

A School Directory Application

Generated by Doxygen 1.9.6

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Entry Class Reference	7
4.1.1 Detailed Description	8
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 Entry() [1/2]	8
4.1.2.2 Entry() [2/2]	9
4.1.2.3 ~Entry()	9
4.1.3 Member Function Documentation	9
4.1.3.1 comesBefore() [1/2]	10
4.1.3.2 comesBefore() [2/2]	10
4.1.3.3 equals() [1/2]	11
4.1.3.4 equals() [2/2]	11
4.1.3.5 print()	12
4.1.4 Friends And Related Function Documentation	13
4.1.4.1 operator<<	13
4.2 Faculty Class Reference	13
4.2.1 Detailed Description	14
4.2.2 Constructor & Destructor Documentation	15
4.2.2.1 Faculty()	15
4.2.3 Member Function Documentation	17
4.2.3.1 print()	17
4.3 SchoolDirectory Class Reference	17
4.3.1 Detailed Description	17
4.3.2 Constructor & Destructor Documentation	18
4.3.2.1 SchoolDirectory()	18
4.3.3 Member Function Documentation	18
4.3.3.1 add()	18
4.3.3.2 lookup()	19
4.3.3.3 print()	20
4.4 Staff Class Reference	20
4.4.1 Detailed Description	21
4.4.2 Constructor & Destructor Documentation	22
4.4.2.1 Staff()	22
4.4.3 Member Function Documentation	23

4.4.3.1 print()	23
4.5 Student Class Reference	24
4.5.1 Detailed Description	25
4.5.2 Constructor & Destructor Documentation	26
4.5.2.1 Student()	26
4.5.3 Member Function Documentation	26
4.5.3.1 print()	26
5 File Documentation	29
5.1 Entry.cpp File Reference	29
5.1.1 Function Documentation	29
5.1.1.1 main()	29
5.1.1.2 operator<<()	30
5.2 Entry.h File Reference	30
5.3 Entry.h	31
5.4 Faculty.cpp File Reference	31
5.5 Faculty.h File Reference	32
5.6 Faculty.h	32
5.7 SchoolDirectory.cpp File Reference	33
5.7.1 Function Documentation	33
5.7.1.1 main()	33
5.8 Staff.cpp File Reference	34
5.9 Staff.h File Reference	34
5.10 Staff.h	35
5.11 Student.cpp File Reference	36
5.12 Student.h File Reference	36
5.13 Student.h	37
Index	39

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Entry	7
Faculty	13
Staff	20
Student	24
SchoolDirectory	17

Chapter 2

Class Index

2.1 Class List

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

Entry	7
Faculty	13
SchoolDirectory	17
Staff	20
Student	24

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Entry.cpp	29
Entry.h	30
Faculty.cpp	31
Faculty.h	32
SchoolDirectory.cpp	33
Staff.cpp	34
Staff.h	34
Student.cpp	36
Student.h	36

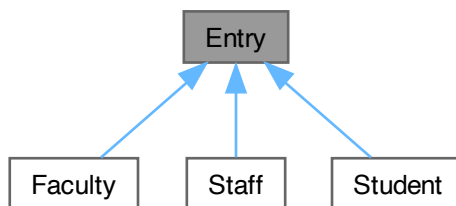
Chapter 4

Class Documentation

4.1 Entry Class Reference

```
#include <Entry.h>
```

Inheritance diagram for Entry:



Public Member Functions

- [Entry](#) ()
- [Entry](#) (std::string first, std::string last, std::string eAddress)
- bool [equals](#) (std::string first, std::string second)
- bool [equals](#) ([Entry](#) otherEntry)
- bool [comesBefore](#) (std::string first, std::string second)
- bool [comesBefore](#) ([Entry](#) otherEntry)
- virtual std::ostream & [print](#) (std::ostream &os) const
- virtual [~Entry](#) ()

Friends

- std::ostream & [operator<<](#) (std::ostream &os, const [Entry](#) &ent)

4.1.1 Detailed Description

Remarks

Entry: a base class for entries within a School Directory * A base entry holds a first name, last name, and email address *

•

Base capabilities include: * Base constructors * Comparison operations depend upon last names, then first names * Formatted output *

