

# Отчёт к ЛР3

Евгений Черников

1	Алфавитный указатель классов	3
1.1	Классы	3
2	Список файлов	5
2.1	Файлы	5
3	Классы	7
3.1	Класс Child	7
3.1.1	Конструктор(ы)	7
3.1.1.1	Child()	8
3.1.2	Документация по друзьям класса и функциям, относящимся к классу	8
3.1.2.1	operator<	8
3.1.2.2	operator<<	8
3.1.2.3	operator<=	8
3.1.2.4	operator==	8
3.1.2.5	operator>	9
3.1.2.6	operator>=	9
3.1.3	Данные класса	9
3.1.3.1	date	9
3.1.3.2	hash	9
3.1.3.3	hashFunc	9
3.1.3.4	name	9
3.1.3.5	shift	9
3.1.3.6	squad	10
3.1.3.7	year	10
3.2	Структура HashItem	10
3.2.1	Данные класса	10
3.2.1.1	values	10
3.3	Класс HashTable	10
3.3.1	Конструктор(ы)	11
3.3.1.1	HashTable()	11
3.3.2	Методы	11
3.3.2.1	clear()	11
3.3.2.2	find()	11
3.3.2.3	getCollisions()	11
3.3.2.4	insert()	11
4	Файлы	13
4.1	Файл child.cpp	13
4.1.1	Функции	13
4.1.1.1	operator<()	13
4.1.1.2	operator<<()	13
4.1.1.3	operator<=()	14
4.1.1.4	operator==()	14

4.1.1.5 operator>()	14
4.1.1.6 operator>=()	14
4.2 Файл child.h	14
4.3 child.h	15
4.4 Файл files.hpp	15
4.4.1 Функции	16
4.4.1.1 createInfo()	16
4.4.1.2 fout()	16
4.4.1.3 randomDate()	16
4.4.1.4 randomString()	16
4.4.1.5 readInfo()	16
4.4.1.6 writeResult()	16
4.4.1.7 writeTime()	17
4.4.2 Переменные	17
4.4.2.1 info	17
4.4.2.2 resultFile	17
4.4.2.3 sizes	17
4.4.2.4 sizesNum	17
4.4.2.5 timesFile	17
4.5 files.hpp	18
4.6 Файл hash.hpp	19
4.6.1 Функции	19
4.6.1.1 complicatedHash()	19
4.6.1.2 naiveHash()	20
4.6.1.3 p_pows()	20
4.6.2 Переменные	20
4.6.2.1 hashDims	20
4.7 hash.hpp	20
4.8 Файл main.cpp	21
4.8.1 Функции	21
4.8.1.1 main()	21
4.9 Файл search.cpp	21
4.9.1 Функции	22
4.9.1.1 binarySearch()	22
4.9.1.2 getLowerBound()	22
4.9.1.3 getUpperBound()	22
4.9.1.4 linearSearch()	22
4.10 Файл search.h	22
4.10.1 Функции	23
4.10.1.1 binarySearch()	23
4.10.1.2 getLowerBound()	23
4.10.1.3 getUpperBound()	23
4.10.1.4 linearSearch()	23

---

4.11 search.h . . . . .	23
4.12 Файл sorts.cpp . . . . .	24
4.12.1 Функции . . . . .	24
4.12.1.1 insertSort() . . . . .	24
4.12.1.2 selectSort() . . . . .	24
4.12.1.3 shakerSort() . . . . .	24
4.13 Файл sorts.h . . . . .	24
4.13.1 Функции . . . . .	25
4.13.1.1 insertSort() . . . . .	25
4.13.1.2 selectSort() . . . . .	25
4.13.1.3 shakerSort() . . . . .	25
4.14 sorts.h . . . . .	25
5 Ссылка на репозиторий . . . . .	27
6 Графики . . . . .	29



# Глава 1

## Алфавитный указатель классов

### 1.1 Классы

Классы с их кратким описанием.

<a href="#">Child</a>	7
<a href="#">HashItem</a>	10
<a href="#">HashTable</a>	10



## Глава 2

# Список файлов

### 2.1 Файлы

Полный список файлов.

