

Registras

Generated by Doxygen 1.9.4

1 cpp-2024-1	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 Person::Impl Struct Reference	9
5.1.1 Constructor & Destructor Documentation	9
5.1.1.1 Impl() [1/3]	9
5.1.1.2 Impl() [2/3]	10
5.1.1.3 Impl() [3/3]	10
5.1.2 Member Data Documentation	10
5.1.2.1 firstName	10
5.1.2.2 lastName	10
5.2 Person Class Reference	11
5.2.1 Constructor & Destructor Documentation	12
5.2.1.1 Person() [1/3]	12
5.2.1.2 Person() [2/3]	13
5.2.1.3 ~Person()	13
5.2.1.4 Person() [3/3]	13
5.2.2 Member Function Documentation	13
5.2.2.1 changeName()	13
5.2.2.2 getName()	14
5.2.2.3 operator=() [1/2]	14
5.2.2.4 operator=() [2/2]	15
5.2.2.5 print()	15
5.2.3 Friends And Related Function Documentation	15
5.2.3.1 operator<<	15
5.2.3.2 operator>>	16
5.2.4 Member Data Documentation	17
5.2.4.1 NID	17
5.2.4.2 parentp	17
5.2.4.3 pimpl	17
5.2.4.4 prt	17
5.3 Personprint Class Reference	17
5.3.1 Member Function Documentation	18
5.3.1.1 print()	18
5.4 PersonPrintFullName Class Reference	18

5.4.1 Member Function Documentation	19
5.4.1.1 print()	19
5.5 PersonPrintId Class Reference	19
5.5.1 Member Function Documentation	20
5.5.1.1 print()	20
5.6 Trackable Class Reference	21
5.6.1 Constructor & Destructor Documentation	22
5.6.1.1 Trackable() [1/3]	22
5.6.1.2 Trackable() [2/3]	23
5.6.1.3 Trackable() [3/3]	23
5.6.1.4 ~Trackable()	23
5.6.2 Member Function Documentation	23
5.6.2.1 getName()	23
5.6.2.2 getProtected() [1/2]	24
5.6.2.3 getProtected() [2/2]	24
5.6.2.4 operator=() [1/2]	24
5.6.2.5 operator=() [2/2]	24
5.6.2.6 print()	25
5.6.3 Member Data Documentation	25
5.6.3.1 amount	25
5.6.3.2 ID	25
5.6.3.3 proc	25
5.6.3.4 type	26
5.7 TrackableProtected Class Reference	26
5.7.1 Constructor & Destructor Documentation	26
5.7.1.1 TrackableProtected() [1/3]	26
5.7.1.2 TrackableProtected() [2/3]	27
5.7.1.3 TrackableProtected() [3/3]	27
5.7.2 Member Data Documentation	27
5.7.2.1 Name	27
5.7.2.2 NID	27
5.7.2.3 Time	27
6 File Documentation	29
6.1 cmake-build-debug/CMakeFiles/3.27.0/CompilerIdC/CMakeCCompilerId.c File Reference	29
6.1.1 Macro Definition Documentation	29
6.1.1.1 __has_include	30
6.1.1.2 ARCHITECTURE_ID	30
6.1.1.3 C_VERSION	30
6.1.1.4 COMPILER_ID	30
6.1.1.5 DEC	30
6.1.1.6 HEX	30

6.1.1.7 PLATFORM_ID	31
6.1.1.8 STRINGIFY	31
6.1.1.9 STRINGIFY_HELPER	31
6.1.2 Function Documentation	31
6.1.2.1 main()	31
6.1.3 Variable Documentation	31
6.1.3.1 info_arch	32
6.1.3.2 info_compiler	32
6.1.3.3 info_language_extensions_default	32
6.1.3.4 info_language_standard_default	32
6.1.3.5 info_platform	32
6.2 cmake-build-debug/CMakeFiles/3.27.0/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference . .	32
6.2.1 Macro Definition Documentation	33
6.2.1.1 __has_include	33
6.2.1.2 ARCHITECTURE_ID	33
6.2.1.3 COMPILER_ID	33
6.2.1.4 CXX_STD	33
6.2.1.5 DEC	34
6.2.1.6 HEX	34
6.2.1.7 PLATFORM_ID	34
6.2.1.8 STRINGIFY	34
6.2.1.9 STRINGIFY_HELPER	34
6.2.2 Function Documentation	34
6.2.2.1 main()	35
6.2.3 Variable Documentation	35
6.2.3.1 info_arch	35
6.2.3.2 info_compiler	35
6.2.3.3 info_language_extensions_default	35
6.2.3.4 info_language_standard_default	36
6.2.3.5 info_platform	36
6.3 main.cpp File Reference	36
6.3.1 Function Documentation	36
6.3.1.1 main()	37
6.3.1.2 printAmount()	37
6.4 Person.cpp File Reference	38
6.4.1 Function Documentation	38
6.4.1.1 operator<<()	38
6.4.1.2 operator>>()	39
6.5 Person.h File Reference	40
6.6 Person.h	40
6.7 Personprint.cpp File Reference	41
6.8 Personprint.h File Reference	42

6.9 Personprint.h	42
6.10 PersonPrintFullName.cpp File Reference	42
6.11 PersonPrintFullName.h File Reference	43
6.12 PersonPrintFullName.h	44
6.13 PersonPrintId.cpp File Reference	44
6.14 PersonPrintId.h File Reference	45
6.15 PersonPrintId.h	45
6.16 README.md File Reference	46
6.17 Trackable.cpp File Reference	46
6.18 Trackable.h File Reference	46
6.19 Trackable.h	47
6.20 TrackableProtected.cpp File Reference	47
6.21 TrackableProtected.h File Reference	48
6.22 TrackableProtected.h	48
Index	49

Chapter 1

cpp-2024-1

Sukurti universiteto registrą. Registras saugos universiteto informaciją (dabar gali saugoti tik informaciją apie žmones) ir registracijos momentą (kada buvo informacija įvesta į sistemą).

Vėliau duomenys bus saugomi viename objekte, kurio turinį būtų galima išsaugoti failę ir iš kurio būtų galima atkurti registrą.

Iskilo problemu su MIF Gitlab, todėl projekto kopija, kuri bus pagrindine atnaujinant yra [cia](#) I MIF gitlab bus atnaujinami duomenys, kuomet busiu fakultete, nes nepavyksta ikelti saugyklos neprisijungus prie fakulteto interneto, net naujus VU VPN

Žmonių duomenų įrašai gali būti srautais rašomi bei skaitomi iš failų. Taip pat galima vykdymo metu pakeisti ką daro "print" funkcija. Tam kad tai padaryti, reikia viešą lauką prt pakeisti klasės objektu, kuris įgyvendina "interface"ą [Personprint](#). Iškvičiant print funkcija yra iskvičiama klasėje esantčio Personprint* objekto print.

