Assignment 5 - Processing in Linear Time

Due: 02-12-2020 (Wednesday @ 3:30 p.m.)

Necessary Files

File	Description	Location
`dict_w_defs.json`	Json input file	Resources/04-Data
`json.hpp`	Json class written by nlohmann	Resources/03-Json
`JsonFacade.hpp`	Json helper class I wrote to assist with the `json.hpp` class	Resources/03-Json
`Timer.hpp`	Timer helper class	Resources/05-Timing
`read_dict.cpp`	Example json reader with some timing.	This folder

Background

Json

- We are going to use `Javascript Object Notation` or JSON as our input data format.
- JSON is a nice resume builder, and used everywhere in industry as a platform independent data exchange format.
- For a quick intro look here
- I will also provide some example code that reads in a json file with this assignment.
- I wrote my JsonFacade class (which is a wrapper around `json.hpp`) to simplify the functionality of nlohmanns class. He did a great job and I'm trying to filter only what we need. Therefore I would appreciate lots of feedback on how to make it better and or simpler.

Timing

- Timing becomes important when you want to benchmark how fast code is running.
- There are many things that effect run times, so you should try to run your code with the same conditions (like the same machine) as much as possible.
- The library here will give us milli-second granularity. So go look at the example code.
- Check this out.

Getching 😃

- Getch: a word that implies the capture of keyboard input, with hitting the enter key and optionally not even reflecting on the console that anything happened.
- This is obvious with games, since not all key strokes imply an attempt to type, they may be trying to control movement or communication in other ways.
- The function here provides a `getch` function for both windows or linux / osx.
- See example here

Assignment

Requirements Part 1

- Write a program that will read in a dictionary file from dict_w_defs.json and store it in a vector.
- Time how long it takes to load
- After your dictionary is loaded, we are going to perform "autosuggestions" when a user types characters at the console.
- Suggestions will not start until X number of characters are typed, and at most N possible suggestions will be displayed.
- The time it takes to find each suggestion will be displayed in milliseconds (somewhere).
- In addition, Jeremey Glebe has a library called TermIO which gives us a decent amount of control over the standard console.
- I highly recommend you use this to make your program work to its fullest. I will get him to give us an overview.

```
digraph finite_state_machine {
    rankdir=LR;
    size="8,5"

    node [shape = doublecircle]; S;
    node [shape = point ]; qi

    node [shape = circle];
    qi -> S;
    S  -> q1 [ label = "a" ];
    S  -> S  [ label = "a" ];
    q1 -> S  [ label = "a" ];
    q1 -> q2 [ label = "ddb" ];
    q2 -> q1 [ label = "b" ];
    q2 -> q2 [ label = "b" ];
}
```