实验报告

一、分析与设计思路

在本次实验中，要实现一个确定性、多条无限纸带的图灵机的模拟器。具体可以分为三个任务：解析输入、模拟图灵机运行、完成两个具体图灵机设计。

程序结构和设计

程序分为TM.h、TM.cpp和main.cpp三个文件，TM.h和TM.cpp中实现了图灵机的相关数据结构和功能，main.cpp中是主函数。

由于输入按照`turing [-v|--verbose] [-h|--help] <tm> <input>`的格式，在main函数中，首先使用数组`vector<string> params`保存传入的所有参数。main函数中实现了函数`bool parse(vector<string>& params, string& path, string& input)`，传入`params`数组，使用条件判断格式是否正确并将解析得到的路径和输入字符串传入main函数中定义的字符串`path`和`input`。由于实现较为简单，需要保证输入的参数顺序。当检测到`-v|--verbose`时，会将TM.cpp中定义的bool类型全局变量`VERBOSE`设为True，表示verbose模式。

TM.h中定义了类`Tape`，表示一条纸带，含有成员变量：`string content`表示纸带上所有有效内容；`int begin`表示起始符号的索引；`int end`表示纸带上最后一个有效符号的索引；`int ptr`表示纸带读写头的索引。定义了类`Transition`，表示一个转移函数的部分内容，转移函数格式为：<旧状态> <旧符号组> <新符号组> <方向组> <新状态>，`Transition`包含了 <旧符号组> <新符号组> <方向组> <新状态>，分别使用`string oldSym, string newSym, string newState, string dir`表示。为简便起见，这两个类成员全部为公有成员。

自动机类`TM`，包括私有成员：`vector<string> content`表示.tm文件的每一行的内容；`vector<string> Q, S, G, F`表示状态集、输入符号集、纸带符号集、终结状态集；`string q0`表示初始状态；`char B = '_'`表示空格符号，本实验中为`'_'`；`int N`表示纸带数；`unordered_map<string, vector<Transition>> delta`表示所有转移函数；`string curState`表示当前的状态；`vector<Tape> Tapes`表示所有纸带；`int steps`表示步数，用于verbose模式输出。其中所有转移函数用了一个`unordered_map`作为容器，第一个键值是转移函数中的旧状态，主要考虑到在模拟运行时便于根据当前状态查找对应的转移函数，关联到一个`vector<Transition>`，在模拟运行时遍历数组，根据纸带上的内容与旧符号组相比较获得对应的的新符号组、方向组和新状态。

TM.cpp中实现了模拟图灵机运行的相关方法。构造函数`TM(string path)`主要使用`getline()`读取对应路径的.tm文件，先将每一行内容依次存于数组`vector<string> content`中，因为.tm的格式选择这样逐行解析。

`void parse()`方法解析`content`中的内容，首先删除所有注释和空行，然后主要使用正则表达式进行匹配，根据格式定义了七个正则表达式和一个匹配空格的表达式，依次匹配，将结果存于对应的数据结构中。

`void check(string& input)`方法检查解析得到的自动机的七部分内容正确性以及输入字符串的合法性。考虑使用`set`进行检验。函数内定义了状态集、纸带符号集、输入符号集`set<string> allStates, allSymbols, inputSymbols`，首先遍历Q将所有状态，加入集合`allStates`，然后搜索初始状态和终结状态是否在集合中。之后遍历G检查是否符合条件并将每个符号放入集合`allSymbols`，然后遍历S检查每个符号是否符合要求且是否在G中，将这些符号放入集合`inputSymbols`。然后检查转移函数五元组是否合法。最后，检查输入字符串中各字符是否在集合`inputSymbols`中。若发现有错误，会报错退出，verbose模式会说明错误原因，但这些报错没有做定位功能。

`void turing(string& input)`方法模拟图灵机运行。首先初始化各纸带，然后在一个循环中，寻找当前状态、纸带符号组是否对应某一个转移函数。希望实现`*`定义为“若有其他更为具体的转移就走其他转移，否则走这条转移。用当前状态在`unordered_map<string, vector<Transition>> delta`中获得图灵机当前状态映射的转移函数数组，然后初始化一个`map<int, Transition> choice`，遍历当前状态的转移函数数组，将当前纸带符号组与某转移函数旧符号组比较，同时统计出该转移函数旧符号组`*`的个数，若比对成功，则放入`choice`备选。最终若`choice`为空，说明当前没有转移函数匹配，退出循环停机；若不为空，则`choice`第一个元素即为符合条件的`*`最少的转移函数，选择该转移函数并进行一步执行`execute()`。

`void TM::execute(Transition& next)`方法表示一步执行。首先根据转移函数改写读写头对应的纸带符号，然后根据方向将读写头加/减一表示右/左移。若移动后超出纸带有效符号范围，则需要在开头或结尾新增空格符，改变有效符号范围。最后检查是否有多余的空格符可以消除。

图灵机设计

任务一要求实现二进制数的循环右移操作。使用7个状态的单带图灵机，起始状态开始，读取0或1使用两个状态read0、read1表示，即可记录上一个比特，然后右移，根据状态写出上一个比特，并根据这一个比特改变状态即可。到达末尾时还需两个状态head0、head1记录最后一个比特，然后左移到开头写下即完成循环右移。

