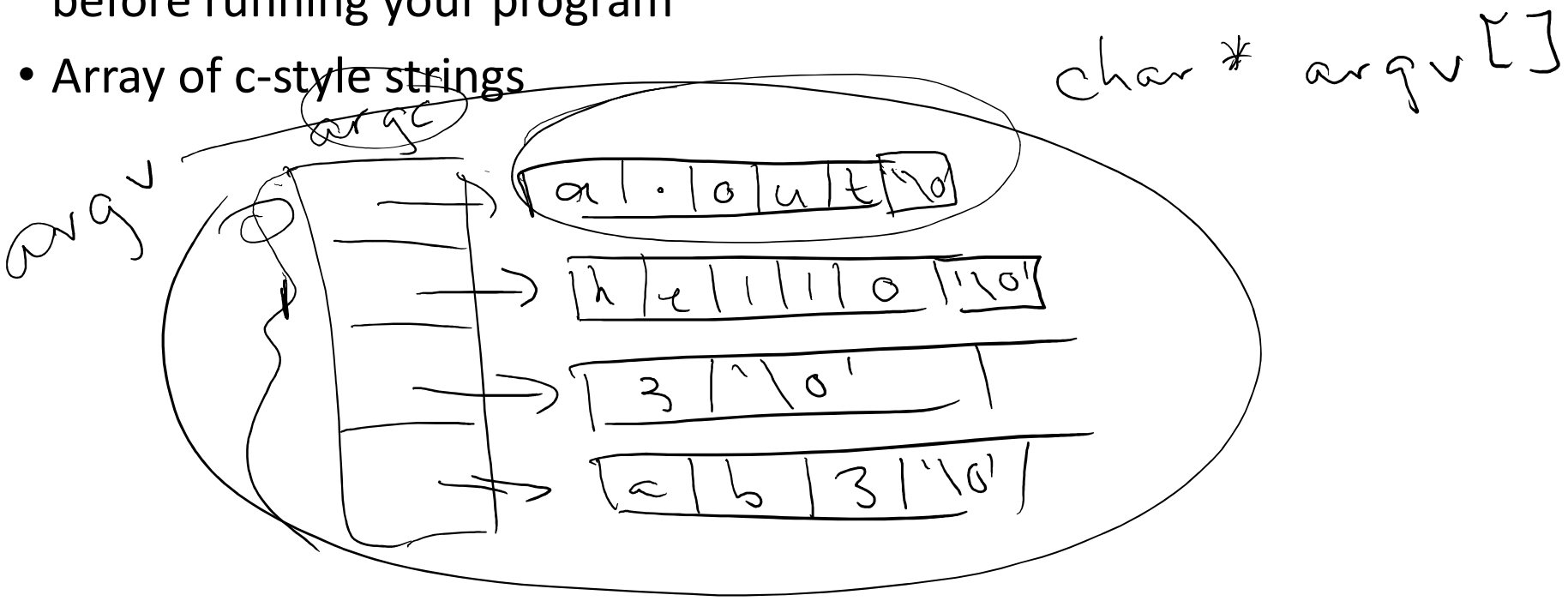
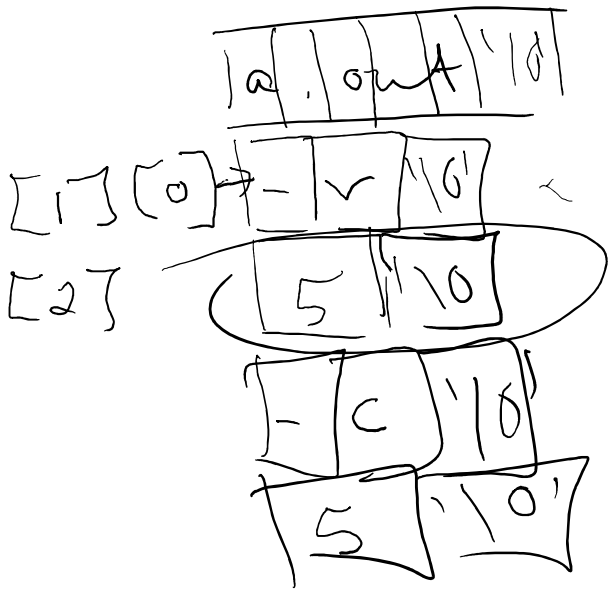
CS 161, Lecture 21: Command Line and More Practice – 2 March 2018

