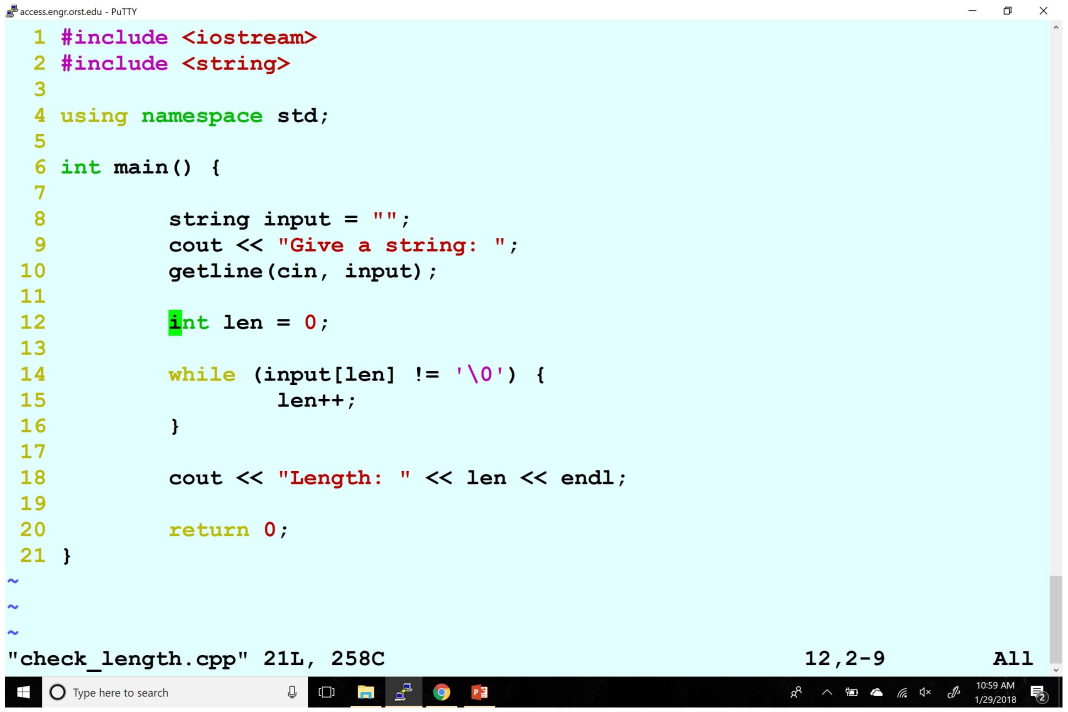
CS 161, Lecture 8: Error Handling and Functions – 29 January 2018

Revisit Error Handling

- Prevent our program from crashing
  - Reasons programs will crash or have issues:
    - Syntax Error – prevents compilation, the programmer caused this by mistyping or breaking language rules
    - Logic Errors – the code does not perform as expected because the underlying logic is incorrect such as off by one, iterating in the wrong direction, having conditions which will never end or be met, etc.
    - Runtime Errors – program stops running due to segmentation fault or infinite loop potentially caused by trying to access memory that is not allocated, bad user input that was not handled, etc.

check_length

- The end of a string is determined by the invisible null character ‘\0’
- while the character in the string is not null, keep counting



```
#include <iostream>
#include <string>

using namespace std;

int main() {

        string input = "";
        cout << "Give a string: ";
        getline(cin, input);

        int len = 0;

        while (input[len] != '\0') {
                len++;
        }

        cout << "Length: " << len << endl;

        return 0;
}
~
~
~
"check_length.cpp" 21L, 258C                                  12,2-9        All
```

Assignment 3 Notes

- Allowed functions:
  - From <string>: .length(), getline(), [], +=
  - From <cmath>: pow()
- Typecasting allowed only if the character being converted fits the stated criterion (i.e. character was confirmed as an int, letter, etc.)
- ASCII Chart should be used heavily http://www.asciitable.com/

