```
Script started on 2023-06-22 18:09:14-05:00 [TERM="xterm" TTY="/dev/pts/7" COLUMNS=
bj94684@ares:~$ pwd
/home/students/bj94684
bj94684@ares:~$ cat sen.info
/*****************************************************************************
 *                                                                           *
 *  NAME:  Jose Barron                          CLASS:  CSC122-002           *
 *                                                                           *
 *         Lab:  Say WHAT?                      Level:   4                   *
 *       Option: Improvements on finding syll.  Level: + 1.0                 *
 *                                          Total Level:   5.0               *
 * This program is intended to calculate the readibilty index of given       *
 * sentences inputted sentences. This program is able to tell how many       *
 * sentences, words, and syllables were inputted through the use of my       *
 * custom strextra.h library and helper function within the code. The        *
 * program does the searching word for word, rather than sentence for        *
 * sentence to be able to detect when a word has an adjacent vowel or        *
 * vowel or both or neither and then the syllable count is properly          *
 * counted for each case. With the information gathered by the program       *
 * from the user, the program is able to calculate the readibilty index      *
 * for the user to the closest whole number.                                 *
 *                                                                           *
 *****************************************************************************/


bj94684@ares:~$ show-code strextra.h


strextra.h:
```

```
 1  #ifndef STREXTRA_H_INC
 2  #define STREXTRA_H_INC
 3
 4  #include<iostream>
 5  #include<string>
 6  #include<vector>
 7
 8  using namespace std;
 9
10  // Helper function that tells caller the location of a character in a
11  // string then stores those location in a vector. If the character is not
12  // in the string then the vector remains empty which helps caller
13  // identify when no character is in a string.
14  inline vector<string::size_type> num_in( string s, char t)
15  {
16      vector<string::size_type> num;
17      for (string::size_type pos = 0; pos != s.length(); ++pos)
18      {
19          if (static_cast<char>( s[pos] ) == t )
20          {
21              num.push_back(pos);
22          }
23      }
24      return num;
25  }
26
27  // Helper function that tells caller if the whole string t, is in
28  // string s from two given locations in s and it returns a bool
29  // for caller to user.
30
31  inline bool incl(string s, string t, string::size_type beg,
32                                       string::size_type end)
33  {
34      string nr = s.substr(beg, end-beg+1);
35      bool tf;
36      if ( nr == t)
37      {
38          return tf = true;
39      }
40      return tf = false;
41  }
42
43  // Helper function that takes two vectors of positions in a string
44  // which in this function are the positions of s, then compares
45  // each of those combinations of positions for s to a given string
46  // and if they match then the location of the first vector of position.
47  // to tell the caller where the given string and the string match
48  //
49
50  inline string::size_type com(vector<string::size_type> b, vector<string::s
51                   string s, string t)
52  {
53      string::size_type num;
54      for(auto p : b)
55      {
56          for(auto p1 : e)
57          {
58              if ( p > p1)    // The starter position can never be greater
59              {               // than the end position
60                  num;
61              }
62              else
63              {
64                  bool ys = incl(s, t, p, p1);
65                  if ( ys)
66                  {
67                      return num = p;
68                  }
69                  num;
70              }
71          }
72      }
73      return num = s.length();
74  }
75
76  // Helper function that takes two vectors of positions in a string
77  // which in this function are the positions of s, then compares
```

```
// each of those combinations of positions for s to a given string
// and each time they match then the num becomes bigger by 1. Returns
// zero if no matches occurs or returns the number of times matches
// occured.

inline short com1(vector<string::size_type> b, vector<string::size_type> e,
                  string s, string t)
{
    short num;
    for(auto p : b)
    {
        for(auto p1 : e)
        {
            if ( p > p1)
            {
                num;
            }
            else
            {
                bool ys = incl(s, t, p, p1);
                if ( ys)
                {
                    ++num;
                }
                num;
            }
        }
    }
    return num;
}

// Helper function tells if string t is in string s and returns the result
// through a bool.

