$\begin{array}{c} \text{MP-Frolov-Lab-1} \\ \text{1.0} \end{array}$

Создано системой Doxygen 1.9.6

1 Методы программирования. Лабора	торн	ая І	рабо	та Л	№1.	Алг	ори	ITM	ысс	рти	poi	зки				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 Kласс Element														 		 7
4.1.1 Подробное описание														 		 7
4.1.2 Конструктор(ы)														 		 7
$4.1.2.1 \; \text{Element}() \; [1/2]$														 		 8
$4.1.2.2 \; \mathrm{Element}() \; [2/2]$														 		 8
4.1.3 Методы														 		 8
$4.1.3.1~{ m get}{ m ChildAmoun}$	s() .													 		 8
$4.1.3.2 \text{ getFam}() \dots$	**															
$4.1.3.3 \text{ get Im}() \dots$														 		 9
$4.1.3.4 \operatorname{get} \operatorname{Ot}() \dots$																
$4.1.3.5 \text{ getYear}() \dots$																
4.1.3.6 getYearD()																
4.1.3.7 operator < ()																
4.1.3.8 operator <=().																
4.1.3.9 operator = ()																
4.1.3.10 operator > ().																
4.1.3.11 operator >=()																
4.1.3.12 setChildAmour																
4.1.3.13 setFam()	.,															
4.1.3.14 setIm()																
$4.1.3.15 \text{ setOt}() \dots$																
4.1.3.16 setYear()																
4.1.3.17 setYearD()																
5 Файлы																15
5.1 Файл Element.cpp																15
5.1.1 Переменные																
5.1.1.1 alph																
5.2 Element.cpp																16
5.3 Файл Element.h																
5.4 Element.h																19
5.5 Файл Main.cpp																19
5.5.1 Функции																
5.5.1 Функции																
a.a.r.r pappichoro() .													•	 	•	 20

5	.5.1.2 ł	eapif	y1() .															20
5	.5.1.3 ł	ıeapS	ort1	L()															21
5	.5.1.4 r	nain() .																21
5	.5.1.5 s	aveT	oFil	e()															23
5	.5.1.6 s	hake	rSor	t()															23
5.6 Main.cpp														 •			•		23
Предметный ука	азатель																		27

Методы программирования. Лабораторная работа №1. Алгоритмы сортировки.

1.1 Вариант 26

Массив данных генеологического фонда: ФИО, год рождения, год смерти, число детей (сравнение по полям - год рождения, ФИО, число детей). Сортировки: пузырьком, шейкер-сортировка, пирамидальная сортировка.

1.2 Выполнил

Фролов Олег, СКБ201

2	Методы	программирования.	Лабораторная	работа №1.	Алгоритмы с	ортировки.
					Converse	* D

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.

```
      См. 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.

```
Cм. определение в файле Element.cpp строка 104 00104 { 00105 return this->child_amount; 00106 }
```

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 62
00062
00063
         return this->otchestvo;
00064 }
4.1.3.5 \text{ getYear}()
int Element::getYear ( )
Функция получения года рождения.
C_{\rm M.} определение в файле \; Element.cpp строка 76 ^{00076} \; ^{\rm return\ this->year;} 00078 }
```

10 Классы

```
4.1.3.6 \text{ getYearD}()
int Element::getYearD ( )
Функция получения года смерти.
См. определение в файле Element.cpp строка 90
00090 \\ 00091
          return this->year_death;
00092 }
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 166
00166
00167
          \mathbf{if} \ (\mathtt{year} \ < \ \mathtt{elem.year}) \ \{
00168
              return 1;
          00169
00170
00171
00172
                 if (imya[0] == elem.imya[0]) {
    if (otchestvo[0] == elem.otchestvo[0]) {
        if (child_amount < elem.child_amount)
        return 1;
00173
00174
00175
00176
                        else
00177
00178
                           return 0;
00179
00180
                     else {
                        int first = alph.find(std::tolower(otchestvo[0]));
int second = alph.find(std::tolower(elem.otchestvo[0]));
00181
00182
00183
                        if (first > second)
00184
                           return 1;
00185
                        else
00186
                           return 0;
                     }
00187
00188
00189
00190
                     int first = alph.find(std::tolower(imya[0]));
00191
                     int second = alph.find(std::tolower(elem.imya[0]));
00192
                     if (first > second)
00193 \\ 00194
                     return 1; else
00195
                        return 0;
00196
                 }
00197
00198
                 \label{eq:int_first} \begin{array}{l} \text{int} \ \operatorname{first} \ = \ \operatorname{alph.find}(\operatorname{std}::\operatorname{tolower}(\operatorname{familiya}[0])); \end{array}
00199
00200
                 int\ second = alph.find(std::tolower(elem.familiya[0]));\\
00201
                 if (first > second)
00202
                     return 1;
00203
00204
                     return 0;
00205
              }
00206
          } else {
00207
00208
             return 0;
00209
```

