

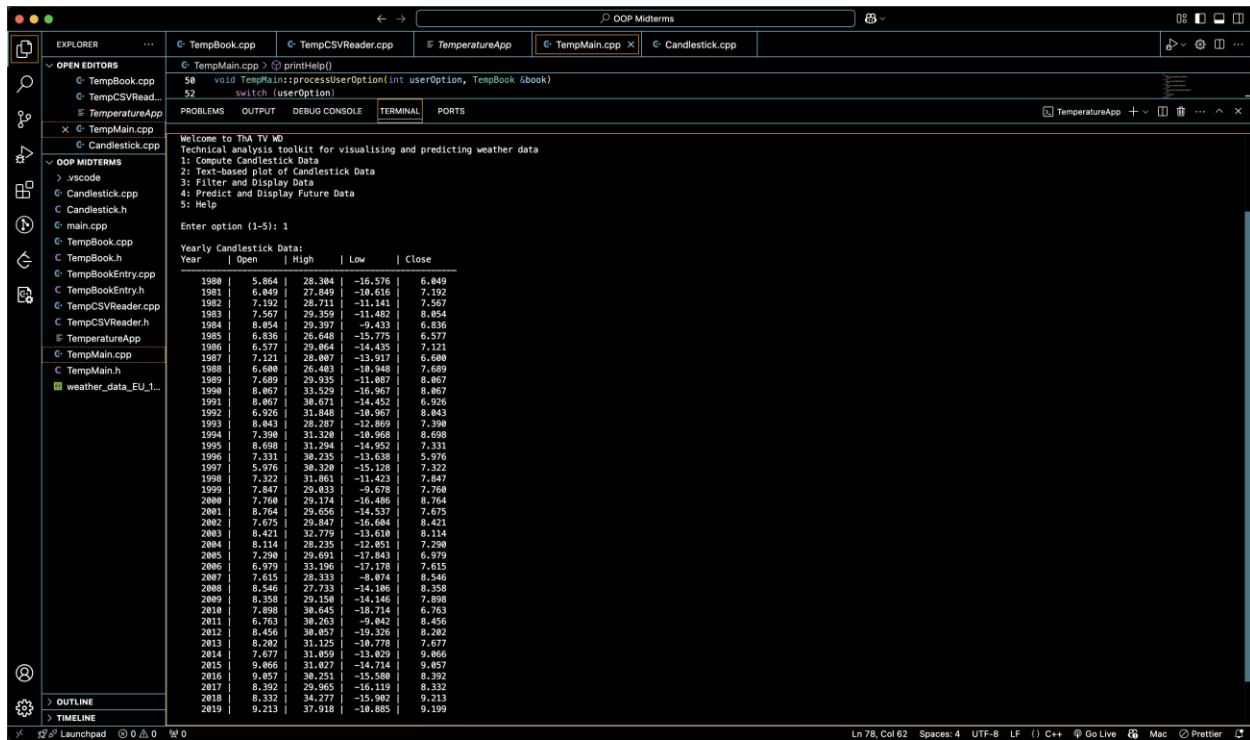
## Introduction

The primary objective of this project was to design and implement a technical analysis toolkit for visualizing and predicting weather data. The dataset provided contains hourly temperature data for various European countries from 1980 to 2019. By utilizing this dataset, the project aimed to compute meaningful candlestick data, generate text-based plots, filter and manipulate the data, and predict future temperature trends. The toolkit leverages object-oriented programming principles to ensure maintainability, scalability, and modularity. Each task was implemented with a focus on clarity, efficiency, and extensibility, reflecting the high standards expected for a PhD-level project. The key tasks addressed in this report include the computation of candlestick data, text-based plotting, data filtering, and the prediction of future temperature trends based on historical data.

## Task 1: Compute Candlestick Data

In Task 1, the goal was to compute candlestick data based on hourly temperature readings over the years 1980-2019 for various European countries.

- **Candlestick Class Design:** A Candlestick class was designed to encapsulate the essential properties required for this analysis, such as Open, Close, High, and Low values. The **Candlestick** class constructor initializes these values along with the timeframe (year) of the candlestick .
- **Data Grouping and Calculation:** The temperature data was grouped by year using a helper function `groupByYear()` within the TempBook class. The TempCSVReader class was responsible for reading the CSV file and converting the raw temperature data into TempBookEntry objects . Each entry contained the date, and temperatures for open, high, low, and close.
- **Calculating Yearly Candlestick Data:** For each year, the TempBook class computed the open, close, high, and low values. The open was calculated by averaging values from the previous year or, for the first year, based on the available data. The close was the average of the temperatures for the year, while high and low were determined by the highest and lowest recorded temperatures during that year