•

: files include header ([Entry.h](#)) and Implementation ([Entry.cpp](#)) *

•

Uncomment a main program for unit testing *

•

Author

Henry M. Walker *

Date

January 11, 2023 *

•

Remarks

References *

A School Directory as an Example of Object-Oriented Design * <http://localhost/courses/cpp-style-guide/c.php> *

•

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Entry() [1/2]

`Entry::Entry ()`

Remarks

Default constructor (with no parameters) *

Remarks

Entry: a base class for entries within a School Directory * A base entry holds a first name, last name, and email address *

•

Base capabilities include: * Base constructors * Comparison operations depend upon last names, then first names * Formatted output *

•

: files include header ([Entry.h](#)) and Implementation ([Entry.cpp](#)) *

•

Uncomment a main program for unit testing *

•

Author

Henry M. Walker *

Date

January 11, 2023 *

•

Remarks

References *

A School Directory as an Example of Object-Oriented Design * <http://localhost/courses/cpp-style-guide/c.php> *

•

Default constructor (with no parameters) *

4.1.2.2 Entry() [2/2]

```
Entry::Entry (
    std::string first,
    std::string last,
    std::string eAddress )
```

Remarks

Full-parameter constructor *

•

Parameters

<i>first</i>	a person's first name *
<i>last</i>	a person's last name *
<i>eAddress</i>	a person's email address *
	•

4.1.2.3 ~Entry()

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

Remarks

a Default destructor *

identify as virtual, since [Entry](#) has virtual functions ***Remarks**

a Default descructor *

4.1.3 Member Function Documentation

4.1.3.1 comesBefore() [1/2]

```
bool Entry::comesBefore (
    Entry otherEntry )
```

Remarks

check if this object comes before the parameter object *

-

Parameters

<i>otherEntry</i>	an entry to be compared with this object *
	•

Returns

true if [Entry](#)'s first/last names come before parameter's names * in directory order *

4.1.3.2 comesBefore() [2/2]

```
bool Entry::comesBefore (
    std::string first,
    std::string second )
```

Remarks

check if this object comes before the given first/last names *

-

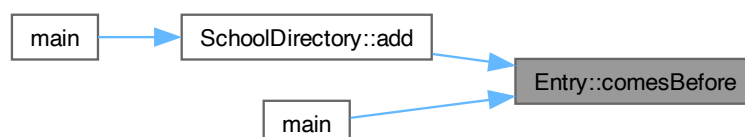
Parameters

<i>first</i>	a person's first name *
<i>last</i>	a person's last name *
	•

Returns

true if [Entry](#)'s first/last names come before parameter names * in directory order *

Here is the caller graph for this function:



4.1.3.3 equals() [1/2]

```
bool Entry::equals (
    Entry otherEntry )
```

Remarks

- check whether first and last names of two Entries match *

Parameters

<i>otherEntry</i>	an entry to be compared with this object *
-------------------	--

-

Returns

- true if this Entry's names match those of the parameter *
-

Remarks

check if this object comes before the given first/last names *

Parameters

<i>first</i>	a person's first name *
<i>last</i>	a person's last name *

-

Returns

- true if [Entry](#)'s first/last names come before parameter names * in directory order *

4.1.3.4 equals() [2/2]

```
bool Entry::equals (
    std::string first,
    std::string second )
```

Remarks

check whether first and last name of an [Entry](#) match two strings*

-

Parameters

<i>first</i>	a person's first name *
<i>last</i>	a person's last name *
	•

Returns

true if [Entry](#) names match first and last name strings *

-

Here is the caller graph for this function:

**4.1.3.5 print()**

```
std::ostream & Entry::print (
    _____ std::ostream & os ) const [virtual]
```

Remarks

output a format [Entry](#) object *

-

Parameters

<i>os</i>	output stream which will receive the formatted Entry data * <ul style="list-style-type: none"> • @ewmrk by being a virtual function, implementations in subclasses * will be interpreted via polymorphism * •
-----------	---

Returns

formatted string on the given output stream *

-

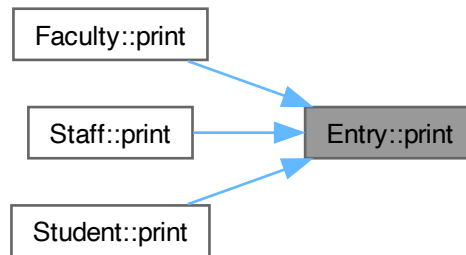
Remarks

use of a `to_string` function with a string return type * would require allocating space for a long string, * yielding a potential memory leak *

•

Reimplemented in [Faculty](#), [Staff](#), and [Student](#).

