

24th October 2020

Check for Balanced Parentheses

Detailed Analysis: I must write a program that reads another program and checks for balanced parenthesis. I will be using a stack and checking for two conditions: if the stack is empty before popping and if the program is empty, and if I have finished reading the program. If the program was read in its entirety and the stack is empty, then I have the balance. Likewise, if the program was read but the stack is not empty, I have too many. Also, if the stack is empty and if you can't pop it and still make more of the program characters to read, you have too many. These are situations that I will expertly deal with in the following project.


I have to check for balanced parentheses in this C++ coding project. I first typed into code lines 17-27 a summary of the stack functions I was going to use. This comes from the Standard Template Library, or STL for short. I can use this to my advantage by adding `#include <stack>` at the top of my project so that it may work and compile successfully. The STL is really just an extensive library of generic templates for classes and functions. One of these templates will be used to my advantage and that is the template of iterators. Iterators can be used as class templates for objects that behave like pointers, and they can be used to access individual data elements in a container (a separate template of its own). For example, I am using this concept in action in code lines 59 -65, and more specifically I called `vector<int> :: iterator` it. This is useful for checking one of the conditions, which is if the program is done being read so that the stack empty will be popping accurately.

That's not to say if the program is completely empty in of itself, which would fulfill another requirement in order for this project to be successful. I also must take into account sequence containers and associative containers while we are accounting for the subject of containers in general. Sequence containers are when I would have to theoretically store important data sequentially in a memory, and this can be described as similar to a computer array. On the flip side, associative containers will store data in a nonsequential order, and this makes my program more efficient when it comes to locating certain elements.



