PTC Turing Project

王明 161220124

一、分析与设计思路

1、分析

第一个任务是对于.tm 文件进行分析，需要设计一个图灵机结构将它解析并且存储下来。所使用的结构在我的实验中是一个叫 Turing 的结构体，具体的设计细节会在后面提及。设计好相应的结构后，就是将.tm 文件中的文本解析下来，具体的解析方法和实验说明中一样，主要用到的技术就是字符串解析。按行读取文本，然后有序遍历每一个字符，穷举可能出现的情况，对于每一种情况按照实验说明中相应的内容存储到设计好的图灵机结构中。

第二个任务是对于一个输入模拟图灵机的运行过程，需要设计一个描述每一步过程的结构体来表示每一步，在我的实验中是一个叫 ID 的结构体。该结构体里存储了每一步之后图灵机的状态，具体的设计细节会在后面提及。执行的过程就是读取输入的一个字符，去访问之前存下来的 Turing，结合当前的 ID 去确定下一个 ID 是什么，这里面主要是对于转换函数的访问和解析，以及对于通配符的解析。

第三个任务是就是自己设计图灵机，生成相应的.tm 文件。第一个是识别斐波拉契个 0 的字符串。我设计的方法就是，在别的纸带上按照斐波拉契数列生成斐波拉契个 0，然后比较输入的字符串中 0 的个数和生成的斐波拉契个 0 的长度，从而来判断是否是含有斐波拉契个 0。如果输入串长度等于对比串的长度，那么就 accept。如果输入串的长度大于对比串的长度，那么就更新对比串的长度为下一个斐波拉契数的长度。如果输入串的长度小于对比串的长度，那么就 reject。第二个是判断字符串中是否有 WW 的重复结构。我的方法是从头往后遍历，每次按照读头所在的地方将字符串分为前后两个部分，然后来比较这两个部分是否相同，在读头从左到右的过程中只要有一个满足 WW 结构就输出 accept，如果读头走到最后还没有出现前后相同的字符串，就输出 reject。

2、设计思路

（1）数据结构的设计

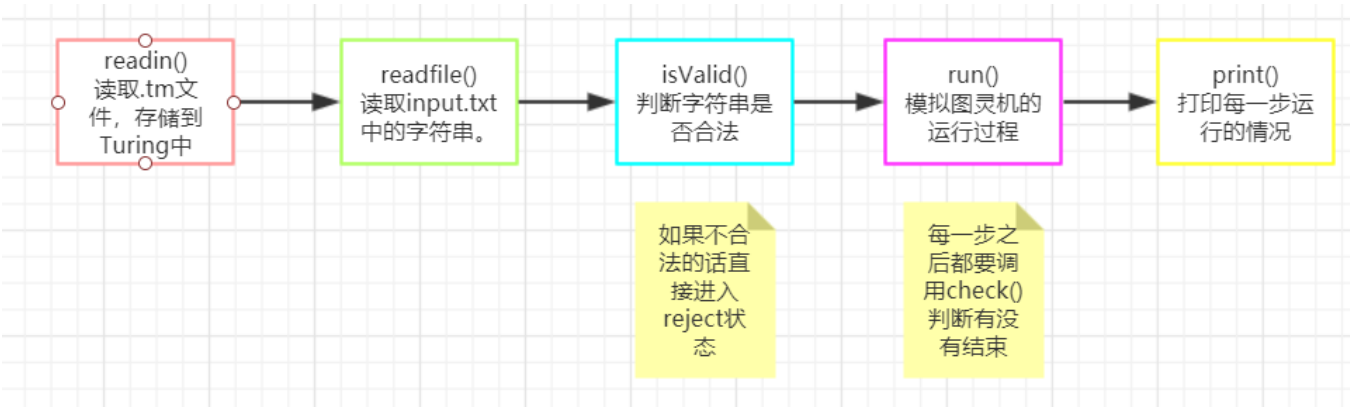
存储图灵机的结构体 Turing

```
15  struct Transfunc {
16      string oldstate;          //当前状态
17      vector<char> oldstr;      //当前字符
18      vector<char> newstr;      //新产生的字符
19      vector<char> dir;         //读头的下一步运行方向
20      string newstate;          //下一步状态
21  };
22
23  struct Turing {
24      vector<string> stateSet;  //状态集
25      vector<char> inputSet;    //输入字符集
26      vector<char> tapeSet;     //纸带字符集
27      string q0;                //起始状态
28      char space;               //空格符
29      vector<string> finalSet;  //终止状态集
30      int tapenum;              //纸带个数
31      vector<Transfunc *> tf;   //转换函数列表
32  };
```