Here is the caller graph for this function:



4.1.4 Friends And Related Function Documentation

4.1.4.1 operator<<

```
std::ostream & operator<< (  
    std::ostream & os,  
    const Entry & ent ) [friend]
```

Remarks

overload << for printing an [Entry](#) *

•

use of the virtual print method allows tailored output * by subclasses *

•

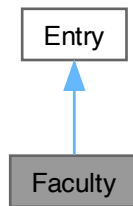
The documentation for this class was generated from the following files:

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

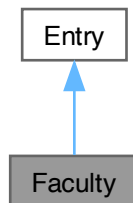
4.2 Faculty Class Reference

```
#include <Faculty.h>
```

Inheritance diagram for Faculty:



Collaboration diagram for Faculty:



Public Member Functions

- [Faculty](#) (std::string first, std::string last, std::string addr, std::string room, int ext, std::string department, int yr)
- virtual std::ostream & [print](#) (std::ostream &os) const

Public Member Functions inherited from [Entry](#)

- [Entry](#) ()
- [Entry](#) (std::string first, std::string last, std::string eAddress)
- bool [equals](#) (std::string first, std::string second)
- bool [equals](#) ([Entry](#) otherEntry)
- bool [comesBefore](#) (std::string first, std::string second)
- bool [comesBefore](#) ([Entry](#) otherEntry)
- virtual std::ostream & [print](#) (std::ostream &os) const
- virtual [~Entry](#) ()

4.2.1 Detailed Description

Remarks

[Faculty](#): a class derived from [Entry](#) for a School Directory * faculty entry * inherits a first name, last name, and email address from [Entry](#) * additional fields are the faculty member's office, extension, * department, and year of initial appointment to the school *

•

Inherited capabilities include: * Base constructors * Comparison operations depend upon last names, then first names * Formatted output *
 *

Overwritten capabilities include: * Multi-parameter constructor * Formatted print method *
 *

: files include header ([Faculty.h](#)), Implementation ([Faculty.cpp](#))*
 *

Uncomment a main program for unit testing *
 *

Author

Henry M. Walker *

Date

January 11, 2023 *
 *

Remarks

References *

A School Directory as an Example of Object-Oriented Design * <http://localhost/courses/cpp-style-guide/01-01-01.php> *
 *

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Faculty()

```
Faculty::Faculty (
    std::string first,
    std::string last,
    std::string addr,
    std::string room,
    int ext,
    std::string department,
    int yr )
```

Remarks

Full-parameter constructor *
 *

Parameters

<i>first</i>	a faculty member's first name *
<i>last</i>	a faculty member's last name *
<i>eAddress</i>	a faculty member's email address *
<i>room</i>	a faculty member's office *
<i>ext</i>	the telephone number extension for the office *
<i>department</i>	the faculty member's [primary] department *
<i>yr</i>	the year of the faculty member's first appointment * *

4.2.3 Member Function Documentation

4.2.3.1 print()

```
std::ostream & Faculty::print (
    std::ostream & os ) const [virtual]
```

Remarks

output a format [Faculty](#) object *

-

Parameters

<code>os</code>	<p>output stream which will receive the formatted Faculty data *</p> <ul style="list-style-type: none"> • @ewmrk by being a virtual function, implementations in subclasses * will be interpreted via polymorphism * •
-----------------	--

Returns

formatted string on the given output stream *

-

Reimplemented from [Entry](#).

