Lab 2 Search

1.0

Generated by Doxygen 1.9.5

Chapter 1

Bug List

File generator.cpp

Currently, there are no any known bugs.

File generator.hpp

Currently, there are no any known bugs.

File model.cpp

Currently, there are no any known bugs.

File model.hpp

Currently, there are no any known bugs.

File search.cpp

Currently, there are no any known bugs.

File sorting.cpp

Currently, there are no any known bugs.

File sorting.hpp

Currently, there are no any known bugs.

Currently, there are no any known bugs.

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 Generator Class Reference

Public Member Functions

- Model **model_generator** ()

The documentation for this class was generated from the following files:

- src/generator/generator.hpp
- src/generator/generator.cpp

4.2 Model Class Reference

Public Member Functions

- **Model** (std::string full_name, std::string department, std::string job_title, std::chrono::year_month_day employment_date)
- **Model** (std::string full_name, std::string department, std::string job_title, std::string employment_date)
- **Model** (std::uint8_t decor_type)
- void **set_model** (std::string full_name, std::string department, std::string job_title, std::chrono::year_month_day employment_date)
- void **set_model** (std::string full_name, std::string department, std::string job_title, std::string employment_date)
- void **set_decor** (std::uint8_t decor_type)
- ModelComp **compare_type** (const Model &r_model, uint8_t mode)
- template<typename T >
  ModelComp **compare_type** (uint8_t mode, T r_value)
- template<typename T >
  T **get_field** (uint8_t field)

Static Public Member Functions

- static void **save_model** (const std::vector< Model > &model_vector, std::filesystem::path file_path)
- static void **load_model** (std::vector< Model > &model_vector, std::filesystem::path file_path)
- static void **print_model** (const std::vector< Model > &model_vector)

Friends

- std::ostream & **operator**<< (std::ostream &stream, const Model &model)
- ModelComp **operator**< (const Model &l_model, const Model &r_model)
- ModelComp **operator**> (const Model &l_model, const Model &r_model)
- ModelComp **operator**<= (const Model &l_model, const Model &r_model)
- ModelComp **operator**>= (const Model &l_model, const Model &r_model)
- ModelComp **operator==** (const Model &l_model, const Model &r_model)
- ModelComp **operator!=** (const Model &l_model, const Model &r_model)

The documentation for this class was generated from the following files:

- src/model/model.hpp
- src/model/model.cpp

4.3 ModelComp Class Reference

Public Member Functions

- **ModelComp** (uint8_t value)
- void **set_type_masked** (uint8_t value, uint8_t offset)
- void **set_name_type** (uint8_t value)
- void **set_dept_type** (uint8_t value)
- void **set_jobt_type** (uint8_t value)
- void **set_date_type** (uint8_t value)
- uint8_t **get_type_masked** (uint8_t offset) const
- uint8_t **get_name_type** () const
- uint8_t **get_dept_type** () const
- uint8_t **get_jobt_type** () const
- uint8_t **get_date_type** () const
- ModelComp **operator!** () const

Friends

- ModelComp **operator==** (const ModelComp &l_bool, const ModelComp &r_bool)
- ModelComp **operator!=** (const ModelComp &l_bool, const ModelComp &r_bool)
- std::ostream & **operator**<< (std::ostream &stream, const ModelComp &model)

The documentation for this class was generated from the following files:

- src/model/model.hpp
- src/model/model.cpp

4.4 Search Class Reference

A class that provides static searching methods for sorting a vector of Model objects based on a specific field.

`#include <search.hpp>`

Static Public Member Functions

- template<typename T >
  static int **binary_search** (std::vector< Model > &model_vector, T search_value, std::uint8_t field)
- template<typename T >
  static int **straight_search** (std::vector< Model > &model_vector, T search_value, std::uint8_t field)

4.4.1 Detailed Description

A class that provides static searching methods for sorting a vector of Model objects based on a specific field.

Note

Currently provides implementations for binary search.

The documentation for this class was generated from the following file:

- src/search/search.hpp

4.5 Sorting Class Reference

A class that provides static sorting methods for sorting a vector of Model objects based on a specific field.

`#include <sorting.hpp>`

Static Public Member Functions

- static void bubble_sort (std::vector< Model > &model_vector, uint8_t field)
  Sorts the given vector of Model objects using bubble sort algorithm.
- static void heap_sort (std::vector< Model > &model_vector, uint8_t field)
  Performs heap sort on a vector of Model objects.
- static void merge_sort (std::vector< Model > &model_vector, uint8_t field, std::size_t left=0, std::size_t right=0, bool initial=true)
  Sorts a vector of Model objects using merge sort algorithm.

