$\begin{array}{c} \text{MP-Frolov-Lab-2} \\ \text{1.0} \end{array}$

Создано системой Doxygen 1.9.6

l Методы программирования. Лабораторная работа №2. Алгоритмы поиска.	1
1.1 Вариант_26	1
1.2 Выполнил	1
2 Алфавитный указатель классов	3
2.1 Классы	3
3 Список файлов	5
3.1 Файлы	5
4 Классы	7
4.1 Класс Element	7
4.1.1 Подробное описание	7
4.1.2 Конструктор(ы)	7
$4.1.2.1 \; ext{Element} () \; [1/2] \; \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots$	8
$4.1.2.2 \; ext{Element} () \; [2/2] \; \dots $	8
4.1.3 Методы	8
$4.1.3.1 \operatorname{getChildAmount}() \dots \dots \dots \dots \dots \dots \dots \dots \dots$	
4.1.3.2 getFam()	9
$4.1.3.3~{ m getIm}()$	9
4.1.3.4 getOt()	
4.1.3.5 getYear()	9
4.1.3.6 getYearD()	10
4.1.3.7 operator<()	
$4.1.3.8 \text{ operator} <= () \dots $	
4.1.3.9 operator=()	
4.1.3.10 operator>()	
$4.1.3.11 \text{ operator} = () \dots $	
$4.1.3.12 \operatorname{setChildAmount}() \dots \dots \dots \dots \dots \dots \dots \dots \dots$	
4.1.3.13 setFam()	
$4.1.3.14 \text{ setIm}() \dots \dots$	
4.1.3.15 setOt()	
$4.1.3.16 \text{ setYear}() \dots \dots$	
4.1.3.17 setYearD()	
5 Файлы	15
5.1 Файл Element.cpp	
5.1.1 Переменные	
5.1.1.1 alph	
5.2 Element.cpp	
5.3 Файл Element.h	
5.4 Element.h	
5.5 Файл Main.cpp	
5.5.1 Функции	
5.5.1.1 binarySearch1()	
0.0.1.1 DinaryDearent()	20

5.5.1.2 checkFIO()	21
$5.5.1.3~\mathrm{checkTwoStrings}()$	21
$5.5.1.4 \; \mathrm{findMultimap}() \; \ldots \; \ldots \; \ldots \; \ldots \; \ldots \; \ldots \; 2$	22
$5.5.1.5 \text{ heapify1}() \dots \dots$	22
$5.5.1.6~\mathrm{heapSort1}()$	23
$5.5.1.7 \ \mathrm{linearSearch}() \ \ldots \ \ldots \ \ldots \ \ldots \ \ldots \ 2$	23
$5.5.1.8~\mathrm{main}()$	23
5.6 Main.cpp	25
редметный указатель	31

Методы программирования. Лабораторная работа №2. Алгоритмы поиска.

1.1 Вариант 26

Массив данных генеологического фонда: ФИО, год рождения, год смерти, число детей (сравнение по полям - год рождения, ФИО, число детей).

Сортировка: пирамидальная.

Алгоритмы поиска: линейный, бинарный и встроенный в multimap

1.2 Выполнил

Фролов Олег, СКБ201

2	Методы программирования.	Лабораторная работа №2.	Алгоритмы поиска.

Алфавитный указатель классов

2.1 Классы

Классы с их	крат	ким (опис	аниє	М.										
Element						 		 	 		 	 	 	 	 •

Алфавитный	указатель	классов
TITOTH	y Masar Corp	MIGCOOL

Список файлов

3.1 Файлы

Полный список файлов.

Element.cpp			 														 			15
Element.h .			 														 			19
Main.cpp .			 														 			19

6 Список файлов

Классы

4.1 Класс Element

```
#include <Element.h>
```

Открытые члены

- Element ()
- Element (std::string fam, std::string im, std::string ot, int y, int yd, int ca)
- std::string getFam ()
- void setFam (std::string newFam)
- std::string getIm ()
- void setIm (std::string newIm)
- std::string getOt ()
- void setOt (std::string newOt)
- int getYear ()
- void setYear (int newYear)
- int getYearD ()
- void setYearD (int newYearD)
- int getChildAmount ()
- void setChildAmount (int newAmount)
- bool operator> (const Element & elem) const
- bool operator< (const Element &elem) const
- bool operator>= (const Element &elem) const
- bool operator <= (const Element & elem) const
- Element & operator= (Element &elem)

4.1.1 Подробное описание

См. определение в файле Element.h строка 5

4.1.2 Конструктор(ы)

8 Классы

```
4.1.2.1 Element() [1/2]
```

```
Element::Element ()
```

Стандартный конструктор.

4.1.2.2 Element() [2/2]

```
Element::Element (
std::string fam,
std::string im,
std::string ot,
int y,
int yd,
int ca )
```

Конструктор, которому передаются значения для всех полей класса.

```
C_{\rm M.} определение в файле Element.cpp строка 21 00021 this->familiya = fam; 00022 this->imya = im; 00024 this->otchestvo = ot; 00025 this->year = y; 00026 this->year_death = yd; 00027 this->child_amount = ca; 00028 }
```

4.1.3 Методы

4.1.3.1 getChildAmount()

```
int Element::getChildAmount ( )
```

Функция получения количества детей.

```
C_{\rm M.} определение в файле \ Element.cpp строка \ 103 _{00103} \ _{\rm return\ this->child\_amount;} \ _{00105} }
```

4.1 Класс Element 9

```
4.1.3.2 \quad \text{getFam}()
std::string Element::getFam ( )
Функция получения фамилии.
См. определение в файле Element.cpp строка 33
00033
00034
00035 }
        return this->familiya;
4.1.3.3 \text{ getIm}()
std::string Element::getIm ( )
Функция получения имени.
См. определение в файле Element.cpp строка 47
00047
00048
00049 }
         return this->imya;
4.1.3.4 \text{ getOt()}
std::string Element::getOt ( )
Функция получения отчества.
См. определение в файле Element.cpp строка 61
00061
00062
         return this->otchestvo;
00063 }
4.1.3.5 \text{ getYear}()
int Element::getYear ( )
Функция получения года рождения.
См. определение в файле Element.cpp строка 75 _{00075} _{\{00076\ \text{return this-}>\text{year};\ 00077\ \}}
```

