MP-Frolov-Lab-1

1.0

Создано системой Doxygen 1.9.6

Глава 1

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

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

Глава 3

Список файлов

3.1 Файлы

Полный список файлов.

Глава 4

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)

4.1.2.1 Element() [1/2]

Element::Element ( )

Стандартный конструктор.

См. определение в файле Element.cpp строка 12

```
00012                {
00013     this->familiya = "Fam";
00014     this->imya = "Im";
00015     this->otchestvo = "Ot";
00016 }
```

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                                                    {
00022     this->familiya = fam;
00023     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.

См. определение в файле Element.cpp строка 104

```
00104                    {
00105     return this->child_amount;
00106 }
```

4.1.3.2 getFam()

std::string Element::getFam ( )

Функция получения фамилии.

См. определение в файле Element.cpp строка 33

```
00033                       {
00034     return this->familiya;
00035 }
```

4.1.3.3 getIm()

std::string Element::getIm ( )

Функция получения имени.

См. определение в файле Element.cpp строка 47

```
00047                       {
00048     return this->imya;
00049 }
```

4.1.3.4 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 getYear()

int Element::getYear ( )

Функция получения года рождения.

См. определение в файле Element.cpp строка 76

```
00076                   {
00077     return this->year;
00078 }
```

4.1.3.6 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     if (year < elem.year) {
00168         return 1;
00169     }
00170     else if (year > elem.year) {
00171         //цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей
00172         if (familiya[0] == elem.familiya[0]) {
00173             if (imya[0] == elem.imya[0]) {
00174                 if (otchestvo[0] == elem.otchestvo[0]) {
00175                     if (child_amount < elem.child_amount)
00176                         return 1;
00177                     else
00178                         return 0;
00179                 }
00180                 else {
00181                     int first = alph.find(std::tolower(otchestvo[0]));
00182                     int second = alph.find(std::tolower(elem.otchestvo[0]));
00183                     if (first > second)
00184                         return 1;
00185                     else
00186                         return 0;
00187                 }
00188             }
00189             else {
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         else {
00199             int first = alph.find(std::tolower(familiya[0]));
00200             int second = alph.find(std::tolower(elem.familiya[0]));
00201             if (first > second)
00202                 return 1;
00203             else
00204                 return 0;
00205         }
00206     }
00207     else {
00208         return 0;
00209     }
00210 }
```

4.1.3.8 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 264

```
00264                                       {
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
00270         if (familiya[0] == elem.familiya[0]) {
00271             if (imya[0] == elem.imya[0]) {
00272                 if (otchestvo[0] == elem.otchestvo[0]) {
00273                     if (child_amount <= elem.child_amount)
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                     else
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;
00292                 else
00293                     return 0;
00294             }
00295         }
00296         else {
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     }
00305     else {
00306         return 0;
00307     }
00308 }
```

4.1.3.9 operator=()

```
Element & Element::operator= (
            Element & elem )
```

Перегрузка оператора = для сравнения двух объектов класса Element (стр. 7).

См. определение в файле Element.cpp строка 313

```
00313                                     {
00314     this->setFam(elem.getFam());
00315     this->setIm(elem.getIm());
00316     this->setOt(elem.getOt());
00317     this->setYear(elem.getYear());
00318     this->setYearD(elem.getYearD());
00319     this->setChildAmount(elem.getChildAmount());
00320     return *this;
00321 }
```

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     }
00122     else if (year == elem.year) {
00123         //цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей
00124         if (familiya[0] == elem.familiya[0]) {
00125             if (imya[0] == elem.imya[0]) {
00126                 if (otchestvo[0] == elem.otchestvo[0]) {
00127                     if (child_amount > elem.child_amount)
00128                         return 1;
00129                     else
00130                         return 0;
00131                 }
00132                 else {
00133                     int first = alph.find(std::tolower(otchestvo[0]));
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 {
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         else {
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 }
```

4.1.3.11 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]) {
00222             if (imya[0] == elem.imya[0]) {
00223                 if (otchestvo[0] == elem.otchestvo[0]) {
00224                     if (child_amount >= elem.child_amount)
00225                         return 1;
00226                     else
```