Serializacija, vykdoma binariniais srautais.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Person::Impl	9
Personprint	17
PersonPrintFullName	18
PersonPrintId	19
Trackable	21
Person	11
TrackableProtected	26

Chapter 3

Class Index

3.1 Class List

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

Person::Impl	9
Person	11
Personprint	17
PersonPrintFullName	18
PersonPrintId	19
Trackable	21
TrackableProtected	26

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

main.cpp	36
Person.cpp	38
Person.h	40
Personprint.cpp	41
Personprint.h	42
PersonPrintFullName.cpp	42
PersonPrintFullName.h	43
PersonPrintId.cpp	44
PersonPrintId.h	45
Trackable.cpp	46
Trackable.h	46
TrackableProtected.cpp	47
TrackableProtected.h	48
cmake-build-debug/CMakeFiles/3.27.0/CompilerIdC/CMakeCCompilerId.c	29
cmake-build-debug/CMakeFiles/3.27.0/CompilerIdCXX/CMakeCXXCompilerId.cpp	32

Chapter 5

Class Documentation

5.1 Person::Impl Struct Reference

Public Member Functions

- [Impl](#) (std::string Fname, std::string Lname)
- [Impl](#) ()
- [Impl](#) (const [Person](#) &oth)

Public Attributes

- std::string [firstName](#)
- std::string [lastName](#)

5.1.1 Constructor & Destructor Documentation

5.1.1.1 Impl() [1/3]

```
Person::Impl::Impl (
    std::string Fname,
    std::string Lname ) [inline]
```

Paprastas konstruktorius

Parameters

<i>Fname</i>	vardas
<i>Lname</i>	Pavarde

```
19                                     {
20         firstName=Fname;
21         lastName=Lname;
22     }
```


5.1.1.2 Impl() [2/3]

```
Person::Impl::Impl ( ) [inline]
```

Tuščias konstruktorius

```
26 {};
```

5.1.1.3 Impl() [3/3]

```
Person::Impl::Impl (
    const Person & oth ) [inline]
```

Konstruktoriaus kopijavimas

Parameters

<i>oth</i>	Person is kurio kopijuojama
------------	---

```
31         {
32     firstName = oth.pimpl->firstName;
33     lastName = oth.pimpl->lastName;
34 }
```

5.1.2 Member Data Documentation

5.1.2.1 firstName

```
std::string Person::Impl::firstName
```

5.1.2.2 lastName

```
std::string Person::Impl::lastName
```

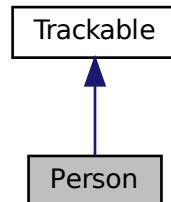
The documentation for this struct was generated from the following file:

- [Person.cpp](#)

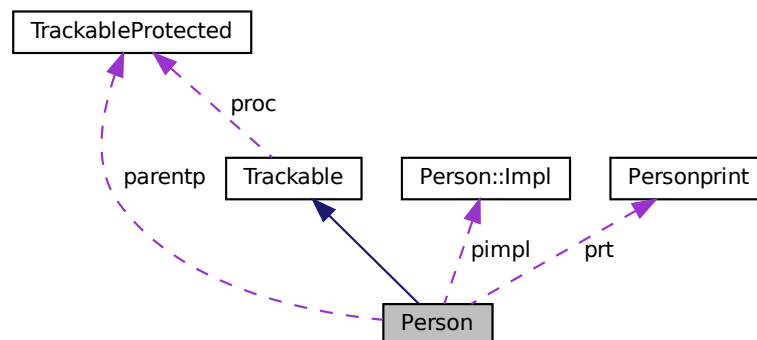
5.2 Person Class Reference

```
#include <Person.h>
```

Inheritance diagram for Person:



Collaboration diagram for Person:



Classes

- struct [Impl](#)

Public Member Functions

- [Person](#) (std::string name, std::string lastName)
- [Person](#) ()
- std::string [getName](#) () const
- void [print](#) () const override
- [~Person](#) ()
- [Person](#) (const [Person](#) &other)
- [Person](#) & [operator=](#) (const [Person](#) &other)
- [Person](#) & [operator=](#) ([Person](#) &&other)

Static Public Member Functions

- static void `changeName` (std::string NewName, `Trackable` *obj)

Public Attributes

- `Personprint` * `pnt` = nullptr

Static Public Attributes

- static unsigned long `NID`

Private Attributes

- `Impl` * `pimpl`
- `TrackableProtected` * `parentp`

Friends

- std::ostream & `operator<<` (std::ostream &o, const `Person` &p)
- std::istream & `operator>>` (std::istream &i, `Person` &p)

Additional Inherited Members

5.2.1 Constructor & Destructor Documentation

5.2.1.1 `Person()` [1/3]

```
Person::Person (
    std::string name,
    std::string lastName )
```

Konstruktorius

Parameters

<i>name</i>	Žmogaus vardas
<i>lastName</i>	Žmogaus pavardė.

```
41                                     :   pimpl(new Impl(name, lastName)) {
42     parentp = getProtected();
43     parentp ->Name = name + " " + lastName;
44     prt = new PersonPrintFullName();
45 }
```

5.2.1.2 Person() [2/3]

```
Person::Person ( )
```

Tuščias konstruktorius.

```
165         : pimpl(new Impl("", "")) {
166     parentp = getProtected();
167     prt = new PersonPrintFullName();
168 }
```

5.2.1.3 ~Person()

```
Person::~~Person ( )
```

destruktorius.

```
56     {
57     delete parentp;
58     delete pimpl;
59 }
```

5.2.1.4 Person() [3/3]

```
Person::Person (
    const Person & other )
```

Copy konstruktorius

Parameters

<i>other</i>	iš ko kopijuojamaa.
--------------	---------------------

```
85         :pimpl(new Impl(other)) {
86     parentp = getProtected();
87     parentp->Name = other.parentp->Name;
88     prt = other.prt;
89 }
```

5.2.2 Member Function Documentation

5.2.2.1 changeName()

```
void Person::changeName (
    std::string NewName,
    Trackable * obj ) [static]
```

Pakeičia vardą į naują.

Parameters

<i>NewName</i>	naujas vardas.
<i>obj</i>	Sekamas objektas, kurio vardas bus pakeistas.

```
65                                     {
66     Person *cast = dynamic_cast<Person>(obj);
67     if(cast){
68         cast ->pimpl->firstName = NewName;
69         cast ->parentp->Name = NewName + cast->pimpl->lastName;
70     }else{
71         throw std::bad_cast();
72     }
73 }
```

5.2.2.2 getName()

```
std::string Person::getName ( ) const [virtual]
```

Gauti vardą.

Returns

Vardas

Reimplemented from [Trackable](#).

```
78     {
79     return this->pimpl->firstName + " " + this->pimpl->lastName;
80 }
```

5.2.2.3 operator=() [1/2]

```
Person & Person::operator= (
    const Person & other )
```

Copy assignment

Parameters

<i>other</i>	Žmogus, iš kurio imami duomenys.
--------------	----------------------------------

Returns

```
95                                     {
96     return *this = Person(other);
97 }
```

5.2.2.4 operator=() [2/2]

```
Person & Person::operator= (
    Person && other )
```

Move assignment

Parameters

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

Returns

Žmogus, iš kurio imami duomenys.

```
103                                     {
104     if(this == &other){
105         return *this;
106     }
107     parentp->Name = other.parentp->Name;
108     pimpl->firstName = other.pimpl->firstName;
109     pimpl->lastName = other.pimpl->lastName;
110     ID = other.ID;
111     other.pimpl->firstName = "";
112     other.pimpl->lastName = "";
113     other.parentp->Name = "";
114     other.ID = 0;
115     return *this;
116 }
```

5.2.2.5 print()

```
void Person::print ( ) const [override], [virtual]
```

de facto "to String"

Reimplemented from [Trackable](#).

```
49     {
50         prt->print(*this);
51         //std::cout<<"ENTRY ID: " << ID << " Given name: " <<pimpl->firstName<< " Last name: " <<pimpl->lastName<< "
52         Date of entry: " << std::ctime(&parentp->Time);
52     }
```

5.2.3 Friends And Related Function Documentation

5.2.3.1 operator<<

```
std::ostream & operator<< (
    std::ostream & o,
    const Person & p ) [friend]
```

Išvedimo srauto perrašymas.

Parameters

<i>o</i>	output stautas
<i>p</i>	Žmogus p, kuris paduodamas srautą.

Returns

srautas o

```

123                                     {
124     o<<p.pimpl->firstName;
125     o<<" "<<p.pimpl->lastName<<" ";
126     std::ostringstream ss;
127     ss<<p.parentp->Time;
128     o<<ss.str()<<" ";
129     o<<p.ID<<" ";
130     o<<p.type<<" ";
131     PersonPrint* pri = dynamic_cast<PersonPrintFullName*>(p.prt);
132     if(pri){
133         o<<"N\n";
134     }else{
135         o<<"I\n";
136     }
137     return o;
138 }
```

5.2.3.2 operator>>

```

std::istream & operator>> (
    std::istream & i,
    Person & p ) [friend]
```

Ivesties binarinio srauto perrašymas, kada būtų galima objektą perduoti į jį.

