Employee Database Classes and Files Documentation

Generated by Doxygen 1.9.7

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

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

Chapter 3

Class Documentation

3.1 Record Class Reference

Employee record class.

`#include <record.hpp>`

Public Types

- using EmpWorkDays = std::vector< std::string >

  Type alias for vector of employee working days (3-letter strings).
- using DaySet = std::set< std::string >

  Type alias for set of given working days (3-letter strings).

Public Member Functions

- Record (const std::string &name, const std::string &age, const std::string &dep, const std::string &pos, const std::string &boss, const EmpWorkDays &days)

  Initializing constructor of a new Record object.
- Record (const Record &other)

  Copy constructor of a new Record object.
- ~Record ()

  Destructor of the Record object.
- Record & operator= (const Record &rhv)

  Assignment operator overload.
- const std::string & getName () const

  Get employee name.
- size_t getAge () const

  Get employee age.
- const std::string & getDep () const

  Get employee department.
- const std::string & getPos () const

  Get employee position.
- const std::string & getBoss () const

  Get employee boss name.
- const EmpWorkDays & getDays () const

  Get employee working days.

Static Public Member Functions

- static void swap (Record &lhv, Record &rhv) noexcept
  Object swapper.

Private Attributes

- std::string _emp_name
  Employee name.
- size_t _emp_age
  Employee age.
- std::string _emp_dep
  Employee department.
- std::string _emp_pos
  Employee position.
- std::string _emp_boss
  Employee boss name.
- EmpWorkDays _emp_days
  Employee working days.

3.1.1 Detailed Description

Employee record class.

3.1.2 Member Typedef Documentation

3.1.2.1 DaySet

```
using Record::DaySet = std::set<std::string>
```

Type alias for set of given working days (3-letter strings).

3.1.2.2 EmpWorkDays

```
using Record::EmpWorkDays = std::vector<std::string>
```

Type alias for vector of employee working days (3-letter strings).

3.1.3 Constructor & Destructor Documentation

3.1.3.1 Record() [1/2]

```
Record::Record (
            const std::string & name,
            const std::string & age,
            const std::string & dep,
            const std::string & pos,
            const std::string & boss,
            const EmpWorkDays & days )
```

Initializing constructor of a new Record object.

Parameters

name	Name
age	Age
dep	Department
pos	Position
boss	Boss name
days	Working days

3.1.3.2 Record() [2/2]

```
Record::Record (
            const Record & other )
```

Copy constructor of a new Record object.

Parameters

other	Record object

3.1.3.3 ~Record()

```
Record::~Record ( )
```

Destructor of the Record object.

3.1.4 Member Function Documentation

3.1.4.1 getAge()

```
size_t Record::getAge ( ) const
```

Get employee age.

Returns

size_t

3.1.4.2 getBoss()

```
const std::string & Record::getBoss ( ) const
```

Get employee boss name.

Returns

const std::string&

3.1.4.3 getDays()

```
const Record::EmpWorkDays & Record::getDays ( ) const
```

Get employee working days.

Returns

const EmpWorkDays&

3.1.4.4 getDep()

```
const std::string & Record::getDep ( ) const
```

Get employee department.

Returns

const std::string&

3.1.4.5 getName()

```
const std::string & Record::getName ( ) const
```

Get employee name.

Returns

const std::string& const

3.1.4.6 getPos()

```
const std::string & Record::getPos ( ) const
```

Get employee position.

Returns

const std::string&

3.1.4.7 operator=()

```
Record & Record::operator= (
            const Record & rhv )
```

Assignment operator overload.

Parameters

rhv	Record object

Returns

Record&

3.1.4.8 swap()

```
void Record::swap (
            Record & lhv,
            Record & rhv )  [static], [noexcept]
```

Object swapper.

Parameters

lhv	Record object
rhv	Record object

3.1.5 Member Data Documentation

3.1.5.1 _emp_age

```
size_t Record::_emp_age  [private]
```

Employee age.

3.1.5.2 _emp_boss

```
std::string Record::_emp_boss  [private]
```

Employee boss name.

