COSC 385.001 – Automata
Spring 20105
Project # 1

Due: Monday, March 07, 2016, 03:00 PM

Problem - 10

Student: Mateus Meruvia
Instructor: Vojislav Stojkovic

Points:

Problem - 10 - Description

Write a program in your favorite programming language to implement a finite automaton that accepts only comments.

Algorithm

```
//
//  main.cpp
//  is_comment
//
//  Created by Mateus Mesturini Meruvia on 2/19/16.
//  Copyright © 2016 Mateus Mesturini Meruvia. All rights reserved.
//
#include <iostream>
#include <string>

bool is_identifier(std::string str){
    int n,i, current=0;
    n = str.length();

    if(str[1] == '/'){ // comments of the "//" type

        for(i=0; i<n; i++){

            if (i == 0 || i == 1) { // two first postion are "/"
                if( str[i] == '/' ){ // check if is a slash
                    if (current == i) { // check if the current position is equal to
                                                the 'i' position, if not skips
                        current++;
                    }
                }
            }

            if (i > 1) { // from the third position can be any char
                if( str[i] >= 32 && str[i] <= 126 ){ // check if is a char from the
                                                                ascII table
                    if (current == i) { // check if the current position is equal to
                                                the 'i' position, if not skips
                        current++;
                    }
                }
            }

            if(current == i) { // if 'current' and 'i' have the same value at this
                                                        point it means that
                return false;  // none of the previous conditions were fulfilled, so
                                        its not a integer, returns FALSE
            }

        }
        return true;
    }

    if(str[1] == '*'){ // comments of the "/*  */" type

        for(i=0; i<n; i++){

            if (i == 0) { // first position has to be a  "/"
                if( str[i] == '/' ){ // check if is a slash
                    if (current == i) { // check if the current position is equal to
                                                the 'i' position, if not skips
                        current++;
                    }
                }
            }

            if (i == 1) { // first position has to be a  "*"
                if( str[i] == '*' ){ // check if is a slash
```

```
                if (current == i) { // check if the current position is equal to
                                                    the 'i' position, if not skips
                    current++;
                }
            }
        }

        if (i > 1 && i < n-2) { // from the third position can be any char until
                                                                the position n-2
            if( str[i] >= 32 && str[i] <= 126 ){ // check if is a char from the
                                                                    ascII table
                if (current == i) { // check if the current position is equal to
                                                    the 'i' position, if not skips
                    current++;
                }
            }
        }

        if (i == n-2) { // first position has to be a  "*"
            if( str[i] == '*' ){ // check if is a slash
                if (current == i) { // check if the current position is equal to
                                                    the 'i' position, if not skips
                    current++;
                }
            }
        }
        if (i == n-1) { // first position has to be a  "/"
            if( str[i] == '/' ){ // check if is a slash
                if (current == i) { // check if the current position is equal to
                                                    the 'i' position, if not skips
                    current++;
                }
            }
        }


        if(current == i) { // if 'current' and 'i' have the same value at this
                                                                point it means that
            return false;  // none of the previous conditions were fulfilled, so
                                                    its not a integer, returns FALSE
        }
    }
    return true;
}
return false;
}

int main(int argc, const char * argv[]) {

    std::string comment;

    while (1) {

        std::cout << "Enter comment ";
        std::getline (std::cin,comment); // reads a string from the standard input


        if(is_identifier(comment)==true){
            printf("ACCEPTED\n\n");
        }else{
            printf("NOT ACCEPTED\n\n");}
    }
    return 0;
}
```

Explanations

My approach to solve this problem was dividing it in parts. This way I could have a better look at each part individually. For this project I used the definition of comments from C++.