```
00227                     return 0;
00228                 }
00229                 else {
00230                     int first = alph.find(std::tolower(otchestvo[0]));
00231                     int second = alph.find(std::tolower(elem.otchestvo[0]));
00232                     if (first <= second)
00233                         return 1;
00234                     else
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         else {
00248             int first = alph.find(std::tolower(familiya[0]));
00249             int second = alph.find(std::tolower(elem.familiya[0]));
00250             if (first <= second)
00251                 return 1;
00252             else
00253                 return 0;
00254         }
00255     }
00256     else {
00257         return 0;
00258     }
00259 }
```

4.1.3.12 setChildAmount()

```
void Element::setChildAmount (
                int newAmount )
```

Функция установки количества детей.

См. определение в файле Element.cpp строка 111

```
00111                                   {
00112     this->child_amount = newAmount;
00113 }
```

4.1.3.13 setFam()

```
void Element::setFam (
                std::string newFam )
```

Функция установки фамилии.

См. определение в файле Element.cpp строка 40

```
00040                                  {
00041     this->familiya = newFam;
00042 }
```

4.1.3.14 setIm()

```
void Element::setIm (
                std::string newIm )
```

Функция установки имени.

См. определение в файле Element.cpp строка 55
```
00055                                {
00056     this->imya= newIm;
00057 }
```

4.1.3.15 setOt()

```
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
```
00083                              {
00084     this->year = newYear;
00085 }
```

4.1.3.17 setYearD()

```
void Element::setYearD (
                int newYearD )
```

Функция установки года смерти.

См. определение в файле Element.cpp строка 97
```
00097                               {
00098     this->year_death = newYearD;
00099 }
```

Объявления и описания членов классов находятся в файлах:

- Element.h
- Element.cpp

Глава 5

Файлы

5.1 Файл Element.cpp

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

Переменные

- std::string alph = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя"

5.1.1 Переменные

5.1.1.1 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>
00003
00007 std::string alph = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";
00008
00012 Element::Element() {
00013     this->familiya = "Fam";
00014     this->imya = "Im";
00015     this->otchestvo = "Ot";
00016 }
00017
00021 Element::Element(std::string fam, std::string im, std::string ot, int y, int yd, int ca) {
00022     this->familiya = fam;
00023     this->imya = im;
00024     this->otchestvo = ot;
00025     this->year = y;
00026     this->year_death = yd;
00027     this->child_amount = ca;
00028 }
00029
00033 std::string Element::getFam() {
00034     return this->familiya;
00035 }
00036
00040 void Element::setFam(std::string newFam) {
00041     this->familiya = newFam;
00042 }
00043
00047 std::string Element::getIm() {
00048     return this->imya;
00049 }
00050
00051
00055 void Element::setIm(std::string newIm) {
00056     this->imya= newIm;
00057 }
00058
00062 std::string Element::getOt() {
00063     return this->otchestvo;
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 = newYear;
00085 }
00086
00090 int Element::getYearD() {
00091     return this->year_death;
00092 }
00093
00097 void Element::setYearD(int newYearD) {
00098     this->year_death = newYearD;
00099 }
00100
00104 int Element::getChildAmount() {
00105     return this->child_amount;
00106 }
00107
00111 void Element::setChildAmount(int newAmount) {
00112     this->child_amount = newAmount;
00113 }
00114
00118 bool Element::operator> (const Element& elem) const {
00119     if (year > elem.year) {
00120         return 1;
00121     }
00122     else if (year == elem.year) {
00123         //цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей
00124         if (familiya[0] == elem.familiya[0]) {
00125             if (imya[0] == elem.imya[0]) {
00126                 if (otchestvo[0] == elem.otchestvo[0]) {
00127                     if (child_amount > elem.child_amount)
00128                         return 1;
00129                     else
00130                         return 0;
```

```
00131                 }
00132                 else {
00133                     int first = alph.find(std::tolower(otchestvo[0]));
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 {
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         else {
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) {
00168         return 1;
00169     }
00170     else if (year > elem.year) {
00171         //цикл для проверки первенства алфавита, если равные ФИО, то далее проверка по кол-ву детей
00172         if (familiya[0] == elem.familiya[0]) {
00173             if (imya[0] == elem.imya[0]) {
00174                 if (otchestvo[0] == elem.otchestvo[0]) {
00175                     if (child_amount < elem.child_amount)
00176                         return 1;
00177                     else
00178                         return 0;
00179                 }
00180                 else {
00181                     int first = alph.find(std::tolower(otchestvo[0]));
00182                     int second = alph.find(std::tolower(elem.otchestvo[0]));
00183                     if (first > second)
00184                         return 1;
00185                     else
00186                         return 0;
00187                 }
00188             }
00189             else {
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         else {
00199             int first = alph.find(std::tolower(familiya[0]));
00200             int second = alph.find(std::tolower(elem.familiya[0]));
00201             if (first > second)
00202                 return 1;
00203             else
00204                 return 0;
00205         }
00206     }
00207     else {
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
00221         if (familiya[0] == elem.familiya[0]) {
00222             if (imya[0] == elem.imya[0]) {
00223                 if (otchestvo[0] == elem.otchestvo[0]) {
```

