作业 **1**：本作业的目的是练习有限状态机的使用。

作业要求： 从下面 **2** 个题目中选择其 **1** 完成代码的编写，选择的语言可以是 **C++**或者是 **Java** 均可。

任务 **1**：机器人

Figure 1 机器人 John

Here is John, as you may recognize him from Futurama. John would like to turn on and off, walk, run, raise and lower its arms, turn its head, and talk. There are many other features left to be desired, like ducking, jumping, repairing itself, reacting to environments, give it emotions, etc… As you can see, the list goes on and this is quite normal in a FSM. Keep in mind however, as the term **finite** implies you will need to limit the scope of the problem to a finite set of entities. We should construct a table listing all the possible events and states associated with them.

Event	**State**
turnOn	Activated
turnOff	Deactivated (Idle)
stop	Stopped
walk	Walking
run	Running
raiseLeftArm	LeftArmRaised
lowerLeftArm	LeftArmLowered
lowerLeftArm	LeftArmLowered
raiseRightArm	RightArmRaised
lowerRightArm	RightArmLowered
turnHead	HeadTurned(direction)
speak	Talking(text)

Table: 1 Small list of known events with John

问题的详细说明：

1）States do not have multiple roles. Every time John is turned on, the same flow of direction will always occur. If at any time in your projects there is a breech in that flow, it would need to be fixed. This is one of the problems with FSMs because sooner or later someone is going to pick up the pattern. Ways around this would be to add more events to a given situation or randomize the actions performed.

2）Secondly, states are unique. No two states should have the same reaction to a specified event. For example, having John "speak" and "shout" are synonymous in terms of speech; therefore we eliminate one while keeping the other and supplement arguments to dictate how a certain text is pronounced (be it calm, yelling, stutter, etc…).

3）Thirdly, state transitions do not normally branch. By branch I mean through the use of conditional statements. For example,John has a set of movement options. Such movements can be decided via IF/THEN statements, however it is best to avoid that since the purpose behind states is to replace the need for IF/THEN statements with events. Although states are flexible and do allow such conditional operations, it is always best to avoid them. It reduces complexity and lessens the chance for logic errors.

4）A state table is nothing more than a table listing all the states, the actions to get to them, and the reaction performed by them. So the state table illustrates to us what and when can John perform certain actions.

#	**Event**	**Actions**	**Comment**	**Caused By**	**Will Effect**
1	turnOn	bootUp	Bender is turning on		2-10
2	bootUp	Activity	Allow various activities	1	3-8
3	walk	goWalk	John will begin to walk		
4	run	goRun	John will begin to run		
5	talk	say(text)	John will say "text"		
6	turnHead	turningHead	John rotates head		
7	raiseArm	raisingArm	John raises arm (left or right)		
8	lowerArm	laweringArm	John lowers arm (left or right)		
9	stop	powerDown	John stops current action		3-8
10	turnOff	shutDown	John will shut off		1-9

Table: 2. State Table for Random Scenarios

As you see, defining a complete FSM is an extremely long process and we have not even done any coding yet! The state table is left incomplete as the number of events and actions are quite large, but there should be enough presented to illustrate the idea.

Putting it all together, the state diagram helps illustrate what the functionality is like and the state table defines the types of inputs to those functions and the expected outputs. So, with a complete state diagram and state table, you have the **blueprints** to develop your FSM.

解决方案提示：

There are two major classes.

1) State: The state class is responsible for setting transitions, specifying actions, and pointing to the right function that represents the action. In this code, a state is allowed to support up to 6 incoming and outgoing events. This should help you practice with designing much larger FSMs.

2) FSM: The FSM class is a collection of states. Its sole purpose is to function as a controller between states and execute the necessary state actions based on the events passed to it.

You can make use of any conventional programming style: switch statements and IF/THEN expressions; but we recommend that everything in the FSM is controlled by sending the correct events down the pipeline. After that, the FSM will choose the correct course based on how you configured your states.

When developing FSMs, you should expect to see an even greater list of events and states to support.

任务 2：词法分析器

任务定义：用程序识别输入的字符串

- Operators

  + - * / = > >= < <= == !=
- Parentheses and Semicolumn

  ( ) ;
- Keywords

  if, then, else, endif, print, newline
- Integers An integer is any string made of decimal digits: 0...9. Examples:

  129, 000234, 1789000
- Identifiers An identifier is any string that starts with a letter and then continues with any combination of letters and decimal digits. Examples:

  var1, x, y, z34r4
- Quoted Strings A quoted string is any string that appears between two quotes. Example:

  “hello, I am a quoted string”
  - quoted string should not include the quotes and it should not occupy more than one lines.

- A quoted string can include spaces and tabs.
- Comments A comment is any string that starts with the characters “//” and ends at the end of a line. Example:

  // this is a comment

  - A comment can include spaces and tabs.

运行示例

输入：

```
==
!=
(
)
;
if
then
else
endif
print
newline
0000000000000345
id1
id2
id1
"hello, how are you?"
```

输出：

```
operator: +
operator: -
operator: *
operator: /
operator: =
operator: >
operator: >=
operator: <
operator: <=
operator: ==
operator: !=
parenthesis: (
parenthesis: )
semicolumn: ;
keyword: if
keyword: then
keyword: else
keyword: endif
```

```
keyword: print
keyword: newline
integer: 345
New Identifier "id1"
New Identifier "id2"
Identifier "id1" already in symbol table
quoted string: hello, how are you?
```