Here is the call graph for this function:



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

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

4.3 SchoolDirectory Class Reference

Public Member Functions

- [SchoolDirectory](#) ()
- void [add](#) ([Entry](#) person)
- void [print](#) ()
- [Entry](#) * [lookup](#) (std::string first, std::string second)

4.3.1 Detailed Description

Remarks

Example of a School Directory application * Entries take the form of Students, [Faculty](#) and [Staff](#) *

-

Example illustrates a class hierarchy * * Base class: [Entry](#) * Subclasses: [Student](#), [Faculty](#), [Staff](#) *

-

Each class has a header (.h) and implementation (.cpp) files *

Other features: overwritten << operator and virtual print * *

-

: file: [SchoolDirectory.cpp](#) *

-

Author

Henry M. Walker *

Date

January 11, 2023 *

-

Remarks

References *

A School Directory as an Example of Object-Oriented Design * <http://localhost/courses/cpp-style-guide/c.php> *

-

4.3.2 Constructor & Destructor Documentation**4.3.2.1 SchoolDirectory()**

```
SchoolDirectory::SchoolDirectory ( ) [inline]
```

Remarks

Default constructor (with no parameters) *

4.3.3 Member Function Documentation**4.3.3.1 add()**

```
void SchoolDirectory::add (
    Entry person ) [inline]
```

Remarks

insert a person into the [SchoolDirectory](#) *

irectory entries are maintained in lastname/firstname order *

-

Parameters

<i>person</i>	the entry to be inserted into the underlying directory *
	•

Precondition

entries in the underlying directory are ordered by name *

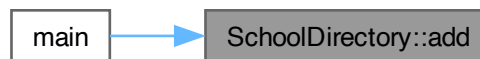
Postcondition

the underlying directory continues to be ordered by name *

Here is the call graph for this function:



Here is the caller graph for this function:



4.3.3.2 lookup()

```

Entry * SchoolDirectory::lookup (
    std::string first,
    std::string second ) [inline]
  
```

Remarks

entries in the directory are searched by first and last name *

•

Parameters

<i>first</i>	the first name of a person *
<i>last</i>	the last name of a person *
	•

Precondition

the underlying directory is ordered by last/first name *

Returns

if the name is found, a pointer to the entry is returned * if the name is not found, NULL is returned *

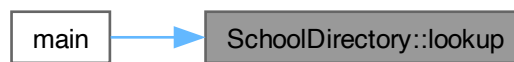
-

Remarks

searching is performed via a binarysearch *

-

Here is the caller graph for this function:

**4.3.3.3 print()**

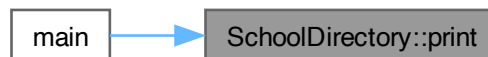
```
void SchoolDirectory::print ( ) [inline]
```

Remarks

entries in the underlying directory are printed to cout * with beginning and end markers *

-

Here is the caller graph for this function:



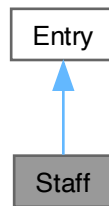
The documentation for this class was generated from the following file:

- [SchoolDirectory.cpp](#)

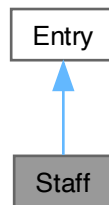
4.4 Staff Class Reference

```
#include <Staff.h>
```


Inheritance diagram for Staff:



Collaboration diagram for Staff:



Public Member Functions

- [Staff](#) (std::string first, std::string last, std::string addr, std::string room, int ext, std::string ttl)
- std::ostream & [print](#) (std::ostream &os) const

Public Member Functions inherited from [Entry](#)

- [Entry](#) ()
- [Entry](#) (std::string first, std::string last, std::string eAddress)
- bool [equals](#) (std::string first, std::string second)
- bool [equals](#) ([Entry](#) otherEntry)
- bool [comesBefore](#) (std::string first, std::string second)
- bool [comesBefore](#) ([Entry](#) otherEntry)
- virtual std::ostream & [print](#) (std::ostream &os) const
- virtual [~Entry](#) ()

4.4.1 Detailed Description

Remarks

[Student](#): a class derived from [Entry](#) for a School Directory * a staff member's entry * inherits a first name, last name, and email address from [Entry](#) * additional fields are the member's office, extension, title *

•

Inherited capabilities include: * Base constructors * Comparison operations depend upon last names, then first names * Formatted output *
 *

Overwritten capabilities include: * Multi-parameter constructor * Formatted print method *
 *

: files include header ([Staff.h](#)), Implementation ([Staff.cpp](#)) *
 *

Uncomment a main program for unit testing *
 *

Author

Henry M. Walker *

Date

January 11, 2023 *
 *

Remarks

References *

A School Directory as an Example of Object-Oriented Design * <http://localhost/courses/cpp-style-guide/01.php> *
 *

