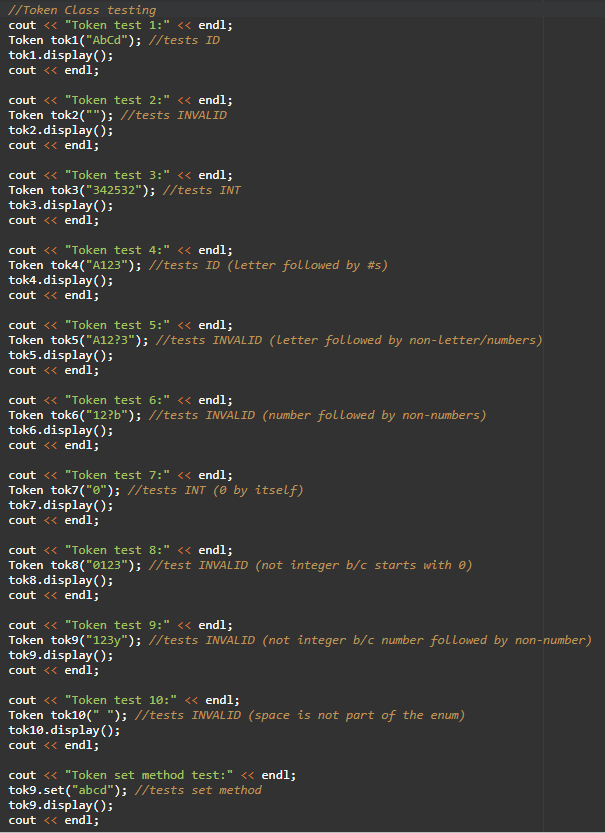
**Homework Three Project Report**

In this homework project, an observation I made was the scope of variables, classes, and methods throughout the assignment, and how that was critical for variable definition in C++. The most prevalent case of this was in Expression.cpp where we needed to create a vector composed of Token variables (Token class). For this reason, I needed to use Token class methods to add tokens from an input string into the vector. Something I learned in this assignment were string library and vector library functions such as .clear(), .append(str, pos, n), and .push\_back(x). I used this functions in Expression.cpp to split a string character by character with conditional statements, add them to a vector (after converting into Token using the Token constructor), then clearing the string.

Another thing I learned was that a vector is similar to an array in C++, but it’s focus is in dynamic application. In this assignment, we used the vector library, string library, and header files to create a project that both tests tokens and expressions made up of tokens. The use of header files in this lab further expanded upon previous lab understanding of header files as we focused on Class definitions that contain enums, scope (public, private), methods (implemented in specific cpp files), and data members. The biggest takeaway from this project was the creation and modification of vectors and building Classes from scratch and implementing their methods.

A continuing theme is the importance of test cases, as this homework assignment contained many test cases to cover a wide range of possible outputs from calling constructors and methods.

**Token Class testing code in main:**



**Token Class testing output:**

**Token test 1:**

**type = ID**

**token = AbCd (value is -1)**

**priority = -1**

**Token test 2:**

**type = INVALID**

**token = (value is -2)**

**priority = -1**

**Token test 3:**

**type = INT**

**token = 342532 (value is 342532)**

**priority = -1**

**Token test 4:**

**type = ID**

**token = A123 (value is -1)**

**priority = -1**

**Token test 5:**

**type = INVALID**

**token = A12?3 (value is -2)**

**priority = -1**

**Token test 6:**

**type = INVALID**

**token = 12?b (value is -2)**

**priority = -1**

**Token test 7:**

**type = INT**

**token = 0 (value is 0)**

**priority = -1**

**Token test 8:**

**type = INVALID**

**token = 0123 (value is -2)**

**priority = -1**

**Token test 9:**

**type = INVALID**

**token = 123y (value is -2)**

**priority = -1**

**Token test 10:**

**type = INVALID**

**token = (value is -2)**

**priority = -1**

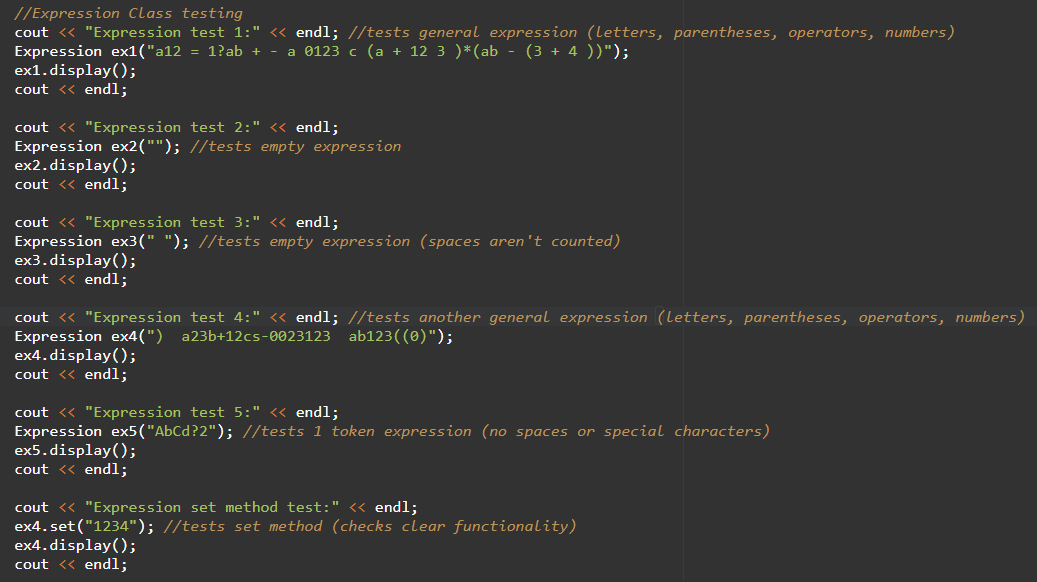
**Token set method test:**

**type = ID**

**token = abcd (value is -1)**

**priority = -1**

**Expression Class Testing:**

****

**Expression Class testing output:**

**Expression test 1:**

**original = a12 = 1?ab + - a 0123 c (a + 12 3 )\*(ab - (3 + 4 ))**

**tokenized = a12;=;1?ab;+;-;a;0123;c;(;a;+;12;3;);\*;(;ab;-;(;3;+;4;););**

**number of tokens = 24**

**postfix =**

**valid = 0**

**type = ILLEGAL**

**Expression test 2:**

**original =**

**tokenized =**

**number of tokens = 0**

**postfix =**

**valid = 0**

**type = ILLEGAL**

**Expression test 3:**

**original =**

**tokenized =**

**number of tokens = 0**

**postfix =**

**valid = 0**

**type = ILLEGAL**

**Expression test 4:**

**original = ) a23b+12cs-0023123 ab123((0)**

**tokenized = );a23b;+;12cs;-;0023123;ab123;(;(;0;);**

**number of tokens = 11**

**postfix =**

**valid = 0**

**type = ILLEGAL**

**Expression test 5:**

**original = AbCd?2**

**tokenized = AbCd?2;**

**number of tokens = 1**

**postfix =**

**valid = 0**

**type = ILLEGAL**

**Expression set method test:**

**original = 1234**

**tokenized = 1234;**

**number of tokens = 1**

**postfix =**

**valid = 0**

**type = ILLEGAL**