EESZT

Készítette Nemes Lili

Pontosított specifikáció és osztálydiagram

Created	@April 20, 2022 7:20 PM
Tags	GLJD1D Nemes Lili

A feladat célja

A feladat tulajdonképpen egy orvosi nyilvántartórendszer leegyszerűsített modellje. Az ábrázolt folyamat egy háziorvosi rendelést ír le: a beteg elmegy a háziorvoshoz, onnan valamilyen vizsgálati eredménnyel távozik, mely vagy von maga után további intézkedéseket, vagy nem. Amennyiben igen, a beteg kérhet beutalót a szakorvoshoz, aki időpontot ad neki, vagy fájlba kiírattathatja a receptet, oltási igazolást. Ehhez a keretet egy olyan rendszer adja, melybe a program felhasználója orvosok, betegek, szakorvosok profiljaival belépve különböző akciókat hajthat végre.

A program működése

A program először arra kéri a felhasználót, hogy válasszon, háziorvosként betegként vagy szakorvosként kíván belépni a rendszerbe. Ezt a felhasználó a kívánt személyhez kapcsolódó kóddal jelezheti. Ha nem belépni akar, hanem már rögzített felállást betölteni fájlból, vagy a jelenlegi állapotot fájlba menteni akkor ezeket is megteheti.

```
1. Physician
2. Patient
3. Specialist
4. Import state
5. Export state
6. Exit

1.
```

Háziorvos

A felhasználónak meg kell adnia az orvos nevét, mely az orvosnak (és úgy általában, bármelyik profilnak) egyedi azonosítója lesz (nem lehet más ugyanilyen nevű profil! Erre a felhasználónak kell figyelnie, mivel a program hibás jelszóként fogja értelmezni a

belépési kísérletet). Ha korábban még nem használta a profilját, akkor meg kell adnia egy jelszót, ami a TAJ száma (tehát szintén egyedi azonosító), ezzel létrehozva azt. Mostantól ez a jelszó tartozik a megadott névhez, a felhasználó pedig belépett a rendszerbe a profillal. Ha már van az adott névhez tartozó profil, akkor meg kell adnia a korábban megadott jelszót, különben nem tud belépni. Ha a felhasználó elfelejtette a jelszavát kérhet emlékeztetőt, ekkor a program fájlba kiírja az elfelejtett jelszót. Amennyiben a felhasználó nem a felsorolt lehetőségekhez tartozó kódok egyikét adja meg, a program leáll.

```
name: Dr. Kovacs Istvan
password:
(in case of forgotten password press 111111!)
```

```
//password.txt fájl tartalma:
123456
```

📌 A jelszónak 6 karakternek kell lennie. Nem lehet jelszó az 111111 sorozat, mivel ez a kombináció az elfelejtett jelszóhoz tartozik.

Orvosként belépve a felhasználó 3 főbb funkció közül választhat.

```
1. Admittance
2. Care activity
3. Exit
```

Új beteg felvétele ~ Admittance

Ha az orvoshoz új beteg érkezik akkor fel kell vegye az adatait (név TAJ szám).

📌 A betegek TAJ száma egy 6 karakterből álló sorozat. A TAJ szám egyedi azonosító!

Betegek adatainak megnézése, beteggel kapcsolatos teendők ~ Care activity

Ide belépve a betegek adatai fogadják az orvost. Egy adott beteget a TAJ száma beírásával választhat ki.

A pácienst kiválasztva többféle cselekvést hajthat végre:

1.) Beutalót adhat szakorvoshoz

2.) Receptet írhat fel

3.) Megjelölheti, hogy a beteg megkapta az oltást

4.) Megjegyzést írhat

5.) A kezelést befejezettnek jelöli. (Kilépés)

> 📌 Amíg szakorvos nem jelentkezett be a rendszerbe csak háziorvosi ellátásra van lehetőség!

Kilépés

Ezt választva a felhasználó kilép az orvos profiljából, a kezdőoldalra kerül vissza.

Beteg

Beteg akkor léphet be a rendszerbe, ha már járt a háziorvosnál. Belépéskor a nevét és a jelszavát, ami a TAJ száma, kell megadnia. Elfelejtett jelszó esetén a megszokott 111111 kombinációval kérhet segítséget.

Belépve

1.) Megtekintheti, hogy van e kiírt beutalója, illetve ezt jelezheti egy adott típusú szakorvos felé időpontkérés formájában. (egyszerre egy beutalót tud megtekinteni, ha több is van neki, akkor a menüpont újbóli megnyomásával tudja azt megnézni)

```
0. Exit
1. Ophtalmology
1. Dr. Kovacs Istvan
2. Surgery
1. Dr. Kiss Virag
```

2.) Megnézheti azt is, hogy van-e már időpontja valamilyen szakorvoshoz.

3.) Megtekintheti ha van számára felírt recept és exportálhatja azt fájlba

```
//prescription.txt tartalma
Nurofen painkiller 400mg
```

4.) Ahogy az oltási igazolványát is (abban az esetben ha kapott oltást).

```
//vaccination.txt tartalma
Kiss Virag has alreagdy got the vaccine.
```

5.) Kilépéskor a felhasználó kilép a beteg profiljából és a kezdőoldalra kerül vissza.

Szakorvos

Szakorvosként történő belépéskor ugyanaz, mint orvosnál, de itt a szakot is meg kell adni, egy listából kiválasztva.

```
Type:
  1. Otolaryngology
  2. Surgery
  3. Internal medicine
  4. Ophthalmology

Name: Dr. Kovacs Janos
Password: 345678
Type: 2.
```

Szakorvosként belépve a felhasználó (1 / 2 / 3 karakterekkel)

1.) A szakorvoshoz érkezett beutaló kérelmeket tekintheti meg. Ezek közül egyet-egyet kiválasztva a naptárát megtekintve adhat időpontokat a betegeknek (a naptár március hónapra szól, minden napra 4 foglalási időpont van).

```
1. Jozsef Attila - 123456
2. Ady Endre - 987654
```

```
//naptár naponként sorokban, foglalt időponton TAJszám, szabadon 0
1 23KJ67 0 0 0
2 0 0 0 0
3 0 12RT56 JK7654 0
.
.
.
31 0 0 0 0
//Választás nap időpont száma formátumban:
Day 31
Section 1
```

2.) A beteghez megjegyzést írhat.

```
//orvos korábbi megjegyzése
Surgical intervention needed.
//szakorvos mostani megjegyzése
Surgical intervention succesfully done.
```

📌 Egy időpontra értelemszerűen csak egy beteg érkezhet.

Kilépéskor a felhasználó kilép a szakorvos profiljából és a kezdőoldalra kerül vissza.

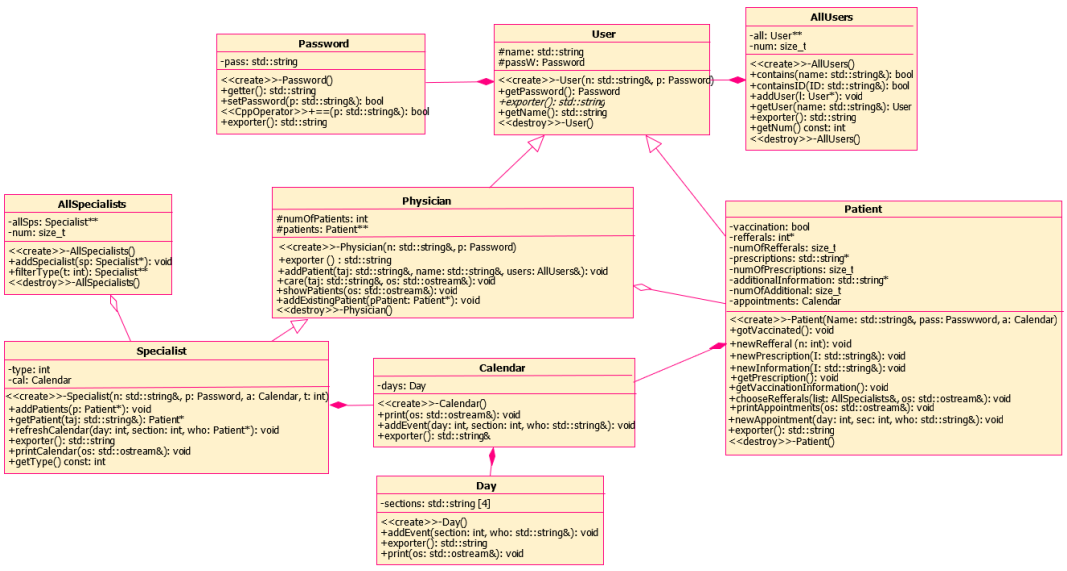
Import state

Fájlba mentet adatokból rekonstruálja a rendszert, újra létrehozza a rendszer felhasználóinak (orvosok, betegek) profiljait. Csak akkor lehet importálni, ha még nem hoztunk létre egyetlen profilt sem. (Az AllUsers és az AllSPecialists heterogén kollekciók üresek).

Export state

Az adott rendszer felhasználóinak profiljait fájlba exportálja (minden felhasználót az összes hozzájuk tartozó információval együtt).

Osztálydiagram



Lesz egy modul a menü kezelésére, egy a vezérlésre, egy a tesztelésre és egy-egy minden osztály számára. A jelszavak stringként, a felhasználók dinamikus tömbben rájuk mutató pointerekkel-, a többi szövegesen tárolt információ pedig dinamikus, stringeket tároló tömbben lesz eltárolva. A programmal a standard inputon keresztül lehet kommunikálni.

Kiegészítő magyarázat az UML diagramhoz

A program alapja, hogy a felhasználó különböző típusú profilokba léphet be, így a User absztrakt ősosztálynak több leszármazottja is van (azért absztrakt, mivel nincs lehetőség olyan felhasználó létrehozására aki nem tartozik a később említett kategóriákba). Beléphet betegként (patient) illetve orvosként, azon belül is háziorvosként(physician), vagy szakorvosként (specialist). A szakorvos azért a háziorvos leszármazott osztálya, mivel vannak közös attribútumaik is, illetve vannak feladatok, melyeket mindketten elláthatnak - azonban a szakorvoshoz tartoznak saját attribútumok is: a típusa (*int type*) illetve egy kalendár (*Calendar cal*). Mivel a programnak el kell tárolnia az összes létrehozott profilt ezért szükség van az allUsers **heterogén kollekció**ra, mely egy User pointereket tároló tömbben tartja nyilván a program felhasználója által regisztrált összes különböző típusú profilt.

Első belépés egy profillal

Egy, a vezérlő modulba tartozó függvény írja ki a menüt, amelyből kiválaszthatja a felhasználó, hogy milyen profillal kíván belépni. Ezek után meg kell adnia egy nevet, illetve egy jelszót (ekkor ellenőrzi azt a vezérlő függvény, hogy benne van-e a heterogén kollekcióban az adott profil, azaz korábban létrehozták-e már. Ezt a program elején automatikusan létrehozott AllUsers objektumnak a *contains(string)* illetve a *containsID(string)* tagfüggvényének a segítségével teheti meg. A contains csak a név egyediségét ellenőrzi, de mivel annak a felhasználó felelősségére egyedi azonosítónak kell lennie így ez a programrész működőképes). A contains ID a TAJ szám egyedi azonosító jellegét ellenőrzi, kivételt dob, ha olyan TAJ számot adunk meg ami már létezik. Először meghívódik a Password osztály konstruktora, mely egy üres stringet hoz létre, majd pedig (első bejelentkezés esetén) meghívódik a *setPassword(string)* függvény. Amennyiben a felhasználó érvényes jelszót adott meg, a profilhoz tartozó jelszó (Password passW) értéke arra módosul és a függvény visszatérési értéke igaz. Amennyiben a felhasználó érvénytelen jelszót adott meg, a visszatérési érték hamis és új jelszót kell megadjon. Ezek után meghívódik az adott típusú profil konstruktora - szakorvos esetén választani kell egy szakot is a vezérlő függvény által kiírt listából. Szintén a szakorvos sajátossága, hogy létrehozáskor

bekerül a program elején létrehozott allSpecialists objektum Specialist pointereket tároló attribútumába.

Háziorvos funkciói

A háziorvos felvehet új beteget az *addPatient(int, string, allUsers*) függvény segítségével. Ekkor létrejön egy Patient objektum, a háziorvos attribútumai közé tartozó, Patient mutatókat tároló patients tömbbe bekerül az adott páciens, illetve a paraméterként kapott allUsers heterogén kollekcióba is bekerül a profil. Amennyiben a beteg profilja már létezik ez nem történhet meg. Ezek után a háziorvos *showPatients (ostream&)* függvénye segítségével kiíródnak az orvos betegei. Választania kell közülük a tajszám kiírásával → meghívódik a *care(char*, ostream&*) függvény. A care függvényen belül kikereső dik a patients tömbből az a beteg, akinek a tajszámát a függvény paraméterként megkapta. Ekkor kiíródnak a menüpontok, melyek közül az orvos választhat.

1.) Beutalót adhat szakorvoshoz → meghívja a betegre a *newReferral(int)* függvényt, mely paraméterként a szakorvos típusát kapja.

2.) Receptet írhat fel → meghívja a betegre a *new Prescription(string)* függvényt, mely paraméterként string formájában kapja a felírt gyógyszer nevét, egyébb kiegészítő információkat.

3.) Megjelölheti, hogy a beteg megkapta az oltást → meghívja a betegre a *got Vaccinated()* függvényt

4.) Megjegyzést írhat → meghívja a betegre a *newInformation(string)* függvényt, mely paraméterként string formájában kapja a feljegyzés tartalmát.