<a href="#">child.cpp</a>	13
<a href="#">child.h</a>	14
<a href="#">files.hpp</a>	15
<a href="#">hash.hpp</a>	19
<a href="#">main.cpp</a>	21
<a href="#">search.cpp</a>	21
<a href="#">search.h</a>	22
<a href="#">sorts.cpp</a>	24
<a href="#">sorts.h</a>	24





## Глава 3

# Классы

### 3.1 Класс Child

```
#include <child.h>
```

#### Открытые члены

- `Child` (int `year`, int `squad`, std::string `name`, std::string `date`, int `shift`, unsigned long long(\*`hashFunc`)(std::string))

#### Открытые атрибуты

- int `year`
- int `squad`
- std::string `name`
- std::string `date`
- int `shift`
- unsigned long long(\* `hashFunc` )(std::string)
- unsigned long long `hash`

#### Друзья

- bool `operator==` (const `Child` &a, const `Child` &b)
- bool `operator<` (const `Child` &a, const `Child` &b)
- bool `operator<=` (const `Child` &a, const `Child` &b)
- bool `operator>` (const `Child` &a, const `Child` &b)
- bool `operator>=` (const `Child` &a, const `Child` &b)
- std::ostream & `operator<<` (std::ostream &os, const `Child` &a)

#### 3.1.1 Конструктор(ы)

#### 3.1.1.1 Child()

```
Child::Child (
    int year,
    int squad,
    std::string name,
    std::string date,
    int shift,
    unsigned long long (*)(std::string) hashFunc )
```

### 3.1.2 Документация по друзьям класса и функциям, относящимся к классу

#### 3.1.2.1 operator<

```
bool operator< (
    const Child & a,
    const Child & b ) [friend]
```

#### 3.1.2.2 operator<<

```
std::ostream & operator<< (
    std::ostream & os,
    const Child & a ) [friend]
```

#### 3.1.2.3 operator<=

```
bool operator<= (
    const Child & a,
    const Child & b ) [friend]
```

#### 3.1.2.4 operator==

```
bool operator== (
    const Child & a,
    const Child & b ) [friend]
```

### 3.1.2.5 operator>

```
bool operator> (  
    const Child & a,  
    const Child & b ) [friend]
```

### 3.1.2.6 operator>=

```
bool operator>= (  
    const Child & a,  
    const Child & b ) [friend]
```

## 3.1.3 Данные класса

### 3.1.3.1 date

```
std::string Child::date
```

### 3.1.3.2 hash

```
unsigned long long Child::hash
```

### 3.1.3.3 hashFunc

```
unsigned long long(* Child::hashFunc) (std::string)
```

### 3.1.3.4 name

```
std::string Child::name
```

### 3.1.3.5 shift

```
int Child::shift
```

#### 3.1.3.6 squad

```
int Child::squad
```

#### 3.1.3.7 year

```
int Child::year
```

Объявления и описания членов классов находятся в файлах:

- [child.h](#)
- [child.cpp](#)

## 3.2 Структура HashItem

```
#include <hash.hpp>
```

Открытые атрибуты

- `std::vector< Child > values`

### 3.2.1 Данные класса

#### 3.2.1.1 values

```
std::vector<Child> HashItem::values
```

Объявления и описания членов структуры находятся в файле:

- [hash.hpp](#)

## 3.3 Класс HashTable

```
#include <hash.hpp>
```

Открытые члены

- [HashTable](#) ()
- void [insert](#) ([Child](#) &item)
- [Child](#) [find](#) ([Child](#) &item)
- void [clear](#) ()
- unsigned int [getCollisions](#) ()

### 3.3.1 Конструктор(ы)

#### 3.3.1.1 HashTable()

```
HashTable::HashTable ( ) [inline]
```

### 3.3.2 Методы

#### 3.3.2.1 clear()

```
void HashTable::clear ( ) [inline]
```

#### 3.3.2.2 find()

```
Child HashTable::find (
    Child & item ) [inline]
```

#### 3.3.2.3 getCollisions()

```
unsigned int HashTable::getCollisions ( ) [inline]
```

#### 3.3.2.4 insert()

```
void HashTable::insert (
    Child & item ) [inline]
```

Объявления и описания членов класса находятся в файле:

