ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА (РОСАВИАЦИЯ)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра вычислительных машин, комплексов, сетей и систем.

Курсовая работа
защищена с оценкой

_____

_____
(подпись преподавателя, дата)

КУРСОВАЯ РАБОТА
по дисциплине: «Программирование»

Вариант № 26

Тема: «Разработать систему обработки данных о рождаемости в некотором городе.»

Выполнил студент группы ЭВМ 1-2
Насыубулли С. Р.
Руководитель: доцент, к.ф.- м.н. Надейкина Л. А

МОСКВА - 2022

Приложение 1

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА (РОСАВИАЦИЯ)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра_____ВМКСС_____

ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ (РАБОТУ)

По дисциплине

_____Программирование_____

студента Насыбулли С. Р. ЭВМ - 211037 группы ЭВМ 1-2
(фамилия, имя, отчество, шифр)

1. Наименование темы: «Разработать систему обработки данных о фильмах текущего репертуара ряда кинотеатров»

2. Задание и основные характеристики: разработать систему обработки данных о фильмах текущего репертуара ряда кинотеатров, обеспечивающую хранение, просмотр, изменение данных, получение ряда характеристик.

3. Объем: пояснительная записка на ____8____ листах, графическая часть на _____17____ листах формата А4.

Руководитель: _доцент каф. ВМКСС _____ Надейкина Л.А.
(должность) (подпись) (фамилия, имя, отчество)

«___1____»_____июня___2022_г.

Студент: _____ _____
(подпись) (фамилия, имя, отчество)

«___1_____»_____июня_____2022__г.

Анотация.

В работе представлены результаты выполнения курсовой работы.

В отчете представлены цель работы, техническое задание, структуры программ, таблицы глобальных переменных и функций, листинги и результаты выполнения программы.

В курсовой работе была разработа система просмотра, обработки, удаления записей о рожднеии ребенка. Совершения поиска и сортировки записей.

Оглавление.

Техническое задание на разработку системы.

Вариант 26.

Разработать систему обработки данных о рождаемости в некотором городе.

В городе несколько районов и в каждом районе несколько Родильных домов.

Данные о рождении детей непрерывно поступают на некоторый статистический пункт из различных Роддомов различных районов и помещаются в текстовой файл данных.

Строки файла данных содержат следующие данные:

Номер роддома	Дата рождения ребенка	Район	ФИО матери	Дата рождения матери	Пол. Ребенка 1	Пол. Ребенка 2	Пол. Ребенка 3

Для обозначения пола ребенка использовать символы ‘ж’, ‘м’ соответственно и – ‘0’ – если нет второго, третьего ребенка.

Количество Роддомов и их номера заранее не известны (система должна работать с произвольным файлом данных), эти данные определяются программно из файла данных.

Создаются бинарные файлы с базами данных по каждому Роддому.

Система выполняет несколько видов обработки данных. Для выбора варианта обработки данных используется меню.

Виды обработки:

1. Просмотр данных за любой день (или за любой временной интервал)
   A. по заданному Роддому;
   B. по району;
   C. по городу.

Данные выводить в таблицу в хронологическом порядке.

2. Вывод гистограммы рождаемости по месяцам года и кривой рождаемости за год.
   A. для заданного Роддома;
   B. для заданного района;
   C. для города;
      - общей рождаемости;
      - только мальчиков;
      - только девочек;
      - многодетных родов (более одного ребенка).
3. Определение месяцев максимальной и минимальной рождаемости по заданному Роддому, по району, по городу.
   - общей;

- мальчиков;
- девочек;
- многодетных родов.

4. Удаление записей о родах (поиск по фамилии матери и по дате ее рождения).

Возможности разработанной системы.

Система для управления базами данных о рождении детей. Система обладает минимальным функциями для работы.

Программа представленная в данной работе позволяет:

1. Просматривать записи о рождении детей в хронологическом порядке в заданном временном интервале или за определенный день.
2. Вывод гистограммы рождаемости.
3. Нахождения минимального и максимального месяца рождаемости.
4. Удаления заданной записи.
5. Создания баз данных в бинарных файлах.
6. Созданиях из текстового файла бинарных фалов.
7. Вывод текстового протокола в файл.

Алгоритмы функций обработки данных.

Model:

struct Date	
description()	Возвращает дату в формате: дд.мм.гггг

class Children	
append(Sex newElement)	Добавляет новый элемент в конец массива дитей.
attribute(Attribute attribute, Children children)	Функция проверки по признаку (только девочи итд).
isAttribute(Sex sex)	Проверка детей на один пол.
isMultiple()	Проверка на многодетность.

class SexCast	
toString(Sex sex)	Преобразует enum Sex в английский символ.
toStringRus(Sex sex)	Преобразует enum Sex в русский символ.
toSexEnum(string sex)	Преобразует русский символ в enum Sex.

struct Birth	
getDescription()	Возвращает массив строк полей структуры Birth.

class City	
getName()	Возвращает имя города
setName(string name)	Устанавливает имя города
append(DictionaryRegion region)	Добавляет новый элемент в конец regions
find(string region)	Возвращает регион по слову
getCount()	Возвращает кол-во регионов в городе
getNumbers(Area area, string areaText = "")	Обработка выдачи номера роддомов
getAllRegionName()	Возвращает все регионы в городе
getAllNumberName()	Возвращает все номера в городе
allNumbers()	Возвращает номера всех роддомов входящие в город

class Processing	
processing(const Birth &birth) = 0	Функция обработки в теле цикла.
processingEnd() = 0	Функция обработки в конце цикла.

class ViewBirthProcessing	
processing(const Birth &birth)	Функция проверки вхождения во временной интервал или в день.
processingEnd()	Функция сортировки в хронологическом порядке.

class HistogramProcessing	
processing(const Birth &birth)	Функция подсчета кол-во рождений в каждом месяце.
processingEnd()	-

class BirthrateProcessing	
processing(const Birth &birth)	Функция подсчета кол-во рождений в каждом месяце.
processingEnd()	Поиск максимума и минимума.

class DeleteProcessing	
processing(const Birth &birth)	Функция поиска по фамилии и году рождения
processingEnd()	-

View:

class ViewText	
output(ostream &out) = 0	Вывод текста.

class CastAttribute	
attributeText(Attribute attribute)	Выводит текст по enum Attribute.

class TableViewText:	
output(ostream &out)	Вывод таблицы
tableHeaderViewText (ostream &out)	Текст, который отображается над содержимым таблицы.
tableFooterViewText (ostream &out)	Текст, который отображается под содержимым таблицы.
lineHeader(ostream &out, string text[column])	Вывод шапки
line(ostream &out)	Вывод разделителя

struct TableViewTextCell:	
widthSpace(long int sizeText, int column, Align align)	Возвращает количество - смещение влево первого символа строки

class HistogramViewText:	
output(ostream &out)	Выводит гистограмму рожаемости.
histogramString(int count, bool availability[12])	Выводит строчку гистограммы.
availability(int count, const int mouth[12])	Просматривает есть ли рождаемость на заданном кол-во (уровни).

class BirthrateViewText:	
output(ostream &out)	Вывод максимального и минимального месяца рождаемости.

class DeleteViewText:	
output(ostream &out)	Вывод имени и даты рождения матери.

class Menu	
open(DataModel &dataModel, City &city)	Вывод меню обработки.
choiceProcessingMenu(DataModel &dataModel)	Вывод меню выбора обработки.
dateFormatMenu(DataModel &dataModel)	Вывод меню ввода даты.
menuArea(DataModel &dataModel, City &city)	Вывод меню ввода местности.
menuAttributeBirthrate(DataModel &dataModel)	Вывод меню ввода признака.
menuDelete(DataModel &dataModel)	Вывод меню удаления матери по фамилии и имени.
clearTab()	Очищяет терминал
dateInput (DataModel &dataModel)	Ввод даты.
dateFormatInput (DataModel &dataModel)	Ввод числа формата даты.
choiceProcessingInput (DataModel &dataModel)	Ввод числа выбора процесса.
areaInput(DataModel &dataModel)	Ввод числа выбора местности.
areaTextInput (DataModel &dataModel)	Ввод названия местности.
attributeInput (DataModel &dataModel)	Ввод числа выбора признака.
outCity (Area area, City &city)	Ввод названий: города, региона, номера.
output(vector<string> array)	Выводит массива.
output (string text)	Вывод строчки.

class Input	
number()	Ввод с клавиатуры целого числа int
text()	Ввод с клавиатуры текста string
dateCast (string dataText, DataFormat format)	Приводит строчку к Data
choiceProcessingCast(int number)	Приводит число к перечислению ChoiceProcessing
areaCast(int number)	Приводит число к перечислению Area
dataFormatCast(int number)	Приводит число к перечислению DataFormat
attributeCast(int number)	Приводит число к перечислению Birthrate

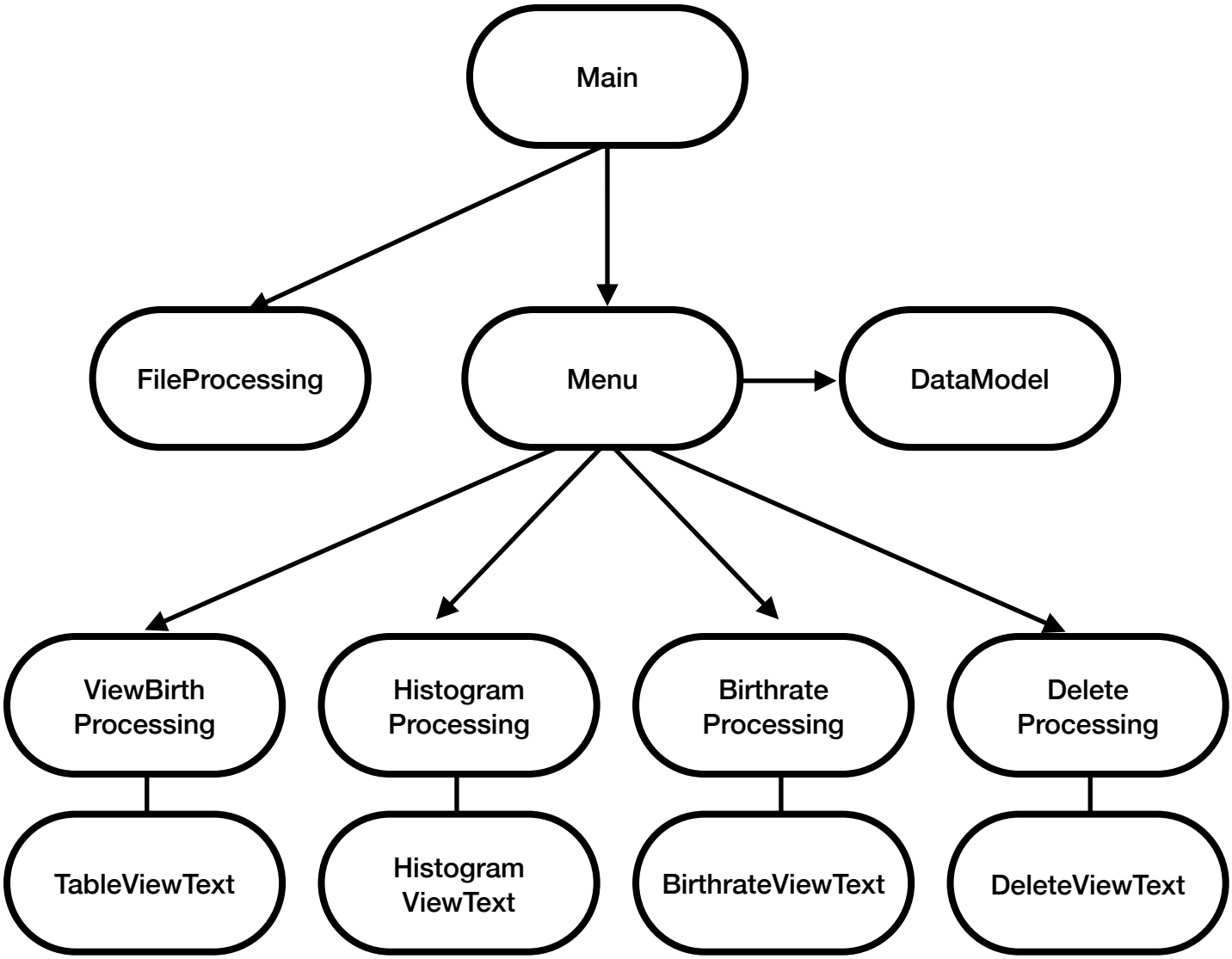
Controller:

class FileProcessing	
initCity(City &city)	Заполнения данных о городе, регионов, номеров роддомов в модель экземпляр класс city.
initHospital()	Чтения из исходных данных и создания бинарных файлов по номерам роддомов.
processing (Processing &processing, const vector<int> &numbers)	Функция обработки файла(ов). Принимает: класс обработки (содержит алгоритм обработки) и номера роддомов.
output(ViewText &viewText)	Функция вывода в файл «протокл». Принимает класс вывода текста.
removeBirth (const DeleteProcessing &processing)	Функция удаления записи из бинарного файла. Принимает класс обработки «Удаления».
removeHospitalFile (City &city)	Функция удаления баз данных (бинарных файлов) после полного завершения работы.
openRead(string resource, ios_base::openmode __mode = ios::in)	Открывает поток read ifstream
openWrite(string resource, ios_base::openmode __mode = ios::out)	Открывает поток write ofstream
openFile(string resource, ios_base::openmode __mode)	Открывает поток file fstream

class ExtensionString	
componentsSeparatedBy(string text, char separatedBy)	Возвращает массив, содержащий подстроки из получателя, которые были разделены на данный разделитель.
leading (int number, int count, char leadingChar = '0')	Добавляет лидирующие символы.

class Error	
what()	Выводит строчку ошибки

Структоура программы.