存储每一步过程的结构体 ID

```
34  struct Tape {
35      vector<int> index;      //纸带每一个位置的编号
36      vector<char> str;       //纸带对应编号上的字符
37      int head;               //读头目前所处的位置对应的编号
38  };
39
40  struct ID {
41      int step;               //当前的步数
42      string state;           //当前所处的状态
43      vector<Tape *> tapes;   //当前每一条纸带的状态
44  };
45
```

（2）算法流程的设计



（3）核心函数说明

注：其实我自己觉得实验的核心就是数据结构的设计，下面介绍的函数虽然有些代码可能长，但是运行逻辑都是很简单的，因此就省去具体的代码注解，仅说明函数的功能。

readin()：

该函数用来解析.tm 文件，其实就是字符串解析，按行读取，再按照实验说明中的对于每种情况进行解析就可以了。

isValid()：

遍历输入字符串的每一个字符，查看是否都在图灵机规定的输入字符集中。

run()：

主要由两个步骤组成。一个是 nextid()，就是根据当前的 ID，再读取图灵机的转换函数进行匹配，确定下一个 ID。另一个是 check()，图灵机每走一步都要执行这个步骤，目的是决定是 accept，还是 reject，还是继续执行下去。

print()：

该函数就是将 ID 上的内容按照实验要求写入到文件中。

二、实验完成度

实验完成度为 100%，完成了实验说明中的所有的任务。

三、实验中遇到的问题及解决方案

1、一开始是在 Window 平台下写的，完成之后迁移到 Linux 系统下，但是发现有些函数在两个系统下不太一样。比如读取文件得到输入串后，输出输入串长度时，Linux 下好像会比 Window 下多出一个，我自己猜测是把字符串最后的'\0'也计算了进去。解决方法就是对于 Linux 下，都把读取字符串的长度进行减 1 的操作。

2、第二个是对于纸带的打印问题，这个地方主要有两个问题，一个是纸带的编号和纸带的字符对应出错，第二个问题就是在打印的时候对于纸带是否输出空字符的判断比较繁琐。第一个问题的解决方法就是同步，对于编号列表和纸带内容列表进行同时操作，同时进行增删。第二个问题就是根据约束条件确定左右边界，这样比直接一个一个判断空字符是否打印要快捷许多。

3、第三个是自己设计.tm 时会遇到很多问题，总是莫名其妙的出错。后来我在设计时使用了一些技巧克服困难。第一个技巧是每一步操作后都记得将读头移到最左边或最右边，以防止下一次操作时忘记还原读头位置。第二个技巧是不要同时考虑 reject 和 accept 的情况，首先默认输入串是 accept 的，然后设计一套动作去识别它，完成之后，再考虑这个过程中，哪个状态会有可能进入 reject 状态，将这些可能的转换函数补充上去。第三个技巧就是用状态数来减缓设计的困难度，这一点我非常有感触，可以多设计一些状态，哪怕是无用的状态，但是可以帮助自己更好地去理清思路，而不是抽象地在脑海中去模拟走的步骤。第四个技巧是就是要学会迁移，有些基本操作，比如读头从左移到右、比较两个纸带上的字符串是否相等，这些操作基本每个图灵机都要用到，因此要学会复用，就能提高效率。