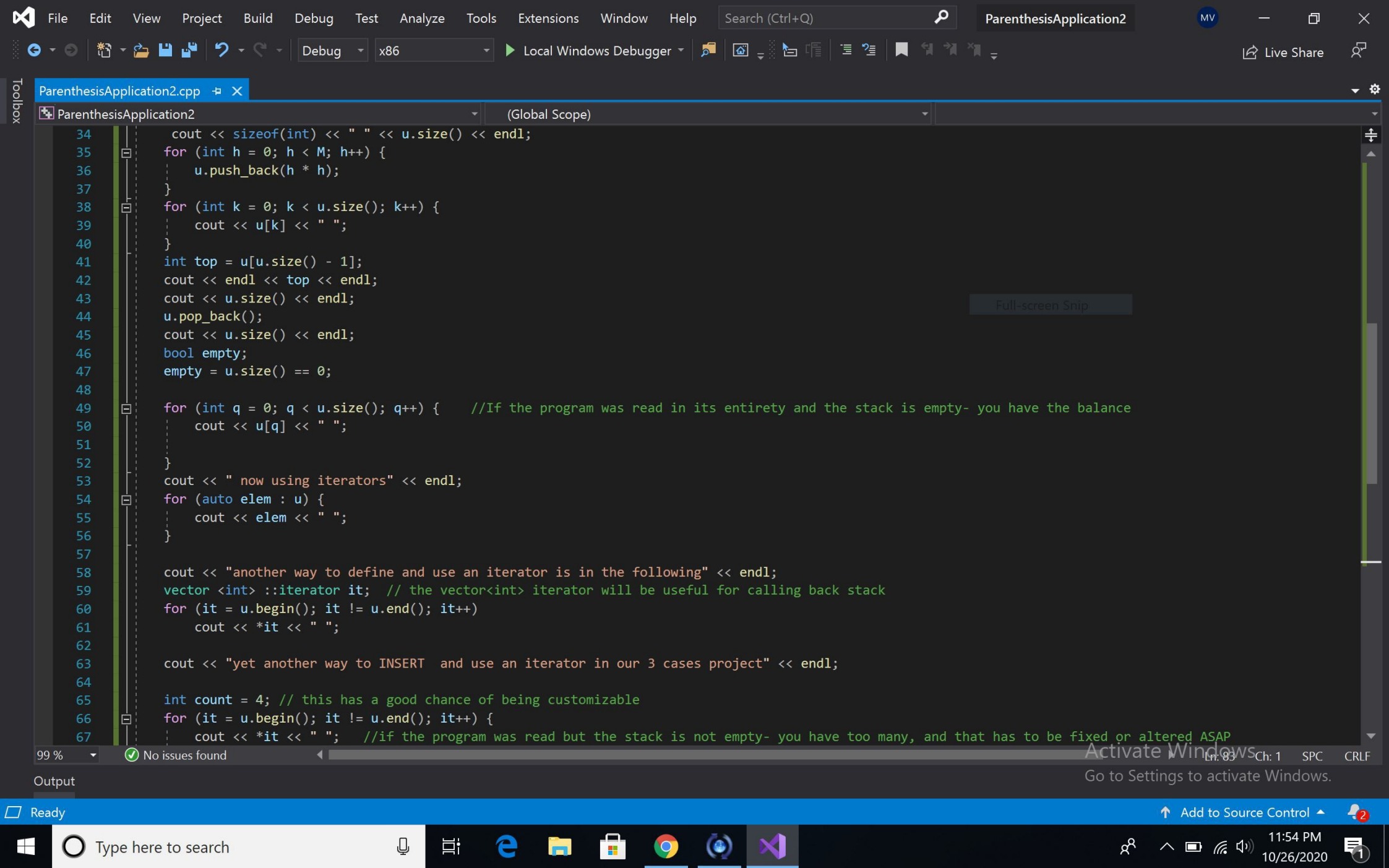
Debug x86 Local Windows Debugger

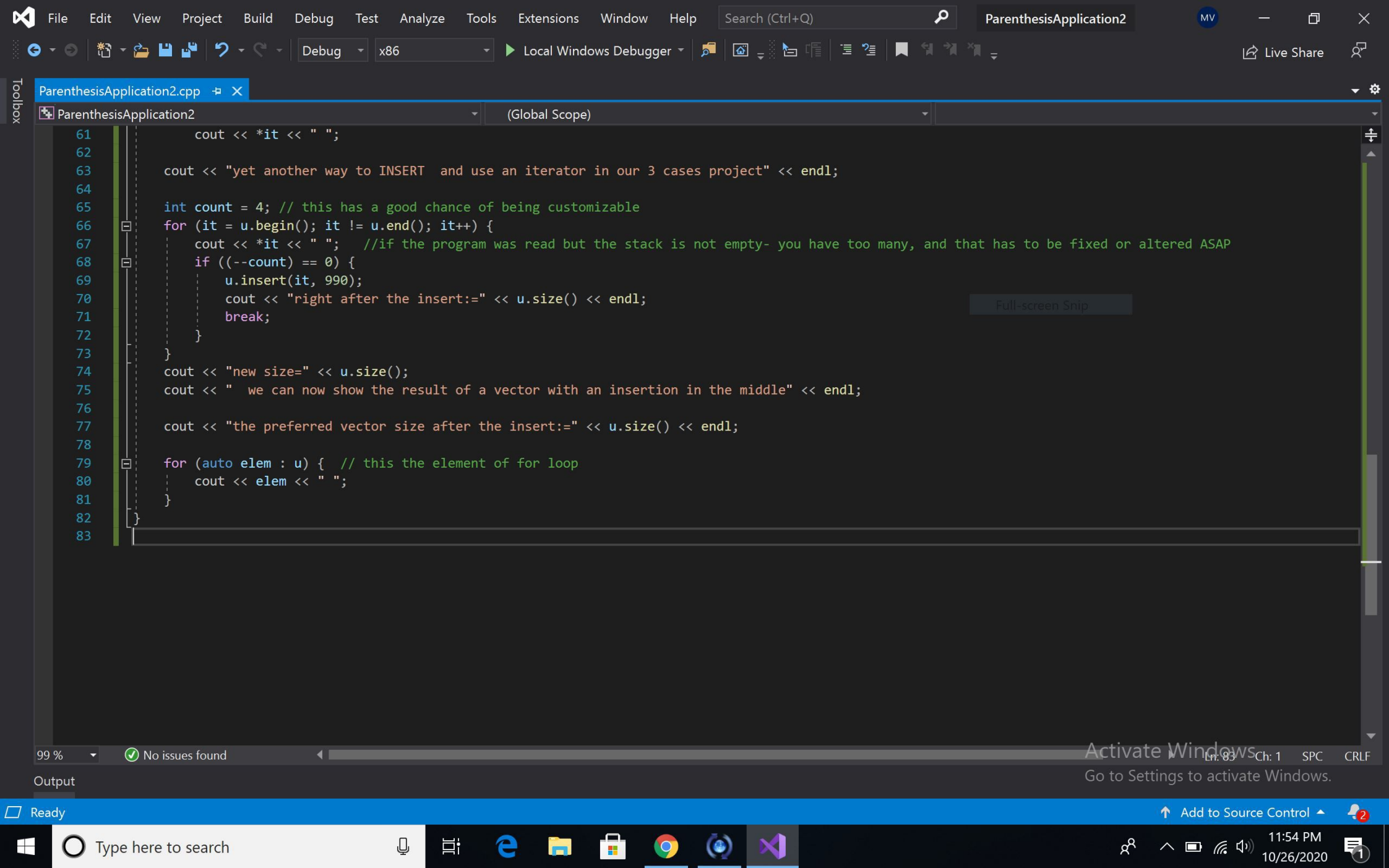
 ParenthesisApplication2
 (Global Scope)