3.1.5.3 _emp_days

```
EmpWorkDays Record::_emp_days  [private]
```

Employee working days.

3.1.5.4 _emp_dep

```
std::string Record::_emp_dep  [private]
```

Employee department.

3.1.5.5 _emp_name

```
std::string Record::_emp_name  [private]
```

Employee name.

3.1.5.6 _emp_pos

```
std::string Record::_emp_pos  [private]
```

Employee position.

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

- record.hpp
- record.cpp

3.2 Register Class Reference

Employees register class.

```
#include <register.hpp>
```

Public Types

- using EmpVec = std::vector< const Record * >

  Type alias for vector of pointers to employee records.
- using EmpSet = std::set< const Record * >

  Type alias for set of pointers to employee records.
- using NameIdx = std::map< const std::string, const Record * >

  Type alias for employee name index.
- using DepPosIdx = std::map< const std::string, EmpVec >

  Type alias for employee department or position index.
- using SubordVec = std::vector< std::string >

  Type alias for employee direct subordinates vector.
- using SubordIdx = std::map< std::string, SubordVec >

  Type alias for employee direct subordinates index.

Public Member Functions

- Register ()

  Default constructor of a new Register object.
- Register (const Register &other)

  Copy constructor of an other Register object (deep copy).
- ~Register ()

  Destructor of the Register object.
- Register & operator= (const Register &rhv)

  Assignment operator overload.
- void clearRegister ()

  Deep clean the Register object.
- size_t getSize () const

  Get register size.
- void add (const Record *pRec)

  Add (push) a pointer to the employee record into employees vector and insert/update indices.
- const EmpVec & getStorage () const

  Get a vector of pointers to all employee records.
- const EmpVec getStorage (size_t age_l, size_t age_h) const

  Overload for getting a vector of pointers to employee records filtered by age range.
- const EmpSet getEmpByWorkDays (const Record::DaySet &days_to_check) const

  Get a set of pointers to employee records filtered by working days.
- const Record * getRecByName (const std::string &name)

  Get the pointer to an employee record by their name from index.
- const EmpVec & getEmpByDep (const std::string &dep)

  Get a vector of pointers to employee records filtered by department from index.
- const EmpVec & getEmpByPos (const std::string &pos)

  Get a vector of pointers to employee records filtered by position from index.
- const SubordVec & getSubordsByBoss (const std::string &boss)

  Get a vector of direct subordinates names by boss name from index.
- const NameIdx & getNameIdx () const

  Get employee name index.
- const DepPosIdx & getDepIdx () const

  Get employee department index.
- const DepPosIdx & getPosIdx () const

  Get employee position index.
- const SubordIdx & getSubordIdx () const

  Get employee direct subordinates index.

Static Public Member Functions

- static void swap (Register &lhv, Register &rhv) noexcept

  Object swapper.

Private Attributes

- EmpVec _employees

  Vector of pointers to employee records.
- NameIdx _name_idx

  Employee name index (map).
- DepPosIdx _dep_idx

  Employee department index (map).
- DepPosIdx _pos_idx

  Employee position index (map).
- SubordIdx _subord_idx

  Employee direct subordinates index (map).

3.2.1 Detailed Description

Employees register class.

3.2.2 Member Typedef Documentation

3.2.2.1 DepPosIdx

```
using Register::DepPosIdx = std::map<const std::string, EmpVec>
```

Type alias for employee department or position index.

3.2.2.2 EmpSet

```
using Register::EmpSet = std::set<const Record*>
```

Type alias for set of pointers to employee records.

3.2.2.3 EmpVec

```
using Register::EmpVec = std::vector<const Record*>
```

Type alias for vector of pointers to employee records.

3.2.2.4 NameIdx

```
using Register::NameIdx = std::map<const std::string, const Record*>
```

Type alias for employee name index.

3.2.2.5 SubordIdx

```
using Register::SubordIdx = std::map<std::string, SubordVec>
```

Type alias for employee direct subordinates index.

