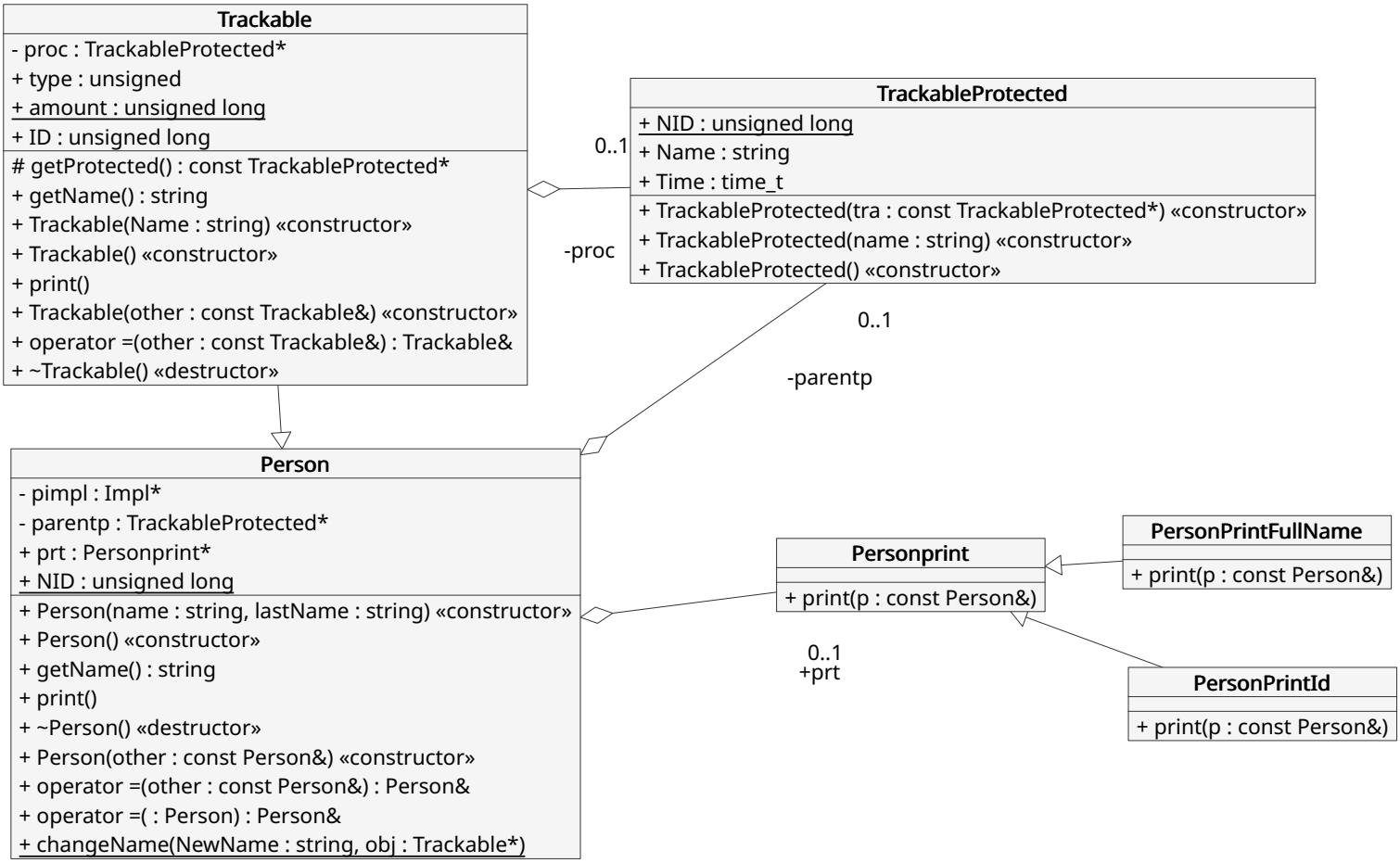

Trackable
- proc : TrackableProtected*
+ type : unsigned
+ amount : unsigned long
+ ID : unsigned long
# getProtected() : const TrackableProtected*
+ getName() : string
+ Trackable(Name : string) «constructor»
+ Trackable() «constructor»
+ print()
+ Trackable(other : const Trackable&) «constructor»
+ operator =(other : const Trackable&) : Trackable&
+ ~Trackable() «destructor»
0..1
-proc
TrackableProtected
+ NID : unsigned long
+ Name : string
+ Time : time_t
+ TrackableProtected(tra : const TrackableProtected*) «constructor»
+ TrackableProtected(name : string) «constructor»
+ TrackableProtected() «constructor»
0..1
-parentp
Person
- pimpl : Impl*
- parentp : TrackableProtected*
+ prt : Personprint*
+ NID : unsigned long
+ Person(name : string, lastName : string) «constructor»
+ Person() «constructor»
+ getName() : string
+ print()
+ ~Person() «destructor»
+ Person(other : const Person&) «constructor»
+ operator =(other : const Person&) : Person&
+ operator =(: Person) : Person&
+ changeName(NewName : string, obj : Trackable*)
Personprint
+ print(p : const Person&)
0..1
+prt
PersonPrintFullName
+ print(p : const Person&)
PersonPrintId
+ print(p : const Person&)

Registras

Generated by Doxygen 1.9.4

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, todel 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 nauojus VU VPN

Žmonių duomenų įršai gali būti srautais rašomi bei skaitomi iš failų. Taip pat gaalima vykdymo metu pakeisti ką daro "print" funkcija. Tam kad tai padaryti, reikia viešą lauką prt pakeisti klasės objektu, kuris įgyvendina "interface"'ą Personprint. Iškviečiant print funkcija yra iskvieciama klasėje esantčio Persionprint$*$ objekto print.

