PrototypeV2

Generated by Doxygen 1.12.0

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 std::hash< Key > Struct Reference

Public Member Functions

- size_t **operator()** (const Key &k) const

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

- CSVParser.h

3.2 Nurse Struct Reference

A structure that stores information about a nurse, including their name, ID number, pay, and work shifts.

`#include <NurseList.h>`

Public Member Functions

- Nurse (const string &nurseName, int number, const string &nurseType, const string &nurseDepartment, const vector< string > &nurseShifts)

  Constructs a Nurse object.

Public Attributes

- string nurseName
- int nurseNumber
- string nurseType
- string nurseDepartment
- vector< string > nurseShifts
- Nurse * next
- Nurse * prev

3.2.1 Detailed Description

A structure that stores information about a nurse, including their name, ID number, pay, and work shifts.

The Nurse structure holds details about a nurse and is used in the doubly linked list for storing nurse records.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Nurse()

```
Nurse::Nurse (
            const string & nurseName,
            int number,
            const string & nurseType,
            const string & nurseDepartment,
            const vector< string > & nurseShifts)
```

Constructs a Nurse object.

Constructs a Nurse object with the given name, number, pay, and shift data.

Parameters

nurseName	The nurse's name.
number	The nurse's unique ID number.
nurseType	
nurseDept	
nurseName	The name of the nurse.
number	The unique identifier for the nurse.
nursePay	The pay of the nurse (hourly or annual).
nurseShifts	The shifts assigned to the nurse in a vector of strings.

3.2.3 Member Data Documentation

3.2.3.1 next

```
Nurse* Nurse::next
```

Pointer to the next nurse in the linked list.

3.2.3.2 nurseDepartment

```
string Nurse::nurseDepartment
```

Nurse's dept.

3.2.3.3 nurseName

```
string Nurse::nurseName
```

Nurse's name.

3.2.3.4 nurseNumber

```
int Nurse::nurseNumber
```

Unique nurse identifier (e.g., employee number).

3.2.3.5 nurseShifts

```
vector< string > Nurse::nurseShifts
```

A vector containing the nurse's assigned shifts (e.g., 42 shifts).

3.2.3.6 nurseType

```
string Nurse::nurseType
```

Nurse's type.

3.2.3.7 prev

```
Nurse* Nurse::prev
```

Pointer to the previous nurse in the linked list.

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

- NurseList.h
- NurseList.cpp

3.3 NurseList Class Reference

A doubly linked list class for storing and managing a collection of nurses.

```
#include <NurseList.h>
```

Public Member Functions

- NurseList ()

  Constructs an empty NurseList object.
- void addNurse (const string &nurseName, int number, const string &nurseType, const string &nurseDepartment, const vector< string > &nurseShifts)

  Adds a new nurse to the list.
- void display () const

  Displays all nurses in the list.
- Nurse * **getHead** () const
- void sortNursesByShift (int shiftIndex)

  Sorts nurses by satisfaction number for a given shift.
- void **swap** (Nurse *a, Nurse *b)

3.3.1 Detailed Description

A doubly linked list class for storing and managing a collection of nurses.

The NurseList class allows adding nurses to a doubly linked list and displaying the list.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 NurseList()

```
NurseList::NurseList ()
```

Constructs an empty NurseList object.

Initializes an empty doubly linked list where both head and tail pointers are set to nullptr.

The constructor initializes the head and tail pointers to nullptr, indicating that the list is empty.

3.3.3 Member Function Documentation

3.3.3.1 addNurse()

```
void NurseList::addNurse (
            const string & nurseName,
            int number,
            const string & nurseType,
            const string & nurseDepartment,
            const vector< string > & nurseShifts)
```

Adds a new nurse to the list.

Adds a new nurse to the end of the list.

Parameters

nurseName	The nurse's name.
number	The nurse's unique ID number.
nurseType	
nurseDept	

The new nurse is added to the end of the doubly linked list.

