```
 1: #include "main.h"
 2:
 3: using namespace std;
 4: using namespace std::chrono;
 5:
 6: int main(int argc, char **argv) {
 7: #ifdef TIME_MEASURING
 8:   cout << "Time measuring ON" << endl;
 9:   high_resolution_clock::time_point full_start, full_end, alg_start, alg_end;
10:   full_start = high_resolution_clock::now();
11: #endif
12:   vector<SegmentData> segments;
13:   CmdDriver parser(argc, argv);
14:
15:   std::atexit(exit_handler);
16:
17:   SqliteDriver sql(parser.getParam("sql"));
18:   FileDriver file(parser.getParam("bounds"));
19:
20:   try {
21:     segments = sql.load(segments);
22:     file.load();
23:   }
24:   catch (string& msg){
25:     cerr << "Load file problem!\n" << msg << endl;
26:     return(EXIT_FAILURE);
27:   }
28:
29:   EquationBounds::printEquationBounds();
30:
31: #ifdef TIME_MEASURING
32:   alg_start = high_resolution_clock::now();
33: #endif
34: #ifdef _INTEL_TBB_H
35:   cout << "Run with Intel TTB" << endl;
36:   int size = segments.size();
37:   tbb::task_scheduler_init init(4);
38:   tbb::parallel_for(0, size, [&](int i){
39:     NelderMead alg(segments[i]);
40:     cout << "Segment " << std::setfill(' ') << std::setw(3) <<
    segments[i].getId() << " " << segments[i].getResult() << endl;
41:   });
42: #else
43:   cout << "Run without Intel TTB" << endl;
44:   for (int size = segments.size(), i = 0; i < size; i++) {
45:     NelderMead alg(segments[i]);
46:     cout << "Segment " << std::setfill(' ') << std::setw(3) <<
    segments[i].getId() << " " << segments[i].getResult() << endl;
47:   }
48: #endif
49: #ifdef TIME_MEASURING
50:   alg_end = high_resolution_clock::now();
51: #endif
52:
53:   try {
54:     sql.save(segments);
55:   }
56:   catch (string& msg){
57:     cerr << "Save SQLite problem!\n" << msg << endl;
58:   }
59:
60: #ifdef TIME_MEASURING
61:   full_end = high_resolution_clock::now();
62:   auto full_duration = duration_cast<microseconds>(full_end - full_start).count();
```

```
63:   auto alg_duration = duration_cast<microseconds>(alg_end - alg_start).count();
64:
65:   cout << "Time execute for Nelder-Mead algorithm: " << alg_duration << "us => "
    << alg_duration / 1000.0 << "ms => " << alg_duration / 1000000.0 << "s" <<  endl;
66:   cout << "Time execute for the entire program: " << full_duration << "us => " <<
    alg_duration / 1000.0 << "ms => " << alg_duration / 1000000.0 << "s" << endl;
67: #endif
68:
69:   return(EXIT_SUCCESS);
70: }
71:
72: void exit_handler(){
73:   if (DEBUG){
74:     std::cout << "Press Enter to exit..." << std::endl;
75:     std::cin.get();
76:   }
77: }
```