Puneet Sandher (1174249)

Ada Calendar Reflection

Ada is a programming language released in 1980 for the United States Defense Department and was useful for real-time applications and in mission critical. There are many benefits and limitations to using Ada. Its strengths are modularity, type-checking, readability, data abstraction, exception handling, and tasking. During the testing process of the development of this program, I found that there are many range and overflow checks in Ada compilers making it much easier and descriptive to make a high-quality program. Ada's modularity allows easier development to scale programs, program maintenance, and large teams working on the same program. However, Ada's complexity and rigidness is difficult, which can deter developers from using this language for projects compared to other programming languages available. For example, in my program I had difficulty creating a string variable that could be of any size or a maximum size, I resorted to using an unbound string but the code was unnecessarily clunky. Ada also makes it complex to work with large codebases due to its thorough error checking and it has many compatibility issues with modern tools.

Ada was well suited to solve this problem because of its modularity, reliability, strong typing, readability and portability. My program had many variables to store various arrays (calendardate, months, headers, banners) and other variables, and it helped prevent bugs when storing data and displaying it. By using Ada, my program had multiple subprograms for handling user input (readcalinfo), building and storing the calendar (buildcalendar), and displaying the various components of the output (printrowheading, printrowmonth, and banner). Ada's syntax is readable and will be easy for others to understand my code, in addition, when I'm reviewing my code I have a clear understanding of what it is. This program is portable as it can be used on different operating systems and platforms. Lastly, Ada was used for various important applications with an emphasis on reliability, thus it works great for long-intensive programs like this one where many calculations and data storage is important.

Design Process

User Input and Validation: My program asks for language preference and year to display (readcalinfo), which is error proof by using exceptions in Ada specifically, data_error. In addition, Ada's ability to specify ranges for integers. A separate subprogram (isvalid) was called to validate the user inputted error.

Storing Calendar Data: The calendar display depicts week *i* for three months in a row, it is not possible to print each month at a time. By using a 2D array to store the calendar days that are populated using nested for loops, with various conditions to consider all the possible cases which is all done in the subprogram buildcalendar(). Storing in a 2D array required specific array indexing, as the loops represented each row and columns, but each month is populated all at once since you don't know the start day of the next month without the previous' end date. buildcalendar() calls the numdaysinmonth to store the correct days in the array. Numdaysinmonth calls leapyear to determine how many days are in each month for a specific year.

Banner: The easiest way to develop the banner was to create a data file that had the font of each digit from 0 to 9 where they were all the same length and width. While it is possible to dynamically create the font for each digit in a subprogram, it unnecessarily complicates the program compared to having an additional file to run the program. Storing each line of this data into a 2D array, required a consistent string length due to Ada's rigidness with types, thus some fonts have extra trailing spaces in each row, so each row is nine characters long. Using this array and nested loops, I can print the row of each digit together, as seen in the subprogram banner().

Displaying Calendar: As seen in the printrowheading subprogram, this required string arrays to store the months and days in both English and French. Each month has a different length, which made it difficult to store as it required the use of the unbounded_string. This requires additional libraries and the development of the stringconvert subprogram to convert a string to an unbounded string. Each row displays three months, thus the subprogram takes a parameter specifying which row to print the header for. Specific spacing was needed to ensure the months and days of week were aligned. This program took the parameters for language choice and an integer of row to state which headers to print. Printrowmonth prints the 2D calendar array, with consistent spacing to have the calendar format of 3 months per row and calls the printrowheading to get the headings of the next row of months.