Parameters

nurseName	The nurse's name.
number	The nurse's unique ID number.
nursePay	The nurse's hourly or annual pay.
nurseShifts	A vector of shifts assigned to the nurse.

This method dynamically allocates memory for a new nurse and appends the nurse to the end of the list. If the list is empty, the new nurse becomes both the head and the tail of the list. Otherwise, the new nurse is appended to the tail.

3.3.3.2 display()

```
void NurseList::display () const
```

Displays all nurses in the list.

This function traverses the list and prints out details of each nurse.

Iterates through the list starting at the head and outputs the information for each nurse

3.3.3.3 sortNursesByShift()

```
void NurseList::sortNursesByShift (
            int shiftIndex)
```

Sorts nurses by satisfaction number for a given shift.

Parameters

shiftIndex	Shift number to sort satisfaction by

This function sorts nurses from most willing to least willing to work a given shift, which allows us to pick nurses with a higher satisfaction score

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

- NurseList.h
- NurseList.cpp

Chapter 4

File Documentation

4.1 CSVParser.h

```
00001 // CSVParser.h
00002 #ifndef CSV_PARSER_H
00003 #define CSV_PARSER_H
00004
00005 #include <unordered_map>
00006 #include <string>
00007 #include <tuple>
00008 #include "NurseList.h"
00009 using namespace std;
00010
00020 void loadNurses( const string& filename , unordered_map<string , NurseList>& nurse_lists );
00021
00022
00023 using Key = tuple<string, int, string>; // (dept, shift #, nurseType)
00024 using NurseMap = unordered_map<Key, int>;         // Map to store # needed for each (dept, shift,
      nurseType)
00025
00026 // Hash function for tuple so it can be used in an unordered_map
00027 namespace std {
00028     template <>
00029     struct hash<Key> {
00030         size_t operator()(const Key& k) const;
00031     };
00032 }
00033
00043 void loadConstraints(const string& filename, NurseMap& nurse_map);
00044
00053 void displayNurseMap(const NurseMap& nurse_map);
00054
00055 #endif // CSV_PARSER_H
```

4.2 NurseFunctions.h

```
00001 #ifndef NURSE_FUNCTIONS_H
00002 #define NURSE_FUNCTIONS_H
00003
00004 #include <unordered_map>
00005 #include <string>
00006 #include <vector>
00007 #include "NurseList.h"
00008 using namespace std;
00009
00022 vector<Nurse*> selectBestNurses(const unordered_map<string, NurseList>& nurse_lists,
00023                                 const vector<vector<string>& constraints,
00024                                 int shift_number, const string& department);
00025
00026
00037 double calculateAverageSatisfaction(const std::vector<Nurse*>& assigned_nurses, int shift);
00038
00039 #endif // NURSE_FUNCTIONS_H
```

4.3 NurseList.h

```
00001 // NurseList.h
00002 #ifndef NURSE_LIST_H
00003 #define NURSE_LIST_H
00004
00005 #include <iostream>
00006 #include <vector>
00007 #include <string>
00008 using namespace std;
00009
00016 struct Nurse {
00017
00018     string nurseName;
00019     int nurseNumber;
00020     string nurseType;
00021     string nurseDepartment;
00022     vector< string > nurseShifts;
00023     Nurse* next;
00024     Nurse* prev;
00034     Nurse( const string& nurseName , int number , const string& nurseType , const string&
      nurseDepartment, const vector< string >& nurseShifts );
00035
00036 };
00037
00044 class NurseList {
00045 public:
00051     NurseList();
00052
00063     void addNurse( const string& nurseName , int number , const string& nurseType , const string&
      nurseDepartment, const vector< string >& nurseShifts );
00064
00070     void display() const;
00071
00072     Nurse* getHead() const;
00073
00082     void sortNursesByShift(int shiftIndex);
00083
00084     void swap(Nurse* a, Nurse* b);
00085
00086 private:
00087
00088     Nurse* head;
00089     Nurse* tail;
00091 };
00092
00093 #endif // NURSE_LIST_H
```