inline bool find_str_in(string s, string t)
{
    bool found;
    vector<string::size_type> pos = num_in(s, t[0]);
    vector<string::size_type> pos_b = num_in(s, t[ t.length() - 1]);
    if ( ! pos.empty() && ! pos_b.empty())
    {
        short ys = com1( pos, pos_b, s, t);
        if ( ys != 0)
        {
            return found = true;
        }
        else
        {
            return found = false;
        }
    }
    else
    {
        return found = false;
    }
}

// Helper function tells if string t is in string s and returns the result
// by giving the location on where it occured or if it didnt occur
// then it returns the size of string s to indicate to the caller that
// string t is not in string s.

inline string::size_type find_str_loc(string s, string t)
{
    string::size_type loc;
    vector<string::size_type> pos = num_in(s, t[0]);
    vector<string::size_type> pos_b = num_in(s, t[ t.length() - 1]);
    if ( ! pos.empty() && ! pos_b.empty())
    {
        string::size_type ys = com(pos, pos_b, s, t);
        if ( ys != s.length() )
        {
            return loc = ys;
        }
        else
        {
            return loc = s.length();
        }
    }
    else
    {
        return loc = s.length();
    }
}

inline short find_str(string s, string t)
{
    short loc = 0;
    vector<string::size_type> pos = num_in(s, t[0]);
    vector<string::size_type> pos_b = num_in(s, t[ t.length() - 1]);
    if ( ! pos.empty() && ! pos_b.empty())
    {
        string::size_type ys = com(pos, pos_b, s, t);
        if ( ys != s.length() )
        {
            ++loc;
        }
    }
    return loc;

}

// Helper function that combines each possible combination of two vectors
// and returns that vector. In the future, I could template it to
// fit any data type just not strings.

inline vector<string>combine(vector<string> b, vector<string> e)
{
```

```
    vector <string> com;
    for(auto p : b)
    {
        for(auto p1 : e)
        {
            com.push_back( p + p1 );
        }
    }
    return com;
}

// Helper function that tells caller if a char is in a string returns
// results as a bool.

inline bool find_char_in( string s, char t)
{
    bool in;
    vector<string::size_type> t_in = num_in(s, t);
    if ( ! t_in.empty())
    {
        return in = true;
    }
    else
    {
        return in = false;
    }
}

// Helper function that tells the caller the location of where char t
// is in string s. Returns a vector of locations or a empty vector if
// the char was not found in t.

inline vector<string::size_type> find_char_loc( string s, char t)
{
    vector<string::size_type> in;
    vector<string::size_type> t_in = num_in(s, t);
    if ( ! t_in.empty())
    {
        for (auto p : t_in)
        {
            in.push_back(p);
        }
        return t_in;
    }
    else
    {
        return in;
    }
}

// Helper function that compares two string. Returns 1 if true
// and 0 if wrong. This function can also be templated for
// all data types not just strings and could also be a bool.

inline short compare(string s, string t)
{
    short n=0;
    if (s == t)
    {
        return ++n;
    }
    return n;
}


#endif
```

```
bj94684@ares:~$ show-code sen.cpp
```

sen.cpp:

```
#include<iostream>
#include<string>
#include<vector>
#include<limits>
#include<cmath>
#include"strextra.h"

using namespace std;

// helper function that gets a char from user, keeps looping until the use
// inputs any string
inline char get_char(void)
{
    char t;
    while ( cin.peek() == '\n' )
    {
        cin.ignore();
        cout << "\n[INVALID]Enter the char again;   ";
        cout.flush();
    }
    cin >> t;
    return t;
}
// helper function that gets a line from user, keeps looping until the use
// inputs any string
inline string get_line(void)
{
    string line;
    while ( cin.peek() == '\n')
    {
        cin.ignore();
        cout << "\n[INVALID]Enter a line of text again;   ";
        cout.flush();
    }
    getline(cin, line);

```

```
    return line;
}

// helper function that extracts a substring of a string with given
// parameters by caller.