Программа системы.

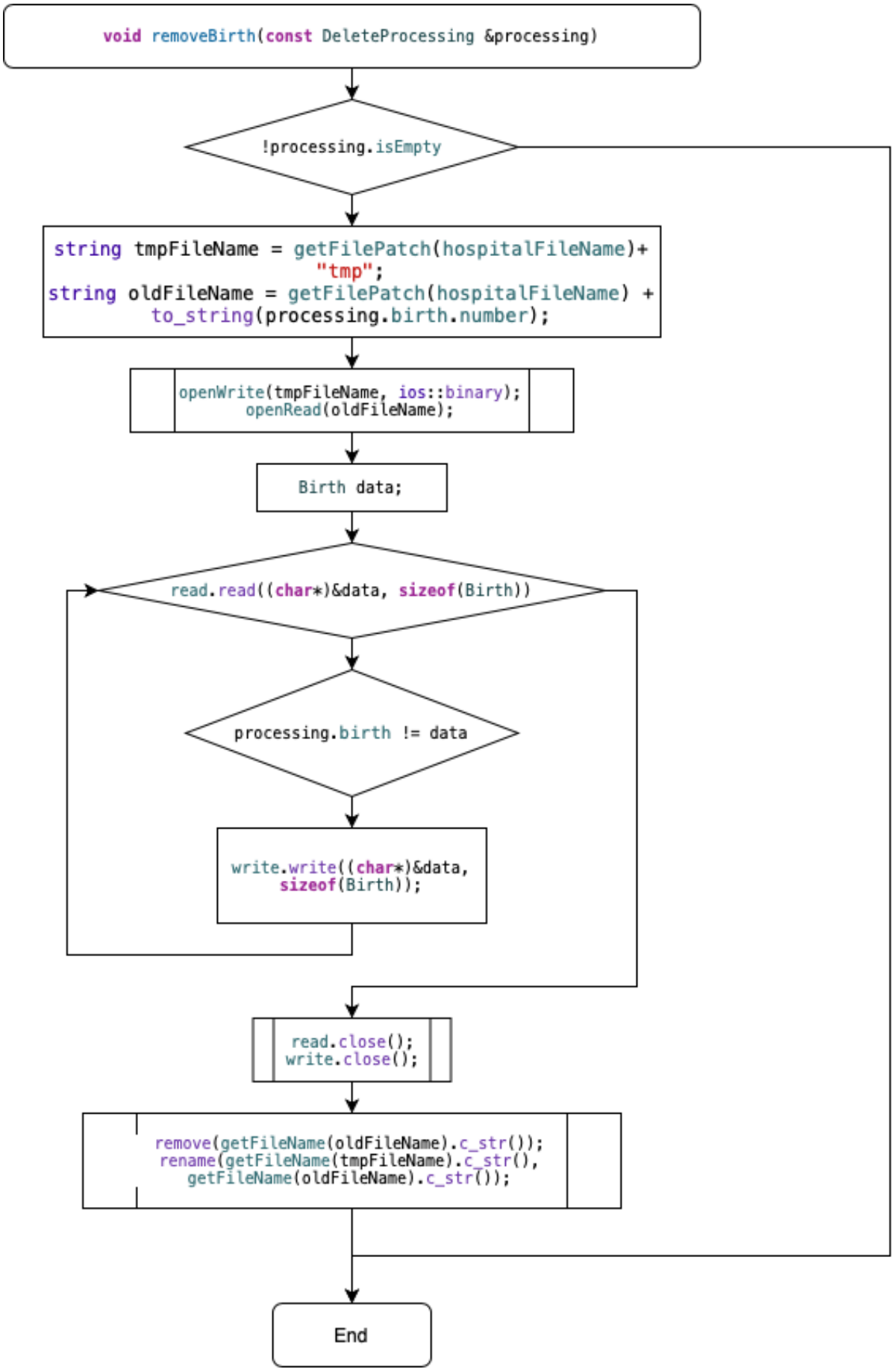
Таблицу файлов приложения

Наименование файла	Назначение файла	Объем файла байтах	Стр. в Приложении с данными файла
Date.hpp	Реализует класс дата.	463,25	
Children.hpp	Реализует класс дети.	384,375	
Birth.hpp	Модель записи о рождении	493,75	
City.hpp	Модель города с регионами и номерами	432,75	
DataModel.hpp	Модель для передачи состояния приложения	92,75	
Processing.hpp	Реализует методы обработки данных	464,75	
FileProcessing.hpp	Реализует работу с фалами. Запись или чтение из файл, открытия файла, закрытие.	703,125	
main.cpp	Главный файл	441,375	
ChoiceMenu.hpp	Содержит перечисления для выбора обработки данных.	178,625	
ClassError.hpp	Реализует класс ошибки	145,875	
ExtensionString.hpp	Класс расширения для string	211,125	
Input.hpp	Реализует класс ввода данных.	498,5	
Menu.hpp	Вывод меню	876,125	
ViewText.hpp	Реализует интерфейс для вывода текста	65,375	
DeleteViewText.hpp	Вывод текста удаления	98,5	
BirthrateViewText.hpp	Вывод текста для максимальной и минимально рождаемости	289,375	
HistogramViewText.hpp	Вывод текстовой гистограммы	412,25	
TableViewText.hpp	Вывод таблицы	352,875	
TableViewTextCell.hpp	Вывод строчки таблицы	137,875	
Birth.txt	Данные для тестирования программы	360,125	
Protocol.txt	Протокол для вывода результатов	-	

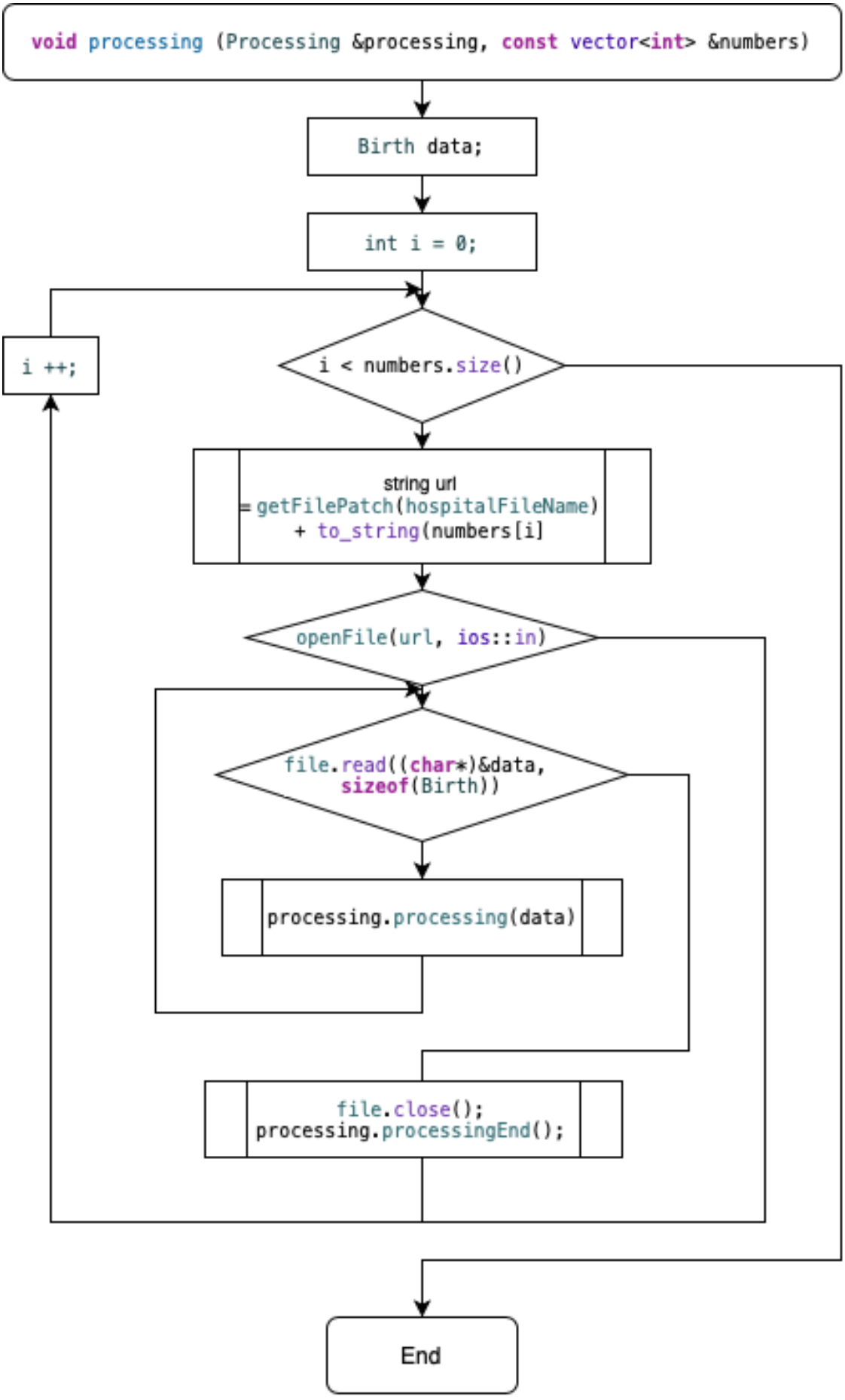
Таблицу глобальных переменных.

fstream file;	Поток – чтения – записи
ifstream read;	Поток – чтения
ofstream write;	Поток – записи
using DictionaryRegion = map<string, vector<int>>;	Псевдоним контейнира map<string, vector<int>>

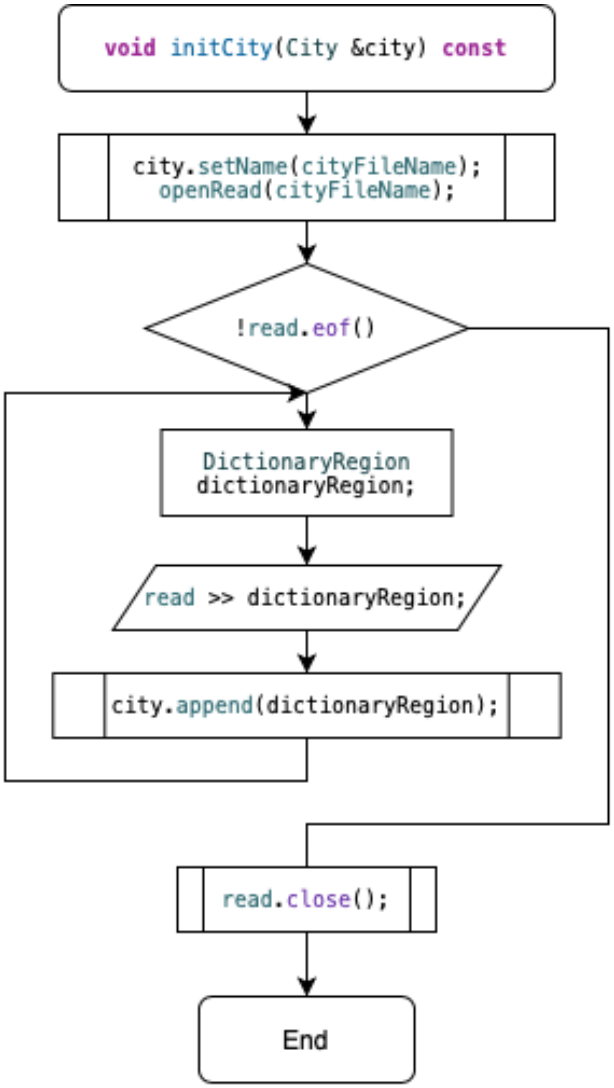
Блок – схемы:



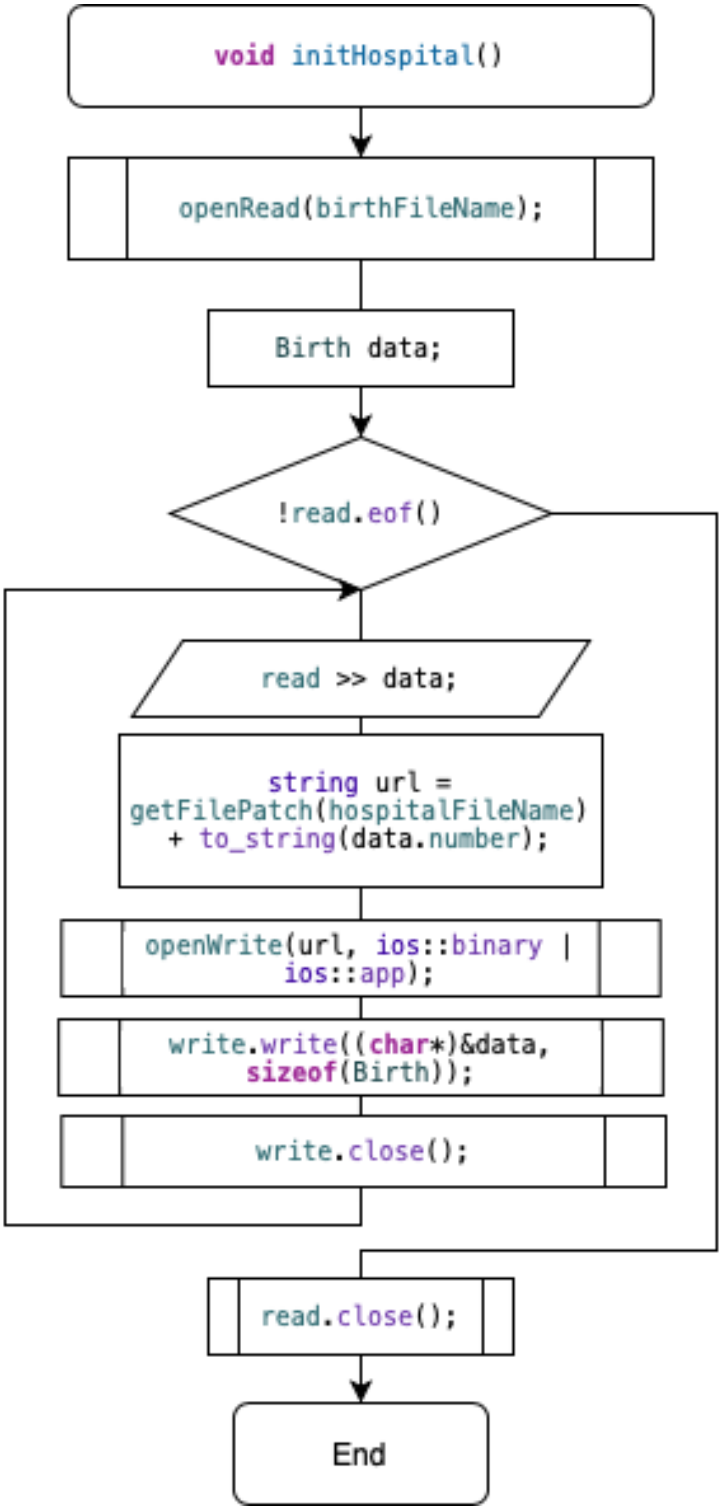
Функция удаления записи.



