

My Project

Generated by Doxygen 1.9.3

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 Date Class Reference	5
3.1.1 Constructor & Destructor Documentation	6
3.1.1.1 Date()	6
3.1.2 Member Function Documentation	6
3.1.2.1 getToday()	6
3.1.2.2 operator()	6
3.1.2.3 operator++()	7
3.1.2.4 operator-()	7
3.1.2.5 operator<()	7
3.1.2.6 operator==(())	8
3.1.2.7 operator>()	8
3.1.3 Friends And Related Function Documentation	8
3.1.3.1 operator<<	8
3.1.3.2 operator>>	9
3.2 DatePeriod Struct Reference	9
3.3 Hotel Class Reference	10
3.3.1 Constructor & Destructor Documentation	10
3.3.1.1 Hotel()	10
3.3.2 Member Function Documentation	11
3.3.2.1 freeRoom()	11
3.3.2.2 getName()	11
3.3.2.3 getReport()	11
3.3.2.4 reserveRoom()	12
3.3.2.5 serviceRoom()	12
3.3.2.6 showAvailableRooms()	13
3.3.2.7 today()	13
3.4 HotelBuilding Class Reference	13
3.5 Reservation Class Reference	14
3.5.1 Constructor & Destructor Documentation	14
3.5.1.1 Reservation()	14
3.5.2 Member Function Documentation	15
3.5.2.1 getFrom()	15
3.5.2.2 getNights()	15
3.5.2.3 getTo()	15
3.5.2.4 isActive()	16
3.5.2.5 isPast()	16

3.5.2.6 isServiced()	16
3.5.2.7 LeavingInAdvance()	16
3.5.2.8 onDate()	16
3.5.2.9 stateOnDate()	17
3.6 Room Class Reference	17
3.6.1 Constructor & Destructor Documentation	18
3.6.1.1 Room()	18
3.6.2 Member Function Documentation	18
3.6.2.1 freeRoom()	18
3.6.2.2 isFreeOnDate()	18
3.7 RoomAnalyzer Class Reference	19
4 File Documentation	21
4.1 Date.hpp	21
4.2 Hotel.hpp	21
4.3 HotelBuilding.hpp	22
4.4 Reservation.hpp	22
4.5 Room.hpp	23
4.6 RoomAnalyzer.hpp	24
4.7 Types.hpp	24
Index	25

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Date	5
DatePeriod	9
Hotel	10
HotelBuilding	13
Reservation	14
Room	17
RoomAnalyzer	19

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Date.hpp	??
Hotel.hpp	??
HotelBuilding.hpp	??
Reservation.hpp	??
Room.hpp	??
RoomAnalyzer.hpp	??
Types.hpp	??

Chapter 3

Class Documentation

3.1 Date Class Reference

Public Member Functions

- [Date](#) (unsigned short d=1, unsigned short m=1, unsigned short y=1900)
Construct a new [Date](#) object from day, month and year. Default [Date](#) is 1/1/1900.
- bool [operator<](#) ([Date](#) other) const
checks if this [Date](#) is chronologically before other [Date](#)
- bool [operator<=](#) ([Date](#) other) const
see [operator<](#)
- bool [operator>](#) ([Date](#) other) const
checks if this [Date](#) is chronologically after other [Date](#)
- bool [operator>=](#) ([Date](#) other) const
see [operator>](#)
- bool [operator==](#) ([Date](#) other) const
checks if two dates are identical
- const char * [operator\(\)](#) (char *buf) const
records this [Date](#) in buffer in format YYYY-MM-DD
- int [operator-](#) ([Date](#) other) const
- [Date](#) & [operator++](#) ()
overloaded prefix incrementation operator for [Date](#)

Static Public Member Functions

- static [Date](#) [getToday](#) ()
Get the today date.

Friends

- std::istream & [operator>>](#) (std::istream &is, [Date](#) &d)
overloaded operator for inputing [Date](#)
- std::ostream & [operator<<](#) (std::ostream &os, const [Date](#) &d)
overloaded operator for outputing [Date](#)

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Date()

```
Date::Date (
    unsigned short d = 1,
    unsigned short m = 1,
    unsigned short y = 1900 ) [inline]
```

Construct a new [Date](#) object from day, month and year. Default [Date](#) is 1/1/1900.