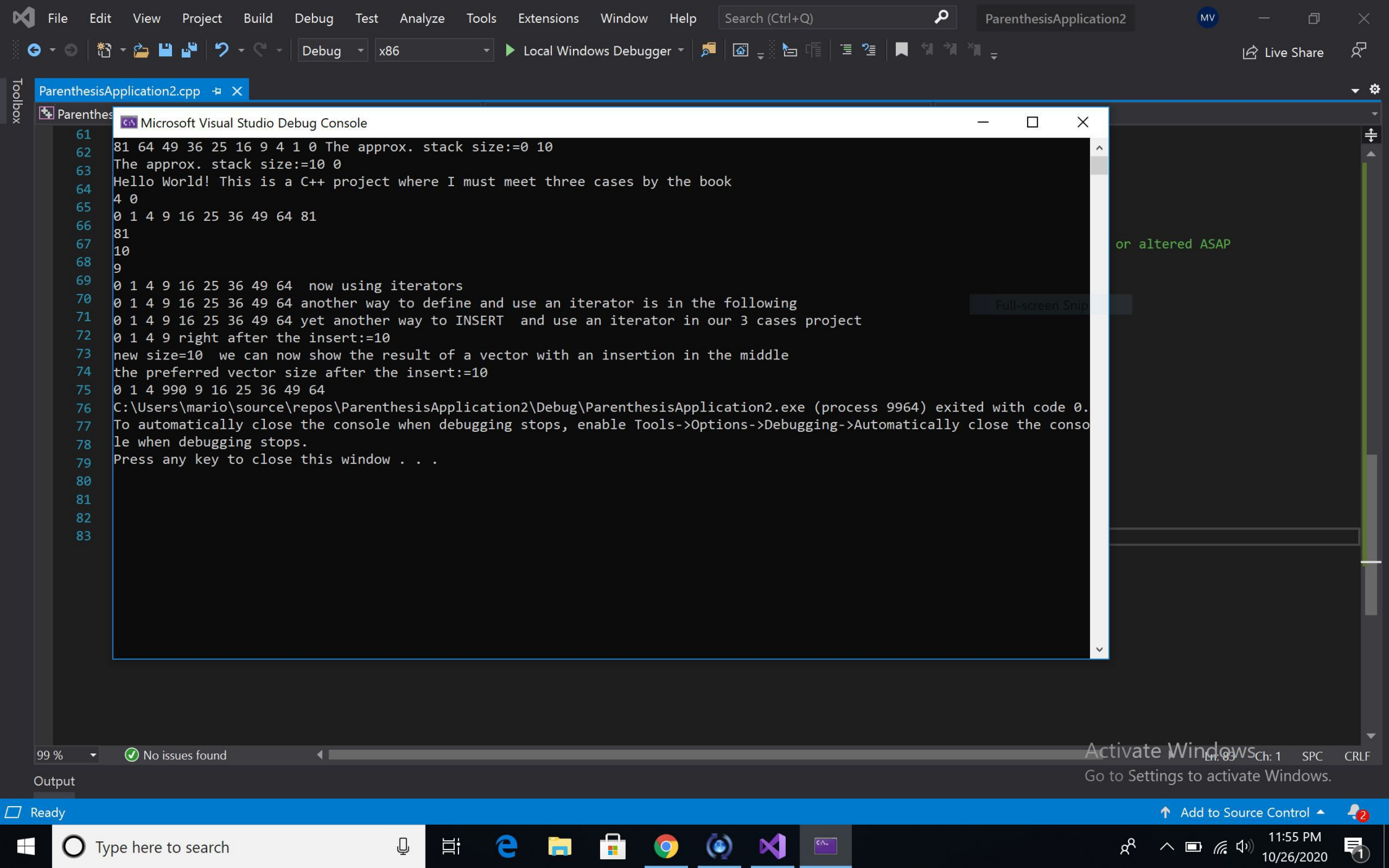
99 %  No issues found Ln: 83 Ch: 1 SPC CRLF

ready [Add to Source Control](#) 

Type here to search







ParenthesisApplication2.cpp

Parenthesis

```
Microsoft Visual Studio Debug Console
61 81 64 49 36 25 16 9 4 1 0 The approx. stack size:=0 10
62 The approx. stack size:=10 0
63 Hello World! This is a C++ project where I must meet three cases by the book
64 4 0
65 0 1 4 9 16 25 36 49 64 81
66 81
67 10
68 9
69 0 1 4 9 16 25 36 49 64 now using iterators
70 0 1 4 9 16 25 36 49 64 another way to define and use an iterator is in the following
71 0 1 4 9 16 25 36 49 64 yet another way to INSERT and use an iterator in our 3 cases project
72 0 1 4 9 right after the insert:=10
73 new size=10 we can now show the result of a vector with an insertion in the middle
74 the preferred vector size after the insert:=10
75 0 1 4 990 9 16 25 36 49 64
76 C:\Users\mario\source\repos\ParenthesisApplication2\Debug\ParenthesisApplication2.exe (process 9964) exited with code 0.
77 To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
78 Press any key to close this window . . .
80
81
82
83
```

99 % No issues found

Output