```
00224                 if (child_amount >= elem.child_amount)
00225                     return 1;
00226                 else
00227                     return 0;
00228             }
00229             else {
00230                 int first = alph.find(std::tolower(otchestvo[0]));
00231                 int second = alph.find(std::tolower(elem.otchestvo[0]));
00232                 if (first <= second)
00233                     return 1;
00234                 else
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     else {
00248         int first = alph.find(std::tolower(familiya[0]));
00249         int second = alph.find(std::tolower(elem.familiya[0]));
00250         if (first <= second)
00251             return 1;
00252         else
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
00270         if (familiya[0] == elem.familiya[0]) {
00271             if (imya[0] == elem.imya[0]) {
00272                 if (otchestvo[0] == elem.otchestvo[0]) {
00273                     if (child_amount <= elem.child_amount)
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                     else
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;
00292                 else
00293                     return 0;
00294             }
00295         }
00296         else {
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     }
00305     else {
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());
```

```
00317     this->setYear(elem.getYear());
00318     this->setYearD(elem.getYearD());
00319     this->setChildAmount(elem.getChildAmount());
00320     return *this;
00321 }
```

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 = "";
00009     std::string otchestvo = "";
00010     int year = 1900;
00011     int year_death = 1920;
00012     int child_amount = 0;
00013     int sex = 0;
00014 public:
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     std::string getIm();
00020     void setIm(std::string newIm);
00021     std::string getOt();
00022     void setOt(std::string newOt);
00023     int getYear();
00024     void setYear(int newYear);
00025     int getYearD();
00026     void setYearD(int newYearD);
00027     int getChildAmount();
00028     void setChildAmount(int newAmount);
00029     bool operator> (const Element& elem) const;
00030     bool operator< (const Element& elem) const;
00031     bool operator>= (const Element& elem) const;
00032     bool operator<= (const Element& elem) const;
00033     Element& operator= (Element& elem);
00034 };
```

5.5 Файл Main.cpp

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

Функции

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

См. определение в файле Main.cpp строка 104

```
00104                                   {
00105
00106     //сортировка по году рождения, потом по ФИО, потом по кол-ву детей
00107     for (int i = 0; i < spisok.size(); ++i) {
00108         for (int j = spisok.size() - 1; j > i; --j) {
00109             if (spisok[j - 1] > spisok[j]) {
00110                 Element temp = spisok[j - 1];
00111                 spisok[j - 1] = spisok[j];
00112                 spisok[j] = temp;
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 Main.cpp строка 68

```
00068                                                   {
00069     int largest = i;
00070     int left = 2 * i + 1;
00071     int right = 2 * i + 2;
00072     if (left<N && spisok[left]>spisok[largest])
00073         largest = left;
00074     if (right<N && spisok[right]>spisok[largest])
00075         largest = right;
00076     if (largest != i) {
00077         Element temp;
00078         temp = spisok[i];
00079         spisok[i] = spisok[largest];
00080         spisok[largest] = temp;
00081         heapify1(spisok, N, largest);
00082     }
00083 }
```

5.5.1.3 heapSort1()

```
void heapSort1 (
                std::vector< Element > & spisok,
                int N )
```

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

См. определение в файле Main.cpp строка 88

```
00088                                                {
00089     for (int i = N / 2 - 1; i >= 0; --i) {
00090         heapify1(spisok, N, i);
00091     }
00092     for (int i = N - 1; i >= 0; --i) {
00093         Element temp;
00094         temp = spisok[0];
00095         spisok[0] = spisok[i];
00096         spisok[i] = temp;
00097         heapify1(spisok, i, 0);
00098     }
00099 }
```

5.5.1.4 main()