Parameters

<i>i</i>	input srautas
<i>p</i>	Žmogus p, kurio duomenys imami iš srauto.

Returns

input srautas

```

145                                     {
146     i>>p.pimpl->firstName;
147     i>>p.pimpl->lastName;
148     std::string raw, type;
149     i>>raw;
150     std::istringstream is(raw);
151     is>>p.parentp->Time;
152     i>>p.ID;
153     i>>p.type;
154     i>>type;
155     if(type == "N"){
156         p.prt = new PersonPrintFullName;
157     }else{
158         p.prt = new PersonPrintId;
159     }
160     return i;
161 }
```

5.2.4 Member Data Documentation

5.2.4.1 NID

```
unsigned long Person::NID [static]
```

5.2.4.2 parentp

```
TrackableProtected* Person::parentp [private]
```

5.2.4.3 pimpl

```
Impl* Person::pimpl [private]
```

5.2.4.4 prt

```
Personprint* Person::prt = nullptr
```

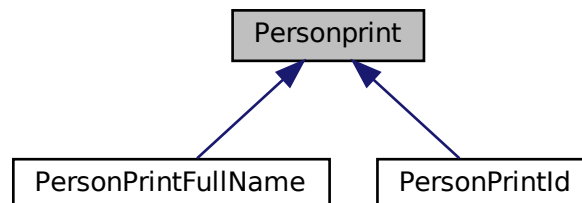
The documentation for this class was generated from the following files:

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

5.3 Personprint Class Reference

```
#include <Personprint.h>
```

Inheritance diagram for Personprint:



Public Member Functions

- virtual void [print](#) (const [Person](#) &p)

5.3.1 Member Function Documentation

5.3.1.1 [print\(\)](#)

```
void Personprint::print (  
    const Person & p ) [virtual]
```

De facto interfaceo funkcijos default deklaracija

Parameters

<i>p</i>	Zmogus is kurio iskviesta funkcija
----------	------------------------------------

Reimplemented in [PersonPrintFullName](#), and [PersonPrintId](#).

```
11                                     {  
12  
13 }
```

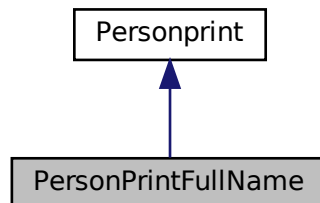
The documentation for this class was generated from the following files:

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

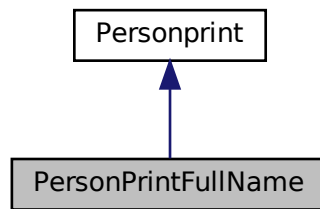
5.4 PersonPrintFullName Class Reference

```
#include <PersonPrintFullName.h>
```

Inheritance diagram for PersonPrintFullName:



Collaboration diagram for PersonPrintFullName:



Public Member Functions

- void `print` (const `Person` &p) override

5.4.1 Member Function Documentation

5.4.1.1 `print()`

```
void PersonPrintFullName::print (
    const Person & p ) [override], [virtual]
```

Atspauzdinti varda i console

Parameters

<i>p</i>	Zmogus is kurio iskviesta funkcija
----------	------------------------------------

Reimplemented from `Personprint`.

```
12                                     {
13
14     std::cout<<p.getName()<<"\n";
15
16 }
```

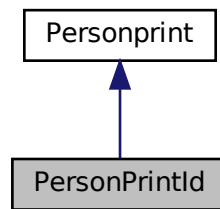
The documentation for this class was generated from the following files:

- `PersonPrintFullName.h`
- `PersonPrintFullName.cpp`

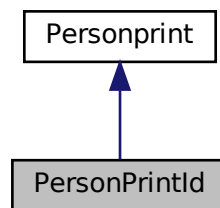
5.5 PersonPrintId Class Reference

```
#include <PersonPrintId.h>
```

Inheritance diagram for PersonPrintId:



Collaboration diagram for PersonPrintId:



Public Member Functions

- void `print` (const `Person` &p) override

5.5.1 Member Function Documentation

5.5.1.1 `print()`

```
void PersonPrintId::print (  
    const Person & p ) [override], [virtual]
```

Atspauzdinti ID i konsole

Parameters

<i>p</i>	Zmogus is kurio iskviesta funkcija
----------	------------------------------------

Reimplemented from [Personprint](#).

```
13 {  
14  
15     std::cout << p.ID << "\n";  
16  
17 }
```

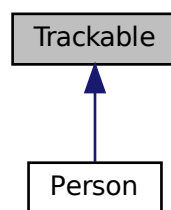
The documentation for this class was generated from the following files:

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

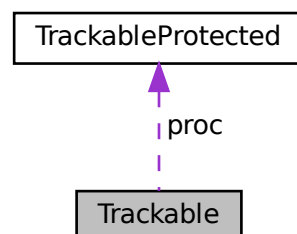
5.6 Trackable Class Reference

```
#include <Trackable.h>
```

Inheritance diagram for Trackable:



Collaboration diagram for Trackable:



Public Member Functions

- virtual std::string [getName](#) () const
- [Trackable](#) (std::string Name)
- [Trackable](#) ()
- virtual void [print](#) () const
- [Trackable](#) (const [Trackable](#) &other)
- [Trackable](#) & [operator=](#) (const [Trackable](#) &other)
- [Trackable](#) & [operator=](#) ([Trackable](#) &&other) noexcept
- [~Trackable](#) ()

Public Attributes

- unsigned [type](#)
- unsigned long [ID](#)

Static Public Attributes

- static unsigned long [amount](#) = 0

Protected Member Functions

- const [TrackableProtected](#) * [getProtected](#) () const
- [TrackableProtected](#) * [getProtected](#) ()

Private Attributes

- [TrackableProtected](#) * [proc](#)

5.6.1 Constructor & Destructor Documentation

5.6.1.1 [Trackable\(\)](#) [1/3]

```
Trackable::Trackable (
    std::string Name )
```

Paprastas konstruktorius

Parameters

<i>Name</i>	
-------------	--

```
14                                     : proc(new TrackableProtected(Name)) {
15     this->ID = TrackableProtected::NID;
16     TrackableProtected::NID++;
17     amount++;
18 }
```

5.6.1.2 Trackable() [2/3]

```
Trackable::Trackable ( )
```

Tuscias konstruktorius

```
30      : proc(new TrackableProtected()) {
31      ID = proc->NID;
32      TrackableProtected::NID++;
33      amount++;
34 }
```

5.6.1.3 Trackable() [3/3]

```
Trackable::Trackable (
    const Trackable & other )
```

Copy konstruktorius

Parameters

<i>other</i>	Trackable , is kurio kopijuojama
--------------	--

```
46      :proc(new TrackableProtected(other.getProtected())) {
47      ID = other.ID;
48 }
```

5.6.1.4 ~Trackable()

```
Trackable::~~Trackable ( )
```

Destruktorius

```
24      {
25      amount--;
26 }
```

5.6.2 Member Function Documentation

5.6.2.1 getName()

```
std::string Trackable::getName ( ) const [virtual]
```

de facto to string

Returns

Reimplemented in [Person](#).

```
39      {
40      return proc->Name;
41 }
```

5.6.2.2 getProtected() [1/2]

```
TrackableProtected * Trackable::getProtected ( ) [protected]
```

5.6.2.3 getProtected() [2/2]

```
const TrackableProtected * Trackable::getProtected ( ) const [protected]
```