5.) A kezelést befejezettnek jelöli. (Kilépés)

Beteg funkciói

A vezérlőfüggvény kiírja a betegnek, hogy milyen menüpontok közül választhat. A választás hatására hívódhatnak meg a következő függvények:

1.) Megtekintheti, hogy van e kiírt beutalója, illetve ezt jelezheti egy adott típusú szakorvos felé időpontkérés formájában. → A *chooseRefferals(AllSpecialists, ostream&*) függvény kiírja a beutalókat, illetve a paraméterként kapott AllSpecialists objektumból a *filterType(int)* és a Specialist *getType()* segítségével kiválogatja azokat a specialistákat, akik egy, a paraméterként átadott szakterületen dolgoznak. Majd a függvény ezeket a specifikációban leírt módon kiírja. Egy adott szakorvos kiválasztásakor a kiválasztott orvosra meghívódik az *add Patients(Patient*)*

függvény, mely a szakorvos Patient pointereket tároló patients tömbjébe teszi az adott Páciensre mutató pointert.

2.) Megnézheti azt is, hogy van-e már időpontja valamilyen szakorvoshoz. → *printAppointments(ostream&)*, kiírja a beteg kalendárját.

3.) Megtekintheti ha van számára felírt recept és exportálhatja azt fájlba → *getPrescription()*

4.) Ahogy az oltási igazolványát is (abban az esetben, ha kapott oltást). → *getVaccinationInformation*()

5.) Kilépés

Szakorvos funkciói

A vezérlőfüggvény kiírja a szakorvosnak, hogy milyen menüpontok közül választhat. A választás hatására hívódhatnak meg a következő függvények:

1.) A szakorvoshoz érkezett beutaló kérelmeket tekintheti meg. Ezek közül egyet-egyet kiválasztva a naptárát megtekintve adhat időpontokat a betegeknek (a naptár március hónapra szól, minden napra 4 foglalási időpont van). → először a *showPatients(ostream&*) segítségével kiíródnak a betegei. Majd miután a tajszámuk alapján választ egyet közülük a *getPatient(string)* segítségével, meghívja a *printCalendar()* függvényt, mely kiírja a naptárát(Calendar *print(ostream&*) illetve Day *print(ostream&*) segítségével). Ezek után kiválaszt és beír egy napot és egy idősávot, majd meghívódik a *refreshCalendar(int, int, Patient&*) függvény, mely:

1. meghívja a Specialist kalendárjának *addEvent(int, int, string*) függvényét, mely paraméterként kapja a korábban paraméterként kapott napot, idősávot, illetve a paraméterként kapott páciens tajszámát (mely a *getPassword()* függvény segítségével határozható meg). A tömb nap-odik elemére meghívja annak is az *addEvent (int, string)* függvényét és annak idősávodik elemére az alapértelmezett nulla helyett a páciens paraméterként kapott Taj-számát írja.
2. meghívja a paraméterként kapott Páciensre annak *newAppointment(int, int, string)* függvényét, mely hasonlóan viselkedik, mint az előző menüpontban említett függvények, csak a páciens TAJ száma helyett a szakorvos nevét írja bele(*getName()*).

2.) A beteghez megjegyzést írhat. → *showPatients(ostream&*), *getPatient(string)*, páciens *newInformation(string)* tagfüggvénye.

3.) Kilépés.

Többszöri belépés esetén

A program a beírt név alapján kikeresi a profilt a heterogén kollekcióból, a *getPassword()* segítségével megtudja annak jelszavát, az *== operátor* felhasználásával összehasonlítja a beírt string jelszóval. Ha a visszatérési érték igaz, akkor a felhasználó belép a profilba, ha a segélykérő kombináció akkor a visszatérési érték hamis és a jelszó fájlba íródik, ha csak simán hibás a jelszó a visszatérési érték hamis és tovább kell próbálkozni.

Exportálás, Importálás

Az importálás egy vezérlőfüggvény dolga lesz, az exportálás pedig az AllUsersen keresztül az *export()* függvények segítségével történik.

1. fejezet

Hierarchikus mutató

1.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

2. fejezet

Osztálymutató

2.1. Osztálylista

Az összes osztály, struktúra, unió és interfész listája rövid leírásokkal:

3. fejezet

Fájlmutató

3.1. Fájllista

Az összes fájl listája rövid leírásokkal:

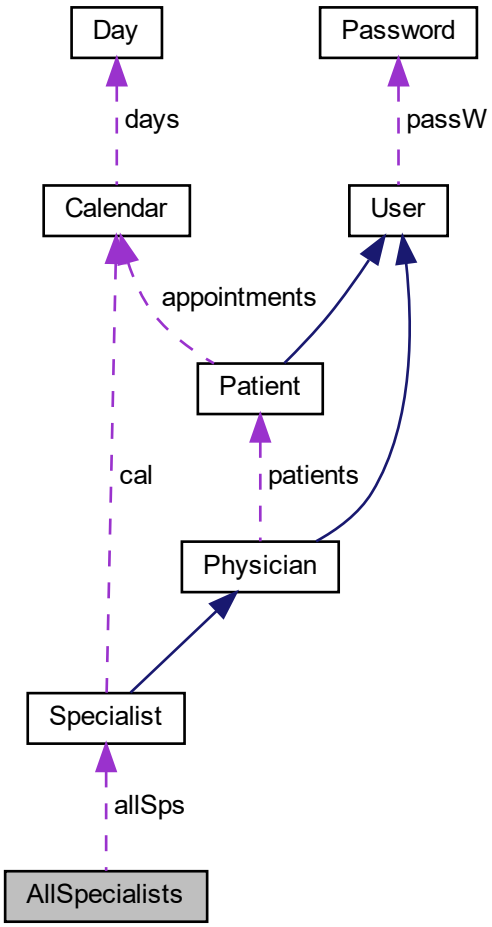
4. fejezet

Osztályok dokumentációja

4.1. AllSpecialists osztályreferencia

`#include <AllSpecialists.h>`

Az AllSpecialists osztály együttműködési diagramja:



Publikus tagfüggvények

- AllSpecialists ()
- void addSpecialist (Specialist ∗sp)
- Specialist ∗∗ filterType (int t)
- ∼AllSpecialists ()

Privát attribútumok

- Specialist ∗∗ allSps
- size_t num

4.1.1. Részletes leírás

A szakorvosokat tároló osztály.

4.1.2. Konstruktorok és destruktorok dokumentációja

4.1.2.1. AllSpecialists()

```
AllSpecialists::AllSpecialists ( )  [inline]
```

Konstruktor. Paraméter nélküli.

4.1.2.2. ∼AllSpecialists()

```
AllSpecialists::∼AllSpecialists ( )  [inline]
```

Destruktor.Felszabadítja a dinamikusan foglalt allSps tömböt.

4.1.3. Tagfüggvények dokumentációja

4.1.3.1. addSpecialist()

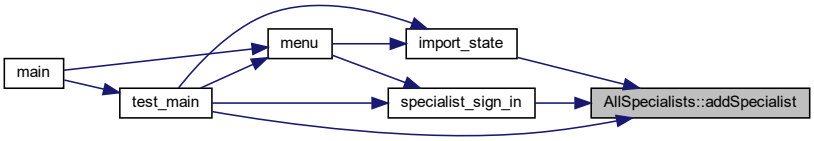
```
void AllSpecialists::addSpecialist (
            Specialist * sp )
```

A paraméterként kapott szakorvosra mutató pointert hozzáadja a Specialist∗-okat tároló allSps tömbhöz.

Paraméterek

sp	A tömbhöz hozzáadandó szakorvosra mutató pointer.

A függvény hívó gráfja:



4.1.3.2. filterType()

```
Specialist ** AllSpecialists::filterType (
            int t )
```

A paraméterként megadott típusú szakorvosok tömbjével tér vissza. A szakorvosokat a Specialist::getType függvénye segítségével válogatja ki. A dinamikusan foglalt tömb végjeles: utolsó eleme mindig nullpointer.

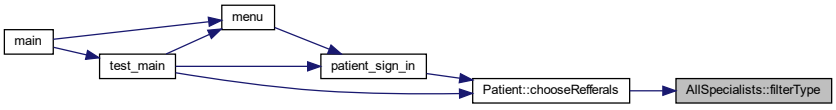
Paraméterek

t	milyen típusú szakorvosokkal térjen vissza

Visszatérési érték

az adott típusú szakorvosok végjeles tömbje.

A függvény hívó gráfja:



4.1.4. Adattagok dokumentációja

4.1.4.1. allSps

```
Specialist** AllSpecialists::allSps  [private]
```

Szakorvosok mutatóit tároló dinamikusan foglalt tömb.

4.1.4.2. num

```
size_t AllSpecialists::num  [private]
```

Tárolt szakorvosok száma.

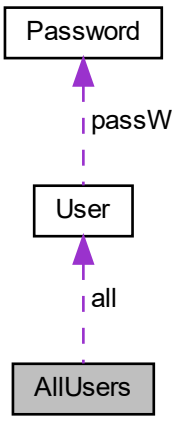
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- C:/Users/Lili/Desktop/Prog2/nhf/AllSpecialists.h
- C:/Users/Lili/Desktop/Prog2/nhf/AllSpecialists.cpp

4.2. AllUsers osztályreferencia

```
#include <AllUsers.h>
```

Az AllUsers osztály együttműködési diagramja:



Publikus tagfüggvények

- AllUsers ()
- bool contains (const std::string &name)
- bool containsID (const std::string &ID)
- void addUser (User *l)
- User * getUser (const std::string &name)
- std::string exporter ()
- int getNum () const
- ~AllUsers ()

Privát attribútumok

- User ** all
- size_t num

4.2.1. Részletes leírás

A felhasználókra mutatókat tároló heterogén kollekció.

4.2.2. Konstruktorok és destruktorok dokumentációja

4.2.2.1. AllUsers()

```
AllUsers::AllUsers ( )  [inline]
```

Paraméter nélküli konstruktor

4.2.2.2. ~AllUsers()

```
AllUsers::~AllUsers ( )  [inline]
```

Destruktor, nem csak a dinamikusan foglalt tömböt szabadítja fel, hanem a benne tárolt pointereket is.

4.2.3. Tagfüggvények dokumentációja

4.2.3.1. addUser()

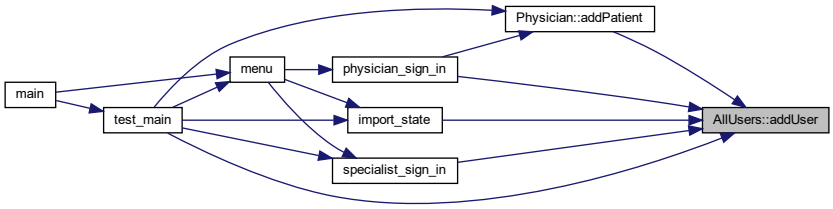
```
void AllUsers::addUser (
            User * l )
```

A paraméterként átvett felhasználót hozzáadja a felhasználók tömbjéhez.

Paraméterek

l	a hozzáadandó felhasználó.

A függvény hívó gráfja:



4.2.3.2. contains()

```
bool AllUsers::contains (
            const std::string & name )
```

A paraméterként megadott név alapján ellenőrzi, hogy létezik-e ilyen nevű felhasználó.

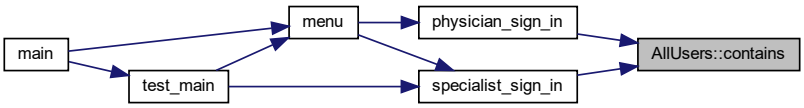
Paraméterek

name	a keresett személy neve.

Visszatérési érték

létezik-e ilyen nevű felhasználó.

A függvény hívó gráfja:



4.2.3.3. containsID()

```
bool AllUsers::containsID (
            const std::string & ID )
```

A paraméterként megadott TAJ szám alapján ellenőrzi, hogy létezik-e ilyen TAJ számú felhasználó.

Paraméterek

ID	a keresett személy TAJ száma.

Visszatérési érték

létezik-e ilyen TAJ számú/jelszavú felhasználó.

A paraméterként megadott TAJ szám alapján ellenőrzi, hogy létezik-e ilyen TAJ számú/jelszavú felhasználó.

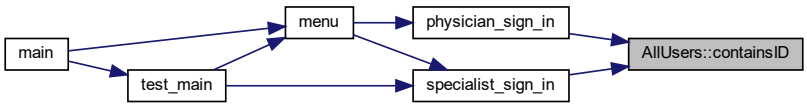
Paraméterek

ID	a keresett személy azonosítója.

Visszatérési érték

létezik-e ilyen jelszavú/TAJ számú felhasználó.

A függvény hívó gráfja:



4.2.3.4. exporter()

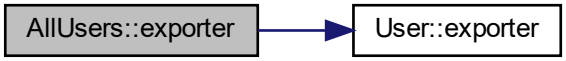
```
std::string AllUsers::exporter ( )
```

File-ba kiíráshoz előállít egy stringet a tartalmából, melyet később majd vissza lehet olvasni.

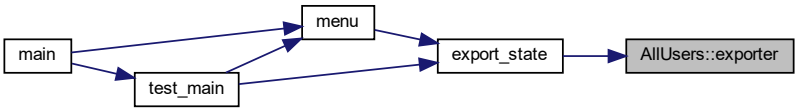
Visszatérési érték

stringbe írt tartalom.