```
int main ( )
```

Функция main для основной работы программы.

См. определение в файле Main.cpp строка 122

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

```
00166         while (getline(in, line)) {
00167             std::string word = "";
00168             int flag = 0;
00169             int amount = 0;
00170             Element human;
00171             for (int i = 0; i < line.size(); ++i) {
00172                 if (line[i] == '(')
00173                     flag = 2;
00174                 else if (line[i] == ')')
00175                     flag = 0;
00176                 else if (line[i] == ' '||line[i]=='\n') {
00177                     if (amount == 0) {
00178                         human.setFam(word);
00179                     }
00180                     else if (amount == 1)
00181                         human.setIm(word);
00182                     else if (amount == 2)
00183                         human.setOt(word);
00184                     else if (amount == 3)
00185                         human.setYear(std::stoi(word));
00186                     else if (amount == 4)
00187                         human.setYearD(std::stoi(word));
00188                     word = "";
00189                     ++amount;
00190                 }
00191                 else if (flag == 0) {
00192                     char symbol = line[i];
00193                     if (symbol != ' ') {
00194                         word.push_back(symbol);
00195                     }
00196                 }
00197 
00198                 if (amount == 5) {
00199                     amount = 0;
00200                     human.setChildAmount(line[line.size()-1]-'0');
00201                     spisok.push_back(human);
00202                 }
00203             }
00204         }
00205     }
00206     in.close();
00207     std::cout « spisok.size() « std::endl;
00208     std::cout « "\nParsing done!\n" « std::endl;
00209     while (choosedAlg != -2) {
00210         if (choosedAlg == 0) {
00211             std::cout « "\n\nPuzir started!\n";
00212             auto begin = std::chrono::steady_clock::now();
00213             std::vector<Element> res = spisok;
00214             bubbleSort(res);
00215             auto end = std::chrono::steady_clock::now();
00216             auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
00217             std::cout « "The time: " « elapsed_ms.count() « " ms";
00218             saveToFile(res, "C:\\Users\\fov-2\\source\\repos\\out1.txt");
00219             std::cout « "\nPuzir ended!\n\n";
00220         }
00221         else if (choosedAlg == 1) {
00222             std::cout « "\n\nShaker started!\n";
00223             auto begin = std::chrono::steady_clock::now();
00224             std::vector<Element> res = spisok;
00225             shakerSort(res);
00226 
00227             auto end = std::chrono::steady_clock::now();
00228             auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
00229             std::cout « "The time: " « elapsed_ms.count() « " ms";
00230             saveToFile(res, "C:\\Users\\fov-2\\source\\repos\\out11.txt");
00231             std::cout « "\nShaker ended!\n\n";
00232         }
00233         else if (choosedAlg == 2) {
00234             std::cout « "\n\nHeap started!\n";
00235             auto begin = std::chrono::steady_clock::now();
00236             std::vector<Element> res = spisok;
00237             heapSort1(res, res.size());
00238             auto end = std::chrono::steady_clock::now();
00239             auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
00240             std::cout « "The time: " « elapsed_ms.count() « " ms";
00241             saveToFile(res, "C:\\Users\\fov-2\\source\\repos\\out111.txt");
00242             std::cout « "\nHeap ended!\n\n";
00243         }
00244         std::cout « "\n\nEnd!\n\n";
00245         std::cout « "Would you like to sort again (0 - puzir, 1 - shaker, 2 - heap, -2 - end)?\n";
00246         std::cin » choosedAlg;
00247     }
00248     return 0;
00249 }
```

5.5.1.5 saveToFile()

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

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

См. определение в файле Main.cpp строка 22

```
00022                                                {
00023     std::ofstream out;
00024     out.open(filename);
00025     if (out.is_open()) {
00026         for (int i = 0; i < spisok.size(); ++i) {
00027             out « spisok[i].getFam() « " " « spisok[i].getIm() « " " « spisok[i].getOt() « " " « spisok[i].getYear() « " " «
      spisok[i].getYearD() « " " « spisok[i].getChildAmount() « '\n';
00028         }
00029     }
00030     out.close();
00031 }
```

5.5.1.6 shakerSort()

```
void shakerSort (
                std::vector< Element > & spisok )
```

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

См. определение в файле Main.cpp строка 37