Returns

rodykle i protected klase

```
76                                     {
77     return proc;
78 }
```

5.6.2.4 operator=() [1/2]

```
Trackable & Trackable::operator= (
    const Trackable & other )
```

Copy assignment

Parameters

<i>other</i>	Trackable , is kurio kopijuojama
--------------	--

Returns

nukopijuotas trackable

```
54                                     {
55     return *this = Trackable(other);
56 }
```

5.6.2.5 operator=() [2/2]

```
Trackable & Trackable::operator= (
    Trackable && other ) [noexcept]
```

Move assignment

Parameters

<i>other</i>	Trackable is kurio imti
--------------	---

ReturnsMoveintas [Trackable](#)

```

62                                     {
63         if (this == &other) {
64             return *this;
65         }
66         proc->Name = other.proc->Name;
67         ID=other.ID;
68         other.proc->Name = "";
69         other.ID = 0;
70         return *this;
71     }

```

5.6.2.6 print()

```
void Trackable::print ( ) const [virtual]
```

vardo rasymas

Reimplemented in [Person](#).

```

82         {
83         std::cout << proc->Name << "\n";
84     }

```

5.6.3 Member Data Documentation**5.6.3.1 amount**

```
unsigned long Trackable::amount = 0 [static]
```

5.6.3.2 ID

```
unsigned long Trackable::ID
```

5.6.3.3 proc

```
TrackableProtected* Trackable::proc [private]
```


5.6.3.4 type

```
unsigned Trackable::type
```

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

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

5.7 TrackableProtected Class Reference

```
#include <TrackableProtected.h>
```

Public Member Functions

- [TrackableProtected](#) (const [TrackableProtected](#) *tra)
- [TrackableProtected](#) (std::string name)
- [TrackableProtected](#) ()

Public Attributes

- std::string [Name](#)
- time_t [Time](#)

Static Public Attributes

- static unsigned long [NID](#) = 0

5.7.1 Constructor & Destructor Documentation

5.7.1.1 TrackableProtected() [1/3]

```
TrackableProtected::TrackableProtected (
    const TrackableProtected * tra )
```

Copy konstruktorius

Parameters

<i>tra</i>	TrackableProtected , kuris kopijuojamas
------------	---

```
12
13 Name = tra->Name;
```

```
: Time(tra->Time) {
```

```
14 }
```

5.7.1.2 TrackableProtected() [2/3]

```
TrackableProtected::TrackableProtected (
    std::string name )
```

Paprastaas konstruktorius

Parameters

<i>name</i>	Pavadinimas
-------------	-------------

```
20                                     :Time(time(0)) {
21 Name = name;
22
23 }
```

5.7.1.3 TrackableProtected() [3/3]

```
TrackableProtected::TrackableProtected ( )
```

tuscias kostruktorius

```
27                                     : Time(time(0)) {
28
29 }
```

5.7.2 Member Data Documentation

5.7.2.1 Name

```
std::string TrackableProtected::Name
```

5.7.2.2 NID

```
unsigned long TrackableProtected::NID = 0 [static]
```

5.7.2.3 Time

```
time_t TrackableProtected::Time
```

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

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

Chapter 6

File Documentation

6.1 cmake-build-debug/CMakeFiles/3.27.0/CompilerIdC/CMakeCCompilerId.c File Reference

Macros

- `#define __has_include(x) 0`
- `#define COMPILER_ID ""`
- `#define STRINGIFY_HELPER(X) #X`
- `#define STRINGIFY(X) STRINGIFY_HELPER(X)`
- `#define PLATFORM_ID`
- `#define ARCHITECTURE_ID`
- `#define DEC(n)`
- `#define HEX(n)`
- `#define C_VERSION`

Functions

- `int main (int argc, char *argv[])`

Variables

- `char const * info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"`
- `char const * info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"`
- `char const * info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"`
- `const char * info_language_standard_default`
- `const char * info_language_extensions_default`

6.1.1 Macro Definition Documentation

6.1.1.1 __has_include

```
#define __has_include(  
    x ) 0
```

6.1.1.2 ARCHITECTURE_ID

```
#define ARCHITECTURE_ID
```

6.1.1.3 C_VERSION

```
#define C_VERSION
```

6.1.1.4 COMPILER_ID

```
#define COMPILER_ID ""
```

6.1.1.5 DEC

```
#define DEC(  
    n )
```

Value:

```
('0' + ((n) / 10000000) % 10), \  
( '0' + ((n) / 1000000) % 10), \  
( '0' + ((n) / 100000) % 10), \  
( '0' + ((n) / 10000) % 10), \  
( '0' + ((n) / 1000) % 10), \  
( '0' + ((n) / 100) % 10), \  
( '0' + ((n) / 10) % 10), \  
( '0' + ((n) % 10))
```

6.1.1.6 HEX

```
#define HEX(  
    n )
```

Value:

```
('0' + ((n) >> 28 & 0xF)), \  
( '0' + ((n) >> 24 & 0xF)), \  
( '0' + ((n) >> 20 & 0xF)), \  
( '0' + ((n) >> 16 & 0xF)), \  
( '0' + ((n) >> 12 & 0xF)), \  
( '0' + ((n) >> 8 & 0xF)), \  
( '0' + ((n) >> 4 & 0xF)), \  
( '0' + ((n) & 0xF))
```

6.1.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

6.1.1.8 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY_HELPER(X)
```

6.1.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(  
    X ) #X
```

6.1.2 Function Documentation

6.1.2.1 main()

```
int main (  
    int argc,  
    char * argv[] )  
841 {  
842     int require = 0;  
843     require += info_compiler[argc];  
844     require += info_platform[argc];  
845     require += info_arch[argc];  
846 #ifdef COMPILER_VERSION_MAJOR  
847     require += info_version[argc];  
848 #endif  
849 #ifdef COMPILER_VERSION_INTERNAL  
850     require += info_version_internal[argc];  
851 #endif  
852 #ifdef SIMULATE_ID  
853     require += info_simulate[argc];  
854 #endif  
855 #ifdef SIMULATE_VERSION_MAJOR  
856     require += info_simulate_version[argc];  
857 #endif  
858 #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)  
859     require += info_cray[argc];  
860 #endif  
861     require += info_language_standard_default[argc];  
862     require += info_language_extensions_default[argc];  
863     (void)argv;  
864     return require;  
865 }
```

6.1.3 Variable Documentation

6.1.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

6.1.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

6.1.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["  
  "OFF"  
"]"
```

6.1.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

```
=  
  "INFO" ":" "standard_default[" C_VERSION "]"
```

6.1.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

6.2 cmake-build-debug/CMakeFiles/3.27.0/CompilerIdCXX/CMakeCXXCompilerId.cpp File Reference

Macros

- `#define __has_include(x) 0`
- `#define COMPILER_ID ""`
- `#define STRINGIFY_HELPER(X) #X`
- `#define STRINGIFY(X) STRINGIFY_HELPER(X)`
- `#define PLATFORM_ID`
- `#define ARCHITECTURE_ID`
- `#define DEC(n)`
- `#define HEX(n)`
- `#define CXX_STD __cplusplus`

Functions

- int [main](#) (int argc, char *argv[])

Variables

- char const * [info_compiler](#) = "INFO" ":" "compiler[" COMPILER_ID "]"
- char const * [info_platform](#) = "INFO" ":" "platform[" PLATFORM_ID "]"
- char const * [info_arch](#) = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
- const char * [info_language_standard_default](#)
- const char * [info_language_extensions_default](#)

6.2.1 Macro Definition Documentation

6.2.1.1 `__has_include`

```
#define __has_include(  
    x ) 0
```

6.2.1.2 `ARCHITECTURE_ID`

```
#define ARCHITECTURE_ID
```

6.2.1.3 `COMPILER_ID`

```
#define COMPILER_ID ""
```

6.2.1.4 `CXX_STD`

```
#define CXX_STD __cplusplus
```


6.2.1.5 DEC

```
#define DEC(  
    n )
```

Value:

```
('0' + ((n) / 10000000) % 10), \
('0' + ((n) / 1000000) % 10), \
('0' + ((n) / 100000) % 10), \
('0' + ((n) / 10000) % 10), \
('0' + ((n) / 1000) % 10), \
('0' + ((n) / 100) % 10), \
('0' + ((n) / 10) % 10), \
('0' + ((n) % 10))
```

6.2.1.6 HEX

```
#define HEX(  
    n )
```

Value:

```
('0' + ((n) >> 28 & 0xF)), \
('0' + ((n) >> 24 & 0xF)), \
('0' + ((n) >> 20 & 0xF)), \
('0' + ((n) >> 16 & 0xF)), \
('0' + ((n) >> 12 & 0xF)), \
('0' + ((n) >> 8 & 0xF)), \
('0' + ((n) >> 4 & 0xF)), \
('0' + ((n) & 0xF))
```

6.2.1.7 PLATFORM_ID

```
#define PLATFORM_ID
```

6.2.1.8 STRINGIFY

```
#define STRINGIFY(  
    X ) STRINGIFY_HELPER(X)
```

6.2.1.9 STRINGIFY_HELPER

```
#define STRINGIFY_HELPER(  
    X ) #X
```

6.2.2 Function Documentation

6.2.2.1 main()

```
int main (
    int argc,
    char * argv[] )
831 {
832     int require = 0;
833     require += info_compiler[argc];
834     require += info_platform[argc];
835     require += info_arch[argc];
836     #ifdef COMPILER_VERSION_MAJOR
837     require += info_version[argc];
838     #endif
839     #ifdef COMPILER_VERSION_INTERNAL
840     require += info_version_internal[argc];
841     #endif
842     #ifdef SIMULATE_ID
843     require += info_simulate[argc];
844     #endif
845     #ifdef SIMULATE_VERSION_MAJOR
846     require += info_simulate_version[argc];
847     #endif
848     #if defined(__CRAYXT_COMPUTE_LINUX_TARGET)
849     require += info_cray[argc];
850     #endif
851     require += info_language_standard_default[argc];
852     require += info_language_extensions_default[argc];
853     (void)argv;
854     return require;
855 }
```

6.2.3 Variable Documentation

6.2.3.1 info_arch

```
char const* info_arch = "INFO" ":" "arch[" ARCHITECTURE_ID "]"
```

6.2.3.2 info_compiler

```
char const* info_compiler = "INFO" ":" "compiler[" COMPILER_ID "]"
```

6.2.3.3 info_language_extensions_default

```
const char* info_language_extensions_default
```

Initial value:

```
= "INFO" ":" "extensions_default["
  "OFF"
  "]"
```

6.2.3.4 info_language_standard_default

```
const char* info_language_standard_default
```

Initial value:

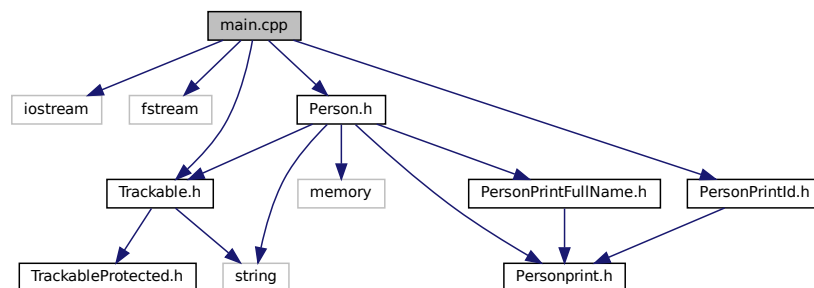
```
= "INFO" ":" "standard_default["  
  "98"  
"]"
```

6.2.3.5 info_platform

```
char const* info_platform = "INFO" ":" "platform[" PLATFORM_ID "]"
```

6.3 main.cpp File Reference

```
#include <iostream>  
#include <fstream>  
#include "Trackable.h"  
#include "Person.h"  
#include "PersonPrintId.h"  
Include dependency graph for main.cpp:
```



Functions

- void [printAmount](#) ()
- int [main](#) ()

6.3.1 Function Documentation

6.3.1.1 main()

```
int main ( )
```

Pagrindinė funkcija skirta pademonstruoti reigistro pagrindinį veikimą.

```
15     {
16         Trackable* test1 = new Trackable("Netipizuotas irasas");
17         std::ofstream out("TEST.abc");
18         test1->print();
19         printAmount();
20         Trackable* test2 = new Person("vardas", "pavarde");
21         test2->print();
22         printAmount();
23         //Trackable *Entries[7];
24         Trackable ** Ent;
25         Ent = new Trackable*[7];
26         for(int i = 0; i < 3; ++i){
27             std::string name;
28             std::cin >> name;
29             Ent[i] = new Trackable(name);
30             //Entries[i] = new Trackable(name);
31         }
32         for(int i = 0; i < 4; ++i){
33             std::string f, l;
34             std::cin >> f >> l;
35             Ent[3 + i] = new Person(f, l);
36             //Entries[3 + i] = new Person(f, l);
37         }
38         /*for(auto Entry : Entries){
39 Entry->print();
40 }*/
41         for(int i = 0; i < 7; ++i){
42             Ent[i]->print();
43         }
44         try {
45             // Person::changeName("Vardenis", Entries[6]);
46             Person::changeName("Vardenis", Ent[6]);
47         }catch(std::bad_cast) {
48             std::cout<<"Klaida keiciant 6\n";
49         }
50         try {
51             // Person::changeName("Vardenis", Entries[0]);
52             Person::changeName("Vardenis", Ent[0]);
53         }catch(std::bad_cast) {
54             std::cout<<"Klaida keiciant 0\n";
55         }
56         Person *cast = dynamic_cast<Person*>(Ent[6]);
57         out<<cast;
58         out.close();
59         /* for(auto Entry : Entries){
60 Entry->print();
61 }*/
62         for(int i = 0; i < 7; i++){
63             Ent[i]->print();
64         }
65         printAmount();
66         //delete Entries[6];
67         delete Ent[6];
68         printAmount();
69         Person *p = new Person("A", "B");
70         p->print();
71         p->prt=new PersonPrintId();
72         p->print();
73         p->print();
74
75         std::ifstream in("TEST.abc");
76         Person per;
77
78         in>>per;
79
80         per.print();
81         return 0;
82     }
```

6.3.1.2 printAmount()

```
void printAmount ( )
```

Funkcija atspauzdinti registre esanciu elementu kieki.

```

9      {
10         std::cout<<Trackable::amount<<std::endl;
11     }

```

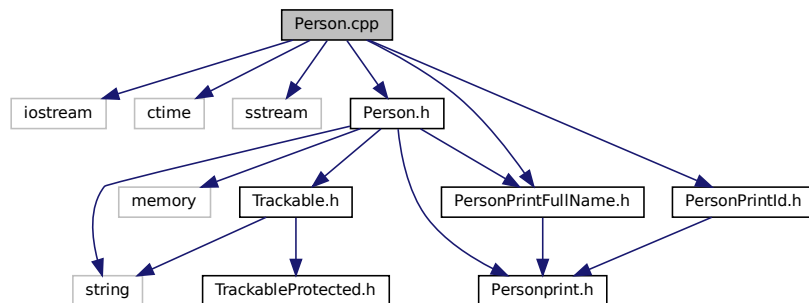
6.4 Person.cpp File Reference

```

#include <iostream>
#include <ctime>
#include <sstream>
#include "Person.h"
#include "PersonPrintFullName.h"
#include "PersonPrintId.h"

```

Include dependency graph for Person.cpp:



Classes

- struct [Person::Impl](#)

Functions

- std::ostream & [operator<<](#) (std::ostream &o, const [Person](#) &p)
- std::istream & [operator>>](#) (std::istream &i, [Person](#) &p)

6.4.1 Function Documentation

6.4.1.1 operator<<()