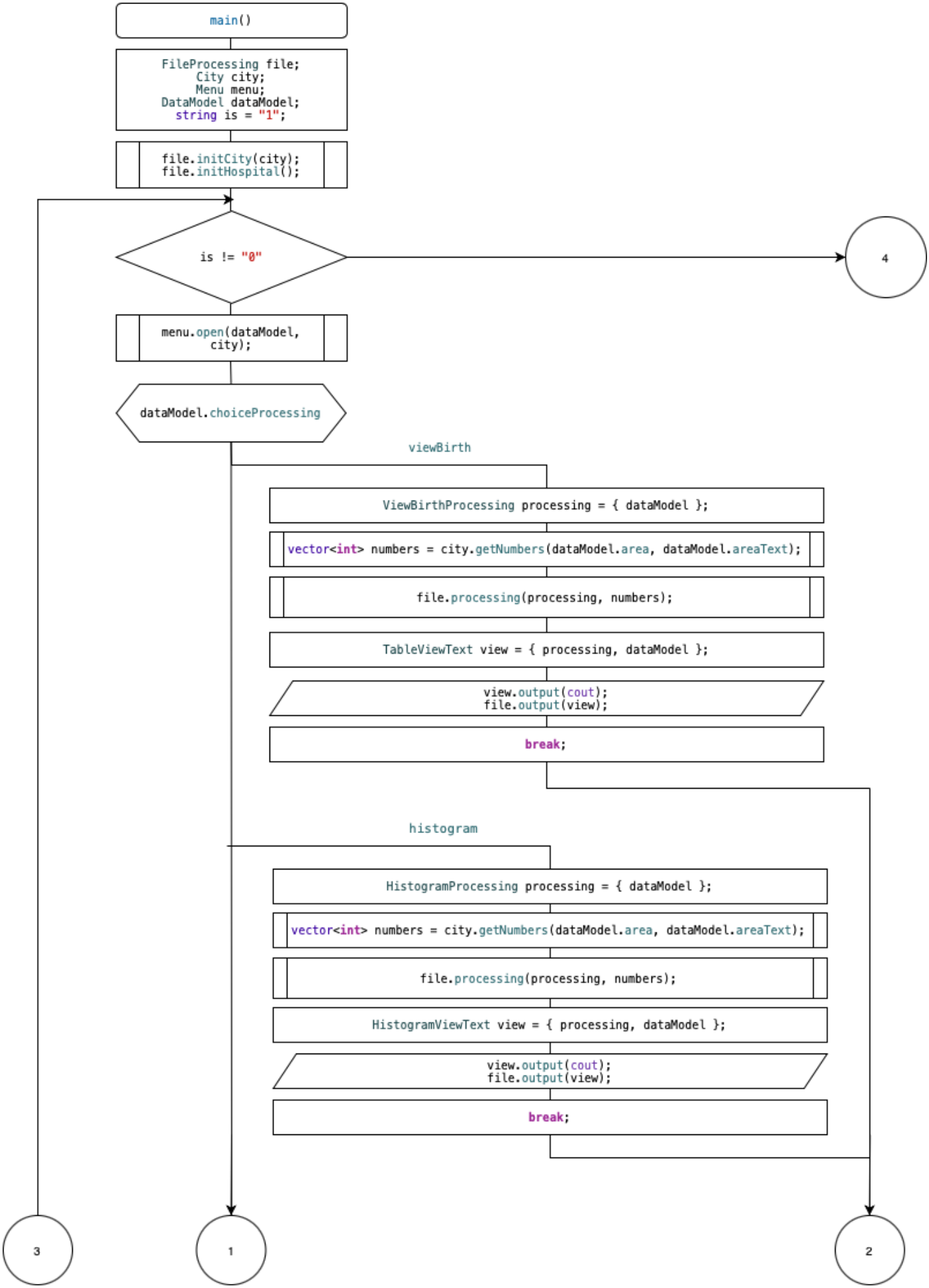
Функция обработки

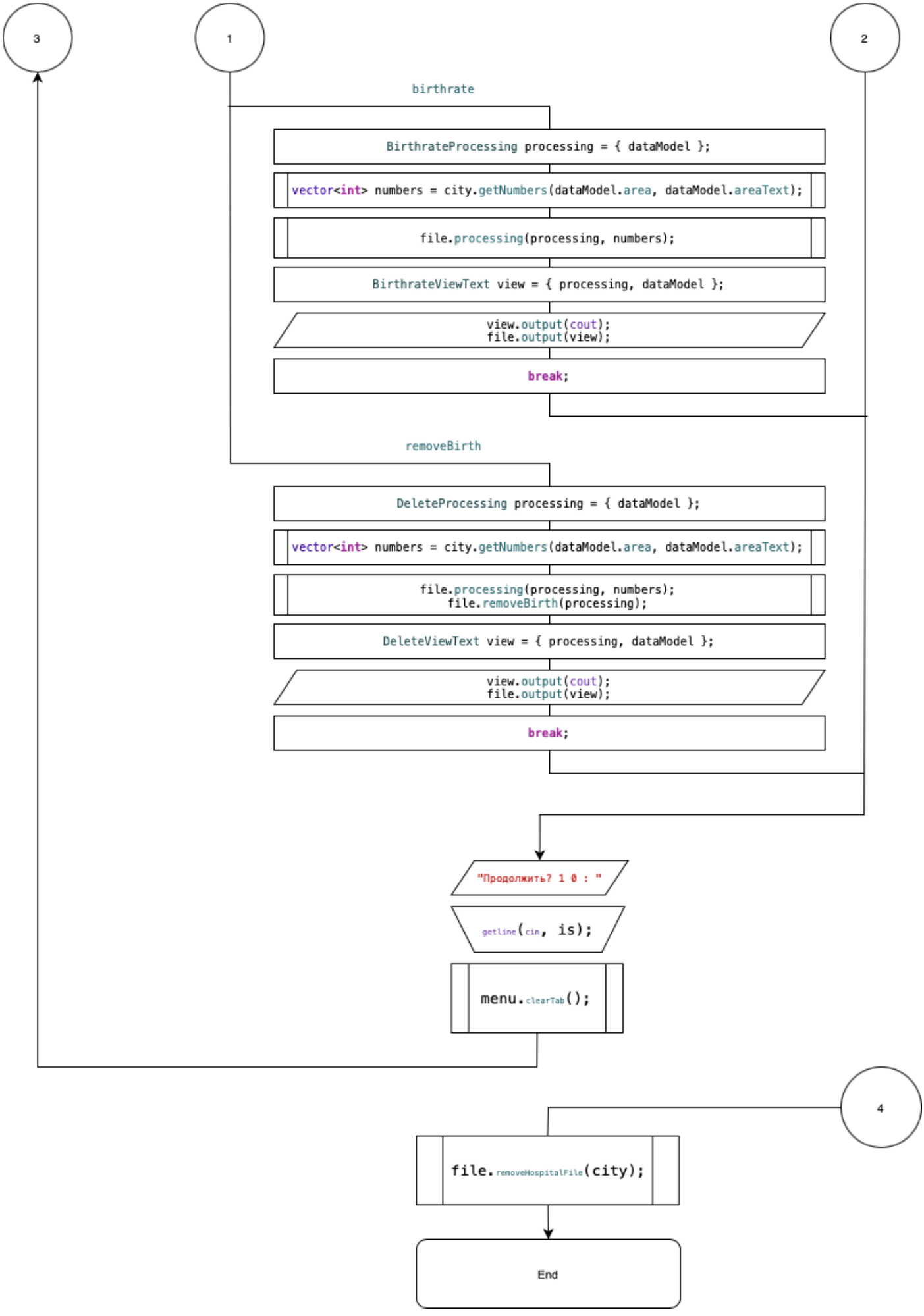


Функция чтения и записи регионов города

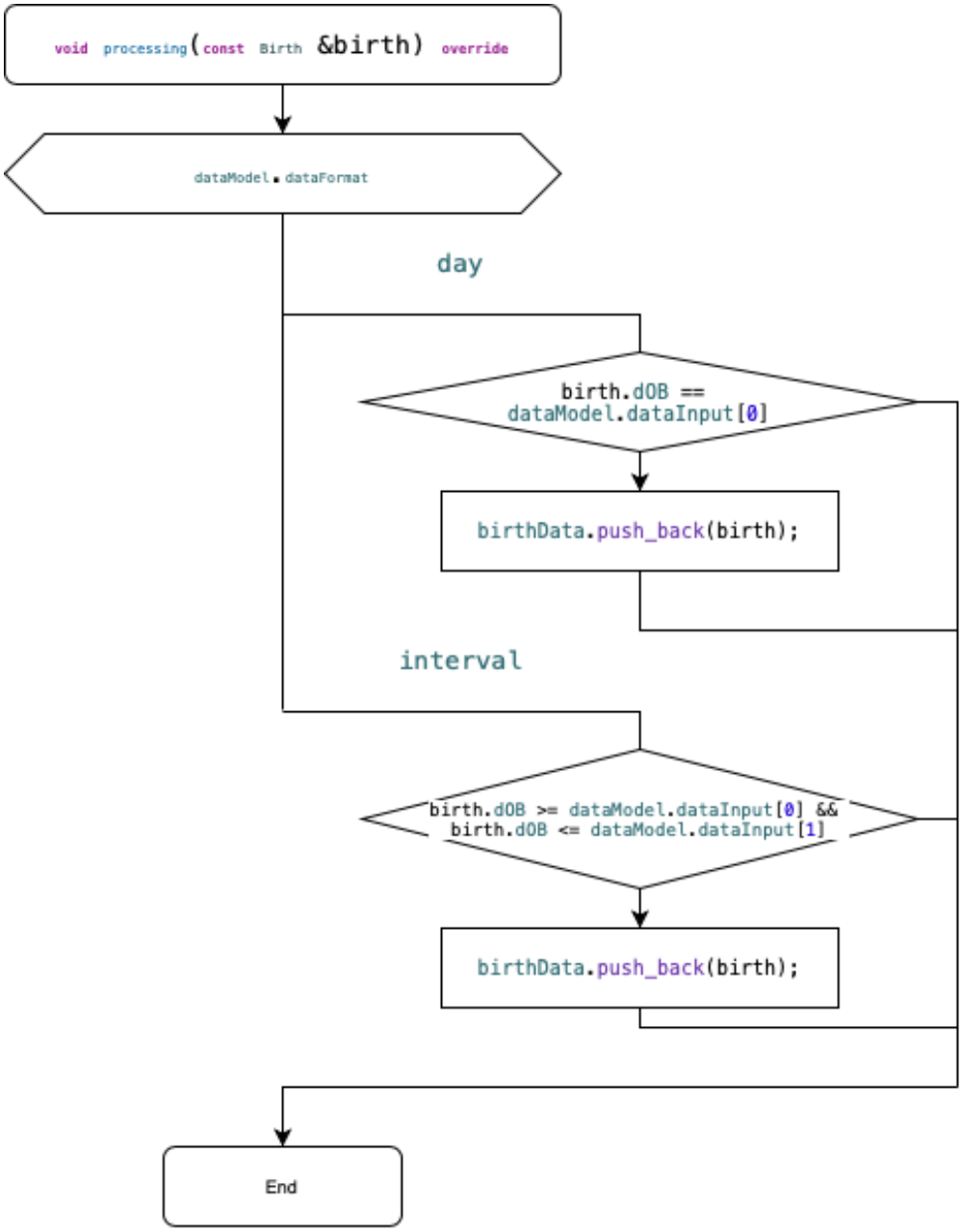


Создания чтения записей и создания бинарных файлов

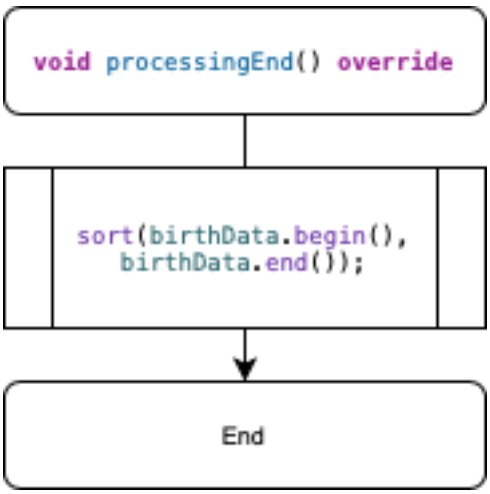

main()
FileProcessing file;
City city;
Menu menu;
DataModel dataModel;
string is = "1";
file.initCity(city);
file.initHospital();
is != "0"
4
menu.open(dataModel, city);
dataModel.choiceProcessing
viewBirth
ViewBirthProcessing processing = { dataModel };
vector<int> numbers = city.getNumbers(dataModel.area, dataModel.areaText);
file.processing(processing, numbers);
TableViewText view = { processing, dataModel };
view.output(cout);
file.output(view);
break;
histogram
HistogramProcessing processing = { dataModel };
vector<int> numbers = city.getNumbers(dataModel.area, dataModel.areaText);
file.processing(processing, numbers);
HistogramViewText view = { processing, dataModel };
view.output(cout);
file.output(view);
break;
3
1
2


3
1
2
birthrate
BirthrateProcessing processing = { dataModel };
vector<int> numbers = city.getNumbers(dataModel.area, dataModel.areaText);
file.processing(processing, numbers);
BirthrateViewText view = { processing, dataModel };
view.output(cout);
file.output(view);
break;
removeBirth
DeleteProcessing processing = { dataModel };
vector<int> numbers = city.getNumbers(dataModel.area, dataModel.areaText);
file.processing(processing, numbers);
file.removeBirth(processing);
DeleteViewText view = { processing, dataModel };
view.output(cout);
file.output(view);
break;
"Продолжить? 1 0 : "
getline(cin, is);
menu.clearTab();
4
file.removeHospitalFile(city);
End

Для класс ViewBirthProcessing:

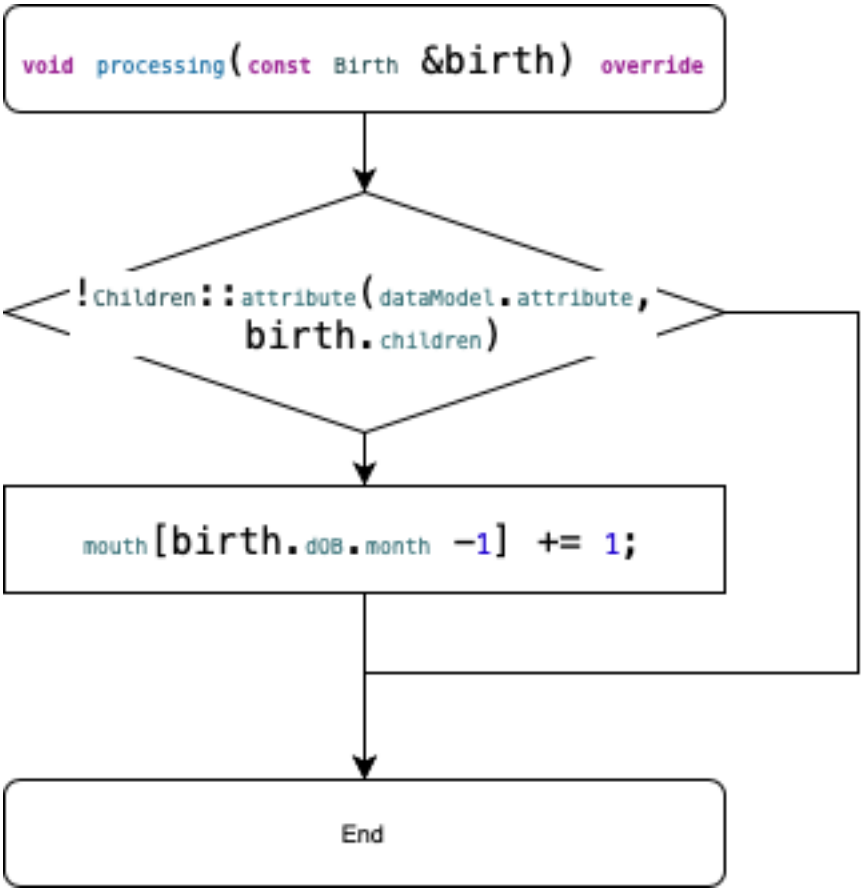


Алгоритм отбора по дате (поиск по дате, интервалу)



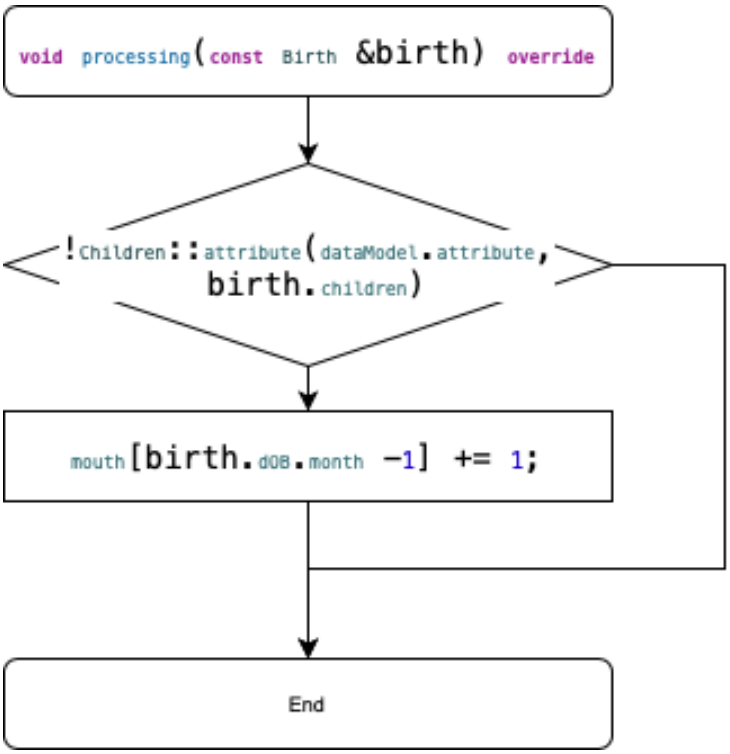
Сортировка записей входящих
во временой интервал

Для класс HistogramProcessing:

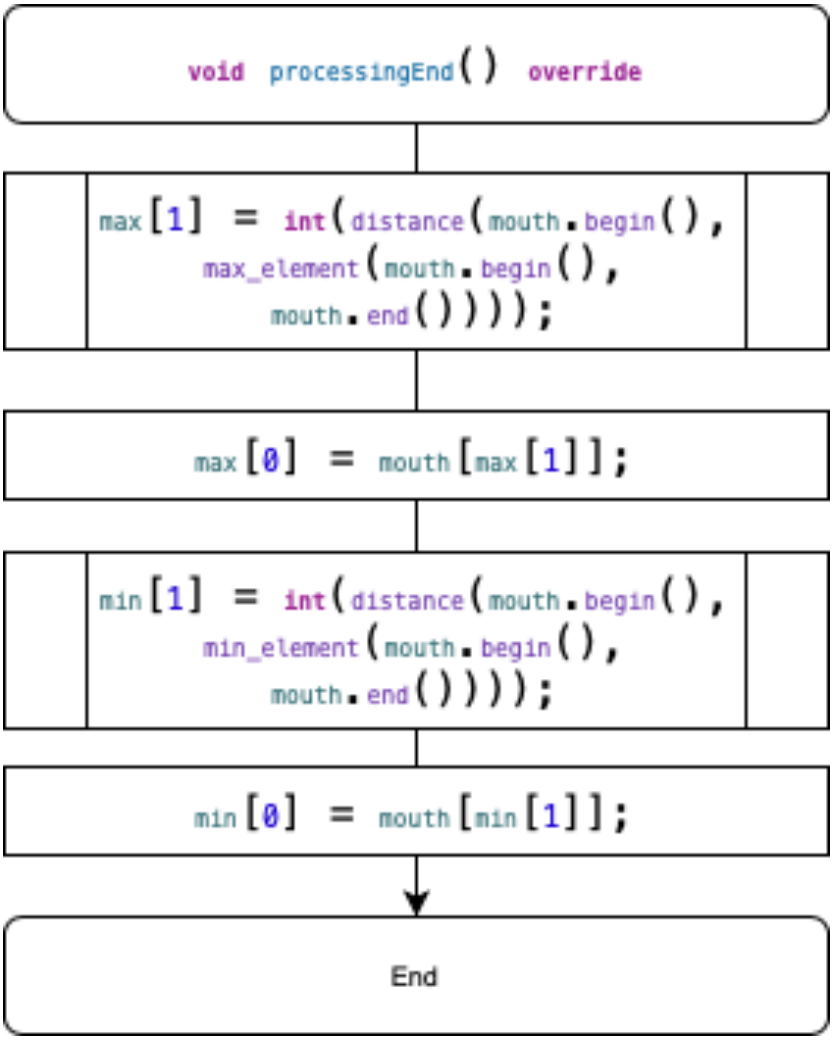


Отбор по критерию

Для класс BirthrateProcessing:

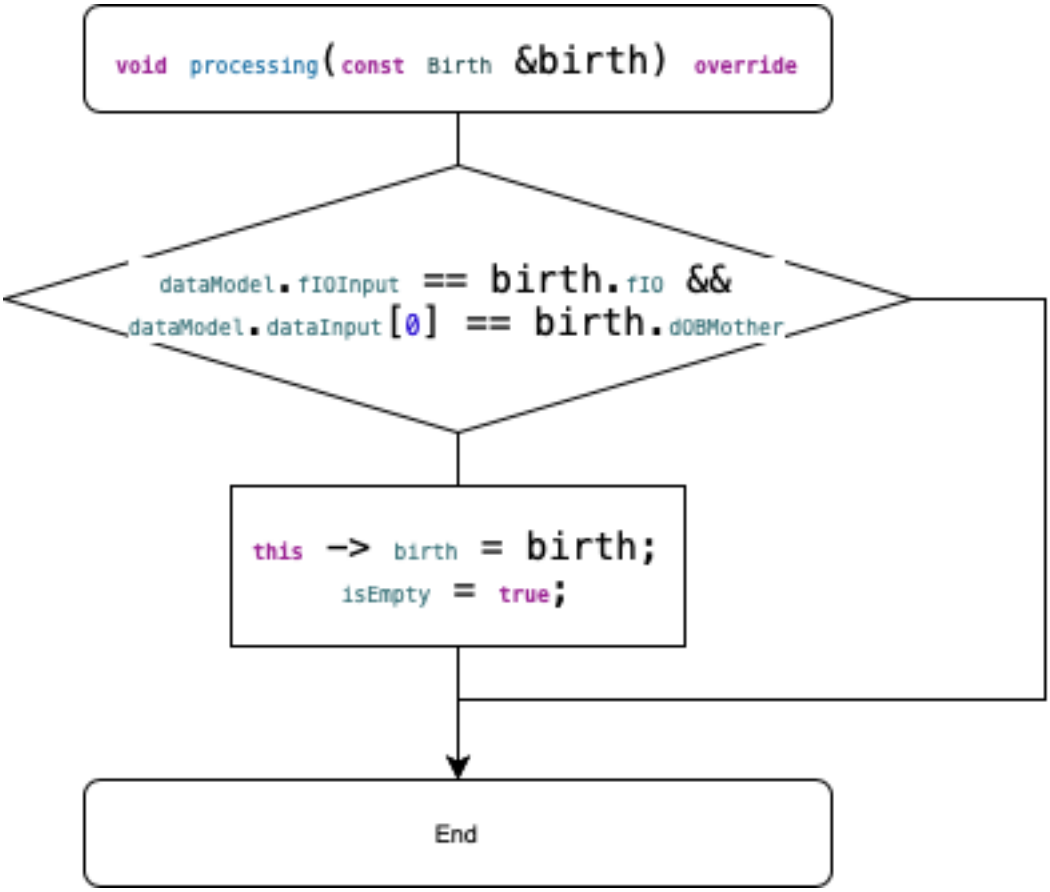


Отбор по критерию



Поиск максимума и минимума

Для класс DeleteProcessing:



Поиск записо о рождении. По дате рождению и ФИО

Руководство по использованию.