A függvény hívási gráfja:



A függvény hívó gráfja:



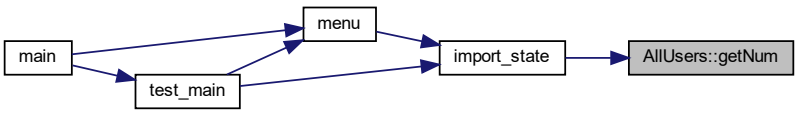
4.2.3.5. getNum()

```
int AllUsers::getNum ( ) const
```

Visszatérési érték

a tárolt felhasználók száma.

A függvény hívó gráfja:



4.2.3.6. getUser()

```
User * AllUsers::getUser (
            const std::string & name )
```

A paraméterként kapott felhasználót megkeresi.

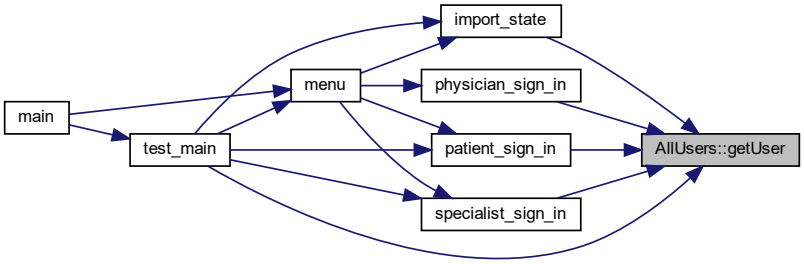
Paraméterek

name	a keresett profil neve.

Visszatérési érték

kapott névhez tartozó profilra mutató pointer vagy nullptr.

A függvény hívó gráfja:



4.2.4. Adattagok dokumentációja

4.2.4.1. all

```
User** AllUsers::all  [private]
```

A felhasználók mutatóit tároló, dinamikusan lefoglalt heterogén kollekció.

4.2.4.2. num

```
size_t AllUsers::num  [private]
```

A tárolt felhasználók száma

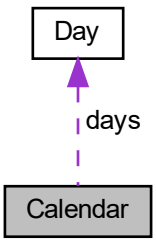
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- C:/Users/Lili/Desktop/Prog2/nhf/AllUsers.h
- C:/Users/Lili/Desktop/Prog2/nhf/AllUsers.cpp

4.3. Calendar osztályreferencia

```
#include <Calendar.h>
```

A Calendar osztály együttműködési diagramja:



Publikus tagfüggvények

- Calendar ()
- void print (std::ostream &os)
- void addEvent (int day, int section, const std::string &who)
- std::string exporter ()
- ~Calendar ()=default

Privát attribútumok

- Day days [31]

4.3.1. Részletes leírás

Naptár osztály egy hónap bejegyzett eseményei tárolására.

4.3.2. Konstruktorok és destruktorok dokumentációja

4.3.2.1. Calendar()

```
Calendar::Calendar ( )  [inline]
```

Paraméter nélküli konstruktor, mely nullákkal tölti fel a kalendárt.

4.3.2.2. ~Calendar()

```
Calendar::~Calendar ( )  [default]
```

Destruktor

4.3.3. Tagfüggvények dokumentációja

4.3.3.1. addEvent()

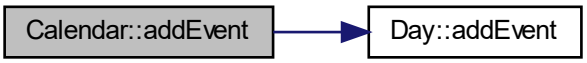
```
void Calendar::addEvent (
            int day,
            int section,
            const std::string & who )
```

Időpont hozzáadása a naptárhoz.

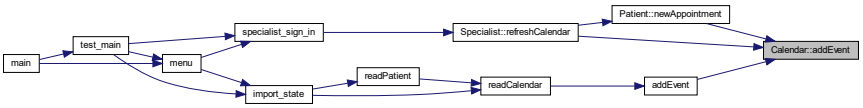
Paraméterek

day	a nap száma, 1 és 31 közötti egész szám.
section	napszak, 1 és 4 közötti egész szám.
who	a szakorvos neve.

A függvény hívási gráfja:



A függvény hívó gráfja:



4.3.3.2. exporter()

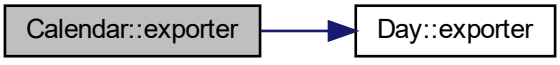
```
std::string Calendar::exporter ( )
```

Exportálásért felelős függvény.

Visszatérési érték

a file-ba kiírandó szöveg tartalom.

A függvény hívási gráfja:



A függvény hívó gráfja:

4.3.3.3. print()

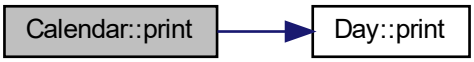
```
void Calendar::print (
            std::ostream & os )
```

/Kiírja a paraméterként kapott ostreamre a naptárat úgy, hogy a naptárban szereplő 31 napra meghívja a Day osztály print függvényét.

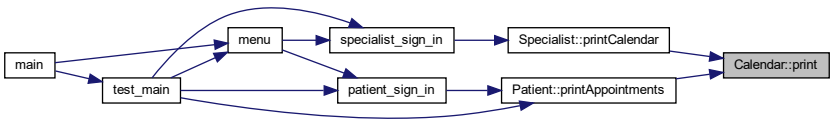
Paraméterek

os	paraméterként kapott ostream.

A függvény hívási gráfja:



A függvény hívó gráfja:



4.3.4. Adattagok dokumentációja

4.3.4.1. days

```
Day Calendar::days[31]  [private]
```

A napok tömbje.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- C:/Users/Lili/Desktop/Prog2/nhf/Calendar.h
- C:/Users/Lili/Desktop/Prog2/nhf/Calendar.cpp

4.4. Day osztályreferencia

```
#include <Day.h>
```

Publikus tagfüggvények

- Day ()
- void addEvent (int section, const std::string &who)
- std::string exporter ()
- void print (std::ostream &os)
- ~Day ()=default

Privát attribútumok

- std::string sections [4]

4.4.1. Részletes leírás

Egy naptárban szereplő napot jelképező osztály.

4.4.2. Konstruktorok és destruktorok dokumentációja

4.4.2.1. Day()

```
Day::Day ( )  [inline]
```

Konstruktor.

4.4.2.2. ~Day()

```
Day::~Day ( )  [default]
```

Destruktor.

4.4.3. Tagfüggvények dokumentációja

4.4.3.1. addEvent()

```
void Day::addEvent (
            int i,
            const std::string & who )
```

A nap egy kiválasztott section-jébe beleírja a paraméterként kapott stringet, amennyiben a sectionben 0 szerepel (nem foglal).

Paraméterek

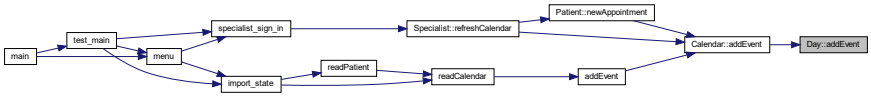
section	a megadott section (1-4). •
who	a megadott string.

A nap egy kiválasztott section-jébe beleírja a paraméterként kapott stringet, amennyiben a sectionben 0 szerepel (nem foglal).

Paraméterek

section	a megadott section (1-4).
who	a megadott string.

A függvény hívó gráfja:



4.4.3.2. exporter()

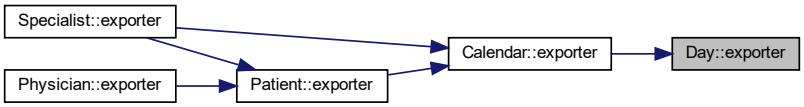
```
std::string Day::exporter ( )
```

Exportálásért felelős függvény.

Visszatérési érték

a file-ba kiírandó szöveg tartalom.

A függvény hívó gráfja:



4.4.3.3. print()

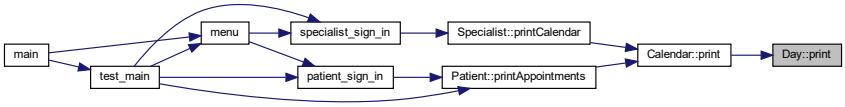
```
void Day::print (
            std::ostream & os )
```

Kiírja a napot section-önként, tabulátorokkal elválasztva a megadott outputstreamre.

Paraméterek

os	a megadott outputstream.

A függvény hívó gráfja:



4.4.4. Adattagok dokumentációja

4.4.4.1. sections

```
std::string Day::sections[4]  [private]
```

A napon belüli időszakok tömbje.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- C:/Users/Lili/Desktop/Prog2/nhf/Day.h
- C:/Users/Lili/Desktop/Prog2/nhf/Day.cpp

4.5. Password osztályreferencia

`#include <Password.h>`

Publikus tagfüggvények

- Password ()
- std::string getter ()
- bool setPassword (const std::string &p)
- bool operator== (const std::string &p)
- std::string exporter ()
- ~Password ()=default

Privát attribútumok

- std::string pass

4.5.1. Részletes leírás

Jelszót kezelő osztály.

4.5.2. Konstruktorok és destruktorok dokumentációja

4.5.2.1. Password()

`Password::Password ( )`

A Password osztály konstruktora, mely egy üres sztringet hoz létre.

4.5.2.2. ~Password()

`Password::~Password ( )  [default]`

Destruktor

4.5.3. Tagfüggvények dokumentációja

4.5.3.1. exporter()

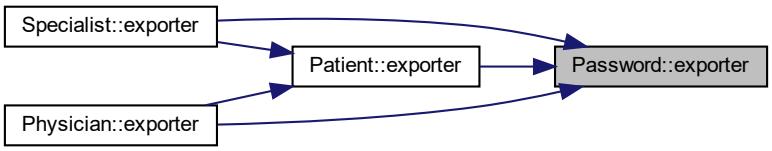
```
std::string Password::exporter ( )
```

Exportálásért felelős függvény.

Visszatérési érték

a file-ba kiírandó szöveg tartalom.

A függvény hívó gráfja:



4.5.3.2. getter()

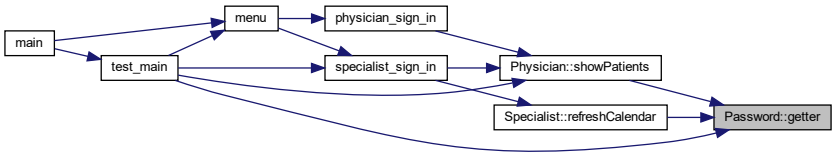
```
std::string Password::getter ( )  [inline]
```

Magával a string formájú jelszóval tér vissza.

Visszatérési érték

a jelszó (pass).

A függvény hívó gráfja:



4.5.3.3. operator==()

```
bool Password::operator== (
            const std::string & p )
```

Az == operátor felüldefiniálása, hogy összehasonlítható legyen a felhasználó által beírt string a profilhoz tartozó jelszóval. Visszatérési értéke true ha megegyezik, false ha nem. Ha a felhasználó átal megadott string "111111", akkor a program fájlba írja az adott profilhoz tartoó jelszót.

Paraméterek

p	a felhasználó által megadott jelszó (string).

Visszatérési érték

true, ha jót adott meg, false, ha nem.

Az == operátor felüldefiniálása, hogy összehasonlítható legyen a felhasználó által beírt string a profilhoz tartozó jelszóval. Visszatérési értéke true ha megegyezik, false ha nem. Ha a felhasználó átal megadott string "111111", akkor a program fájlba írja az adott profilhoz tartozó jelszót.

Paraméterek

p	a felhasználó által megadott jelszó (string).

Visszatérési érték

true, ha jót adott meg, false, ha nem.

4.5.3.4. setPassword()

```
bool Password::setPassword (
            const std::string & p )
```

Amennyiben a felhasználó érvényes jelszót adott meg, az lesz pass értéke és a visszatérési érték true. Ha nem, false és a jelszó nem változik.

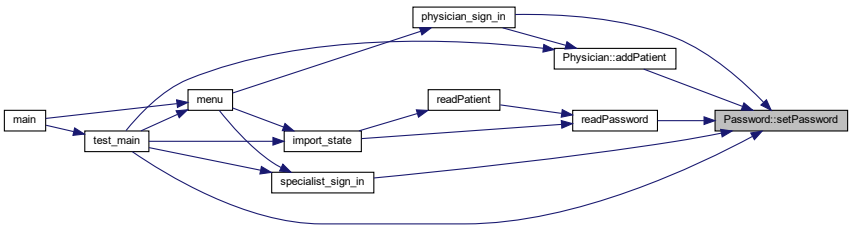
Paraméterek

p	a felhasználó által jelszónak megadott string.

Visszatérési érték

true, ha jót adott meg, false, ha nem.

A függvény hívó gráfja:



4.5.4. Adattagok dokumentációja

4.5.4.1. pass

```
std::string Password::pass  [private]
```

A jelszó maga.