3.2.2.6 SubordVec

```
using Register::SubordVec = std::vector<std::string>
```

Type alias for employee direct subordinates vector.

3.2.3 Constructor & Destructor Documentation

3.2.3.1 Register() [1/2]

```
Register::Register ( )
```

Default constructor of a new Register object.

3.2.3.2 Register() [2/2]

```
Register::Register (
            const Register & other )
```

Copy constructor of an other Register object (deep copy).

Parameters

other	Register

3.2.3.3 ~Register()

```
Register::~Register ( )
```

Destructor of the Register object.

3.2.4 Member Function Documentation

3.2.4.1 add()

```
void Register::add (
            const Record * pRec )
```

Add (push) a pointer to the employee record into employees vector and insert/update indices.

Time complexity:

$O(\text{vector push back}) + \sum O(\text{map insert for each index}) = O(1) + \sum O(\log \text{Each index size})$

$= \sum O(\log \text{Each index size})$.

Parameters

pRec	Pointer to employee record object

3.2.4.2 clearRegister()

```
void Register::clearRegister ( )
```

Deep clean the Register object.

3.2.4.3 getDepIdx()

```
const Register::DepPosIdx & Register::getDepIdx ( ) const
```

Get employee department index.

Returns

const DepIdx&

3.2.4.4 getEmpByDep()

```
const Register::EmpVec & Register::getEmpByDep (
            const std::string & dep )
```

Get a vector of pointers to employee records filtered by department from index.

Time complexity: $O(\text{map lookup}) = O(\log \text{Department index size})$.

Parameters

dep	Employee department

Returns

const EmpVec&

3.2.4.5 getEmpByPos()

```
const Register::EmpVec & Register::getEmpByPos (
            const std::string & pos )
```

Get a vector of pointers to employee records filtered by position from index.

Time complexity: $O(\text{map lookup}) = O(\log \text{Position index size})$.

Parameters

pos	Employee position

Returns

const EmpVec&

3.2.4.6 getEmpByWorkDays()

```
const Register::EmpSet Register::getEmpByWorkDays (
            const Record::DaySet & days_to_check ) const
```

Get a set of pointers to employee records filtered by working days.

Time complexity:

$O(\text{Number of records}) \times O(\text{Number of days to check} \ll \text{Number of records})$

$\times [O(\text{vector find in Employee working days} \ll \text{Number of records}) + O(\text{set insert})]$

$\approx O(\text{Number of records}) \times O(\log \text{Number of records})$

$= O(\text{Number of records} \log \text{Number of records})$.

Parameters

days_to_check	Vector of given working days (3-letter strings) to check against

Returns

const EmpSet

3.2.4.7 getNameIdx()

```
const Register::NameIdx & Register::getNameIdx ( ) const
```

Get employee name index.

Returns

const NameIdx&

3.2.4.8 getPosIdx()

```
const Register::DepPosIdx & Register::getPosIdx ( ) const
```

Get employee position index.

Returns

const PosIdx&

3.2.4.9 getRecByName()

```
const Record * Register::getRecByName (
            const std::string & name )
```

Get the pointer to an employee record by their name from index.

Time complexity: $O(\text{map lookup}) = O(\log \text{Name index size})$.

Parameters

name	Employee name

Returns

Record*

3.2.4.10 getSize()

```
size_t Register::getSize ( ) const
```

Get register size.

Returns

size_t

3.2.4.11 getStorage() [1/2]

```
const Register::EmpVec & Register::getStorage ( ) const
```

Get a vector of pointers to all employee records.

Returns

const EmpVec&

3.2.4.12 getStorage() [2/2]

```
const Register::EmpVec Register::getStorage (
            size_t age_l,
            size_t age_h ) const
```

Overload for getting a vector of pointers to employee records filtered by age range.

Time complexity:

$O(\text{Number of records}) \times O(\text{vector push back}) = O(\text{Number of records}) \times O(1)$

$= O(\text{Number of records})$.