- [hash.hpp](#)



## Глава 4

# Файлы

### 4.1 Файл child.cpp

```
#include "child.h"
```

#### Функции

- bool `operator==` (const `Child` &a, const `Child` &b)
- bool `operator<` (const `Child` &a, const `Child` &b)
- bool `operator<=` (const `Child` &a, const `Child` &b)
- bool `operator>` (const `Child` &a, const `Child` &b)
- bool `operator>=` (const `Child` &a, const `Child` &b)
- std::ostream & `operator<<` (std::ostream &os, const `Child` &a)

#### 4.1.1 Функции

##### 4.1.1.1 `operator<()`

```
bool operator< (  
    const Child & a,  
    const Child & b )
```

##### 4.1.1.2 `operator<<()`

```
std::ostream & operator<< (  
    std::ostream & os,  
    const Child & a )
```



#### 4.1.1.3 operator<=()

```
bool operator<= (
    const Child & a,
    const Child & b )
```

#### 4.1.1.4 operator==( )

```
bool operator==(
    const Child & a,
    const Child & b )
```

#### 4.1.1.5 operator>()

```
bool operator> (
    const Child & a,
    const Child & b )
```

#### 4.1.1.6 operator>=()

```
bool operator>= (
    const Child & a,
    const Child & b )
```

## 4.2 Файл child.h

```
#include <iostream>
```

### Классы

- class Child

## 4.3 child.h

См. документацию.

```
00001 #ifndef CHILD_H
00002 #define CHILD_H
00003 #include <iostream>
00004
00005 class Child {
00006 public:
00007     int year;
00008     int squad;
00009     std::string name;
00010     std::string date;
00011     int shift;
00012     unsigned long long (*hashFunc)(std::string);
00013     unsigned long long hash;
00014
00015     Child(
00016         int year,
00017         int squad,
00018         std::string name,
00019         std::string date,
00020         int shift,
00021         unsigned long long (*hashFunc)(std::string)
00022     );
00023
00024     friend bool operator== (const Child& a, const Child& b);
00025     friend bool operator< (const Child& a, const Child& b);
00026     friend bool operator<= (const Child& a, const Child& b);
00027     friend bool operator> (const Child& a, const Child& b);
00028     friend bool operator>= (const Child& a, const Child& b);
00029
00030     friend std::ostream& operator<<(std::ostream& os, const Child& a);
00031 };
00032
00033 #endif // CHILD_H
```

## 4.4 Файл files.hpp

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

### Функции

- std::ofstream fout (resultFile)
- std::string randomString (const int len)
- std::string randomDate (int minY, int maxY)
- void createInfo ()
- std::vector< std::vector< Child > > readInfo (unsigned long long(\*hashFunc)(std::string))
- void writeResult (std::string title, std::vector< std::vector< Child > > &res)
- void writeTime (std::string title, std::chrono::steady\_clock::time\_point start, std::chrono::steady\_clock::time\_point end, int divideBy)

### Переменные

- std::string info = "info.txt"
- std::string resultFile = "res.txt"
- std::string timesFile = "times.txt"
- int sizesNum = 7
- int sizes [7] = { 100, 1000, 2000, 5000, 10000, 25000, 100000 }

### 4.4.1 Функции

#### 4.4.1.1 createInfo()

```
void createInfo ( )
```

#### 4.4.1.2 fout()

```
std::ofstream fout (
    resultFile )
```

#### 4.4.1.3 randomDate()

```
std::string randomDate (
    int minY,
    int maxY )
```

#### 4.4.1.4 randomString()

```
std::string randomString (
    const int len )
```

#### 4.4.1.5 readInfo()

```
std::vector< std::vector< Child > > readInfo (
    unsigned long long(*) (std::string) hashFunc )
```

#### 4.4.1.6 writeResult()

```
void writeResult (
    std::string title,
    std::vector< std::vector< Child > > & res )
```

#### 4.4.1.7 writeTime()

```
void writeTime (
    std::string title,
    std::chrono::steady_clock::time_point start,
    std::chrono::steady_clock::time_point end,
    int divideBy )
```