00210 }

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 264
00265
          if (year <= elem.year) {
00266
             return 1;
00267
          else if (year > elem.year) {
00268
00269
               /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]) {
00270
00271
                     \inf (\text{otchestvo}[0] == \text{elem.otchestvo}[0]) 
00272
00273
                        if (child_amount <= elem.child_amount)</pre>
00274
                        else
00275
00276
                           return 0;
00277
                    }
00278
                     else {
                       int first = alph.find(std::tolower(otchestvo[0]));
int second = alph.find(std::tolower(elem.otchestvo[0]));
00279
00280
00281
                        _{\bf if}~({\rm first}~>\equiv~{\rm second})
00282
                           return 1;
00283
                        else
00284
                           return 0;
00285
                    }
00286
00287
00288
                    int first = alph.find(std::tolower(imya[0]));
00289
                    int second = alph.find(std::tolower(elem.imya[0]));
00290
                    if (first >= second)
00291
                       return 1;
00292
                    else
00293 \\ 00294
                       return 0;
                 }
00295
00296
00297
                 int first = alph.find(std::tolower(familiya[0]));
00298
                 int second = alph.find(std::tolower(elem.familiya[0]));
00299
                 if (first > \equiv second)
00300
                    return 1;
00301
                 else
00302
                    return 0;
00303
             }
00304
00305
          else {
00306
              return 0;
00307
          }
00308 }
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 313
          this->setFam(elem.getFam());
00314
00315
          this-\!>\!setIm(elem.getIm());
          this \hspace{-0.05cm}-\hspace{-0.05cm} > \hspace{-0.05cm} setOt(elem.getOt());
00316
          this->setYear(elem.getYear());
this->setYearD(elem.getYearD());
00317
00319
          this->setChildAmount(elem.getChildAmount());
00320
          return *this;
00321 }
```

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 118
00118
00119
          if (year > elem.year) {
00120
             return 1;
00121
           \begin{array}{c} \textbf{else if (year == elem.year) } \\ \textbf{\{} \end{array} 
00122
             //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]) {
00123
00124
00125
00126
                    if (otchestvo[0] == elem.otchestvo[0]) {
00127
                       if (child_amount > elem.child_amount)
00128
                          return 1;
                       else
00129
00130
                          return 0;
00131
00132
00133
                       int\ first = alph.find(std::tolower(otchestvo[0]));
                       int second = alph.find(std::tolower(elem.otchestvo[0]));
00134
00135
                       if (first < second)
00136
                          return 1;
00137
00138
                          return 0;
00139
                   }
00140
00141
                else {
00142
                   int\ first = alph.find(std::tolower(imya[0]));
00143
                   int second = alph.find(std::tolower(elem.imya[0]));
00144
                   if (first < second)
00145
                       return 1;
00146
                   else
00147
                       return 0;
00148
                }
00149
00150
00151
                int first = alph.find(std::tolower(familiya[0]));
00152
                int\ second = alph.find(std::tolower(elem.familiya[0]));\\
00153
                if (first < second)
00154
                    return 1;
                else
00155
00156
                   return 0;
00157
00158
          } else {
00159
             return 0;
          }
00160
00161 }
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 215
00215
00216
          if (year >= elem.year) {
00217
             return 1;
00218
00219
          else if (year < elem.year) {
00220
             //цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей
00221
             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;
00222
00223
00224
00225
00226
                       else
```

4.1 Класс Element 13

```
00227
                               return 0;
00228
00229
                        else {
                           int first = alph.find(std::tolower(otchestvo[0]));
int second = alph.find(std::tolower(elem.otchestvo[0]));
if (first <= second)
00230
00231
00232
00233
                              return 1;
00234
00235
                               return 0;
00236
                       }
00237
00238
                       int first = alph.find(std::tolower(imya[0]));
int second = alph.find(std::tolower(elem.imya[0]));
00239
00240
00241
                       if (first <= second)
00242 \\ 00243
                           return 1;
                       else
00244
                           return 0;
00245
                   }
00246
00247
00248 \\ 00249
                    \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}{ll} & \text{int second} = alph.find(std::tolower(elem.familiya[0]));} \\ & \text{if (first } <= second) \end{array}
00250
00251
                       return 1;
00252
00253
                       return 0;
00254
               }
00255
00256
            else {
00257
               return 0;
            }
00258
00259~\}
4.1.3.12 set ChildAmount()
void Element::setChildAmount (
                     int newAmount )
Функция установки количества детей.
См. определение в файле Element.cpp строка 111
00111
           this->child_amount = newAmount;
00112
00113 }
4.1.3.13 \operatorname{setFam}()
void Element::setFam (
                     std::string newFam )
Функция установки фамилии.
См. определение в файле Element.cpp строка 40
00040
```