10 Классы

```
4.1.3.6 \text{ getYearD}()
int Element::getYearD ( )
Функция получения года смерти.
См. определение в файле Element.cpp строка 89
00089 \\ 00090
         return this->year_death;
00091 }
4.1.3.7 operator<()
bool Element::operator< (
                 const Element & elem ) const
Перегрузка оператора < для сравнения двух объектов класса Element (стр. 7).
См. определение в файле Element.cpp строка 165
00166
         \mathbf{if} \ (\mathtt{year} \ < \ \mathtt{elem.year}) \ \{
00167
             return 1;
         00168
00169
00170
00171
                if (imya[0] == elem.imya[0]) {
    if (otchestvo[0] == elem.otchestvo[0]) {
        if (child_amount < elem.child_amount)
        return 1;
00172
00173
00174 \\ 00175
00176
                      else
00177
                         return 0;
00178
00179
                   else {
                      int first = alph.find(std::tolower(otchestvo[0]));
int second = alph.find(std::tolower(elem.otchestvo[0]));
00180
00181
00182
                      if (first > second)
00183
                         return 1;
00184
                      else
00185
                         return 0;
                   }
00186
00187
00188
00189
                   int first = alph.find(std::tolower(imya[0]));
00190
                   int second = alph.find(std::tolower(elem.imya[0]));
00191
                   if (first > second)
00192 \\ 00193
                   return 1; else
00194
                      return 0;
00195
               }
00196
00197
                int first = alph.find(std::tolower(familiya[0]));
00198
00199
                int\ second = alph.find(std::tolower(elem.familiya[0]));\\
00200
                if (first > second)
00201
                   return 1;
00202
00203
                   return 0;
00204
            }
00205
         } else {
00206
00207
            return 0;
00208
```

00209 }

4.1 Класс Element 11

```
4.1.3.8 \quad \text{operator} \leq = ()
bool Element::operator<= (
                  const Element & elem ) const
Перегрузка оператора <= для сравнения двух объектов класса Element (стр. 7).
См. определение в файле Element.cpp строка 263
00264
          if (year <= elem.year) {
00265
             return 1;
00266
          else if (year > elem.year) {
00267
00268
               /цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей
              //цикл для проверки перволего:
if (familiya[0] == elem.familiya[0]) {
    if (imya[0] == elem.imya[0]) {
00269
00270
                     \inf (\text{otchestvo}[0] == \text{elem.otchestvo}[0]) 
00271
00272
                        if (child_amount <= elem.child_amount)</pre>
00273
00274
                        else
00275
                           return 0;
00276
                    }
00277
                     else {
                        int first = alph.find(std::tolower(otchestvo[0]));
int second = alph.find(std::tolower(elem.otchestvo[0]));
00278
00279
00280
                        _{\bf if}~({\rm first}~>\equiv~{\rm second})
00281
                           return 1;
00282
                        else
00283
                           return 0;
00284
                    }
00285
00286
00287
                    int first = alph.find(std::tolower(imya[0]));
00288
                    int second = alph.find(std::tolower(elem.imya[0]));
00289
                    if (first >= second)
00290
                        return 1;
00291
                    else
00292 \\ 00293
                        return 0;
                 }
00294
00295
00296
                 int first = alph.find(std::tolower(familiya[0]));
00297
                 int second = alph.find(std::tolower(elem.familiya[0]));
00298 \\ 00299
                 if (first > \equiv second)
                     return 1;
00300
                 else
00301
                     return 0;
00302
             }
00303
00304
          else {
00305
              return 0;
00306
          }
00307 }
4.1.3.9 \quad \text{operator} = ()
 Element & Element::operator= (
                    Element & elem )
Перегрузка оператора = для сравнения двух объектов класса Element (стр. 7).
См. определение в файле Element.cpp строка 312
          this->setFam(elem.getFam());
00313
00314
          this-\!>\!setIm(elem.getIm());
          this \hspace{-0.05cm}-\hspace{-0.05cm} > \hspace{-0.05cm} setOt(elem.getOt());
00315
          this->setYear(elem.getYear());
this->setYearD(elem.getYearD());
00316
00318
          this->setChildAmount(elem.getChildAmount());
00319
          return *this;
00320 }
```

12 Классы

```
4.1.3.10 operator>()
bool Element::operator> (
                 const Element & elem ) const
Перегрузка оператора > для сравнения двух объектов класса Element (стр. 7).
См. определение в файле Element.cpp строка 117
00117
00118
          if (year > elem.year) {
00119
             return 1;
00120
           \begin{array}{l} \text{else if (year == elem.year) } \{ \end{array} 
00121
             //цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей if (familiya[0] == elem.familiya[0]) { if (imya[0] == elem.imya[0]) {
00122
00123
00124
00125
                    if (otchestvo[0] == elem.otchestvo[0]) {
00126
                      if (child_amount > elem.child_amount)
00127
                          return 1;
                       else
00128
00129
                          return 0;
00130
00131
00132
                       int\ first = alph.find(std::tolower(otchestvo[0]));
                       int second = alph.find(std::tolower(elem.otchestvo[0]));
00133
00134
                       if (first < second)
00135
                          return 1;
00136
00137
                          return 0;
00138
                   }
00139
00140
                else {
                   int\ first = alph.find(std::tolower(imya[0]));
00141
00142
                   int second = alph.find(std::tolower(elem.imya[0]));
                   if (first < second)
00143
00144
                      return 1;
00145
                   else
00146
                       return 0;
00147
                }
00148
00149
00150
                int first = alph.find(std::tolower(familiya[0]));
00151
                int\ second = alph.find(std::tolower(elem.familiya[0]));\\
00152
                if (first < second)
00153
                   return 1;
00154
                else
00155
                   return 0;
00156
00157
          } else {
00158
             return 0;
          }
00159
00160 }
4.1.3.11 \text{ operator} = ()
bool Element::operator>= (
                 const Element & elem ) const
Перегрузка оператора >= для сравнения двух объектов класса Element (стр. 7).
См. определение в файле Element.cpp строка 214
00214
00215
          if (year >= elem.year) {
00216
             return 1;
00217
00218
          else if (year < elem.year) {
00219
             //цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей
00220
             if (familiya[0] == elem.familiya[0]) {
                if (imya[0] == elem.imya[0]) {
    if (otchestvo[0] == elem.otchestvo[0]) {
        if (child_amount >= elem.child_amount)
        return 1;
00221
00222
00223
00224
00225
                       else
```

4.1 Класс Element 13

```
00226
                                 return 0;
00227
00228
                          else {
                             int first = alph.find(std::tolower(otchestvo[0]));
int second = alph.find(std::tolower(elem.otchestvo[0]));
if (first <= second)
00229
00230
00231
00232
                                 return 1;
00233
00234
                                 return 0;
00235
                         }
00236
00237
                         int first = alph.find(std::tolower(imya[0]));
int second = alph.find(std::tolower(elem.imya[0]));
00238
00239
00240
                         if (first <= second)
00241 \\ 00242
                             return 1;
                         else
00243
                             return 0;
00244
                     }
00245
00246
00247 \\ 00248
                     \label{eq:int_first} \begin{array}{l} \text{int} \ \operatorname{first} \ = \ \operatorname{alph.find}(\operatorname{std}::\operatorname{tolower}(\operatorname{familiya}[0])); \end{array}
                     \begin{array}{l} \text{int second} = \text{alph.find}(\text{std::tolower}(\text{elem.familiya}[0])); \\ \text{if } (\text{first} <= \text{second}) \end{array}
00249
00250
                         return 1;
00251
00252
                         return 0;
00253
                }
00254
00255
             else {
00256
                return 0;
             }
00257
00258~\}
4.1.3.12 set ChildAmount()
void Element::setChildAmount (
                       int newAmount )
Функция установки количества детей.
{
m Cm.} определение в файле Element.cpp строка 110 _{00110} _{00111} this->child_amount = newAmount;
            this->child_amount = newAmount;
00112 }
4.1.3.13 \operatorname{setFam}()
void Element::setFam (
                      std::string newFam )
Функция установки фамилии.
См. определение в файле Element.cpp строка 40
```

this->familiya = new Fam;

