Design

General Overview of the design:

- main ()
    - Uses argv and argc to get input from command line.
    - Checks for required input.
    - Loops through given files.
    - Loops till everything is read from file or hit the number of lines wanted
    - Loops through buffer read from file

Main

- Checks if there were any arguments given
- Sets the wanted lines given through argument
- Checks if number if veiled by converting the number given back to string
    - If they aren't the same program will throw an error that the number is invalid.
- Checks if there was any file given if not, it will automatically set file to "-" to read from std input.
- Loops through the given files.
    - Sets int file to file given or Std in if "-" is given or no filename is specified.
    - Checks if program was able to open file if not it will throw an error.
    - Loops through the file till it hit the number of lines given
        - Sets the buffer of size 1000 to prevent timeout when reading big files.
        - Reads file to int temp storing the read bytes in buffer
        - If file is at the end meaning it is at the end of file break the for loop and go to next file and free buffers
        - Sets a temporary buffer of size 1000 this, will store all the bytes till the last new line, unless actual buffer is filled and the last new line is not in it
        - Loop through the buffer
            - If new line "/n" is in the buffer increment j
            - If j == the amount of lines write the temp buffer to standard output
            - And break the loop
        - If lines is still smaller than the amount of lines requested then write the buffer to standard out.
        - Free buffers
    - Close the current file that was opened

Buffer size:

The reasoning behind choosing the size for the buffer was that 1 byte is way to slow. I did some testing and 100 works with the test cases. I just made it way bigger because this allows for a faster code, decreasing the runtime, pretty significantly.

Temp buffer:

The reasoning behind the buffer is that when having buffer that is not just one byte there is a chance of having multiple new lines. So when looking for new lines within the buffer, if the buffer has the last new line in the middle of the buffer, we need everything up to that written to standard output. So

everything is getting copied to temp buffer and if the case happens that the new line is in the middle, the temp buffer get written out instead of the whole buffer.

To check if the number is valid:

To check if the number is a valid number, which is 0 or higher and not a letter, I converted the integer that I converted back to a string to compare with the original string that contained the number. This is to prevent inputs like 1a. Since when using atoi the number is 1 eventhough the input was 1a.

Errors:

To check for errors and print them, I used warn.