Full-parameter constructor *
 *

Parameters

<i>first</i>	a staff member's first name *
<i>last</i>	a staff member's last name *
<i>eAddress</i>	a staff member's email address *
<i>room</i>	a staff member's office *
<i>ext</i>	the telephone number extension for the office *
<i>title</i>	the staff member's title * *

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Staff()

```
Staff::Staff (
    std::string first,
    std::string last,
    std::string addr,
```

```

std::string room,
int ext,
std::string ttl )

```

Remarks

Student: a class derived from [Entry](#) for a School Directory * a staff member's entry * inherits a first name, last name, and email address from [Entry](#) * additional fields are the member's office, extension, title *

-

Inherited capabilities include: * Base constructors * Comparison operations depend upon last names, then first names * Formatted output *

-

Overwritten capabilities include: * Multi-parameter constructor * Formatted print method *

-

: files include header ([Staff.h](#)), Implementation ([Staff.cpp](#)) *

-

Uncomment a main program for unit testing *

-

Author

Henry M. Walker *

Date

January 11, 2023 *

-

Remarks

References *

A School Directory as an Example of Object-Oriented Design * <http://localhost/courses/cpp-style-guide/c.php> *

-

4.4.3 Member Function Documentation**4.4.3.1 print()**

```

std::ostream & Staff::print (
std::ostream & os ) const [virtual]

```

Remarks

output a format [Faculty](#) object *

-

Parameters

<code>os</code>	<p>output stream which will receive the formatted Faculty data *</p> <ul style="list-style-type: none"> • <code>@ewmrk</code> by being a virtual function, implementations in subclasses * will be interpreted via polymorphism * •
-----------------	---

Returns

formatted string on the given output stream *

-

Reimplemented from [Entry](#).

Here is the call graph for this function:



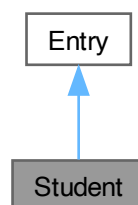
The documentation for this class was generated from the following files:

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

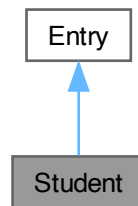
4.5 Student Class Reference

```
#include <Student.h>
```

Inheritance diagram for Student:



Collaboration diagram for Student:



Public Member Functions

- `Student` (`std::string first`, `std::string last`, `std::string addr`, `int yr`, `std::string box`)
- `std::ostream & print` (`std::ostream &os`) `const`

Public Member Functions inherited from `Entry`

- `Entry` ()
- `Entry` (`std::string first`, `std::string last`, `std::string eAddress`)
- `bool equals` (`std::string first`, `std::string second`)
- `bool equals` (`Entry otherEntry`)
- `bool comesBefore` (`std::string first`, `std::string second`)
- `bool comesBefore` (`Entry otherEntry`)
- `virtual std::ostream & print` (`std::ostream &os`) `const`
- `virtual ~Entry` ()

4.5.1 Detailed Description

Remarks

`Student`: a class derived from `Entry` for a School Directory * a student entry * inherits a first name, last name, and email address from `Entry` * additional fields are the student's year and PO Box *

•

Inherited capabilities include: * Base constructors * Comparison operations depend upon last names, then first names * Formatted output *

•

Overwritten capabilities include: * Multi-parameter constructor * Formatted print method *

•

: files include header (`Student.h`), Implementation (`Student.cpp`)*

•

Uncomment a main program for unit testing *

•

Author

Henry M. Walker *

Date

January 11, 2023 *

-

Remarks

References *

A School Directory as an Example of Object-Oriented Design * <http://localhost/courses/cpp-style-guide/c.php> *

-

4.5.2 Constructor & Destructor Documentation**4.5.2.1 Student()**

```
Student::Student (
    std::string first,
    std::string last,
    std::string addr,
    int yr,
    std::string box )
```

Remarks

Full-parameter constructor *

-

Parameters

<i>first</i>	a student's first name *
<i>last</i>	a student's last name *
<i>eAddress</i>	a student's email address *
<i>year</i>	the student's class or expected-graduation year *
<i>box</i>	the student's campus post office box *
	•

4.5.3 Member Function Documentation**4.5.3.1 print()**

```
std::ostream & Student::print (
    std::ostream & os ) const [virtual]
```

Remarks

output a format `Faculty` object *

-

Parameters

<i>os</i>	output stream which will receive the formatted Faculty data * <ul style="list-style-type: none">• @ewmrk by being a virtual function, implementations in subclasses * will be interpreted via polymorphism *•
-----------	--

Returns

formatted string on the given output stream *

•

Reimplemented from [Entry](#).

