```
Script started on 2023-12-13 12:06:19-06:00 [TERM="xterm" TTY="/dev/pts/1" COLUMNS=
mf98604@ares:~$ pwd
/home/students/mf98604
mf98604@ares:~$ cat sorting.info
             Philip May'r
Name:
Class:
             CSC121-001
Activity:
             Sorting Algorithm Comparison Program
Options:
            Add (Level 0.5) to allow the user to specify the bounds of the range
             for the random values you are filling the vector with.
            Add (Level 0.5) to allow the user the option of filling the vector
             randomly or entering the data by hand.
            Add (Level 1.5) to use the time function to measure the time the
             sorting algorithms take to run.
            Add (Level 1.5) to do the bubble sort as well.
Level:
             8 (4 base + 4 options)
Description: Compares the run time of selection, insertion, and bubble
            sorting algorithms.
mf98604@ares:~$ CPP sorting random
random.cpp...
sorting.cpp***
mf98604@ares:~$ show-code sorting.cpp
sorting.cpp:
    1 #include "random.h"
      #include <ctvpe.h>
     3 #include <iostream>
     4 #include <limits>
     5 #include <vector>
       #include <ctime>
    8
        using namespace std;
    9
       typedef vector<long>::size type vec lng sz;
    11
        constexpr streamsize INF FLAG{numeric limits<streamsize>::max()};
    12
    13
    14 template <typename Base Type>
    15 typename vector<Base Type>::size type
    16 find smallest(vector<Base Type> list,
                      typename vector<Base Type>::size type start position = 0);
    17
   18
      template <typename Base Type>
    20 void selection sort(vector<Base Type> &list);
```

```
21
    template <typename Base Type>
    void selection sort(vector<Base Type> &list,
24
                        typename vector<Base Type>::size type start.
25
                        typename vector<Base Type>::size type end):
26
27
    template <typename Base Type>
   void insertion sort(vector<Base Type> &list);
29
30
    template <typename Base Type>
    void insertion sort(vector<Base Type> &list.
32
                        typename vector<Base Type>::size type start.
33
                        typename vector<Base Type>::size type end);
34
35
    template <typename Base Type>
36
    void bubble sort(vector<Base Type> &list);
37
    template <typename Base Type>
38
39
    void bubble sort(vector<Base Type> &list,
                     typename vector<Base Type>::size type start,
41
                     typename vector<Base Type>::size type end):
42
    template <typename Base Type>
    bool is sorted(vector<Base Type> &list.
45
                   typename vector<Base Type>::size type start,
46
                   typename vector<Base Type>::size type end);
47
    template <typename Base Type>
    void compare sorts(vector<Base Type> list, long runs);
51 void report times(time t start, time t end, long runs, string label);
52
53 int main()
54 {
        string welcome = "Welcome to the Sorting Algorithm Comparison Program."
55
        cout << "\n" << string((80 - welcome.length()) / 2, ' ') << welcome <<</pre>
56
57
58
        vec lng sz len;
59
        vector<long> list:
60
        long lower bound, upper bound;
61
62
        char y n;
63
64
        cout << "\nNew vector to be initialized with random values. \n"</pre>
65
                "Would you rather fill in vector values by hand?\n"
                "Enter v/n: ":
66
67
68
        cin >> y n;
        cin.ignore(INF FLAG, '\n');
69
70
71
        srand(static cast<unsigned>(time(nullptr))):
72
73
        if (tolower(y n) != 'n')
74
        {
```

```
75
             cout << "\nEnter vector values below -one per line- "</pre>
 76
                   << "and enter 'done' to stop:\n":
 77
             long value = 0;
 78
             for (;;)
 79
                  cin >> value;
 80
 81
                  cin.ignore(INF FLAG, '\n');
 82
                  if (!cin.fail())
 83
 84
                      list.push back(value);
 85
                  }
 86
                  else
 87
 88
                      cin.clear();
                      cin.ignore(INF FLAG, '\n');
 89
 90
                      break:
                  }
 91
 92
 93
             cout << "\nVector of size " << list.size()</pre>
 94
                   << " initialized and filled in:\n":
 95
         }
 96
         else
 97
 98
             cout << "\nEnter number of elements to initialize vector with:\n":</pre>
 99
             cin >> len;
100
             cin.ignore(INF FLAG, '\n');
             cout << "\nEnter lower and upper bounds (both inclusive) "</pre>
101
                      "for range of random value generator.\n"
102
                   << "lower bound: ";