```

std::ostream & operator<< (
    std::ostream & o,
    const Person & p )

```

Išvedimo srauto perrašymas.

Parameters

<i>o</i>	output stautas
<i>p</i>	Žmogus p, kuris paduodamas srautą.

Returns

srautas o

```

123                                     {
124     o<<p.pimpl->firstName;
125     o<<" "<<p.pimpl->lastName<<" ";
126     std::ostringstream ss;
127     ss<<p.parentp->Time;
128     o<<ss.str()<<" ";
129     o<<p.ID<<" ";
130     o<<p.type<<" ";
131     PersonPrint* pri = dynamic_cast<PersonPrintFullName*>(p.prt);
132     if(pri){
133         o<<"N\n";
134     }else{
135         o<<"I\n";
136     }
137     return o;
138 }
```

6.4.1.2 operator>>()

```

std::istream & operator>> (
    std::istream & i,
    Person & p )
```

Ivesties binarinio srauto perrašymas, kada būtų galima objektą perduoti į jį.

Parameters

<i>i</i>	input srautas
<i>p</i>	Žmogus p, kurio duomenys imami iš srauto.

Returns

input srautas

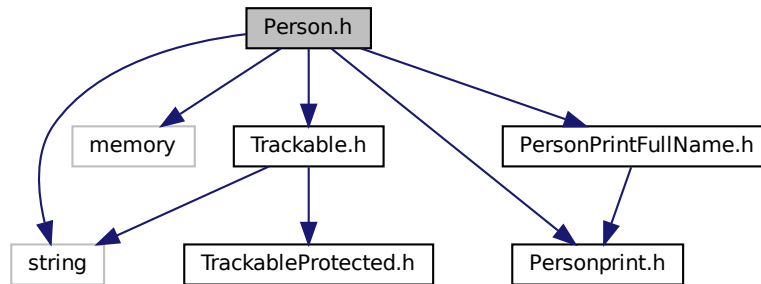
```

145                                     {
146     i>>p.pimpl->firstName;
147     i>>p.pimpl->lastName;
148     std::string raw, type;
149     i>>raw;
150     std::istringstream is(raw);
151     is>>p.parentp->Time;
152     i>>p.ID;
153     i>>p.type;
154     i>>type;
155     if(type == "N"){
156         p.prt = new PersonPrintFullName;
157     }else{
158         p.prt = new PersonPrintId;
159     }
160     return i;
161 }
```

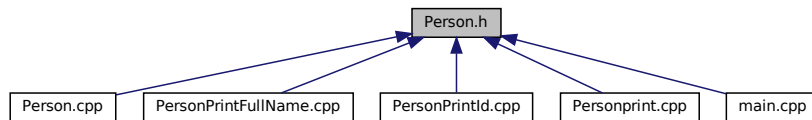
6.5 Person.h File Reference

```
#include <string>
#include <memory>
#include "Trackable.h"
#include "Personprint.h"
#include "PersonPrintFullName.h"
```

Include dependency graph for Person.h:



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



Classes

- class [Person](#)

6.6 Person.h

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

```
1 //
2 // Created by arnas on 2/12/24.
3 //
4
5 #ifndef CPP_2024_1_PERSON_H
6 #define CPP_2024_1_PERSON_H
7
8
9 #include <string>
10 #include <memory>
11
12
13
14 #include "Trackable.h"
```

```

15 #include "Personprint.h"
16 class Personprint;
17 #include "PersonPrintFullName.h"
18 class PersonPrintFullName;
19
20 class Person: public Trackable {
21 private:
22     class Impl;
23     Impl *pimpl;
24     TrackableProtected *parentp;
25 public:
26     Personprint *prt = nullptr;
27     static unsigned long NID;
28     Person(std::string name, std::string lastName);
29     Person();
30     std::string getName() const;
31     void print() const override;
32     ~Person();
33     Person(const Person& other);
34     Person& operator=(const Person& other);
35     Person& operator=(Person&& other);
36     static void changeName(std::string NewName, Trackable *obj);
37     friend std::ostream & operator << (std::ostream &o, const Person &p);
38     friend std::istream & operator >> (std::istream &i, Person &p);
39 };
40
41
42 #endif //CPP_2024_1_PERSON_H

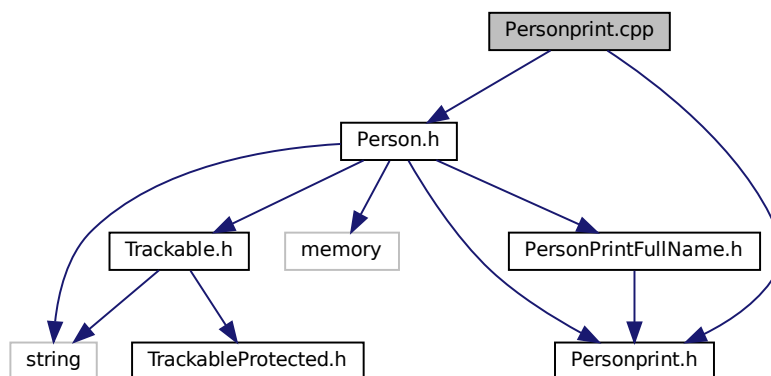
```

6.7 Personprint.cpp File Reference

```
#include "Personprint.h"
```

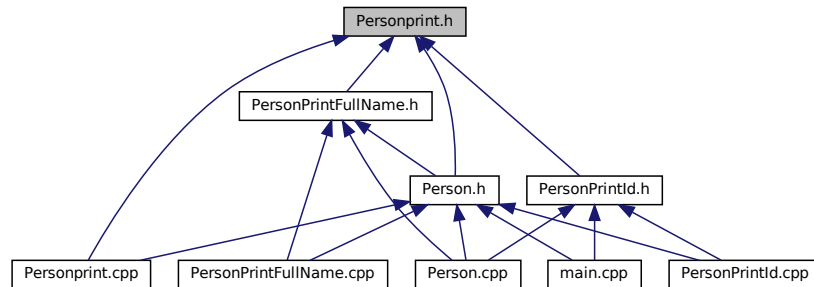
```
#include "Person.h"
```

Include dependency graph for Personprint.cpp:



6.8 Personprint.h File Reference

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



Classes

- class [Personprint](#)

6.9 Personprint.h

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

```

1 //
2 // Created by arnas on 4/28/24.
3 //
4
5 #ifndef CPP_2024_1_PERSONPRINT_H
6 #define CPP_2024_1_PERSONPRINT_H
7
8
9 class Person;
10
11 class Personprint { //Interface
12
13 public:
14     virtual void print(const Person &p);
15 };
16
17
18 #endif //CPP_2024_1_PERSONPRINT_H

```

6.10 PersonPrintFullName.cpp File Reference

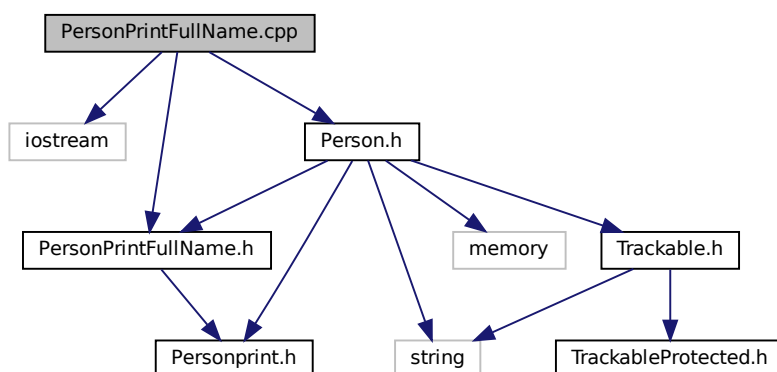
```