00040

00041 00042 } 14 Классы

```
4.1.3.14 \text{ set Im}()
void Element::setIm (
               std::string newIm )
Функция установки имени.
См. определение в файле _{\{00054\}} Element.cpp строка _{\{00054\}}
00055
00056 }
       this->imya= newIm;
4.1.3.15 \operatorname{set} Ot()
void Element::setOt (
               std::string newOt )
Функция установки отчества.
См. определение в файле Element.cpp строка 68
       this-> otchestvo = newOt;
00069
00070 }
4.1.3.16 setYear()
void Element::setYear (
               int newYear)
Функция установки года рождения.
См. определение в файле Element.cpp строка 82
        this->year = new Year;
00084 }
4.1.3.17 setYearD()
void Element::setYearD (
               int newYearD )
Функция установки года смерти.
{
m Cm.} определение в файле Element.cpp строка 96 00096 {
m this->_{year}} death = newYearD;
        this->year_death = newYearD;
00098 }
```

Объявления и описания членов классов находятся в файлах:

- Element.h
- Element.cpp

Файлы

5.1 Файл Element.cpp

```
#include "Element.h"
#include <string>
```

Переменные

• std::string alph = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

5.1.1 Переменные

$5.1.1.1 \quad alph$

std::string alph = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

Кириллица для сравнения ФИО.

См. определение в файле Element.cpp строка 7

5.2 Element.cpp

```
См. документацию.
00001 #include "Element.h"
00002 #include <string>
00007 std::string alph = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";
00008
00015
         this->otchestvo = "Ot";
00016 }
00017
00021\ Element:: Element(std::string\ fam,\ std::string\ im,\ std::string\ ot,\ int\ yd,\ int\ ca)\ \{
00022
         this\text{-}{>}familiya=fam;
00023
         this->imya = im;
00024
         this->otchestvo = ot;
00025
         this\text{-}{>}year=y;
         this->year_death = yd;
this->child_amount = ca;
00026
00027
00028 }
00029
00033 std::string Element::getFam() {
00034
         return this->familiya;
00035 }
00036
00040 void Element::setFam(std::string newFam) {
00041
         this->familiya = new Fam;
00043
00047 std::string Element::getIm() {
00048
         return this->imya;
00049 }
00050
00054 void Element::setIm(std::string newIm) {
         this->imya=newIm;
00056 }
00057
00061 std::string Element::getOt() {
00062
         return this->otchestvo;
00063 }
00068 void Element::setOt(std::string newOt) {
00069
         this-> otchestvo = newOt;
00070 }
00071
00075 int Element::getYear() {
00076
         return this->year;
00077 }
00078
00082 void Element::setYear(int newYear) {
00083
         this\text{-}{>}year=newYear;
00084 }
00085
00089 int Element::getYearD() {
00090
         return this->year_death;
00091 }
00092
00096 void Element::setYearD(int newYearD) {
         this->year death = newYearD;
00098 }
00099
00103 int Element::getChildAmount() {
         return this->child_amount;
00104
00105 }
00106
00110 void Element::setChildAmount(int newAmount) {
00111
         this\text{-}>child\_amount = new Amount;
00112 }
00113
00117 bool Element::operator> (const Element& elem) const {
         if (year > elem.year) {
00118
            return 1;
00120
00121
         else if (year == elem.year) {
            //цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей if (familiya[0] == elem.familiya[0]) { if (imya[0] == elem.imya[0]) {
00122
00123
00124
                  if (otchestvo[0] == elem.otchestvo[0]) {
00126
                      if (child_amount > elem.child_amount)
00127
                      else
00128
                         return 0;
00129
00130
                  }
```