– Правила установки, запуска и использования системы.

Для запуска необходима:

- Windows 10 и Visual Studio 2022.
- Mac OS 11 и Xcode 13.

– Необходимы файлы.

- Файл «*город» с наименование регионов с номерами роддомов которые входят в них. (*Названия города на русском).
- Папка «hospital» в которой будут хранится бинарные файла для каждого роддома.
- Файл с иходными данными.

Все файлы должны находится в одной папке.

– Правила подготовки файлов с исходными данными.

Правило заполнения исходных данных рождения детей:

- Записи в конце должно разделятся «\n».
- Каждое поле разделяется «|». Между данных нет пробелов.
- Имя и регион заполняется на английском.

Пример:

1|28.10.7929|Central|Jennifer Hill|2.5.2084|ж|ж|м

Правило заполнения файл «*Город»:

- В файле идет перечисления регионов в конце разделятся «\n». А в каждом регионе пересечения номеров роддомов через «,».
- Регионы заполняются на английском.

Пример:

Central|1,2,3

– Правила пользования системой диалога: запросами меню;

При запуске программы будет выведена меню. Следуете указанием меню. Для выхода из программы нужно заполнить все предложенные формы до тех пор пока не будет предложено завершить программу.

Список литературы.

1. Л.А. Надейкина «Пособие к выполнению курсовой работы по дисциплине «программирование» для студентов 2 курса специальности 23.01.00 дневного обучения.», 2011г.;
2. Интернет-ресурс: en.cppreference.com
3. Интернет-ресурс: www.cplusplus.com
4. Интернет-ресурс: docs.microsoft.com
5. Бертран Мейер. Объектно-ориентированное конструирование программных систем 2005 года издания

Приложение 1. Исходные тексты программы.

Date:

```
#ifndef Date_hpp
#define Date_hpp
struct Date {
    static const char sep = '.';
    int day; int month; int year;
    Date() {}
    Date(int day, int month, int year) {
        this -> day = day;
       this -> month = month;
        this -> year = year;
    }
    Date (string text) {
        vector<string> components = ExtensionString::componentsSeparatedBy(text, sep);
        try {
            this -> day = stoi(components[0]);
            this -> month = stoi(components[1]);
            this -> year = stoi(components[2]);
            if (day > 31 || day < 0) throw DateError();
            if (month > 12 || month < 0) throw DateError();
            if (year > 9999 || year < 0) throw DateError();
        }
        catch (...) {
            throw DateError(text);
        }
    }
    Date& operator = (const Date &second) {
        day = second.day;
        month = second.month;
        year = second.year;
        return *this;
    }
    const bool operator == ( Date two) const {
        return (day == two.day && month == two.month && year == two.year);
    }
    const bool operator >= (Date two) const {
        if (year == two.year) {
            if (month == two.month) {
                return day >= two.day;
            } else {
                return month >= two.month;
            }
        }
        return year >= two.year;
    }
    const bool operator <= (Date two) const {
        if (year == two.year) {
            if (month == two.month) {
```

```
            return day <= two.day;
        } else {
            return month <= two.month;
        }
    }
    return year <= two.year;
}
const bool operator > (Date two) const {
    if (year == two.year) {
        if (month == two.month) {
            return day > two.day;
        } else {
            return month > two.month;
        }
    }
    return year > two.year;
}
const bool operator < (Date two) const {
    if (year == two.year) {
        if (month == two.month) {
            return day < two.day;
        } else {
            return month < two.month;
        }
    }
    return year < two.year;
}
string description() const {
    string s;
    s += ExtensionString::leading(day, 2) + sep;
    s += ExtensionString::leading(month, 2) + sep;
    s += ExtensionString::leading(year, 4);
    return s;
}
};
ostream& operator << (ostream &out, const Date& date) {
    out << ExtensionString::leading(date.day, 2) + Date::sep;
    out << ExtensionString::leading(date.month, 2) + Date::sep;
    out << ExtensionString::leading(date.year, 4);
    return out;
}
#endif /* Date_hpp */
```

Children:

```
#ifndef Children_hpp
#define Children_hpp
enum Sex {
    m, g, no
};
class SexCast {
public:
    static string toString(Sex sex) {
        map<Sex, string> dict = {
            {m, "M"}, {g, "G"}, {no, "0"}
        };
        return dict[sex];
    }
    static string toStringRus(Sex sex) {
        map<Sex, string> dict = {
            {m, "м"}, {g, "ж"}, {no, "0"}
        };
        return dict[sex];
    }
    static Sex toSexEnum(string sex) {
        map<string, Sex> dict = {
            {"м", m}, {"ж", g}, {"0", no}
        };
        return dict[sex];
    }
};
class Children {
    Sex child[3] = {no, no, no};
    int count = 0;
public:
    Children() {}
    void append(Sex newElement) { child[count++] = newElement; }
    const static bool attribute(Attribute attribute, Children children)  {
        switch (attribute) {
            case general: return true;
            case boys: return children.isAttribute(m);
            case girls: return children.isAttribute(g);
            case multiple: return children.isMultiple();
        }
    }
    const Sex operator [] (int index) const { return child[index]; }
    Children& operator = (const Children &second) {
        count = second.count;
        for (int i = 0; i < count; i++)
            child[i] = second.child[i];
        return *this;
    }
private:
    const bool isAttribute(Sex sex) const {
        bool flag = true;
        for (int i=0; i<count; i++)
```

```
            if (child[i] != no)
                flag = child[i] == sex && flag;
        return flag;
    }
    const bool isMultiple() const { return count >= 2;}
};
#endif /* Children_hpp */
```

Birth:

```
#ifndef Birth_hpp
#define Birth_hpp
struct Birth {
    int number;
    Date dOB;
    string region;
    string fIO;
    Date dOBMother;
    Children children;
    Birth() {}
    Birth (int number, Date dOB, string region, string fIO, Date dOBMother, Children children) {
        this -> number = number;
        this -> dOB = dOB;
        this -> region = region;
        this -> fIO = fIO;
        this -> dOBMother = dOBMother;
        this -> children = children;
    }
    Birth(string fIO, Date data) {
        this -> fIO = fIO;
        this -> dOBMother = data;
    }
    Birth& operator = (const Birth &second) {
        number = second.number;
        dOB = second.dOB;
        region = second.region;
        fIO = second.fIO;
        dOBMother = second.dOBMother;
        children = second.children;
        return *this;
    }
    bool operator != (Birth second) const {
            return !(fIO == second.fIO && dOBMother == second.dOBMother);
    }
    bool operator == (Birth second) const {
            return (fIO == second.fIO && dOBMother == second.dOBMother);
    }
    bool operator > (Birth second) const { return dOB > second.dOB; }
    bool operator >= (Birth second) const { return dOB >= second.dOB; }
    bool operator < (Birth second) const { return dOB < second.dOB; }
    bool operator <= (Birth second) const { return dOB <= second.dOB; }
    vector<string> getDescription() {
        vector<string> description = {
            ExtensionString::leading(number, 2, ' '), dOB.description(),
            region, fIO, dOBMother.description(), SexCast::toString(children[0]),
            SexCast::toString(children[1]), SexCast::toString(children[2])
        };
        return description;
    }
};
ifstream& operator >> (ifstream &in, Birth &birth) {
```

```
    string line;
    char sep = '|';
    getline(in, line, sep);
    birth.number = stoi(line);
    getline(in, line, sep);
    birth.dOB = Date(line);
    getline(in, birth.region, sep);
    getline(in, birth.fIO, sep);
    getline(in, line, sep);
    birth.dOBMother = Date(line);
    Children children;
    for (int i = 0; i < 2; i++) {
        getline(in, line, sep);
        children.append(SexCast::toSexEnum(line));
    }
    getline(in, line);
    children.append(SexCast::toSexEnum(line));
    birth.children = children;
    return in;
}
ostream& operator << (ostream &out, const Birth& birth) {
    out
    << birth.number << "|"
    << birth.dOB << "|"
    << birth.region << "|"
    << birth.fIO << "|"
    << birth.dOBMother << "|"
    << SexCast::toStringRus(birth.children[0]) << "|"
    << SexCast::toStringRus(birth.children[1]) << "|"
    << SexCast::toStringRus(birth.children[2]);
    return out;
}
#endif /* ModelBirth_hpp */
```

City:

```
#ifndef City_hpp
#define City_hpp
using DictionaryRegion = map<string, vector<int>>;
class City {
    string name;
    DictionaryRegion regions = {};
public:
    City () {}
    const string getName() const { return name; }
    void setName(string name) { this -> name = name; }
    void append(DictionaryRegion region) {
        DictionaryRegion::iterator it = region.begin();
        regions[it->first] = it->second;
    }
    const vector<int> find(string region) { return regions[region]; }
    long int getCount() const { return regions.size(); }
    vector<int> getNumbers(Area area, string areaText = "") {
        switch (area) {
            case city:
                if (areaText == name) { return allNumbers();}
            case region:
                return regions[areaText];
            case hospital:
                return vector<int>(1, stoi(areaText));
        }
    }
    vector<string> getAllRegionText() {
        vector<string> name;
        DictionaryRegion::iterator reg;
        for (reg = regions.begin(); reg != regions.end(); reg++)
            name.push_back(reg -> first);
        return name;
    }
```

```
    vector<string> getAllNumberText() {
        vector<string> name;
        DictionaryRegion::iterator reg;
        for (reg = regions.begin(); reg != regions.end(); reg++) {
            string text = "";
            for (int i : reg -> second)
                text += to_string(i) + ", ";
            name.push_back(text);
        }
        return name;
    }
private:
    vector<int> allNumbers() {
        vector<int> numbers;
        DictionaryRegion::iterator it = regions.begin();
        for (int i = 0; it != regions.end(); it++, i++)
            numbers.insert(numbers.end(), (it -> second).begin(), (it -> second).end());
        return numbers;
    }
};
ifstream& operator >> (ifstream &in, DictionaryRegion &region) {
    string name;
    getline(in, name, '|');
    string text;
    getline(in, text);
    vector<string> numbersText = ExtensionString::componentsSeparatedBy(text, ',');
    vector<int> numbers;
    for (int i = 0; i<numbersText.size(); i++)
        numbers.push_back(stoi(numbersText[i]));
    region[name] = numbers;
    return in;
}
#endif /* City_hpp */
```

DataModel:

```
#ifndef DataModel_hpp
#define DataModel_hpp
struct DataModel {
    ChoiceProcessing choiceProcessing;
    Area area;
    DataFormat dataFormat;
    Attribute attribute;
    string flOInput;
    vector<Date> dataInput;
    string areaText;
};
#endif /* DataModel_hpp */
```

ViewText:

```
#ifndef ViewText_hpp
#define ViewText_hpp
class ViewText {
public:
    DataModel dataModel;
    ViewText(const DataModel &dataModel) {
        this -> dataModel = dataModel;
    }
    virtual void output(ostream &out) = 0;
};
class CastAttribute {
public:
    static string attributeText(Attribute attribute) {
        switch (attribute) {
            case general:
                return "Общая рождаемость.";
            case boys:
                return "Только мальчики.";
            case girls:
                return "Только девочки.";
            case multiple:
                return "Многодетные родители.";
        }
    }
};
#include "TableViewText.hpp"
#include "HistogramViewText.hpp"
#include "BirthrateViewText.hpp"
#include "DeleteViewText.hpp"
#endif /* ViewText_hpp */
```