4.5.1 Detailed Description

A class that provides static sorting methods for sorting a vector of Model objects based on a specific field.

Note

Currently provides implementations for bubble sort, heap sort, and merge sort.

4.5.2 Member Function Documentation

4.5.2.1 bubble_sort()

```
void Sorting::bubble_sort (
            std::vector< Model > & model_vector,
            uint8_t field ) [static]
```

Sorts the given vector of Model objects using bubble sort algorithm.

This function uses bubble sort algorithm to sort the given vector of Model objects based on the field specified by the `field` parameter. The objects are compared using the `compare_type()` method of the Model class.

Parameters

model_vector	The vector of Model objects to be sorted.
field	The index of the field to be used for sorting the objects.

Returns

void.

Note

This function modifies the original vector passed to it.

The `compare_type()` method of the Model class must return a Type object.

The Type object must have a method `get_type_masked()` that takes an offset and returns a boolean indicating whether the specified bit is set or not.

This function prints the number of iterations taken to sort the vector.

4.5.2.2 heap_sort()

```
void Sorting::heap_sort (
            std::vector< Model > & model_vector,
            uint8_t field ) [static]
```

Performs heap sort on a vector of Model objects.

This function sorts a vector of Model objects using the heap sort algorithm. The function uses the make_heap function to create a heap from the input vector, then sorts the heap by repeatedly extracting the maximum element from the heap and placing it at the end of the vector.

Parameters

model_vector	The vector of Model objects to be sorted.
field	The field of the Model object to sort by.

Returns

void.

4.5.2.3 merge_sort()

```
void Sorting::merge_sort (
            std::vector< Model > & model_vector,
            uint8_t field,
            std::size_t left = 0,
```

```
std::size_t right = 0,
bool initial = true )  [static]
```

Sorts a vector of Model objects using merge sort algorithm.

This function sorts a given vector of Model objects using merge sort algorithm. It takes the field to be sorted as input, along with left and right indices of the sub-vector to be sorted. If left and right indices are not provided, it sorts the entire vector by setting left and right indices accordingly.

Parameters

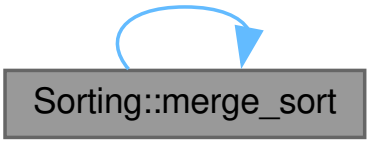
model_vector	The vector of Model objects to be sorted.
field	The field to be sorted.
left	The left index of the sub-vector to be sorted (default is 0).
right	The right index of the sub-vector to be sorted (default is size-1).
initial	A boolean flag indicating whether this is the initial call to the function (default is true).

Returns

void.

Here is the call graph for this function:

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- src/sorting/sorting.hpp
- src/sorting/sorting.cpp

Chapter 5

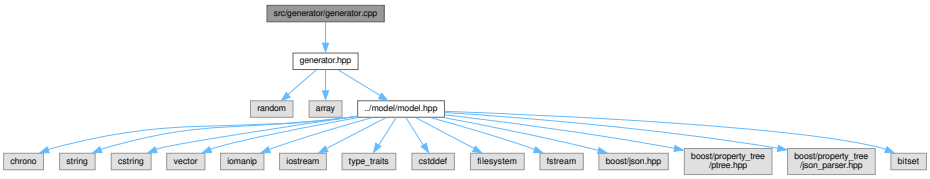
File Documentation

5.1 src/generator/generator.cpp File Reference

This source file holds implementation of Generator class.

```
#include "generator.hpp"
```

Include dependency graph for generator.cpp:



5.1.1 Detailed Description

This source file holds implementation of Generator class.

>

```
This calss implements model generator needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

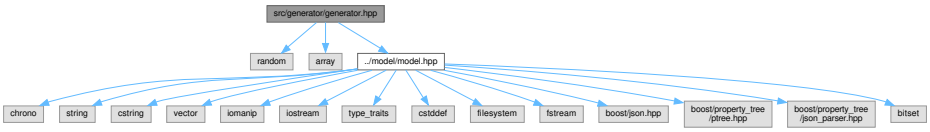
Copyright

5.2 src/generator/generator.hpp File Reference

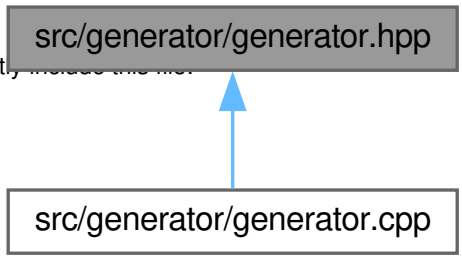
This header file holds implementation of Generator class.