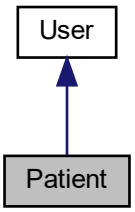
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- C:/Users/Lili/Desktop/Prog2/nhf/Password.h
- C:/Users/Lili/Desktop/Prog2/nhf/Password.cpp

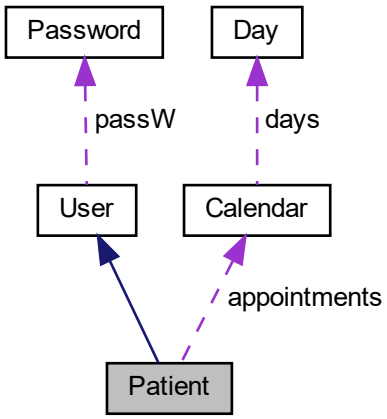
4.6. Patient osztályreferencia

```
#include <Patient.h>
```

A Patient osztály származási diagramja:



A Patient osztály együttműködési diagramja:



Publikus tagfüggvények

- Patient (const std::string &Name, Password pass, Calendar a)
- void gotVaccinated ()
- void newRefferal (int n)
- void newPrescription (const std::string &p)
- void newInformation (const std::string &I)
- void getPrescription ()
- void getVaccinationInformation ()
- void chooseRefferals (AllSpecialists &list, std::ostream &os)
- void printAppointments (std::ostream &os)
- void newAppointment (int day, int sec, const std::string &who)
- std::string exporter () override
- ~Patient () override

Privát attribútumok

- bool vaccination
- int * refferals
- size_t numOfRefferals
- std::string * prescriptions
- size_t numOfPrescriptions
- std::string * additionalInformation
- size_t numOfAdditional
- Calendar appointments

További örökölt tagok

4.6.1. Részletes leírás

Pácienst megvalósító osztály.

4.6.2. Konstruktorok és destruktorok dokumentációja

4.6.2.1. Patient()

```
Patient::Patient (
            const std::string & Name,
            Password pass,
            Calendar a )  [inline]
```

Konstruktor.

Paraméterek

Name	string név
pass	Password jelszó.
a	Calendar kalendár.

4.6.2.2. ~Patient()

```
Patient::~Patient ( )  [inline], [override]
```

Destruktor. Megszünteti a dinamikusan létrehozott attribútumait.

4.6.3. Tagfüggvények dokumentációja

4.6.3.1. chooseRefferals()

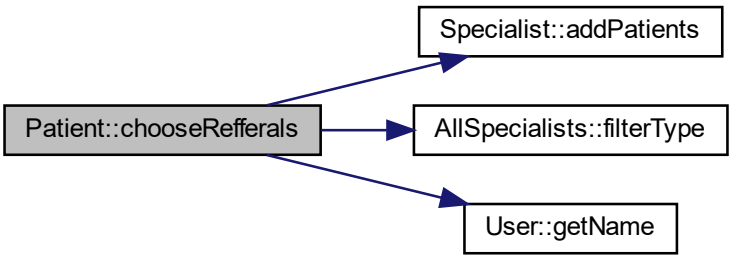
```
void Patient::chooseRefferals (
            AllSpecialists & list,
            std::ostream & os )
```

A függvény kiírja a páciens háziorvosoktól kapott beutalóit. Minden beutalóhoz kiírja a paraméterként kapott ostreamre az adott szakterületen dolgozó orvosok neveit.

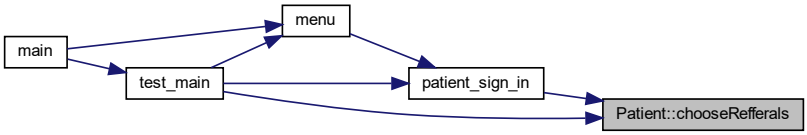
Paraméterek

list	Az összes szakorvos listája, melyből a függvény a filterType függvény segítségével válogat.
os	ide ír a függvény .

A függvény hívási gráfja:



A függvény hívó gráfja:



4.6.3.2. exporter()

```
std::string Patient::exporter ( )  [override], [virtual]
```

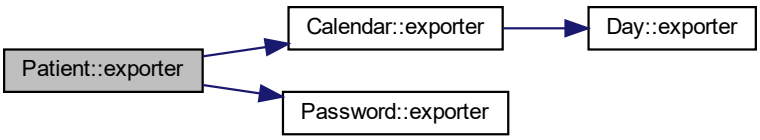
Exportálásért felelős függvény.

Visszatérési érték

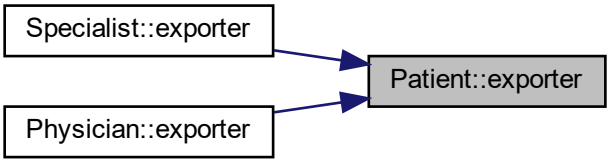
a file-ba kiírandó szöveg tartalom.

Megvalósítja a következőket: User.

A függvény hívási gráfja:



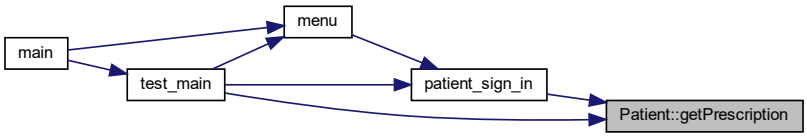
A függvény hívó gráfja:



4.6.3.3. getPrescription()

```
void Patient::getPrescription ( )
```

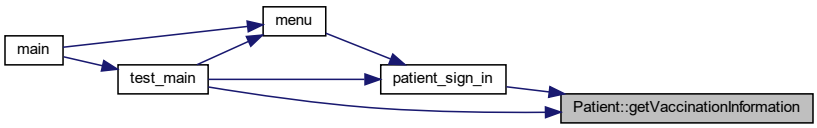
A beteg ezzel megtekinti, hogy van-e számára felírt recept. Amennyiben van, azt a függvény file-ba exportálja. A függvény hívó gráfja:



4.6.3.4. getVaccinationInformation()

```
void Patient::getVaccinationInformation ( )
```

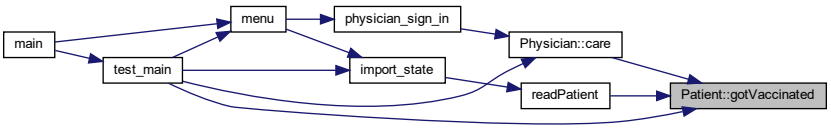
A beteg megtekintheti, hogy megkapta-e az oltást. Eportálhatja az oltási igazolványát file-ba. A függvény hívó gráfja:



4.6.3.5. gotVaccinated()

```
void Patient::gotVaccinated ( )
```

Bejelöli, hogy egy beteg megkapta az oltást. A függvény hívó gráfja:



4.6.3.6. newAppointment()

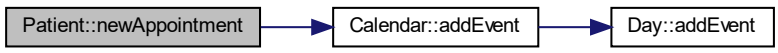
```
void Patient::newAppointment (
            int day,
            int sec,
            const std::string & who )
```

Új szakorvosi időpont adható a beteg kalendárjába. Meghívja a kalendár addEvent függvényét.

Paraméterek

day	melyik napra (1-31).
sec	a nap melyik szekciójába (1-4).
who	string, a szakorvos neve.

A függvény hívási gráfja:



A függvény hívó gráfja:

4.6.3.7. newInformation()

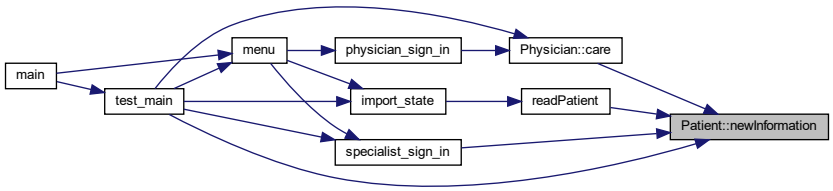
```
void Patient::newInformation (
            const std::string & I )
```

Megjegyzés adható vele a beteg profiljához.

Paraméterek

I	a hozzáadandó információ.

A függvény hívó gráfja:



4.6.3.8. newPrescription()

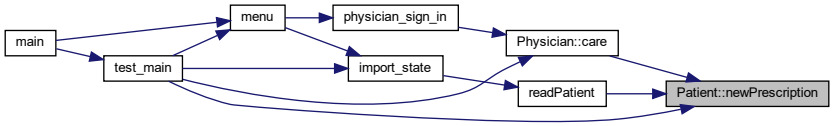
```
void Patient::newPrescription (
            const std::string & p )
```

Recept írható fel vele.

Paraméterek

p	a recept szövege.

A függvény hívó gráfja:



4.6.3.9. newRefferal()

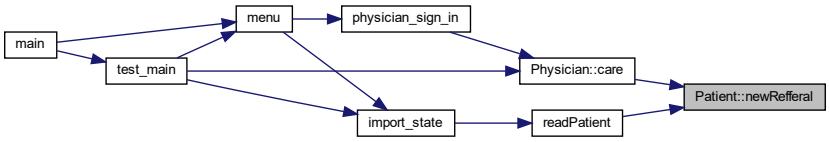
```
void Patient::newRefferal (
            int n )
```

Szakorvoshoz beutalót adó függvény. A beutalók tömbjéhez hozzáadódik.

Paraméterek

n	a célzott szakorvos típusa.

A függvény hívó gráfja:



4.6.3.10. printAppointments()

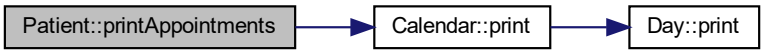
```
void Patient::printAppointments (
            std::ostream & os )
```

Kiírja a beteg kalendárját (meghívja a kalendár print függvényét).

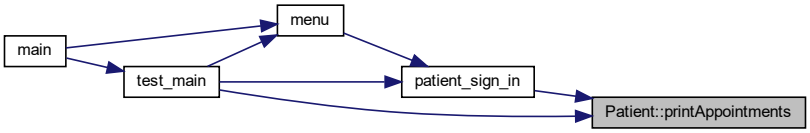
Paraméterek

os	ostream, ide ír a függvény.

A függvény hívási gráfja:



A függvény hívó gráfja:



4.6.4. Adattagok dokumentációja

4.6.4.1. additionalInformation

```
std::string* Patient::additionalInformation  [private]
```

dinamikusan tárolt egyéb információk a pácienssel kapcsolatban.

4.6.4.2. appointments

```
Calendar Patient::appointments  [private]
```

naptár, melyben a lefoglalt időpontok szerepelnek.

4.6.4.3. numOfAdditional

```
size_t Patient::numOfAdditional  [private]
```

dinamikusan tárolt egyéb információk darabszáma.

4.6.4.4. numOfPrescriptions

```
size_t Patient::numOfPrescriptions  [private]
```

tárolt receptek darabszáma

4.6.4.5. numOfRefferals

```
size_t Patient::numOfRefferals  [private]
```

tárolt beutalók darabszáma.

4.6.4.6. prescriptions

```
std::string* Patient::prescriptions  [private]
```

dinamikusan tárolt receptek.

4.6.4.7. refferals

```
int* Patient::refferals  [private]
```

beutalókat tároló dinamikus tömb.

4.6.4.8. vaccination

```
bool Patient::vaccination  [private]
```

oltottság.

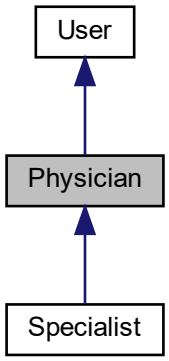
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- C:/Users/Lili/Desktop/Prog2/nhf/Patient.h
- C:/Users/Lili/Desktop/Prog2/nhf/Patient.cpp

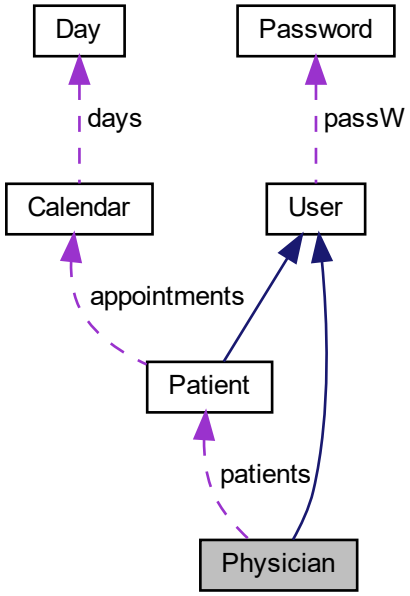
4.7. Physician osztályreferencia

```
#include <Physician.h>
```

A Physician osztály származási diagramja:



A Physician osztály együttműködési diagramja:



Publikus tagfüggvények

- Physician (const std::string &n, Password p)
- std::string exporter () override
- void addPatient (const std::string &taj, const std::string &name, AllUsers &users)
- void care (const std::string &taj, std::ostream &os)
- void showPatients (std::ostream &os) const
- void addExistingPatient (Patient ∗pPatient)
- ∼Physician ()

Védett attribútumok

- int numOfPatients
- Patient ∗∗ patients

4.7.1. Részletes leírás

A háziorvost megvalósító osztály.

4.7.2. Konstruktorok és destruktorok dokumentációja

4.7.2.1. Physician()

```
Physician::Physician (
            const std::string & n,
            Password p )  [inline]
```

Konstruktor.

Paraméterek

n	string név.
p	Password jelszó.

4.7.2.2. ~Physician()

```
Physician::~Physician ( )  [inline]
```

Destruktor

4.7.3. Tagfüggvények dokumentációja

4.7.3.1. addExistingPatient()

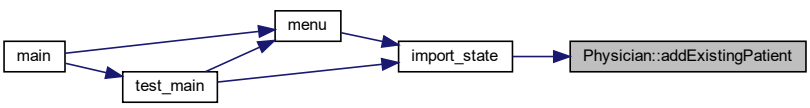
```
void Physician::addExistingPatient (
            Patient * pPatient )
```

Már létező páciens hozzáadása a háziorvoshoz. Az állapot file-ból történő visszaállításánál van rá szükség.

Paraméterek

pPatient	az eltárolandó páciens.