inline string included(string s, string::size_type beg = 0, string::size_ty
{
    string nr = s.substr(beg, end-beg+1);
    return nr;
}

/* used this helper function to make the one below
inline void show_char_loc( string s, char t)
{
    vector<string::size_type> in;
    vector<string::size_type> t_in = num_in(s, t);
    string::size_type beg = 0;
    if ( ! t_in.empty())
    {
        string nr = included(s, beg, t_in[0] );
        cout << "\nThe words in your string are " << nr << ' ';
        for (vector<string::size_type>::size_type p=0; p + 1 != t_in.size(
             ++p)
        {
            string w = included( s, t_in[p], t_in[static_cast<string::size_
            cout << w << ' ';
        }
        string lw = included(s, t_in.back(), s.length() - 1);
        cout << lw << '\n';
    }
    else
    {
        cout << "\nNo spaces in word so 1 word";
    }
}
*/
// helper function used to extract all the words in a sentences to a
// new vector of strings which contains all words.
inline vector<string> find_word( string s, char t)
{
    vector<string::size_type> in;
    vector<string::size_type> t_in = num_in(s, t);
    string::size_type beg = 0;
    vector<string>words;
    if ( ! t_in.empty())
    {
        string nr = included(s, beg, t_in[0] );
        words.push_back(nr);

        for (vector<string::size_type>::size_type p=0; p + 1 != t_in.size(
             ++p)
        {
            string w = included( s, t_in[p], t_in[static_cast<string::size_
            words.push_back(w);
        }
        string lw = included(s, t_in.back(), s.length() - 1);
        words.push_back(lw);
        return words;
    }
    else
    {
        return words;
    }
}

// helper function that tells caller how much times a char was found in a
// given string
inline short find_char( string s, char t)
{
    short num = 0;
    for (string::size_type pos = 0; pos != s.length(); ++pos)
    {
        if (static_cast<char>( s[pos] ) == t )
        {
            ++num;
        }
    }
    return num;
}


int main ()
{

    cout << "\n\t\tWelcome to the Readibility Index Program\n\n";
    char choice;
    bool done;
    done = false;
    vector<string> p;
    cout <<"Insert Sentence: ";
    string s = get_line();
    p.push_back(s);
    while ( ! done ) // loop to for user to keep inputting sentences
    {
        cout << "\nDo you want to insert another sentence?  ";
        choice = get_char();
        choice = static_cast<char>( toupper( choice ) );
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        if ( choice == 'Y')
        {
            cout <<"\nInsert Sentence: ";
            string l = get_line();
            p.push_back(l); // a vector of string where each string is
                            // each sentence inputted by user
        }
        else if ( choice == 'N')
        {
            done = true;
```

```
        }
        else
        {
            cout << " You did not input any correct answer."
                 << " Please Try Again\n";
        }
    }
    short num_of_sentences = static_cast<short>( p.size() );

    vector<string> o = {"a","e", "i", "o", "u"};
    vector<string> o1 = {"a","e", "i", "o", "u"};
    vector<string> a_p = combine(o, o1);   // a vector of all possible
                                           // adjacent vowels
    vector<char> vowels = {'a', 'e', 'i', 'o', 'u'};

    // This loops through all sentences and the number of words
    // is the size of the vector of words which is pretty straighforward.
    // It can also detect if the line is only one word.

    short num_of_words = 0;
    for (auto i:p)
    {
        vector<string>word = find_word(i, ' ');
        if ( ! word.empty())
        {
            num_of_words += static_cast<short>( word.size() );
        }
        else
        {
            num_of_words += 1;
        }
    }

    // This huge loop goes through one sentence then through all its
    // words then it searches each word for vowels or adjacent vowels.
    // It is able to detect if one word has an adjacent vowel and vowel
    // or either or neither and for each of these cases it adds to the
    // number of syllables accordingly. It can also detect if the letter e
    // is the last word of the sentence but not each word.

    short num_of_v = 0;

    for (auto i:p)
    {
        vector<string>word = find_word(i, ' ');
        if ( ! word.empty())
        {
            for (auto w: word)
            {
                short num_of_v_word = 0;
                short num_of_av_word = 0;
                short total = 0;
                for(auto x: vowels)
                {
                    short v = find_char(w,x);
                    num_of_v_word += v;
                }
                for(auto y: a_p)
                {
                    short av = find_str(w,y);
                    num_of_av_word += av;
                }
                total = static_cast<short>( fabs(num_of_v_word-num_of_av_w
                if (w[w.length() - 1] == 'e') // if last letter is e, then
                {                             // loop
                    if( total >= 1)           // This loop only does
                    {                         // anything if the total is
                        total = total - 1;    // equal to 1 or greater
                    }                         // to ensure no neg totals.