5.2 Element.cpp 17

```
00131
                     else {
00132
                        int first = alph.find(std::tolower(otchestvo[0]));
00133
                        int second = alph.find(std::tolower(elem.otchestvo[0]));
00134
                        if (first < second)
00135
                            return 1;
                        else
00136
                            return 0;
00137
00138
00139
00140
                     int first = alph.find(std::tolower(imya[0]));
00141
                     int second = alph.find(std::tolower(elem.imya[0]));
00142
00143
                     if (first < second)
00144
                        return 1;
                     else
00145
00146
                        return 0;
00147
                 }
00148
00149
00150
                  int first = alph.find(std::tolower(familiya[0]));
00151
                  int second = alph.find(std::tolower(elem.familiya[0]));
00152
                  \quad \  \  \mathbf{if} \ (\mathbf{first} \, < \, \mathbf{second}) \\
00153
                     return 1;
00154
                  else
00155
                     return 0;
00156
00157
          } else {
00158
              return 0;
          }
00159
00160 }
00161
00165 bool Element::operator< (const Element& elem) const {
00166
          if (year < elem.year) {
00167
              return 1;
00168
          else if (year > elem.year) {
00169
00170
               /цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей
              //цикл для проверки первопольсь — if (familiya[0] == elem.familiya[0]) {
00171
00172
                  if (imya[0] == elem.imya[0]) {
00173
                     if (otchestvo[0] == elem.otchestvo[0]) {
00174
                        \begin{array}{l} \textbf{if} \ (\textbf{child} \_ \textbf{amount} < \textbf{elem.child} \_ \textbf{amount}) \end{array}
00175
                            return 1;
00176
                        else
00177
                            return 0;
00178
                     else {
00179
00180
                        \label{eq:int_first} int \ first = \ alph.find(std::tolower(otchestvo[0]));
                        int\ second = alph.find(std::tolower(elem.otchestvo[0]));\\
00181
00182
                        if (first > second)
00183
                            return 1:
00184
                        else
00185
                            return 0;
00186
                     }
00187
                  else {
00188
00189
                     int first = alph.find(std::tolower(imya[0]));
00190
                     int second = alph.find(std::tolower(elem.imya[0]));
00191
                     if (first > second)
00192
                        return 1;
                     else
00193
00194
                        return 0;
00195
                 }
00196
00197
                  int first = alph.find(std::tolower(familiya[0]));
00198
                  int\ second = alph.find(std::tolower(elem.familiya[0]));\\
00199
                 if (first > second)
return 1;
00200
00201
                  else
00202
00203
                     return 0;
00204
              }
00205
          else {
00206
00207
              return 0;
00208
          }
00209 }
00210
00214 bool Element::operator>= (const Element & elem) const {
00215
          if (year > = elem.year) {
00216
              return 1:
00217
00218
          else if (year < elem.year) {
00219
                /цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей
00220
              if (familiya[0] == elem.familiya[0]) {
                  if (imya[0] == elem.imya[0]) {
   if (otchestvo[0] == elem.otchestvo[0]) {
     if (child_amount >= elem.child_amount)
00221
00222
00223
```

```
00224
                           return 1;
00225
00226
                            return 0;
00227
                     else {
00228
                        int first = alph.find(std::tolower(otchestvo[0]));
00229
00230
                         int second = alph.find(std::tolower(elem.otchestvo[0]));
00231
                         if (first <= second)
                           return 1;
00232
                        else
00233
00234
                           return 0:
00235
                     }
00236
00237
                     \label{eq:int_state} \begin{split} &\inf \ first \, = \, alph.find(std::tolower(imya[0])); \end{split}
00238
00239 \\ 00240
                     int\ second = alph.find(std::tolower(elem.imya[0]));\\
                     if (first <= second)
00241
                        return 1;
00242
00243
                        return 0;
00244
00245
00246
                 int first = alph.find(std::tolower(familiya[0]));
int second = alph.find(std::tolower(elem.familiya[0]));
00247
00248
00249
                  if (first <= second)
00250
                  else
00251
                     return 0;
00252
00253
              }
00254
          } else {
00255
00256
              return 0;
          }
00257
00258 }
00259
00263 bool Element::operator<= (const Element& elem) const {
00264
          if (year <= elem.year) {
00265
              return 1;
00266
          else if (year > elem.year) {
00267 \\ 00268
              //цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей if (familiya[0] == elem.familiya[0]) {
    if (imya[0] == elem.imya[0]) {
00269
00270
00271
                     if (otchestvo[0] == elem.otchestvo[0]) {
00272
                        if (child_amount <= elem.child_amount)</pre>
00273
                            return 1;
00274
                        else
00275
                           return 0;
00276
                     }
else {
00277
00278
                        int first = alph.find(std::tolower(otchestvo[0]));
00279
                        int\ second = \ alph.find(std::tolower(elem.otchestvo[0]));
00280
                        if (first >= second)
00281
                            return 1;
00282
                        else
00283
                           return 0;
00284
                     }
00285
                  else {
00286
00287
                     int first = alph.find(std::tolower(imya[0]));
00288
                     int second = alph.find(std::tolower(elem.imya[0]));
00289
                     if (first >= second)
00290
                        return 1;
                     else
00291
00292
                        return 0;
00293
                 }
00294
00295
00296
                  int first = alph.find(std::tolower(familiya[0]));
00297
                  int\ second = alph.find(std::tolower(elem.familiya[0]));\\
00298
                  if (first >= second)
00299
                 return 1; else
00300
00301
                     return 0;
00302
             }
00303
00304
           else {
00305
              return 0;
00306
          }
00307 }
00308
00312 Element& Element::operator= (Element& elem) {
00313
          this->setFam(elem.getFam());
00314
          this-\!>\!setIm(elem.getIm());
00315
          this->setOt(elem.getOt());
00316
          this->setYear(elem.getYear());
```

5.3 Файл Element.h 19

```
00317 this->setYearD(elem.getYearD());
00318 this->setChildAmount(elem.getChildAmount());
00319 return *this;
```

5.3 Файл Element.h

```
#include <iostream>
#include <string>
```

Классы

• class Element

5.4 Element.h

```
См. документацию.
00001 #pragma once
00002 #include <iostream>
00003 #include <string>
00004
00005 class Element {
00006 private:

00007 std::string familiya = "";

00008 std::string imya = "";
             std::string otchestvo = "";
00009
00010
             int year = 1900;
             int year_death = 1920;
int child amount = 0;
00011 \\ 00012
             int sex = 0;
00013
00014 public:
00015
00016
             Element(std::string fam, std::string im, std::string ot, int y, int yd, int ca);
00017
             std::string getFam()
             {\tt void} \ {\tt setFam}({\tt std} {\tt ::string} \ {\tt newFam});\\
00018
00019
             std::string getIm();
void setIm(std::string newIm);
00020
00021
             std::string getOt();
00022
             void setOt(std::string newOt);
00023
             int getYear();
00024 \\ 00025
             void setYear(int newYear);
int getYearD();
void setYearD(int newYearD);
00026
00027
             int getChildAmount();
00028
             void setChildAmount(int newAmount);
             bool operator> (const Element& elem) const;
bool operator< (const Element& elem) const;
bool operator>= (const Element& elem) const;
bool operator<= (const Element& elem) const;
00029
00030
00031
00032
00033
             Element & operator = (Element & elem);
00034 };
```

5.5 Файл Маіп.срр

```
#include <iostream>
#include <fstream>
#include "Element.h"
#include <vector>
#include <chrono>
#include <cstdlib>
#include <map>
#include <sstream>
```

Функции

- void findMultimap (std::multimap< std::string, int >map, std::string searchQuery)
- void linearSearch (std::vector< Element > spisok, std::string elem)
- int checkTwoStrings (std::string firstS, std::string secondS)
- int checkFIO (Element elem1, Element elem2, int flag)
- void binarySearch1 (std::vector< Element > spisok, std::string query)
- void heapify1 (std::vector< Element > &spisok, int N, int i)
- void heapSort1 (std::vector< Element > &spisok, int N)
- int main ()

5.5.1 Функции

5.5.1.1 binarySearch1()

Функция для бинарного поиска заданного элемента в контейнере объектов класса Element (стр. 7).

```
См. определение в файле Маіп.срр строка 128
00128
          int low = 0;
00129
00130
          int high = spisok.size() - 1;
00131
          int mid:
00132
00133
          std::stringstream ss(query);
00134
          std::vector<std::string> words;
00135
          std::string word;
00136
          Element elem:
00137
          while (ss * word) {
             words.push\_back(word);
00138
00139
00140
          elem.setFam(words[0]);
00141
          elem.setIm(words[1]);
00142
          elem.setOt(words[2]);
00143
          while (low \leq high) {
mid = (high + low) / 2;
00144
             iff (checkFIO(spisok[mid], elem, 2)==2) {
    std::cout « query « " was found at index " « mid « "\n";
00145
00146
00147
00148
             else if (checkFIO(spisok[mid], elem, 0)) {
   high = mid - 1;
00149
00150
00151
00152
00153
                low = mid + 1;
00154
00155
          if (low > high)
00156
00157
             std::cout « query « " is not in this array!\n";
00158 }
```

5.5 Файл Main.cpp 21

5.5.1.2 checkFIO()

```
int checkFIO (

Element elem1,

Element elem2,

int flag )
```