A függvény hívó gráfja:



4.7.3.2. addPatient()

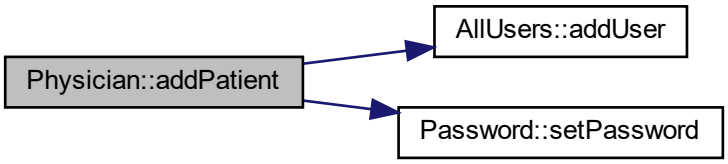
```
void Physician::addPatient (
            const std::string & taj,
            const std::string & name,
            AllUsers & users )
```

Létrehoz egy beteg-profilt, azt az összes profil és a saját betegei közé adja.

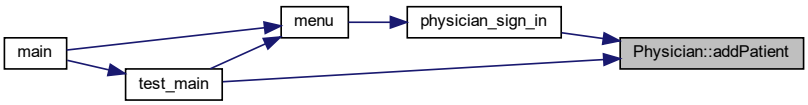
Paraméterek

taj	a beteg (leendő) jelszava.
name	a beteg (leendő) neve.
users	az összes profilt tároló heterogén kollekció, ebbe is bele kell tegye a függvény az új Patientet.

A függvény hívási gráfja:



A függvény hívó gráfja:



4.7.3.3. care()

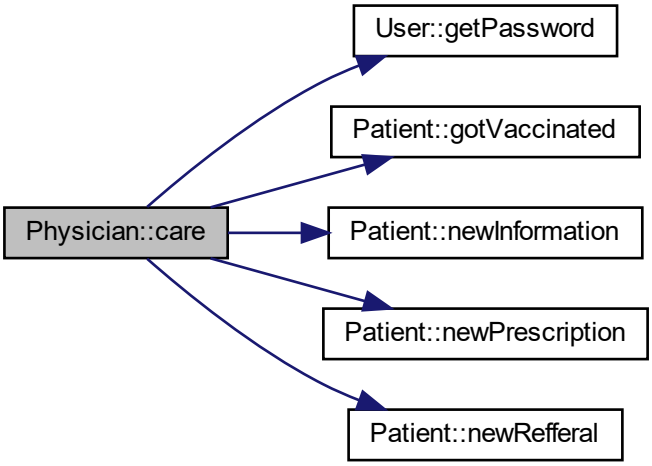
```
void Physician::care (
            const std::string & taj,
            std::ostream & os )
```

A paraméterként kapott string segítségével kiválasztja az egyik betegét, különböző műveleteket hajthat végre rajta (a lehetőségek a paraméterként lapott ostreamre íródhatnak).

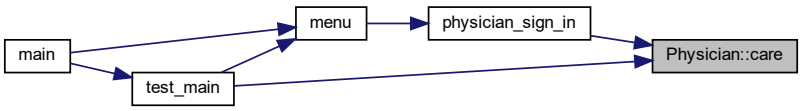
Paraméterek

taj	annak a felhasználónak a TAJszáma/jelszava, akivel a műveleteket végre lehet hajtani.
os	ide íródnak a függvény által kiírt dolgok.

A függvény hívási gráfja:



A függvény hívó gráfja:



4.7.3.4. exporter()

```
std::string Physician::exporter ( )  [override], [virtual]
```

Exportálásért felelős függvény.

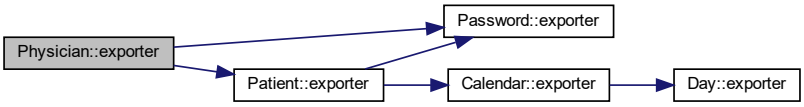
Visszatérési érték

a file-ba kiírandó szöveg tartalom.

Megvalósítja a következőket: User.

Újraimplementáló leszármazottak: Specialist.

A függvény hívási gráfja:



4.7.3.5. showPatients()

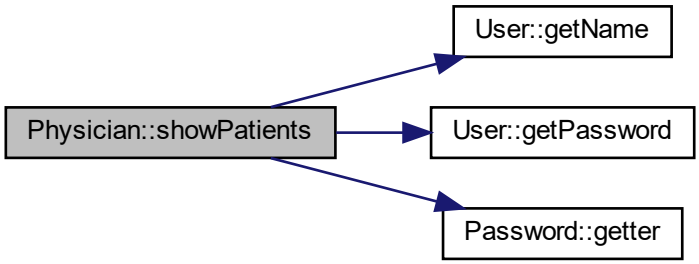
```
void Physician::showPatients (
            std::ostream & os ) const
```

Kiírja az orvos betegeit a paraméterként kapott ostreamre.

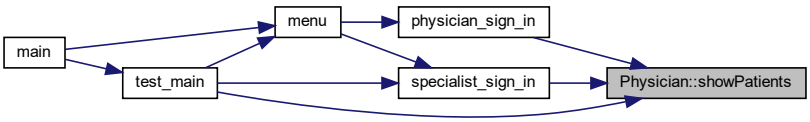
Paraméterek

os	ide ír a függvény.

A függvény hívási gráfja:



A függvény hívó gráfja:



4.7.4. Adattagok dokumentációja

4.7.4.1. numOfPatients

```
int Physician::numOfPatients  [protected]
```

páciensek darabszáma.

4.7.4.2. patients

```
Patient** Physician::patients  [protected]
```

páciensekre mutatókat tároló dinamikus tömb.

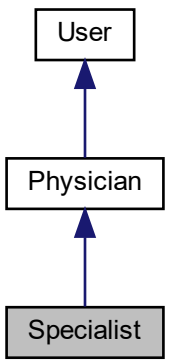
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- C:/Users/Lili/Desktop/Prog2/nhf/Physician.h
- C:/Users/Lili/Desktop/Prog2/nhf/Physician.cpp

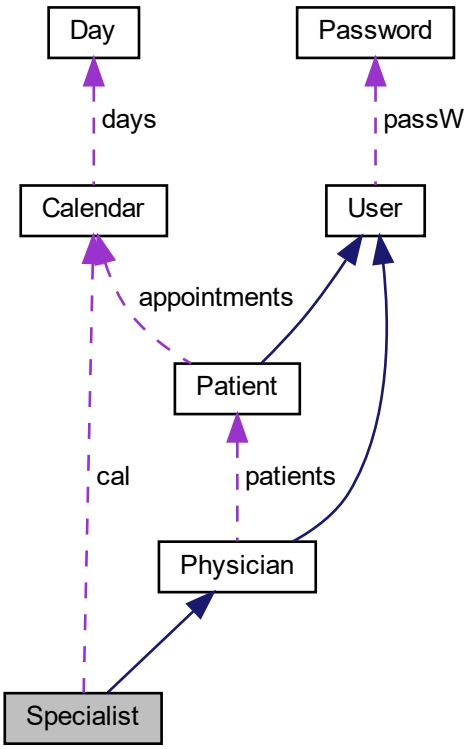
4.8. Specialist osztályreferencia

```
#include <Specialist.h>
```

A Specialist osztály származási diagramja:



A Specialist osztály együttműködési diagramja:



Publikus tagfüggvények

- Specialist (const std::string &n, Password p, Calendar a, int t)
- void addPatients (Patient ∗p)
- Patient ∗ getPatient (const std::string &taj)
- void refreshCalendar (int day, int section, Patient ∗who)
- std::string exporter ()
- void printCalendar (std::ostream &os)
- int getType () const

Privát attribútumok

- int type
- Calendar cal

További örökölt tagok

4.8.1. Részletes leírás

Szakorvost megvalósító osztály.

4.8.2. Konstruktorok és destruktorok dokumentációja

4.8.2.1. Specialist()

```
Specialist::Specialist (
            const std::string & n,
            Password p,
            Calendar a,
            int t )  [inline]
```

Konstruktor

Paraméterek

n	név (name)
p	jelszó (password)
a	kalendár (appointments)
t	típus (type)

4.8.3. Tagfüggvények dokumentációja

4.8.3.1. addPatients()

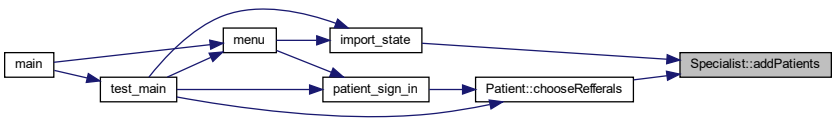
```
void Specialist::addPatients (
            Patient * p )
```

A szakorvos páciensei közé adódik egy beteg ha időpontot kér tőle. A szakorvos egy dinamikus tömbben tárolja a pácienseire mutató pointereket. Ehhez a tömbhöz adódik hozzá a paraméterként kapott pointer. Maga a hozzáadás egy új, kibővített tömb létrehozásából, az előző tömb törléséből áll.

Paraméterek

p	a hozzáadandó páciensre mutató pointer.

A függvény hívó gráfja:



4.8.3.2. exporter()

```
std::string Specialist::exporter ( )  [virtual]
```

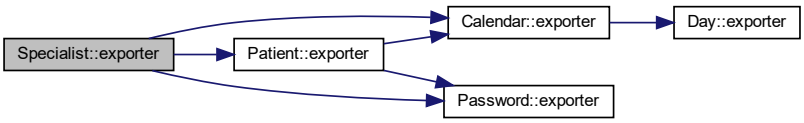
Exportálásért felelős függvény.

Visszatérési érték

a file-ba kiírandó szöveg tartalom.

Újraimplementált ősök: Physician.

A függvény hívási gráfja:



4.8.3.3. getPatient()

```
Patient * Specialist::getPatient (
            const std::string & taj )
```

TAJ szám alapján kiválasztott beteggel (vagyis a rá mutató pointerrel) tér vissza. A Specialist patient pointereket tároló patients tömbjén végigmegy, ha az adott páciens tajszáma (=jelszava) egyezik a megadottal visszatér egy rá mutató pointerrel. Amennyiben nincs egyezés, kivételt dob.

Paraméterek

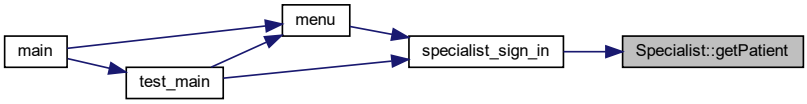
taj	a keresett páciens tajszáma stringként.

Visszatérési érték

a keresett páciensre mutató pointer.

A függvény hívási gráfja:

A függvény hívó gráfja:



4.8.3.4. getType()

```
int Specialist::getType ( ) const
```

A szakorvos típusának lekérdezésére szolgál.

Visszatérési érték

szakorvos típusa.

4.8.3.5. printCalendar()

```
void Specialist::printCalendar (
            std::ostream & os )
```

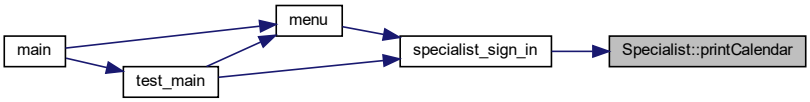
A Specialist kalendárjának kiírására szolgáló függvény. Meghívja a Calendar print függvényét.

Paraméterek

os	ostream&, ahová a függvény kiírja a kalendárt.

A függvény hívási gráfja:

A függvény hívó gráfja:



4.8.3.6. refreshCalendar()

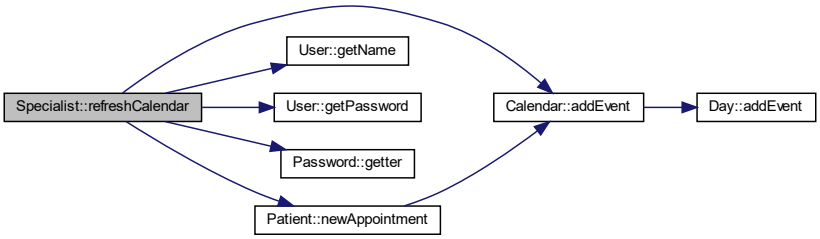
```
void Specialist::refreshCalendar (
            int day,
            int section,
            Patient * who )
```

A felhasználó által kiválasztott, paraméterként átadott időpontra beírja a beteg tajszámát. Ugyanakkor beírja a paraméterként megadott páciens kalendárjába ugyanarra az időpontra a szakorvos nevét. Meghívja a Specialist kalendárjára az addEvent függvényt, a páciensre a newAppointment függvényt.

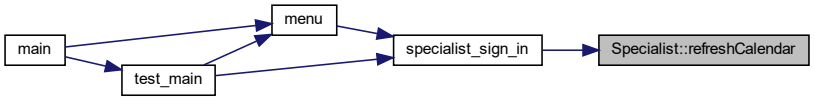
Paraméterek

day	int, a megadott nap száma (1-31).
section	int, a napon belül megadott szekció száma (1-4).
who	a beírandó páciensre mutató pointer.

A függvény hívási gráfja:



A függvény hívó gráfja:



4.8.4. Adattagok dokumentációja

4.8.4.1. cal

```
Calendar Specialist::cal  [private]
```

a szakorvos naptárja, melyben a páciensek foglalt időpontjai találhatók.

4.8.4.2. type

```
int Specialist::type  [private]
```

a szakorvos típusát jelölő egész szám.

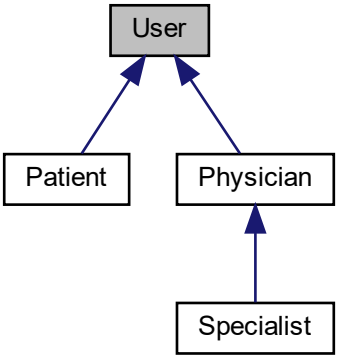
Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- C:/Users/Lili/Desktop/Prog2/nhf/Specialist.h
- C:/Users/Lili/Desktop/Prog2/nhf/Specialist.cpp

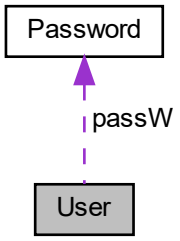
4.9. User osztályreferencia

`#include <User.h>`

Az User osztály származási diagramja:



Az User osztály együttműködési diagramja:



Publikus tagfüggvények

- User (const std::string &n, Password p)
- Password getPassword ()
- virtual std::string exporter ()=0
- std::string getName ()
- virtual ~User ()

Védett attribútumok

- std::string name
- Password passW

4.9.1. Részletes leírás

Absztrakt alaposztály.