103
104
             cin >> lower bound:
105
             cin.ignore(INF FLAG, '\n');
             cout << "upper bound: ";</pre>
106
107
             cin >> upper bound;
             cin.ignore(INF FLAG, '\n');
108
109
             for (vec lng sz i = 0; i < len; i++)
110
111
112
                  // rand min and max both inclusive
                  list.push back(get random num(lower bound, upper bound));
113
114
115
116
             cout << "\nVector of size " << len</pre>
117
                   << " initialized and filled in with random values:\n";</pre>
118
         }
119
120
         vector<long> og list = list;
121
122
         cout << "\n":
123
124
         for (vec lng sz i = 0; i < list.size(); i++)
125
126
             cout << list[i] << " ";
127
         }
128
```

```
129
         cout << "\n\nChoose sorting algorithm: \n\n"</pre>
130
                  "Enter:\n's' for selection sort. "
131
                  "\n'i' for insertion sort, "
132
                  "\n'b' for bubble sort:\n ":
133
134
         char sort choice;
135
         cin >> sort choice;
136
         cin.ignore(INF FLAG, '\n');
137
138
         for (;;)
139
         {
140
             if (!(tolower(sort choice) == 's' ||
141
                    tolower(sort choice) == 'i' ||
142
                    tolower(sort choice) == 'b'))
143
144
                  cout << "\nInvalid choice.";</pre>
145
146
                  cout << "\n\nChoose sorting algorithm: \n\n"</pre>
147
                  "Enter:\n's' for selection sort, "
148
                  "\n'i' for insertion sort, '
149
                  "\n'b' for bubble sort:\n ":
150
151
                  cin >> sort choice;
152
                  cin.ignore(INF FLAG, '\n');
153
                  sort choice = char(tolower(sort choice));
154
             }
155
             else
156
157
                  break;
158
159
         }
160
161
         vec lng sz start, end;
162
163
         cout << "\nAll values will be sorted by default. \n"</pre>
164
               << "Would you rather sort only those values within a subrange?\n"</pre>
165
               << "Enter y/n: ";
166
         cin >> y_n;
167
         cin.ignore(INF FLAG, '\n');
168
169
         if (tolower(v n) != 'n')
170
171
             cout << "\nEnter vector subrange start index: ";</pre>
172
             cin >> start;
173
             cin.ignore(INF FLAG, '\n');
174
             cout << "Enter vector subrange end index: ";</pre>
175
             cin >> end;
176
             cin.ignore(INF FLAG, '\n');
177
178
         else
179
180
             start = 0;
181
             end = list.size() - 1;
182
         }
```

```
183
184
         switch(sort choice)
185
186
             case 's':
187
                 selection sort(list, start, end);
188
                  if (is sorted(list, start, end))
189
190
                      cout << "\nVector sorted using selection sort.\n";</pre>
191
192
             break:
193
             case 'i':
194
                  insertion sort(list. start. end):
195
                  if (is sorted(list, start, end))
196
197
                      cout << "\nVector sorted using insertion sort.\n";</pre>
198
199
             break:
200
             case 'b':
201
                  bubble sort(list, start, end);
202
                  if (is sorted(list, start, end))
203
204
                      cout << "\nVector sorted using bubble sort.\n";</pre>
205
206
             break:
207
             default:
208
                  cout << "\n\nInvalid choice.\n";</pre>
209
                  return 0;
210
         }
211
212
         cout << "\nWould vou like to print the sorted vector?\n"</pre>
213
                  "Enter v/n: ":
214
         cin >> y n;
215
         cin.ignore(INF FLAG, '\n');
216
         if (tolower(y n) != 'n')
217
218
219
             for (vec lng sz i = 0; i < list.size(); i++)</pre>
220
221
                  cout << list[i] << " ";
222
223
         }
224
225
226
         cout << "\n\nEnter number of times to run all three sorting algorithms'</pre>
227
               << " on original vector:\n";
228
         cin >> runs;
229
         compare sorts(og list, runs);
230
231
         cout << "\n";
232
233
             return 0:
234 }
235
236 template <typename Base Type>
```

```
237 void compare sorts(vector<Base Type> list, long runs)
238 {
239
         time t start, end;
240
241
         start = time(nullptr):
242
         vector<Base Type> unsorted list = list;
243
         for (long i = 0; i < runs; i++)
244
245
             list = unsorted list;
246
             selection sort(list);
247
248
         end = time(nullptr):
249
         report times(start, end, runs, "Selection");
250
251