Функция для проверки равенства ФИО двух объектов класса Element (стр. 7).

```
См. определение в файле Маіп.срр строка 87
00087
00088
          std::string alph = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";
          std::string fio = "";
std::string fio2 = ""
00089
00090
00091
          fio = elem1.getFam();
00092
          fio2 = elem2.getFam();
00093
          int first = alph.find(std::tolower(fio[0]))
00094 \\ 00095
          int second = alph.find(std::tolower(fio2[0]));
          if (first == second) {
    for (int i = 0; i < fio.length() && i < fio2.length() && first == second; ++i) {</pre>
00096
00097
                first = alph.find(std::tolower(fio[i]));
00098
                 second = alph.find(std::tolower(fio2[i]));
00099
00100
          if (first == second) {
00101
             fio = elem1.getIm();

fio2 = elem2.getIm();
00102
00103
00104
             for (int i = 0; i < \text{fio.length}() && i < \text{fio2.length}() && first == second; ++i) {
00105
                 first = alph.find(std::tolower(fio[i]));
00106
                 second = alph.find(std::tolower(fio2[i]));
00107
00108
          00109
             fio = elem1.getÓt();
00110
00111
             fio2 = elem 2.getOt();
00112
             for (int i = 0; i < fio.length() && i < fio2.length() && first == second; ++i) {
00113
                 first = alph.find(std::tolower(fio[i]))
                 second = alph.find(std::tolower(fio2[i]));
00114
00115
00116
00117
          if (first > second)
00118
             return 1;
          else if (flag == 2 && first == second)
00119
00120
             return 2;
00121
          else
00122
             return 0;
00123 }
```

5.5.1.3 checkTwoStrings()

Функция для сравнения двух строк по кириллице. Если в ходе цикла, пробегающегося по всем буквам двух строк,

в одном из двух слов встречаются разные буквы, то та строка, чья буква встречается в алфавите раньше, считается большей.

```
См. определение в файле Маіп.срр строка 60
```

```
00060
00061
         std::string alph = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";
00062
          //if true return 1, if false return 0
00063
         int first = alph.find(std::tolower(firstS[0]));
00064
         int second = alph.find(std::tolower(secondS[0]));
00065
         if (first == second) {
00066
            int count = 0;
00067
00068
             while (first == second) {
00069
                first = alph.find(std::tolower(firstS[count]));
```

```
00070
                second = alph.find(std::tolower(secondS[count]));
00071
                firstS.insert(count, "");
00072
                 ++count;
00073
             formula = 0 } if (first < second) {
00074
00075
                return 0;
00076
             } else
00077
                return 1;
          } else if (first < second)
00078
00079
             return 0;
00080
          else
00081
             return 1:
00082 }
```

5.5.1.4 findMultimap()

Функция для поиска заданного элемента в multimap, который содержит объекты класса Element (стр. 7).

5.5.1.5 heapify1()

```
void heapify1 ( std::vector < \  \, Element \, > \& \ spisok, \\ int \ N, \\ int \ i \ )
```

Функция для построения кучи, требуемой для корректной работы пирамидальной сортировки.

```
См. определение в файле Маіп.срр строка 163
00163
          \begin{array}{l} {\rm int\ largest} = i; \\ {\rm int\ left} = 2\ ^*\ i + 1; \\ {\rm int\ right} = 2\ ^*\ i + 2; \end{array}
00164
00165
00166
          00167
          largest = left; if (right<N && /*spisok[right]>spisok[largest]*/checkFIO(spisok[right], spisok[largest], 0)==1)
00168
00169
              largest = right;
00170
00171
          if (largest != i) {
00172
              Element temp
             temp = spisok[i];
spisok[i] = spisok[largest];
spisok[largest] = temp;
00173
00174
00175
00176
              heapify1(spisok, N, largest);
00177
00178 }
```

5.5 Файл Main.cpp 23

```
5.5.1.6 heapSort1()
void heapSort1 (
                 std::vector< Element > & spisok,
                 int N)
Функция для пирамидальной сортировки контейнера из объектов класса Element (стр. 7).
См. определение в файле Маіп.срр строка 183
         for (int i = N / 2 - 1; i >= 0; --i) {
00184
00185
            heapify1(spisok, N, i);
00186
         for (int i = N - 1; i >= 0; --i) {
00187
            Element temp;
00188
            temp = spisok[0];
spisok[0] = spisok[i];
spisok[i] = temp;
00189
00190
00191
            heapify1(spisok,\ i,\ 0);
00192
00193
00194 }
5.5.1.7 linearSearch()
void linearSearch (
                 std::vector< Element > spisok,
                 std::string elem )
Функция для прямого поиска заданного элемента в контейнере объектов класса Element (стр. 7).
См. определение в файле Маіп.срр строка 38
         \quad \  \  for \ (int \ i = 0; \ i < spisok.size(); \ ++i) \ \{
00039
00040
             std::string fio =
            fio.append(spisok[i].getFam());
fio.append(" ");
00041
            fio.append(" ");
fio.append(spisok[i].getIm());
00042
00043
            fio.append(spisok[i].getOt());
00044
00045
00046
             if (fio == elem) {
                std::cout « elem « " was found at index " « i « "\n";
00047
00048
00049
00050
            else if (i + 1 == spisok.size()) {
00051
                std::cout « elem « " is not in this array!\n";
00052
00053
00054 }
5.5.1.8 \, \text{main}()
int main ()
См. определение в файле Маіп.срр строка 196
00196
00197
00198
         std::system("chcp 1251");
00199
         int file_number = 0;
00200
         std::vector<Element> spisok;
00201
         std::multimap\!<\!std::string,\ int\!>\ map;
         int choosedAlg = 0;
std::cout « "\nChoose search algorithm (0 - linear (already sorted), 1 - binar (already sorted), 2 - sort and binar, 3 -
00202
00203
```

00204

multimap (search by key)): \n";