00041 00042 } this->familiya = new Fam;

14 Классы

```
4.1.3.14 \text{ set Im}()
void Element::setIm (
               std::string newIm )
Функция установки имени.
См. определение в файле _{\{} Element.cpp строка 55
00056
00057 }
       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 69
00069
00070
       this-> otchestvo = newOt;
00071 }
4.1.3.16 setYear()
void Element::setYear (
               int newYear)
Функция установки года рождения.
См. определение в файле Element.cpp строка 83
        this->year = new Year;
00084
00085 }
4.1.3.17 setYearD()
void Element::setYearD (
               int newYearD )
Функция установки года смерти.
_{00097}^{Cm.} определение в файле Element.cpp строка 97 _{00098}^{00098} this->year death = newYearD;
       this->year_death = newYearD;
00099 }
```

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
00055 void Element::setIm(std::string newIm) {
00056
         this->imya= newIm;
00057 }
00058
00062 std::string Element::getOt() {
         return this->otchestvo;
00063
00064 }
00065
00069 void Element::setOt(std::string newOt) {
00070
         this->otchestvo = newOt;
00071 }
00072
00076 int Element::getYear() {
00077
         return this->year;
00078 }
00079
00083 void Element::setYear(int newYear) {
00084
         this->year = new Year;
00085 }
00086
00090 int Element::getYearD() {
00091
         return this->year_death;
00092 }
00093
00097 void Element::setYearD(int newYearD) {
         this->year_death = newYearD;
00099 }
00100
00104 int Element::getChildAmount() {
00105
         {\color{red}\mathbf{return}}\ this-{\color{blue}>} child\_{\color{blue}\mathbf{amount}};
00106 }
00107
00111\ void\ Element::setChildAmount(int\ newAmount)\ \{
00112
         this\text{-}{>}child\_amount = newAmount;
00113 }
00114
00118 bool Element::operator> (const Element& elem) const {
         if (year > elem.year) {
00120
            return 1;
00121
        00122
00123
00124
00126
                  if (otchestvo[0] == elem.otchestvo[0]) {
00127
                     if (child_amount > elem.child_amount)
00128
                        return 1;
                     else
00129
00130
                       return 0:
```

5.2 Element.cpp 17

```
00131
00132
                         int first = alph.find(std::tolower(otchestvo[0]));
00133
00134
                         int\ second = alph.find(std::tolower(elem.otchestvo[0]));
00135
                         if (first < second)
00136
                            return 1;
00137
                         else
00138
                            return 0;
00139
                     }
00140
00141
                  else {
                     int first = alph.find(std::tolower(imya[0]));
int second = alph.find(std::tolower(elem.imya[0]));
00142
00143
00144
                     if (first < second)
00145
                         return 1;
                     else
00146
00147
                         return 0:
00148
                  }
00149
00150
00151
                  int first = alph.find(std::tolower(familiya[0]));
00152
                  int\ second = alph.find(std::tolower(elem.familiya[0]));\\
00153
                  if (first < second)
00154
                     return 1;
00155
                  else
00156
                     return 0;
00157
00158
          } else {
00159
              return 0;
          }
00160
00161 }
00162
00166 bool Element::operator< (const Element& elem) const {
00167
          if (year < elem.year) {</pre>
00168
              return 1;
00169
          else if (year > elem.year) {
//цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей
00170
00171
              if (familiya[0] == elem.familiya[0]) {
    if (imya[0] == elem.imya[0]) {
00172
00173
                     if (otchestvo[0] == elem.otchestvo[0]) {
   if (child_amount < elem.child_amount)</pre>
00174
00175
00176
                            return 1;
                         else
00177
00178
                            return 0;
00179
00180
                      else {
                         int first = alph.find(std::tolower(otchestvo[0]));
int second = alph.find(std::tolower(elem.otchestvo[0]));
00181
00182
00183
                         if (first > second)
00184
                            return 1;
00185
                         else
00186
                            return 0;
00187
                     }
00188
00189
00190
                     int first = alph.find(std::tolower(imya[0]));
00191
                     int second = alph.find(std::tolower(elem.imya[0]));
00192
                     if (first > second)
00193
                         return 1;
00194
                     else
00195
                         return 0;
00196
                  }
00197
00198
                  int first = alph.find(std::tolower(familiya[0]));
00199
00200
                  int second = alph.find(std::tolower(elem.familiya[0]));
00201
                  if (first > second)
00202
                     return 1;
00203
                  else
00204
                     return 0;
00205
              }
00206
          else {
00207
00208
              return 0;
00209
00210 }
00211
00215 bool Element::operator>= (const Element & elem) const \{
00216
          if (year >= elem.year) {
00217
              return 1;
00218
00219
          else if (year < elem.year) {
00220
                /цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей
              // (familiya[0] == elem.familiya[0]) {
    if (imya[0] == elem.imya[0]) {
        if (otchestvo[0] == elem.otchestvo[0]) {
00221
00222
00223
```