Here is the call graph for this function:



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

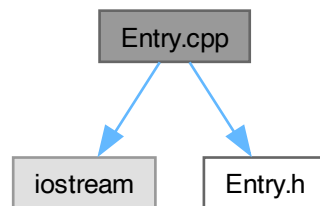
- [Student.h](#)
- [Student.cpp](#)

Chapter 5

File Documentation

5.1 Entry.cpp File Reference

```
#include <iostream>
#include "Entry.h"
Include dependency graph for Entry.cpp:
```



Functions

- `std::ostream & operator<< (std::ostream &os, const Entry &ent)`
- `int main ()`

5.1.1 Function Documentation

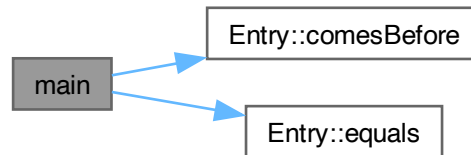
5.1.1.1 main()

```
int main ( )
```

Remarks

main procedure to control processing r uncomment this procedure for unit testing *

Here is the call graph for this function:

**5.1.1.2 operator<<()**

```

std::ostream & operator<< (
    std::ostream & os,
    const Entry & ent )
  
```

Remarks

overload << for printing an [Entry](#) *

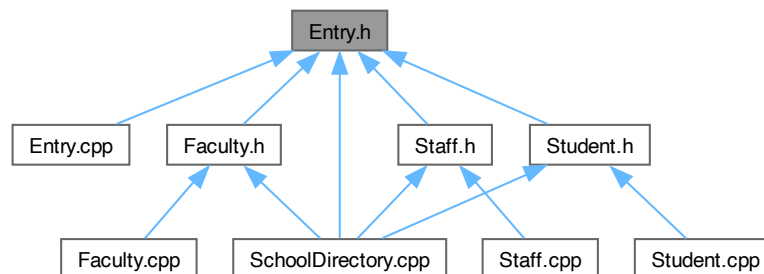
-

use of the virtual print method allows tailored output * by subclasses *

-

5.2 Entry.h File Reference

This graph shows which files directly or indirectly include this file:

**Classes**

- class [Entry](#)

5.3 Entry.h

[Go to the documentation of this file.](#)

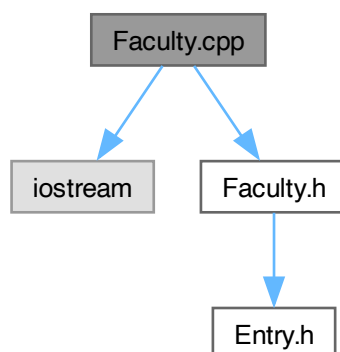
```
00001
00023 #ifndef ENTRY_H
00024 #define ENTRY_H
00025
00026 class Entry {
00027
00028 public:
00029     // constructors
00033     Entry () ;
00034
00043     Entry (std::string first, std::string last, std::string eAddress) ;
00044
00054     bool equals (std::string first, std::string second) ;
00055
00064     bool equals (Entry otherEntry) ;
00065
00075     bool comesBefore (std::string first, std::string second) ;
00076
00085     bool comesBefore (Entry otherEntry) ;
00086
00102     virtual
00103         std::ostream& print (std::ostream &os) const;
00104
00112     friend std::ostream& operator<< (std::ostream &os, const Entry& ent) ;
00113
00118     virtual
00119         ~Entry () ;
00120
00121 private:
00122     // could use "protected" here, so variables may be accessed in subclasses
00123     std::string firstName;
00124     std::string lastName;
00125     std::string eMail;
00126
00127 };
00128
00129 #endif
```

5.4 Faculty.cpp File Reference

```
#include <iostream>
```

```
#include "Faculty.h"
```

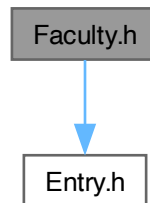
Include dependency graph for Faculty.cpp:



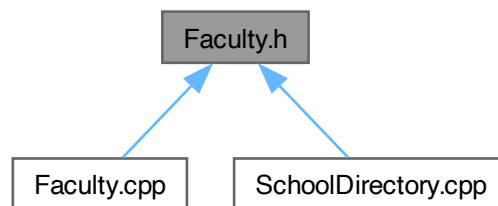
5.5 Faculty.h File Reference

```
#include "Entry.h"
```

Include dependency graph for Faculty.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Faculty](#)

5.6 Faculty.h

[Go to the documentation of this file.](#)

```

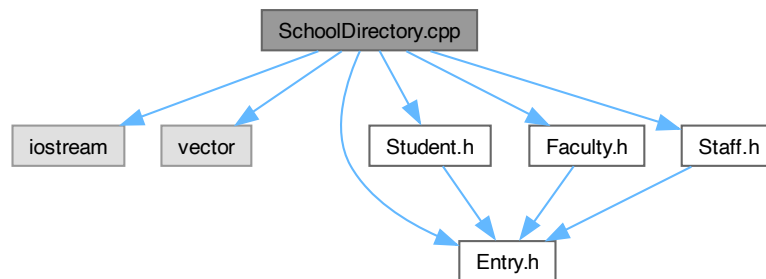
00001
00029 #ifndef FACULTY_H
00030 #define FACULTY_H
00031
00032 #include "Entry.h"
00033
00034 // Directory entries specific to faculty
00035 class Faculty : public Entry {
00036     // Faculty have four special fields
00037
00038 public:
00039
00052     Faculty (std::string first, std::string last, std::string addr, std::string room,
00053             int ext, std::string department, int yr);
00054
00066     virtual std::ostream& print (std::ostream &os) const ;
00067
00068     // Faculty have four special fields
00069 private:
00070     std::string office ;
00071     int extension ;
  
```

```
00072     std::string dept ;
00073     int firstYear ;
00074
00075 };
00076
00077 #endif
```

5.7 SchoolDirectory.cpp File Reference

```
#include <iostream>
#include <vector>
#include "Entry.h"
#include "Student.h"
#include "Faculty.h"
#include "Staff.h"
```

Include dependency graph for SchoolDirectory.cpp:



Classes

- class [SchoolDirectory](#)

Functions

- int [main](#) ()

5.7.1 Function Documentation

5.7.1.1 main()

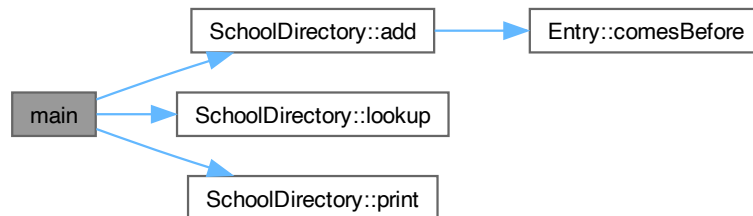
```
int main ( )
```

Remarks

main performs a reasonable level of testing *

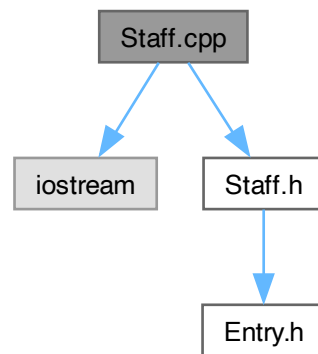
•

Here is the call graph for this function:



5.8 Staff.cpp File Reference

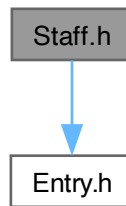
```
#include <iostream>
#include "Staff.h"
Include dependency graph for Staff.cpp:
```



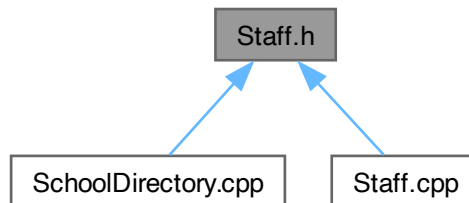
5.9 Staff.h File Reference

```
#include "Entry.h"
```