Parameters

age↩ _l	Lower age limit
age↩ _h	Higher age limit

Returns

const EmpVec

3.2.4.13 getSubordIdx()

```
const Register::SubordIdx & Register::getSubordIdx ( ) const
```

Get employee direct subordinates index.

Returns

const SubordIdx&

3.2.4.14 getSubordsByBoss()

```
const Register::SubordVec & Register::getSubordsByBoss (
            const std::string & boss )
```

Get a vector of direct subordinates names by boss name from index.

Time complexity: $O(\text{map lookup}) = O(\log \text{Subordinates index size})$.

Parameters

boss	Employee boss name

Returns

const SubordVec&

3.2.4.15 operator=()

```
Register & Register::operator= (
            const Register & rhv )
```

Assignment operator overload.

Parameters

rhv	Register object

Returns

Register&

3.2.4.16 swap()

```
void Register::swap (
            Register & lhv,
            Register & rhv )  [static], [noexcept]
```

Object swapper.

Parameters

lhv	Register object
rhv	Register object

3.2.5 Member Data Documentation

3.2.5.1 _dep_idx

```
DepPosIdx Register::_dep_idx  [private]
```

Employee department index (map).

3.2.5.2 _employees

```
EmpVec Register::_employees  [private]
```

Vector of pointers to employee records.

3.2.5.3 _name_idx

```
NameIdx Register::_name_idx  [private]
```

Employee name index (map).

3.2.5.4 _pos_idx

```
DepPosIdx Register::_pos_idx  [private]
```

Employee position index (map).

3.2.5.5 _subord_idx

```
SubordIdx Register::_subord_idx  [private]
```

Employee direct subordinates index (map).

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

- register.hpp
- register.cpp

Chapter 4

File Documentation

4.1 printers.hpp File Reference

Typedefs

- using VisMap = std::map< std::string, bool >
  Type alias for map of visited employee flags.

Functions

- void printMenu ()
  Show the main menu.
- std::ostream & operator<< (std::ostream &s, const Record::EmpWorkDays &d)
  Print employee working days.
- std::ostream & operator<< (std::ostream &s, const Record *pRec)
  Print one record as a table row (long).
- void printOneRecord (const Record *pRec)
  Print one record as a standalone card (tall).
- void printRecNum (const Register &r)
  Print number of employees records in current register.
- void printEmpHeader ()
  Print header row for employees records table.
- void printIdxKeys (const Register::DepPosIdx &idx)
  Print keys of employee department or position index.
- void dfs (const std::string &subord_name, VisMap &visited, size_t level, Register &r)
  Visit every employee once. Recursive depth-first search (DFS) algorithm.
- void printSubordsByBoss (const std::string &boss, Register &r)
  Print all direct and indirect (recursively) subordinates from index.
- template<typename T >
  void printEmpCollection (const T &emps)
  Template for printing all employee records from the given collection of pointers.

4.1.1 Detailed Description

Author

Andrei Batyrov (arbatyrov@edu.hse.ru)

Version

0.1

Date

2024-02-27

Copyright

4.1.2 Typedef Documentation

4.1.2.1 VisMap

```
using VisMap = std::map<std::string, bool>
```

Type alias for map of visited employee flags.

4.1.3 Function Documentation

4.1.3.1 dfs()

```
void dfs (
            const std::string & subord_name,
            VisMap & visited,
            size_t level,
            Register & r )
```

Visit every employee once. Recursive depth-first search (DFS) algorithm.

Parameters

subord_name	Subordinate employee name
visited	Map storing visited employee flags
level	Level of subordination
r	Register object

4.1.3.2 operator<<() [1/2]

```
std::ostream & operator<< (
            std::ostream & s,
            const Record * pRec )
```

Print one record as a table row (long).

Parameters

s	Output stream
pRec	Pointer to the employee record object

Returns

std::ostream&

4.1.3.3 operator<<() [2/2]

```
std::ostream & operator<< (
            std::ostream & s,
            const Record::EmpWorkDays & d )
```

Print employee working days.