### 4.4.2 Переменные

#### 4.4.2.1 info

```
std::string info = "info.txt"
```

#### 4.4.2.2 resultFile

```
std::string resultFile = "res.txt"
```

#### 4.4.2.3 sizes

```
int sizes[7] = { 100, 1000, 2000, 5000, 10000, 25000, 100000 }
```

#### 4.4.2.4 sizesNum

```
int sizesNum = 7
```

#### 4.4.2.5 timesFile

```
std::string timesFile = "times.txt"
```

## 4.5 files.hpp

[См. документацию.](#)

```

00001 #ifndef FILES_H
00002 #define FILES_H
00003 #include <iostream>
00004 #include <string>
00005 #include <vector>
00006 #include <chrono>
00007 #include <fstream>
00008 #include "child.h"
00009
00010 std::string info = "info.txt";
00011 std::string resultFile = "res.txt";
00012 std::string timesFile = "times.txt";
00013
00014 std::ofstream fout(resultFile);
00015
00016 int sizesNum = 7;
00017
00018 int sizes[7] = { 100, 1000, 2000, 5000, 10000, 25000, 100000 };
00019
00020 std::string randomString(const int len) {
00021     static const char alphanum[] =
00022         "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
00023         "abcdefghijklmnopqrstuvwxyz";
00024     std::string tmp_s;
00025     tmp_s.reserve(len);
00026     for (int i = 0; i < len; ++i) {
00027         tmp_s += alphanum[rand() % (sizeof(alphanum) - 1)];
00028     }
00029     return tmp_s;
00030 }
00031
00032 std::string randomDate(int minY, int maxY)
00033 {
00034     return std::to_string(rand() % 28 + 1) + '.' + std::to_string(rand() % 12 + 1) + '.' + std::to_string(minY + rand() % (maxY
- minY));
00035 }
00036
00037 void createInfo() {
00038     srand((unsigned)time(NULL));
00039     std::ofstream fout(info);
00040
00041     for (int i = 0; i < sizesNum; ++i) {
00042         //Вывод числа записей
00043         fout << sizes[i] << "\n";
00044         for (int j = 0; j < sizes[i]; ++j) {
00045             //Вывод полей по порядку
00046             fout << 2016 - rand() % 40 << "\n"; // Год
00047             fout << rand() % 13 + 1 << "\n"; // Отряд
00048             fout << randomString(10) << "\n"; // ФИО
00049             fout << randomDate(1990, 2016) << "\n"; // Дата рождения
00050             fout << rand() % 5 + 1 << "\n"; // Смена
00051         }
00052     }
00053 }
00054
00055 std::vector<std::vector<Child> readInfo(unsigned long long (*hashFunc)(std::string)) {
00056     std::ifstream fin(info);
00057
00058     std::vector<std::vector<Child> result;
00059
00060     int dim;
00061     int year;
00062     int squad;
00063     std::string name;
00064     std::string date;
00065     int shift;
00066
00067     for (int i = 0; i < sizesNum; ++i) {
00068         //Ввод числа записей
00069         fin >> dim;
00070         std::vector<Child> v;
00071         for (int j = 0; j < dim; ++j) {
00072             //Ввод полей по порядку
00073             fin >> year >> squad >> name >> date >> shift;
00074             Child temp(year, squad, name, date, shift, hashFunc);
00075             v.push_back(temp);
00076         }
00077         result.push_back(v);
00078     }
00079
00080     return result;
00081 }

```

```

00082
00083 void writeResult(std::string title, std::vector<std::vector<Child>& res) {
00084     std::ofstream fout(resultFile, std::ios::app);
00085
00086     fout << title << "\n";
00087
00088     for (int i = 0; i < sizesNum; ++i) {
00089         //Вывод числа записей
00090         fout << "Размер: " << res[i].size() << "\n";
00091         for (int j = 0; j < res[i].size(); ++j) {
00092             //Вывод объекта
00093             fout << res[i][j] << "\n";
00094         }
00095     }
00096 }
00097
00098 void writeTime(std::string title, std::chrono::steady_clock::time_point start, std::chrono::steady_clock::time_point end,
00099               int divideBy) {
00100     fout << title;
00101     fout << std::chrono::duration_cast<std::chrono::microseconds>(end - start).count() / divideBy << " [микросекунд]\n";
00102 }
00103
00104 #endif // FILES_H

