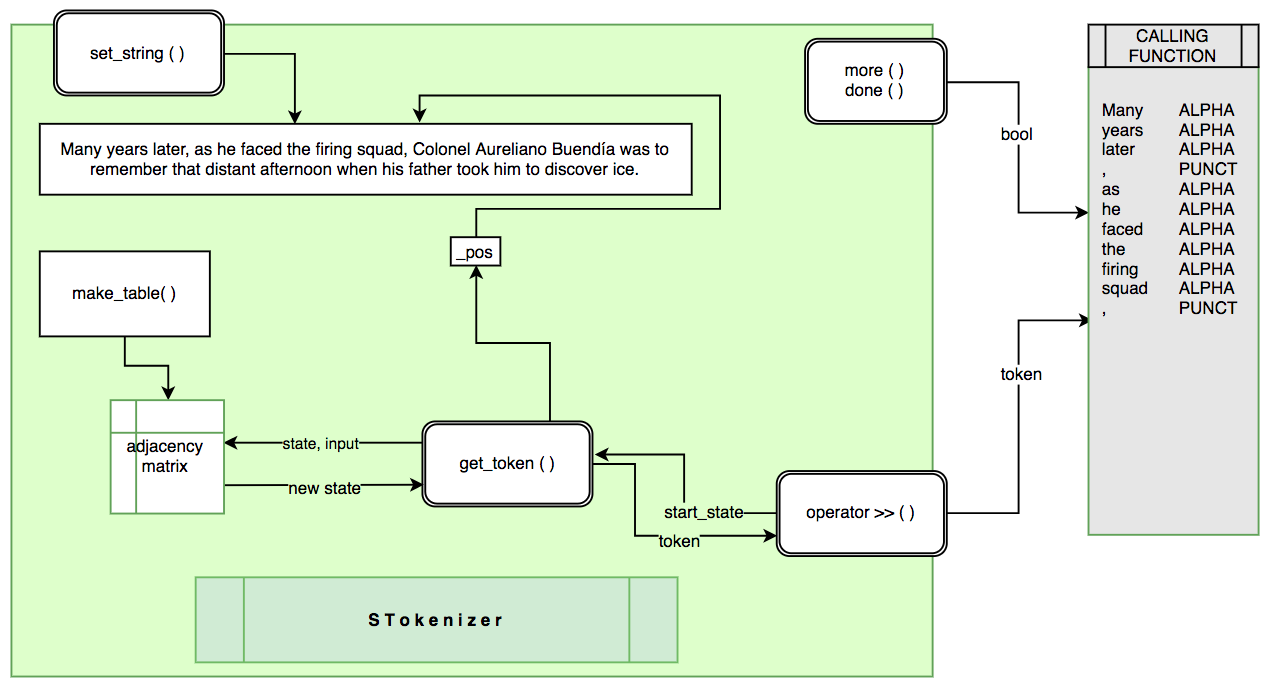
String Tokenizer

[Submit Assignment](https://canvas.pasadena.edu/courses/1088464/assignments/7994340)

* **Due** Monday by 11:59pm

* **Points** 10
* **Submitting** a text entry box, a website url, a media recording, or a file upload
* **Available** until Sep 17 at 11:59pm

**STokenizer Class**



**Purpose:**

The String Tokenizer (STokenizer) returns a single token (via the extraction operator) from a string according to the rules set by its internal state machine. For  now, the state machine is hard coded inside the STokenizer class, but it should not be difficult to allow the user to set or change the state machine from the calling entity.

The extraction operator of the STokenizer object is repeatedly called to grab the  next token in the string.

When no more token can be found in the string, more() and done() will return false and true respectively.

**Token Class:**

The token class is a package that is used by the STokenizer to send strings and types of tokens extracted from the input buffer to the calling entity. When a calling function calls the extract operator of the STokenizer, a Token object is returned. The Token class provides simple ways to interact with and report about these token strings.

class Token

{

public:

Token();

Token(string str, int type);

friend ostream& operator <<(ostream& outs, const Token& t);

int type();

string type\_string();

string token\_str();

private:

string \_token;

int \_type;

};

**STokenizer Class:**

* Receives an input string via the CTOR or the set\_string function.
* Maintains the current position in this string.
* Uses a state machine (a graph implemented as an Adjacency Matrix) to recognize valid tokens
* \_more() and \_done() report existance of more tokens in the input string.
  + Test \_pos<=strlen(\_buffer)
* Extraction operator grabs the next token, stores the string and its type in a Token object.
  + Returns an STokenizer reference.

class STokenizer

{

public:

STokenizer();

STokenizer(char str[]);

bool done(); //true: there are no more tokens

bool more(); //true: there are more tokens

//-- big three --

//---------------

//extract one token (very similar to the way cin >> works)

friend STokenizer& operator >> (STokenizer& s, Token& t);

//set a new string as the input string

void set\_string(char str[]);

private:

//create table for all the tokens we will recognize

// (e.g. doubles, words, etc.)

void make\_table(int \_table[][MAX\_COLUMNS]);

//extract the longest string that match

// one of the acceptable token types

bool get\_token(int start\_state, string& token);

//---------------------------------

char \_buffer[MAX\_BUFFER]; //input string

int \_pos; //current position in the string

static int \_table[MAX\_ROWS][MAX\_COLUMNS];

};

In the stokenizer.cpp, you must reintroduce the \_table static member variable once again:

int STokenizer::\_table[MAX\_ROWS][MAX\_COLUMNS];

**Testing:**

char s[] = "it was the night of october 17th. pi was still 3.14.";

STokenizer stk(s);

Token t;

//The all too familiar golden while loop:

stk>>t;

while(stk.more()){

//process token here...

cout<<setw(10)<<t.type\_string()<<setw(10)<<t<<endl;

t = Token();

stk>>t;

}

**Output:**

ALPHA |it|

SPACE | |

ALPHA |was|

SPACE | |

ALPHA |the|

SPACE | |

ALPHA |night|

SPACE | |

ALPHA |of|

SPACE | |

ALPHA |october|

SPACE | |

NUMBER |17|

ALPHA |th|

UNKNOWN |.|

SPACE | |

ALPHA |pi|

SPACE | |

ALPHA |was|

SPACE | |

ALPHA |still|

SPACE | |

NUMBER |3.14|

UNKNOWN |.|