四、编译运行说明

我的源文件有 2 个，一个是 PTC.cpp，一个是 pch.h。PTC.cpp 中就是一个 main 函数，启动程序。pch.h 中定义了 TM 这个类，实现了实验所需要的所有函数。

编译方法是：g++ PTC.cpp –o turing
运行方法是：./turing case1 或者./turing case2

注：每次运行之前请手动删除每个 case 文件夹下的 console.txt 和 result.txt。

五、演示和总结感想

1、识别斐波拉契数列个 0
输入：

```
 1 0123abcd
 2
 3 0
 4 00
 5 000
 6 0000
 7 00000
 8 000000
 9 0000000
10 00000000
11 000000000
12 0000000000
```

输出:

```
 1 Error
 2 False
 3 True
 4 True
 5 True
 6 False
 7 True
 8 False
 9 False
10 True
11 False
12 False
```

中间过程:

```
  1 Input: 0123abcd
  2 ==================== ERR ====================
  3 The input 0123abcd is illegal
  4 ==================== END ====================
  5 Input:
  6 ==================== RUN ====================
  7 Step    :       0
  8 Index0  :       0
  9 Tape0   :       _
 10 Head0   :       ^
 11 Index1  :       0
 12 Tape1   :       _
 13 Head1   :       ^
 14 Index2  :       0
 15 Tape2   :       _
 16 Head2   :       ^
 17 Index3  :       0
 18 Tape3   :       _
 19 Head3   :       ^
 20 State   :       0
 21 ------------------------------------------------------
 22 Step    :       1
 23 Index0  :       0
 24 Tape0   :       _
 25 Head0   :       ^
 26 Index1  :       0
 27 Tape1   :       _
 28 Head1   :       ^
 29 Index2  :       0
```

2、识别字符串是否满足 WW 结构

输入:

```
12351asasd

0
0101
1010
01010111
11101110
110101011
0101101011
```

输出:

```
Error
False
False
True
True
False
True
False
True
```

中间过程:

```
Input: 12351asasd
==================== ERR ====================
The input 12351asasd is illegal
==================== END ====================
Input:
==================== RUN ====================
Step    :       0
Index0  :       0
Tape0   :       _
Head0   :       ^
Index1  :       0
Tape1   :       _
Head1   :       ^
Index2  :       0
Tape2   :       _
Head2   :       ^
State   :       0
-----------------------------------------------------
Step    :       1
Index0  :       0
Tape0   :       _
Head0   :       ^
Index1  :       0
Tape1   :       _
Head1   :       ^
Index2  :       0
Tape2   :       _
Head2   :       ^
State   :       reject_mh0
-----------------------------------------------------
```

3、总结:

读取图灵机并且模拟其运行过程难度其实不大，主要是要细心和耐心。到最后，发现自己设计一个图灵机，写出一个.tm 文件才是最难的事情，因为需要考虑所有的可能性，各种琐碎但是重要的步骤都不能遗漏。说道这儿必须非常感谢助教和老师的要求是使用多带，如果强制使用单带的话，我可能当场就去世了。做完实验我觉得自己的收获还是很大的，因为这门课本身非常的理论，作业也是非常的理论，但是这个实验就让我真切感受到图灵机真实运行一个算法，完成一个功能的能力，亲自动手实现图灵机也是加深了对于图灵机的理解，有一种自豪感。最后就是感谢助教对我的帮助和指导。

六、对课程和项目的建议

课程的话就是感觉还是太偏理论了，建议在上课时能够再配一些这些自动机实际应用的例子，或者讲一些关于这些自动机的有趣的故事，这样有利于提升课堂兴趣，提高注意力。然后就是强烈建议作业真的可以考虑降低一下难度，那种做 3 个小时做不出一题的无力感和挫败感还是相当难受的。项目五星好评，难度适中，所需要花的时间也适中，也有助于提升对于课程知识的理解，此外 ddl 也是非常人性化。