```
00037                                          {
00038     int j, k = spisok.size() - 1;
00039     int lb = 1, ub = spisok.size() - 1;
00040     Element temp;
00041     while (lb < ub) {
00042         for (j = ub; j > 0; --j) {
00043             if (spisok[j - 1] > spisok[j]) {
00044                 temp = spisok[j - 1];
00045                 spisok[j - 1] = spisok[j];
00046                 spisok[j] = temp;
00047                 k = j;
00048             }
00049         }
00050         lb = k + 1;
00051         for (j = 1; j <= ub; ++j) {
00052             if (spisok[j - 1] > spisok[j]) {
00053                 temp = spisok[j - 1];
00054                 spisok[j - 1] = spisok[j];
00055                 spisok[j] = temp;
00056                 k = j;
00057             }
00058         }
00059         ub = k - 1;
00060     }
00061 }
```

5.6 Main.cpp

См. документацию.

```
00001 #include <iostream>
00002 #include <fstream>
00003 #include "Element.h"
00004 #include <vector>
00005 #include <chrono>
00006
```

```
00022 void saveToFile(std::vector<Element> spisok, std::string filename) {
00023     std::ofstream out;
00024     out.open(filename);
00025     if (out.is_open()) {
00026         for (int i = 0; i < spisok.size(); ++i) {
00027             out « spisok[i].getFam() « " " « spisok[i].getIm() « " " « spisok[i].getOt() « " " « spisok[i].getYear() « " " « spisok[i].getYearD() « " " « spisok[i].getChildAmount() « '\n';
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() - 1;
00040     Element temp;
00041     while (lb < ub) {
00042         for (j = ub; j > 0; --j) {
00043             if (spisok[j - 1] > spisok[j]) {
00044                 temp = spisok[j - 1];
00045                 spisok[j - 1] = spisok[j];
00046                 spisok[j] = temp;
00047                 k = j;
00048             }
00049         }
00050         lb = k + 1;
00051         for (j = 1; j <= ub; ++j) {
00052             if (spisok[j - 1] > spisok[j]) {
00053                 temp = spisok[j - 1];
00054                 spisok[j - 1] = spisok[j];
00055                 spisok[j] = temp;
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) {
00069     int largest = i;
00070     int left = 2 * i + 1;
00071     int right = 2 * i + 2;
00072     if (left<N && spisok[left]>spisok[largest])
00073         largest = left;
00074     if (right<N && spisok[right]>spisok[largest])
00075         largest = right;
00076     if (largest != i) {
00077         Element temp;
00078         temp = spisok[i];
00079         spisok[i] = spisok[largest];
00080         spisok[largest] = temp;
00081         heapify1(spisok, N, largest);
00082     }
00083 }
00084
00088 void heapSort1(std::vector<Element> &spisok, int N) {
00089     for (int i = N / 2 - 1; i >= 0; --i) {
00090         heapify1(spisok, N, i);
00091     }
00092     for (int i = N - 1; i >= 0; --i) {
00093         Element temp;
00094         temp = spisok[0];
00095         spisok[0] = spisok[i];
00096         spisok[i] = temp;
00097         heapify1(spisok, i, 0);
00098     }
00099 }
00100
00104 void bubbleSort(std::vector<Element>& spisok) {
00105
00106     //сортировка по году рождения, потом по ФИО, потом по кол-ву детей
00107     for (int i = 0; i < spisok.size(); ++i) {
00108         for (int j = spisok.size() - 1; j > i; --j) {
00109             if (spisok[j - 1] > spisok[j]) {
00110                 Element temp = spisok[j - 1];
00111                 spisok[j - 1] = spisok[j];
00112                 spisok[j] = temp;
00113             }
00114         }
00115     }
00116 }
00117 //--------------
00118
00122 int main() {
```

```
00123
00124     int file_number = 0;
00125     std::vector<Element> spisok;
00126     int choosedAlg = 0;
00127     std::cout « "\nChoose sorting algorithm (0 - puzir, 1 - shaker, 2 - heap): \n";
00128     std::cin » choosedAlg;
00129     while (choosedAlg < 0 || choosedAlg >= 3 || std::cin.fail()) {
00130         if (std::cin.fail()) {
00131             std::cin.clear();
00132             std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00133             std::cout « "Your input is not a number!" « '\n';
00134             choosedAlg = -1;
00135         }
00136         else {
00137             std::cout « "Your input must be in interval 0-7!" « '\n';
00138             choosedAlg = 0;
00139         }
00140         std::cout « "Input again!" « '\n';
00141         std::cin » choosedAlg;
00142     }
00143     std::cout « "Choose file(0-7)" « '\n';
00144     std::cin » file_number;
00145     while (file_number < 0 || file_number >= 8 || std::cin.fail()) {
00146         if (std::cin.fail()) {
00147             std::cin.clear();
00148             std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00149             std::cout « "Your input is not a number!" « '\n';
00150             file_number = -1;
00151         }
00152         else {
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     }
00159     setlocale(LC_ALL, "rus");
00160     std::string line = "";
00161     std::string filename = "C:\\Users\\fov-2\\source\\repos\\data_";
00162     filename.append(std::to_string(file_number));
00163     filename.append(".txt");
00164     std::ifstream in(filename);
00165     if (in.is_open()) {
00166         while (getline(in, line)) {
00167             std::string word = "";
00168             int flag = 0;
00169             int amount = 0;
00170             Element human;
00171             for (int i = 0; i < line.size(); ++i) {
00172                 if (line[i] == '(')
00173                     flag = 2;
00174                 else if (line[i] == ')')
00175                     flag = 0;
00176                 else if (line[i] == ' '||line[i]=='\n') {
00177                     if (amount == 0) {
00178                         human.setFam(word);
00179                     }
00180                     else if (amount == 1)
00181                         human.setIm(word);
00182                     else if (amount == 2)
00183                         human.setOt(word);
00184                     else if (amount == 3)
00185                         human.setYear(std::stoi(word));
00186                     else if (amount == 4)
00187                         human.setYearD(std::stoi(word));
00188                     word = "";
00189                     ++amount;
00190                 }
00191                 else if (flag == 0) {
00192                     char symbol = line[i];
00193                     if (symbol != ' ') {
00194                         word.push_back(symbol);
00195                     }
00196                 }
00197
00198                 if (amount == 5) {
00199                     amount = 0;
00200                     human.setChildAmount(line[line.size()-1]-'0');
00201                     spisok.push_back(human);
00202                 }
00203             }
00204         }
00205     }
00206     in.close();
00207     std::cout « spisok.size() « std::endl;
00208     std::cout « "\nParsing done!\n" « std::endl;
00209     while (choosedAlg != -2) {
```

```
00210         if (choosedAlg == 0) {
00211             std::cout « "\n\nPuzir started!\n";
00212             auto begin = std::chrono::steady_clock::now();
00213             std::vector<Element> res = spisok;
00214             bubbleSort(res);
00215             auto end = std::chrono::steady_clock::now();
00216             auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
00217             std::cout « "The time: " « elapsed_ms.count() « " ms";
00218             saveToFile(res, "C:\\Users\\fov-2\\source\\repos\\out1.txt");
00219             std::cout « "\nPuzir ended!\n\n";
00220         }
00221         else if (choosedAlg == 1) {
00222             std::cout « "\n\nShaker started!\n";
00223             auto begin = std::chrono::steady_clock::now();
00224             std::vector<Element> res = spisok;
00225             shakerSort(res);
00226 
00227             auto end = std::chrono::steady_clock::now();
00228             auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
00229             std::cout « "The time: " « elapsed_ms.count() « " ms";
00230             saveToFile(res, "C:\\Users\\fov-2\\source\\repos\\out11.txt");
00231             std::cout « "\nShaker ended!\n\n";
00232         }
00233         else if (choosedAlg == 2) {
00234             std::cout « "\n\nHeap started!\n";
00235             auto begin = std::chrono::steady_clock::now();
00236             std::vector<Element> res = spisok;
00237             heapSort1(res, res.size());
00238             auto end = std::chrono::steady_clock::now();
00239             auto elapsed_ms = std::chrono::duration_cast<std::chrono::milliseconds>(end - begin);
00240             std::cout « "The time: " « elapsed_ms.count() « " ms";
00241             saveToFile(res, "C:\\Users\\fov-2\\source\\repos\\out111.txt");
00242             std::cout « "\nHeap ended!\n\n";
00243         }
00244         std::cout « "\n\nEnd!\n\n";
00245         std::cout « "Would you like to sort again (0 - puzir, 1 - shaker, 2 - heap, -2 - end)?\n";
00246         std::cin » choosedAlg;
00247     }
00248     return 0;
00249 }
```

Предметный указатель