```

## 4.6 Файл hash.hpp

```

#include <vector>
#include <string>
#include "child.h"

```

### Классы

- struct [HashItem](#)
- class [HashTable](#)

### Функции

- std::vector< unsigned long long > [p\\_pows](#) (20)
- unsigned long long [naiveHash](#) (std::string key)
- unsigned long long [complicatedHash](#) (std::string key)

### Переменные

- const int [hashDims](#) = 1000000

#### 4.6.1 Функции

##### 4.6.1.1 complicatedHash()

```

unsigned long long complicatedHash (
    std::string key )

```

#### 4.6.1.2 naiveHash()

```
unsigned long long naiveHash (
    std::string key )
```

#### 4.6.1.3 p\_pows()

```
std::vector< unsigned long long > p_pows (
    20 )
```

### 4.6.2 Переменные

#### 4.6.2.1 hashDims

```
const int hashDims = 1000000
```

## 4.7 hash.hpp

[См. документацию.](#)

```
00001 #ifndef HASH_HPP
00002 #define HASH_HPP
00003 #include <vector>
00004 #include <string>
00005 #include "child.h"
00006
00007 const int hashDims = 1000000;
00008 std::vector<unsigned long long> p_pows(20);
00009
00010 unsigned long long naiveHash(std::string key) {
00011     unsigned long long result = 1;
00012
00013     for (int i = 0; i < key.length(); ++i)
00014         result = (result * (key[i] - 'a' + 1)) % hashDims;
00015
00016     return result % hashDims;
00017 }
00018
00019 unsigned long long complicatedHash(std::string key) {
00020     unsigned long long hash = 0;
00021     for (size_t j = 0; j < key.length(); ++j)
00022         hash = (hash + (key[j] - 'a' + 1) * p_pows[j]);
00023
00024     return hash % hashDims;
00025 }
00026
00027 struct HashItem {
00028     std::vector<Child> values;
00029 };
00030
00031 class HashTable {
00032     std::vector<HashItem> table;
00033
00034 public:
00035     HashTable() {
00036         table.resize(hashDims);
00037     }
00038
00039     void insert(Child& item) {
00040         HashItem& currentItem = this->table[item.hash];
00041     }
```

```

00042     //при совпадении элементов выходим
00043     for (auto i : currentItem.values)
00044         if (i.name == item.name)
00045             return;
00046
00047     currentItem.values.push_back(item);
00048 }
00049
00050 Child find(Child& item) {
00051     HashItem& currentItem = this->table[item.hash];
00052
00053     for(int i = 0; i<currentItem.values.size(); ++i)
00054         if(currentItem.values[i].name == item.name)
00055             return currentItem.values[i];
00056     return item;
00057 }
00058
00059 void clear() {
00060     this->table.clear();
00061     this->table.resize(hashDims);
00062 }
00063
00064 unsigned int getCollisions() {
00065     unsigned int result = 0;
00066
00067     for (auto n : this->table)
00068         if (n.values.size() > 1)
00069             result += n.values.size() - 1;
00070
00071     return result;
00072 }
00073 };
00074 };
00075
00076 #endif // HASH_HPP

```

## 4.8 Файл main.cpp

```

#include <map>
#include <fstream>
#include "child.h"
#include "files.hpp"
#include "sorts.h"
#include "search.h"
#include "hash.hpp"

```

### Функции

- int main ()

#### 4.8.1 Функции

##### 4.8.1.1 main()

```
int main ( )
```

## 4.9 Файл search.cpp

```
#include "search.h"
```



## Функции

- `std::vector< int > linearSearch (std::vector< Child > &v, std::string key)`
- `int getLowerBound (std::vector< Child > &v, std::string key)`
- `int getUpperBound (std::vector< Child > &v, std::string key)`
- `std::vector< Child > binarySearch (std::vector< Child > &v, std::string key)`

### 4.9.1 Функции

#### 4.9.1.1 binarySearch()

```
std::vector< Child > binarySearch (  
    std::vector< Child > & v,  
    std::string key )
```