Debugging Side Bar

- Read compiler messages when you have a syntax error
- If you suspect a logic error -> print everything!
  - Allows you to track the values stored in your variables, especially in loops and changing scopes
  - Gives you a sense of what is executing when in your program

Decomposition

- Divide problem into subtasks
- Procedural Decomposition: get ready in the morning, cooking, etc.
- Incremental Programming: iterative enhancement (program still works without it all being there)

Intro to Functions

- A block of code to perform an action or subroutine (singular)
- Previously used built in functions from libraries
- Purpose
  - Reduce code
  - Reuse code
  - Readability

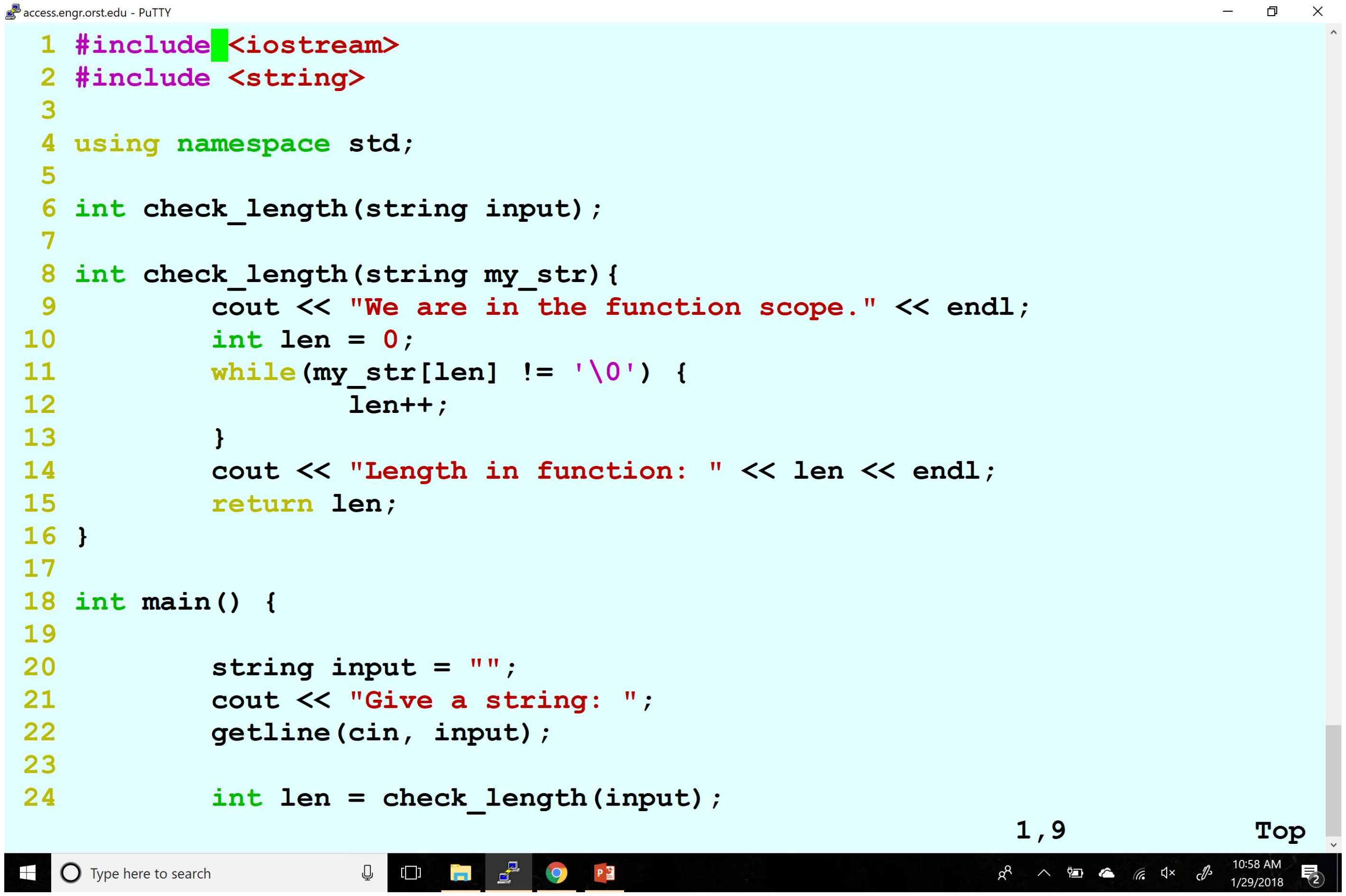
Anatomy of a function

```
int check_length (string input); //declaration: return type, name, parameters

int check_length(string input) { //definition: return type, name, parameters, code
          int len = 0;
          while (input[len] != '\0') {
                    len++;
          }
          return len;
}

int res_length = check_length("hello world"); //function call
```