```
#include <random>
#include <array>
#include "../model/model.hpp"
```

Include dependency graph for generator.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class Generator

5.2.1 Detailed Description

This header file holds implementation of Generator class.

>

```
This calss implements model generator needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright


```
(Not really)
```

5.3 generator.hpp

Go to the documentation of this file.

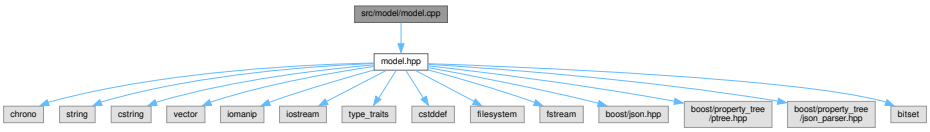
```
1
20 #ifndef GENERATOR_HPP
21 #define GENERATOR_HPP
22
23 #include <random>
24
25 #include <array>
26
27
28 #ifndef MODEL_HPP
29 #include "../model/model.hpp"
30 #endif // MODEL_HPP
31
32 class Generator
33 {
34 public:
35     Generator();
36     ~Generator();
```

```
37
38     Model model_generator();
39
40 private:
41     std::array<std::string, 2738> _first_name_list;
42     std::array<std::string, 1000> _last_name_list;
43     std::array<std::string, 449> _department_list;
44     std::array<std::string, 357> _job_title_list;
45
46     std::random_device       _random_device;
47     std::mt19937             _generator;
48
49     std::uniform_int_distribution<uint32_t> _first_name_distribution;
50     std::uniform_int_distribution<uint32_t> _last_name_distribution;
51     std::uniform_int_distribution<uint32_t> _department_distribution;
52     std::uniform_int_distribution<uint32_t> _job_title_distribution;
53     std::uniform_int_distribution<uint16_t> _year_distribution;
54     std::uniform_int_distribution<uint16_t> _month_distribution;
55     std::uniform_int_distribution<uint16_t> _day_distribution;
56     std::uniform_int_distribution<uint16_t> _sex_distribution;
57 };
58
59 #endif // GENERATOR_HPP
```

5.4 src/model/model.cpp File Reference

This source file holds implementation of Model class.

`#include "model.hpp"`
Include dependency graph for model.cpp:



Functions

- std::ostream & **operator**<< (std::ostream &stream, const Model &model)
- ModelComp **operator**< (const Model &l_model, const Model &r_model)
- ModelComp **operator**>**=** (const Model &l_model, const Model &r_model)
- ModelComp **operator**<**=** (const Model &l_model, const Model &r_model)
- ModelComp **operator**> (const Model &l_model, const Model &r_model)
- ModelComp **operator==** (const Model &l_model, const Model &r_model)
- ModelComp **operator!=** (const Model &l_model, const Model &r_model)
- ModelComp **operator==** (const ModelComp &l_bool, const ModelComp &r_bool)
- ModelComp **operator!=** (const ModelComp &l_bool, const ModelComp &r_bool)
- std::ostream & **operator**<< (std::ostream &stream, const ModelComp &r_bool)

5.4.1 Detailed Description

This source file holds implementation of Model class.

>

```
This calss implements model needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

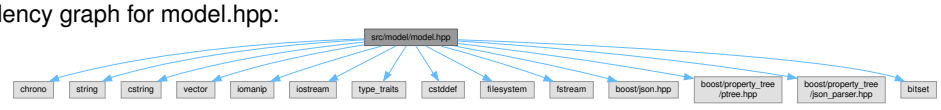
Copyright

5.5 src/model/model.hpp File Reference

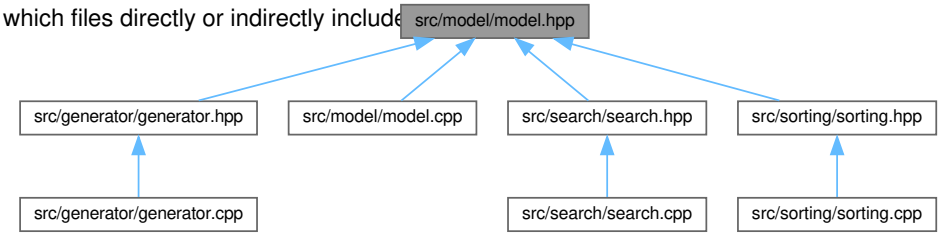
This header file holds implementation of Model class.