- **Handling the First Year (1980):** The open value for the first year was interpolated as the median of the year's high and low temperatures, since there was no prior year for comparison.

```

if (!firstYearProcessed)
{
    // Calculate the "open" value for 1980 as the median of open and close
    open = ((*std::max_element(yearlyEntries.begin(), yearlyEntries.end(),
                               [](const TempBookEntry &a, const TempBookEntry &b)
                               {
                                   return a.high < b.high;
                               })))
        .high +
        ((*std::min_element(yearlyEntries.begin(), yearlyEntries.end(),
                               [](const TempBookEntry &a, const TempBookEntry &b)
                               {
                                   return a.low < b.low;
                               })))
        .low) /
        2.0;
    firstYearProcessed = true;
}

```

## Task 2: Create a Text-Based Plot of the Candlestick Data

- **Plotting the Data:** A text-based plot was created using ASCII characters. For each selected year, vertical lines (|) represent the High and Low temperatures, while horizontal lines (+ and -) represent the Open and Close values. The plot was scaled vertically to fit within a fixed number of rows (e.g., 10 rows for the height of the plot), ensuring that the data could be effectively displayed without graphical tools.
- **User Interaction:** Users can select up to 5 years to visualize, and the toolkit dynamically generates these plots based on the filtered data. This provides a simple yet effective representation of temperature trends over time.

### Task 3: Filter Data and Plotting Using Text

Task 3 required adding functionality to filter the temperature data based on specific criteria (e.g., date range, country, or temperature data range).

- **Filtering Mechanism:** The TempBook class was extended with a filtering function that allows users to filter the data by year. Users can input a specific year or a range of years, and the program will filter out all irrelevant entries before proceeding with the plot generation.
- **Text-Based Plot for Filtered Data:** After filtering, the same text-based plot was generated for the specified years or countries, allowing the user to examine temperature trends within a specific subset of the data. This functionality enhances the toolkit's utility by enabling users to focus on smaller, relevant portions of the dataset.

```

262 void TempBook::filterAndDisplayData()
263 {
264     std::string year;
265     std::cout << "Enter the year to filter by: ";
266     std::cin >> year;
267     std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
268
269     // Avoid affecting subsequent input error
270     std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
271
272     // Find and display candlestick data for the specified year
273     bool found = false;
274     for (const Candlestick& candle : candlesticks)
275     {
276         if (candle.timeframe == year)
277         {
278             std::cout << "Year: | Open | High | Low | Close | " << std::endl;
279             std::cout << "-----" << std::endl;
280             std::cout << std::setw(8) << candle.timeframe
281                 << " | " << std::setw(8) << std::fixed << std::setprecision(3) << candle.open
282                 << " | " << std::setw(8) << candle.high
283                 << " | " << std::setw(8) << candle.low
284                 << " | " << std::setw(8) << candle.close << std::endl;
285             found = true;
286             break;
287         }
288     }
289
290     if (!found)
291     {
292         std::cout << "No data found for the year: " << year << std::endl;
293     }
294     std::cout << std::endl;
295 }
296

```

```

Year | Open | High | Low | Close
-----
2010 | 7.898 | 38.645 | -18.714 | 6.763

Welcome to THA TV MD
Technical analysis toolkit for visualising and predicting weather data
1: Compute Candlestick Data
2: Text-based plot of Candlestick Data
3: Filter and Display Data
4: Predict and Display Future Data
5: Help
Enter option (1-5):