4.9.2. Konstruktorok és destruktorok dokumentációja

4.9.2.1. User()

```
User::User (
            const std::string & n,
            Password p )  [inline]
```

Konstruktor.

Paraméterek

n	felhasználó neve.
p	felhasználó jelszava.

4.9.2.2. ~User()

```
virtual User::~User ( )  [inline], [virtual]
```

Destruktor

4.9.3. Tagfüggvények dokumentációja

4.9.3.1. exporter()

```
virtual std::string User::exporter ( )  [pure virtual]
```

Exportálásért felelős függvény.

Megvalósítják a következők: Specialist, Patient és Physician.

A függvény hívó gráfja:

4.9.3.2. getName()

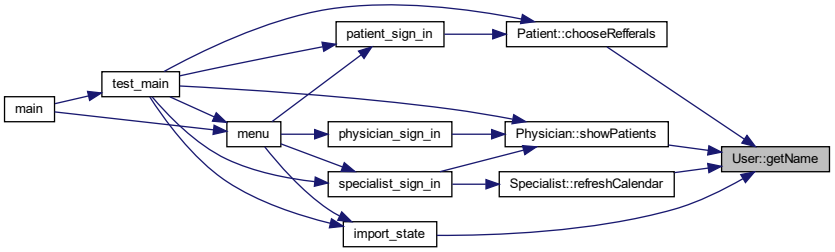
```
std::string User::getName ( )
```

Getter a névnek.

Visszatérési érték

a profilhoz tartozó név.

A függvény hívó gráfja:



4.9.3.3. getPassword()

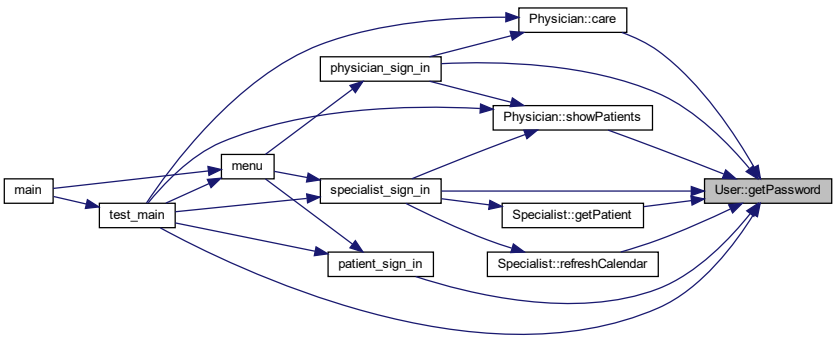
```
Password User::getPassword ( )
```

Getter a jelszónak.

Visszatérési érték

a felhasználóhoz tartozó jelszóval

A függvény hívó gráfja:



4.9.4. Adattagok dokumentációja

4.9.4.1. name

```
std::string User::name  [protected]
```

a felhasználó neve.

4.9.4.2. passW

```
Password User::passW  [protected]
```

a felhasználó jelszava.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- C:/Users/Lili/Desktop/Prog2/nhf/User.h
- C:/Users/Lili/Desktop/Prog2/nhf/User.cpp

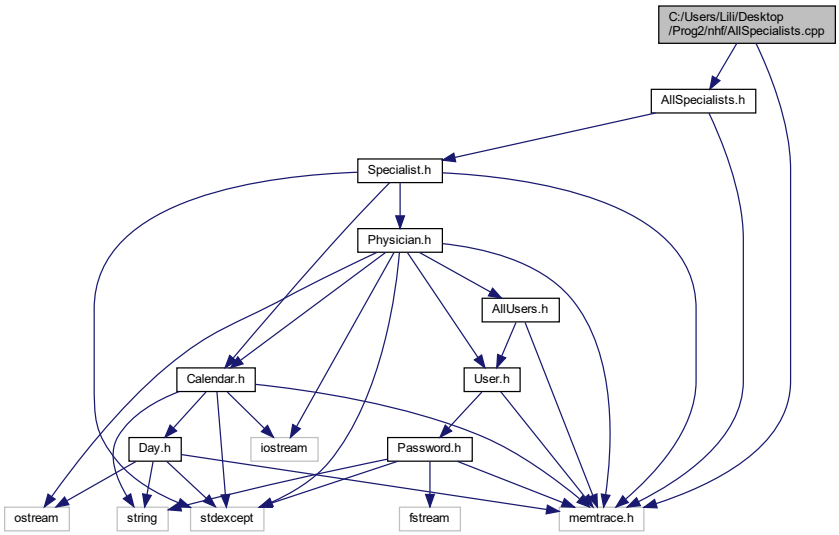
5. fejezet

Fájlok dokumentációja

5.1. C:/Users/Lili/Desktop/Prog2/nhf/AllSpecialists.cpp fájlreferencia

```
#include "memtrace.h"
#include "AllSpecialists.h"
```

Az AllSpecialists.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

5.1.1. Makródefiníciók dokumentációja

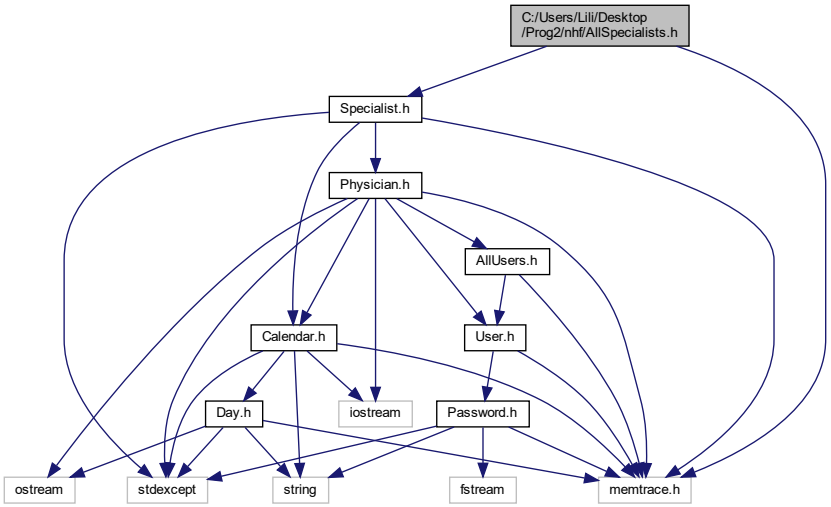
5.1.1.1. MEMTRACE

```
#define MEMTRACE
```

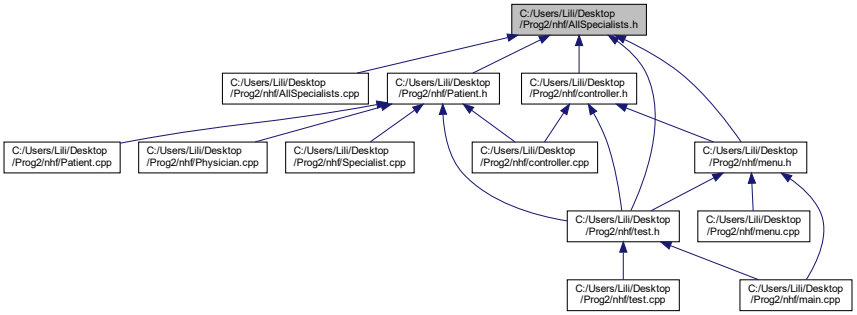
5.2. C:/Users/Lili/Desktop/Prog2/nhf/AllSpecialists.h fájlreferencia

```
#include "memtrace.h"
#include "Specialist.h"
```

Az AllSpecialists.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class AllSpecialists

Makródefiníciók

- #define MEMTRACE

5.2.1. Makródefiníciók dokumentációja

5.2.1.1. MEMTRACE

```
#define MEMTRACE
```

5.3. C:/Users/Lili/Desktop/Prog2/nhf/AllSpecialists.h

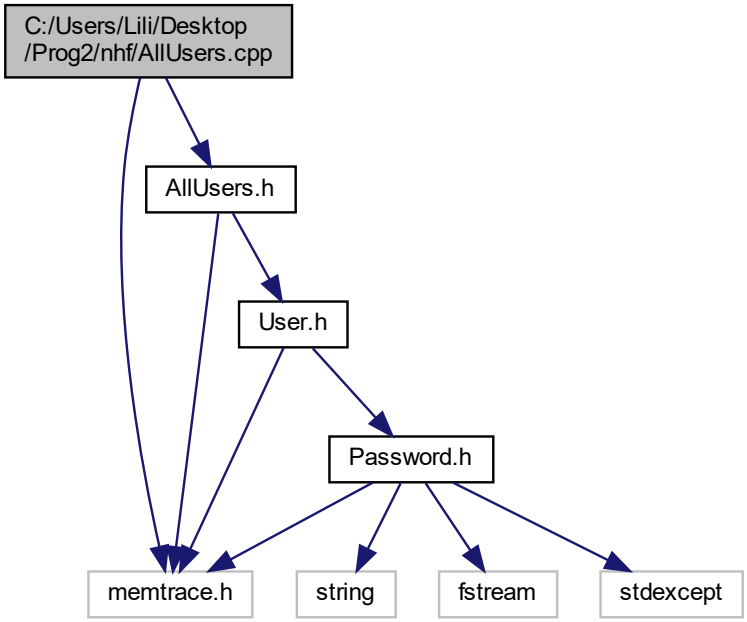
Ugrás a fájl dokumentációjához.

```
1
5 #if !defined(_ALLSPECIALISTS_H)
6 #define _ALLSPECIALISTS_H
7 #define MEMTRACE
8
9 #include "memtrace.h"
10
11 #include "Specialist.h"
12
16 class AllSpecialists
17 {
18 private:
19
20     Specialist** allSps;
22     size_t num;
23 public:
27     AllSpecialists()
28     {
29         num=0;
30         allSps=new Specialist*[num];
31     }
36     void addSpecialist(Specialist* sp);
44     Specialist** filterType(int t);
45
49     ~AllSpecialists()
50     {
51         delete[] allSps;
52     }
53 };
54
55 #endif  //_ALLSPECIALISTS_H
```

5.4. C:/Users/Lili/Desktop/Prog2/nhf/AllUsers.cpp fájlreferencia

```
#include "memtrace.h"
#include "AllUsers.h"
```

Az AllUsers.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

5.4.1. Makródefiníciók dokumentációja

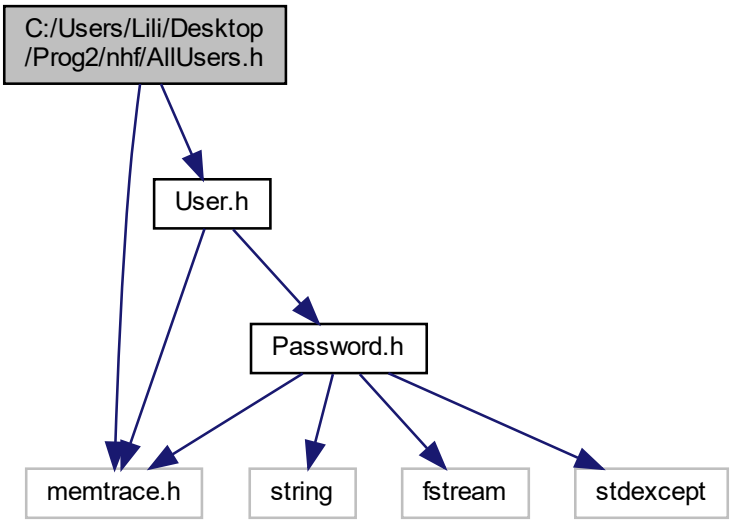
5.4.1.1. MEMTRACE

```
#define MEMTRACE
```

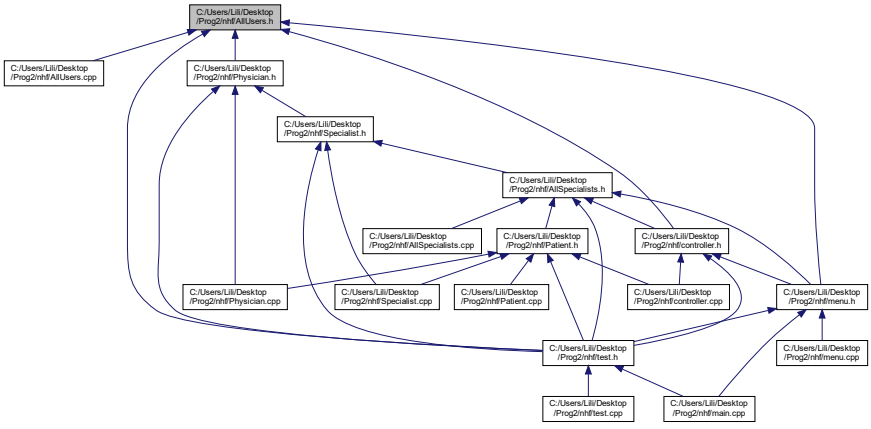
5.5. C:/Users/Lili/Desktop/Prog2/nhf/AllUsers.h fájlreferencia

```
#include "memtrace.h"
#include "User.h"
```