```
#include <chrono>
#include <string>
#include <cstring>
#include <vector>
#include <iomanip>
#include <iostream>
#include <type_traits>
#include <cstddef>
#include <filesystem>
#include <fstream>
#include <boost/json.hpp>
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#include <bitset>
```

Include dependency graph for model.hpp:



This graph shows which files directly or indirectly include



Classes

- class Model
- class ModelComp

5.5.1 Detailed Description

This header file holds implementation of Model class.

>

```
This calss implements model needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright


```
(Not really)
```

5.6 model.hpp

Go to the documentation of this file.

```
1
20 #ifndef MODEL_HPP
21 #define MODEL_HPP
22
23 #include <chrono>
24 #include <string>
25 #include <cstring> // strcmp has better performance
26 #include <vector>
27 #include <iomanip>
28 #include <iostream>
29 #include <type_traits>
30 #include <cstddef>
31
32
33 #include <filesystem>
34 #include <fstream>
35 #include <boost/json.hpp>
36 #include <boost/property_tree/ptree.hpp>
37 #include <boost/property_tree/json_parser.hpp>
38
39 #include <bitset>
40
41 class ModelComp;
42
43 class Model
44 {
45 public:
46     Model(std::string full_name, std::string department, std::string job_title,
      std::chrono::year_month_day employment_date);
47     Model(std::string full_name, std::string department, std::string job_title, std::string
      employment_date);
48     Model(std::uint8_t decor_type);
49     ~Model();
50
51     void set_model(std::string full_name, std::string department, std::string job_title,
      std::chrono::year_month_day employment_date);
52     void set_model(std::string full_name, std::string department, std::string job_title, std::string
      employment_date);
53     void set_decor(std::uint8_t decor_type);
54
55     ModelComp compare_type(const Model& r_model, uint8_t mode);
56
57     template<typename T>
58     ModelComp compare_type(uint8_t mode, T r_value);
59
60     template<typename T>
61     T get_field(uint8_t field);
62
63     static void save_model(const std::vector<Model>& model_vector, std::filesystem::path file_path);
64     static void load_model(std::vector<Model>& model_vector, std::filesystem::path file_path);
65     static void print_model(const std::vector<Model>& model_vector);
66
67     friend std::ostream& operator« (std::ostream& stream, const Model& model);
68
69     friend ModelComp     operator<  (const Model& l_model, const Model& r_model);
70     friend ModelComp     operator>  (const Model& l_model, const Model& r_model);
71     friend ModelComp     operator<= (const Model& l_model, const Model& r_model);
72     friend ModelComp     operator>= (const Model& l_model, const Model& r_model);
73
74     friend ModelComp     operator== (const Model& l_model, const Model& r_model);
75     friend ModelComp     operator!= (const Model& l_model, const Model& r_model);
76
77 private:
78     std::string                    _full_name;
79     std::string                    _department;
80     std::string                    _job_title;
81     std::chrono::year_month_day    _employment_date;
82
83     std::uint8_t                   _decor_type;
84 };
85
86 class ModelComp
87 {
88 public:
89     ModelComp();
90     ModelComp(uint8_t value);
91     ~ModelComp();
92
93     void set_type_masked(uint8_t value, uint8_t offset);
94
95     void set_name_type(uint8_t value);
96     void set_dept_type(uint8_t value);
```

```
97      void set_jobt_type(uint8_t value);
98      void set_date_type(uint8_t value);
99
100     uint8_t get_type_masked(uint8_t offset) const;
101
102     uint8_t get_name_type() const;
103     uint8_t get_dept_type() const;
104     uint8_t get_jobt_type() const;
105     uint8_t get_date_type() const;
106
107     friend ModelComp operator== (const ModelComp& l_bool, const ModelComp& r_bool);
108     friend ModelComp operator!= (const ModelComp& l_bool, const ModelComp& r_bool);
109
110     friend std::ostream& operator« (std::ostream& stream, const ModelComp& model);
111
112     ModelComp operator! () const;
113
114 private:
115     uint8_t _value;
116 };
117
118
119 template<typename T>
120 ModelComp Model::compare_type(uint8_t mode, T r_value)
121 {
122     int8_t comp_result = 4;
123     ModelComp model_comp;
124
125     if (mode > 3)
126     {
127         mode = 0;
128     }
129
130     switch (mode)
131     {
132     case 0:
133         {
134             if (std::is_same<T, decltype(this->_full_name)>::value)
135             {
136                 comp_result = this->_full_name.compare(r_value);
137                 if (comp_result < 0)
138                 {
139                     comp_result = 1;
140                 }
141                 else if(comp_result > 0)
142                 {
143                     comp_result = 2;
144                 }
145                 else
146                 {
147                     comp_result = 0;
148                 }
149             }
150             else
151             {
152                 throw std::invalid_argument("r_value should be _full_name");
153             }
154             model_comp.set_name_type(comp_result);
155             break;
156         }
157     case 1:
158         {
159             if (std::is_same<T, decltype(this->_department)>::value)
160             {
161                 comp_result = this->_department.compare(r_value);
162                 if (comp_result < 0)
163                 {
164                     comp_result = 1;
165                 }
166                 else if(comp_result > 0)
167                 {
168                     comp_result = 2;
169                 }
170                 else
171                 {
172                     comp_result = 0;
173                 }
174             }
175             else
176             {
177                 throw std::invalid_argument("r_value should be _department");
178             }
179             model_comp.set_dept_type(comp_result);
180             break;
181         }
182     case 2:
183         {
```

