

```
Script started on 2023-12-07 17:55:44-06:00 [TERM="xterm" TTY="/dev/pts/15" COLUMNS=
a_vitale3@ares:~$ pwd
/home/students/a_vitale3
a_vitale3@ares:~$ cat NamesAndAddr.
Name: Andrew Vitale
```

Class: CSC121

Activity: Names And Addresses

Level: 6

Description:

Allows a user to input a first and last name, address, email, and phone number.  
this program also allows the user to edit, delete, search, and list all info.

```
a_vitale3@ares:~$ show-code NamesAndAddr.cpp
```

NamesAndAddr.cpp:

```
1 #include <iostream>
2 #include <string>
3 #include <vector>
4 #include <limits>
5
6 #include "rolodex.h"
7
8 using namespace std;
9
10 void interface();
11 void sub_interface(string text);
12 void listEntries(bool list);
13
14 vector<RolodexEntry> entries;
15
16 int main()
17 {
18     bool exit = false;
19     int input_int = 0;
20     long unsigned int index;
21     string input_str;
22     do
23     {
24         interface();
```

```
25     cin >> input_str;
26     if (input_str.find('1') != string::npos)
27     {
28         RolodexEntry entry;
29         entry.input();
30         entries.push_back(entry);
31     }
32     else if (input_str.find('2') != string::npos)
33     {
34         while(true)
35         {
36             sub_interface("edit");
37             cin >> input_str;
38             if (input_str.find('1') != string::npos)
39             {
40                 listEntries(true);
41                 cout << "Edit the name of entry number: ";
42                 std::cin.ignore(std::numeric_limits<std::streamsize>::r
43                 if (isdigit(std::cin.peek()))
44                 {
45                     cin >> input_int;
46                 }
47                 else
48                 {
49                     std::cout << "Error.\n";
50                     input_int = 0;
51                 }
52                 input_int -= 1;
53                 if (input_int >= 0 && static_cast<long unsigned int>(i
54                 {
55                     cout << "New first name of entry: ";
56                     cin >> input_str;
57                     entries[static_cast<long unsigned int>(input_int)]
58                     cout << "New last name of entry: ";
59                     cin >> input_str;
60                     entries[static_cast<long unsigned int>(input_int)]
61                 }
62             }
63             else if (input_str.find('2') != string::npos)
64             {
65                 listEntries(true);
66                 cout << "Edit the address of entry number: ";
67                 std::cin.ignore(std::numeric_limits<std::streamsize>::r
68                 if (isdigit(std::cin.peek()))
69                 {
70                     cin >> input_int;
71                 }
72                 else
73                 {
74                     std::cout << "Error.\n";
75                     input_int = 0;
76                 }
77                 input_int -= 1;
78                 if (input_int >= 0 && static_cast<long unsigned int>(i
```

```

79         {
80             cout << "New street of entry: ";
81             cin >> input_str;
82             entries[static_cast<long unsigned int>(input_int)]
83             cout << "New last town of entry: ";
84             cin >> input_str;
85             entries[static_cast<long unsigned int>(input_int)]
86             cout << "New last state of entry: ";
87             cin >> input_str;
88             entries[static_cast<long unsigned int>(input_int)]
89         }
90     }
91     else if (input_str.find('3') != string::npos)
92     {
93         listEntries(true);
94         cout << "Edit the phone number of entry number: ";
95         std::cin.ignore(std::numeric_limits<std::streamsize>::r
96         if (isdigit(std::cin.peek()))
97         {
98             cin >> input_int;
99         }
100         else
101         {
102             std::cout << "Error.\n";
103             input_int = 0;
104         }
105         input_int -= 1;
106         if (input_int >= 0 && static_cast<long unsigned int>(i
107         {
108             cout << "New phone number of entry: ";
109             cin >> input_str;
110             entries[static_cast<long unsigned int>(input_int)]
111         }
112     }
113     else if (input_str.find('4') != string::npos)
114     {
115         listEntries(true);
116         cout << "Edit the email of entry number: ";
117         std::cin.ignore(std::numeric_limits<std::streamsize>::r
118         if (isdigit(std::cin.peek()))
119         {
120             cin >> input_int;
121         }
122         else
123         {
124             std::cout << "Error.\n";
125             input_int = 0;
126         }
127         input_int -= 1;
128         if (input_int >= 0 && static_cast<long unsigned int>(i
129         {
130             cout << "New email of entry: ";
131             cin >> input_str;
132             entries[static_cast<long unsigned int>(input_int)]