Az AllUsers.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class AllUsers

Makródefiníciók

- #define MEMTRACE

5.5.1. Makródefiníciók dokumentációja

5.5.1.1. MEMTRACE

```
#define MEMTRACE
```

5.6. C:/Users/Lili/Desktop/Prog2/nhf/AllUsers.h

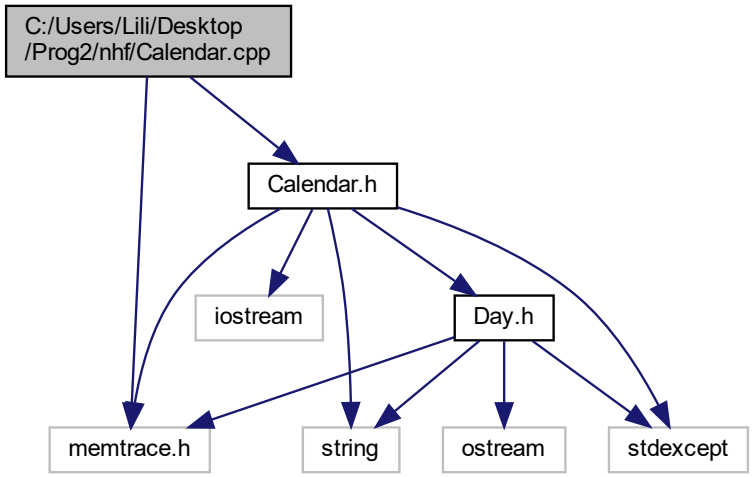
Ugrás a fájl dokumentációjához.

```
1
5 #if !defined(_ALLUSERS_H)
6 #define _ALLUSERS_H
7 #define MEMTRACE
8
9 #include "memtrace.h"
10
11 #include "User.h"
12
16 class AllUsers
17 {
18 private:
19     User** all;
20     size_t num;
21 public:
25     AllUsers() {
26         num = 0;
27         all = new User *[0];
28     }
34     bool contains(const std::string& name);
40     bool containsID(const std::string& ID);
45     void addUser(User *l);
46
52     User* getUser(const std::string& name);
53
58     std::string exporter ();
59
63     int getNum() const;
64
68     ~AllUsers()
69     {
70         for(size_t i=0; i<num; i++)
71             delete all[i];
72         delete[] all;
73     }
74 };
75
76 #endif  //_ALLUSERS_H
```

5.7. C:/Users/Lili/Desktop/Prog2/nhf/Calendar.cpp fájlreferencia

```
#include "memtrace.h"
#include "Calendar.h"
```

A Calendar.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

5.7.1. Makródefiníciók dokumentációja

5.7.1.1. MEMTRACE

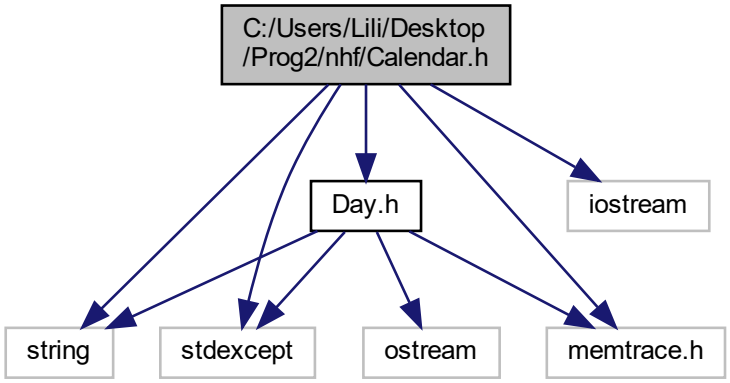
```
#define MEMTRACE
```

5.8. C:/Users/Lili/Desktop/Prog2/nhf/Calendar.h fájlreferencia

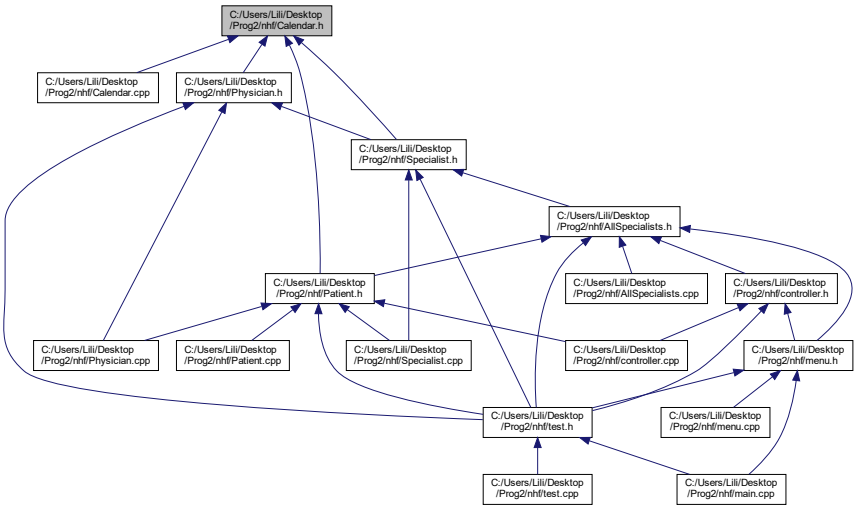
```
#include <string>
#include <stdexcept>
#include <iostream>
#include "memtrace.h"
```

```
#include "Day.h"
```

A Calendar.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class Calendar

Makródefiníciók

- #define MEMTRACE

5.8.1. Makródefiníciók dokumentációja

5.8.1.1. MEMTRACE

```
#define MEMTRACE
```

5.9. C:/Users/Lili/Desktop/Prog2/nhf/Calendar.h

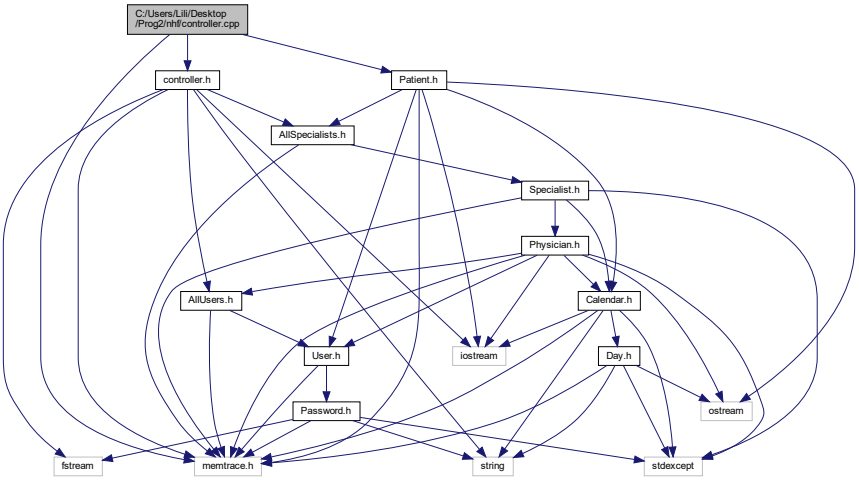
Ugrás a fájl dokumentációjához.

```
1
5 #if !defined(_CALENDAR_H)
6 #define _CALENDAR_H
7 #define MEMTRACE
8
9 #include <string>
10 #include <stdexcept>
11 #include <iostream>
12
13 #include "memtrace.h"
14
15 #include "Day.h"
16
20 class Calendar
21 {
22 private:
23     Day days [31];
24 public:
28     Calendar()
29     {
30         for(int i=0; i<31; i++)
31         {
32             days[i]=Day();
33         }
34     }
40     void print(std::ostream& os);
47     void addEvent(int day, int section, const std::string& who);
52     std::string exporter();
56     ~Calendar() = default;
57 };
58
59 #endif  //_CALENDAR_H
```

5.10. C:/Users/Lili/Desktop/Prog2/nhf/controller.cpp fájlreferencia

```
#include "memtrace.h"
#include "controller.h"
#include "Patient.h"
```

A controller.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE
- #define DEBUG

Függvények

- void physician_sign_in (AllUsers &Users)
- void patient_sign_in (AllUsers &Users, AllSpecialists &Specialists)
- void specialist_sign_in (AllUsers &Users, AllSpecialists &Specialists)
- void addEvent (std::string s, Calendar &c)
- Password readPassword (std::fstream &fs, std::string &line)
- Calendar readCalendar (std::fstream &fs, std::string &line)
- Patient ∗ readPatient (std::fstream &fs, std::string &line)
- void import_state (AllUsers &allUsers, AllSpecialists &allSpecialists, const std::string &file)
- void export_state (AllUsers &users, AllSpecialists &specialists)

5.10.1. Makródefiníciók dokumentációja

5.10.1.1. DEBUG

```
#define DEBUG
```

5.10.1.2. MEMTRACE

```
#define MEMTRACE
```

5.10.2. Függvények dokumentációja

5.10.2.1. addEvent()

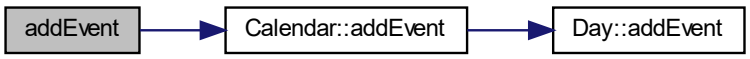
```
void addEvent (
            std::string s,
            Calendar & c )
```

Egy esemény beolvasása egy sor szövegből és hozzáadása a naptárhoz.

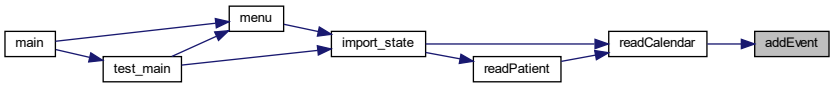
Paraméterek

s	a sor szöveg, melyből beolvasunk.
c	a naptár, amibe az eseményt beírjuk.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.10.2.2. export_state()

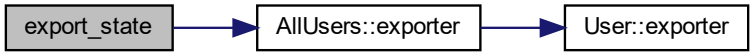
```
void export_state (
            AllUsers & users,
            AllSpecialists & specialists )
```

A fileba kiírás vezérléséért felelős függvény.

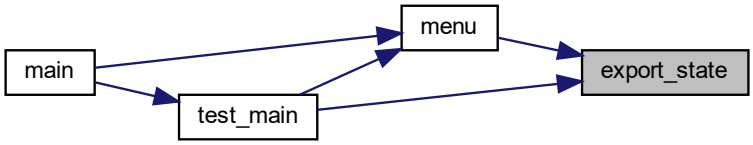
Paraméterek

users	a felhasználókat tároló objektum.
specialists	a szakorvosokat tároló objektum.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.10.2.3. import_state()

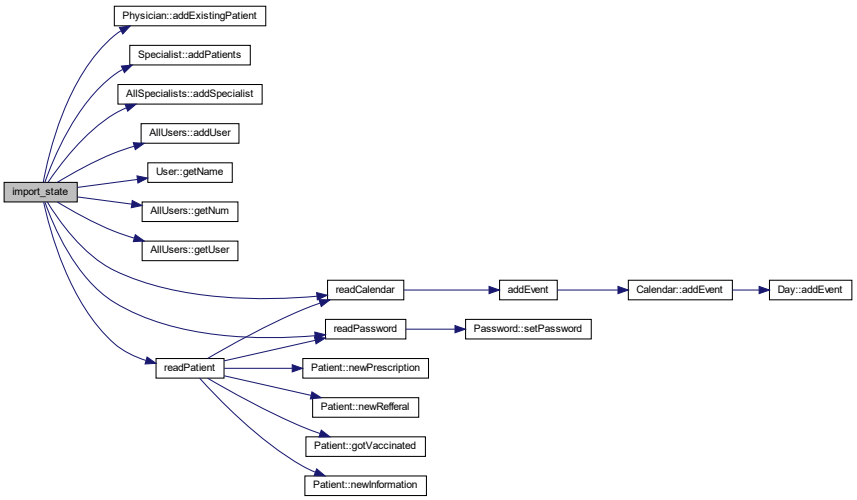
```
void import_state (
            AllUsers & allUsers,
            AllSpecialists & allSpecialists,
            const std::string & file )
```

A paraméterként megadott string segítségével beolvassa a rendszer állapotát.

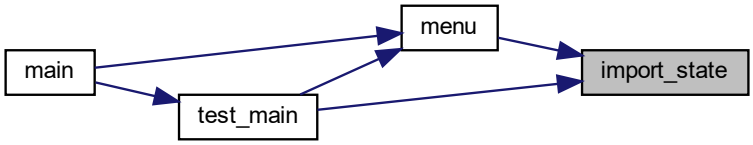
Paraméterek

allUsers	a felhasználókat tároló objektum.
allSpecialists	a szakorvosokat tároló objektum.
file	a beolvasandó file elérési útja.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.10.2.4. patient_sign_in()

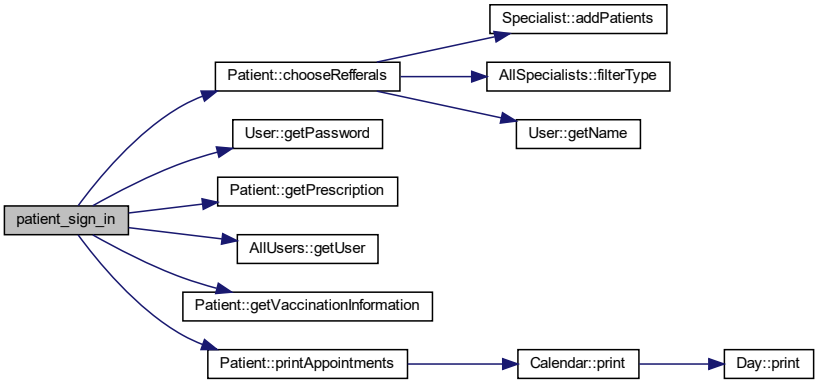
```
void patient_sign_in (
            AllUsers & Users,
            AllSpecialists & Specialists )
```

A pácienskénti történő belépést, cselekmény-végrehajtást vezérli, kontrollálja. A menu() függvény hívja meg, ő hív meg további függvényeket.