Parameters

s	Output stream
d	Vector of working days (3-letter strings)

Returns

std::ostream&

4.1.3.4 printEmpCollection()

```
template<typename T >
void printEmpCollection (
            const T & emps )
```

Template for printing all employee records from the given collection of pointers.

This template is needed to avoid having several identical overloaded functions for printing employee collections stored in different types, such as vectors or sets, having the same iteration mechanism.

Note: This project was compiled for C++11. Starting from C++20, the `auto` keyword is supported in declarations, so we could simply declare `void printEmpCollection(const auto& emps)` and the type would be deduced automatically from the actual type of the `emps` container, and this template would not be required.

Template Parameters

T	Type of container of pointers to employee records

Parameters

emps	Container of pointers to employee records

4.1.3.5 printEmpHeader()

```
void printEmpHeader ( )
```

Print header row for employees records table.

4.1.3.6 printIdxKeys()

```
void printIdxKeys (
            const Register::DepPosIdx & idx )
```

Print keys of employee department or position index.

Parameters

idx	Employee index

4.1.3.7 printMenu()

```
void printMenu ( )
```

Show the main menu.

4.1.3.8 printOneRecord()

```
void printOneRecord (
            const Record * pRec )
```

Print one record as a standalone card (tall).

Parameters

pRec	Pointer to the employee record object

4.1.3.9 printRecNum()

```
void printRecNum (
            const Register & r )
```

Print number of employees records in current register.

Parameters

r	Register object

4.1.3.10 printSubordsByBoss()

```
void printSubordsByBoss (
            const std::string & boss,
            Register & r )
```

Print all direct and indirect (recursively) subordinates from index.

Parameters

boss	Employee boss name
r	Register object

4.2 printers.hpp

Go to the documentation of this file.

```
00001
00012 #ifndef PRINTERS_HPP
00013 #define PRINTERS_HPP
00014
00019 using VisMap = std::map<std::string, bool>;
00020
00025 void printMenu();
00026
00034 std::ostream& operator«(std::ostream& s, const Record::EmpWorkDays& d);
00035
00043 std::ostream& operator«(std::ostream& s, const Record* pRec);
00044
00050 void printOneRecord(const Record* pRec);
00051
00057 void printRecNum(const Register& r);
00058
00063 void printEmpHeader();
00064
00070 void printIdxKeys(const Register::DepPosIdx& idx);
00071
00080 void dfs(const std::string& subord_name, VisMap& visited, size_t level, Register& r);
00081
00088 void printSubordsByBoss(const std::string& boss, Register& r);
00089
00104 template <typename T>
00105 void printEmpCollection(const T& emps);
00106
00107 #endif // PRINTERS_HPP
```

4.3 record.hpp File Reference

```
#include <string>
#include <vector>
#include <set>
```

Classes

- class Record

  Employee record class.

4.3.1 Detailed Description

Author

Andrei Batyrov (arbatyrov@edu.hse.ru)

Version

0.1

Date

2024-02-27

Copyright

Copyright (c) 2024

4.4 record.hpp

Go to the documentation of this file.

```
00001
00012 #ifndef RECORD_HPP
00013 #define RECORD_HPP
00014
00015 #include <string>
00016 #include <vector>
00017 #include <set>
00018
00023 class Record
00024 {
00025     public:
00026
00031         using EmpWorkDays = std::vector<std::string>;
00032
00037         using DaySet = std::set<std::string>;
00038
00049         Record(const std::string& name,
00050                const std::string& age,
00051                const std::string& dep,
00052                const std::string& pos,
00053                const std::string& boss,
00054                const EmpWorkDays& days
00055                );
00056
00062         Record(const Record& other);
00063
```

```
00068         ~Record();
00069
00076         Record& operator=(const Record& rhv);
00077
00084         static void swap(Record& lhv, Record& rhv) noexcept;
00085
00091         const std::string& getName() const;
00092
00098         size_t getAge() const;
00099
00105         const std::string& getDep() const;
00106
00112         const std::string& getPos() const;
00113
00119         const std::string& getBoss() const;
00120
00126         const EmpWorkDays& getDays() const;
00127
00128     private:
00129
00134         std::string _emp_name;
00135
00140         size_t _emp_age;
00141
00146         std::string _emp_dep;
00147
00152         std::string _emp_pos;
00153
00158         std::string _emp_boss;
00159
00164         EmpWorkDays _emp_days;
00165 };
00166
00167 #endif // RECORD_HPP
```

4.5 register.hpp File Reference

```
#include <string>
#include <vector>
#include <map>
#include <set>
#include ¨record.hpp¨
```

Classes

- class Register

  Employees register class.

