Documentation for “Function to Graph (Fun_to_graph)”

Overview

“Function to Graph” is a command-line C++ application designed to read a mathematical function from a text file, generate its graph using LaTeX, and output the result as a PDF file. This document provides an in-depth overview of the project’s architecture, development process, and usage.

Project Architecture

Components

1. **FunctionReader** (`FunctionReader.h/cpp`):
    - **Purpose**: Reads a mathematical function from a text file.
    - **Implementation**:
        - Constructor takes a filename and stores it.
        - `readFunction` method reads and validates the function’s format, throwing exceptions for errors.
2. **LaTeXGraphGenerator** (`LaTeXGraphGenerator.h/cpp`):
    - **Purpose**: Generates LaTeX code to represent the function graphically.
    - **Implementation**:
        - Constructor takes the mathematical function string.
        - `generateGraphCode` constructs a LaTeX document as a string, embedding the function.
3. **PDFCreator** (`PDFCreator.h/cpp`):
    - **Purpose**: Converts LaTeX code into a PDF file.
    - **Implementation**:
        - Constructor initializes with LaTeX code.
        - `createPDF` method writes LaTeX to a temp file, executes `pdflatex`, and handles file cleanup.
4. **Main Application** (`main.cpp`):
    - **Purpose**: Orchestrates the application flow.
    - **Implementation**:
        - Parses command-line arguments for the input filename.
        - Integrates `FunctionReader`, `LaTeXGraphGenerator`, and `PDFCreator` to produce the output PDF.

Flow of Execution

1. **Input Processing**: The `FunctionReader` reads the function from the provided file.
2. **Graph Generation**: `LaTeXGraphGenerator` takes the function and generates LaTeX code.
3. **PDF Creation**: `PDFCreator` compiles the LaTeX code into a PDF file.

4. **Error Handling**: Each component includes error handling for robust operation.

Development Environment and Tools

- **IDE**: Developed using a C++ IDE for efficient coding and debugging.
- **LaTeX**: Used for graph generation within the PDF.
- **CMake**: Simplifies the build process across different systems.
- **Git**: Manages version control.

Building and Running the Application

Prerequisites

- C++ compiler (C++17 support).
- LaTeX distribution (e.g., TeX Live).
- CMake.

Compilation

1. Clone the repository and navigate to the project directory.
2. Create and navigate to a build directory.
3. Use CMake to configure and compile the project.

Execution

Run the application using `./Fun_to_graph <input_file_path>`.

Code Details and Architecture

`FunctionReader`

- **Responsibility**: Handles the reading and basic validation of the input function.
- **Key Methods**: `readFunction`, `validateFunction`.
- **Error Handling**: Throws runtime errors for file access and validation issues.

`LaTeXGraphGenerator`

- **Responsibility**: Translates the mathematical function into LaTeX code.
- **Key Method**: `generateGraphCode`.
- **LaTeX Integration**: Uses LaTeX commands to ensure accurate graphical representation.

`PDFCreator`

- **Responsibility**: Manages the creation of the PDF file from LaTeX code.

- **Key Method**: `createPDF`.
- **System Interaction**: Utilizes system calls to run LaTeX compilation tools.

Main Application

- **Responsibility**: Coordinates the workflow and manages user interactions.
- **Error Handling**: Catches and reports exceptions from other components.

Documentation and Submission

- **Source Code**: The complete source code is provided.
- **Compiled Program**: A compiled version is available for immediate use.
- **Sample Output**: A sample output PDF demonstrates the application's functionality.
- **Documentation**: This document details the design and usage.

Conclusion

"Function to Graph" successfully meets its requirements by offering a way to visualize mathematical functions. This comprehensive documentation aims to provide clear insights into the project's inner workings, making it accessible for everyone.