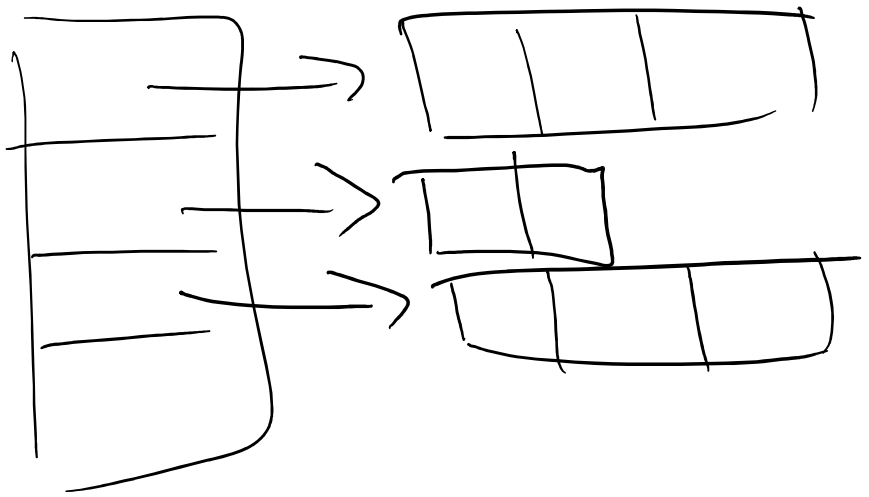
Make it Better: Command Line Arguments

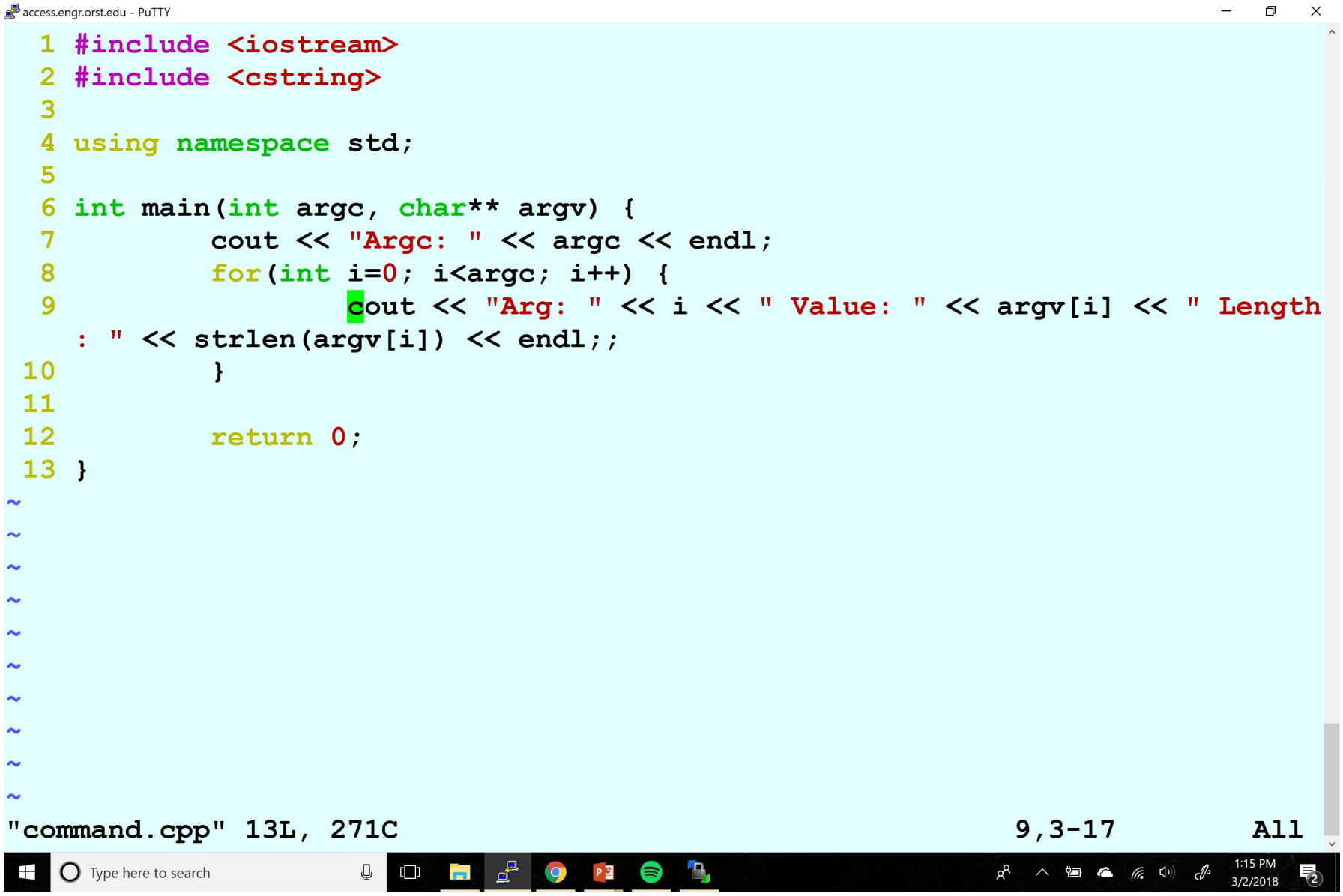
- Command line arguments allow you to take input from the user before running your program
- Array of c-style strings