Parameters

<i>d</i>	day
<i>m</i>	month
<i>y</i>	year

3.1.2 Member Function Documentation

3.1.2.1 getToday()

```
Date Date::getToday ( ) [static]
```

Get the today date.

Returns

[Date](#)

3.1.2.2 operator()()

```
const char * Date::operator() (
    char * buf ) const
```

records this [Date](#) in buffer in format YYYY-MM-DD

Parameters

<i>buf</i>	buffer where Date is recorded
------------	---

Returns

const char* pointer to beginning of buf

3.1.2.3 operator++()

```
Date & Date::operator++ ( )
```

overloaded prefix incrementation operator for [Date](#)

Returns

[Date](#)& reference to this [Date](#)

3.1.2.4 operator-()

```
int Date::operator- (
    Date other ) const
```

Parameters

<i>other</i>	Date
--------------	----------------------

Returns

int difference between of this [Date](#) and other

3.1.2.5 operator<()

```
bool Date::operator< (
    Date other ) const
```

checks if this [Date](#) is chronologically before other [Date](#)

Parameters

<i>other</i>	compared Date
--------------	-------------------------------

Returns

true this [Date](#) is chronologically before other

false this [Date](#) is not chronologically before other

3.1.2.6 operator==()

```
bool Date::operator== (
    Date other ) const
```

checks if two dates are identical

Parameters

<i>other</i>	compared Date
--------------	-------------------------------

Returns

true the dates are identical

false the dates are not identical

3.1.2.7 operator>()

```
bool Date::operator> (
    Date other ) const
```

checks if this [Date](#) is chronologically after other [Date](#)

Parameters

<i>other</i>	compared Date
--------------	-------------------------------

Returns

true this [Date](#) is chronologically after other

false this [Date](#) is not chronologically after other

3.1.3 Friends And Related Function Documentation

3.1.3.1 operator<<

```
std::ostream & operator<< (
    std::ostream & os,
    const Date & d ) [friend]
```

overloaded operator for outputting [Date](#)

Parameters

<i>os</i>	output stream
<i>d</i>	Date to be output

Returns

std::ostream& reference to the output stream

3.1.3.2 operator>>

```
std::istream & operator>> (  
    std::istream & is,  
    Date & d ) [friend]
```

overloaded operator for inputing [Date](#)

Parameters

<i>is</i>	input stream
<i>d</i>	Date to be input

Returns

std::istream& reference to the input stream

The documentation for this class was generated from the following files:

- Date.hpp
- Date.cpp

3.2 DatePeriod Struct Reference

Public Member Functions

- unsigned **length** () const

Public Attributes

- [Date](#) from
- [Date](#) to

The documentation for this struct was generated from the following file:

- Date.hpp

3.3 Hotel Class Reference

Public Member Functions

- [Hotel](#) (std::string hotelDataFile)
Construct a new [Hotel](#) object.
- [Hotel](#) (const [Hotel](#) &)=delete
- [Hotel](#) & **operator=** (const [Hotel](#) &)=delete
- **~Hotel** ()
Destroy the [Hotel](#) object.
- std::string [getName](#) () const
get the name of this [Hotel](#)
- void **nextDay** ()
advance to the nextDay
- bool [reserveRoom](#) (unsigned number, [DatePeriod](#) period, std::string name="-", std::string note="None\n")
makes a new [Reservation](#) for particular [Room](#) and period with options for name of guest and notes to the [Reservation](#)
- [Hotel](#) & [showAvailableRooms](#) (std::ostream &, [Date](#))
output to stream all available rooms for a particular [Date](#)
- bool [freeRoom](#) (unsigned number)
tries to free [Room](#) with particular ID
- [Hotel](#) & [getReport](#) ([DatePeriod](#) period)
Creates report for the usage of this [Hotel](#)'s rooms in the period from-to. Report written in file named "report-YYYY-MM-DD.txt" where YYYY-MM-DD is the beginning of the period.
- void **searchRoom** (unsigned minBeds, [DatePeriod](#) period) const
- bool [serviceRoom](#) (unsigned number, [DatePeriod](#) period, std::string note)
plans maintenance for particular [Room](#) and period leaving note for the service

