Project Report - Team G

MNXB11/NAFY018 - Physics: Introduction to Programming and Computing for Scientists

Albin, Atle & Ebbe

Our goal

In our second attempt at this project, our main goal was to make things simpler and ensure everything worked smoothly as planned. We wanted to create a chart showing the temperature in Lund from 1780 to 2023, comparing it year by year.

As the chart took shape, it revealed interesting details about Lund's climate history. The ups and downs in temperature showed us how different weather factors played out over time, giving us a better understanding of the city's climate

Our workflow (Simplified)

Data preparation:
Identify and download CSV parsing Library
Extract and clean Raw Data
Convert SMHI datasets to CSV format, addressing formatting issues.

Data extraction and cleaning:
Develop C++ code for data extraction
Extract targeted data from cleaned datasets and store output in vectors.

Data analysis:
Decide on three analyses
Create a histogram
Use ROOT to visualize analysis results in the histogram

Documentation:
Regularly update workplan.md
Project report in LaTeX
Document goals, results, code descriptions, and plots.

(README.md provides instructions on compilation, running, and reproducing results.)

Read “Workplan” in GitHub for a more detailed explanation!

Example of our use of vectors for storing data

In the code we developed, we opted for vectors over TTrees due to the ease of handling and manipulating data. Vectors provide a convenient and efficient way to represent and operate on numerical data, making mathematical operations and transformations straightforward. This choice allows for simpler code implementation and facilitates tasks such as vector addition, multiplication, and other linear algebra operations.

On the other hand, TTrees, or ternary trees, are hierarchical data structures that may be more suitable for certain types of problems, particularly those involving hierarchical relationships or recursive structures. However, in our specific context, the simplicity and efficiency of vector representations better suited our computational needs, streamlining the development process and enhancing code readability. However since we only needed to create one histogram we opted for the use of vectors.

```
for (size_t i = 0; i < years.size(); ++i) {
hist->Fill(years[i], temperatures[i]);
}
```

In this example we can simply call on the vector and use a simple for loop to run through every year and get the temperatures.

A problem that we encountered during the process of trying to extract the data from the csv file was the missed difference between columns and row. When we first tried to get the data out we told the program to take the third row and put it as temperature. But the third row was actually the day. This was solved when we realized that instead of trying to get the year, month and so on directly we could split the data up and then get the desired data. The way we did this was using vectors to put the data in and then we could get the data from the vector instead.

Our approach for this project was a more narrow scope than last time. This way we hoped to execute it better, quality over quantity. We focused only on Lund data and to produce only one plot of how the temperature changed over time. First we extracted the values date, time, temperature and grade. We used a string for the date and time to later break them up into days, months, years and hours, minutes, sec respectively. We decided to use vectors to store the variables since they are able to hold a lot of values. We then looped and for every day we matched it to the temperature and filled the histogram.

We successfully made a Histogram that displayed the temperatures in Lund over the time period. The histogram does however have some missing data since we did not do any control of the values that could have prevented this.