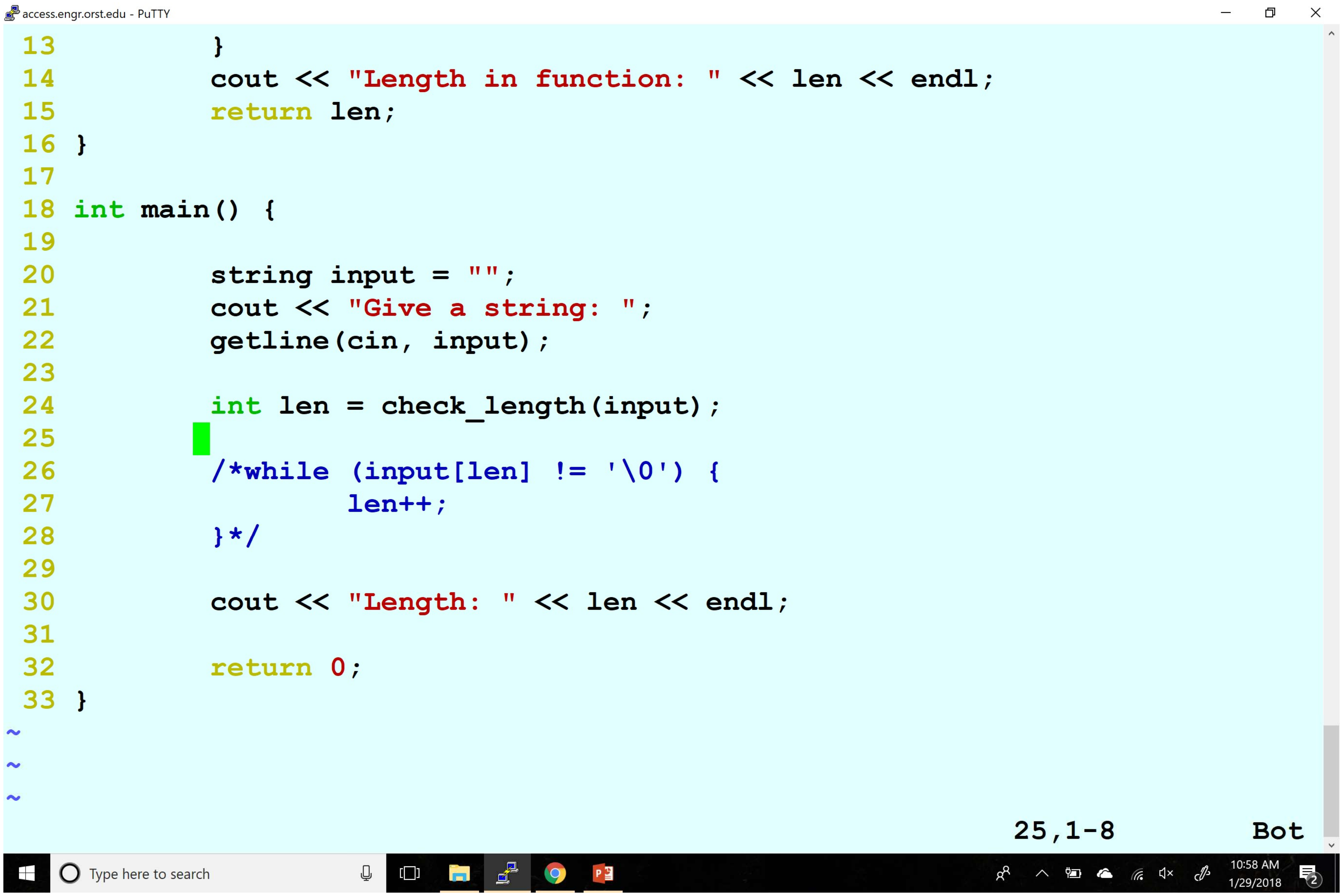
Vocab

- Return type: the data type the function will be returning
  - Any type including void
  - Void means the function does not return anything
  - Can only return one thing at a time
- Declaration: also called prototype, sets up the basic information for the function, does not have to be declared separately from the definition
- Definition: the code that will be executed when the function is called, comprised of header and body
- Parameter: type and name of variable/data the function will be accepting
- Argument: occurs in function call, values which the parameter will take on