Serializacija, vykdoma binariniais srautais.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

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

Fname	vardas
Lname	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

oth	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
```

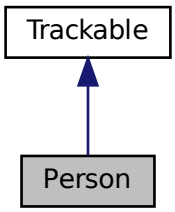
The documentation for this struct was generated from the following file:

- Person.cpp

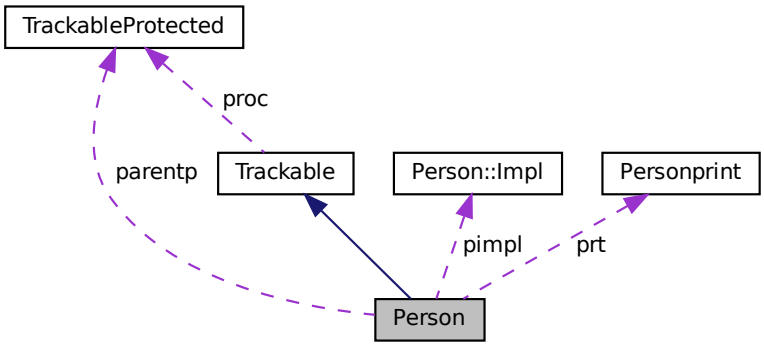
5.2 Person Class Reference

`#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 ∗ prt = 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

name	Žmogaus vardas
lastName	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

other	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

NewName	naujas vardas.
obj	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

other	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

other	

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«"
       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

o	output stautas
p	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	input sraautas
p	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
```

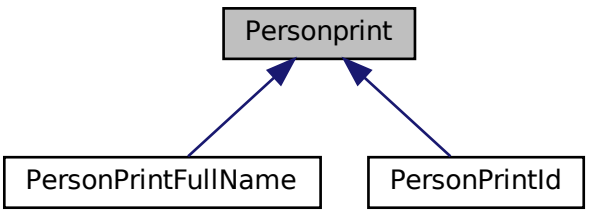
The documentation for this class was generated from the following files:

- 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

p	Zmogus is kurio iskviesta funkcija

Reimplemented in PersonPrintFullName, and PersonPrintId.

```
11                                      {
12
13 }
```

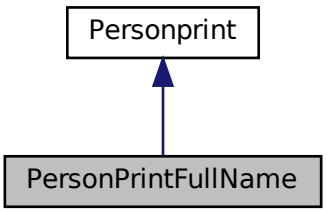
The documentation for this class was generated from the following files:

- Personprint.h
- Personprint.cpp

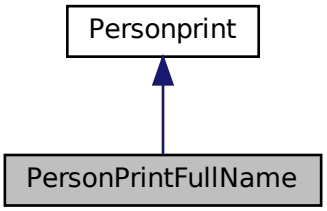
5.4 PersonPrintFullName Class Reference

```
#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

p	Zmogus is kurio iskviesta funkcija

Reimplemented from Personprint.

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

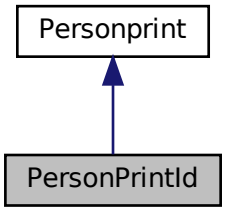
The documentation for this class was generated from the following files:

- PersonPrintFullName.h
- PersonPrintFullName.cpp

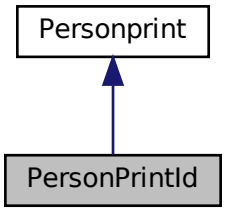
5.5 PersonPrintId Class Reference

```
#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

p	Zmogus is kurio iskviesta funkcija

Reimplemented from Personprint.

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

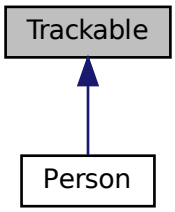
The documentation for this class was generated from the following files:

- PersonPrintId.h
- PersonPrintId.cpp

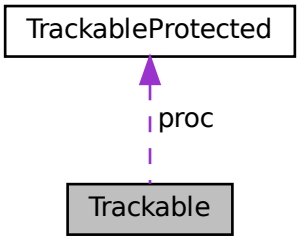
5.6 Trackable Class Reference

`#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

Name	

```
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 construktorius

Parameters

other	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

other	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

other	Trackable is kurio imti

Returns

Moveintas 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

tra	TrackableProtected, kuris kopijuojamas

```
12                                                          :  Time(tra->Time) {
13 Name = tra->Name;
```

```
14 }
```

5.7.1.2 TrackableProtected() [2/3]

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

Paprastaas konstruktorius

Parameters

name	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

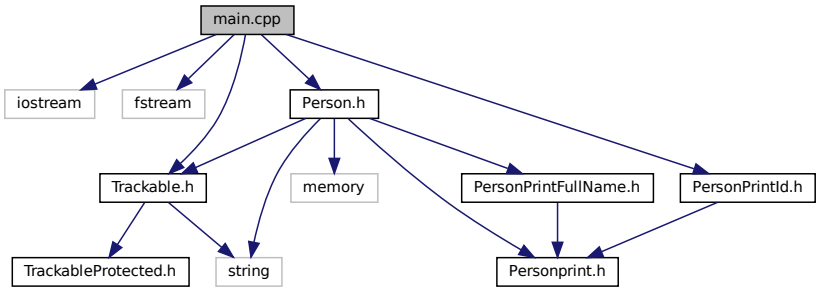
Chapter 6

File Documentation

6.1 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 ()

6.1.1 Function Documentation

6.1.1.1 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
48     }catch(std::bad_cast) {
49         std::cout«"Klaida keiciant 6\n";
50     }
51     try {
52    //     Person::changeName("Vardenis", Entries[0]);
53         Person::changeName("Vardenis", Ent[0]);
54     }catch(std::bad_cast) {
55         std::cout«"Klaida keiciant 0\n";
56     }
57     Person *cast = dynamic_cast<Person*>(Ent[6]);
58     out«*cast;
59     out.close();
60    /* for(auto Entry :  Entries){
61 Entry->print();
62 }*/
63     for(int i = 0; i < 7; i++){
64         Ent[i]->print();
65     }
66     printAmount();
67     //delete Entries[6];
68     delete Ent[6];
69     printAmount();
70     Person *p = new Person("A", "B");
71     p->print();
72     p->prt=new PersonPrintId();
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 }
```

6.1.1.2 printAmount()

```
void printAmount ( )
```

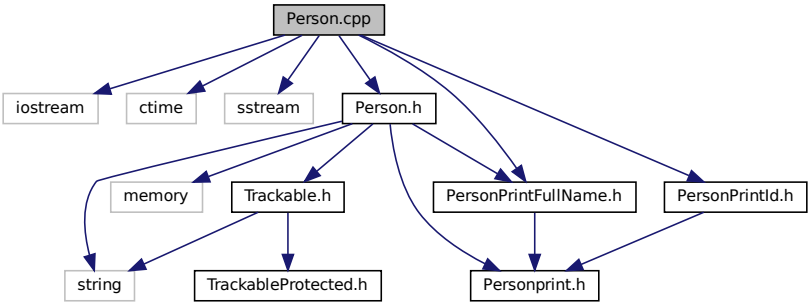
Funkcija atspaauzdinti registre esanciu elementu kieki.