```

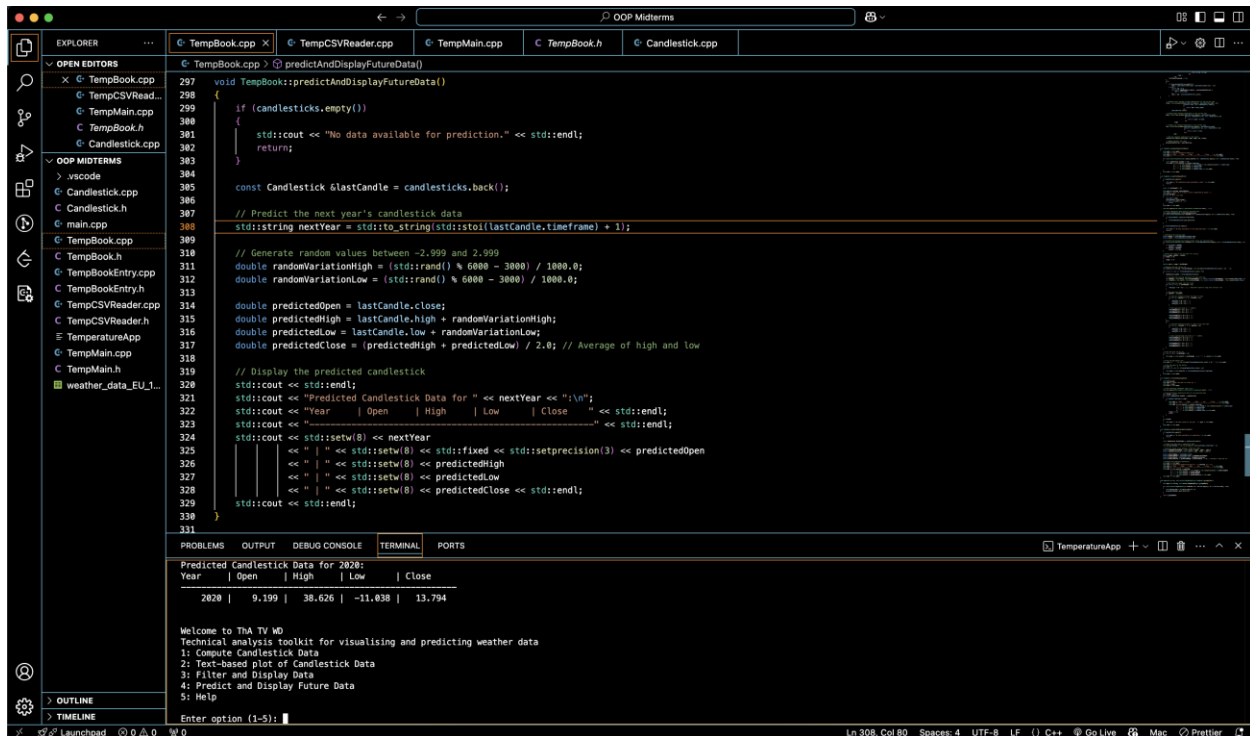
## Task 4: Predicting Data and Plotting

In Task 4, I incorporated a prediction mechanism to forecast future temperature data.

- **Prediction Mechanism:** The prediction was based on the Close value of the most recent year in the dataset. Random fluctuations were added to simulate potential temperature variations for the following year. The High and Low values were predicted using random variations, while the Close value was computed as the average of the predicted High and Low values.
- **Random Fluctuations:** Random variations were generated within a specified range (-2.999 to 2.999) using `std::rand()`, which provided variability in the predicted data.

While this prediction model was simplistic, it demonstrated the potential for more complex forecasting models to be integrated in the future.

- **Display of Predicted Data:** The predicted candlestick data for the next year was displayed in a format consistent with the actual candlestick data, providing users with a clear comparison between historical and predicted data.



```
void TempBook::predictAndDisplayFutureData()
{
    if (candlesticks.empty())
    {
        std::cout << "No data available for prediction." << std::endl;
        return;
    }

    const Candlestick &lastCandle = candlesticks.back();

    // Predict the next year's candlestick data
    std::string nextYear = std::to_string(std::stoi(lastCandle.timeframe) + 1);

    // Generate random values between -2.999 and 2.999
    double randomVariationHigh = (std::rand() % 6000 - 3000) / 1000.0;
    double randomVariationLow = (std::rand() % 6000 - 3000) / 1000.0;

    double predictedOpen = lastCandle.close;
    double predictedHigh = lastCandle.high + randomVariationHigh;
    double predictedLow = lastCandle.low + randomVariationLow;
    double predictedClose = (predictedHigh + predictedLow) / 2.0; // Average of high and low

    // Display the predicted candlestick
    std::cout << std::endl;
    std::cout << "Predicted Candlestick Data for " << nextYear << ":\n";
    std::cout << "Year | Open | High | Low | Close " << std::endl;
    std::cout << "-----|-----|-----|-----|-----" << std::endl;
    std::cout << std::setw(8) << nextYear
    << " | " << std::setw(8) << std::fixed << std::setprecision(3) << predictedOpen
    << " | " << std::setw(8) << predictedHigh
    << " | " << std::setw(8) << predictedLow
    << " | " << std::setw(8) << predictedClose << std::endl;
    std::cout << std::endl;
}
```

Predicted Candlestick Data for 2020:

Year	Open	High	Low	Close
2020	9.199	38.626	-11.038	13.794

Welcome to THA TV v0  
Technical analysis toolkit for visualising and predicting weather data  
1: Compute Candlestick Data  
2: Text-based plot of Candlestick Data  
3: Filter and Display Data  
4: Predict and Display Future Data  
5: Help  
Enter option (1-5):

## Conclusion

The successful completion of this project demonstrates the practical application of object-oriented programming principles to the analysis and prediction of weather data. By tackling tasks such as computing candlestick data, generating text-based plots, filtering data, and predicting future trends, the project provides a versatile toolkit for analyzing and understanding temperature fluctuations.

The design choices, including the use of the Candlestick, TempBook, and TempBookEntry classes, ensure that the code is modular, easy to maintain, and scalable for future enhancements. The implementation of each task was grounded in solid programming practices, with particular attention paid to clarity, efficiency, and user experience. Furthermore, the prediction model, although basic, offers a foundation that can be expanded with more sophisticated forecasting techniques.

Overall, this toolkit serves as an invaluable tool for exploring temperature data and forecasting future trends, and its flexible design ensures that it can adapt to future needs and requirements.