Static Public Member Functions

- static [Date](#) [today](#) ()
get today's [Date](#) according to all [Hotels](#)

3.3.1 Constructor & Destructor Documentation

3.3.1.1 [Hotel](#)()

```
Hotel::Hotel (
    std::string hotelDataFile )
```

Construct a new [Hotel](#) object.

Parameters

<i>hotelDataFile</i>	path to file where rooms are recorded
----------------------	---------------------------------------

3.3.2 Member Function Documentation

3.3.2.1 freeRoom()

```
bool Hotel::freeRoom (
    unsigned number )
```

tries to free [Room](#) with particular ID

Parameters

<i>number</i>	Room 's ID
---------------	----------------------------

Returns

true room is now free
false room not found

3.3.2.2 getName()

```
std::string Hotel::getName ( ) const [inline]
```

get the name of this [Hotel](#)

Returns

std::string

3.3.2.3 getReport()

```
Hotel & Hotel::getReport (
    DatePeriod period )
```

Creates report for the usage of this [Hotel](#)'s rooms in the period from-to. Report written in file named "report-YYYY-MM-DD.txt" where YYYY-MM-DD is the beginning of the period.

Parameters

<i>period</i>	desired period of time
---------------	------------------------

Returns

[Hotel](#) & this [Hotel](#)

3.3.2.4 reserveRoom()

```
bool Hotel::reserveRoom (
    unsigned number,
    DatePeriod period,
    std::string name = "-",
    std::string note = "None\n" )
```

makes a new [Reservation](#) for particular [Room](#) and period with options for name of guest and notes to the [Reservation](#)

Parameters

<i>number</i>	of the desired Room
<i>from</i>	accomodation Date
<i>to</i>	leaving Date
<i>name</i>	guest's name
<i>note</i>	note to the reservation

Returns

true successfull reservation
false failed reservation (not made)

3.3.2.5 serviceRoom()

```
bool Hotel::serviceRoom (
    unsigned number,
    DatePeriod period,
    std::string note )
```

plans maintenance for particular [Room](#) and period leaving note for the service

Parameters

<i>number</i>	Room 's ID
<i>period</i>	desired period of time
<i>note</i>	any notes to the service

Returns

true service planned successfully
false service planning failed (room not found or is reserved for the period)

3.3.2.6 showAvailableRooms()

```
Hotel & Hotel::showAvailableRooms (
    std::ostream & os,
    Date d )
```

output to stream all available rooms for a particular [Date](#)

Returns

[Hotel](#)& this [Hotel](#)

3.3.2.7 today()

```
static Date Hotel::today ( ) [inline], [static]
```

get today's [Date](#) according to all Hotels

Returns

[Date](#)

The documentation for this class was generated from the following files:

- Hotel.hpp
- Hotel.cpp

3.4 HotelBuilding Class Reference**Public Member Functions**

- **HotelBuilding** (std::ifstream &ifs)
- **HotelBuilding** (const [HotelBuilding](#) &other)=delete
- [HotelBuilding](#) & **operator=** ([HotelBuilding](#) &other)=delete
- size_t **getRoomCount** () const
- [Room](#) * **operator[]** (unsigned roomNumber) const
- void **newDate** ([Date](#) d)
- void **showAvailableRooms** (std::ostream &os, [Date](#) d) const
- void **createReport** ([DatePeriod](#) period) const
- void **suggestRoom** (unsigned beds, [DatePeriod](#) period)

Friends

- class **RoomAnalyzer**

The documentation for this class was generated from the following files:

- HotelBuilding.hpp
- HotelBuilding.cpp

3.5 Reservation Class Reference

Public Member Functions

- [Reservation](#) (std::string name, [DatePeriod](#) p, std::string n="None.\n", bool s=false)
Construct a new [Reservation](#) object.
- **Reservation** (const [Reservation](#) &)=delete
- [Reservation](#) & **operator=** (const [Reservation](#) &)=delete
- bool [isActive](#) () const
see if this [Reservation](#) is active (today is part of the period)
- bool [isPast](#) () const
see if this [Reservation](#) is past (today is after end of period)
- bool [isServiced](#) () const
see if this [Reservation](#) is a maintenance
- [Date](#) [getFrom](#) () const
get beginning [Date](#) of this [Reservation](#)
- [Date](#) [getTo](#) () const
get end [Date](#) of this [Reservation](#)
- unsigned [getNights](#) () const
get count of nights of this [Reservation](#)
- void [onDate](#) ([Date](#) d)
update the state of the reservation based on new today's [Date](#) (d)
- ReservationState [stateOnDate](#) ([Date](#)) const
see what would the state of this [Reservation](#) be on particular [Date](#)
- bool [LeavingInAdvance](#) ([Date](#))
try to change end of period for earlier end of this [Reservation](#)

3.5.1 Constructor & Destructor Documentation

3.5.1.1 Reservation()

```
Reservation::Reservation (
    std::string name,
    DatePeriod p,
    std::string n = "None.\n",
    bool s = false )
```

Construct a new [Reservation](#) object.

Parameters

<i>name</i>	of the reserver
<i>p</i>	
<i>n</i>	note left for the reservation
<i>s</i>	whether it is reservation or maintenance

3.5.2 Member Function Documentation

3.5.2.1 getFrom()

```
Date Reservation::getFrom ( ) const [inline]
```

get beginning [Date](#) of this [Reservation](#)

Returns

[Date](#) beginning of the [Reservation](#)

3.5.2.2 getNights()

```
unsigned Reservation::getNights ( ) const [inline]
```

get count of nights of this [Reservation](#)

Returns

unsigned count of nights of this [Reservation](#)

3.5.2.3 getTo()

```
Date Reservation::getTo ( ) const [inline]
```

get end [Date](#) of this [Reservation](#)

Returns

[Date](#) end of the [Reservation](#)

3.5.2.4 isActive()

```
bool Reservation::isActive ( ) const [inline]
```

see if this [Reservation](#) is active (today is part of the period)

Returns

true [Reservation](#) is active
false [Reservation](#) is not active (past or future)

3.5.2.5 isPast()

```
bool Reservation::isPast ( ) const [inline]
```

see if this [Reservation](#) is past (today is after end of period)

Returns

true [Reservation](#) is past
false [Reservation](#) is not past (active or future)

3.5.2.6 isServiced()

```
bool Reservation::isServiced ( ) const [inline]
```

see if this [Reservation](#) is a maintenance

Returns

true this [Reservation](#) is a maintenace
false this [Reservation](#) is for a guest

3.5.2.7 LeavingInAdvance()

```
bool Reservation::LeavingInAdvance (
    Date newTo )
```

try to change end of period for earlier end of this [Reservation](#)

Returns

true end date modified for leaving in advance
false new leaving [Date](#) not appropriate for earlier leaving

3.5.2.8 onDate()

```
void Reservation::onDate (
    Date d )
```

update the state of the reservation based on new today's [Date](#) (d)

Parameters

<i>d</i>	new today's Date
----------	----------------------------------

3.5.2.9 stateOnDate()

```
ReservationState Reservation::stateOnDate (
    Date d ) const
```

see what would the state of this [Reservation](#) be on particular [Date](#)

Returns

ReservationState state on desired [Date](#)

The documentation for this class was generated from the following files:

- Reservation.hpp
- Reservation.cpp

3.6 Room Class Reference

Public Member Functions

- [Room](#) (unsigned n, unsigned bC)
Construct a new [Room](#) object.
- **Room** (const [Room](#) &)=delete
forbidden copying of rooms
- [Room](#) & **operator=** (const [Room](#) &)=delete
forbidden copying of rooms
- **~Room** ()
Destroy the [Room](#) object.
- unsigned **getNumber** () const
- unsigned **getBedCount** () const
- bool **isFreeNow** () const
- bool **freeRoom** ([Reservation](#) *¤tRes)
try to free this room
- void **changeLeaving** ([Reservation](#) *, [Date](#) newDate)
- void **newDate** ([Date](#))
apply new [Date](#) to state of all reservations and respectively of the room availability
- bool **isFreeOnDate** ([Date](#)) const
see if this room is free in certain date
- bool **isFreeInPeriod** ([DatePeriod](#) period) const
- void **showReservationsInPeriod** (std::ostream &os, [DatePeriod](#) period) const
- bool **addReservation** (std::string name, std::string note, [DatePeriod](#) period)
- bool **closeForService** (std::string note, [DatePeriod](#) period)

3.6.1 Constructor & Destructor Documentation

3.6.1.1 Room()

```
Room::Room (
    unsigned n,
    unsigned bC )
```

Construct a new [Room](#) object.

Parameters

<i>n</i>	number of constructed Room
<i>bC</i>	number of beds in constructed Room

3.6.2 Member Function Documentation

3.6.2.1 freeRoom()

```
bool Room::freeRoom (
    Reservation *& currentRes )
```

try to free this room

Returns

true sucesfully freed room
false room is already free

3.6.2.2 isFreeOnDate()

```
bool Room::isFreeOnDate (
    Date d ) const
```

see if this room is free in certain date

Returns

true the room is free
false the room is taken

The documentation for this class was generated from the following files:

- [Room.hpp](#)
- [Room.cpp](#)

3.7 RoomAnalyzer Class Reference

Static Public Member Functions

- static void **suggest** ([HotelBuilding](#) &hB, unsigned beds, [DatePeriod](#) period)

The documentation for this class was generated from the following files:

- RoomAnalyzer.hpp
- RoomAnalyzer.cpp

Chapter 4

File Documentation

4.1 Date.hpp

```
1 #ifndef __DATE_HPP
2 #define __DATE_HPP
3 #include <iostream>
4 #include <ctime>
5
10 const unsigned daysFromBeginning[] = {0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334};
11
12 class Date
13 {
14     unsigned short day, month, year;
25     bool isValid() const;
33     bool isLeap(unsigned y) const;
34
35 public:
43     Date(unsigned short d = 1, unsigned short m = 1, unsigned short y = 1900) : day(d), month(m), year(y)
    {}
51     bool operator<(Date other) const;
55     bool operator<=(Date other) const;
63     bool operator>(Date other) const;
67     bool operator>=(Date other) const;
75     bool operator==(Date other) const;
82     const char *operator()(char *buf) const;
89     int operator-(Date other) const;
95     Date &operator++();
96
102     static Date getToday();
103
111     friend std::istream &operator>(std::istream &is, Date &d);
119     friend std::ostream &operator<<(std::ostream &os, const Date &d);
120 };
121
122 struct DatePeriod
123 {
124     Date from, to;
125     unsigned length() const { return to - from; }
126 };
127
128 std::istream &operator>(std::istream &is, DatePeriod &dP);
129
130 #endif
```

4.2 Hotel.hpp

```
1 #ifndef __HOTEL_HPP
2 #define __HOTEL_HPP
3 #include "Types.hpp"
4 #include "Date.hpp"
5 #include "Room.hpp"
6 #include "Reservation.hpp"
7 #include "HotelBuilding.hpp"
8 #include <string>
9
17 std::string readFromIfstream(std::ifstream &ifis, size_t len);
18
```

```

19 class Hotel
20 {
21     std::string name;
22
23     static Date now;
24
25     HotelBuilding *building;
26
27 public:
28     Hotel() = delete;
29
30     Hotel(std::string hotelDataFile);
31     Hotel(const Hotel &) = delete;
32     Hotel &operator=(const Hotel &) = delete;
33
34     ~Hotel();
35
36     static Date today() { return now; }
37
38     std::string getName() const { return name; }
39
40     void nextDay();
41
42     bool reserveRoom(unsigned number, DatePeriod period, std::string name = "-", std::string note =
43         "None\n");
44
45     Hotel &showAvailableRooms(std::ostream &, Date);
46
47     bool freeRoom(unsigned number);
48
49     Hotel &getReport(DatePeriod period);
50
51     void searchRoom(unsigned minBeds, DatePeriod period) const;
52
53     bool serviceRoom(unsigned number, DatePeriod period, std::string note);
54 };
55 #endif