                }
                if(total == 0) // if there is a adjacent vowel and vowel
                {              // then total is zero but every word has
                    total = total + 1; // atleast one syllable which this
                }                      // if loop guarantees.
                num_of_v += total;
            }
        }
        else // if there is only word in the sentence
        {
            string wrd = included(i,0,i.length()-1);
            short num_of_v_word = 0;
            short num_of_av_word = 0;
            short total;
            for(auto r: vowels)
            {
                short v = find_char(wrd,r);
                num_of_v_word += v;
            }
            for(auto y: a_p)
            {
                short av = find_str(wrd,y);
                num_of_av_word += av;
            }
            total = static_cast<short>( fabs(num_of_v_word-num_of_av_word)
            if (wrd[wrd.length() - 1] == 'e')
            {
                if( total >= 1)
                {
                    total = total - 1;
                }
            }
            if(total == 0)
            {
                total = total + 1;
            }
            num_of_v += total;
        }
    }
```

```
   253      cout << "\nNumber of Sentences: " << num_of_sentences;
   254      cout << "\nNumber of Words: " << num_of_words;
   255      cout << "\nNumber of Syllables: " << num_of_v;
   256      double one = static_cast<double>(num_of_v)/static_cast<double>(num_of_
   257      double two = static_cast<double>(num_of_words)/static_cast<double>(num
   258      double index = round(206.835-84.6*one-1.015*two);
   259      cout << "\nThe Readibility Index is " << index << '\n';
   260
   261      return 0;
   262  }
bj94684@ares:~$ CPP sen
sen.cpp***
In file included from sen.cpp:6:
strextra.h: In function
‘std::__cxx11::basic_string<char>::size_type
com(std::vector<long unsigned int>, std::vector<long unsigned int>,
std::string, std::string)’:
strextra.h:60:17: warning: statement
has no effect [-Wunused-value]
   60 |                 num;
      |                 ^~~
strextra.h:69:17: warning: statement
has no effect [-Wunused-value]
   69 |                 num;
      |                 ^~~
strextra.h: In function ‘short int
com1(std::vector<long unsigned int>, std::vector<long unsigned int>,
std::string, std::string)’:
strextra.h:93:17: warning: statement
has no effect [-Wunused-value]
   93 |                 num;
      |                 ^~~
strextra.h:102:17: warning: statement
has no effect [-Wunused-value]
  102 |                 num;
      |                 ^~~


bj94684@ares:~$ ./sen.out

               Welcome to the Readibility Index Program

Insert Sentence: i went to the store

Do you want to insert another sentence?  yes

Insert Sentence: to eat food and be

Do you want to insert another sentence?  yes

Insert Sentence: satisfied for more

Do you want to insert another sentence?  yes

Insert Sentence: yessir

Do you want to insert another sentence?  no

Number of Sentences: 4
Number of Words: 14
Number of Syllables: 17
The Readibility Index is 101
bj94684@ares:~$ ./sen.out

               Welcome to the Readibility Index Program

Insert Sentence: ind in foode

Do you want to insert another sentence?  jamal went home for work
 You did not input any correct answer. Please Try Again

Do you want to insert another sentence?  yes

Insert Sentence: jamal went home for work

Do you want to insert another sentence?  yes

Insert Sentence: he needed food

Do you want to insert another sentence?  no

Number of Sentences: 3
Number of Words: 11
Number of Syllables: 14
The Readibility Index is 95
bj94684@ares:~$ ./sen.out

               Welcome to the Readibility Index Program

Insert Sentence: i went tom store cuz hungry

Do you want to insert another sentence?  no

Number of Sentences: 1
Number of Words: 6
Number of Syllables: 7
The Readibility Index is 102
bj94684@ares:~$ exit
exit

Script done on 2023-06-22 18:15:42-05:00 [COMMAND_EXIT_CODE="0"]
```