std::cin » choosedAlg;

```
while (choosedAlg < 0 || choosedAlg >= 4 || std::cin.fail()) {
00206
               if (std::cin.fail()) {
00207
                   std::cin.clear();
00208
                   std::cin.ignore(std::numeric\_limits < std::streamsize > ::max(), \ '\n'); std::cout \ \ ''Your input is not a number!'' \ \ ''\n';
00209
00210
                   choosedAlg = -1;
00211
00212
                   std::cout « "Your input must be in interval 0-3!" « '\n';
00213
00214
                   choosed Alg\,=\,0;
00215
               std::cout « "Input again!" « '\n';
00216
00217
               std::cin » choosedAlg;
00218
00219
            std::cout « "Choose file(0-7)" « '\n';
           std::cin > file_number;
while (file_number < 0 || file_number >= 8 || std::cin.fail()) {
    if (std::cin.fail()) {
00220 \\ 00221
00222
00223
                   std::cin.clear();
00224
                   std::cin.ignore(std::numeric limits<std::streamsize>::max(), '\n');
00225
                   std::cout « "Your input is not a number!" « '\n';
00226
                   file\_number = -1;
00227
00228
00229
                   std::cout « "Your input must be in interval 0-7!" « '\n';
00230
                   file_number = 0;
00231
00232
               std::cout « "Input again!" « '\n';
               std::cin » file_number;
00233
00234
           setlocale(LC_ALL, "rus");
std::string line = "";
std::string filename = "C:\\Users\\fov-2\\source\\repos\\data_";
00235
00236
00237
           00238
00239 \\ 00240
           std::ifstream in(filename);
           if (in.is_open()) {
   while (get line(in, line)) {
00241
00242
00243
                   std::string word =
00244
                   int flag = 0;
                   int amount = 0;
00245 \\ 00246
                  for (int i = 0; i < line.size(); ++i) {
    if (line[i] == '(')
        flag = 2;
00247
00248
00249
00250
                       else if (line[i] == ')')
00251
                          flag = 0;
                      else if (line[i] == ' ' || line[i] == '\n') {
    if (amount == 0) {
00252
00253
                              human.setFam(word);
00254
00255
00256
                           else if (amount == 1)
00257
                              human.setIm(word);
                          else if (amount == 2)
human.setOt(word);
00258
00259
00260
                          else if (amount == 3)
00261
                              human.setYear(std::stoi(word));
00262
                           else if (amount = \hat{=} 4)
                              \label{eq:human.setYearD} \\ \underset{\mbox{\tiny std}}{\text{\tiny human.setYearD}}(\overset{'}{\text{\tiny std}} :: stoi(word));
00263
00264
                           word =
00265
                           ++amount;
00266
00267
                       else if (flag == 0) {
                          char symbol = line[i];
if (symbol != ' ') {
00268
00269
00270 \\ 00271
                              word.push_back(symbol);
00272
                      }
00273
00274
                      if (amount == 5) {
00275
                          human.setChildAmount(line[line.size() - 1] - '0');
00276
00277
                           spisok.push_back(human);
00278
00279
                  }
00280
               }
00281
00282
           in.close();
00283
            \stackrel{//}{\text{for}} (\text{int } i = 0; i < spisok.size(); ++i) \ \{
00284
00285
               std::string fio =
00286
               fio.append(spisok[i].getFam());
00287
               fio.append(" ");
               fio.append(spisok[i].getIm());
00288
               fio.append("");
fio.append(spisok[i].getOt());
00289
00290
               map.insert(std::make_pair(fio, i));
00291
```

5.6 Main.cpp 25

```
00292
00293
               std::cout « spisok.size() « std::endl;
std::cout « "\nParsing done!\n" « std::endl;
00294
00295
00296
00297
               while (choosedAlg != -2) {
00298
                   std::string searchQuery;
00299
                    std::cin.get();
00300
                    std::cout « "\n\nInput your search query, please!\n";
                   getline(std::cin, searchQuery);
std::cout « "\n" « searchQuery « "\n";
if (choosedAlg == 0) {
   //check time of linear search!
00301
00302
00303
00304
00305
                         std::cout « "\n\nLinear searching started!\n";
00306
                         auto begin = std::chrono::steady_clock::now();
                        linearSearch(spisok, searchQuery);
auto end = std::chrono::steady_clock::now();
auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
std::cout « "The time: " « elapsed_ms.count() « " ms";
std::cout « "\nLinear searching ended!\n\n";
00307 \\ 00308
00309
00310
00311
00312
                    else if (choosedAlg == 1) {
    //sort and check time of binary search!
    std::vector<Element> res = spisok;
00313
00314
00315
00316
                         heapSort1(res, res.size());
                        std::cout « "\n\nBinary searching started!\n";
auto begin = std::chrono::steady_clock::now();
00317
00318
00319
                         binarySearch1(res, searchQuery);
                        auto end = std::chrono::steady_clock::now();
auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
std::cout « "The time: " « elapsed_ms.count() « " ms";
std::cout « "\nBinary searching ended!\n\n";
00320
00321
00322
00323
00324
00325
                    else if (choosedAlg == 2) {
                        //check time of sort and binary search both! std::cout « "\nHeap started!\n";
00326 \\ 00327
                        auto begin = std::chrono::steady_clock::now();
std::vector<Element> res = spisok;
00328
00329
00330
                         heapSort1(res, res.size());
                         std::cout « "\Heap ended!\n";
std::cout « "\nBinary searching started!\n";
00331
00332 \\ 00333
                        binary Search1 (res, searchQuery);
                        binary searchites, searching early, std::cout « "\nBinary searching ended!\n"; auto end = std::chrono::steady_clock::now(); auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin); std::cout « "The time: " « elapsed_ms.count() « " ms\n\n";
00334
00335
00336
00337
00338
                    else if (choosedAlg == 3) {
//find in multimap by key
00339
00340
                         auto begin = std::chrono::steady_clock::now();
00341
00342
                         findMultimap(map, searchQuery);
00343
                         auto end = std::chrono::steady_clock::now();
                        auto\ elapsed\_ms = std::chrono::duration\_cast < std::chrono::milliseconds > (end\ -\ begin); \\ std::cout\  \  \, "The\ time:\  \  \, "\  \  elapsed\_ms.count()\  \  \, "\  \  ms"; \\ \\
00344
00345
00346
                    std::cout « "\n\nEnd!\n\n";
std::cout « "Would you like to search again (0 - linear (already sorted), 1 - binary (already sorted), 2 - sort and
00347
00348
           binary, 3 - multimap and search by key, -2 - end)?\n";
00349
                    std::cin » choosedAlg;
00350
                    std::cout « "\nAgain.\n";
00351
00352
               return 0;
00353 }
```