```

4.3 HotelBuilding.hpp

```

1 #ifndef __HOTELBUILDING_HPP
2 #define __HOTELBUILDING_HPP
3 #include "Types.hpp"
4 #include "Room.hpp"
5 #include "RoomAnalyzer.hpp"
6 #include <fstream>
7
8 class HotelBuilding
9 {
10     Room **rooms;
11     size_t size;
12
13 public:
14     HotelBuilding(std::ifstream &if); // todo String
15     HotelBuilding(const HotelBuilding &other) = delete;
16     HotelBuilding &operator=(HotelBuilding &other) = delete;
17     ~HotelBuilding();
18
19     size_t getRoomCount() const { return size; }
20
21     Room *operator[](unsigned roomNumber) const;
22
23     void newDate(Date d);
24
25     void showAvailableRooms(std::ostream &os, Date d) const;
26
27     void createReport(DatePeriod period) const;
28
29     void suggestRoom(unsigned beds, DatePeriod period);
30
31     friend class RoomAnalyzer;
32 };
33
34 #endif

```

4.4 Reservation.hpp

```

1 #ifndef __RESERVATION_HPP

```

```

2 #define __RESERVATION_HPP
3 #include "Types.hpp"
4 #include "Room.hpp"
5 #include "Date.hpp"
6 #include "Hotel.hpp"
7 #include <cstring>
8 #include <cassert>
9 #include <fstream>
10 #include <string>
11
12 enum ReservationState
13 {
14     UNKNOWN = 0,
15     PAST,
16     ACTIVE,
17     FUTURE
18 };
19
20 class Reservation
21 {
22     std::string guestName;
23     std::string note;
24     DatePeriod period;
25     ReservationState state;
26     bool service;
27
28 public:
29     Reservation(std::string name, DatePeriod p, std::string n = "None.\n", bool s = false);
30     Reservation(const Reservation &) = delete;
31     Reservation &operator=(const Reservation &) = delete;
32
33     bool isActive() const { return state == ACTIVE; }
34     bool isPast() const { return state == PAST; }
35     bool isServiced() const { return service; }
36     Date getFrom() const { return period.from; }
37     Date getTo() const { return period.to; }
38     unsigned getNights() const { return period.length(); }
39     void onDate(Date d);
40     ReservationState stateOnDate(Date) const;
41     bool LeavingInAdvance(Date);
42 };
43
44 std::ostream &operator<<(std::ostream &, const Reservation &);
45
46 #endif

```

4.5 Room.hpp

```

1 #ifndef __ROOM_HPP
2 #define __ROOM_HPP
3 #include <iostream>
4 #include <string>
5 #include "Types.hpp"
6 #include "Reservation.hpp"
7 #include "Hotel.hpp"
8
9 const size_t INIT_CAPACITY = 2;
10
11 class Room
12 {
13     unsigned number;
14     unsigned bedCount;
15     Reservation **reservations;
16     size_t resCount, resCapacity;
17
18     Reservation **pastReservations;
19     size_t pastCount, pastCapacity;
20
21     void expand(Reservation **&arr, size_t &size, size_t &capacity);
22     void shrink(Reservation **&arr, size_t &size, size_t &capacity);
23
24     unsigned daysTakenInPeriod(DatePeriod period) const;
25
26     bool newReservation(std::string name, std::string note, DatePeriod period, bool service);
27 }