4.5.1 Detailed Description

Author

Andrei Batyrov (arbatyrov@edu.hse.ru)

Version

0.1

Date

2024-02-27

Copyright

4.6 register.hpp

Go to the documentation of this file.

```
00001
00012 #ifndef REGISTER_HPP
00013 #define REGISTER_HPP
00014
00015 #include <string>
00016 #include <vector>
00017 #include <map>
00018 #include <set>
00019 #include "record.hpp"
00020
00025 class Register
00026 {
00027
00028     public:
00029
00034         using EmpVec = std::vector<const Record*>;
00035
00040         using EmpSet = std::set<const Record*>;
00041
00046         using NameIdx = std::map<const std::string, const Record*>;
00047
00052         using DepPosIdx = std::map<const std::string, EmpVec>;
00053
00058         using SubordVec = std::vector<std::string>;
00059
00064         using SubordIdx = std::map<std::string, SubordVec>;
00065
00070         Register();
00071
00077         Register(const Register& other);
00078
00083         ~Register();
00084
00091         Register& operator=(const Register& rhv);
00092
00099         static void swap(Register& lhv, Register& rhv) noexcept;
00100
00106         void clearRegister();
00107
00113         size_t getSize() const;
00114
00126         void add(const Record* pRec);
00127
00133         const EmpVec& getStorage() const;
00134
00148         const EmpVec getStorage(size_t age_l, size_t age_h) const;
00149
00166         const EmpSet getEmpByWorkDays(const Record::DaySet& days_to_check) const;
00167
00176         const Record* getRecByName(const std::string& name);
00177
00186         const EmpVec& getEmpByDep(const std::string& dep);
00187
00196         const EmpVec& getEmpByPos(const std::string& pos);
00197
00206         const SubordVec& getSubordsByBoss(const std::string& boss);
00207
00213         const NameIdx& getNameIdx() const;
00214
00220         const DepPosIdx& getDepIdx() const;
00221
00227         const DepPosIdx& getPosIdx() const;
00228
00234         const SubordIdx& getSubordIdx() const;
00235
00236     private:
00237
00242         EmpVec _employees;
00243
00248         NameIdx _name_idx;
00249
00254         DepPosIdx _dep_idx;
00255
00260         DepPosIdx _pos_idx;
00261
00266         SubordIdx _subord_idx;
00267 };
00268
00269 #endif // REGISTER_HPP
```

4.7 main.cpp File Reference

```
#include <iostream>
#include <fstream>
#include <sstream>
#include "record.hpp"
#include "register.hpp"
#include "printers.hpp"
```

Functions

- int **main** ()

4.7.1 Detailed Description

Author

Andrei Batyrov (arbatyrov@edu.hse.ru)

Version

0.1

Date

2024-02-29

Copyright

4.8 printers.cpp File Reference

```
#include <iostream>
#include <iomanip>
#include <map>
#include "record.hpp"
#include "register.hpp"
#include "printers.hpp"
```

Functions

- void printMenu ()

  Show the main menu.
- std::ostream & operator<< (std::ostream &s, const Record::EmpWorkDays &d)

  Print employee working days.
- std::ostream & operator<< (std::ostream &s, const Record *pRec)

  Print one record as a table row (long).
- void printOneRecord (const Record *pRec)

  Print one record as a standalone card (tall).
- void printRecNum (const Register &r)

  Print number of employees records in current register.
- void printEmpHeader ()