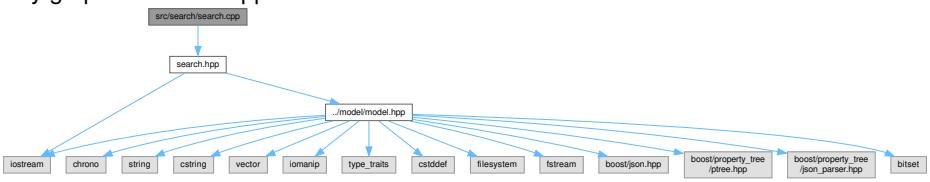
```
184             if (std::is_same<T, decltype(this->_job_title)>::value)
185             {
186                 comp_result = this->_job_title.compare(r_value);
187                 if (comp_result < 0)
188                 {
189                     comp_result = 1;
190                 }
191                 else if(comp_result > 0)
192                 {
193                     comp_result = 2;
194                 }
195                 else
196                 {
197                     comp_result = 0;
198                 }
199             }
200             else
201             {
202                 throw std::invalid_argument("r_value should be _job_title");
203             }
204             model_comp.set_jobt_type(comp_result);
205             break;
206         }
207     case 3:
208         {
209 
210             throw std::invalid_argument("r_value should be _employment_date");
211             model_comp.set_date_type(comp_result);
212             break;
213         }
214     default:
215         {
216             break;
217         }
218     }
219 
220     return model_comp;
221 }
222 
223 template<typename T>
224 T Model::get_field(uint8_t field)
225 {
226     switch (field)
227     {
228 
229         case 1:
230         {   if (std::is_same<T, decltype(this->_department)>::value)
231             {
232                 return this->_department;
233             }
234             else
235             {
236                 throw std::invalid_argument("T should be _department");
237             }
238             break;
239         }
240 
241         case 2:
242         {   if (std::is_same<T, decltype(this->_job_title)>::value)
243             {
244                 return this->_job_title;
245             }
246             else
247             {
248                 throw std::invalid_argument("T should be _job_title");
249             }
250             break;
251         }
252 
253         case 3:
254         {
255             throw std::invalid_argument("T should be _employment_date");
256             break;
257         }
258 
259         default:
260         {   if (std::is_same<T, decltype(this->_full_name)>::value)
261             {
262                 return this->_full_name;
263             }
264             else
265             {
266                 throw std::invalid_argument("T should be _full_name");
267             }
268             break;
269         }
270 
```

```
271     }
272 }
273
274
275 #endif // MODEL_HPP
```

5.7 src/search/search.cpp File Reference

This source file holds implementation of Search class.

`#include "search.hpp"`
Include dependency graph for search.cpp:



5.7.1 Detailed Description

This source file holds implementation of Search class.

