FLA Project 实验报告

完成了PDA和TM的普通模式和verbose模式，和三个程序的构造，额外完成了PDA的verbose模式

程序设计思路

使用面向对象的思想，定义对象类PDA，转移函数PDA_Transition，TM，TM_Transition，纸带Tape。这样做结构清晰，也方便访问内部信息，比如PDA类内部有 Q S G q0 z0 F 和 current state 和 stack，Tape类有read write move 等函数

main函数：首先解析命令行参数得到用户传入信息，选择PDA还是TM；

然后调用PDA或TM类的构造函数，相当于解析器；

接着调用PDA或TM类的执行函数，相当于模拟器。

命令行参数解析

当用户输入 -h 或 --help时，正常输出程序用法，正常退出

```
root@xkxk:~/workspace/fla# ./bin/fla -h
usage:   fla [-h|--help] <pda> <input>
         fla [-v|--verbose] [-h|--help] <tm> <input>
```

当用户输入参数错误时，错误输出Argument error，退出代码为1

```
root@xkxk:~/workspace/fla# ./bin/fla
Argument error: Missing arguments
Please use -h or --help to get help
root@xkxk:~/workspace/fla# ./bin/fla mytm.txt 101
Argument error: Invalid file name
Please use -h or --help to get help
```

当用户输入正确时，解析参数并进行下一步的模拟

解析器

在utils中定义了一系列字符串处理的函数，然后一行行对文件进行读入

```
std::vector<std::string> split(std::string str, char delimiter);
std::string clean_string(std::string str);
char string_to_char(std::string str);
std::string char_to_string(char ch);
std::string clean_line(std::string str);

std::set<std::string> get_set_of_string(std::string str);
std::set<char> get_set_of_char(std::string str);
std::string get_symbol_from_assignment(std::string str);
std::string get_rvalue_from_assignment(std::string str);

bool check_string_in_stringset(std::string str, std::set<std::string> strset);
bool check_char_in_charset(char ch, std::set<char> chset);
bool check_stringset_in_stringset(std::set<std::string> strset1, std::set<std::string> strset2);
```

遇到错误就输出syntax error，退出代码1

模拟器

首先初始化，初始状态q0，压入初始栈符号 / 输入写入纸带

接着模拟执行，PDA是遍历输入，每次遍历所有转移函数找到一个可行的转移函数进行转移，注意有空转移就走空转移；

TM也是每步找到转移函数，进行转移，具体到每条纸带，就是进行读取、移动、修改等操作。

执行结果

普通模式

```
root@xkxk:~/workspace/fla# ./bin/fla pda/anbn.pda aaabbb
true
root@xkxk:~/workspace/fla# ./bin/fla pda/anbn.pda aabbb
false
root@xkxk:~/workspace/fla# ./bin/fla tm/palindrome_detector_2tapes.tm 100010001
true
root@xkxk:~/workspace/fla# ./bin/fla tm/palindrome_detector_2tapes.tm 10001000
false
root@xkxk:~/workspace/fla# ./bin/fla tm/palindrome_detector_2tapes.tm 100A1A001
illegal input
```

TM的verbose模式

```
root@xkxk:~/workspace/fla# ./bin/fla -v tm/palindrome_detector_2tapes.tm 100A1A001
Input: 100A1A001
==================== ERR ====================
error: 'A' was not declared in the set of input symbols
Input: 100A1A001
          ^
==================== END ====================
```

```
root@xkxk:~/workspace/fla# ./bin/fla -v tm/palindrome_detector_2tapes.tm 1001001
Input: 1001001
==================== RUN ====================
Step   : 0
State  : 0
Index0 : 0 1 2 3 4 5 6
Tape0  : 1 0 0 1 0 0 1
Head0  : ^
Index1 : 0
Tape1  : _
Head1  : ^
---------------------------------------------
Step   : 1
State  : cp
Index0 : 0 1 2 3 4 5 6
Tape0  : 1 0 0 1 0 0 1
```

```
Step   : 29
State  : halt_accept
Index0 : 7 8 9 10
Tape0  : t r u e
Head0  :       ^
Index1 : 1
Tape1  : _
Head1  : ^
---------------------------------------------
Result: true
==================== END ====================
```

