

ContactAddressBook

beta 0.0.1

制作者 Doxygen 1.9.5

1 类索引	1
1.1 类列表	1
2 文件索引	3
2.1 文件列表	3
3 类说明	5
3.1 AddressBook类 参考	5
3.1.1 详细描述	5
3.1.2 构造及析构造函数说明	5
3.1.2.1 AddressBook()	5
3.1.2.2 ~AddressBook()	6
3.1.3 成员函数说明	6
3.1.3.1 AddContact()	6
3.1.3.2 Delete()	6
3.1.3.3 Find()	6
3.1.3.4 List()	6
3.1.3.5 ListGroup()	6
3.1.3.6 operator[]()	7
3.1.3.7 Sort()	7
3.2 CContact类 参考	7
3.2.1 详细描述	8
3.2.2 构造及析构造函数说明	8
3.2.2.1 CContact() [1/3]	8
3.2.2.2 CContact() [2/3]	8
3.2.2.3 CContact() [3/3]	8
3.2.2.4 ~CContact()	9
3.2.3 成员函数说明	9
3.2.3.1 getContact()	9
3.2.3.2 operator<()	9
3.2.3.3 operator=()	10
3.2.3.4 PatternMatch()	10
3.2.3.5 setContact()	10
3.2.4 友元及相关函数文档	11
3.2.4.1 operator<<	11
3.2.4.2 operator>>	11
3.2.4.3 pr	12
4 文件说明	13
4.1 AddressBook.h 文件参考	13
4.2 AddressBook.h	13
4.3 CContact.cpp 文件参考	14
4.3.1 函数说明	14

4.3.1.1 operator<<()	14
4.3.1.2 operator>>()	14
4.4 CContact.h 文件参考	15
4.5 CContact.h	15
4.6 main.cpp 文件参考	16
4.6.1 函数说明	16
4.6.1.1 main()	16
Index	17

Chapter 1

类索引

1.1 类列表

这里列出了所有类、结构、联合以及接口定义等，并附带简要说明：

AddressBook	
地址簿类	5
CContact	
CContact 是联系人类	7

Chapter 2

文件索引

2.1 文件列表

这里列出了所有文件，并附带简要说明:

AddressBook.h	13
CContact.cpp	14
CContact.h	15
main.cpp	16

Chapter 3

类说明

3.1 AddressBook类 参考

地址簿类

```
#include <AddressBook.h>
```

Public 成员函数

- [AddressBook](#) ()
- [~AddressBook](#) ()
- void [AddContact](#) (std::string &, std::string &, std::string &)
- int [Find](#) (int startIndex, std::string &, std::string &, std::string &)
- [CContact operator\[\]](#) (int)
- int [Delete](#) (std::string &, std::string &, std::string &)
- void [Sort](#) ()
- void [List](#) ()
- void [ListGroup](#) (std::string &group)

3.1.1 详细描述

地址簿类

Detailed Describer

3.1.2 构造及析构函数说明

3.1.2.1 AddressBook()

```
AddressBook::AddressBook ( )
```

3.1.2.2 ~AddressBook()

```
AddressBook::~~AddressBook ( )
```

3.1.3 成员函数说明

3.1.3.1 AddContact()

```
void AddressBook::AddContact (
    std::string & ,
    std::string & ,
    std::string & )
```

3.1.3.2 Delete()

```
int AddressBook::Delete (
    std::string & ,
    std::string & ,
    std::string & )
```

3.1.3.3 Find()

```
int AddressBook::Find (
    int startIndex,
    std::string & ,
    std::string & ,
    std::string & )
```

3.1.3.4 List()

```
void AddressBook::List ( )
```

3.1.3.5 ListGroup()

```
void AddressBook::ListGroup (
    std::string & group )
```

3.1.3.6 operator[]()

```
CContact AddressBook::operator[] (
    int )
```

3.1.3.7 Sort()

```
void AddressBook::Sort ( )
```

该类的文档由以下文件生成:

- [AddressBook.h](#)