#include <iostream>
#include "PersonPrintFullName.h"

```

```
#include "Person.h"
```

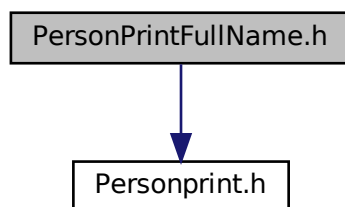
Include dependency graph for PersonPrintFullName.cpp:



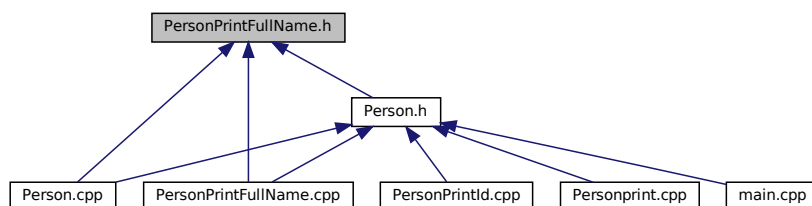
6.11 PersonPrintFullName.h File Reference

```
#include "Personprint.h"
```

Include dependency graph for PersonPrintFullName.h:



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



Classes

- class [PersonPrintFullName](#)

6.12 PersonPrintFullName.h

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

```

1 //
2 // Created by arnas on 4/28/24.
3 //
4
5 #ifndef CPP_2024_1_PERSONPRINTFULLNAME_H
6 #define CPP_2024_1_PERSONPRINTFULLNAME_H
7
8
9
10 #include "Personprint.h"
11
12 class PersonPrintFullName : public Personprint{
13 public:
14     void print(const Person &p) override;
15
16 };
17
18
19 #endif //CPP_2024_1_PERSONPRINTFULLNAME_H

```

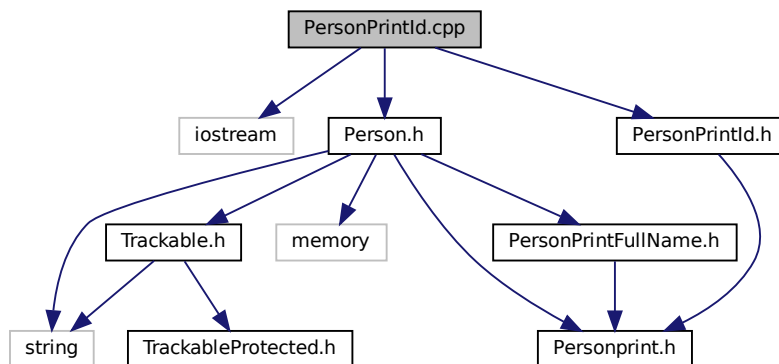
6.13 PersonPrintId.cpp File Reference

```

#include <iostream>
#include "PersonPrintId.h"
#include "Person.h"

```

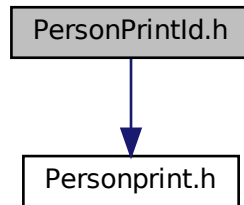
Include dependency graph for PersonPrintId.cpp:



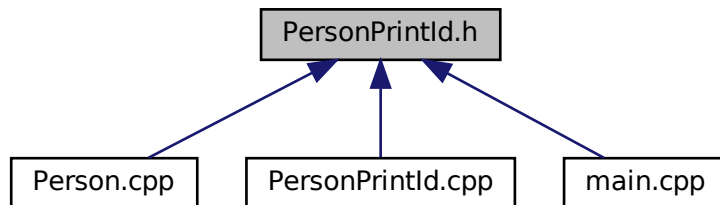
6.14 PersonPrintId.h File Reference

```
#include "Personprint.h"
```

Include dependency graph for PersonPrintId.h:



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



Classes

- class [PersonPrintId](#)

6.15 PersonPrintId.h

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

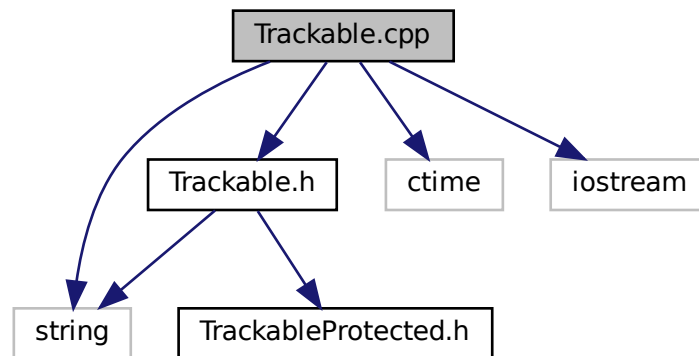
```
1 //
2 // Created by arnas on 4/28/24.
3 //
4
5 #ifndef CPP_2024_1_PERSONPRINTID_H
6 #define CPP_2024_1_PERSONPRINTID_H
7
8
9 #include "Personprint.h"
10
11 class PersonPrintId : public Personprint{
12 public:
13     void print(const Person &p) override;
14
15 };
16
17
18 #endif //CPP_2024_1_PERSONPRINTID_H
```

6.16 README.md File Reference

6.17 Trackable.cpp File Reference

```
#include "Trackable.h"  
#include <string>  
#include <ctime>  
#include <iostream>
```

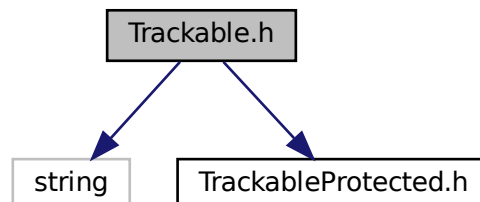
Include dependency graph for Trackable.cpp:



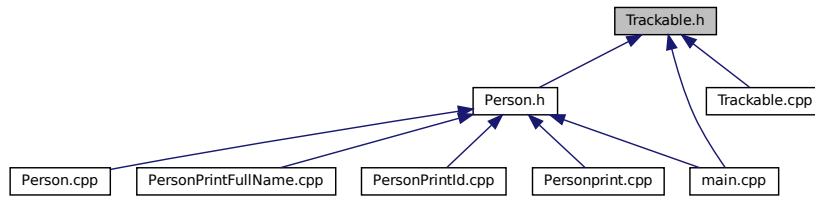
6.18 Trackable.h File Reference

```
#include <string>  
#include "TrackableProtected.h"
```

Include dependency graph for Trackable.h:



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



Classes

- class [Trackable](#)

6.19 Trackable.h

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

```

1 //
2 // Created by arnas on 2/25/24.
3 //
4
5 #ifndef CPP_2024_1_TRACKABLE_H
6 #define CPP_2024_1_TRACKABLE_H
7 #include <string>
8 #include "TrackableProtected.h"
9
10 class Trackable {
11 private:
12     TrackableProtected *proc;
13 protected:
14     const TrackableProtected* getProtected() const;
15     TrackableProtected* getProtected();
16 public:
17     virtual std::string getName() const;
18     unsigned type;
19     static unsigned long amount;
20     unsigned long ID;
21     Trackable(std::string Name);
22     Trackable();
23     virtual void print() const;
24     Trackable(const Trackable& other);
25     Trackable& operator=(const Trackable& other);
26     Trackable& operator=(Trackable&& other) noexcept;
27     ~Trackable();
28 };
29
30
31 #endif //CPP_2024_1_TRACKABLE_H
  
```

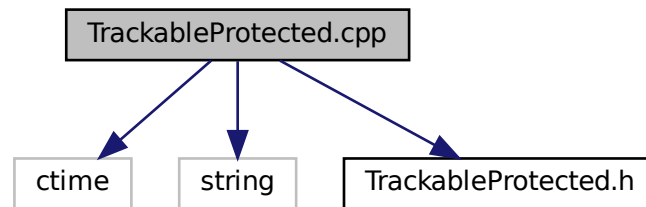
6.20 TrackableProtected.cpp File Reference