```

```

133         }
134     }
135     else
136     {
137         break;
138     }
139 }
140 }
141 else if (input_str.find('3') != string::npos)
142 {
143     listEntries(true);
144     std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\
145     if (isdigit(std::cin.peek()))
146     {
147         cin >> input_int;
148     }
149     else
150     {
151         std::cout << "Error.\n";
152         input_int = 0;
153     }
154     input_int -= 1;
155     if(input_int >= 0 && static_cast<long unsigned int>(input_int)
156     {
157         entries.erase(entries.begin()+static_cast<long unsigned in
158     }
159 }
160 else if (input_str.find('4') != string::npos)
161 {
162     while (true)
163     {
164         sub_interface("search by");
165         cin >> input_str;
166         if (input_str.find('1') != string::npos)
167         {
168             cout << "Entry name: ";
169             cin >> input_str;
170             index = 0;
171             for (auto entry : entries)
172             {
173                 if (entry.get_firstName().find(input_str) != string
174                 {
175                     cout << "Entry is at postion " << index + 1 <<
176                 }
177                 else
178                 {
179                     index += 1;
180                 }
181             }
182             if (index >= entries.size())
183             {
184                 cout << "Entry does not exist." << endl;
185             }
186         }

```

```

187 else if (input_str.find('2') != string::npos)
188 {
189     cout << "Entry address: ";
190     cin >> input_str;
191     index = 0;
192     for (auto entry : entries)
193     {
194         if (entry.get_street().find(input_str) != string::npos)
195         {
196             cout << "Entry is at position " << index + 1 << endl;
197         }
198         else
199         {
200             index += 1;
201         }
202     }
203     if (index >= entries.size())
204     {
205         cout << "Entry does not exist." << endl;
206     }
207 }
208 else if (input_str.find('3') != string::npos)
209 {
210     cout << "Entry phone number: ";
211     cin >> input_str;
212     index = 0;
213     for (auto entry : entries)
214     {
215         if (entry.get_phone().find(input_str) != string::npos)
216         {
217             cout << "Entry is at position " << index + 1 << endl;
218         }
219         else
220         {
221             index += 1;
222         }
223     }
224     if (index >= entries.size())
225     {
226         cout << "Entry does not exist." << endl;
227     }
228 }
229 else if (input_str.find('4') != string::npos)
230 {
231     cout << "Entry email: ";
232     cin >> input_str;
233     index = 0;
234     for (auto entry : entries)
235     {
236         if (entry.get_email().find(input_str) != string::npos)
237         {
238             cout << "Entry is at position " << index + 1 << endl;
239         }
240         else

```

```

241     {
242         index += 1;
243     }
244 }
245 if (index >= entries.size())
246 {
247     cout << "Entry does not exist." << endl;
248 }
249 }
250 else
251 {
252     break;
253 }
254 }
255 }
256 else if (input_str.find('5') != string::npos)
257 {
258     if (entries.size() != 0)
259     {
260         listEntries(false);
261     }
262     else
263     {
264         cout << "You have not input any entries.\n";
265     }
266 }
267 else if (input_str.find('6') != string::npos)
268 {
269     exit = true;
270 }
271 } while (exit != true);
272 return 0;
273 }
274
275 void interface()
276 {
277     cout << "1) Add entry\n" <<
278           "2) Edit entry\n" <<
279           "3) Delete entry\n" <<
280           "4) Search for entry\n" <<
281           "5) Print all entries\n" <<
282           "6) Quit\n";
283 }
284
285 void sub_interface(string text)
286 {
287     cout << "1) " << text << " Name\n" <<
288           "2) " << text << " Address\n" <<
289           "3) " << text << " Phone number\n" <<
290           "4) " << text << " Email address\n" <<
291           "5) return to Main Menu\n";
292 }
293
294 void listEntries(bool list)