```
if (child amount >= elem.child amount)
00224
00225
                          return 1;
                       else
00226
00227
                          return 0;
00228
                   else {
  int first = alph.find(std::tolower(otchestvo[0]));
  int second = alph.find(std::tolower(elem.otchestvo[0]));
}
00229
00230
00231
00232
                      return 1;
                       if (first <= second)
00233
00234
00235
                          return 0:
00236
                   }
00237
00238
                else {
00239
                   int\ first = alph.find(std::tolower(imya[0]));\\
00240
                   int second = alph.find(std::tolower(elem.imya[0]));
00241
                   if (first <= second)
00242
                      return 1;
00243
                   else
00244
                      return 0;
00245
                }
00246
00247
00248
                int first = alph.find(std::tolower(familiya[0]));
00249
                int second = alph.find(std::tolower(elem.familiya[0]));
00250
                if (first \le second)
00251
                   return 1;
                else
00252
00253
                   return 0:
00254
            }
00255
         }
00256
         else {
00257
             return 0;
         }
00258
00259 }
00260
00264 bool Element::operator<= (const Element& elem) const {
00265
         if (year <= elem.year) {
00266
             return 1;
00267
00268
         else if (year > elem.year) {
00269
              /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]) {
00270
00271
00272
                   if (otchestvo[0] == elem.otchestvo[0]) {
                       if (child_amount <= elem.child_amount)
00273
00274
                          return 1;
00275
                       else
00276
                          return 0;
00277
                   }
00278
                    else {
00279
                      int\ first = alph.find(std::tolower(otchestvo[0]));
00280
                       int\ second = \ alph.find(std::tolower(elem.otchestvo[0]));
00281
                       if (first >= second)
00282
                          return 1;
00283
00284
                          return 0;
00285
                   }
00286
00287
                else {
00288
                   int first = alph.find(std::tolower(imya[0]));
00289
                   int second = alph.find(std::tolower(elem.imya[0]));
00290
                   if (first >= second)
00291
                       return 1;
                   else
00292
00293
                      return 0;
00294
                }
00295
00296
00297
                int first = alph.find(std::tolower(familiya[0]));
00298
                int second = alph.find(std::tolower(elem.familiya[0]));
00299
                if (first >= second)
00300
                   return 1;
00301
                else
00302
                   return 0;
00303
            }
00304
         else {
00305
00306
             return 0;
         }
00307
00308 }
00309
00313 Element& Element::operator= (Element& elem) {
00314
         this-\!>\!setFam(elem.getFam());
00315
         this->setIm(elem.getIm())
00316
         this->setOt(elem.getOt());
```

5.3 Файл Element.h 19

```
00317 this->setYear(elem.getYear());
00318 this->setYearD(elem.getYearD());
00319 this->setChildAmount(elem.getChildAmount());
00320 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>
00005 class Element {
00006 private:
             ivate:
    std::string familiya = "";
    std::string imya = "";
    std::string otchestvo = "";
    int year = 1900;
    int year_death = 1920;
    int child_amount = 0;
    int sex = 0;
    intlended
00007
00008
00009
00010
00011
00012
00013
00014~\mathrm{public}\colon
00015
              Element();
00016
              Element(std::string fam, std::string im, std::string ot, int y, int yd, int ca);
00017
              std::string getFam();
00018
              void setFam(std::string newFam);
00019 \\ 00020
              std::string getIm();
              void setIm(std::string newIm);
00021
              std::string getOt();
void setOt(std::string newOt);
00022
00023
              int getYear();
00024
              void setYear(int newYear);
              int getYearD();
void setYearD(int newYearD);
int getChildAmount();
void setChildAmount(int newAmount);
00025
00026 \\ 00027
00028
              bool operator > (const Element& elem) const;
bool operator < (const Element& elem) const;
00029
00030
              bool operator>= (const Element& elem) const;
bool operator<= (const Element& elem) const;
00031
00032
00033
              Element& operator= (Element& elem);
00034 };
```

5.5 Файл Main.cpp

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

• void saveToFile (std::vector< Element > spisok, std::string filename)

Функции