```

#include <ctime>
#include <string>
  
```

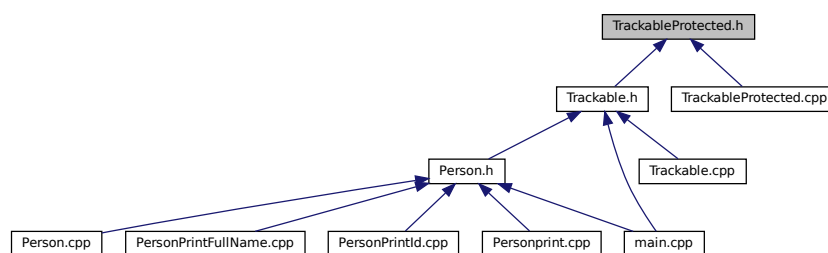
```
#include "TrackableProtected.h"
```

Include dependency graph for TrackableProtected.cpp:



6.21 TrackableProtected.h File Reference

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



Classes

- class [TrackableProtected](#)

6.22 TrackableProtected.h

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

```

1 //
2 // Created by arnas on 3/24/24.
3 //
4
5 #ifndef CPP_2024_1_TRACKABLEPROTECTED_H
6 #define CPP_2024_1_TRACKABLEPROTECTED_H
7
8
9 class TrackableProtected {
10 public:
11     static unsigned long NID;
12     std::string Name;
13     time_t Time;
14     TrackableProtected(const TrackableProtected *tra);
15     TrackableProtected(std::string name);
16     TrackableProtected();
17 };
18
19
20 #endif //CPP_2024_1_TRACKABLEPROTECTED_H
  
```

Index

- __has_include
 - CMakeCCompilerId.c, [29](#)
 - CMakeCXXCompilerId.cpp, [33](#)
 - ~Person
 - Person, [13](#)
 - ~Trackable
 - Trackable, [23](#)
- amount
 - Trackable, [25](#)
- ARCHITECTURE_ID
 - CMakeCCompilerId.c, [30](#)
 - CMakeCXXCompilerId.cpp, [33](#)
- C_VERSION
 - CMakeCCompilerId.c, [30](#)
- changeName
 - Person, [13](#)
- cmake-build-debug/CMakeFiles/3.27.0/CompilerIdC/CMakeCCompilerId.c,
 - [29](#)
- cmake-build-debug/CMakeFiles/3.27.0/CompilerIdCXX/CMakeCXXCompilerId.cpp,
 - [32](#)
- CMakeCCompilerId.c
 - __has_include, [29](#)
 - ARCHITECTURE_ID, [30](#)
 - C_VERSION, [30](#)
 - COMPILER_ID, [30](#)
 - DEC, [30](#)
 - HEX, [30](#)
 - info_arch, [31](#)
 - info_compiler, [32](#)
 - info_language_extensions_default, [32](#)
 - info_language_standard_default, [32](#)
 - info_platform, [32](#)
 - main, [31](#)
 - PLATFORM_ID, [30](#)
 - STRINGIFY, [31](#)
 - STRINGIFY_HELPER, [31](#)
- CMakeCXXCompilerId.cpp
 - __has_include, [33](#)
 - ARCHITECTURE_ID, [33](#)
 - COMPILER_ID, [33](#)
 - CXX_STD, [33](#)
 - DEC, [33](#)
 - HEX, [34](#)
 - info_arch, [35](#)
 - info_compiler, [35](#)
 - info_language_extensions_default, [35](#)
 - info_language_standard_default, [35](#)
 - info_platform, [36](#)
- main, [34](#)
- PLATFORM_ID, [34](#)
- STRINGIFY, [34](#)
- STRINGIFY_HELPER, [34](#)
- COMPILER_ID
 - CMakeCCompilerId.c, [30](#)
 - CMakeCXXCompilerId.cpp, [33](#)
- CXX_STD
 - CMakeCXXCompilerId.cpp, [33](#)
- DEC
 - CMakeCCompilerId.c, [30](#)
 - CMakeCXXCompilerId.cpp, [33](#)
- firstName
 - Person::Impl, [10](#)
- getName
 - Person, [14](#)
 - Trackable, [23](#)
- getProtected
 - Trackable, [23](#), [24](#)
- HEX
 - CMakeCCompilerId.c, [30](#)
 - CMakeCXXCompilerId.cpp, [34](#)
- ID
 - Trackable, [25](#)
- Impl
 - Person::Impl, [9](#), [10](#)
- info_arch
 - CMakeCCompilerId.c, [31](#)
 - CMakeCXXCompilerId.cpp, [35](#)
- info_compiler
 - CMakeCCompilerId.c, [32](#)
 - CMakeCXXCompilerId.cpp, [35](#)
- info_language_extensions_default
 - CMakeCCompilerId.c, [32](#)
 - CMakeCXXCompilerId.cpp, [35](#)
- info_language_standard_default
 - CMakeCCompilerId.c, [32](#)
 - CMakeCXXCompilerId.cpp, [35](#)
- info_platform
 - CMakeCCompilerId.c, [32](#)
 - CMakeCXXCompilerId.cpp, [36](#)
- lastName
 - Person::Impl, [10](#)
- main

- CMakeCCompilerId.c, 31
 - CMakeCXXCompilerId.cpp, 34
 - main.cpp, 36
- main.cpp, 36
 - main, 36
 - printAmount, 37
- Name
 - TrackableProtected, 27
- NID
 - Person, 17
 - TrackableProtected, 27
- operator<<
 - Person, 15
 - Person.cpp, 38
- operator>>
 - Person, 16
 - Person.cpp, 39
- operator=
 - Person, 14
 - Trackable, 24
- parentp
 - Person, 17
- Person, 11
 - ~Person, 13
 - changeName, 13
 - getName, 14
 - NID, 17
 - operator<<, 15
 - operator>>, 16
 - operator=, 14
 - parentp, 17
 - Person, 12, 13
 - pimpl, 17
 - print, 15
 - prt, 17
- Person.cpp, 38
 - operator<<, 38
 - operator>>, 39
- Person.h, 40
- Person::Impl, 9
 - firstName, 10
 - Impl, 9, 10
 - lastName, 10
- Personprint, 17
 - print, 18
- Personprint.cpp, 41
- Personprint.h, 42
- PersonPrintFullName, 18
 - print, 19
- PersonPrintFullName.cpp, 42
- PersonPrintFullName.h, 43
- PersonPrintId, 19
 - print, 20
- PersonPrintId.cpp, 44
- PersonPrintId.h, 45
- pimpl
 - Person, 17
- PLATFORM_ID
 - CMakeCCompilerId.c, 30
 - CMakeCXXCompilerId.cpp, 34
- print
 - Person, 15
 - Personprint, 18
 - PersonPrintFullName, 19
 - PersonPrintId, 20
 - Trackable, 25
- printAmount
 - main.cpp, 37
- proc
 - Trackable, 25
- prt
 - Person, 17
- README.md, 46
- STRINGIFY
 - CMakeCCompilerId.c, 31
 - CMakeCXXCompilerId.cpp, 34
- STRINGIFY_HELPER
 - CMakeCCompilerId.c, 31
 - CMakeCXXCompilerId.cpp, 34
- Time
 - TrackableProtected, 27
- Trackable, 21
 - ~Trackable, 23
 - amount, 25
 - getName, 23
 - getProtected, 23, 24
 - ID, 25
 - operator=, 24
 - print, 25
 - proc, 25
 - Trackable, 22, 23
 - type, 25
- Trackable.cpp, 46
- Trackable.h, 46
- TrackableProtected, 26
 - Name, 27
 - NID, 27
 - Time, 27
 - TrackableProtected, 26, 27
- TrackableProtected.cpp, 47
- TrackableProtected.h, 48
- type
 - Trackable, 25