Demo






access.engr.orst.edu - PuTTY
1 #include <iostream>
2 #include <cstring>
3
4 using namespace std;
5
6 int main(int argc, char** argv) {
7 cout << "Argc: " << argc << endl;
8 for(int i=0; i<argc; i++) {
9 cout << "Arg: " << i << " Value: " << argv[i] << " Length
: " << strlen(argv[i]) << endl;;
10 }
11
12 return 0;
13 }
~
~
~
~
~
~
~
~
~
~
"command.cpp" 13L, 271C 9,3-17 All
Type here to search
1:15 PM
3/2/2018

access.engr.orst.edu - PuTTY

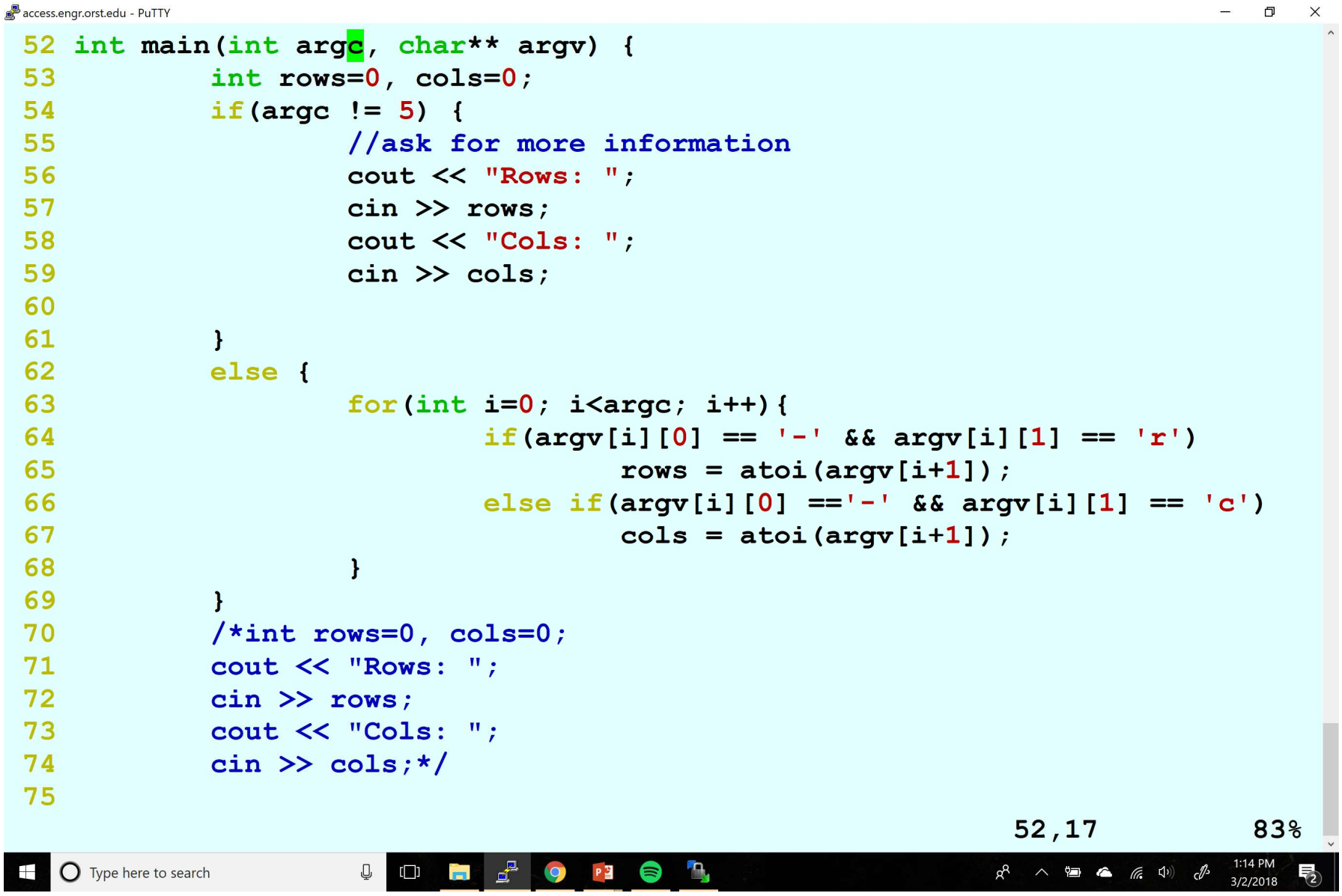
```
 1 #include <iostream>
 2 #include <cstring>
 3 #include <cstdlib>
 4
 5 using namespace std;
 6 //create memory on the heap and return the
 7 //first address
 8 int** init_array(int rows, int cols) {
 9         int** ar = new int*[rows];
10         for(int i=0; i<rows; i++) {
11                 ar[i] = new int[cols];
12         }
13         return ar;
14 }
15 //print the array contents
16 void print_array(int** array, int rows, int cols) {
17         for(int i=0; i<rows; i++) {
18                 for(int j=0; j<cols; j++) {
19                         cout << array[i][j] << " ";
20                 }
21                 cout << endl;
22         }
23 }
24 //set all values in the array to zero
                                                  24,17         Top
```