```
• void shakerSort (std::vector< Element > &spisok)
     • void heapify1 (std::vector< Element > &spisok, int N, int i)
     • void heapSort1 (std::vector< Element > &spisok, int N)
     • void bubbleSort (std::vector< Element > &spisok)
     • int main ()
5.5.1
          Функции
5.5.1.1 bubbleSort()
void bubbleSort (
                 std::vector< Element > & spisok)
Функция для пузырьковой сортировки контейнера из объектов класса Element (стр. 7).
_{00104}^{\rm CM.} определение в файле _{00104}^{\rm Main.cpp} строка _{104}^{\rm CM.}
00105
          //сортировка по году рождения, потом по ФИО, потом по кол-ву детей for (int i=0; i< spisok.size(); ++i) { for (int j= spisok.size() - 1; j>i; --j) {
00106
00107
00108
                (ms = spisoksink(); if (spisok[j - 1] > spisok[j]) {
    Element temp = spisok[j - 1];
    spisok[j - 1] = spisok[j];
    spisok[j] = temp;
00109
00110
00111
00112
00113
00114
00115
00116 }
5.5.1.2 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 68
00068
00069
           int largest = i;
int left = 2 * i + 1;
int right = 2 * i + 2;
00070
00071
00072
           if (left<N && spisok[left]>spisok[largest])
00073
               largest = left;
           if (right<N && spisok[right]>spisok[largest])
00074
00075
               largest = right;
           if (largest != i) {
00076
00077
               Element temp:
              temp = spisok[i];
spisok[i] = spisok[largest];
spisok[largest] = temp;
00078
00079
00080
00081
               heapify1(spisok,\ N,\ largest);
00082
00083 }
```

5.5 Файл Main.cpp 21

5.5.1.3 heapSort1()

```
void heapSort1 (  \label{eq:std:vector} {\rm std::vector} < \ {\rm Element} \, > \& \ {\rm spisok}, \\ {\rm int} \ {\rm N} \ )
```

Функция для пирамидальной сортировки контейнера из объектов класса Element (стр. 7).

```
См. определение в файле Маіп.срр строка 88
00089
          for (int i = N / 2 - 1; i >= 0; --i) {
             heapify1(spisok, N, i);
00090
00091
00092
          for (int i = N - 1; i >= 0; --i) {
00093
             Element temp
             temp = spisok[0];
spisok[0] = spisok[i];
spisok[i] = temp;
00094
00095
00096
00097
             heapify1(spisok,\ i,\ 0);
00098
00099 }
```

5.5.1.4 main()

int main ()

Функция main для основной работы программы.

```
См. определение в файле Маіп.срр строка 122
 00122
 00123
00124
                         int\ file\_number\,=\,0;
00125
                         std::vector<Element> spisok;
                         int\ choosed Alg = 0;\\
00126
 00127
                         std::cout « "\nChoose sorting algorithm (0 - puzir, 1 - shaker, 2 - heap): \n";
                         std::cin » choosedAlg;
while (choosedAlg < 0 || choosedAlg >= 3 || std::cin.fail()) {
 00128
00129
 00130
                                  if (std::cin.fail()) {
 00131
                                          std::cin.clear():
                                          std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
std::cout « "Your input is not a number!" « '\n';
 00132
 00133
 00134
                                          choosedAlg = -1;
 00135
00136
                                          std::cout « "Your input must be in interval 0-7!" « '\n';
00137
                                          choosedAlg = 0;
00138
 00139
 00140
                                  std::cout « "Input again!" « '\n';
 00141
                                  std::cin » choosedAlg;
 00142
                          std::cout « "Choose file(0-7)" « '\n';
00143
                         std::cin » file_number;
while (file_number < 0 || file_number >= 8 || std::cin.fail()) {
00144
00145
 00146
                                 if (std::cin.fail()) {
 00147
                                           \begin{array}{l} \texttt{std::cin.ignore}(\texttt{std::numeric\_limits} < \texttt{std::streamsize} > :: \texttt{max}(), \ ' \backslash n'); \\ \texttt{std::cout} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ \ _{}^{*} \ 
00148
00149 \\ 00150
                                          file\_number = -1;
 00151
 00152
 00153
                                          std::cout « "Your input must be in interval 0-7!" « '\n';
 00154
                                          file\_number = 0;
00155
00156
                                  std::cout « "Input again!" « '\n';
00157
                                 std::cin » file_number;
 00158
                          setlocale(LC_ALL, "rus");
 00159
                         std::string line = ""; std::string filename = "C:\\Users\\fov-2\\source\\repos\\data_";
 00160
00161 \\ 00162
                         filename.append(std::to_string(file_number));
filename.append(".txt");
00163
00164
                         std::ifstream in(filename);
00165
                         if (in.is_open()) {
```

```
while (getline(in, line)) {
  std::string word = "";
00167
00168
                      int flag = 0;
00169
                      int amount = 0;
                     Element human;