>

```
This calss implements search algorithms needed for successfull completion of laboratory work 2.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright


```
(Not really)
```

5.8 search.hpp

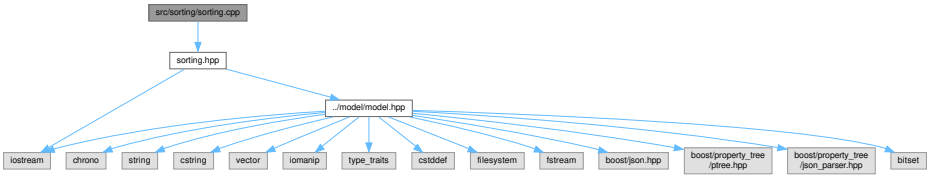
```
1
20 #ifndef SEARCH_HPP
21 #define SEARCH_HPP
22
23 #ifndef MODEL_HPP
24 #include "../model/model.hpp"
25 #endif // MODEL_HPP
26
27 #include <iostream>
28
36 class Search
37 {
38 public:
39     template<typename T>
40     static int binary_search(std::vector<Model>& model_vector, T search_value, std::uint8_t field);
41
42     template<typename T>
43     static int straight_search(std::vector<Model>& model_vector, T search_value, std::uint8_t field);
44 };
45
46
47 template<typename T>
48 int Search::binary_search(std::vector<Model>& model_vector, T search_value, std::uint8_t field)
49 {
50     int dummy = 0;
51     int left = 0;
52     int right = model_vector.size() - 1;
53     uint8_t offset = field * 2;
54
55     while (left <= right)
56     {
57         int mid = (left + right) / 2;
58
59         if (((int)model_vector.at(mid).compare_type<T>(field, search_value).get_type_masked(offset) ==
    0))
60         {
61             return mid;
62         }
63
64         else if (((int)model_vector.at(mid).compare_type<T>(field, search_value).get_type_masked(offset)
    == 1))
65         {
66             left = mid + 1;
67         }
68
69         else if (((int)model_vector.at(mid).compare_type<T>(field, search_value).get_type_masked(offset)
    == 2))
70         {
71             right = mid - 1;
72         }
73     }
74     return -1;
75 }
76
77 template<typename T>
78 int Search::straight_search(std::vector<Model>& model_vector, T search_value, std::uint8_t field)
79 {
80     uint8_t offset = field * 2;
81
82     for (std::size_t index = 0; index < model_vector.size(); ++index)
83     {
84         if (((int)model_vector.at(index).compare_type<T>(field, search_value).get_type_masked(offset) ==
    0))
85         {
86             return index;
87         }
88     }
89     return -1;
90 }
91
92 #endif // SEARCH_HPP
```

5.9 src/sorting/sorting.cpp File Reference

This source file holds implementation of Sorting class.

```
#include "sorting.hpp"
```

Include dependency graph for sorting.cpp:



5.9.1 Detailed Description

This source file holds implementation of Sorting class.

>

```
This calss implements sorting algorithms needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright

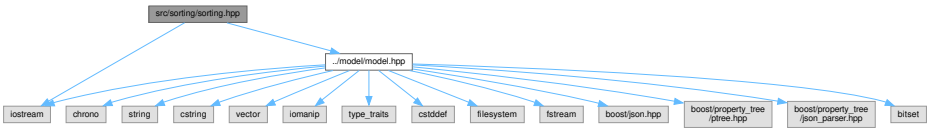

```
(Not really)
```

5.10 src/sorting/sorting.hpp File Reference

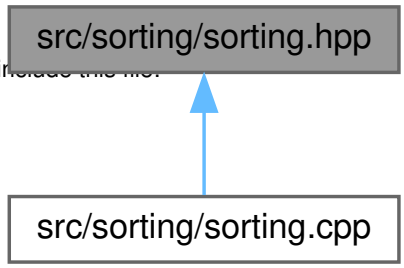
This header file holds implementation of Search class.

```
#include "../model/model.hpp"
#include <iostream>
```

Include dependency graph for sorting.hpp:



This graph shows which files directly or indirectly i



Classes

- class Sorting

  A class that provides static sorting methods for sorting a vector of Model objects based on a specific field.

5.10.1 Detailed Description

This header file holds implementation of Search class.

This header file holds implementation of Sorting class.