#1 Part - Definition of the language

There are two kinds of comments in C++:

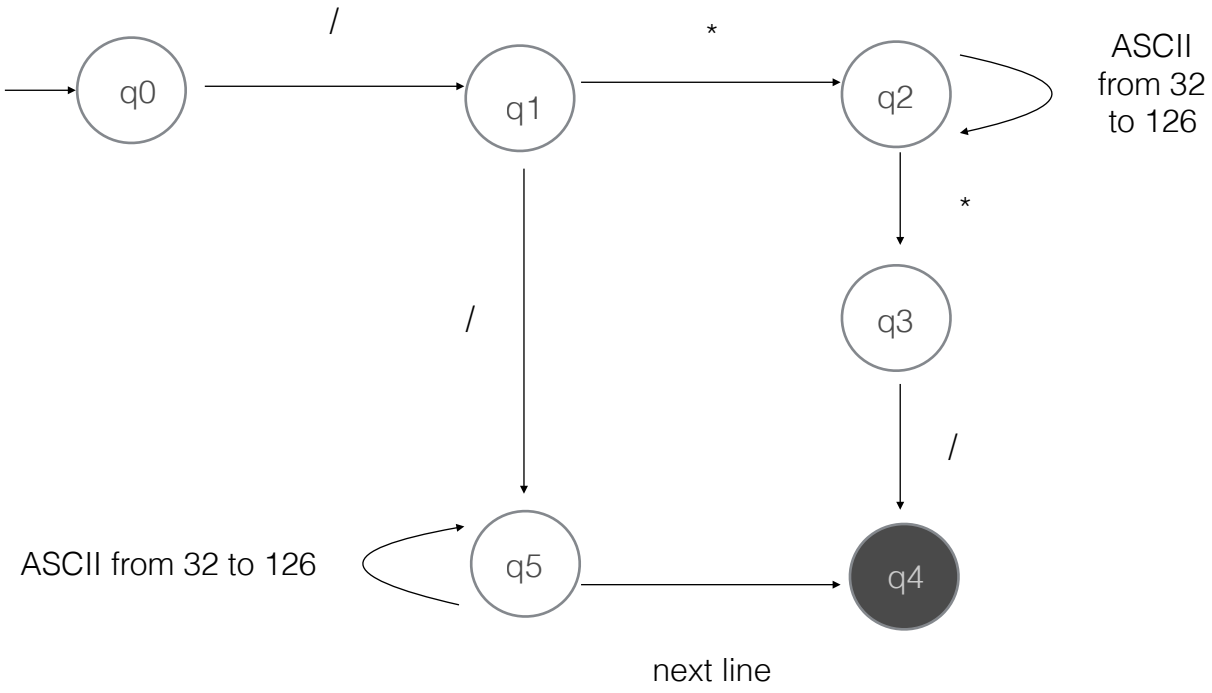
1. Single line comment: anywhere in the code where two consecutive (“//“) slashes are read by the compiler, it recognizes as a single line comment (if out of a comment). The comment section ends with a next line.
2. Multiple lines comment: anywhere in the code where a slash and a asterisk are read side by side by the compiler, it recognizes as the beginning of a multiple line comment (if out of a comment). The comment section ends with an asterisk (“*”) followed by a slash (“/")

#2 Part - Finite Automata

I first drew the finite automata for the proposed problem in a sheet of paper. The automata has a total of five states, one of which is a finite state (represented as a black circle).

1. Path for single line comment:
    1. Initial state q0
    2. Read “/“
    3. Go to q1
    4. Read “*”
    5. Go to q2
    6. Read ASCII from 32 to 126
    7. Read “*”
    8. Go to q3
    9. Read “/“
    10. Go to final state q4

2. Path or multiple line comment:
    1. Initial state q0
    2. Read “/“
    3. Go to q1
    4. Read “/“
    5. Go to q5
    6. Read ASCII from 32 to 126
    7. Read “next line"
    8. Go to final state q4



Finite automata for signed for comments in C++

#2 Part - From automata to C++ coding

In this second part I started coding in C++. My most important insight was that every state becomes an if statement in C++. The full code can be seen in the page 3-4.

Test Examples

```
Input: //test
Output: ACCEPTED

Input: /*test*/
Output: ACCEPTED

Input: ////////
Output: ACCEPTED

Input: /*//
Output: NOT ACCEPTED

Input: test
Output: NOT ACCEPTED

Input: test//
Output: NOT ACCEPTED

Input: /*test
Output: NOT ACCEPTED

Input: test*/
Output: NOT ACCEPTED

Input: //1010test
Output: ACCEPTED

Input: /*1010*/
Output: ACCEPTED
```