Include dependency graph for Staff.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Staff](#)

5.10 Staff.h

[Go to the documentation of this file.](#)

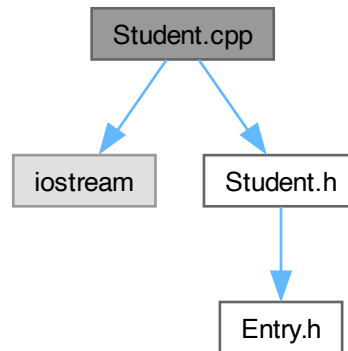
```
00001
00028 #ifndef STAFF_H
00029 #define STAFF_H
00030
00031 #include "Entry.h"
00032
00044 class Staff : public Entry {
00045 public:
00046     Staff (std::string first, std::string last, std::string addr,
00047           std::string room, int ext, std::string ttl) ;
00048
00060     std::ostream& print (std::ostream &os) const ;
00061
00062 // Staff have two special fields
00063 private:
00064     std::string office ;
00065     int extension ;
00066     std::string title ;
00067
00068 };
00069
00070 #endif
```

5.11 Student.cpp File Reference

```
#include <iostream>
```

```
#include "Student.h"
```

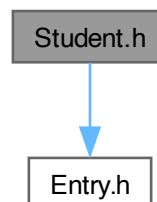
Include dependency graph for Student.cpp:



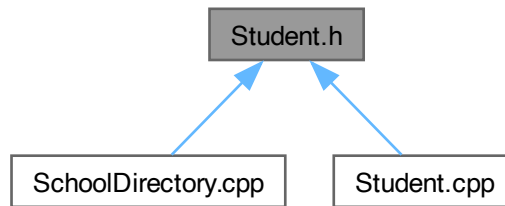
5.12 Student.h File Reference

```
#include "Entry.h"
```

Include dependency graph for Student.h:



This graph shows which files directly or indirectly include this file:



Classes

- class [Student](#)

5.13 Student.h

[Go to the documentation of this file.](#)

```
00001
00028 #ifndef STUDENT_H
00029 #define STUDENT_H
00030
00031 #include "Entry.h"
00032
00033
00034 class Student : public Entry {
00035 public:
00046     Student (std::string first, std::string last, std::string addr, int yr, std::string box) ;
00047
00059     std::ostream& print (std::ostream &os) const ;
00060
00061 // Students have two special fields
00062 private:
00063     int year;
00064     std::string POBox;
00065
00066 };
00067
00068 #endif
```


Index

- ~Entry
 - Entry, [9](#)
- add
 - SchoolDirectory, [18](#)
- comesBefore
 - Entry, [9](#), [10](#)
- Entry, [7](#)
 - ~Entry, [9](#)
 - comesBefore, [9](#), [10](#)
 - Entry, [8](#), [9](#)
 - equals, [10](#), [11](#)
 - operator<<, [13](#)
 - print, [12](#)
- Entry.cpp, [29](#)
 - main, [29](#)
 - operator<<, [30](#)
- Entry.h, [30](#)
- equals
 - Entry, [10](#), [11](#)
- Faculty, [13](#)
 - Faculty, [15](#)
 - print, [17](#)
- Faculty.cpp, [31](#)
- Faculty.h, [32](#)
- lookup
 - SchoolDirectory, [19](#)
- main
 - Entry.cpp, [29](#)
 - SchoolDirectory.cpp, [33](#)
- operator<<
 - Entry, [13](#)
 - Entry.cpp, [30](#)
- print
 - Entry, [12](#)
 - Faculty, [17](#)
 - SchoolDirectory, [20](#)
 - Staff, [23](#)
 - Student, [26](#)
- SchoolDirectory, [17](#)
 - add, [18](#)
 - lookup, [19](#)
 - print, [20](#)
 - SchoolDirectory, [18](#)
 - SchoolDirectory.cpp, [33](#)
 - main, [33](#)
 - Staff, [20](#)
 - print, [23](#)
 - Staff, [22](#)
 - Staff.cpp, [34](#)
 - Staff.h, [34](#)
 - Student, [24](#)
 - print, [26](#)
 - Student, [26](#)
 - Student.cpp, [36](#)
 - Student.h, [36](#)