>

```
This calss implements search algorithms needed for successfull completion of laboratory work 2.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright


```
(Not really)
```

>

```
This calss implements sorting algorithms needed for successfull completion of laboratory work 1.
```

Author

Alexander Chudnikov (THE_CHOODICK)

Date

15-02-2023

Version

0.0.1

Warning

This library is under development, so there might be some bugs in it.

Bug Currently, there are no any known bugs.

```
In order to submit new ones, please contact me via admin@redline-software.xyz.
```

Copyright


```
(Not really)
```

5.11 sorting.hpp

Go to the documentation of this file.

```
1
20 #ifndef SORTING_HPP
21 #define SORTING_HPP
22
23 #ifndef MODEL_HPP
24 #include "../model/model.hpp"
25 #endif // MODEL_HPP
26
27 #include <iostream>
28
36 class Sorting
37 {
38 public:
39     static void bubble_sort(std::vector<Model>& model_vector, uint8_t field);
40     static void heap_sort(std::vector<Model>& model_vector, uint8_t field);
41     static void merge_sort(std::vector<Model>& model_vector, uint8_t field, std::size_t left = 0,
       std::size_t right = 0, bool initial = true);
42
43 private:
44     static void make_heap(std::vector<Model>& model_vector, std::size_t index, uint8_t field, std::size_t