Processing:

```
#ifndef Processing_hpp
#define Processing_hpp
class Processing {
public:
    DataModel dataModel;
    Processing() {}
    Processing(DataModel dataModel) { this -> dataModel = dataModel;}
    virtual void processing(const Birth&) = 0;
    virtual void processingEnd() = 0;
};
class ViewBirthProcessing: public Processing {
public:
    vector<Birth> birthData;
    ViewBirthProcessing() {}
    ViewBirthProcessing(DataModel dataModel) : Processing(dataModel) {}
    void processing(const Birth &birth) override {
        switch (dataModel.dataFormat) {
            case day:
                if (birth.dOB == dataModel.dataInput[0]) birthData.push_back(birth);
            case interval:
                if (birth.dOB >= dataModel.dataInput[0] && birth.dOB <= dataModel.dataInput[1])
                    birthData.push_back(birth);
        }
    }
    void processingEnd() override {
        sort(birthData.begin(), birthData.end());
    }
};
class HistogramProcessing: public Processing {
public:
    int mouth[12] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
    HistogramProcessing() {}
    HistogramProcessing(DataModel dataModel) : Processing(dataModel) {}
    void processing(const Birth &birth) override {
        if ( !Children::attribute(dataModel.attribute, birth.children) ) { return; }
        mouth[birth.dOB.month -1] += 1;
    }
    void processingEnd() override {}
};
class BirthrateProcessing: public Processing {
public:
    int min[2] = {0, 0};
    int max[2] = {0, 0};
    vector<int> mouth = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
    BirthrateProcessing() {}
    BirthrateProcessing(DataModel dataModel) : Processing(dataModel) {}
    void processing(const Birth &birth) override {
        if (!Children::attribute(dataModel.attribute, birth.children)) { return; }
        mouth[birth.dOB.month -1] += 1;
    }
    void processingEnd() override {
```

```
        max[1] = int(distance(mouth.begin(), max_element(mouth.begin(), mouth.end())));
        max[0] = mouth[max[1]];
        min[1] = int(distance(mouth.begin(), min_element(mouth.begin(), mouth.end())));
        min[0] = mouth[min[1]];
    }
};
class DeleteProcessing: public Processing {
public:
    Birth birth;
    DeleteProcessing() {};
    bool isEmpty = false;
    DeleteProcessing(DataModel dataModel) : Processing(dataModel) {}
    void processing(const Birth &birth) override {
        if (dataModel.fIOInput == birth.fIO && dataModel.dataInput[0] == birth.dOBMother) {
            this -> birth = birth;
            isEmpty = true;
        }
    }
    void processingEnd() override {}
};
#endif /* Processing_hpp */
```

Table:

```
#ifndef TableViewText_hpp
#define TableViewText_hpp
class Table {
public:
    const static int column = 8;
    enum Align { Left, Middle };
    Align align[column] = {
        Middle, Middle, Left, Left,
        Middle, Middle, Middle, Middle
    };
    int getLayoutWidth(int column) {
        return (int)width*layout[column];
    }
    int width = 100;
    char border[3] = { '-','|','+'};
private:
    float layout[column] = {
        0.04, 0.12, 0.15, 0.27,
        0.12, 0.07, 0.07, 0.07
    };
};
#include "TableViewTextCell.hpp"
class TableViewText: public ViewText, public Table {
    ViewBirthProcessing *processing;
    string headerText[2][column] = {
        {"N", "Date of bi-", "Region", "FIO", "Date of bi-", "Sex 1", "Sex 2", "Sex 3" },
        {"", "rth child", "", "mothers", "rth mothers", "chi.", "chi.", "chi."}
    };
public:
    TableViewText (ViewBirthProcessing *processing, const DataModel &dataModel) :
ViewText(dataModel) {
        this -> processing = processing;
    }
    void output (ostream &out) override {
        if (processing->birthData.size() == 0) {
            out << "Нет данных." << endl;
        } else {
            tableHeaderViewText(out);
            for (Birth i : processing->birthData) { TableViewTextCell cell = {out, i}; }
            tableFooterViewText(out);
        }
    }
private:
    void tableHeaderViewText (ostream &out) {
        line(out);
        lineHeader(out, headerText[0]);
        lineHeader(out, headerText[1]);
        line(out);
    }
    void tableFooterViewText (ostream &out) {
        line(out);
```

```
            out << endl;
        }
        void lineHeader(ostream &out, string text[column]) {
            for (int i=0; i<column; i++) {
                out << border[1] << ' ' << left << setw(getLayoutWidth(i)-1) << text[i];
            }
            out << border[1] << endl;
        }
        void line(ostream &out) {
            for (int i=0; i<column; i++) {
                out << border[1] << string(getLayoutWidth(i), border[0]);
            }
            out << border[1] << endl;
        }
    };
    #endif /* TableViewText_hpp */
```

TabelViewTextCell:

```
    #ifndef TabelViewTextCell_hpp
    #define TabelViewTextCell_hpp
    struct TableViewTextCell: public Table {
        TableViewTextCell(ostream &out, Birth birth) {
            vector<string> description = birth.getDescription();
            for (int i=0; i<description.size(); i++) {
                out << border[1] << left << setw(getLayoutWidth(i)) <<
(widthSpace(description[i].size(), i, align[i]) + description[i]);
            }
            out << border[1]  << endl;
        }
    private:
        string widthSpace(long int sizeText, int column, Align align) {
            int widthLayout = getLayoutWidth(column);
            if (sizeText >= widthLayout) return "";
            if (align == Left) return " ";
            long int middle = (widthLayout - sizeText) / 2;
            return string(middle, ' ');
        }
    };
    #endif /* TabelViewTextCell_hpp */
```

HistogramViewText:

```
#ifndef HistogramViewText_hpp
#define HistogramViewText_hpp
class HistogramViewText: public ViewText {
    HistogramProcessing histogram;
    string bricks ="#";
public:
    HistogramViewText(const HistogramProcessing &histogram, const DataModel
&dataModel) : ViewText(dataModel) {
        this -> histogram = histogram;
    }
    void output(ostream &out) override {
        int max = 0;
        for (int i = 0; i<12; i++)
            if (max < histogram.mouth[i]) max = histogram.mouth[i];
        out << "Вывод гистрограммы рождаемости: " <<
CastAttribute::attributeText(dataModel.attribute) << endl;
        out << "По городу: " << dataModel.area << endl;
        out << "     |------------------------|" << endl;
        for (int i=0; i < (max + 1); i++) {
            int count = (max + 1) - i;
            bool *avail = availability(count, histogram.mouth);
            out << setw(4) << count << " | " << histogramString(count, avail) << "|" << endl;
            delete [] avail;
        }
        out << "     |------------------------|" << endl;
        out << "     | я ф м а м и и а с о н д |" << endl;
    }
private:
    string histogramString(int count, bool availability[12]) {
        string s;
        for (int i=0; i<12; i++) {
            s += availability[i] ? bricks : " ";
            s += " ";
        }
        return s;
    }
    bool* availability(int count, const int mouth[12]) {
        bool *array = new bool[13];
        for (int i=0; i<12; i++)
            array[i] = count <= mouth[i];
        return array;
    }
};
#endif /* HistogramViewText_hpp */
```

BirthrateViewText:

```
#ifndef BirthrateViewText_hpp
#define BirthrateViewText_hpp
class BirthrateViewText: public ViewText {
    BirthrateProcessing birthrate;
    string textData[12] = {
        "Январь", "Февраль", "Март", "Апрель", "Май", "Июнь",
        "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"
    };
public:
    BirthrateViewText(const BirthrateProcessing &birthrate, const DataModel &dataModel) :
ViewText(dataModel) {
        this -> birthrate = birthrate;
    }
    void output(ostream &out) override {
        out << endl;
        if (birthrate.min[0] == birthrate.max[0]) {
            out <<
            "Статистика по городу: " << dataModel.areaText + "." << endl <<
            "Искать по: " << CastAttribute::attributeText(dataModel.attribute) << endl <<
            "В каждом месяцы было одинаковое кол-во рождений." << endl <<
            "Что составило: " << max(birthrate.min, birthrate.max) << endl;
        } else {
            out <<
            "Статистика по городу: " << dataModel.areaText + "." << endl <<
            "Искать по: " << CastAttribute::attributeText(dataModel.attribute) << endl <<
            "Максимальное кол-во рождений в " << textData[birthrate.max[1]] << endl
            << "Что составило: " << birthrate.max[0] << endl;
            out << "Минимальное кол-во рождений: " << textData[birthrate.min[1]] << endl
            << "Что составило: " << birthrate.min[0] << endl;
        }
    }
};
#endif /* BirthrateViewText_hpp */
```

DeleteViewText:

```
#ifndef DeleteViewText_hpp
#define DeleteViewText_hpp
class DeleteViewText: public ViewText {
public:
    DeleteProcessing processing;
    DeleteViewText(const DeleteProcessing &processing, const DataModel &dataModel) :
ViewText(dataModel) {
        this -> processing = processing;
    }
    void output(ostream &out) override {
        if (processing.isEmpty) {
            out << "Удален: " << dataModel.flOInput << ". Дата рождения: " <<
dataModel.dataInput[0] << endl;
        } else {
            out << "Мать не найдена." << endl;
        }
    }
};
#endif /* DeleteViewText_hpp */
```

Menu:

```
#ifndef Menu_hpp
#define Menu_hpp
#include "Input.hpp"
class Menu {
public:
    void open(DataModel &dataModel, City &city) {
        choiceProcessingMenu(dataModel);
        if (dataModel.choiceProcessing != removeBirth) menuArea(dataModel, city);
        switch (dataModel.choiceProcessing) {
            case viewBirth:
                dateFormatMenu(dataModel);
                break;
            case histogram: case birthrate:
                menuAttributeBirthrate(dataModel);
                break;
            case removeBirth:
                menuDelete(dataModel);
                break;
        }
    }
    void clearTab() { system("cls"); }
private:
    void choiceProcessingMenu(DataModel &dataModel) {
        output(textCP);
        choiceProcessingInput(dataModel);
        clearTab();
    }
    void dateFormatMenu(DataModel &dataModel) {
        output(textDataFormat);
        dateFormatInput(dataModel);
        clearTab();
        output(textDataFormatInput[dataModel.dataFormat]);
        dateInput(dataModel);
        clearTab();
    }
    void menuArea(DataModel &dataModel, City &city) {
        output(textArea);
        areaInput( dataModel);
        clearTab();
        outCity(dataModel.area, city);
        cout << endl << textAreaInput[dataModel.area];
        areaTextInput(dataModel);
        clearTab();
    }
    void menuAttributeBirthrate(DataModel &dataModel) {
        output(textBirthrate);
        attributeInput(dataModel);
        clearTab();
    }
    void menuDelete(DataModel &dataModel) {
        output(textDeletInput[0]);
```

```
    dataModel.flOInput = Input::text();
    output(textDeletInput[1]);
    dataModel.dataFormat = day;
    dateInput(dataModel);
    clearTab();
}
void dateInput (DataModel &dataModel) {
    try {
        dataModel.dataInput = Input::dateCast(Input::text(), dataModel.dataFormat);
    }
    catch (DateError error) {
        cout << error.what();
        dateInput(dataModel);
    }
}
void dateFormatInput (DataModel &dataModel) {
    try {
        dataModel.dataFormat = Input::dataFormatCast(Input::number());
    }
    catch (InputError error) {
        cout << error.what();
        dateFormatInput(dataModel);
    }
}
void choiceProcessingInput (DataModel &dataModel) {
    try {
        dataModel.choiceProcessing = Input::choiceProcessingCast(Input::number());
    }
    catch (InputError error) {
        cout << error.what();
        choiceProcessingInput(dataModel);
    }
}
void areaInput(DataModel &dataModel) {
    try {
        dataModel.area = Input::areaCast(Input::number());
    }
    catch (InputError error) {
        cout << error.what();
        areaInput(dataModel);
    }
}
void areaTextInput (DataModel &dataModel) {
    try {
        dataModel.areaText = Input::text();
    }
    catch (InputError error) {
        cout << error.what();
        areaTextInput(dataModel);
    }
}
void attributeInput (DataModel &dataModel) {
    try {
```

```
            dataModel.attribute = Input::attributeCast(Input::number());
        }
        catch (InputError error) {
            cout << error.what();
            attributeInput(dataModel);
        }
    }
    void outCity (Area area, City &city) {
        switch (area) {
            case Area::city:
                cout << city.getName();
                break;
            case Area::region:
                output(city.getAllRegionText());
                break;
            case Area::hospital:
                output(city.getAllNumberText());
                break;
        }
    }
    void output(vector<string> array) {
        for (int i = 0; i < array.size(); i++) {
            cout << array[i];
            if (!(i == array.size()-1))
                cout << endl;
        }
    }
    void output (string text) { cout << text; }
    const vector<string> textCP = {
        "Выберите обработку:",
        "1. Просмотр данных по времени.",
        "2. Вывод гистограммы рождаемости по месяцам года и кривой  рождаемости за
год.",
        "3. Определение месяцев максимальной и минимальной рождаемости.",
        "4. Удаление записей о родах.",
        "Введите число от 1 до 4: "
    };
    const vector<string> textArea = {
        "Искать:",
        "1. По роддому.",
        "2. По району.",
        "3. По городу.",
        "Введите число от 1 до 3: "
    };
    const vector<string> textDataFormat = {
        "Выберите день или интервал:",
        "1. День.",
        "2. Интервал.",
        "Введите число 1 или 2: "
    };
    const vector<string> textBirthrate = {
        "Критерии поиска:",
        "1. Общей рождаемости",
```

```
        "2. Только мальчиков",
        "3. Только девочек",
        "4. Многодетных родов ( более одного ребенка)",
        "Введите число от 1 до 4: "
    };
    const vector<string> textDataFormatInput = {
        "Введите день в формате 'дд.мм.гггг': ",
        "Введите интервал в формате 'дд.мм.гггг - дд.мм.гггг': "
    };
    const vector<string> textAreaInput = {
        "Введите роддом: ",
        "Введите район: ",
        "Введите город: "
    };
    const vector<string> textDeletInput = {
        "Введите ФИО матери: ",
        "Введите дату рождения матери: "
    };
};
#endif /* Menu_hpp */
```