5.6 Main.cpp

```
См. документацию.
00001 \#include <iostream>
00002 #include <fstream>
00003 #include "Element.h"
00004 #include <vector>
00005 #include <chrono>
00006 \ \# include < cstdlib >
00007 #include <map>
00008 #include <sstream>
00009
00025 void findMultimap(std::multimap<std::string, int>map, std::string searchQuery) {
          auto search = map.find(searchQuery);
if (search != map.end()) {
   std::cout « "\nFound " « search->first « " at " « search->second « "\n";
00026
00027
00028
00029
00030
```

```
00031
               std::cout « "\nNot Found\n";
00032
00033 }
00034
00038 void linearSearch(std::vector<Element> spisok, std::string elem) {
           for (int i = 0; i < spisok.size(); ++i) {
   std::string fio = "";
00039
00040
00041
               fio.append(spisok[i].getFam());
00042
               fio.append(" ");
               \begin{array}{l} \text{fio.append}(\operatorname{spisok}[i].\operatorname{getIm}());\\ \text{fio.append}(""); \end{array}
00043
00044
               fio.append(spisok[i].getOt());
if (fio == elem) {
00045
00046
00047
                   std::cout « elem « " was found at index " « i « "\n";
00048
00049 \\ 00050
               else if (i + 1 == spisok.size()) {
00051
                  std::cout « elem « " is not in this array!\n";
00052
00053
           }
00054 }
00055
00060 int checkTwoStrings(std::string firstS, std::string secondS) {
00061 std::string alph = "абвгдеёжзийклинопрстуфхцчшцьыьэюя";
00062 //if true return 1, if false return 0
00063
            int first = alph.find(std::tolower(firstS[0]));
00064
           int\ second = alph.find(std::tolower(secondS[0]));
00065
           \mathbf{if}\;(\mathrm{first}\;==\;\mathrm{second})\;\{
00066
               int count = 0;
00067
00068
               while (first == second) {
00069
                   first = alph.find(std::tolower(firstS[count]));
                  second = alph.find(std::tolower(secondS[count]));
firstS.insert(count, " ");
00070
00071
                   ++count;
00072
00073
               \label{eq:cond} \left.\begin{array}{l} \\ \text{if (first < second) } \end{array}\right. \{
00074
00075
                  return 0;
00076
               } else
00077
                  return 1;
00078
           } else if (first < second)
00079
              return 0;
           else
00080
00081
               return 1;
00082 }
00083
00087 int checkFIO(Element elem1, Element elem2, int flag) {
           std::string alph = "абвгдеёжзийклмнопрстуфхцчищьыьэюя"; std::string fio = ""; std::string fio2 = "";
00088
00089
00090
00091
           fio = elem1.getFam();
00092
           fio2 = elem2.getFam();
00093
           int first = alph.find(std::tolower(fio[0]));
00094
           int\ second = alph.find(std::tolower(fio2[0]));\\
00095
           00096
00097
                   first = alph.find(std::tolower(fio[i]));
00098
                   second = alph.find(std::tolower(fio2[i]));
00099
               }
00100
           }
if (first == second) {
00101
00102
               fio = elem1.getIm();
               fio2 = elem 2.get Im();
00103
00104
               for (int i = 0; i < fio.length() && i < fio2.length() && first == second; ++i) {
00105
                   first = alph.find(std::tolower(fio[i]));
00106
                   second = alph.find(std::tolower(fio2[i]));\\
00107
00108
           if (first == second) {
00109
00110
               fio = elem1.getOt();
00111
               fio2 = elem 2.getOt();
                \begin{tabular}{ll} for (int \ i = 0; \ i < fio.length() \ \&\& \ i < fio2.length() \ \&\& \ first == second; \ ++i) \ \{ \end{tabular} 
00112
                  first = alph.find(std::tolower(fio[i]));
00113
00114
                   second = alph.find(std::tolower(fio2[i]));
00115
               }
00116
00117
           if (first > second)
00118
               return 1;
           else if (flag == 2 && first == second)
  return 2;
00119
00120
           else
00121
00122
               return 0;
00123 }
00124
00128 void binarySearch1(std::vector<Element> spisok, std::string query) {
00129
           int low = 0:
           int high = spisok.size() - 1;
00130
```

5.6 Main.cpp 27

```
00131
           int mid;
00132
00133
           std::stringstream ss(query);
00134
           std::vector{<}std::string{>}\ words;
           \mathbf{std} :: \mathbf{string} \ \mathbf{word};
00135
00136
           Element elem:
00137
           while (ss » word) {
00138
               words.push_back(word);
00139
           elem.setFam(words[0]);
00140
           elem.setIm(words[1]);
00141
           elem.setOt(words[2]);
00142
           while (low <= high) {
mid = (high + low)
00143
00144
               if (checkFIO(spisok[mid], elem, 2)==2) {
    std::cout « query « " was found at index " « mid « "\n";
00145
00146
00147
00148
               else if (checkFIO(spisok[mid], elem, 0)) {
00149
00150
                  high = mid - 1;
00151
               else {
00152
00153
                  low = mid + 1;
00154
               }
00155
00156
           if (low > high)
00157
               std::cout « query « " is not in this array!\n";
00158 }
00159
00163 void heapify1(std::vector<Element>& spisok, int N, int i) {
00164
           int\ largest = i;
00165
           int left = 2 * i + 1;
00166
           int right = 2 * i + 2;
           if \ (left < N \ \& \& \ /*spisok[left] > spisok[largest]*/checkFIO(spisok[left], \ spisok[largest], \ 0) = = 1)
00167
           \begin{array}{l} largest = left; \\ if (right < N & & /*spisok[right] > spisok[largest]*/checkFIO(spisok[right], spisok[largest], 0) == 1) \end{array}
00168
00169
00170
               largest = right;
           if (largest != i) {
00171
00172
               Element temp;
               temp = spisok[i];
spisok[i] = spisok[largest];
spisok[largest] = temp;
00173
00174
00175
00176
               heapify1(spisok, N, largest);
00177
00178 }
00179
00183 void heapSort1(std::vector < Element > & spisok, int N) {
            \begin{array}{ll} \text{for (int } i=N \text{ / } 2\text{ - }1; i>=0; --i) \text{ } \\ \text{heapify1(spisok, N, i);} \end{array} 
00184
00185
00186
00187
           for (int i = N - 1; i >= 0; --i) {
00188
               Element temp;
               temp = spisok[0];

spisok[0] = spisok[i];

spisok[i] = temp;

heapify1(spisok, i, 0);
00189
00190
00191
00192
00193
           }
00194 }
00195
00196 int main() {
00197
           std::system("chcp 1251");
00198
00199
           int file number = 0;
00200
           std::vector<Element> spisok;
00201
           std::multimap\!<\!std::string,\ int\!>\,map;
           int choosedAlg = 0; std::cout « "\nChoose search algorithm (0 - linear (already sorted), 1 - binar (already sorted), 2 - sort and binar, 3 -
00202
00203
        multimap (search by key)): n'';
           while (choosedAlg < 0 || choosedAlg >= 4 || std::cin.fail()) {
00204
00205
               if (std::cin.fail()) {
00206
00207
                   std::cin.clear();
                   std::cin.ignore(std::numeric\_limits < std::streamsize > ::max(), \ '\n'); \\ std::cout \ \ ''Your input is not a number!'' \ \ '\n'; \\
00208
00209
00210
                   choosedAlg = -1;
00211
00212
               else {
00213
                   std::cout « "Your input must be in interval 0-3!" « '\n';
00214
                   choosed Alg\,=\,0;
00215
00216
               std::cout « "Input again!" « '\n';
               std::cin » choosedAlg;
00217
00218
           std::cout « "Choose file(0-7)" « '\n';
00219
           std::cin » file_number;
while (file_number < 0 || file_number >= 8 || std::cin.fail()) {
00220
00221
               if (std::cin.fail()) {
00222
```

```
00223
                    std::cin.clear();
                    std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
std::cout « "Your input is not a number!" « '\n';
00224
00225
00226
                    file_number = -1;
00227
00228
00229
                    std::cout « "Your input must be in interval 0-7!" « '\n';
00230
                    file number = 0;
00231
                 std::cout « "Input again!" « '\n';
00232
00233
                std::cin » file_number;
00234
            setlocale(LC_ALL, "rus");
std::string line = "";
00235
00236
            std::string\ line = ""; \\ std::string\ filename = "C:\Users\fov-2\source\repos\data\_"; \\
00237
00238 \\ 00239
            \label{lem:continuity} filename.append(std::to\_string(file\_number)); \\ filename.append(".txt");
00240
            std::ifstream in(filename);
            if (in.is_open()) {
    while (getline(in, line)) {
    std::string word = "";
00241
00242
00243
00244
                    int flag = 0;
00245
                    int amount = 0;
                    Element human;

for (int i = 0; i < line.size(); ++i) {

  if (line[i] == '(')

  flag = 2'
00246
00247
00248
                        flag = 2;
else if (line[i] == ')')
00249
00250
                        flag = 0;
else if (line[i] == '\'n') {
00251
00252
                            if (amount == 0)
00253
00254
                                human.setFam(word);
00255
00256
                             else if (amount == 1)
00257 \\ 00258
                                human.setIm(word);
                             else if (amount == 2)
00259
                                human.setOt(word);
00260
                             else if (amount == 3)
00261
                                human.setYear(std::stoi(word));
00262
                             else if (amount = \hat{=} 4)
                                \label{eq:human.setYearD} \\ \text{human.setYearD}(\overset{'}{\text{std}}::stoi(word)); \\
00263 \\ 00264
                             word =
00265
                             ++amount;
00266
00267
                         else if (flag == 0) {
                            char symbol = line[i];
if (symbol != ' ') {
00268
00269
00270
                                word.push_back(symbol);
00271
                            }
00272
                        }
00273
00274
                        if (amount == 5) \ \{
00275
                             amount = 0;
                             human.setChildAmount(line[line.size() - 1] - '0');
00276 \\ 00277
                             spisok.push_back(human);
00278
00279
                    }
00280
                }
00281
            in.close();
00282
00283
             \stackrel{\text{\it for }}{\text{for }} (\text{int } i = 0; i < \underset{\dots}{\text{spisok.size}}(); ++i) \ \{
00284
00285
                 std::string fio =
                fio.append(spisok[i].getFam());
fio.append(" ");
00286
00287
                fio.append(spisok[i].getIm());
00288
00289
                fio.append('
00290
                 fio.append(spisok[i].getOt());
                {\tt map.insert(std::make\_pair(fio,\ i));}
00291
00292
00293
            std::cout « spisok.size() « std::endl;
std::cout « "\nParsing done!\n" « std::endl;
00294
00295
00296
00297
            while (choosed Alg != -2) {
00298
                std::string searchQuery;
                std::cin.get();
std::cout « "\n\nInput your search query, please!\n";
00299
00300
                getline(std::cin, searchQuery);
std::cout « "\n" « searchQuery « "\n";
if (choosedAlg == 0) {
00301
00302
00303
00304
                       check time of linear search!
00305
                    std::cout « "\n\nLinear searching started!\n";
                    auto begin = std::chrono::steady clock::now();
00306
                    linear Search (spisok,\ search Query \overline{)};
00307
                    auto end = std::chrono::steady_clock::now();
auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
00308
00309
```