#### 4.9.1.2 getLowerBound()

```
int getLowerBound (  
    std::vector< Child > & v,  
    std::string key )
```

#### 4.9.1.3 getUpperBound()

```
int getUpperBound (  
    std::vector< Child > & v,  
    std::string key )
```

#### 4.9.1.4 linearSearch()

```
std::vector< int > linearSearch (  
    std::vector< Child > & v,  
    std::string key )
```

## 4.10 Файл search.h

```
#include <vector>  
#include "child.h"
```

## Функции

- `std::vector< int > linearSearch` (`std::vector< Child > &v`, `std::string key`)
- `int getLowerBound` (`std::vector< Child > &v`, `std::string key`)
- `int getUpperBound` (`std::vector< Child > &v`, `std::string key`)
- `std::vector< Child > binarySearch` (`std::vector< Child > &v`, `std::string key`)

### 4.10.1 Функции

#### 4.10.1.1 binarySearch()

```
std::vector< Child > binarySearch (
    std::vector< Child > & v,
    std::string key )
```

#### 4.10.1.2 getLowerBound()

```
int getLowerBound (
    std::vector< Child > & v,
    std::string key )
```

#### 4.10.1.3 getUpperBound()

```
int getUpperBound (
    std::vector< Child > & v,
    std::string key )
```

#### 4.10.1.4 linearSearch()

```
std::vector< int > linearSearch (
    std::vector< Child > & v,
    std::string key )
```

## 4.11 search.h

[См. документацию.](#)

```
00001 #ifndef SEARCH_H
00002 #define SEARCH_H
00003 #include <vector>
00004 #include "child.h"
00005
00006 std::vector<int> linearSearch(std::vector<Child>& v, std::string key);
00007
00008 int getLowerBound(std::vector<Child>& v, std::string key);
00009
00010 int getUpperBound(std::vector<Child>& v, std::string key);
00011
00012 std::vector<Child> binarySearch(std::vector<Child>& v, std::string key);
00013
00014 #endif // SEARCH_H
```

## 4.12 Файл sorts.cpp

```
#include "sorts.h"
```

### Функции

- `std::vector< Child > selectSort (std::vector< Child > vec)`
- `std::vector< Child > insertSort (std::vector< Child > vec)`
- `std::vector< Child > shakerSort (std::vector< Child > vec)`

### 4.12.1 Функции

#### 4.12.1.1 insertSort()

```
std::vector< Child > insertSort (  
    std::vector< Child > vec )
```

#### 4.12.1.2 selectSort()

```
std::vector< Child > selectSort (  
    std::vector< Child > vec )
```

#### 4.12.1.3 shakerSort()

```
std::vector< Child > shakerSort (  
    std::vector< Child > vec )
```

## 4.13 Файл sorts.h

```
#include <vector>  
#include "child.h"
```

### Функции

- `std::vector< Child > selectSort (std::vector< Child > vec)`
- `std::vector< Child > insertSort (std::vector< Child > vec)`
- `std::vector< Child > shakerSort (std::vector< Child > vec)`

### 4.13.1 Функции

#### 4.13.1.1 insertSort()

```
std::vector< Child > insertSort (
    std::vector< Child > vec )
```

#### 4.13.1.2 selectSort()

```
std::vector< Child > selectSort (
    std::vector< Child > vec )
```

#### 4.13.1.3 shakerSort()

```
std::vector< Child > shakerSort (
    std::vector< Child > vec )
```

## 4.14 sorts.h

[См. документацию.](#)

```
00001 #ifndef SORTS_H
00002 #define SORTS_H
00003 #include <vector>
00004 #include "child.h"
00005
00006 std::vector<Child> selectSort(std::vector<Child> vec);
00007
00008 std::vector<Child> insertSort(std::vector<Child> vec);
00009
00010 std::vector<Child> shakerSort(std::vector<Child> vec);
00011
00012 #endif // SORTS_H
```



## Глава 5

# Ссылка на репозиторий

Ссылка: [https://github.com/EvgenijCS202/MP\\_3](https://github.com/EvgenijCS202/MP_3)



## Глава 6

## Графики