Input:

```
#ifndef Input_hpp
#define Input_hpp
class Input {
public:
    static int number() {
        string textNumber = text();
        int number;
        number = stoi(textNumber);
        return number;
    }
    static string text() {
        string text;
        getline(cin, text);
        return text;
    }
    static const vector<Date> dateCast (string dataText, DataFormat format) {
        vector<Date> data(format+1);
        switch (format) {
            case day:
                data[0] = Date(dataText);
                break;
            case interval:
              vector<string>components=ExtensionString::componentsSeparatedBy(dataText, '-');
                for (int i=0; i < format + 1 ; i++) data[i] = Date(components[i]);
                break;
        }
        return data;
    }
    static ChoiceProcessing choiceProcessingCast(int number) {
        if (number > 4 || number < 1) { throw InputError("choiceProcessing"); }
        return static_cast<ChoiceProcessing>(number-1);
    }
    static Area areaCast(int number) {
        if (number > 3 || number < 1) { throw InputError("areaCast"); }
        return static_cast<Area>(number-1);
    }
    static DataFormat dataFormatCast(int number) {
        if (number > 2 || number < 1) { throw InputError("dataFormatCast"); }
        return static_cast<DataFormat>(number-1);
    }
    static Attribute attributeCast(int number) {
        if (number > 4 || number < 1) { throw InputError("attributeCast"); }
        return static_cast<Attribute>(number-1);
    }
};
#endif /* Input_hpp */
```

main:

```
#include <iostream>
#include <exception>
#include <fstream>
#include <map>
#include <string>
#include <vector>
using namespace std;
#include "ClassError.hpp"
#include "ExtensionString.hpp"
#include "ChoiceMenu.hpp"
#include "Date.hpp"
#include "Children.hpp"
#include "Birth.hpp"
#include "DataModel.hpp"
#include "City.hpp"
#include "Processing.hpp"
#include "Menu.hpp"
#include "ViewText.hpp"
#include "FileProcessing.hpp"
int main() {
    FileProcessing file;
    City city;
    Menu menu;
    DataModel dataModel;
    string is = "1";
    try {
        file.initCity(city);
        file.initHospital();
        while (is != "0") {
            menu.open(dataModel, city);
            switch (dataModel.choiceProcessing) {
                case viewBirth: {
                    ViewBirthProcessing processing = { dataModel };
                    vector<int> numbers = city.getNumbers(dataModel.area, dataModel.areaText);
                    file.processing(processing, numbers);
                    TableViewText view = {&processing, dataModel};
                    view.output(cout);
                    file.output(view);
                    break;
                }
                case histogram: {
                    HistogramProcessing processing = { dataModel };
                    vector<int> numbers = city.getNumbers(dataModel.area, dataModel.areaText);
                    file.processing(processing, numbers);
                    HistogramViewText view = { processing, dataModel };
                    file.output(view);
                    view.output(cout);
                    break;
                }
```

```
            case birthrate: {
                BirthrateProcessing processing = { dataModel };
                vector<int> numbers = city.getNumbers(dataModel.area, dataModel.areaText);
                file.processing(processing, numbers);
                BirthrateViewText view = { processing, dataModel };
                file.output(view);
                view.output(cout);
                break;
            }
            case removeBirth: {
                DeleteProcessing processing = { dataModel };
                vector<int> numbers = city.getNumbers(Area::city);
                file.processing(processing, numbers);
                file.removeBirth(processing);
                DeleteViewText view = {processing, dataModel };
                file.output(view);
                view.output(cout);
                break;
            }
        }
        cout << "Продолжить? 1 0 : ";
        getline(cin, is);
        menu.clearTab();
    }
  }
  catch(OpenFileError error) {
    cout << error.text << endl;
  }
  catch(...) {
    cout << endl << "Неизвестная ошибка!" << endl;
  }
  file.removeHospitalFile(city);
  return 0;
}
```

FileProcessing:

```
#ifndef FileProces_hpp
#define FileProces_hpp
fstream file;
ifstream read;
ofstream write;
class FileProcessing {
    string cityFileName = «Москва»; string birthFileName = "Birth";
    string resultFileName =  «Protocol»; string hospitalFileName = "Hospital";
    string getFileName(string name) const { return name + ".txt"; }
    string getFilePatch(string name) const { return name + "/"; }
public:
    void initCity(City &city) const {
        city.setName(cityFileName);
        openRead(cityFileName);
        while (!read.eof()) {
            DictionaryRegion dictionaryRegion;
            read >> dictionaryRegion;
            city.append(dictionaryRegion);
        }
        read.close();
    }
    void initHospital() {
        openRead(birthFileName);
        Birth data;
        while (!read.eof()) {
            read >> data;
            openWrite(getFilePatch(hospitalFileName) + to_string(data.number), ios::binary |
ios::app);
            write.write((char*)&data, sizeof(Birth));
            write.close();
        }
        read.close();
    }
    void processing (Processing &processing, const vector<int> &numbers) {
        Birth data;
        for (int i = 0; i < numbers.size(); i++) {
            if (openFile(getFilePatch(hospitalFileName) + to_string(numbers[i]), ios::in)) {
                while (file.read((char*)&data, sizeof(Birth)))
                    processing.processing(data);
                file.close();
                processing.processingEnd();
            }
        }
    }
    void output(ViewText &viewText) {
        openWrite(resultFileName, ios::app);
        viewText.output(write);
        write.close();
    }
    void removeBirth(const DeleteProcessing &processing) {
        if (!processing.isEmpty) return;
```

```
            string tmpFileName = getFilePatch(hospitalFileName)+ "tmp";
            string oldFileName = getFilePatch(hospitalFileName) +
to_string(processing.birth.number);
            openWrite(tmpFileName, ios::binary);
            openRead(oldFileName);
            Birth data;
            while (read.read((char*)&data, sizeof(Birth)))
               if (processing.birth != data)
                  write.write((char*)&data, sizeof(Birth));
            read.close();
            write.close();
            remove(getFileName(oldFileName).c_str());
            rename(getFileName(tmpFileName).c_str(), getFileName(oldFileName).c_str());
         }
         void removeHospitalFile (City &city) {
            const vector<int> n = city.getNumbers(Area::city, cityFileName);
            for (int i = 0; i<n.size(); i++)
               remove(getFileName(getFilePatch(hospitalFileName) + to_string(n[i])).c_str());
         }
      private:
         void openRead(string resource, ios_base::openmode __mode = ios::in) const {
            read.open(getFileName(resource), __mode);
            if (!read.is_open()) { throw ("Ошибка открытия файла: " + resource); }
         }
         void openWrite(string resource, ios_base::openmode __mode = ios::out) const {
            write.open(getFileName(resource), __mode);
            if (!write.is_open()) { throw ("Ошибка открытия файла: " + resource); }
         }
         bool openFile(string resource, ios_base::openmode __mode) const {
            file.open(getFileName(resource), __mode);
            return file.is_open();
         }
      };
      #endif /* FileProcess_hpp */
```

ExtensionString

```
#ifndef ExtensionString_hpp
#define ExtensionString_hpp
class ExtensionString {
public:
    static vector<string> componentsSeparatedBy(string text, char separatedBy) {
        vector<string> components;
        string words;
        for (int i = 0; i < text.size(); i++) {
            if (text[i] != separatedBy) {
                words += text[i];
            } else {
                components.push_back(words);
                words = "";
            }
        }
        components.push_back(words);
        return components;
    }
    static string leading (int number, int count, char leadingChar = '0') {
        string text = to_string(number);
        text = string(count-text.size(), leadingChar) + text;
        return text;
    }
};
#endif /* ExtensionString_hpp */
```

ChoiceProcessing

```
#ifndef ChoiceMenu_hpp
#define ChoiceMenu_hpp
enum ChoiceProcessing {
    viewBirth,
    histogram,
    birthrate,
    removeBirth,
};
enum Area {
    hospital,
    region,
    city,
};
enum DataFormat {
    day, interval,
};
enum Attribute {
    general, boys,
    girls, multiple,
};
#endif /* ChoiceMenu_hpp */
```

Error:

```
#ifndef ClassError_hpp
#define ClassError_hpp
class Error {
public:
    string text;
    Error() {}
    Error(string text) {
        this -> text = text;
    }
    virtual const string what() const = 0;
};
class OpenFileError: public Error {
    OpenFileError(string text) : Error(text) {}
    const string what() const override {
        return "Ошибка открытия файла: " + text;
    }
};
class DateError: public Error {
public:
    DateError() {}
    DateError(string text) : Error(text) {}
    const string what() const override {
        return "Ошибка ввода даты: " + text + ". Введите еще раз: ";
    }
};
class InputError : public Error {
public:
    InputError(string text) : Error(text) {}
    const string what() const override {
        return "Ошибка ввода: " + text + ". Введите еще раз: ";
    }
};
#endif /* ClassError_hpp */
```

Приложение 2. Содержание файла с данными для тестирования.

15|25.3.3816|Yugo-Vos|Sarah Knight|27.7.3911|ж|м|0
8|29.7.8144|Severo-Vos|Clara Thomas|29.8.8221|ж|ж|0
13|5.1.8346|Vostochny|Nicole Silva|13.4.8414|ж|м|ж
9|2.10.7628|Severo-Vos|Christy Jones|12.4.7652|ж|м|0
9|28.7.7929|Severo-Vos|Jennifer Maldonado|1.7.7984|ж|ж|м
4|20.5.7419|Northern|Katherine Hill|20.10.7508|м|ж|0
1|28.10.7929|Central|Jennifer Hill|2.5.2084|ж|ж|м
13|2.11.5112|Vostochny|Tina Ross|30.8.5139|ж|0|0
13|24.11.82|Vostochny|Janet Johnson|15.5.136|ж|м|м
10|20.5.6348|Severo-Vos|Marcia Brooks|6.10.6432|м|м|0
5|8.6.8238|Northern|Sarah Klein|5.7.8255|м|0|0
7|6.7.1565|Severo-Vos|Frances Dixon|6.9.1597|м|0|0
8|19.2.6309|Severo-Vos|Nellie Simpson|29.11.6368|ж|ж|м
6|21.1.5631|Northern|Denise Sanchez|30.12.5705|ж|ж|ж
10|13.11.8907|Severo-Vos|Janet Johnson|27.10.8909|м|м|ж
14|30.9.8935|Yugo-Vos|Tina Ross|28.6.8976|ж|ж|ж
10|30.9.5920|Severo-Vos|Angela Wheeler|28.11.6003|ж|ж|м
11|1.11.2978|Vostochny|Jennifer Goodwin|28.1.3028|м|м|ж
4|13.4.3783|Northern|Laura Snyder|26.8.3831|ж|м|ж
17|29.6.4122|Yugo-Vos|Debbie Flowers|18.3.4168|м|м|0
13|18.10.4657|Vostochny|Erika Perez|10.3.4660|ж|ж|м
4|18.1.9961|Northern|Marcia Bates|28.5.9989|м|м|0
2|6.11.1565|Central|Frances Dixon|6.9.1597|м|0|0
13|6.4.5763|Vostochny|Melanie Phillips|20.9.5802|м|0|0
17|29.2.4210|Yugo-Vos|Jennifer Maldonado|12.10.4255|ж|ж|0
6|6.5.4290|Northern|Dora Perez|20.6.4381|ж|ж|м
15|30.8.7315|Yugo-Vos|Barbara West|24.12.7347|м|м|0
6|15.11.1098|Northern|Mary Garner|20.8.1183|м|ж|ж
7|14.12.3739|Severo-Vos|Barbara West|6.7.3821|ж|м|м
6|2.6.7780|Northern|Susan Phillips|11.12.7838|м|0|0
13|8.12.7943|Vostochny|Dora Perez|22.10.8013|ж|м|м
8|6.4.7744|Severo-Vos|Anne Ruiz|9.1.7753|м|ж|0
9|17.11.9867|Severo-Vos|Jennifer Holt|18.11.9876|м|м|ж
4|4.6.1145|Northern|Kathleen Adams|27.8.1242|ж|ж|ж
16|17.11.3083|Yugo-Vos|Rosemary Nelson|9.2.3103|м|ж|0
17|28.8.5080|Yugo-Vos|Tina Ross|12.2.5138|ж|ж|ж
14|28.3.7198|Yugo-Vos|Florence Fowler|30.7.7226|м|м|0
12|25.8.4835|Vostochny|Rose Hicks|5.7.4879|ж|ж|0
12|8.11.982|Vostochny|Grace Sanders|14.9.1014|ж|ж|ж
5|7.9.6079|Northern|Marie Robertson|13.11.6160|ж|ж|ж
9|22.12.8836|Severo-Vos|Charlene Allen|28.1.8842|м|ж|м
7|20.2.8279|Severo-Vos|Stella Taylor|23.3.8313|ж|ж|м
4|3.2.6621|Northern|Susan Phillips|18.2.6647|м|ж|ж
14|6.8.8224|Yugo-Vos|Angela Wheeler|6.11.8250|ж|м|ж
3|6.11.1565|Central|Frances Dixon|13.2.1999|м|0|0
10|30.12.4644|Severo-Vos|Teresa Mason|25.9.4713|ж|0|0
9|6.8.3712|Severo-Vos|Dorothy Brown|25.5.3803|ж|ж|м
6|1.4.1817|Northern|Grace Sanders|7.7.1830|ж|ж|0