         start = time(nullptr):
252
         for (long i = 0; i < runs; i++)
253
254
             list = unsorted list;
255
             insertion sort(list);
256
257
         end = time(nullptr):
258
         report times(start, end, runs, "Insertion");
259
260
         start = time(nullptr):
261
         for (long i = 0; i < runs; i++)
262
             list = unsorted list;
263
264
             bubble sort(list);
265
266
         end = time(nullptr):
267
         report times(start, end, runs, "Bubble");
268
269
270 void report times(time t start, time t end, long runs, string label)
271 {
272
         double ms = static cast<double>((end-start) * 1000);
273
         cout << "\n" << label << " sort took "
274
              << (ms/static cast<double>(runs))
275
              << " ms on average." << endl:
276
         cout << "Total time (" << runs << " runs): "</pre>
277
              << ms << " ms." << endl:
278 }
279
     template <typename Base Type>
     typename vector<Base Type>::size type
     find smallest(vector<Base Type> list,
283
                   typename vector<Base Type>::size type start position)
284
285
         typename vector<Base Type>::size type min = start position;
286
         for (vec lng sz position = start position + 1:
287
              position < list.size(); position++)</pre>
288
289
             if (list[position] < list[min])</pre>
290
```

```
291
                 min = position;
292
293
         }
294
         return min;
295
296
297
    template <typename Base Type>
298 void selection sort(vector<Base Type> &list)
299 {
         selection sort(list, 0, list.size());
300
301 }
302
303 template <typename Base Type>
304
    void selection sort(vector<Base Type> &list,
305
                         typename vector<Base Type>::size type start,
                         typename vector<Base Type>::size type end)
306
307 {
308
         typename vector<Base Type>::size type min = find smallest(list, start)
309
310
         for(typename vector<Base Type>::size type position = start;
311
             position < end: position++)
312
313
             if (min != position)
314
315
                 swap(list[position], list[min]);
316
317
             min = find smallest(list, position + 1);
318
         }
319 }
320
321 template <typename Base Type>
322
    void insertion sort(vector<Base Type> &list)
323
324
        insertion sort(list, 0, list.size());
325 }
326
     template <typename Base Type>
    void insertion sort(vector<Base Type> &list,
329
                         typename vector<Base Type>::size type start,
330
                         typename vector<Base Type>::size type end)
331 {
332
         typename vector<Base Type>::size type j;
333
334
         for (typename vector<Base Type>::size type i = start + 1;
335
              i < end + 1; i++)
336
337
             key = list[i];
338
             j = i;
339
             while (j > start \&\& list[j - 1] > key)
340
341
                 list[j] = list[j - 1];
342
                 j--;
343
344
             list[j] = key;
```

```
345
            }
   346
       }
   347
       template <typename Base Type>
        void bubble sort(vector<Base Type> &list)
   350
   351
            bubble sort(list, 0, list.size());
   352 }
   353
   354
        template <typename Base Type>
        void bubble sort(vector<Base Type> &list,
   356
                          typename vector<Base Type>::size type start,
   357
                          typename vector<Base Type>::size type end)
   358
   359
            for (typename vector<Base Type>::size type i = start; i < end; i++)</pre>
   360
   361
                bool swap flag = false;
                for (typename vector<Base Type>::size type j = start; j < end; j++)</pre>
   362
   363
   364
                    if (list[j] > list[j + 1])
   365
   366
                        swap(list[j], list[j + 1]);
   367
                        swap flag = true;
   368
   369
   370
                if (!swap flag)
   371
   372
                    break;
   373
   374
            }
   375
   376
   377
        template <typename Base Type>
        bool is sorted(vector<Base Type> &list,
   379
                       typename vector<Base Type>::size type start,
   380
                       typename vector<Base Type>::size type end)
   381
   382
            bool flag = false;
   383
            for (typename vector<Base Type>::size type j = start; j < end; j++)</pre>
   384
   385
                if (list[j] > list[j + 1])
   386
   387
                    flag = true;
   388
   389
   390
            return flag;
   391 }
mf98604@ares:~$ show-code random.cpp
random.cpp:
     1 #include "random.h"
```

