

## Turing Machine

Generated by Doxygen 1.8.20



<b>1 Class Index</b>	<b>1</b>
1.1 Class List	1
<b>2 Class Documentation</b>	<b>3</b>
2.1 Tape Class Reference	3
2.1.1 Constructor & Destructor Documentation	3
2.1.1.1 Tape() [1/3]	3
2.1.1.2 Tape() [2/3]	4
2.1.1.3 Tape() [3/3]	5
2.1.1.4 ~Tape()	5
2.1.2 Member Function Documentation	5
2.1.2.1 clear()	5
2.1.2.2 getIndex()	6
2.1.2.3 getSize()	6
2.1.2.4 isEmpty()	6
2.1.2.5 moveLeft()	6
2.1.2.6 moveRight()	7
2.1.2.7 printTape()	7
2.1.2.8 readSymbol()	7
2.1.2.9 rewind()	7
2.1.2.10 setTape()	7
2.1.2.11 writeBlank()	8
2.1.2.12 writeSymbol()	8
2.2 TuringMachine Class Reference	8
2.2.1 Constructor & Destructor Documentation	9
2.2.1.1 TuringMachine() [1/2]	9
2.2.1.2 TuringMachine() [2/2]	9
2.2.1.3 ~TuringMachine()	10
2.2.2 Member Function Documentation	10
2.2.2.1 getSymbols()	10
2.2.2.2 getTape()	10
2.2.2.3 patternMatch()	10
2.2.2.4 reset()	11
2.2.2.5 setSymbols()	11
2.2.2.6 setTape()	11
2.2.2.7 setupCheck()	12
<b>Index</b>	<b>13</b>



# Chapter 1

## Class Index

### 1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Tape</a> . . . . .	<a href="#">3</a>
<a href="#">TuringMachine</a> . . . . .	<a href="#">8</a>



## Chapter 2

# Class Documentation

## 2.1 Tape Class Reference

### Public Member Functions

- [Tape](#) ()
- [Tape](#) (std::vector< std::string > v)
- [Tape](#) (const [Tape](#) &toCopy)
- [~Tape](#) ()
- int [getSize](#) () const
- int [getIndex](#) () const
- void [setTape](#) ([Tape](#) toCopy)
- bool [isEmpty](#) () const
- bool [moveRight](#) ()
- bool [moveLeft](#) ()
- bool [writeBlank](#) ()
- bool [writeSymbol](#) (std::string s)
- std::string [readSymbol](#) () const
- void [clear](#) ()
- void [printTape](#) () const
- void [rewind](#) ()

### 2.1.1 Constructor & Destructor Documentation

#### 2.1.1.1 [Tape\(\)](#) [1/3]

`Tape::Tape ( )`

Default constructor: creates an empty tape (i.e., no cell).

#### Returns

Sets data fields appropriately.

### 2.1.1.2 **Tape()** [2/3]

```
Tape::Tape (  
    std::vector< std::string > v )
```

Parameterized constructor: creates a tape with symbols from v (one cell per symbol).



**Parameters**

<i>v</i>	A vector containing symbols.
----------	------------------------------

**Returns**

Sets data fields appropriately.

**2.1.1.3 Tape() [3/3]**

```
Tape::Tape (
    const Tape & toCopy )
```

Copy constructor: performs a deep copy of toCopy. If toCopy is empty, then the tape is also empty.

**Parameters**

<i>toCopy</i>	A tape to copy.
---------------	-----------------

**Returns**

Sets data fields appropriately.

**2.1.1.4 ~Tape()**

```
Tape::~~Tape ( )
```

Destructor: deallocates memory as appropriate.

**2.1.2 Member Function Documentation****2.1.2.1 clear()**

```
void Tape::clear ( )
```

Clears the tape so that it is now empty.

**Returns**

Sets data fields appropriately.

#### 2.1.2.2 getIndex()

```
int Tape::getIndex ( ) const
```

Returns the current tape location, i.e., cell. If the tape is empty, then return -1.

##### Returns

An integer denoting the current cell in the tape.

#### 2.1.2.3 getSize()

```
int Tape::getSize ( ) const
```

Gets the current number of cells in the tape.

##### Returns

The integer number of cells currently in the tape.

#### 2.1.2.4 isEmpty()

```
bool Tape::isEmpty ( ) const
```

Sees whether this tape is empty.

##### Returns

True if the tape is empty, or false if not.

#### 2.1.2.5 moveLeft()

```
bool Tape::moveLeft ( )
```

Move the machine head left by one cell if the current location is not the beginning of the tape.

##### Returns

True if the tape can be moved to the left (i.e., there is at least one more cell to the left of the tape), or false if not.

### 2.1.2.6 moveRight()

```
bool Tape::moveRight ( )
```

Move the machine head right by one cell if the current cell is not the end of the tape.

#### Returns

True if the tape can be moved to the right (i.e., there is at least one more cell to the right of the tape), or false if not.

### 2.1.2.7 printTape()

```
void Tape::printTape ( ) const
```

Prints the cells on the tape from beginning to end.

#### Returns

Content of tape to standard output. NOTE: You do not need to test this method.

### 2.1.2.8 readSymbol()

```
std::string Tape::readSymbol ( ) const
```

Reads the symbol at the current cell in the tape if said cell is valid.

#### Returns

True if the tape can be read at the current cell, or false if not.

### 2.1.2.9 rewind()

```
void Tape::rewind ( )
```

Rewinds the tape to the very beginning (i.e., first cell) if the tape is not empty.

#### Returns

Sets data fields appropriately.

### 2.1.2.10 setTape()

```
void Tape::setTape (
    Tape toCopy )
```

Sets the symbols on the tape according to toCopy and overwrites existing symbols, if any. If toCopy is empty, then the tape is also reset to having no symbol.

**Parameters**

<i>toCopy</i>	A tape to copy.
---------------	-----------------

**Returns**

Sets data fields appropriately.

**2.1.2.11 writeBlank()**

```
bool Tape::writeBlank ( )
```

Writes a blank symbol (" ") to the current cell in the tape if said cell is valid.

**Returns**

True if the tape can be written at the current cell, or false if not.

**2.1.2.12 writeSymbol()**

```
bool Tape::writeSymbol (
    std::string s )
```

Writes the symbol specified by s to the current cell in the tape if said cell is valid.

**Parameters**

<i>s</i>	A symbol.
----------	-----------

**Returns**

True if the tape can be written at the current cell, or false if not.

The documentation for this class was generated from the following file:

- Tape.hpp

## 2.2 TuringMachine Class Reference

**Public Member Functions**

- [TuringMachine](#) ( )

- [TuringMachine](#) ([Tape](#) toCopy, const std::vector< std::string > &p)
- [~TuringMachine](#) ()
- std::vector< std::string > [getSymbols](#) () const
- [Tape](#) [getTape](#) ()
- bool [setSymbols](#) (std::vector< std::string > p)
- bool [setTape](#) ([Tape](#) &toCopy)
- bool [setupCheck](#) ()
- void [reset](#) ()
- bool [patternMatch](#) ()

## 2.2.1 Constructor & Destructor Documentation

### 2.2.1.1 TuringMachine() [1/2]

```
TuringMachine::TuringMachine ( )
```

Default constructor: creates an empty Turing machine.

#### Returns

Sets data fields appropriately.

### 2.2.1.2 TuringMachine() [2/2]

```
TuringMachine::TuringMachine (
    Tape toCopy,
    const std::vector< std::string > & p )
```

Parameterized constructor: creates a Turing machine with tape whose content is identical to toCopy and whose legal symbols are as specified by p.

#### Parameters

<i>toCopy</i>	A tape to copy content from.
<i>p</i>	A vector containing legal symbols.

#### Returns

Sets data fields appropriately.

### 2.2.1.3 ~TuringMachine()

```
TuringMachine::~~TuringMachine ( )
```

Destructor: deallocates memory as appropriate.