```

```

295 {
296     int incrementor = 0;
297     for (auto entry : entries)
298     {
299         if (list == true)
300         {
301             cout << incrementor+1 << ": ";
302             incrementor += 1;
303             cout << entry.get_firstName() << " " << entry.get_lastName() << " ";
304         }
305         else
306         {
307             entry.output();
308         }
309     }
310     cout << endl;
311 }

```

a\_vitale3@ares:~\$ show-code NamesAndAddolodex

rolodex.cpp:

```

1  #include "rolodex.h"
2
3  #include <iostream>
4  #include <string>
5  #include <limits>
6
7  // constructor's, accessors, mutators, i/o, equality, etc.
8
9  //constructor
10 RolodexEntry::RolodexEntry()
11 {
12     set_firstName("first name");
13     set_lastName("last name");
14     set_street("street");
15     set_town("town");
16     set_state("state");
17     set_zip(0);
18     set_zip_4(0);
19     set_phone("phone");
20     set_email("email");
21     return;
22 }
23 RolodexEntry::RolodexEntry(std::string text_one, std::string text_two)
24 {
25     set_firstName(text_one);
26     set_lastName(text_two);
27     set_street("street");
28     set_town("town");
29     set_state("state");
30     set_zip(0);
31     set_zip_4(0);

```

```

32     set_phone("phone");
33     set_email("email");
34     return;
35 }
36 /*RolodexEntry::RolodexEntry(std::string fname, std::string lname, std::string street, std::string town, std::string state, int zip, int zip_4, string phone, string email)
37 {
38     set_firstName(fname);
39     set_lastName(lname);
40     set_street(street);
41     set_town(town);
42     set_state(state);
43     set_zip(zip);
44     set_zip_4(zip_4);
45     set_phone(phone);
46     set_email(email);
47     return;
48 }*/
49
50 //i/o
51 void RolodexEntry::input(void)
52 {
53     std::cout << "\nFirst Name: ";
54     std::cin >> fname;
55     std::cout << "Last Name: ";
56     std::cin >> lname;
57     std::cout << "Street Name: ";
58     std::cin >> street;
59     std::cout << "Town Name: ";
60     std::cin >> town;
61     std::cout << "State Name: ";
62     std::cin >> state;
63     std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
64     std::cout << "Zip code: ";
65     if (isdigit(std::cin.peek()))
66     {
67         std::cin >> zip;
68     }
69     else
70     {
71         std::cout << "Error.\n";
72         zip = 0;
73     }
74     std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
75     std::cout << "Zip 4 code: ";
76     if (isdigit(std::cin.peek()))
77     {
78         std::cin >> zip_4;
79     }
80     else
81     {
82         std::cout << "Error.\n";
83         zip_4 = 0;
84     }
85     std::cout << "Phone Number: ";

