Project Proposal *Thomas Shaer- 1905941*

"An array programming language for charting data"

General Topic

My idea for my project is to build a simple array programming language that is designed for charting data. A chart will exist within the tool, and the user can write code to display certain plots on the chart. The language and chart will both be integrated into a single tool for convenience. Additionally, the user can input array data into the tool via excel files, csv etc., which can be charted with the language. The applications for such a tool would be mainly research and education.

Problem we're trying to solve

The problem I'm trying to solve is to create a simple desktop application where a user can build charts quickly and efficiently. The language will be as simple to use as possible, and I want the tool to be offline so it can be used anywhere.

Similar Projects

Many array programming languages exist:

- MATLAB (https://www.mathworks.com/products/matlab.html)
  - A similar tool such that it utilises a language to help create mathematical models. Also has a large emphasis on charting.
- Desmos Graphics Calculator (https://www.desmos.com/calculator)
  - A graphing calculator that implements a simple language that only utilises variables and expressions. The "language" operates on infinite arrays of data (coordinates). Although the syntax of the language is quite unique and its works on infinite arrays, Desmos is fundamentally quite similar to what I have in mind.
- Tradingview's Pine script (https://www.tradingview.com/chart/)
  - A language designed to operate on arrays in a back-testing context. It is used to build "trading strategies" for trading financial instruments. Uses financial data such as stock pricing to build the trading strategies with. This tool is what I am largely basing my product off, except my tool will run on any data the user uploads rather than purely financial.

Challenges, difficulties, and interesting parts of project

One particularly challenging aspect of implementing an array programming language is predicting the impact it will have on how the language functions. Although I plan to keep the language simple (at least at first), acting more like an expression evaluator with variables and functions rather than a fully fletched programming language - since every single variable will be pointing to an array, it might prove difficult to implement more sophisticated language features such as if statements and for loops. While these are not planned features, it opens questions as what other features might be affected by this decision requirement.

Another challenging aspect of the program might be uploading user array data into the tool. I want the tool to support large series of data that might potentially exist in several different formats/filetypes. I will have to design an interface to support a range of different file types which could be challenging.

The largest challenge of the project will be integration of all different components into one single desktop tool. There will be several large components in the tool such as the GUI, data loading and viewing charts and due to the variety of inputs the tool can take, the tool may behave unexpectedly under certain edge cases. I think it will be important to make sure the whole tool performs in a predictable way such that the user can grow comfortable with it over time, so a lot of testing will have to be undergone to make sure it works fluidly.

Methods to solve problem or answer your question

As mentioned before, the tool should be as easy to use as possible so the user can create charts quickly. In terms of simplicity, the language itself will only consist of variables, operations, and built-in methods. This will make it very quick to learn even for people with limited experience in programming. As demonstrated by Desmos, which is used regularly in education, a carefully designed programming language can be just as simple to understand as basic maths.

The tool will be convenient such that all major components such as the language input, the chart, and data loading tools will be integrated into one interface, including documentation. I also want the tool to be completely offline, so it can be accessed at any time.

The tool will be versatile because it will be able to work on any data that is inputted into it. The tool should be able to accept input data of any size (subject to chart performance) in a variety of different formats so that users will not have to do manual formatting of their data.

Major milestones

I have created three major milestones for my project. The first milestone is getting all the basic features of the program working, which includes a working language, integrated chart and a GUI. These are the *essential* components of the project, without them, the user would not be able to create and view the mathematical models that the project is supposed to do. Since some of these major components are built on top of each other, for example it would not be possible to display a chart without a basic GUI, the "sub-milestones" are ordered in such a way to reflect this dependency hierarchy - for example build a basic GUI first, then integrate a chart. Additionally, I have opted to build what I predict to be the most time-consuming yet essential components first, such that if I have time issues towards the end of the project, I will be reassured in knowing that the most essential ones have already been delivered.

The first major milestone will likely take the largest amount of time to complete as it promises to deliver the basic functionality of the application. Thus, once that has been completed, the rest of the time spent working on the project will be related to adding polish/new features to the program. This time is split up into two further milestones. The first of which, *milestone 2* is related to enhancing the user experience with "quality of life" improvements. These improvements will be non-essential but if implemented would greatly enhance the software. The order of these sub milestones won't matter as much since they are independent of each other. Milestone 3 can be seen as the polish stage. Like milestone 2, these features are non-essential but would make the software seem more professional. I do not expect to achieve all the features detailed in milestone 2 and 3 but I would like to achieve as many of them as possible.

Milestone 1

Finish a working prototype with all basic components implemented.

Planned time of completion: between week 10-13

1. Basic version of the language with following features implemented:
    a. Float/Boolean/null type
    b. Built In Methods
    c. Arithmetic/Comparison
    d. Ability to output data that can be rendered by a chart
2. Get basic GUI framework going for interaction with the program
    a. Basic integration of language into the GUI
3. Basic chart working that can display output information from the language.
4. Basic input and labelling for user submitted data so that it can be used and referenced in the language.

Milestone 2

Add extra features that make the software more convenient for the user to use. These items are not required but enhance the product greatly.

Planned time of completion: between week 13-18

- Interface for saving/loading code to and from the user's file system
- Ability for user to export chart data from the program
- Additional built in functions supporting a common basic mathematical operations such as regressions, stochastics etc.
- If required, more sophisticated methods of loading in data from CSVs, e.g. tables of data, different file types etc.
- Testing the application for system compatibility on different OS's

Milestone 3

Add features that add to a sense of "polish" for the software. These features will range in difficulty and its likely that I will not be able to implement all of them. However implementing them would significantly enhance the program.

Planned time of completion: between week 18-22

- Intellisense/autocomplete for writing code
- More language features such as if statements or user defined functions
- Ability to spawn and plot on multiple different charts
- Ability to save a chart to disk as a PNG or JPEG