```

```

64 public:
65     Room(unsigned n, unsigned bC);
66     Room(const Room &) = delete;
67     Room &operator=(const Room &) = delete;
68     ~Room();
69
70     unsigned getNumber() const { return number; }
71     unsigned getBedCount() const { return bedCount; }
72     bool isFreeNow() const;
73
74     bool freeRoom(Reservation *&currentRes);
75     void changeLeaving(Reservation *, Date newDate); // todo must be private
76
77     void newDate(Date);
78
79     bool isFreeOnDate(Date) const;
80
81     bool isFreeInPeriod(DatePeriod period) const;
82
83     void showReservationsInPeriod(std::ostream &os, DatePeriod period) const;
84
85     bool addReservation(std::string name, std::string note, DatePeriod period);
86
87     bool closeForService(std::string note, DatePeriod period);
88 };
89
90 std::ostream &operator<<(std::ostream &os, const Room &R);
91
92 #endif

```

4.6 RoomAnalyzer.hpp

```

1 #ifndef __ROOMANALYZER_HPP
2 #define __ROOMANALYZER_HPP
3 #include "Types.hpp"
4 #include "HotelBuilding.hpp"
5 #include "Date.hpp"
6 const size_t DISPLAY = 5;
7
8 class RoomAnalyzer
9 {
10     static void sortRooms(HotelBuilding &hB, unsigned *score, size_t size);
11
12 public:
13     static void suggest(HotelBuilding &hB, unsigned beds, DatePeriod period);
14 };
15
16 #endif

```

4.7 Types.hpp

```

1 #ifndef __TYPES_HPP
2 #define __TYPES_HPP
3
4 class Date;
5 class Room;
6 class HotelBuilding;
7 class Reservation;
8 class RoomAnalyzer;
9 class Hotel;
10
11 #endif

```

Index

- Date, [5](#)
 - Date, [6](#)
 - getToday, [6](#)
 - operator<, [7](#)
 - operator<<, [8](#)
 - operator>, [8](#)
 - operator>>, [9](#)
 - operator(), [6](#)
 - operator++, [7](#)
 - operator-, [7](#)
 - operator==, [8](#)
- DatePeriod, [9](#)
- freeRoom
 - Hotel, [11](#)
 - Room, [18](#)
- getFrom
 - Reservation, [15](#)
- getName
 - Hotel, [11](#)
- getNights
 - Reservation, [15](#)
- getReport
 - Hotel, [11](#)
- getTo
 - Reservation, [15](#)
- getToday
 - Date, [6](#)
- Hotel, [10](#)
 - freeRoom, [11](#)
 - getName, [11](#)
 - getReport, [11](#)
 - Hotel, [10](#)
 - reserveRoom, [12](#)
 - serviceRoom, [12](#)
 - showAvailableRooms, [13](#)
 - today, [13](#)
- HotelBuilding, [13](#)
- isActive
 - Reservation, [15](#)
- isFreeOnDate
 - Room, [18](#)
- isPast
 - Reservation, [16](#)
- isServiced
 - Reservation, [16](#)
- LeavingInAdvance
 - Reservation, [16](#)
- onDate
 - Reservation, [16](#)
- operator<
 - Date, [7](#)
- operator<<
 - Date, [8](#)
- operator>
 - Date, [8](#)
- operator>>
 - Date, [9](#)
- operator()
 - Date, [6](#)
- operator++
 - Date, [7](#)
- operator-
 - Date, [7](#)
- operator==
 - Date, [8](#)
- Reservation, [14](#)
 - getFrom, [15](#)
 - getNights, [15](#)
 - getTo, [15](#)
 - isActive, [15](#)
 - isPast, [16](#)
 - isServiced, [16](#)
 - LeavingInAdvance, [16](#)
 - onDate, [16](#)
 - Reservation, [14](#)
 - stateOnDate, [17](#)
- reserveRoom
 - Hotel, [12](#)
- Room, [17](#)
 - freeRoom, [18](#)
 - isFreeOnDate, [18](#)
 - Room, [18](#)
- RoomAnalyzer, [19](#)
- serviceRoom
 - Hotel, [12](#)
- showAvailableRooms
 - Hotel, [13](#)
- stateOnDate
 - Reservation, [17](#)
- today
 - Hotel, [13](#)