```

```

86     std::cin >> phone;
87     std::cout << "Email: ";
88     std::cin >> email;
89 }
90 void RolodexEntry::output(void) const
91 {
92     std::cout << "\nFirst Name: "
93     << fname << ", "
94     << "Last Name: "
95     << lname << ", "
96     << "Street Name: "
97     << street << ", "
98     << "Town Name: "
99     << town << ", "
100    << "State Name: "
101    << state << ", "
102    << "Zip code: "
103    << zip << ", "
104    << "Zip 4 Code: "
105    << zip_4 << ", "
106    << "Phone Number: "
107    << phone << ", "
108    << "Email: "
109    << email << ".";
110 }
111 void RolodexEntry::edit(void)
112 {
113     std::string input;
114     std::cout << "Would you like to edit First Name? ";
115     std::cin >> input;
116     if (input == "y" || input == "yes")
117     {
118         std::cout << "\nFirst Name: ";
119         std::cin >> fname;
120     }
121     std::cout << "Would you like to edit Last Name? ";
122     std::cin >> input;
123     if (input == "y" || input == "yes")
124     {
125         std::cout << "Last Name: ";
126         std::cin >> lname;
127     }
128     std::cout << "Would you like to edit Street Name? ";
129     std::cin >> input;
130     if (input == "y" || input == "yes")
131     {
132         std::cout << "Street Name: ";
133         std::cin >> street;
134     }
135     std::cout << "Would you like to edit Town Name? ";
136     std::cin >> input;
137     if (input == "y" || input == "yes")
138     {
139         std::cout << "Town Name: ";

```

```

140         std::cin >> town;
141     }
142     std::cout << "Would you like to edit State Name?";
143     std::cin >> input;
144     if (input == "y" || input == "yes")
145     {
146         std::cout << "State Name: ";
147         std::cin >> state;
148     }
149     std::cout << "Would you like to edit Zip code? ";
150     std::cin >> input;
151     if (input == "y" || input == "yes")
152     {
153         std::cout << "Zip code: ";
154         std::cin >> zip;
155     }
156     std::cout << "Would you like to edit Zip 4 code? ";
157     std::cin >> input;
158     if (input == "y" || input == "yes")
159     {
160         std::cout << "Zip 4 code: ";
161         std::cin >> zip_4;
162     }
163     std::cout << "Would you like to edit Phone Number? ";
164     std::cin >> input;
165     if (input == "y" || input == "yes")
166     {
167         std::cout << "Phone Number: ";
168         std::cin >> phone;
169     }
170     std::cout << "Would you like to edit Email? ";
171     std::cin >> input;
172     if (input == "y" || input == "yes")
173     {
174         std::cout << "Email: ";
175         std::cin >> email;
176     }
177 }
178
179 //equality
180 bool RolodexEntry::operator==(const RolodexEntry rol2)
181 {
182     if (fname == rol2.fname &&
183         lname == rol2.lname &&
184         street == rol2.street &&
185         town == rol2.town &&
186         state == rol2.state &&
187         zip == rol2.zip &&
188         zip_4 == rol2.zip_4 &&
189         phone == rol2.phone &&
190         email == rol2.email)
191     {
192         return true;
193     }

```

```

194     else
195     {
196         return false;
197     }
198 }

```

a\_vitale3@ares:~\$ show-code rolodex.h

rolodex.h:

```

1  #ifndef ROLODEX_HELPER_LIBRARY_INCLUDED
2  #define ROLODEX_HELPER_LIBRARY_INCLUDED
3
4  #include <string>
5
6  class RolodexEntry
7  {
8      std::string fname = "first name", lname = "last name"; // together or
9      std::string street = "street", town = "town", state = "state";
10     long zip = 0;
11     short zip_4 = 0;
12     std::string phone = "phone", // or 3 short's? (area, exchange, line)
13     email = "email";
14 public:
15     // constructor's, accessors, mutators, i/o, equality, etc.
16
17     //constructor
18     RolodexEntry(void);
19     RolodexEntry(std::string text_one, std::string text_two);
20     //RolodexEntry(std::string fname, std::string lname, std::string street,
21
22     //i/o
23     void input(void);
24     void output(void) const;
25     void edit(void);
26
27     //accessors
28     std::string get_firstName() const { return fname; }
29     std::string get_lastName() const { return lname; }
30     std::string get_street() const { return street; }
31     std::string get_town() const { return town; }
32     std::string get_state() const { return state; }
33     long get_zip() const { return zip; }
34     short get_zip_4() { return zip_4; }
35     std::string get_phone() const { return phone; }
36     std::string get_email() const { return email; }
37
38     //mutators
39     void set_firstName(const std::string text) { fname = text; return; }
40     void set_lastName(const std::string text) { lname = text; return; }
41     void set_street(const std::string text) { street = text; return; }
42     void set_town(const std::string text) { town = text; return; }
43     void set_state(const std::string text) { state = text; return; }