Приложение 3. Содержание файла с результатами тестирования.

```
По городу: Vostochny
|----|------------|---------------|--------------------------|------------|-------|-------|-------|
| N  | Date of bi-| Region        | FIO                      | Date of bi-| Sex 1 | Sex 2 | Sex 3 |
|    | rth child  |               | mothers                  | rth mothers| chi.  | chi.  | chi.  |
|----|------------|---------------|--------------------------|------------|-------|-------|-------|
| 11 | 01.11.2978 | Vostochny     | Jennifer Goodwin         | 28.01.3028 |   M   |   M   |   G   |
| 13 | 18.10.4657 | Vostochny     | Erika Perez              | 10.03.4660 |   G   |   G   |   M   |
| 12 | 25.08.4835 | Vostochny     | Rose Hicks               | 05.07.4879 |   G   |   G   |   0   |
| 13 | 02.11.5112 | Vostochny     | Tina Ross                | 30.08.5139 |   G   |   0   |   0   |
| 13 | 06.04.5763 | Vostochny     | Melanie Phillips         | 20.09.5802 |   M   |   0   |   0   |
| 13 | 08.12.7943 | Vostochny     | Dora Perez               | 22.10.8013 |   G   |   M   |   M   |
| 13 | 05.01.8346 | Vostochny     | Nicole Silva             | 13.04.8414 |   G   |   M   |   G   |
|----|------------|---------------|--------------------------|------------|-------|-------|-------|

Вывод гистограммы рождаемости: Только девочки.
По городу: 13
     |-------------------------|
2    |                         |
1    |                     #   |
     |-------------------------|
     | я ф м а м и и а с о н д |

Статистика по городу: Москва.
Искать по: Общая рождаемость.
Максимальное кол-во рождений в Ноябрь
Что составило: 10
Минимальное кол-во рождений: Март
Что составило: 2
Вывод гистограммы рождаемости: Общая рождаемость.
По городу: Москва
     |-------------------------|
  11 |                         |
  10 |                     #   |
   9 |                     #   |
   8 |                     #   |
   7 |                     #   |
   6 |                     #   |
   5 |                 #   #   |
   4 |   #   #   #   #     # # |
   3 | # #   # # # # # # # # # |
   2 | # # # # # # # # # # # # |
   1 | # # # # # # # # # # # # |
     |-------------------------|
     | я ф м а м и и а с о н д |
Удален: Rose Hicks. Дата рождения: 05.07.4879
Нет данных.
```

Приложение 4. Виды экрана с различными вариантами диалога пользователя.

```
Выберите обработку:
1. Просмотр данных по времени.
2. Вывод гистограммы рождаемости по месяцам года и кривой  рождаемости за год.
3. Определение месяцев максимальной и минимальной рождаемости.
4. Удаление записей о родах.
Введите число от 1 до 4: 1
Искать:
1. По роддому.
2. По району.
3. По городу.
Введите число от 1 до 3: 3
Москва
Введите город: Москва
Выберите день или интервал:
1. День.
2. Интервал.
Введите число 1 или 2: 2
Введите интервал в формате 'дд.мм.гггг - дд.мм.гггг': 01.01.0001 - 031.12.9999
```

N	Date of birth child	Region	FIO mothers	Date of birth mothers	Sex 1 chi.	Sex 2 chi.	Sex 3 chi.
13	24.11.0082	Vostochny	Janet Johnson	15.05.0136	G	M	M
12	08.11.0982	Vostochny	Grace Sanders	14.09.1014	G	G	G
6	15.11.1098	Northern	Mary Garner	20.08.1183	M	G	G
4	04.06.1145	Northern	Kathleen Adams	27.08.1242	G	G	G
7	06.07.1565	Severo-Vos	Frances Dixon	06.09.1597	M	0	0
2	06.11.1565	Central	Frances Dixon	06.09.1597	M	0	0
3	06.11.1565	Central	Frances Dixon	13.02.1999	M	0	0
6	01.04.1817	Northern	Grace Sanders	07.07.1830	G	G	0
11	01.11.2978	Vostochny	Jennifer Goodwin	28.01.3028	M	M	G
16	17.11.3083	Yugo-Vos	Rosemary Nelson	09.02.3103	M	G	0
9	06.08.3712	Severo-Vos	Dorothy Brown	25.05.3803	G	G	M
7	14.12.3739	Severo-Vos	Barbara West	06.07.3821	G	M	M
4	13.04.3783	Northern	Laura Snyder	26.08.3831	G	M	G
15	25.03.3816	Yugo-Vos	Sarah Knight	27.07.3911	G	M	0
17	29.06.4122	Yugo-Vos	Debbie Flowers	18.03.4168	M	M	0
17	29.02.4210	Yugo-Vos	Jennifer Maldonado	12.10.4255	G	G	0
6	06.05.4290	Northern	Dora Perez	20.06.4381	G	G	M
10	30.12.4644	Severo-Vos	Teresa Mason	25.09.4713	G	0	0
13	18.10.4657	Vostochny	Erika Perez	10.03.4660	G	G	M
12	25.08.4835	Vostochny	Rose Hicks	05.07.4879	G	G	0
17	28.08.5080	Yugo-Vos	Tina Ross	12.02.5138	G	G	G
13	02.11.5112	Vostochny	Tina Ross	30.08.5139	G	0	0
6	21.01.5631	Northern	Denise Sanchez	30.12.5705	G	G	G
13	06.04.5763	Vostochny	Melanie Phillips	20.09.5802	M	0	0
10	30.09.5920	Severo-Vos	Angela Wheeler	28.11.6003	G	G	M
5	07.09.6079	Northern	Marie Robertson	13.11.6160	G	G	G
8	19.02.6309	Severo-Vos	Nellie Simpson	29.11.6368	G	G	M
10	20.05.6348	Severo-Vos	Marcia Brooks	06.10.6432	M	M	0
4	03.02.6621	Northern	Susan Phillips	18.02.6647	M	G	G
14	28.03.7198	Yugo-Vos	Florence Fowler	30.07.7226	M	M	0
15	30.08.7315	Yugo-Vos	Barbara West	24.12.7347	M	M	0
4	20.05.7419	Northern	Katherine Hill	20.10.7508	M	G	0
9	02.10.7628	Severo-Vos	Christy Jones	12.04.7652	G	M	0
8	06.04.7744	Severo-Vos	Anne Ruiz	09.01.7753	M	G	0
6	02.06.7780	Northern	Susan Phillips	11.12.7838	M	0	0
9	28.07.7929	Severo-Vos	Jennifer Maldonado	01.07.7984	G	G	M
1	28.10.7929	Central	Jennifer Hill	02.05.2084	G	G	M
13	08.12.7943	Vostochny	Dora Perez	22.10.8013	G	M	M
8	29.07.8144	Severo-Vos	Clara Thomas	29.08.8221	G	G	0
14	06.08.8224	Yugo-Vos	Angela Wheeler	06.11.8250	G	M	G
5	08.06.8238	Northern	Sarah Klein	05.07.8255	M	0	0
7	20.02.8279	Severo-Vos	Stella Taylor	23.03.8313	G	G	M
13	05.01.8346	Vostochny	Nicole Silva	13.04.8414	G	M	G
9	22.12.8836	Severo-Vos	Charlene Allen	28.01.8842	M	G	M
10	13.11.8907	Severo-Vos	Janet Johnson	27.10.8909	M	M	G
14	30.09.8935	Yugo-Vos	Tina Ross	28.06.8976	G	G	G
9	17.11.9867	Severo-Vos	Jennifer Holt	18.11.9876	M	M	G
4	18.01.9961	Northern	Marcia Bates	28.05.9989	M	M	0

```
Продолжить? 1 0 :
```

```
Выберите обработку:
1. Просмотр данных по времени.
2. Вывод гистограммы рождаемости по месяцам года и кривой  рождаемости за год.
3. Определение месяцев максимальной и минимальной рождаемости.
4. Удаление записей о родах.
Введите число от 1 до 4: 1
Искать:
1. По роддому.
2. По району.
3. По городу.
Введите число от 1 до 3: 2
Central
Northern
Severo-Vos
Vostochny
Yugo-Vos
Введите район: Vostochny
Выберите день или интервал:
1. День.
2. Интервал.
Введите число 1 или 2: 2
Введите интервал в формате 'дд.мм.гггг - дд.мм.гггг': 13.34.1900 - 03.08.9999
Ошибка ввода даты: 13.34.1900 . Введите еще раз: 13.05.1900 - 09.08.9999
|----|------------|--------------|--------------------------|------------|-------|-------|-------|
| N  | Date of bi-| Region       | FIO                      | Date of bi-| Sex 1 | Sex 2 | Sex 3 |
|    | rth child  |              | mothers                  | rth mothers| chi.  | chi.  | chi.  |
|----|------------|--------------|--------------------------|------------|-------|-------|-------|
| 11 | 01.11.2978 | Vostochny    | Jennifer Goodwin         | 28.01.3028 |   M   |   M   |   G   |
| 13 | 18.10.4657 | Vostochny    | Erika Perez              | 10.03.4660 |   G   |   G   |   M   |
| 12 | 25.08.4835 | Vostochny    | Rose Hicks               | 05.07.4879 |   G   |   G   |   0   |
| 13 | 02.11.5112 | Vostochny    | Tina Ross                | 30.08.5139 |   G   |   0   |   0   |
| 13 | 06.04.5763 | Vostochny    | Melanie Phillips         | 20.09.5802 |   M   |   0   |   0   |
| 13 | 08.12.7943 | Vostochny    | Dora Perez               | 22.10.8013 |   G   |   M   |   M   |
| 13 | 05.01.8346 | Vostochny    | Nicole Silva             | 13.04.8414 |   G   |   M   |   G   |
|----|------------|--------------|--------------------------|------------|-------|-------|-------|

Продолжить? 1 0 :


Выберите обработку:
1. Просмотр данных по времени.
2. Вывод гистограммы рождаемости по месяцам года и кривой  рождаемости за год.
3. Определение месяцев максимальной и минимальной рождаемости.
4. Удаление записей о родах.
Введите число от 1 до 4: 3
Искать:
1. По роддому.
2. По району.
3. По городу.
Введите число от 1 до 3: 3
Москва
Введите город: Москва
Критерии поиска:
1. Общей рождаемости
2. Только мальчиков
3. Только девочек
4. Многодетных родов ( более одного ребенка)
Введите число от 1 до 4: 1

Статистика по городу: Москва.
Искать по: Общая рождаемость.
Максимальное кол-во рождений в Ноябрь
Что составило: 10
Минимальное кол-во рождений: Март
Что составило: 2
Продолжить? 1 0 :
```

```
Выберите обработку:
1. Просмотр данных по времени.
2. Вывод гистограммы рождаемости по месяцам года и кривой  рождаемости за год.
3. Определение месяцев максимальной и минимальной рождаемости.
4. Удаление записей о родах.
Введите число от 1 до 4: 2
Искать:
1. По роддому.
2. По району.
3. По городу.
Введите число от 1 до 3: 3
Москва
Введите город: Москва
Критерии поиска:
1. Общей рождаемости
2. Только мальчиков
3. Только девочек
4. Многодетных родов ( более одного ребенка)
Введите число от 1 до 4: 1
Вывод гистрограммы рождаемости: Общая рождаемость.
По городу: 2
     |------------------------|
11   |                        |
10   |                    #   |
9    |                    #   |
8    |                    #   |
7    |                    #   |
6    |                    #   |
5    |               #    #   |
4    |   #   #   #   #    # # |
3    | # #   # # # # # # # # # |
2    | # # # # # # # # # # # # |
1    | # # # # # # # # # # # # |
     |------------------------|
     | я ф м а м и и а с о н д |
Продолжить? 1 0 :
```

```
Выберите обработку:
1. Просмотр данных по времени.
2. Вывод гистограммы рождаемости по месяцам года и кривой  рождаемости за год.
3. Определение месяцев максимальной и минимальной рождаемости.
4. Удаление записей о родах.
Введите число от 1 до 4: 4
Введите ФИО матери: Tina Ross
Введите дату рождения матери: 28.6.8976
Удален: Tina Ross. Дата рождения: 28.06.8976
Продолжить? 1 0 :
```