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

6.2 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)

6.2.1 Function Documentation

6.2.1.1 operator<<()

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

Išvedimo srauto perrašymas.

Parameters

o	output stautas
p	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 }
```

6.2.1.2 operator>>()

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

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

Parameters

i	input sraautas
p	Žmogus p, kurio duomenys imami iš srauto.

Returns

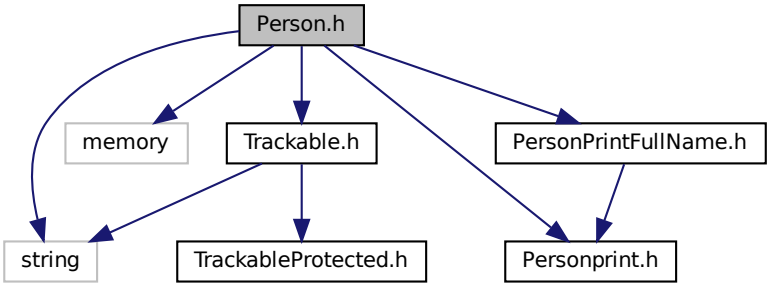
input srautas

```
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 }
```

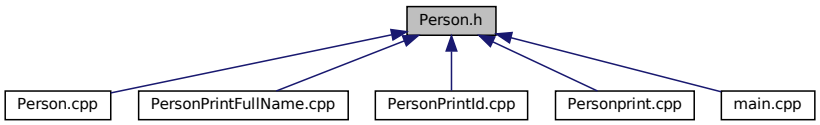
6.3 Person.h File Reference

```
#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

6.4 Person.h

Go to the documentation of this file.

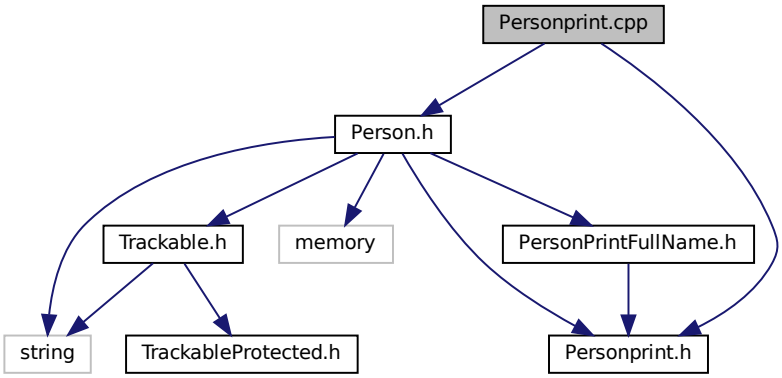
```
1 //
2 // Created by arnas on 2/12/24.
3 //
4
5 #ifndef CPP_2024_1_PERSON_H
6 #define CPP_2024_1_PERSON_H
7
8
9 #include <string>
10 #include <memory>
11
12
13
14 #include "Trackable.h"
```

```
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
```

6.5 Personprint.cpp File Reference

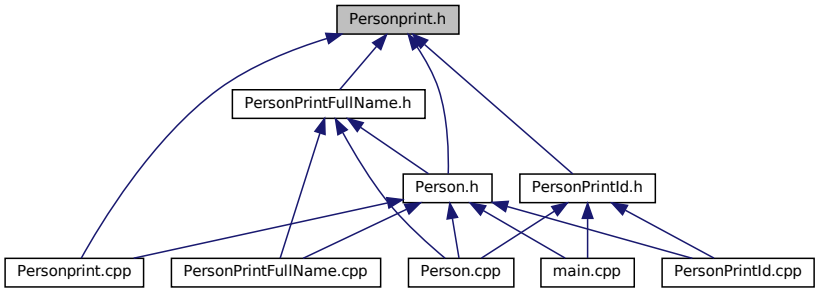
```
#include "Personprint.h"
#include "Person.h"
```

Include dependency graph for Personprint.cpp:



6.6 Personprint.h File Reference

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



Classes

- class Personprint

6.7 Personprint.h

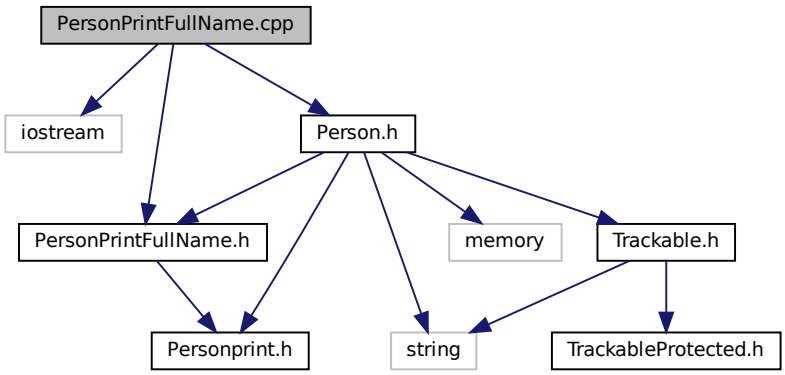
Go to the documentation of this file.

```
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
```

6.8 PersonPrintFullName.cpp File Reference

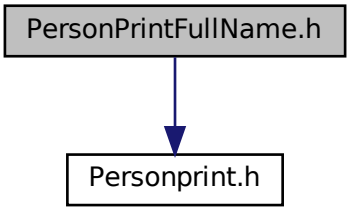
```
#include <iostream>
#include "PersonPrintFullName.h"
```

```
#include "Person.h"
```
Include dependency graph for PersonPrintFullName.cpp:

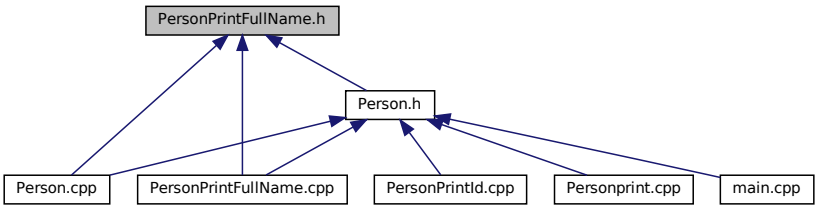


6.9 PersonPrintFullName.h File Reference

```
#include "Personprint.h"
```
Include dependency graph for PersonPrintFullName.h:



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



Classes

- class PersonPrintFullName

6.10 PersonPrintFullName.h

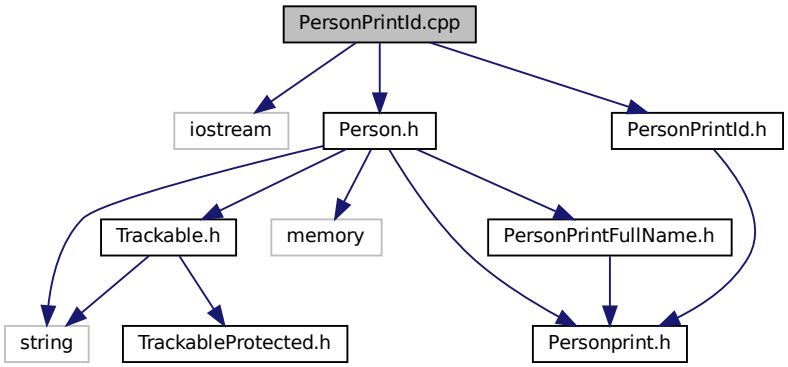
Go to the documentation of this file.

```
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
```

6.11 PersonPrintId.cpp File Reference

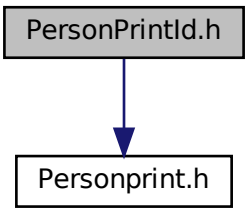
```
#include <iostream>
#include "PersonPrintId.h"
#include "Person.h"
```

Include dependency graph for PersonPrintId.cpp:

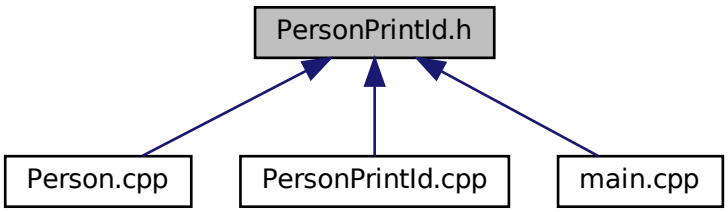


6.12 PersonPrintId.h File Reference

`#include "Personprint.h"`
Include dependency graph for PersonPrintId.h:



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



Classes

- class PersonPrintId

6.13 PersonPrintId.h

Go to the documentation of this file.