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

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

        flag = 2;
00170
00171
00172
00173
                          else if (line[i] == ')')
00174
                          flag = 0;
else if (line[i] == ' '||line[i]=='\n') {
00175
00176
                              if (amount == 0) {
00177
00178
                                  human.setFam(word);
00179
                               \begin{array}{l} \\ \text{else if (amount} == 1) \end{array} 
00180
00181
                                  human.setIm(word);
00182
                              else if (amount == 2)
                                  human.setOt(word);
00183
                              else if (amount == 3)
00184
00185
                                  human.setYear(std::stoi(word));
                              else if (amount =\stackrel{\cdot}{=} 4)
00186
                              human.setYearD(std::stoi(word));
word = "";
00187
00188
00189
                              ++amount;
00190
                          else if (flag == 0) {
                              char symbol = line[i];
if (symbol != ' ') {
00192
00193
00194
                                  word.push_back(symbol);
00195
00196
                          }
00197
00198
                          if (amount == 5) \ \{
00199
                              amount = 0;
00200 \\ 00201
                              human.setChildAmount(line[line.size()-1]-'0');
                              spisok.push_back(human);
00202
                         }
00203
                     }
00204
                 }
00205
             in.close();
00206
             00207
00208
00209
00210
00211
                      auto begin = std::chrono::steady_clock::now();
00212
00213
                      std::vector<Element> res = spisok;
00214
                      bubbleSort(res);
00215
                      auto end = std::chrono::steady clock::now();
                     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";
saveToFile(res, "C:\\Users\\fov-2\\source\\repos\\out1.txt");
std::cout « "\nPuzir ended!\n\n";
00216
00217
00218
00219
00220
                 felse if (choosedAlg == 1) {
   std::cout « "\n\nShaker started!\n";
   auto begin = std::chrono::steady_clock::now();
   std::vector<Element> res = spisok;
00221
00222
00223
00224
00225
                      shakerSort(res);
00226
00227
                      auto end = std::chrono::steady_clock::now();
                     auto elia = 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";
saveToFile(res, "C:\\Users\\fov-2\\source\\repos\\out11.txt");
std::cout « "\nShaker ended!\n\n";
00228
00229
00230
00231
00232
00233
                 else if (choosedAlg == 2) {
                     std::cout « "\n\nHeap started!\n";
auto begin = std::chrono::steady_clock::now();
00234
00235
00236
                      std::vector < Element > res = spisok;
00237
                      heapSort1(res, res.size());
                     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";
saveToFile(res, "C:\\Users\\fov-2\\source\\repos\\out111.txt");
00238
00239
00240
00241
00242
                      std::cout « "\nHeap ended!\n\n";
00243
                 00244
00245
00246
                 std::cin » choosedAlg;
00248
             return 0;
00249 }
```

5.6 Main.cpp 23

5.5.1.5 saveToFile()

Функция для сохранения контейнера, который содержит объекты класса Element (стр. 7), в текстовый файл.

5.5.1.6 shakerSort()

```
void shaker
Sort ( {\rm std::vector} < {\rm \ Element} > \& {\rm \ spisok} \ )
```

Функция для шейкерной сортировки контейнера, который состоит из объектов класса Element (стр. 7).

```
См. определение в файле Маіп.срр строка 37
00037
                   int \ j, \ k = \ spisok.size() \ \text{-} \ 1;
00038
00039
                   int lb = 1, ub = spisok.size() - 1;
Element temp;
00040
                   \begin{split} & Element \; temp; \\ & \text{while} \; (lb < ub) \; \{ \\ & \text{for} \; (j = ub; \; j > 0; --j) \; \{ \\ & \text{if} \; (spisok[j-1] > spisok[j]) \; \{ \\ & \text{temp} = spisok[j-1]; \\ & \text{spisok[j-1]} = spisok[j]; \\ & \text{spisok[j]} = temp; \\ & \text{l.} \quad \dots \quad \dots \quad \dots \end{split} 
00041
00042
00043
00044
00045
00046
00047
                                      k=j;\\
00048
                               }
00049
                         lb = k + 1;
00050
                        for (j = 1; j \le ub; ++j) {
    if (spisok[j - 1] > spisok[j]) {
        temp = spisok[j - 1];
        spisok[j - 1] = spisok[j];
00051
00052
00053
00054
00055
                                      spisok[j] = temp;
00056
                                      \hat{\mathbf{k}} = \mathbf{j};
00057
00058
                         ub = k - 1;
00059
00060
                   }
00061 }
```

5.6 Main.cpp

```
CM. Документацию.
00001 #include <iostream>
00002 #include <fstream>
00003 #include "Element.h"
00004 #include <vector>
00005 #include <chrono>
00005
```