```
2 #include <cstdlib>
     3
       long get random num(long lower bound, long upper bound)
     5
            return static cast<long>(rand() %
     6
                                     (upper bound - lower bound + 1) +
     8
                                     lower bound);
     9 }
mf98604@ares:~$ show-code random.h
random.h:
     1 #ifndef RANDOM H INC
     2 #define RANDOM H INC
     4 long get random num(long lower bound, long upper bound);
     6 #endif
mf98604@ares:~$ ./sorting.out
              Welcome to the Sorting Algorithm Comparison Program.
New vector to be initialized with random values.
Would you rather fill in vector values by hand?
Enter y/n: no
Enter number of elements to initialize vector with:
Enter lower and upper bounds (both inclusive) for range of random value generator.
lower bound: 0
upper bound: 9
                                                                                      6
                                                                                      5
Vector of size 10 initialized and filled in with random values:
6 6 2 7 9 5 5 2 0 1
Choose sorting algorithm:
                                                                                      - 1
Enter:
                                                                                      -2
's' for selection sort,
                                                                                      28
'i' for insertion sort.
'b' for bubble sort:
                                                                                      done
Invalid choice.
Choose sorting algorithm:
Enter:
's' for selection sort,
                                                                                      Enter:
'i' for insertion sort,
```

```
'b' for bubble sort:
All values will be sorted by default.
Would you rather sort only those values within a subrange?
Enter y/n: n
Would you like to print the sorted vector?
Enter y/n: y
0 1 2 2 5 5 6 6 7 9
Enter number of times to run all three sorting algorithms on original vector:
10000000
Selection sort took 0.0026 ms on average.
Total time (10000000 runs): 26000 ms.
Insertion sort took 0.0005 ms on average.
Total time (10000000 runs): 5000 ms.
Bubble sort took 0.0011 ms on average.
Total time (10000000 runs): 11000 ms.
mf98604@ares:~$ ./sorting.out
              Welcome to the Sorting Algorithm Comparison Program.
New vector to be initialized with random values.
Would you rather fill in vector values by hand?
Enter y/n: y
Enter vector values below -one per line- and enter 'done' to stop:
Vector of size 12 initialized and filled in:
7 6 5 4 3 2 1 0 -1 -2 28 8
Choose sorting algorithm:
's' for selection sort,
```

```
'i' for insertion sort,
                                                                                      's' for selection sort,
                                                                                      'i' for insertion sort,
'b' for bubble sort:
                                                                                      'b' for bubble sort:
All values will be sorted by default.
Would you rather sort only those values within a subrange?
                                                                                      All values will be sorted by default.
Enter y/n: n
                                                                                      Would you rather sort only those values within a subrange?
                                                                                      Enter y/n: y
Would you like to print the sorted vector?
                                                                                      Enter vector subrange start index: 3
Enter y/n: y
-2 -1 0 1 2 3 4 5 6 7 8 28
                                                                                      Enter vector subrange end index: 11
Enter number of times to run all three sorting algorithms on original vector:
                                                                                      Would you like to print the sorted vector?
24000000
                                                                                      Enter v/n: v
                                                                                      21347 235788 2030 0 1 2 3 4 5 6 7 8
Selection sort took 0.00325 ms on average.
Total time (24000000 runs): 78000 ms.
                                                                                      Enter number of times to run all three sorting algorithms on original vector:
                                                                                      144000
Insertion sort took 0.000625 ms on average.