任务二要判断读入的全1串长度是否为完全平方数。由$n^2 = (n-1)^2 + 2(n-1) + 1$，使用两条带，第一条为输入符号，第二条作为计数器。第二条带由空开始，每次循环开始，读写头位于第二条带带的末尾。 然后读写头移回开头并再次回到末尾，再在末尾添加一个符号，这样完成了2(n-1)+1。 而第一条带上的读写头随第二个读写头向右而向右边。若两个读写头同时到达末尾则接受，输出true，否则输出false。

二、实验完成度

完成了所有任务以及verbose模式。

```
x:~/project-2022-stu/framework/bin$ ./turing --help
usage: turing [-v|--verbose] [-h|--help] <tm> <input>
```

示例程序

```
~/project-2022-stu/framework/bin$ ./turing ../programs/pa
lindrome_detector_2tapes.tm 100010001
true
~/project-2022-stu/framework/bin$ ./turing ../programs/pa
lindrome_detector_2tapes.tm 10001000
false
x:~/project-2022-stu/framework/bin$ ./turing ../programs/pa
lindrome_detector_2tapes.tm 100A1A001
illegal input
```

```
~/project-2022-stu/framework/bin$ ./turing --verbose ../p
rograms/palindrome_detector_2tapes.tm 100A1A001
Input: 100A1A001
==================== ERR ====================
error: 'A'was not declared in the set of input symbols
Input: 100A1A001
         ^
==================== END ====================
```

```
~/project-2022-stu/framework/bin$ ./turing --verbose ../programs/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
head0  : ^
Index1 : 0
Tape1  : _
head1  : ^
-----------------------------------------
Step   : 2
State  : cp
Index0 : 0 1 2 3 4 5 6
Tape0  : 1 0 0 1 0 0 1
head0  :   ^
Index1 : 0 1
Tape1  : 1 _
head1  :   ^
-----------------------------------------
```

```
-----------------------------------------
Step   : 28
State  : accept4
Index0 : 7 8 9 10
Tape0  : t r u _
head0  :       ^
Index1 : 1
Tape1  : _
head1  : ^
-----------------------------------------
Step   : 29
State  : halt_accept
Index0 : 7 8 9 10
Tape0  : t r u e
head0  :       ^
Index1 : 1
Tape1  : _
head1  : ^
-----------------------------------------
Result: true
==================== END ====================
```

case1

```
~/project-2022-stu/framework/bin$ ./turing -v ../programs/case1.tm 11101001
Input: 11101001
==================== RUN ====================
Step   : 0
State  : 0
Index0 : 0 1 2 3 4 5 6 7
Tape0  : 1 1 1 0 1 0 0 1
head0  : ^
-------------------------------------------
Step   : 1
State  : read1
Index0 : 1 2 3 4 5 6 7
Tape0  : 1 1 0 1 0 0 1
head0  : ^
-------------------------------------------
Step   : 2
State  : read1
Index0 : 1 2 3 4 5 6 7
Tape0  : 1 1 0 1 0 0 1
head0  :   ^
```

```
----------------------------------------
Step   : 17
State  : halt_accept
Index0 : 0 1 2 3 4 5 6 7
Tape0  : 1 1 1 1 0 1 0 0
head0  : ^
----------------------------------------
Result: 11110100
==================== END ====================
```

case2

```
                    ~/project-2022-stu/framework/bin$ ./turing ../programs/case2.tm ""
true
                    ~/project-2022-stu/framework/bin$ ./turing ../programs/case2.tm 1
true
                  x:~/project-2022-stu/framework/bin$ ./turing ../programs/case2.tm 11
false
                    ~/project-2022-stu/framework/bin$ ./turing ../programs/case2.tm 1111
true
                    ~/project-2022-stu/framework/bin$ ./turing ../programs/case2.tm 111111
false
                    ~/project-2022-stu/framework/bin$ ./turing ../programs/case2.tm 111111111
true
```

特性：会检验出.tm文件中不自洽的定义，比如终结状态中存在不在状态集中的状态，输入符号组不是纸带符号组子集，转移函数中未定义状态、符号等问题。

.tm文件中七个部分定义严格按照语法描述，比如逗号两边需无空格，等号两边需恰好有一个空格等，否则会报错。

转移函数中可以出现`*`与具体符号的冲突，匹配时优先选择具体符号。转移函数新旧符号组`*`的对应情况没有特别检查，不会报错。

```
                  x:~/project-2022-stu/framework/bin$ ./turing -v ../programs/error.tm 1001
syntax error
q0: 1 is not in States set.
```

```
                    ~/project-2022-stu/framework/bin$ ./turing -v ../programs/error.tm 1001
syntax error
Input symbol: _ is not illegal.
                  project-2022-stu/framework/bin$ ./turing -v ../programs/error.tm 1001
syntax error
Input symbol: m is not in Symbols set.
```

```
                    ~/project-2022-stu/framework/bin$ ./turing -v ../programs/error.tm 1001
syntax error
States in delta function: reject6 is not in States set.
```

三、实验中遇到的问题及解决方案

最初没考虑到转移函数中的`*`，直接使用`unordered_map<string, unordered_map<string, Transition>>`来保存转移函数，希望直接映射到位。最后改为`unordered_map<string, vector<Transition>>`，映射初始状态，遍历找到转移函数。

测试时发现.tm解析时直接报错，调试后发现是.tm文件在windows操作系统下修改后换行符编码问题。

四、总结感想与建议

感觉实验难点在于解析和verbose模式输出。可以降低这方面要求，实现更多图灵机方面的内容。

本次图灵机当且仅当没有符合的转移函数时停机，终结状态意义不明显。