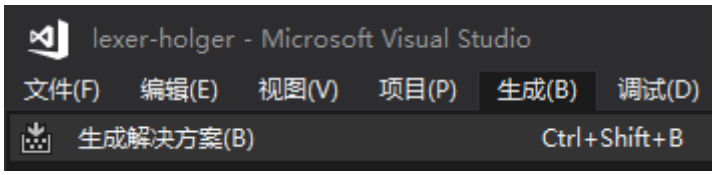
Windows 系统

> 编译前准备：Visual Studio 2017 或**更高**版本

1. 打开项目根目录下的 `vsproj` 文件夹，使用 Visual Studio 2017 或**更高**版本打开 `lexer-holger.sln` 文件。



2. **编译**：点击 “生成” -> "生成解决方案"，即可在 `vsproj/Debug` 目录下找到三个编译好的可执行文件。



Linux 系统（Ubuntu）

> 编译前准备：gcc (>=8.3.0), cmake (>= 3.6), 正确配置了 C++ 编译环境

在 **shell** 中使用如下方法编译：

```
$ pwd
/path/to/project-folder
$ cd src
$ mkdir build  # 创建编译文件夹
$ cd build
$ cmake ..  # 编译
$ make
```

即可在 `build` 目录下获得三个编译好的可执行文件。

运行

以在 Linux 系统下运行为例，输入文件为 `regex.txt`：

```
(a|b)*abb
```

注：所有编译结果、输入输出文件均包含在 `example` 目录下。

正则表达式转 NFA

```
$ ./myt regex.txt nfa.txt
lexer-holger 21.09.15 [MYT]
  [input file] regex.txt
  [output file] nfa.txt
result has been successfully written to the output file!
```

输出 `nfa.txt` 文件如下：

```
6 e 7
6 e 4
4 e 2
4 e 0
0 a 1
2 b 3
1 e 5
3 e 5
5 e 4
5 e 7
7 a 8
8 b 9
9 b 10
start state: 6
accepting states: 10
```

NFA 转 DFA

```
$ ./subset-construction nfa.txt dfa.txt
lexer-holger 21.09.15 [Subset Construction]
  [input file] nfa.txt
  [output file] dfa.txt
result has been successfully written to the output file!
```

上一次的输出结果 `nfa.txt` 作为输入，输出 `dfa.txt` 文件如下：

```
0 a 1
0 b 2
1 a 1
1 b 3
2 a 1
2 b 2
3 a 1
3 b 4
4 a 1
4 b 2
start state: 0
accepting states: 4
```

正则表达式转 NFA

```
$ ./minimize dfa.txt min-dfa.txt
lexer-holger 21.09.15 [DFA Hopcroft Minimize]
  [input file] dfa.txt
  [output file] min-dfa.txt
result has been successfully written to the output file!
```

上一次的输出结果 `dfa.txt` 作为输入，输出 `min-dfa.txt` 文件如下：

```
0 a 2
0 b 3
1 a 2
1 b 0
2 a 2
2 b 1
3 a 2
3 b 3
start state: 3
accepting states: 0
```