  Print header row for employees records table.
- void printIdxKeys (const Register::DepPosIdx &idx)

  Print keys of employee department or position index.
- void dfs (const std::string &subord_name, VisMap &visited, size_t level, Register &r)

  Visit every employee once. Recursive depth-first search (DFS) algorithm.
- void printSubordsByBoss (const std::string &boss, Register &r)

  Print all direct and indirect (recursively) subordinates from index.
- template<typename T >
  void printEmpCollection (const T &emps)

  Template for printing all employee records from the given collection of pointers.
- template void **printEmpCollection**< **Register::EmpVec** > (const Register::EmpVec &)
- template void **printEmpCollection**< **Register::EmpSet** > (const Register::EmpSet &)

4.8.1 Detailed Description

Author

Andrei Batyrov (arbatyrov@edu.hse.ru)

Version

0.1

Date

2024-02-27

Copyright

Copyright (c) 2024

4.8.2 Function Documentation

4.8.2.1 dfs()

```
void dfs (
            const std::string & subord_name,
            VisMap & visited,
            size_t level,
            Register & r )
```

Visit every employee once. Recursive depth-first search (DFS) algorithm.

Parameters

subord_name	Subordinate employee name
visited	Map storing visited employee flags
level	Level of subordination
r	Register object

4.8.2.2 operator<<() [1/2]

```
std::ostream & operator<< (
            std::ostream & s,
            const Record * pRec )
```

Print one record as a table row (long).

Parameters

s	Output stream
pRec	Pointer to the employee record object

Returns

std::ostream&

4.8.2.3 operator<<() [2/2]

```
std::ostream & operator<< (
            std::ostream & s,
            const Record::EmpWorkDays & d )
```

Print employee working days.

Parameters

s	Output stream
d	Vector of working days (3-letter strings)

Returns

std::ostream&

4.8.2.4 printEmpCollection()

```
template<typename T >
void printEmpCollection (
            const T & emps )
```

Template for printing all employee records from the given collection of pointers.

This template is needed to avoid having several identical overloaded functions for printing employee collections stored in different types, such as vectors or sets, having the same iteration mechanism.

Note: This project was compiled for C++11. Starting from C++20, the `auto` keyword is supported in declarations, so we could simply declare `void printEmpCollection(const auto& emps)` and the type would be deduced automatically from the actual type of the `emps` container, and this template would not be required.

Template Parameters

T	Type of container of pointers to employee records

Parameters

emps	Container of pointers to employee records

4.8.2.5 printEmpHeader()

```
void printEmpHeader ( )
```

Print header row for employees records table.

4.8.2.6 printIdxKeys()

```
void printIdxKeys (
            const Register::DepPosIdx & idx )
```

Print keys of employee department or position index.

Parameters

idx	Employee index

4.8.2.7 printMenu()

```
void printMenu ( )
```

Show the main menu.

4.8.2.8 printOneRecord()

```
void printOneRecord (
            const Record * pRec )
```

Print one record as a standalone card (tall).

Parameters

pRec	Pointer to the employee record object

4.8.2.9 printRecNum()

```
void printRecNum (
            const Register & r )
```

Print number of employees records in current register.

Parameters

r	Register object

4.8.2.10 printSubordsByBoss()

```
void printSubordsByBoss (
            const std::string & boss,
            Register & r )
```

Print all direct and indirect (recursively) subordinates from index.

Parameters

boss	Employee boss name
r	Register object

4.9 record.cpp File Reference

```
#include ¨record.hpp¨
```

4.9.1 Detailed Description

Author

Andrei Batyrov (arbatyrov@edu.hse.ru)

Version

0.1

Date

2024-02-27

Copyright

Copyright (c) 2024

4.10 register.cpp File Reference

```
#include <stdexcept>
#include <algorithm>
#include ¨register.hpp¨
```

4.10.1 Detailed Description

Author

Andrei Batyrov (arbatyrov@edu.hse.ru)

Version

0.1

Date

2024-02-27

Copyright

Copyright (c) 2024

Index