Total time (24000000 runs): 15000 ms.
                                                                                      Selection sort took 0 ms on average.
                                                                                      Total time (144000 runs): 0 ms.
Bubble sort took 0.00158333 ms on average.
Total time (24000000 runs): 38000 ms.
                                                                                      Insertion sort took 0 ms on average.
                                                                                      Total time (144000 runs): 0 ms.
mf98604@ares:~$ ./sorting.out
                                                                                      Bubble sort took 0.00694444 ms on average.
              Welcome to the Sorting Algorithm Comparison Program.
                                                                                      Total time (144000 runs): 1000 ms.
New vector to be initialized with random values.
                                                                                      mf98604@ares:~$ ./sorting.out
Would you rather fill in vector values by hand?
Enter v/n: v
                                                                                                    Welcome to the Sorting Algorithm Comparison Program.
Enter vector values below -one per line- and enter 'done' to stop:
                                                                                      New vector to be initialized with random values.
                                                                                      Would you rather fill in vector values by hand?
21347
235788
                                                                                      Enter y/n: n
2030
                                                                                      Enter number of elements to initialize vector with:
8
7
                                                                                      Enter lower and upper bounds (both inclusive) for range of random value generator.
                                                                                      lower bound: -12
4
3
                                                                                      upper bound: 12
2
1
                                                                                      Vector of size 20 initialized and filled in with random values:
0
                                                                                      -12 3 11 12 8 0 -5 -8 12 5 -7 -8 7 -9 7 11 -1 -9 -12 -7
Vector of size 12 initialized and filled in:
                                                                                      Choose sorting algorithm:
21347 235788 2030 8 7 6 5 4 3 2 1 0
                                                                                      Enter:
                                                                                      's' for selection sort.
                                                                                      'i' for insertion sort,
Choose sorting algorithm:
                                                                                      'b' for bubble sort:
Enter:
```

```
All values will be sorted by default.
Would you rather sort only those values within a subrange?
Enter v/n: v
Enter vector subrange start index: 1
Enter vector subrange end index: 11
Would you like to print the sorted vector?
Enter y/n: y
-12 -12 -9 -9 -8 -8 -7 -7 -5 -1 0 12 7 11 7 11 5 12 3 8
Enter number of times to run all three sorting algorithms on original vector:
500000
Selection sort took 0.006 ms on average.
Total time (500000 runs): 3000 ms.
Insertion sort took 0.002 ms on average.
Total time (500000 runs): 1000 ms.
Bubble sort took 0.004 ms on average.
Total time (500000 runs): 2000 ms.
mf98604@ares:~$ ./sorting.out
              Welcome to the Sorting Algorithm Comparison Program.
New vector to be initialized with random values.
Would you rather fill in vector values by hand?
Enter v/n: v
Enter vector values below -one per line- and enter 'done' to stop:
25
75
15
05
77
14
88
97
63
144
12
144000
28
2030
7000
20000000
DONE
Vector of size 16 initialized and filled in:
25 75 15 5 77 14 88 97 63 144 12 144000 28 2030 7000 20000000
```

```
Enter:
's' for selection sort.
'i' for insertion sort.
'b' for bubble sort:
All values will be sorted by default.
Would you rather sort only those values within a subrange?
Enter v/n: v
Enter vector subrange start index: 1
Enter vector subrange end index: 10
Would you like to print the sorted vector?
Enter y/n: y
25 5 12 14 15 63 75 77 88 97 144 144000 28 2030 7000 20000000
Enter number of times to run all three sorting algorithms on original vector:
5000000
Selection sort took 0.0046 ms on average.
Total time (5000000 runs): 23000 ms.
Insertion sort took 0.0006 ms on average.
Total time (5000000 runs): 3000 ms.
Bubble sort took 0.0016 ms on average.
Total time (5000000 runs): 8000 ms.
mf98604@ares:~$ script-print Mayr-Philip-sorting
"typescript" printed to "Mayr-Philip-sorting.pdf".
mf98604@ares:~$ exit
exit
Script done on 2023-12-13 12:22:01-06:00 [COMMAND EXIT CODE="0"]
```

Choose sorting algorithm: