

```

Script started on 2023-06-22 19:12:07-05:00 [TERM="xterm" TTY="/dev/pts/8" COLUMNS=
bj94684@ares:~$ pwd
/home/students/bj94684
bj94684@ares:~$ cat find.info
/*****
*
* NAME: Jose Barron CLASS: CSC122-002
*
* Lab: Now where did I put that? Level: 2
* Option: Level: + 0
* Total Level: 2.0
*
* This program is designed to find a char or a string within a string.
* All this information is provided by the user which is validated by the
* get_char or get_str from the custom library input_prot.h. After the
* the program receives the information, it uses either the find_char_loc
* or the find_str_loc function, in the custom library strextra.h, to
* find the given char or string in the given string. This custom library
* can be used for more than simply one string, but can be used to look
* for a string in a vector/array or a list of strings.
*
*****/
bj94684@ares:~$ show-code input_prot.h

```

input\_prot.h:

```

1 #ifndef INPUT_PROT_H_INC
2 #define INPUT_PROT_H_INC
3
4 #include<iostream>
5 #include<string>
6 #include<limits>
7
8 using namespace std;
9
10
11 //This first function gets a string, and makes sure something written
12 // since a string can be pretty much anything but empty
13
14 inline string get_str(string prompt = "\nInput a string: ",
15 string error_msg = "\nNo string detected. Try again: ");
16 {
17     cout << prompt;
18     cout.flush();
19     string line;
20     while ( cin.peek() == '\n' )
21     {
22         cin.ignore();
23         cout << error_msg;
24         cout.flush();
25     }
26 }

```

```

27     getline(cin, line);
28
29     return line;
30 }
31
32 //This is a simple get char function with no parameters
33
34 inline char get_ch(string prompt = "\nInput a char: ",
35 string error_msg = "\nNo char detected. Try again: ")
36 {
37     cout << prompt;
38     cout.flush();
39     char t;
40     while ( cin.peek() == '\n' )
41     {
42         cin.ignore();
43         cout << error_msg;
44         cout.flush();
45     }
46     cin >> t;
47     return t;
48 }
49
50 //This is a more advanced get char since it forces the char to be
51 //one of the chars provided by the caller, and all other functions follow
52 //this similar function. They all force the certain data type to be one
53 //the caller wants. If caller wants abort, and the user aborts then
54 //the functions return a value that is not what the caller wants, so the
55 //the caller can easily identify when the user aborts
56
57 inline char get_ch_w_p(string prompt = "\nInput a char: ",
58 string error_msg = "\n Incorrect char. Try again: ",
59 string parameter = "yYnN", bool abort = false)
60 {
61     char t;
62     bool done = false;
63     cout << prompt;
64     while ( cin.peek() == '\n' )
65     {
66         cin.ignore();
67         cout << error_msg;
68         cout.flush();
69     }
70     while (!done)
71     {
72         cin >> t;
73         for (string::size_type p = 0; p != parameter.length(); ++p)
74         {
75             if (parameter[p] == t)
76             {
77                 done = true;
78                 return t;
79             }
80 }

```

```

81     }
82 }
83 cout << error_msg;
84 if (abort)
85 {
86     cout << "or You can abort by typing q: ";
87     char a;
88     cin >> a;
89     if (a == 'q')
90     {
91         done = true;
92         return t = '/';
93     }
94 }
95 }
96 return t;
97 }
98
99 inline double get_d_w_BR(string prompt,
100                          string error_msg = "\nNumber is not in range. Try
101                          double L = 0, double U = 100, bool abort = false)
102 {
103     cout << prompt;
104     cout.flush();
105     double num;
106     while ( cin.peek() == '\n' )
107     {
108         cin.ignore();
109         cout << error_msg;
110         cout.flush();
111     }
112     bool done = false;
113     while ( ! done)
114     {
115         cin >> num;
116         if ( num > L && num < U)
117         {
118             done = true;
119             return num;
120         }
121         cout << error_msg;
122         if (abort)
123         {
124             cout << "or You can abort by typing q: ";
125             char a;
126             cin >> a;
127             if (a == 'q')
128             {
129                 done = true;
130                 return num = L-1;
131             }
132         }
133     }
134     return num;

```

```

135 }
136
137 inline double get_d_w_LR(string prompt,
138                          string error_msg = "\nNumber is not in lower end (
139                          double L = 0, bool abort = false)
140 {
141     cout << prompt;
142     cout.flush();
143     double num;
144     while ( cin.peek() == '\n' )
145     {
146         cin.ignore();
147         cout << error_msg;
148         cout.flush();
149     }
150     bool done = false;
151     while ( ! done)
152     {
153         cin >> num;
154         if ( num > L)
155         {
156             done = true;
157             return num;
158         }
159         cout << error_msg;
160         if (abort)
161         {
162             cout << "or You can abort by typing q: ";
163             char a;
164             cin >> a;
165             if (a == 'q')
166             {
167                 done = true;
168                 return num = L-1;
169             }
170         }
171     }
172     return num;
173 }
174
175 inline double get_d_w_UR(string prompt,
176                          string error_msg = "\nNumber is not in upper end (
177                          double U = 100, bool abort = false)
178 {
179     cout << prompt;
180     cout.flush();
181     double num;
182     while ( cin.peek() == '\n' )
183     {
184         cin.ignore();
185         cout << error_msg;
186         cout.flush();
187     }
188     bool done = false;

```

```

189 while ( ! done)
190 {
191     cin >> num;
192     if (num < U)
193     {
194         done = true;
195         return num;
196     }
197     cout << error_msg;
198     if (abort)
199     {
200         cout << "or You can abort by typing q: ";
201         char a;
202         cin >> a;
203         if (a == 'q')
204         {
205             done = true;
206             return num = U+1;
207         }
208     }
209 }
210 return num;
211 }
212 inline long get_l_w_BR(string prompt,
213                        string error_msg = "\nNumber is not in range. Try a",
214                        long L = 0, long U = 100, bool abort = false)
215 {
216     cout << prompt;
217     cout.flush();
218     long num;
219     while ( cin.peek() == '\n' )
220     {
221         cin.ignore();
222         cout << error_msg;
223         cout.flush();
224     }
225     bool done = false;
226     while ( ! done)
227     {
228         cin >> num;
229         if ( num > L && num < U)
230         {
231             done = true;
232             return num;
233         }
234         cout << error_msg;
235         if (abort)
236         {
237             cout << "or You can abort by typing q: ";
238             char a;
239             cin >> a;
240             if (a == 'q')
241             {
242                 done = true;

```

```

243         return num = L-1;
244     }
245 }
246 }
247 return num;
248 }
249
250 inline long get_l_w_LR(string prompt,
251                        string error_msg = "\nNumber is not in lower end of",
252                        long L = 0, bool abort = false)
253 {
254     cout << prompt;
255     cout.flush();
256     long num;
257     while ( cin.peek() == '\n' )
258     {
259         cin.ignore();
260         cout << error_msg;
261         cout.flush();
262     }
263     bool done = false;
264     while ( ! done)
265     {
266         cin >> num;
267         if ( num > L)
268         {
269             done = true;
270             return num;
271         }
272         cout << error_msg;
273         if (abort)
274         {
275             cout << "or You can abort by typing q: ";
276             char a;
277             cin >> a;
278             if (a == 'q')
279             {
280                 done = true;
281                 return num = L-1;
282             }
283         }
284     }
285     return num;
286 }
287
288 inline long get_l_w_UR(string prompt,
289                        string error_msg = "\nNumber is not in upper end of",
290                        long U = 100, bool abort = false)
291 {
292     cout << prompt;
293     cout.flush();
294     long num;
295     while ( cin.peek() == '\n' )
296     {

```

```

297     cin.ignore();
298     cout << error_msg;
299     cout.flush();
300 }
301 bool done = false;
302 while ( ! done)
303 {
304     cin >> num;
305     if (num < U)
306     {
307         done = true;
308         return num;
309     }
310     cout << error_msg;
311     if (abort)
312     {
313         cout << "or You can abort by typing q: ";
314         char a;
315         cin >> a;
316         if (a == 'q')
317         {
318             done = true;
319             return num = U+1;
320         }
321     }
322 }
323 return num;
324 }
325
326
327
328 #endif

```

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
78 // each of those combinations of positions for s to a given string
79 // and each time they match then the num becomes bigger by 1. Returns
80 // zero if no matches occurs or returns the number of times matches
81 // occurred.
82
83 inline short com1(vector<string::size_type> b, vector<string::size_type> e,
84                 string s, string t)
85 {
86     short num;
87     for(auto p : b)
88     {
89         for(auto p1 : e)
90         {
91             if ( p > p1)
92             {
93                 num;
94             }
95             else
96             {
97                 bool ys = incl(s, t, p, p1);
98                 if ( ys)
99                 {
100                     ++num;
101                 }
102                 num;
103             }
104         }
105     }
106     return num;
107 }
108
109 // Helper function tells if string t is in string s and returns the result
110 // through a bool.
111
112 inline bool find_str_in(string s, string t)
113 {
114     bool found;
115     vector<string::size_type> pos = num_in(s, t[0]);
116     vector<string::size_type> pos_b = num_in(s, t[ t.length() - 1]);
117     if ( ! pos.empty() && ! pos_b.empty())
118     {
119         short ys = com1( pos, pos_b, s, t);
120         if ( ys != 0)
121         {
122             return found = true;
123         }
124     }
125     else

```

```

125     {
126         return found = false;
127     }
128 }
129 else
130 {
131     return found = false;
132 }
133 }
134
135 // Helper function tells if string t is in string s and returns the result
136 // by giving the location on where it occurred or if it didn't occur
137 // then it returns the size of string s to indicate to the caller that
138 // string t is not in string s.
139
140 inline string::size_type find_str_loc(string s, string t)
141 {
142     string::size_type loc;
143     vector<string::size_type> pos = num_in(s, t[0]);
144     vector<string::size_type> pos_b = num_in(s, t[ t.length() - 1]);
145     if ( ! pos.empty() && ! pos_b.empty())
146     {
147         string::size_type ys = com(pos, pos_b, s, t);
148         if ( ys != s.length() )
149         {
150             return loc = ys;
151         }
152         else
153         {
154             return loc = s.length();
155         }
156     }
157     else
158     {
159         return loc = s.length();
160     }
161 }
162
163 inline short find_str(string s, string t)
164 {
165     short loc = 0;
166     vector<string::size_type> pos = num_in(s, t[0]);
167     vector<string::size_type> pos_b = num_in(s, t[ t.length() - 1]);
168     if ( ! pos.empty() && ! pos_b.empty())
169     {
170         string::size_type ys = com(pos, pos_b, s, t);
171         if ( ys != s.length() )
172         {
173             ++loc;
174         }
175     }
176     return loc;
177 }
178 }

```

```

179
180 // Helper function that combines each possible combination of two vectors
181 // and returns that vector. In the future, I could template it to
182 // fit any data type just not strings.
183
184 inline vector<string>combine(vector<string> b, vector<string> e)
185 {
186     vector <string> com;
187     for(auto p : b)
188     {
189         for(auto p1 : e)
190         {
191             com.push_back( p + p1 );
192         }
193     }
194     return com;
195 }
196
197 // Helper function that tells caller if a char is in a string returns
198 // results as a bool.
199
200 inline bool find_char_in( string s, char t)
201 {
202     bool in;
203     vector<string::size_type> t_in = num_in(s, t);
204     if ( ! t_in.empty())
205     {
206         return in = true;
207     }
208     else
209     {
210         return in = false;
211     }
212 }
213
214 // Helper function that tells the caller the location of where char t
215 // is in string s. Returns a vector of locations or a empty vector if
216 // the char was not found in t.
217
218 inline vector<string::size_type> find_char_loc( string s, char t)
219 {
220     vector<string::size_type> in;
221     vector<string::size_type> t_in = num_in(s, t);
222     if ( ! t_in.empty())
223     {
224         for (auto p : t_in)
225         {
226             in.push_back(p);
227         }
228         return t_in;
229     }
230     else
231     {
232         return in;

```

```

233     }
234 }
235
236 // Helper function that compares two string. Returns 1 if true
237 // and 0 if wrong. This function can also be templated for
238 // all data types not just strings and could also be a bool.
239
240 inline short compare(string s, string t)
241 {
242     short n=0;
243     if (s == t)
244     {
245         return ++n;
246     }
247     return n;
248 }
249
250
251 #endif

```

bj94684@ares:~\$ show-code find.cpp

find.cpp:

```

1  #include<iostream>
2  #include<string>
3  #include<vector>
4  #include<cctype>
5  #include<limits>
6  #include"input_prot.h"
7  #include"strextra.h"
8
9  using namespace std;
10
11 int main()
12 {
13
14     cout << "\n\t\tWelcome to the Find Program\n\n";
15     string L = get_str("Please input your sentence: ",
16                       "\nInvalid, try again: ");
17     string f = get_str("Please Input what you want to find: ",
18                       "\nInvalid, try again: ");
19
20
21     if (f.length() == 1) //this means that if the string is one character
22                         //which is the same as a char
23     {
24         char t = f[0];
25         vector<string::size_type> times = find_char_loc(L,t); //searches t
26         if (! times.empty()) // if char is found then loop
27         {
28             bool done = false;
29             bool more_than_one;

```

```

30     while ( ! done )
31     {
32
33         char choice = get_ch("\nDo you want to know if your char is:
34             "Invalid, try again: ");
35         choice = static_cast<char>( toupper( choice ) );
36         cin.ignore(numeric_limits<streamsize>::max(), '\n');
37         if ( choice == 'Y')
38         {
39             more_than_one = true; //user wants multiple locations
40             done = true;
41         }
42         else if ( choice == 'N')
43         {
44             more_than_one = false; // user doesnt want mul. locs.
45             done = true;
46         }
47         else
48         {
49             cout << " You did not input any correct answer."
50                 << " Please Try Again ";
51         }
52     }
53     if (! more_than_one) // times[0] is the first location of the
54     {                     // char since it is a vector of locations
55         cout << "\nYour character is in position " << times[0];
56     }
57     else
58     {
59         cout << "\nYour charcter is in positions ";
60         for (vector<string::size_type>::size_type p = 0;
61             p + 1 != times.size(); ++p)
62         {
63             cout << times[p] << ' ';    // goes thru all elements
64             // of the times vector
65             cout << times.back() << '\n'; // and display all pos.
66         }
67     }
68     else // what happens when vector times is empty
69     {
70         cout << "\nYour character is not in the sentence";
71     }
72 }
73 else // if the user wants to find a string within a string
74 {
75
76     bool in = find_str_in(L,f); // is the desired string f, in the
77     if(in)                     // given string L.
78     {
79         string::size_type loc = find_str_loc(L,f);
80         cout << "The string is in the word and it is in position "
81             << loc << '\n'; // list the location of the instance
82     }
83     else // if the desired string isnt in the given string

```

```

84     {
85         cout << "The string is not in the word";
86     }
87
88     }
89     return 0;
90 }
bj94684@ares:~$ CPP find
find.cpp***
In file included from find.cpp:7:
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:~$ ./find.out

Welcome to the Find Program

Please input your sentence: The quick brown fox
Please Input what you want to find: e

Do you want to know if your char is located in several locations? no
Your character is in position 2bj94684@ares:~$ ./find.out

Welcome to the Find Program

Please input your sentence: the quick brown fox
Please Input what you want to find: cow
The string is not in the wordbj94684@ares:~$ ./find.out

Welcome to the Find Program

```

```
Please input your sentence: 11112
Please Input what you want to find: 112
The string is in the word and it is in position 2
bj94684@ares:~$ ./find
bash: ./find: No such file or directory
bj94684@ares:~$ ./find.out
```

Welcome to the Find Program

```
Please input your sentence: 11111212122111212112121
Please Input what you want to find: 12122
The string is in the word and it is in position 6
bj94684@ares:~$ ./find.out
```

Welcome to the Find Program

```
Please input your sentence: i went to store
Please Input what you want to find: zam
The string is not in the wordbj94684@ares:~$ ./find.out
```

Welcome to the Find Program

```
Please input your sentence: I went to school for doing good
Please Input what you want to find: o
```

Do you want to know if your char is located in several locations? yes

Your charcter is in positions 8 13 14 18 22 28 29

```
bj94684@ares:~$ cat find.tpq
/*****
*
* TPQs:
* 1. For the find char function, it takes a string and char argument.
* While the find string function takes two string arguments. These are
* different so to help the caller distinguish between them.
* 2. The char search function returns a vector of string::size_type while
* the string search function returns a string::size type data type.
* The char search function could find multiple locations thats why it is
* a vector but the string search function purpose is to find only one
* location because that would tell the caller that the string is found.
* 3. For the char search function, if it is not found then it simply
* returns an empty vector and for the string search function it returns
* the length of the string that is being searched since it would be not
* possible for the string to be at that position.
* 4. The compiler would distinguish two same named functions by their
* arguments.
* 8. You protect the library from being circularly by including:
* #ifndef LIB_NAME_H_INC
* #define LIB_NAME_H_INC
* //
* #endif
* in the interface file.
* 9. For the main program, In order for it to work I just #include
* "lib_name.h" in the top of my program since all my functions are
*****/
```

```
* inlined in the interface file so there is no need for an implementation *
* file.For compiling, I will need to also include the lib_name.h file. *
* 10. My library consists solely of the interface files since all my *
* functions are inline so they are defined there so no need for an *
* implementation file. When using an implementation file, you need to *
* #include the library made in this one. *
*
*****/
bj94684@ares:~$ exit
exit
```

Script done on 2023-06-22 19:16:59-05:00 [COMMAND\_EXIT\_CODE="0"]