SOEN6011(DELIVERABLE 2)

https://github.com/vik-sandhu/soen_6011

F2: tan(x)

Author:
VIKRAMJIT SINGH

Student ID:
40075774

July 30, 2019

Contents

1 Source Code

1.1 Correctness

As mentioned in the algorithm, I reduced the angle provided by user so that it's in the range $-90° < x \leq 90°$.But using this algorithm, as we move closer to $90°$, answer deviates from the correctness to a large extent.

So, i order to achieve more correctness, I reduced to larger extent. So now I reduced the angle provided by user so that it's in the range $-22.5° < x \leq 22.5°$. After testing this , i concluded that the polynomial approximation used in algorithm is accurate to within ± 0.000006.

The solution provided by my source code is accurate to within ± 0.000006.

1.2 Efficiency

The complexity of program is very less because I used optimized data structures and didn't over used variables. Moreover, the result calculated is very accurate.It takes just 15 milliseconds to compute the result.

1.3 Maintainable

In this program, proper commenting is provided for each function used, making it easier to understand. Furthermore, it's very easy to understand the code flow. Moreover, changes can be made easily to this program without introducing new bugs, if ever needed so.

1.4 Usable

Persistent effort was made towards polishing the functionality of algorithmic java code to make it usable.

1.5 Robust

Instead of writing one big black chunk of code upfront and executing, I'm rather breaking down the code into functional steps and then creating the steps one by one.I focus purely on how I will be breaking down the problem into the overall context of the requirement.Every class, function, and subroutine does only one function making the code robust, clean and modular.

2 Debugger

Debugging gives us a one of a kind knowledge into how a program runs and enables us to pick up an a lot further comprehension of the bit of code we troubleshoot. For this project I used the IntelliJ IDEA debugger. The IntelliJ IDEA debugger offers a rich encounter that encourages us to effortlessly investigate anything from the least complex code to complex multithreaded applications.

2.1 Advantages

- The IntelliJ IDEA debugger is very convenient, much easier, intelligent and faster.
- It also provides features likes autofilling, syntax highlighting.

2.2 Disadvantages

- The Utlimate Edition is very expensive.
- It's not completely open source, due to which community support is less.

3 CheckStyle

To check the source code's adherence to the coding standards and make sure that the quality of source code is maintained, I used CheckStyle's plugin for IntelliJ IDEA.This plugin provides both real-time and on-demand scanning of Java files with CheckStyle from within IDEA. CheckStyle characterizes a lot of accessible modules, every one of which furnishes principles checking with a configurable degree of severity (obligatory, optional...). Each standard can raise notices, admonitions, and mistakes.CheckStyle uses ANTLR (ANother Tool for Language Recognition) parser generator.

3.1 Advantages

- Checkstyle is portable between IDE's.
- Functionality of CheckStyle is mainly implemented as checks, which we can also write according to our need. In other words, we can extent the functionality of CheckStyle.

3.2 Disadvantages

- Checkstyle is limited to the presentation of the code. These checks don't affirm the accuracy of the code.
- Checkstyle is a single file static analysis tool.
- CheckStyle do not support UTF-8.
- Code should be compilable, in order to get violations from CheckStyle.

References

[1] https://en.wikipedia.org/wiki/Checkstyle

[2] https://en.wikipedia.org/wiki/IntelliJ$_IDEA$