Bonus

仿照TM的verbose模式，完成了PDA的verbose模式

```
root@xkxk:~/workspace/fla# ./bin/fla -v pda/anbn.pda abc
Input: abc
==================== ERR ====================
error: 'c' was not declared in the set of input symbols
Input: abc
         ^
==================== END ====================
```

```
root@xkxk:~/workspace/fla# ./bin/fla -v pda/anbn.pda ab
Input: ab
==================== RUN ====================
Step: 0
Input: ab
       ^
State: q0
Stack: z
---------------------------------------------
Transition: delta(q0, a, z) = (q1, 1z)
Step: 1
Input: ab
        ^
State: q1
Stack: 1z
---------------------------------------------
Transition: delta(q1, b, 1) = (q2, _)
Step: 2
Input: ab
         ^
State: q2
Stack: z
---------------------------------------------
Transition: delta(q2, _, z) = (accept, _)
Step: 3
Input: ab
         ^
State: accept
Stack:
---------------------------------------------
Result: true
==================== END ====================
```

构造

1 pda括号匹配问题

栈保护符号z，扫到“(”就压一个1进栈，扫到“)”就从栈顶弹一个1，栈顶为z可以通过空转移到达接收状态

```
root@xkxk:~/workspace/fla# ./bin/fla pda/case.pda "(((())"
true
root@xkxk:~/workspace/fla# ./bin/fla pda/case.pda "((())) "
illegal input
root@xkxk:~/workspace/fla# ./bin/fla pda/case.pda "((()))"
true
root@xkxk:~/workspace/fla# ./bin/fla pda/case.pda "(()((())))"
false
root@xkxk:~/workspace/fla# ./bin/fla pda/case.pda "(()()())"
true
root@xkxk:~/workspace/fla# ./bin/fla pda/case.pda "()(a)(())"
illegal input
```

2 tm计算一进制数乘法

先把第一个纸带上的内容移动到第二条纸带，移动过程中如果出现类似ba的串说明输入串不合法，在第一个纸带上写上illegal input

（从串的末尾）向左扫描找到第一个a变为x，向右扫描每次将b变为y，并在第一条纸带上写c，直到所有b都用完，接着往回扫描，把y都变回b，然后找下一个a

```
root@xkxk:~/workspace/fla# ./bin/fla tm/case1.tm ab
c
root@xkxk:~/workspace/fla# ./bin/fla tm/case1.tm aab
cc
root@xkxk:~/workspace/fla# ./bin/fla tm/case1.tm aabb
cccc
root@xkxk:~/workspace/fla# ./bin/fla tm/case1.tm aaaabbb
cccccccccccc
root@xkxk:~/workspace/fla# ./bin/fla tm/case1.tm abbba
illegal input
```

3 tm识别完全平方数

注意到一个性质：完全平方数 1, 4, 9, 16, 25 ... 的差为 (1), 3, 5, 7, 9, ... 因此考虑在一个纸带m上存放 1,3,5,7,9，在另一个纸带n上累加，并与原数比较

初始化 m = 1, n = 1, 每轮更新 m += 2, n += m，可以一直保持 n = m^2，优化一下直接把原纸带上的1变为0，用来记录当前的数

```
root@xkxk:~/workspace/fla# ./bin/fla  tm/case2.tm 1
true
root@xkxk:~/workspace/fla# ./bin/fla tm/case2.tm 1
true
root@xkxk:~/workspace/fla# ./bin/fla tm/case2.tm 11
false
root@xkxk:~/workspace/fla# ./bin/fla tm/case2.tm 111
false
root@xkxk:~/workspace/fla# ./bin/fla tm/case2.tm 1111
true
root@xkxk:~/workspace/fla# ./bin/fla tm/case2.tm 111111
false
root@xkxk:~/workspace/fla# ./bin/fla tm/case2.tm 111111111
true
```

总结感想

1 PDA和TM的程序在框架上高度相似，只需要修改一少部分差异内容。这样的好处是写好的接口可复用性很强，坏处是没写好的地方需要双倍修改

2 需要处理很多细节的部分，比较费劲，不过讲义说的还是比较清楚的，就是需要先充分理解清楚要求再写代码。