3.2 CContact类 参考

[CContact](#) 是联系人类

```
#include <CContact.h>
```

Public 成员函数

- [CContact](#) ()
- [CContact](#) (std::string &Name, std::string &Number, std::string &Group)
使用 *Name, Number, Group* 创建联系人对象
- [CContact](#) (const [CContact](#) &)
拷贝构造函数
- virtual [~CContact](#) ()
析构函数
- bool [operator<](#) ([CContact](#) &)
重载 < 运算符, 供算法 *sort* 使用, 按姓名排序
- void [operator=](#) (const [CContact](#) &)
重载赋值 = 运算符
- void [getContact](#) (std::string &, std::string &, std::string &)
获取对象的三个成员
- void [setContact](#) (std::string &, std::string &, std::string &)
设定对象的三个成员
- bool [PatternMatch](#) (std::string &, std::string &, std::string &)

友元

- std::ostream & [operator<<](#) (std::ostream &, [CContact](#))
利用友元函数重载运算符 <<
- std::istream & [operator>>](#) (std::istream &, [CContact](#) &)
利用友元函数重载运算符 >>
- bool [pr](#) (const [CContact](#) &, const [CContact](#) &)

3.2.1 详细描述

`CContact` 是联系人类

3.2.2 构造及析构造函数说明

3.2.2.1 CContact() [1/3]

```
CContact::CContact ( )
```

3.2.2.2 CContact() [2/3]

```
CContact::CContact (
    std::string & Name,
    std::string & Number,
    std::string & Group )
```

使用Name,Number,Group创建联系人对象

参数

<i>Name</i>	Contact Name
<i>Number</i>	Phone Number
<i>Group</i>	Contact Group

注解

Visual Studio Code C++ Extension July 2020 Update: Doxygen comments and Log points <https://devblogs.microsoft.com/cppblog/visual-studio-code-c-extension-july-2020-update-doxygen-comments-and-log-points/>

3.2.2.3 CContact() [3/3]

```
CContact::CContact (
    const CContact & ContactInfo )
```

拷贝构造函数

参数

<i>ContactInfo</i>	Initialized Contact
--------------------	---------------------

3.2.2.4 ~CContact()

```
CContact::~~CContact ( ) [virtual]
```

析构函数

3.2.3 成员函数说明

3.2.3.1 getContact()

```
void CContact::getContact (
    std::string & Name,
    std::string & Number,
    std::string & Group )
```

获取对象的三个成员

参数

<i>Name</i>	return Name as reference
<i>Number</i>	return Number as reference
<i>Group</i>	return Group as reference

3.2.3.2 operator<()

```
bool CContact::operator< (
    CContact & contactToBeCompared )
```

重载 < 运算符，供算法sort使用,按姓名排序

参数

<i>contactToBeCompared</i>	contact info to be compared.
----------------------------	------------------------------

返回

if thisCContact < contactToBeCompared, return true;

注解

C++ 重载运算符和重载函数 <https://www.runoob.com/cplusplus/cpp-overloading.html>

3.2.3.3 operator=()

```
void CContact::operator= (
    const CContact & oldContact )
```

重载赋值 = 运算符

参数

<i>oldContact</i>	local variable, newContact = oldContact, set new is equal to old.
-------------------	---

注解

C++ 赋值运算符 = 重载 <https://www.runoob.com/cplusplus/assignment-operators-overloading.html>

3.2.3.4 PatternMatch()

```
bool CContact::PatternMatch (
    std::string & ,
    std::string & ,
    std::string & )
```

3.2.3.5 setContact()

```
void CContact::setContact (
    std::string & Name,
    std::string & Number,
    std::string & Group )
```

设定对象的三个成员

参数

<i>Name</i>	Contact Name
<i>Number</i>	Phone Number
<i>Group</i>	Contact Group

3.2.4 友元及相关函数文档

3.2.4.1 operator<<

```
std::ostream & operator<< (  
    std::ostream & os,  
    CContact contactInfo ) [friend]
```