       last_index = 0);
45     static void make_merge(std::vector<Model>& model_vector, uint8_t field, std::size_t left, std::size_t
       right, std::size_t middle);
46 };
47
48 #endif // SORTING_HPP
```

Chapter 6

Search statistics

6.0.1 Definition for search

FULL NAME	DEPARTMENT	JOB TITLE	DATE
MODE 0	**MODE 1**	**MODE 2**	**MODE 3**
Alexzander Oliver Baxterovna	Data Entry	Architectural Technologist	2015/09/09
...	...	...	...
Billy Barrett Okeogheneovich	Audio Engineering	Production Manager	1999/01/15

Table 6.1 Employee information

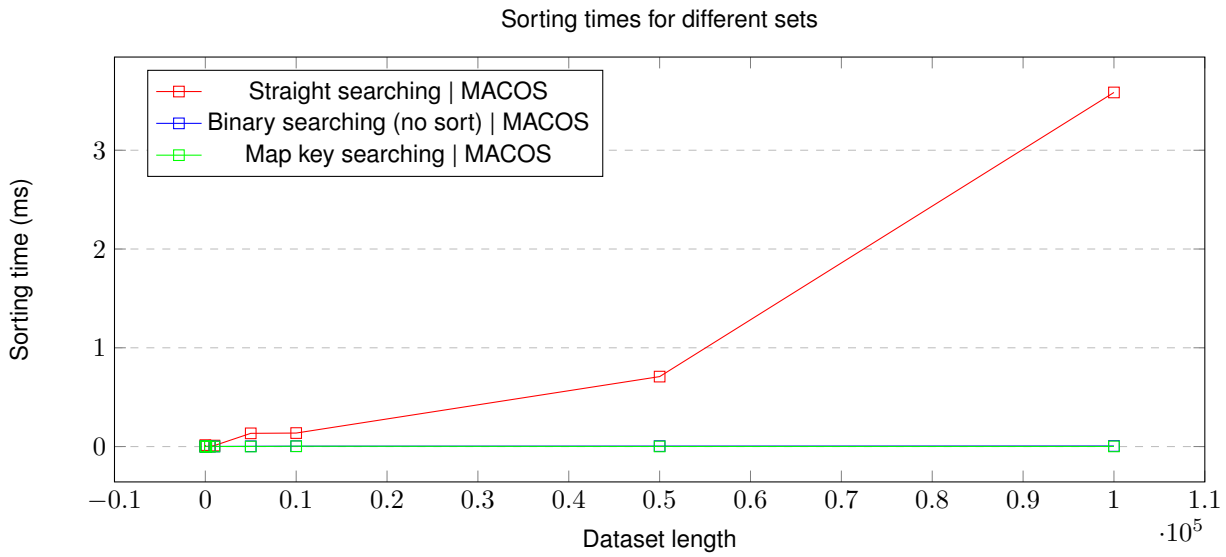
Table 6.2 MACOS specs

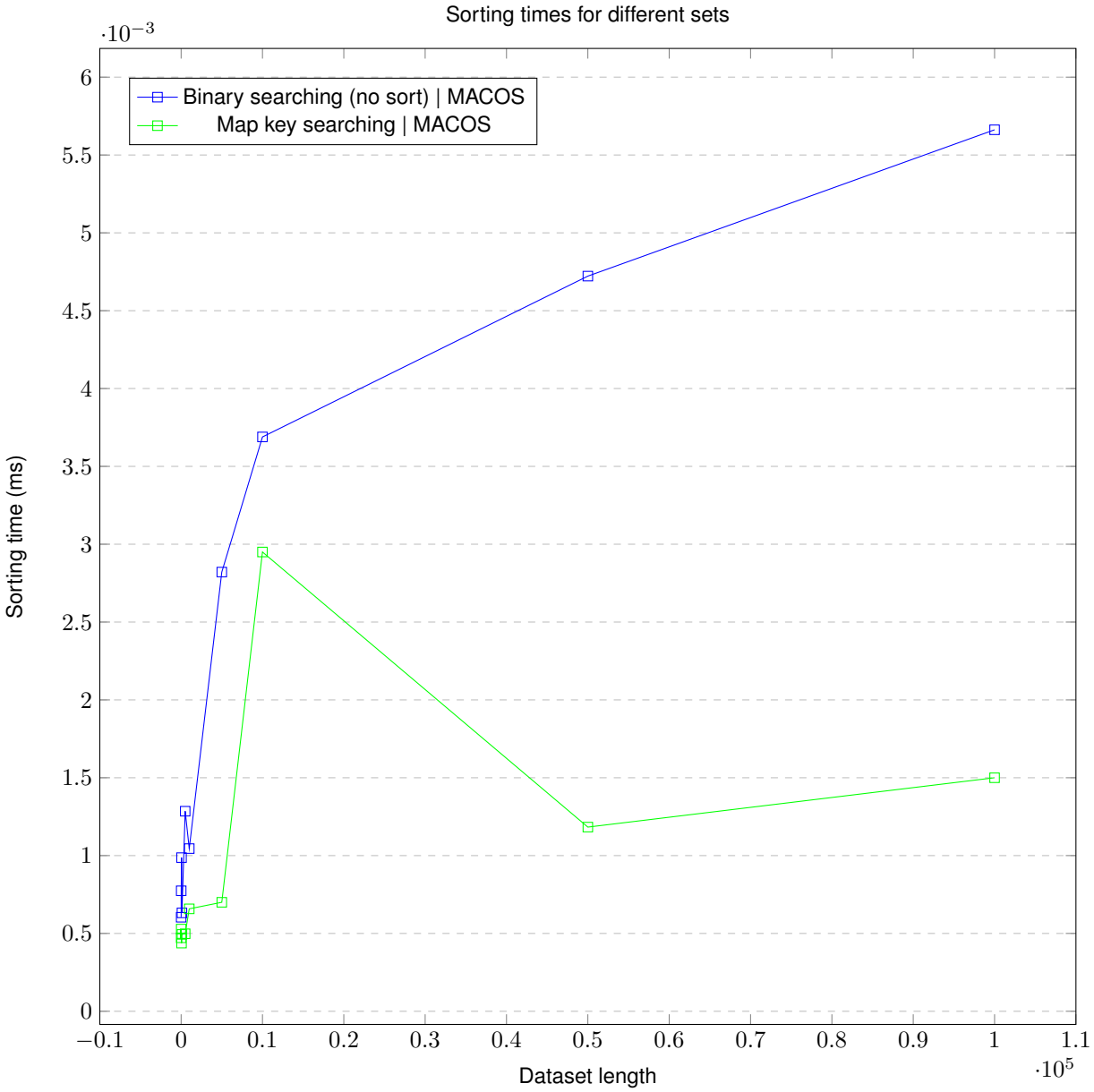
Model	MacBook Air (11-inch, Early 2015)
OS	MacOS Montery 12.6
Processor	1.6 GHz 2-core Intel Core i5
Memory	4 GB 1600 MHz DDR3
Graphics	Intel HD Graphics 6000 1536 MB

Table 6.3 WIN specs

Model	-
OS	Windows 10 Pro 20H2
Processor	Intel Core i7-8086K @ 4.50 GHz
Memory	80,0 GB 2333 MHz DDR4
Graphics	NVIDIA GeForce GTX 1080

6.0.2 Searching compare




Sorting times for different sets
$\cdot 10^{-3}$
Binary searching (no sort) | MACOS
Map key searching | MACOS
Sorting time (ms)
Dataset length
$\cdot 10^{5}$
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
−0.1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1

Index