```
00022 void saveToFile(std::vector<Element> spisok, std::string filename) {
            {\bf std::} {\bf ofstream\ out};\\
00023
00024
            out.open(filename);
         00025
00026
00027
00028
00029
00030
            out.close();
00031 }
00032
00033
         //шейкерная --
00037
        void shakerSort(std::vector<Element> &spisok) {
00038
            int j, k = spisok.size() - 1;
00039
            int\ lb\,=\,1,\ ub\,=\,spisok.size(\,)\,\,\hbox{-}\,\,1;
00040
            Element temp;
           Element temp; while (lb < ub) { for (j = ub; j > 0; --j) { if (spisok[j - 1] > spisok[j]) { temp = spisok[j - 1]; spisok[j - 1] = spisok[j]; spisok[j] = temp; } 
00041
00042
00043
00044
00045
00046
00047
                        k = j;
00048
                    }
00049
00050
                lb = k + 1;
                 \begin{cases} \text{for } (j = 1; j <= \text{ub}; ++j) \; \{ \\ \text{if } (\text{spisok}[j-1] > \text{spisok}[j]) \; \{ \\ \text{temp} = \text{spisok}[j-1]; \\ \text{spisok}[j-1] = \text{spisok}[j]; \\ \text{spisok}[j] = \text{temp}; \end{cases} 
00051
00052
00053
00054
00055
00056
                        k = j;
00057
00058
00059
                 ub = k - 1:
00060
            }
00061 }
00062
00063
00064
         //пирамидальная, сначала строим кучу (дерево), потом сортируем
00068 void heapify1(std::vector < Element > & spisok, int N, int i) {
            int largest = i;
int left = 2 * i + 1;
int right = 2 * i + 2;
00069
00070
00071
00072
            if (left<N && spisok[left]>spisok[largest])
00073
                largest = left;
            if (right < N && spisok[right] > spisok[largest])
largest = right;
00074
00075
            if (largest != i) {
00076
00077
                 Element temp:
                temp = spisok[i];
spisok[i] = spisok[largest];
00078
00079
00080
                 spisok[largest] = temp;
00081
                 heapify1(spisok, N, largest);
00082
            }
00083 }
00084
00088 void heapSort1(std::vector<Element> &spisok, int N) {
            \begin{array}{l} \text{for (int } i=N \ / \ 2 \ -1; \ i>=0; --i) \ \{ \\ \text{heapify1(spisok, N, i);} \end{array}
00089
00090
00091
00092
            for (int i = N - 1; i >= 0; --i) {
00093
                Element temp;
                temp = spisok[0];

spisok[0] = spisok[i];

spisok[i] = temp;

heapify1(spisok, i, 0);
00094
00095
00096
00097
00098
00099 }
00100
00104 void bubbleSort(std::vector<Element>& spisok) {
00105
00106
             //сортировка по году рождения, потом по ФИО, потом по кол-ву детей
            for (int i = 0; i < spisok.size(); ++i) {
for (int j = spisok.size() - 1; j > i; --j) {
00107
00108
                    if (spisok[j - 1] > spisok[j]) {
    Element temp = spisok[j - 1];
    spisok[j - 1] = spisok[j];
    spisok[j] = temp;
00109
00110
00111
00112
00113
00114
                }
00115
            }
00116 }
00117
00118
00122 int main() {
```

5.6 Main.cpp 25

```
00123
00124
           int\ file\_number = 0;
00125
           std::vector<Element> spisok;
           00126
00127
           \begin{array}{l} \text{std::cin} \  \, \text{schosedAlg}; \\ \text{while } (\text{choosedAlg} < 0 \mid\mid \text{choosedAlg} >= 3 \mid\mid \text{std::cin.fail())} \mid \\ \text{if } (\text{std::cin.fail())} \mid \\ \end{array}
00128
00129
00130
00131
                   std::cin.clear();
                   std::cin.ignore(std::numeric\_limits < std::streamsize > ::max(), '\n'); std::cout « "Your input is not a number!" « '\n'; choosedAlg = -1;
00132
00133
00134
00135
00136
00137
                   std::cout « "Your input must be in interval 0-7!" « '\n';
00138
                   choosed Alg\,=\,0;
00139
               std::cout « "Input again!" « '\n'; std::cin » choosedAlg;
00140
00141
00142
00143
           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()) {
00144
00145
00146
00147
                   std::cin.clear();
00148
                   std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
std::cout « "Your input is not a number!" « '\n';
00149
00150
                   file number = -1;
00151
00152
               else {
                   std::cout « "Your input must be in interval 0-7!" « '\n';
00153
00154
                   file number = 0;
00155
               std::cout « "Input again!" « '\n';
00156
               std::cin » file_number;
00157
00158
           setlocale(LC_ALL, "rus");
std::string line = "";
00159
00160
           std::string filename = "C:\\Users\\fov-2\\source\\repos\\data ";
00161
           filename.append(std::to_string(file_number));
filename.append(".txt");
00162
00163 \\ 00164
           std::ifstream in(filename);
           00165
00166
00167
                   std::string word =
00168
                   int flag = 0;
00169
                   int amount = 0;
00170 \\ 00171
                   Element human;
                   for (int i = 0; i < line.size(); ++i) {
    if (line[i] == '(')
        flag = 2;
    else if (line[i] == ')')
00172
00173
00174
                      flag = 0;
else if (line[i] == ' '||line[i]=='\n') {
    if (amount == 0) {
00175
00176
00177
00178
                              human.setFam(word);
00179
00180
                           \overset{\circ}{\text{else}} if (\text{amount} == 1)
00181
                              human.setIm(word);
00182
                           else if (amount == 2)
                              human.setOt(word);
00183
00184
                          else if (amount == 3)
00185
                              human.setYear(std::stoi(word));
00186
                           else if (amount = \stackrel{\frown}{=} 4)
00187
                              human.setYearD(std::stoi(word));
00188
                           word = ""
00189
                           ++amount;
00190
00191
                      else if (flag == 0) {
                          char symbol = line[i];
if (symbol != ' ') {
00192
00193
                              word.push_back(symbol);
00194
00195
                          }
00196
                      }
00197
00198
                       if (amount == 5) {
00199
00200
                           human.setChildAmount(line[line.size()-1]-'0');
00201
                           spisok.push_back(human);
00202
                      }
00203
                   }
00204
               }
00205
00206
           in.close();
           00207
00208
00209
```