利用友元函数重载运算符 <<

参数

<code>std::ostream</code>	file stream
<code>CContact</code>	output contact class

返回

ostream

参见

【懒猫老师-最简版C++-(18)类的友元】 <https://www.bilibili.com/video/BV127411Q7eu/>

注解

Overloading the << Operator for Your Own Classes <https://learn.microsoft.com/en-us/cpp/standard-library/overloading-the-less-than-less-than-operator-for-your-own-classes>

3.2.4.2 operator>>

```
std::istream & operator>> (  
    std::istream & is,  
    CContact & contactToBeRevised ) [friend]
```

利用友元函数重载运算符 >>

参数

<code>std::istream&</code>	is
<code>CContact</code>	

返回

istream

注解

Overloading the >> Operator for Your Own Classes <https://learn.microsoft.com/en-us/cpp/standard-library>

3.2.4.3 pr

```
bool pr (
    const CContact & ,
    const CContact & ) [friend]
```

该类的文档由以下文件生成:

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

Chapter 4

文件说明

4.1 AddressBook.h 文件参考

```
#include <string>
#include <vector>
#include "CContact.h"
```

类

- class [AddressBook](#)
地址簿类

4.2 AddressBook.h

[浏览该文件的文档.](#)

```
1 /*
2  * @Author: Frank Chu
3  * @Date: 2022-11-16 16:04:46
4  * @LastEditors: Frank Chu
5  * @LastEditTime: 2022-11-17 11:17:31
6  * @FilePath: /Cpp/lab/Cpp-lab01-week11/source/AddressBook.h
7  * @Description:
8  *
9  * Copyright (c) 2022 by Frank Chu, All Rights Reserved.
10 */
11
12 #include <string>
13 #include <vector>
14
15 #include "CContact.h"
16
23 class AddressBook
24 {
25     std::vector<CContact> Book;
26     // 以CContact类实例化类模板vector形成CContact向量作为存储结构。
27     // Book是CContact向量类的一个实例
28 public:
29     AddressBook(); //建立空地址簿
30     ~AddressBook(); //地址簿析构函数，其中必须清空联系人向量Book
31
32     void AddContact(std::string &, std::string &, std::string &); //在向量中增加一个联系人
33
34     int Find(int startIndex, std::string &, std::string &, std::string &); //从下标startIndex开始寻找符合匹配
    条件的联系人，如果找到，则返回下标，否则返回-1
35
36     CContact operator[](int); //重载下标运算符
37     int Delete(std::string &, std::string &, std::string &); //按条件删除联系人，返回删除的人数。如果没有删除任何
    人，返回0
38     void Sort(); //按姓名排序Book; void SortGroup(); //按群组排
    序Book;
39     void List(); //输出Book中所有联系人。
40     void ListGroup(std::string &group); //输出某组别中的所有联系人
41 };
```

4.3 CContact.cpp 文件参考

```
#include "CContact.h"
```

函数

- `std::ostream & operator<<` (`std::ostream &os`, `CContact` `contactInfo`)
利用友元函数重载运算符 <<
- `std::istream & operator>>` (`std::istream &is`, `CContact` &`contactToBeRevised`)
利用友元函数重载运算符 >>

4.3.1 函数说明

4.3.1.1 operator<<()

```
std::ostream & operator<< (  
    std::ostream & os,  
    CContact contactInfo )
```

利用友元函数重载运算符 <<

参数

<code>std::ostream</code>	file stream
<code>CContact</code>	output contact class

返回

`ostream`

参见

【懒猫老师-最简版C++-(18)类的友元】 <https://www.bilibili.com/video/BV127411Q7eu/>

注解

Overloading the << Operator for Your Own Classes <https://learn.microsoft.com/en-us/cpp/standard-library/overloading-the-less-than-less-than-operator-for-your-own-classes>

4.3.1.2 operator>>()

```
std::istream & operator>> (  
    std::istream & is,  
    CContact & contactToBeRevised )
```

利用友元函数重载运算符 >>

参数

<code>std::istream&</code>	is
<code>CContact</code>	

返回

istream

注解

Overloading the >> Operator for Your Own Classes <https://learn.microsoft.com/en-us/cpp/standard-library>

