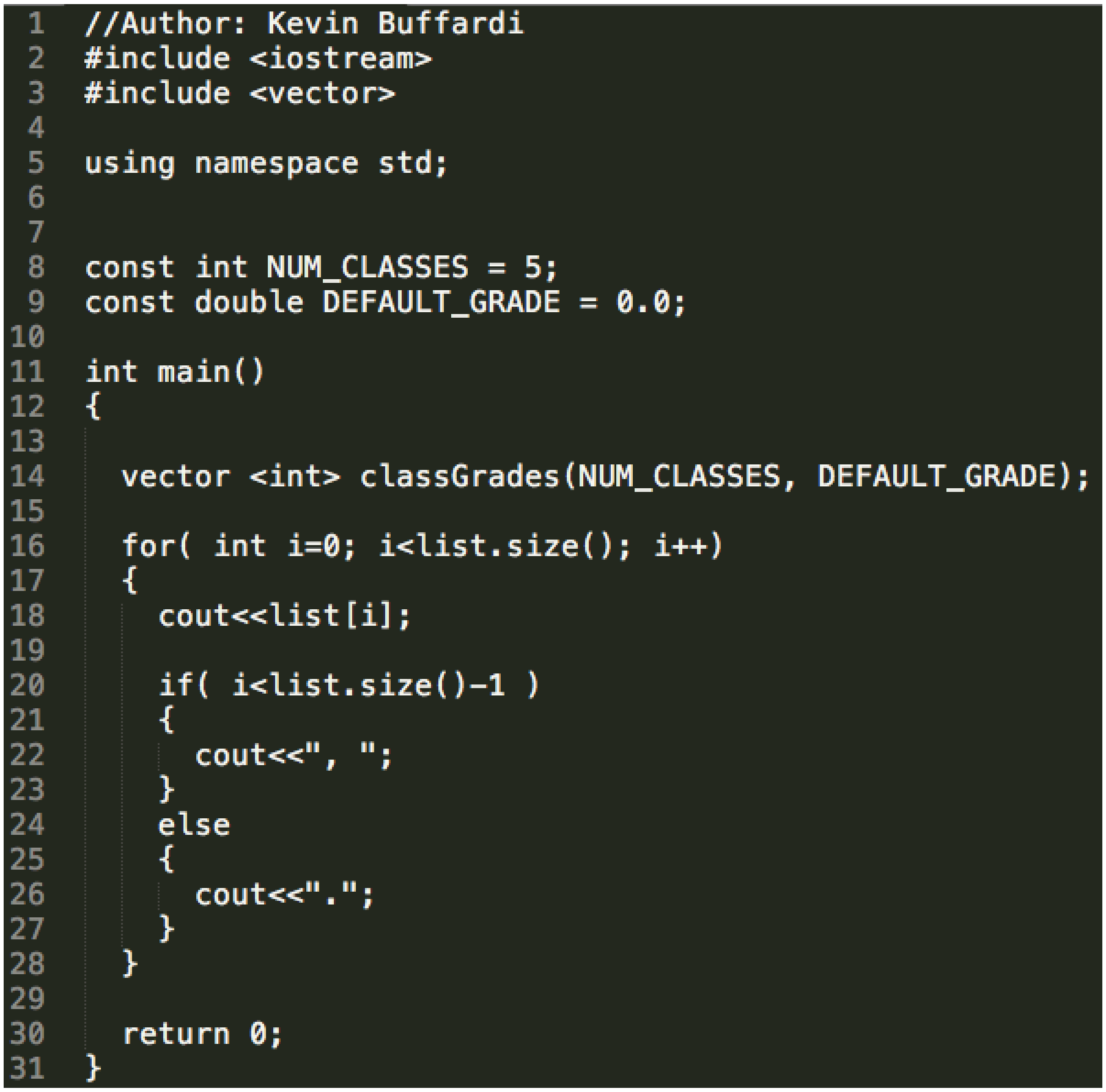
Style Guide

As our code gets more sophisticated (and progressively more difficult to understand and to debug), it becomes more and more important to have good style in our code. Accordingly, for projects, you are expected to follow the style conventions as illustrated below:



```
 1 //Author: Kevin Buffardi
 2 #include <iostream>
 3 #include <vector>
 4
 5 using namespace std;
 6
 7
 8 const int NUM_CLASSES = 5;
 9 const double DEFAULT_GRADE = 0.0;
10
11 int main()
12 {
13
14   vector <int> classGrades(NUM_CLASSES, DEFAULT_GRADE);
15
16   for( int i=0; i<list.size(); i++)
17   {
18     cout<<list[i];
19
20     if( i<list.size()-1 )
21     {
22       cout<<", ";
23     }
24     else
25     {
26       cout<<".";
27     }
28   }
29
30   return 0;
31 }
```

In summary:

- All regular variables should have **descriptive names that most people would understand** and use camelCase (https://en.wikipedia.org/wiki/Camel_case) (see classGrades, line 14) or snake_case (https://en.wikipedia.org/wiki/Snake_case)
- Declare variables at the **beginning of their code blocks** (see line 14)
- Declare variables locally, with as **small a scope as <u>necessary</u>** for each variable

- Avoid "Magic Numbers" and replace them with global constant variables (see lines 8-9, used in line 14). This is the ONLY allowable use of global variables. Name all global variables using **CONSTANT_CASE** **(https://en.wikipedia.org/wiki/Naming_convention_(programming))**
- All constructs with associated code blocks (e.g. loops, control flow statements, functions, etc) should have curly braces
- Code blocks should be arranged with curly braces (both open { and close }) that align vertically with their construct (e.g. lines 17 and 28 aligning with for on line 16). If you prefer, the open brace { can be at the end of the construct instead of on the next line as long as you adopt that practice consistently
- Code contained within blocks should be indented consistently, either with one tab or two spaces, throughout your project
- Nested constructs should follow the same curly brace and indentation conventions as described in the previous criteria (see lines 20-27)
- Use vertical whitespace to separate segments of code that are closely related (see lines 15, 19, and 29)
- Keep lines short (unless impractical to avoid)
- Functions should have prototypes above main and definitions below main
- Classes should have separate interface files (.h) and implementation (.cpp) files