Demo

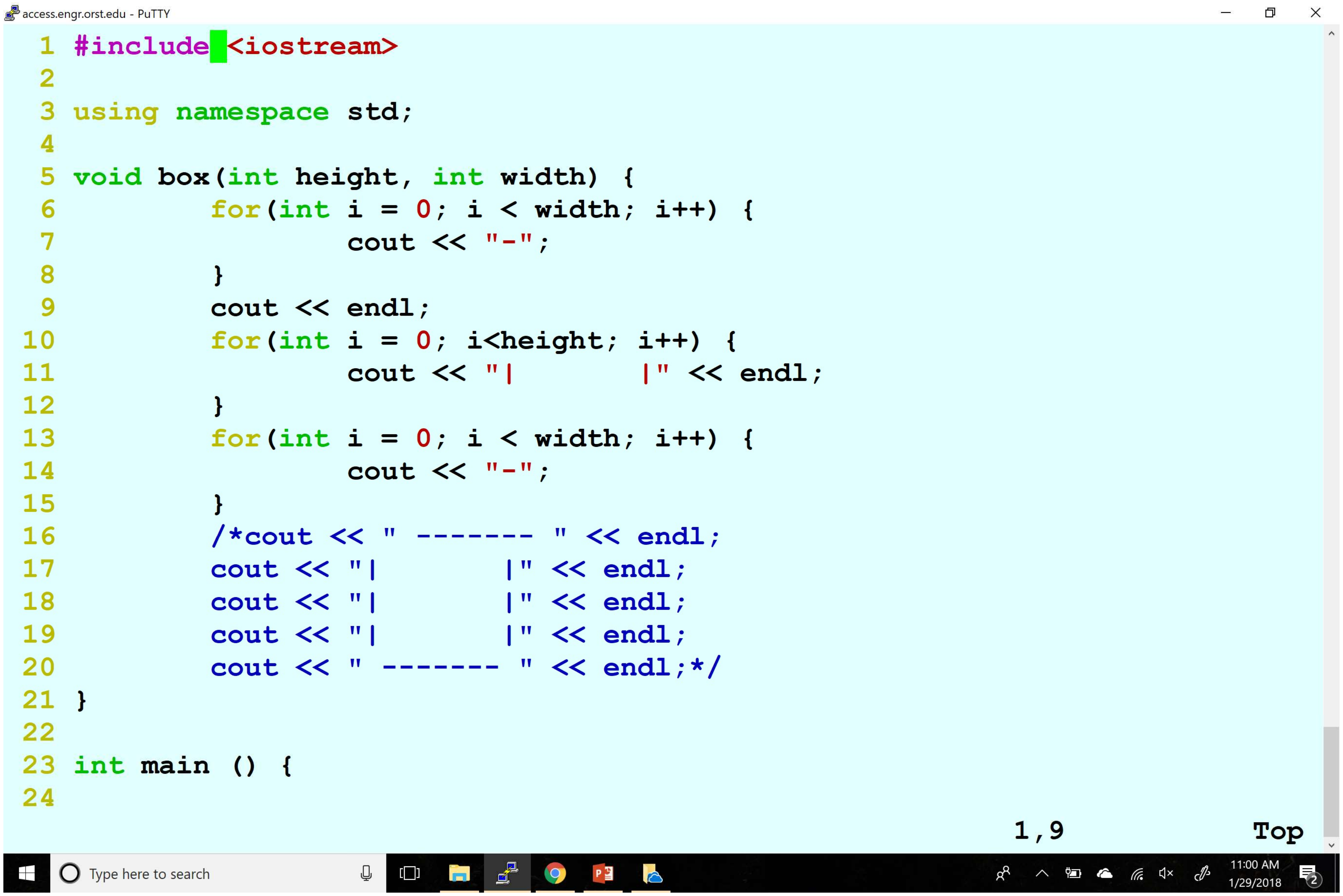


```
#include <iostream>
#include <string>

using namespace std;

int check_length(string input);

int check_length(string my_str){
        cout << "We are in the function scope." << endl;
        int len = 0;
        while(my_str[len] != '\0') {
                len++;
        }
        cout << "Length in function: " << len << endl;
        return len;
}

int main() {

        string input = "";
        cout << "Give a string: ";
        getline(cin, input);

        int len = check_length(input);
                                                        1,9           Top
```