4.4 CContact.h 文件参考

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

类

- class `CContact`
`CContact` 是联系人类

4.5 CContact.h

浏览该文件的文档.

```
1 /*
2  * @Author: Frank Chu
3  * @Date: 2022-11-16 13:11:56
4  * @LastEditors: Frank Chu
5  * @LastEditTime: 2022-11-17 01:07:08
6  * @FilePath: /Cpp/lab/Cpp-lab01-week11/CContact.h
7  * @Description:
8  *
9  * Copyright (c) 2022 by Frank Chu, All Rights Reserved.
10 */
11
12 #ifndef CCONTACT_H
13 #define CCONTACT_H
14 #include <iostream>
15 #include <string>
16
17 // 判断字符串source是否匹配pattern
18 // 或者说字符串source是pattern所表达的集合中的某个成员
19
20 // bool match(std::string &pattern, std::string &source); //字符串匹配
21
22 class CContact
23 {
24 private:
25     std::string Name; // 姓名
26     std::string Number; // 电话号码
27     std::string Group; // 群组
28 public:
29     CContact(); // 默认构造函数
30     CContact(std::string &Name, std::string &Number, std::string &Group); // 使用 Name, Number, Group 创建联系对象
31     CContact(const CContact &); // 拷贝构造函数
32     virtual ~CContact(); // 析构函数
```

```
36
37     bool operator<(CContact &); // 重载 < 运算符, 供算法 sort 使用, 按姓名排序
38     void operator=(const CContact &); // 重载赋值运算符
39     friend std::ostream &operator<<(std::ostream &, CContact); // 利用友元函数重载运算符 <<
40     friend std::istream &operator>>(std::istream &, CContact &); // 利用友元函数重载运算符 >>
41
42     friend bool pr(const CContact &, const CContact &); // 定义组排序函数, 供算法sort使用
43     void getContact(std::string &, std::string &, std::string &); // 获取对象的三个成员
44     void setContact(std::string &, std::string &, std::string &); // 设定对象的三个成员
45     bool PatternMatch(std::string &, std::string &, std::string &); // 判定本对象是否匹配搜索条件
46 };
47
48 #endif
```

4.6 main.cpp 文件参考

```
#include "CContact.h"
#include "CContact.cpp"
#include <iostream>
#include <string>
```

函数

- int main ()

4.6.1 函数说明

4.6.1.1 main()

```
int main ( )
```

Index

- ~AddressBook
 - AddressBook, [5](#)
- ~CContact
 - CContact, [9](#)
- AddContact
 - AddressBook, [6](#)
- AddressBook, [5](#)
 - ~AddressBook, [5](#)
 - AddContact, [6](#)
 - AddressBook, [5](#)
 - Delete, [6](#)
 - Find, [6](#)
 - List, [6](#)
 - ListGroup, [6](#)
 - operator[], [6](#)
 - Sort, [7](#)
- AddressBook.h, [13](#)
- CContact, [7](#)
 - ~CContact, [9](#)
 - CContact, [8](#)
 - getContact, [9](#)
 - operator<, [9](#)
 - operator<<, [11](#)
 - operator>>, [11](#)
 - operator=, [10](#)
 - PatternMatch, [10](#)
 - pr, [12](#)
 - setContact, [10](#)
- CContact.cpp, [14](#)
 - operator<<, [14](#)
 - operator>>, [14](#)
- CContact.h, [15](#)
- Delete
 - AddressBook, [6](#)
- Find
 - AddressBook, [6](#)
- getContact
 - CContact, [9](#)
- List
 - AddressBook, [6](#)
- ListGroup
 - AddressBook, [6](#)
- main
 - main.cpp, [16](#)
- main.cpp, [16](#)
 - main, [16](#)
- operator<
 - CContact, [9](#)
- operator<<
 - CContact, [11](#)
 - CContact.cpp, [14](#)
- operator>>
 - CContact, [11](#)
 - CContact.cpp, [14](#)
- operator=
 - CContact, [10](#)
- operator[]
 - AddressBook, [6](#)
- PatternMatch
 - CContact, [10](#)
- pr
 - CContact, [12](#)
- setContact
 - CContact, [10](#)
- Sort
 - AddressBook, [7](#)