## 2.2.2 Member Function Documentation

### 2.2.2.1 getSymbols()

```
std::vector<std::string> TuringMachine::getSymbols ( ) const
```

Gets the legal symbols that can be written to the tape.

#### Returns

A vector containing all the legal symbols in the Turing machine.

### 2.2.2.2 getTape()

```
Tape TuringMachine::getTape ( )
```

Gets the tape from the Turing machine.

#### Returns

The appropriate tape.

### 2.2.2.3 patternMatch()

```
bool TuringMachine::patternMatch ( )
```

Checks to see if content of the tape matches the pattern:  $a^i b^j c^j$ , where  $i$  and  $j$  are integers greater than or equal to 1. Also checks if the tape contains illegal symbols.

#### Returns

True if the pattern matches and there are no illegal symbols, or false otherwise.

#### 2.2.2.4 reset()

```
void TuringMachine::reset ( )
```

Resets the Turing machine to empty tape and no symbol list.

##### Returns

Set appropriate data fields.

#### 2.2.2.5 setSymbols()

```
bool TuringMachine::setSymbols (
    std::vector< std::string > p )
```

Sets the legal symbols that can be written to the tape, overwriting old symbols if applicable.

##### Parameters

<i>p</i>	A vector containing legal symbols.
----------	------------------------------------

##### Returns

True if the symbols have been set, or false otherwise.

#### 2.2.2.6 setTape()

```
bool TuringMachine::setTape (
    Tape & toCopy )
```

Sets the tape content to be as specified by toCopy, overwriting old content if applicable.

##### Parameters

<i>toCopyA</i>	tape to copy content from.
----------------	----------------------------

##### Returns

True if the tape have been set, or false otherwise.

### 2.2.2.7 `setupCheck()`

```
bool TuringMachine::setupCheck ( )
```

This method is specific to the pattern we're checking. For our purpose, this method simply checks that there are exactly 3 legal, no duplicate symbols.

#### Returns

True if the setup is correct, or false otherwise.

The documentation for this class was generated from the following file:

- TuringMachine.hpp



# Index

- ~Tape
  - Tape, [5](#)
- ~TuringMachine
  - TuringMachine, [9](#)
- clear
  - Tape, [5](#)
- getIndex
  - Tape, [5](#)
- getSize
  - Tape, [6](#)
- getSymbols
  - TuringMachine, [10](#)
- getTape
  - TuringMachine, [10](#)
- isEmpty
  - Tape, [6](#)
- moveLeft
  - Tape, [6](#)
- moveRight
  - Tape, [6](#)
- patternMatch
  - TuringMachine, [10](#)
- printTape
  - Tape, [7](#)
- readSymbol
  - Tape, [7](#)
- reset
  - TuringMachine, [10](#)
- rewind
  - Tape, [7](#)
- setSymbols
  - TuringMachine, [11](#)
- setTape
  - Tape, [7](#)
  - TuringMachine, [11](#)
- setupCheck
  - TuringMachine, [11](#)
- Tape, [3](#)
  - ~Tape, [5](#)
  - clear, [5](#)
  - getIndex, [5](#)
  - getSize, [6](#)
  - isEmpty, [6](#)
  - moveLeft, [6](#)
  - moveRight, [6](#)
  - printTape, [7](#)
  - readSymbol, [7](#)
  - rewind, [7](#)
  - setTape, [7](#)
  - Tape, [3](#), [5](#)
  - writeBlank, [8](#)
  - writeSymbol, [8](#)
- TuringMachine, [8](#)
  - ~TuringMachine, [9](#)
  - getSymbols, [10](#)
  - getTape, [10](#)
  - patternMatch, [10](#)
  - reset, [10](#)
  - setSymbols, [11](#)
  - setTape, [11](#)
  - setupCheck, [11](#)
  - TuringMachine, [9](#)
- writeBlank
  - Tape, [8](#)
- writeSymbol
  - Tape, [8](#)