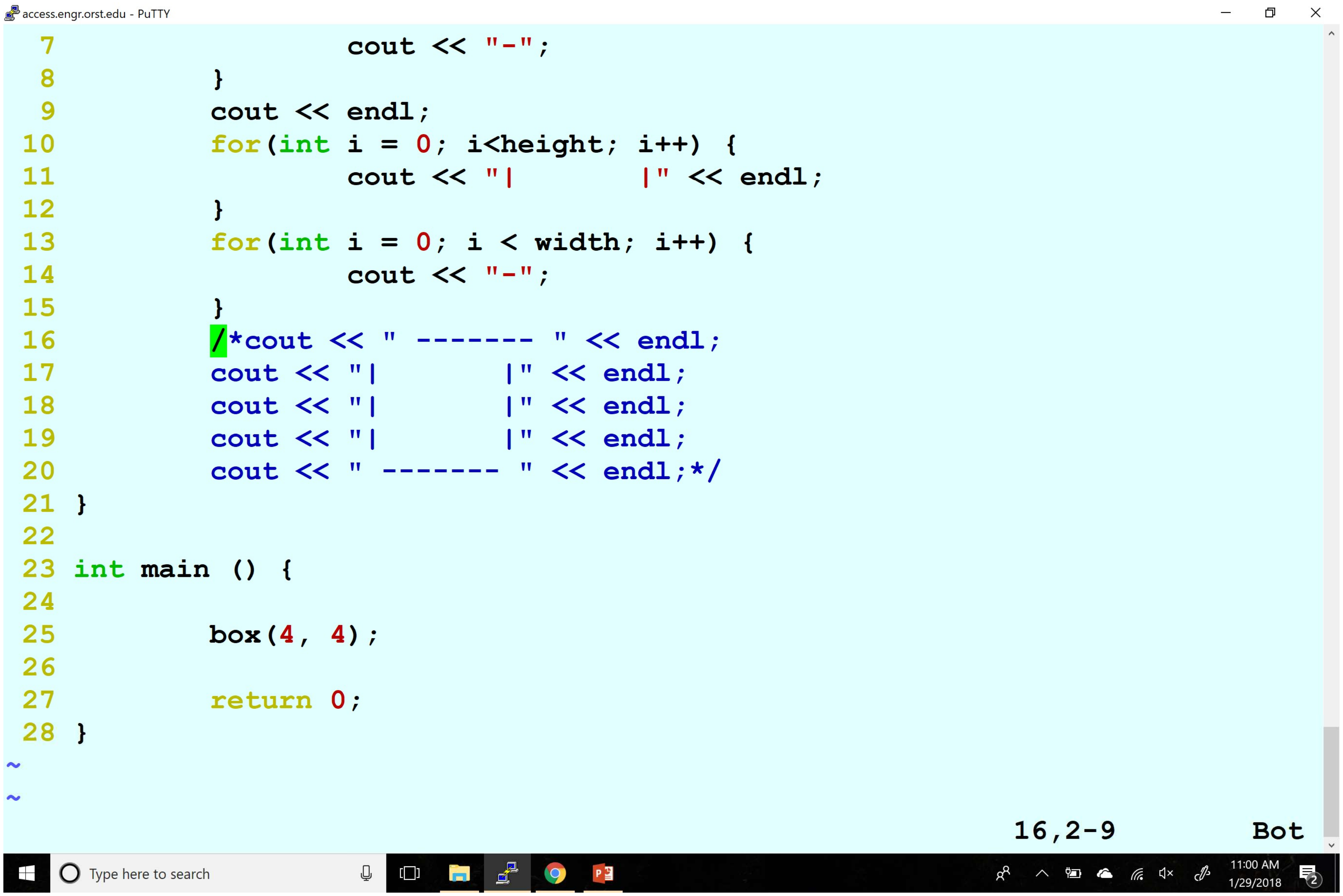
access.engr.orst.edu - PuTTY

```
        }
        cout << "Length in function: " << len << endl;
        return len;
}

int main() {

        string input = "";
        cout << "Give a string: ";
        getline(cin, input);

        int len = check_length(input);

        /*while (input[len] != '\0') {
                len++;
        }*/

        cout << "Length: " << len << endl;

        return 0;
}
~
~
~
                                                    25,1-8          Bot
```