```

```

44     void set_zip(const long num) { zip = num; return; }
45     void set_zip_4(const short num) { zip_4 = num; return; }
46     void set_phone(const std::string text) { phone = text; return; }
47     void set_email(const std::string text) { email = text; return; }
48
49     //equality
50     bool operator==(const RolodexEntry rol2);
51
52 };
53
54 #endif

```

a\_vitale3@ares:~\$ CPP NamesAndAddr rolodex rolodex.h

NamesAndAddr.cpp\*\*\*

rolodex.cpp...

a\_vitale3@ares:~\$ ./NamesAndAddr.out

```

1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
1

```

```

First Name: first
Last Name: last
Street Name: street
Town Name: town
State Name: state
Zip code: 1111
Zip 4 code: 1111
Phone Number: 111-111-1111
Email: email@gmail.com

```

```

1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
5

```

First Name: first, Last Name: last, Street Name: street, Town Name: town, State Nar

```

1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
2
1) edit Name
2) edit Address
3) edit Phone number
4) edit Email address

```

<p>5) return to Main Menu 1 1: first last</p> <p>Edit the name of entry number: 1 New first name of entry: newname New last name of entry: newlast 1) edit Name 2) edit Address 3) edit Phone number 4) edit Email address 5) return to Main Menu 2 1: newname newlast</p> <p>Edit the address of entry number: 1 New street of entry: newstreet New last town of entry: newtown New last state of entry: newstate 1) edit Name 2) edit Address 3) edit Phone number 4) edit Email address 5) return to Main Menu 3 1: newname newlast</p> <p>Edit the phone number of entry number: 1 New phone number of entry: 333-333-3333 1) edit Name 2) edit Address 3) edit Phone number 4) edit Email address 5) return to Main Menu 4 1: newname newlast</p> <p>Edit the email of entry number: 1 New email of entry: newenail@hotmail.com 1) edit Name 2) edit Address 3) edit Phone number 4) edit Email address 5) return to Main Menu 5 1) Add entry 2) Edit entry 3) Delete entry 4) Search for entry 5) Print all entries 6) Quit 5</p> <p>First Name: newname, Last Name: newlast, Street Name: newstreet, Town Name: newtown</p>	<p>1) Add entry 2) Edit entry 3) Delete entry 4) Search for entry 5) Print all entries 6) Quit 4 1) search by Name 2) search by Address 3) search by Phone number 4) search by Email address 5) return to Main Menu 1 Entry name: newname Entry is at postion 1 1) search by Name 2) search by Address 3) search by Phone number 4) search by Email address 5) return to Main Menu 2 Entry address: new Entry is at postion 1 1) search by Name 2) search by Address 3) search by Phone number 4) search by Email address 5) return to Main Menu 3 Entry phone number: 3 Entry is at postion 1 1) search by Name 2) search by Address 3) search by Phone number 4) search by Email address 5) return to Main Menu 4 Entry email: new Entry is at postion 1 1) search by Name 2) search by Address 3) search by Phone number 4) search by Email address 5) return to Main Menu 5 1) Add entry 2) Edit entry 3) Delete entry 4) Search for entry 5) Print all entries 6) Quit 3 1: newname newlast</p>
--	--

```
1
1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
5
You have not input any entries.
1) Add entry
2) Edit entry
3) Delete entry
4) Search for entry
5) Print all entries
6) Quit
6
a_vitale3@ares:~$ exit
exit
```

```
Script done on 2023-12-07 18:00:29-06:00 [COMMAND_EXIT_CODE="0"]
```