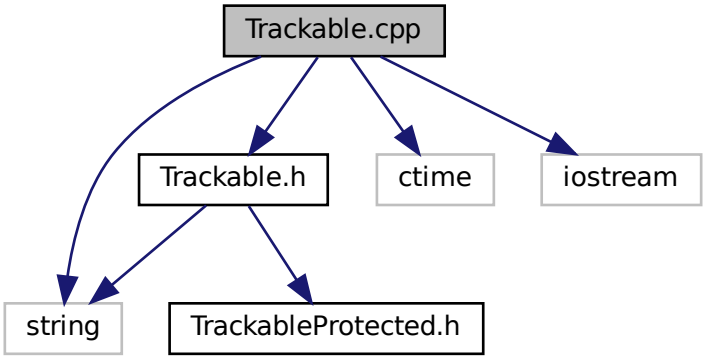
```
1 //
2 // Created by arnas on 4/28/24.
3 //
4
5 #ifndef CPP_2024_1_PERSONPRINTID_H
6 #define CPP_2024_1_PERSONPRINTID_H
7
8
9 #include "Personprint.h"
10
11 class PersonPrintId :  public Personprint{
12 public:
13     void print(const Person &p) override;
14
15 };
16
17
18 #endif //CPP_2024_1_PERSONPRINTID_H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

6.14 README.md File Reference

6.15 Trackable.cpp File Reference

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

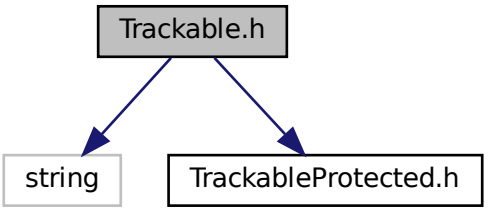
Include dependency graph for Trackable.cpp:



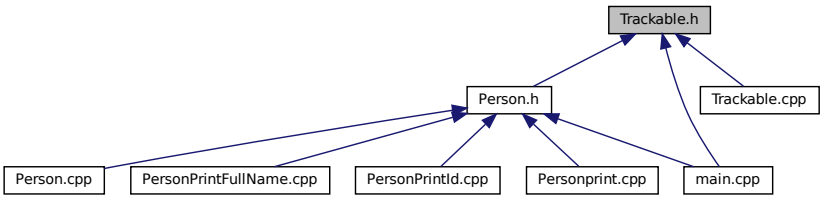
6.16 Trackable.h File Reference

```
#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

6.17 Trackable.h

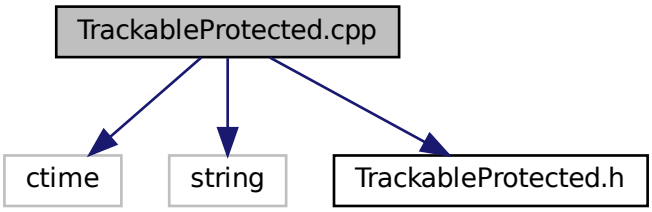
Go to the documentation of this file.

```
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
```

6.18 TrackableProtected.cpp File Reference

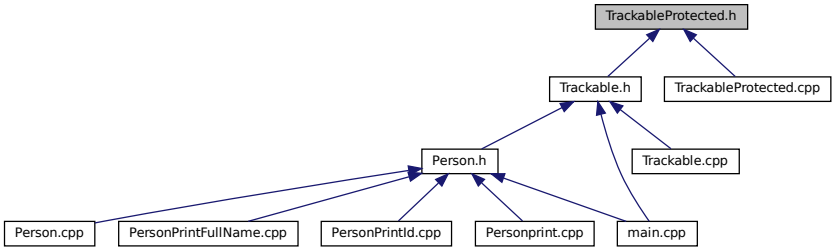
```
#include <ctime>
#include <string>
```

```
#include "TrackableProtected.h"
```
Include dependency graph for TrackableProtected.cpp:



6.19 TrackableProtected.h File Reference

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



Classes

- class TrackableProtected

6.20 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