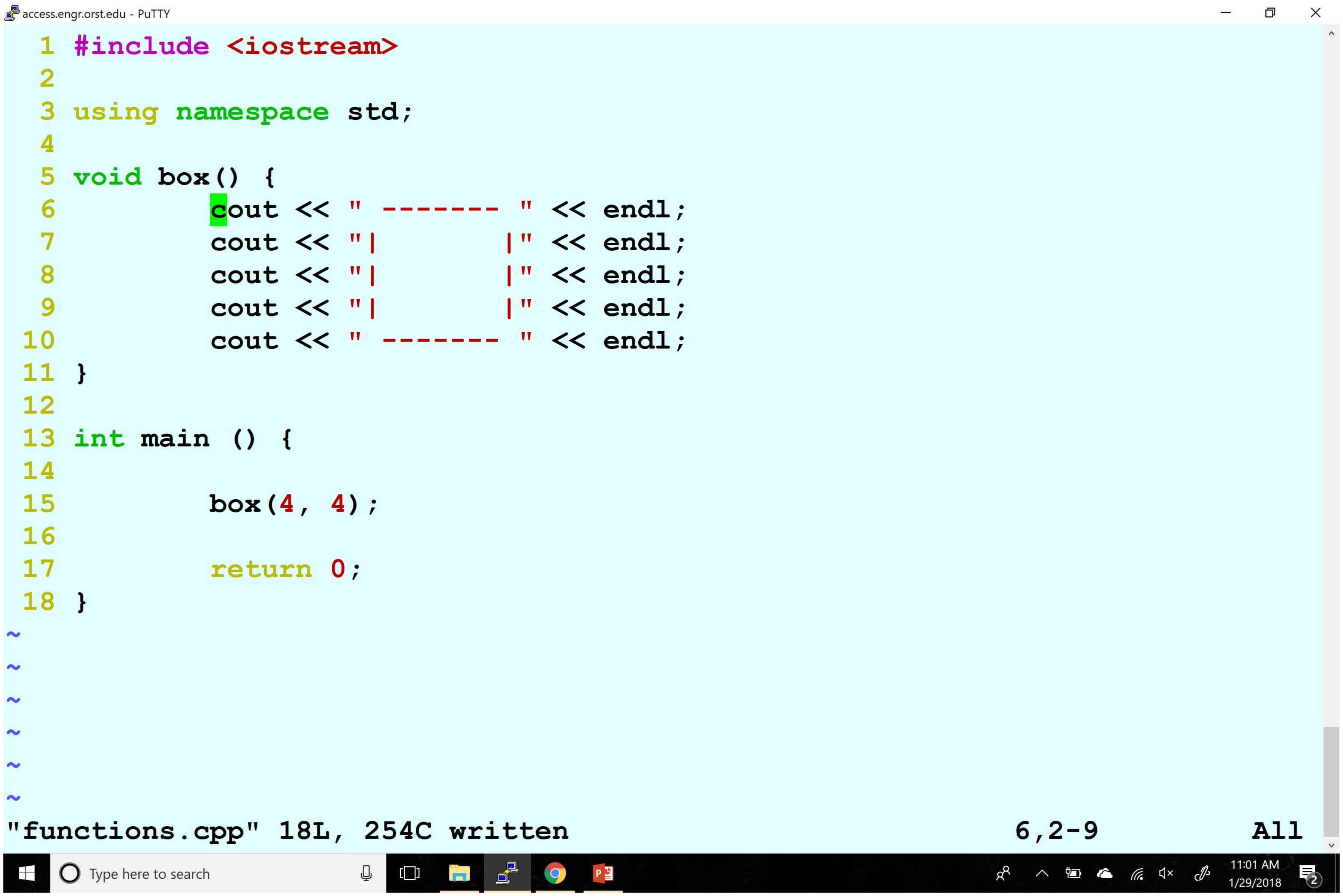

access.engr.orst.edu - PuTTY

```
#include <iostream>

using namespace std;

void box(int height, int width) {
        for(int i = 0; i < width; i++) {
                cout << "-";
        }
        cout << endl;
        for(int i = 0; i<height; i++) {
                cout << "|       |" << endl;
        }
        for(int i = 0; i < width; i++) {
                cout << "-";
        }
        /*cout << " ------- " << endl;
        cout << "|       |" << endl;
        cout << "|       |" << endl;
        cout << "|       |" << endl;
        cout << " ------- " << endl;*/
}

int main () {

                                                            1,9           Top
```

Type here to search
11:00 AM
1/29/2018



```
                cout << "-";
        }
        cout << endl;
        for(int i = 0; i<height; i++) {
                cout << "|       |" << endl;
        }
        for(int i = 0; i < width; i++) {
                cout << "-";
        }
        /*cout << " ------- " << endl;
        cout << "|       |" << endl;
        cout << "|       |" << endl;
        cout << "|       |" << endl;
        cout << " ------- " << endl;*/
}

int main () {

        box(4, 4);

        return 0;
}
~
~
                                                                16,2-9        Bot
```



```
#include <iostream>

using namespace std;

void box() {
        cout << " ------- " << endl;
        cout << "|       |" << endl;
        cout << "|       |" << endl;
        cout << "|       |" << endl;
        cout << " ------- " << endl;
}

int main () {

        box(4, 4);

        return 0;
}
~
~
~
~
~
~
"functions.cpp" 18L, 254C written                                6,2-9         All
```