5.6 Main.cpp 29

```
std::cout \  \, "The \  \, time: "\  \, elapsed\_ms.count() \  \, « "\  \, ms"; \\ std::cout \  \, « "\nLinear \  \, searching \  \, ended!\n\n"; \\
00310
00311
00312
                     else if (choosedAlg == 1) {
   //sort and check time of binary search!
   std::vector<Element> res = spisok;
00313
00314
00315
00316
                           heapSort1(res, res.size());
                           std::cout « "\n\nBinary searching started!\n"; auto begin = std::chrono::steady_clock::now();
00317
00318
                          binary Search1(res, searchQuery);
auto end = std::chrono::steady_clock::now();
auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
std::cout « "The time: " « elapsed_ms.count() « " ms";
std::cout « "\nBinary searching ended!\n\n";
00319
00320
00321
00322
00323
00324
                     else if (choosedAlg == 2) {
    //check time of sort and binary search both!
    std::cout « "\nHeap started!\n";
    auto begin = std::chrono::steady_clock::now();
00325 \\ 00326
00327
00328
00329
                           std::vector<Element> res = spisok;
                           heapSort1(res, res.size());
std::cout « "\Heap ended!\n";
std::cout « "\nBinary searching started!\n";
00330
00331 \\ 00332
                           binary Search1(res, searchQuery);
std::cout « "\nBinary searching ended!\n";
auto end = std::chrono::steady_clock::now();
00333
00334
00335
                           auto\ elapsed\_ms = std::chrono::duration\_cast < std::chrono::milliseconds > (end\ -\ begin); \\ std::cout\  \  \, "The\ time:\  \  \, "\  \  elapsed\_ms.count()\  \  \, "\  \  ms\n\n"; \\ }
00336
00337
00338
00339
                      else if (choosedAlg == 3) {
00340
                           //find in multimap by key
00341
                           auto begin = std::chrono::steady clock::now();
                           findMultimap(map, searchQuery);
auto end = std::chrono::steady_clock::now();
00342
00343
                           auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin); std::cout « "The time: " « elapsed_ms.count() « " ms";
00344 \\ 00345
00346
                     std::cout « "\n\nEnd!\n\n"; std::cout « "Would you like to search again (0 - linear (already sorted), 1 - binary (already sorted), 2 - sort and
00347
00348
           binary, 3 - multimap and search by key, -2 - end)?\n";
                     std::cin » choosedAlg;
std::cout « "\nAgain.\n";
00349
00350
00351
00352
                return 0;
00353 }
```

Предметный указатель

alph Element.cpp, 15	Main.cpp, 22 heapSort1
binarySearch1 Main.cpp, 20	Main.cpp, 22 linearSearch
checkFIO	Main.cpp, 23
Main.cpp, 20	main
checkTwoStrings Main.cpp, 21	Main.cpp, 23 Main.cpp, 19
Element, 7 Element, 7, 8 get Child Amount, 8 get Fam, 8 get Im, 9 get Ot, 9 get Year, 9	binarySearch1, 20 checkFIO, 20 checkTwoStrings, 21 findMultimap, 22 heapify1, 22 heapSort1, 22 linearSearch, 23
get Year D, 9	main, 23
$\begin{array}{l} \mathrm{operator}<,10 \\ \mathrm{operator}<=,10 \\ \mathrm{operator}>,11 \end{array}$	operator< Element, 10
operator >=, 12	operator <= Element, 10
$egin{array}{l} ext{operator}=,\ 11 \ ext{setChildAmount},\ 13 \end{array}$	operator>
set Fam, 13	Element, 11
setIm , $\overset{\cdot}{1}3$	operator>= Element, 12
set Ot, 14	operator=
set Year, 14	Element, 11
set Year D, 14	C1 11 1 A
Element.cpp, 15 alph, 15	setChildAmount
Element.h, 19	Element, 13 setFam
Elonomi, 10	Element, 13
findMultimap	setIm
Main.cpp, 22	Element, 13
$\operatorname{get}\operatorname{Child}\operatorname{Amount}$	$\operatorname{set}\operatorname{Ot}$
Element, 8	Element, 14
getFam	setYear
Element, 8	Element, 14 setYearD
getIm	Element, 14
Element, 9	Dicincia, 14
getOt Element, 9	
getYear	
Element, 9	
getYearD	
Element, 9	

 ${\it heapify1}$