access.engr.orst.edu - PuTTY

```
22          }
23  }
24  //set all values in the array to zero
25  void pop_array(int** array, int rows, int cols) {
26          for(int i=0; i<rows; i++) {
27                  for(int j=0; j<cols; j++) {
28                          array[i][j] = 0;
29                  }
30          }
31  }
32  //set all values in the array to the product of
33  //the indices
34  void mult_array(int** array, int rows, int cols) {
35          for(int i=0; i<rows; i++) {
36                  for(int j=0; j<cols; j++) {
37                          array[i][j] = i*j;
38                  }
39          }
40  }
41  //delete memory on the heap, columns then
42  //rows, set the array pointer to NULL
43  void delete_array(int** array, int rows) {
44          for(int i=0; i<rows; i++) {
45                  delete [] array[i];
                                                    24,17         34%
```

access.engr.orst.edu - PuTTY

```
40 }
41 //delete memory on the heap, columns then
42 //rows, set the array pointer to NULL
43 void delete_array(int** array, int rows) {
44         for(int i=0; i<rows; i++) {
45                 delete [] array[i];
46         }
47         delete [] array;
48         array = NULL;
49 }
50
51
52 int main(int argc, char** argv) {
53         int rows=0, cols=0;
54         if(argc != 5) {
55                 //ask for more information
56                 cout << "Rows: ";
57                 cin >> rows;
58                 cout << "Cols: ";
59                 cin >> cols;
60
61         }
62         else {
63                 for(int i=0; i<argc; i++){
                                                    40,1          63%
```


access.engr.orst.edu - PuTTY
52 int main(int argc, char** argv) {
53 int rows=0, cols=0;
54 if(argc != 5) {
55 //ask for more information
56 cout << "Rows: ";
57 cin >> rows;
58 cout << "Cols: ";
59 cin >> cols;
60
61 }
62 else {
63 for(int i=0; i<argc; i++){
64 if(argv[i][0] == '-' && argv[i][1] == 'r')
65 rows = atoi(argv[i+1]);
66 else if(argv[i][0] =='-' && argv[i][1] == 'c')
67 cols = atoi(argv[i+1]);
68 }
69 }
70 /*int rows=0, cols=0;
71 cout << "Rows: ";
72 cin >> rows;
73 cout << "Cols: ";
74 cin >> cols;*/
75
52,17 83%
Type here to search
1:14 PM
3/2/2018

access.engr.orst.edu - PuTTY

```
67                              cols = atoi(argv[i+1]);
68                      }
69              }
70              /*int rows=0, cols=0;
71              cout << "Rows: ";
72              cin >> rows;
73              cout << "Cols: ";
74              cin >> cols;*/
75
76              int** array = init_array(rows, cols);
77              print_array(array, rows, cols);
78              pop_array(array, rows, cols);
79              print_array(array, rows, cols);
80              mult_array(array, rows, cols);
81              print_array(array, rows, cols);
82              delete_array(array, rows);
83
84              return 0;
85      }
~
~
~
~
~
                                                        67,3-24         Bot
```

C-String Functions

- Recall c-style string are null terminated
- Means we can use c-string library
- We can also make all of these by hand
- Useful
  - Strlen -> length of string up to (not including) null terminator
  - Strcpy -> copies the contents of one c-style string into another
- Get input from user using
  - cin.get() -> takes one character from the buffer at a time (leaves deliminator char in buffer)
  - cin.peek() -> looks at the next thing in the buffer with out taking it out
  - cin.getline() -> takes an entire line of determined size
  - cin.ignore() -> dumps everything in the buffer
  - cin.clear() -> resets error flag for buffer

Exercise: String Manipulation with C-style strings

- Design a program which will take a string from the user (c-style string) and remove all duplicate letters occurring in the string. You should only use the exact amount of memory you need at any given time. Your program should not take input from the user during runtime unless there is an error.
  - What functions will we need?
  - What kind of memory should we use?
  - Write the function prototypes.

Demo