```
if (choosedAlg == 0) {
   std::cout « "\n\nPuzir started!\n";
   auto begin = std::chrono::steady_clock::now();
00210
00211
00212
                           std::vector < Element > res = spisok;
00213
                          bubbleSort(res);
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";
saveToFile(res, "C:\\Users\\fov-2\\source\\repos\\out1.txt");
std::cout « "\nPuzir ended!\n\n";
00214
00215
00216
00217
00218
00219
00220
                     std::cout « "\n\nShaker started!\n";
auto begin = std::chrono::steady_clock::now();
00221
00222
00223
00224
                           std::vector < Element > res = spisok;
00225 \\ 00226
                           shakerSort(res);
00227
                           auto end = std::chrono::steady_clock::now();
                          auto elia = 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";
saveToFile(res, "C:\\Users\\fov-2\\source\\repos\\out11.txt");
std::cout « "\nShaker ended!\n\n";
00228
00229
00230
00231
00232
00233
                     else if (choosedAlg == 2) {
    std::cout « "\n\nHeap started!\n";
    auto begin = std::chrono::steady_clock::now();
00234
00235
00236
                           std::vector < Element > res = spisok;
00237
                           heapSort1(res, res.size());
                          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";
saveToFile(res, "C:\\Users\\fov-2\\\source\\repos\\out111.txt");
std::cout « "\nHeap ended!\n\n";
00238
00239
00240
00241
00242
00243
                     00244 \\ 00245
00246
                     std::cin » choosedAlg;
00247
00248
                return 0;
00249 }
```

Предметный указатель

alph	bubbleSort, 20
Element.cpp, 15	heapify1, 20
11/	heapSort1, 20
bubbleSort	main, 21
Main.cpp, 20	saveToFile, 22
	shakerSort, 23
Element, 7	511611615515, 25
Element, 7, 8	operator<
$\operatorname{get}\operatorname{Child}\operatorname{Amount},8$	Element, 10
getFam, 8	operator<=
m getIm,~9	Element, 10
$\operatorname{get}\operatorname{Ot},9$	operator>
get Year, 9	Element, 11
getYearD, 9	operator>=
operator $<$, 10	Element, 12
operator <=, 10	operator=
operator>, 11	Element, 11
operator >=, 12	,
operator=, 11	saveToFile
$\operatorname{set}\operatorname{ChildAmount},13$	Main.cpp, 22
setFam, 13	${\rm setChildAmount}$
setIm, 13	Element, 13
$\operatorname{set}\operatorname{Ot},14$	setFam
set Year, 14	Element, 13
set Year D, 14	setIm
Element.cpp, 15	Element, 13
alph, 15	$\operatorname{set}\operatorname{Ot}$
Element.h, 19	Element, 14
	$\operatorname{setYear}$
$\operatorname{get}\operatorname{Child}\operatorname{Amount}$	Element, 14
Element, 8	$\operatorname{setYearD}$
getFam	Element, 14
Element, 8	${ m shaker Sort}$
m get Im	Main.cpp, 23
Element, 9	
$\operatorname{get}\operatorname{Ot}$	
Element, 9	
getYear	
Element, 9	
getYearD	
Element, 9	
hoonify1	
heapify1	
Main.cpp, 20	
heapSort1	
Main.cpp, 20	
main	
Main.cpp, 21	
Main.cpp, 19	
mannopp, 10	