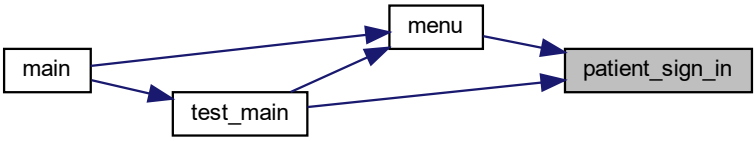
Paraméterek

Users	az összes profilt tartalmazó heterogén kollekció, a menüben lett létrehozva.
Specialists	az sszes szakorvosi profilt tartalmazó heterogén kollekció, a menüben lett létrehozva.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.10.2.5. physician_sign_in()

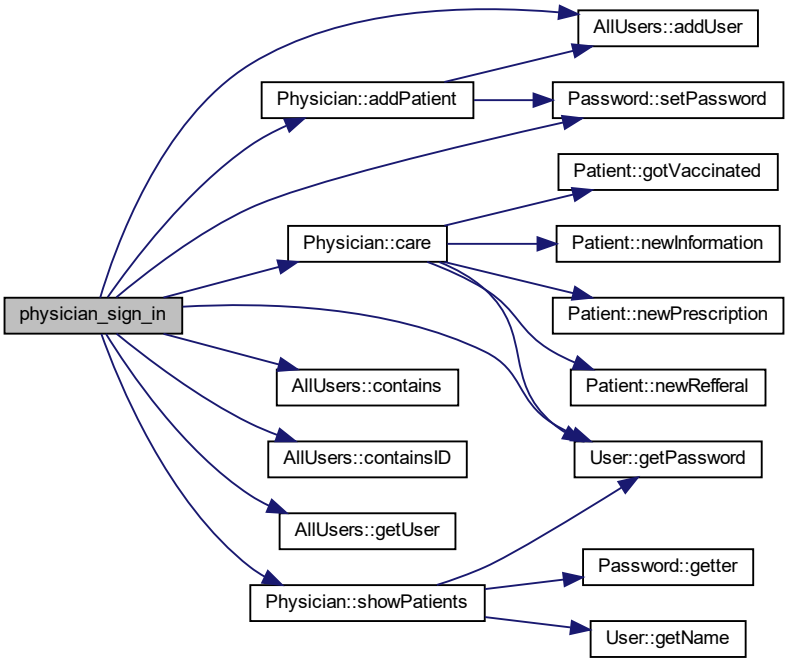
```
void physician_sign_in (
            AllUsers & Users )
```

A háziorvosként történő belépést, cselekmény-végrehajtást vezérli, kontrollálja. A menu() függvény hívja meg, ő hív meg további függvényeket.

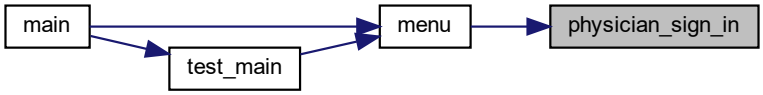
Paraméterek

Users	az összes profilt tartalmazó heterogén kollekció, a menüben lett létrehozva.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.10.2.6. readCalendar()

```
Calendar readCalendar (
            std::fstream & fs,
            std::string & line )
```

Naptár beolvasása egy file alapú adatfolyamból.

Paraméterek

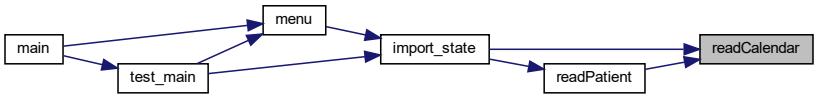
fs	az adatfolyam, amiből olvasunk.
line	az aktuálisan kiolvasott sor.

Visszatérési érték

a létrehozott naptár.

A függvény hívási gráfja:

A függvény hívó gráfja:



5.10.2.7. readPassword()

```
Password readPassword (
            std::fstream & fs,
            std::string & line )
```

Jelszó beolvasása egy file alapú adatfolyamból.

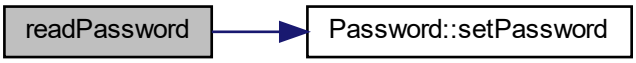
Paraméterek

fs	az adatfolyam, amiből olvasunk.
line	az aktuálisan kiolvasott sor.

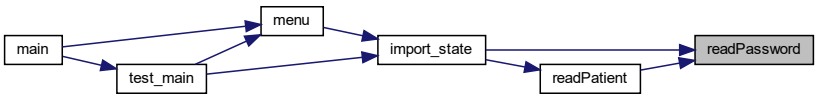
Visszatérési érték

a létrehozott jelszó.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.10.2.8. readPatient()

```
Patient * readPatient (
            std::fstream & fs,
            std::string & line )
```

Páciens beolvasása egy file alapú adatfolyamból.

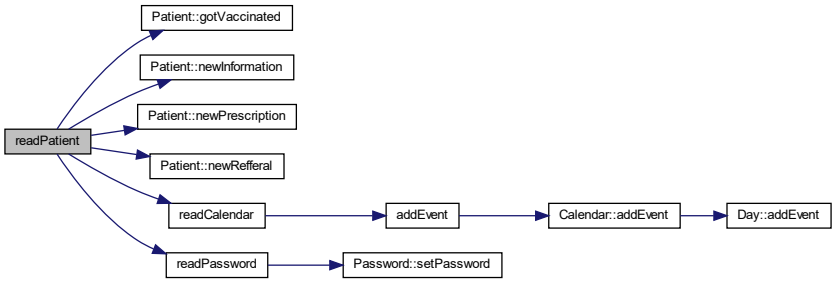
Paraméterek

fs	az adatfolyam, amiből olvasunk.
line	az aktuálisan kiolvasott sor.

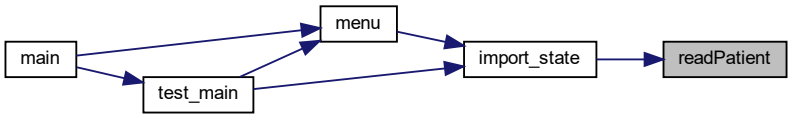
Visszatérési érték

egy dinamikusan lefoglalt páciensre mutató mutató.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.10.2.9. specialist_sign_in()

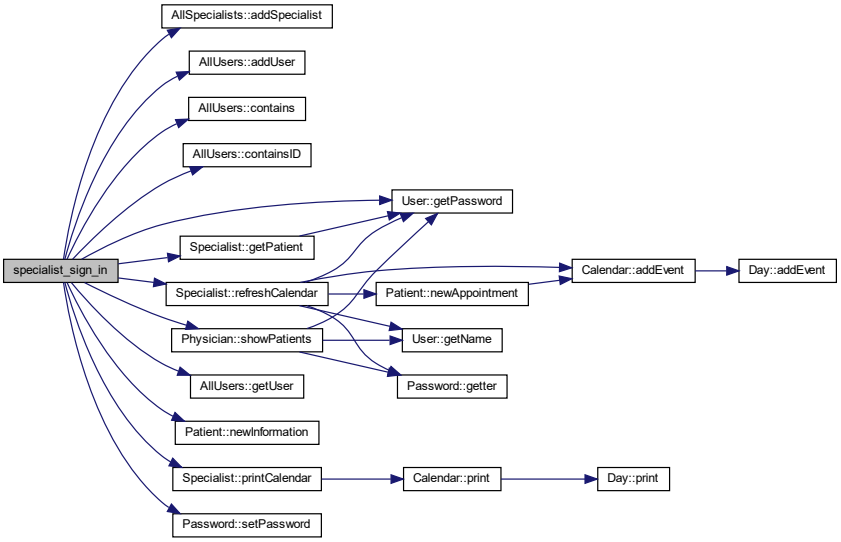
```
void specialist_sign_in (
            AllUsers & Users,
            AllSpecialists & Specialists )
```

A szakorvosként történő belépést, cselekmény-végrehajtást vezérli, kontrollálja. A menu() függvény hívja meg,ő hív meg további függvényeket.

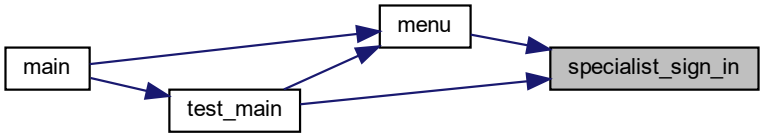
Paraméterek

Users	az összes profilt tartalmazó heterogén kollekció, a menüben lett létrehozva.
Specialists	az sszes szakorvosi profilt tartalmazó heterogén kollekció, a menüben lett létrehozva.

A függvény hívási gráfja:



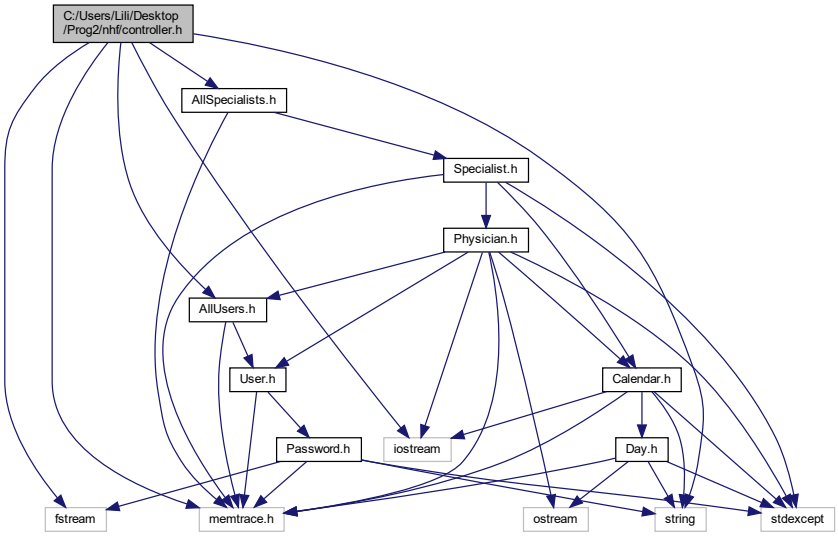
A függvény hívó gráfja:



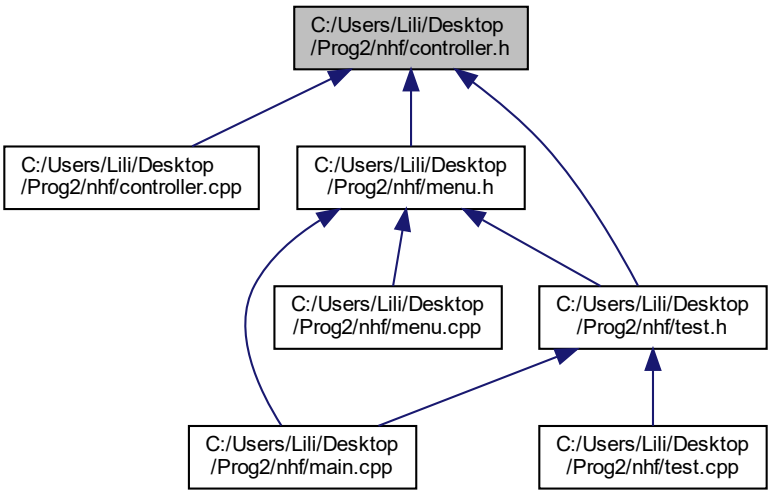
5.11. C:/Users/Lili/Desktop/Prog2/nhf/controller.h fájlreferencia

```
#include <string>
#include <iostream>
#include <fstream>
#include "memtrace.h"
#include "AllUsers.h"
#include "AllSpecialists.h"
```

A controller.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Makródefiníciók

- #define MEMTRACE

Függvények

- void physician_sign_in (AllUsers &Users)

- void patient_sign_in (AllUsers &Users, AllSpecialists &Specialists)
- void specialist_sign_in (AllUsers &Users, AllSpecialists &Specialists)
- void import_state (AllUsers &allUsers, AllSpecialists &allSpecialists, const std::string &file)
- void export_state (AllUsers &users, AllSpecialists &specialists)

5.11.1. Makródefiníciók dokumentációja

5.11.1.1. MEMTRACE

```
#define MEMTRACE
```

5.11.2. Függvények dokumentációja

5.11.2.1. export_state()

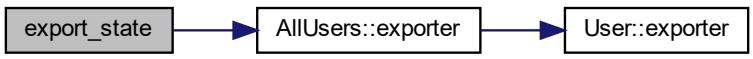
```
void export_state (
            AllUsers & users,
            AllSpecialists & specialists )
```

A fileba kiírás vezérléséért felelős függvény.

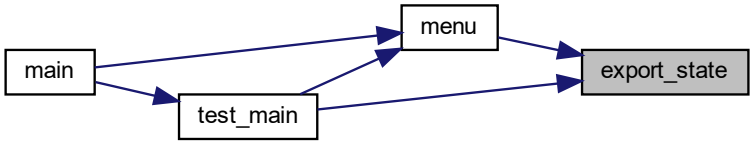
Paraméterek

users	a felhasználókat tároló objektum.
specialists	a szakorvosokat tároló objektum.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.11.2.2. import_state()

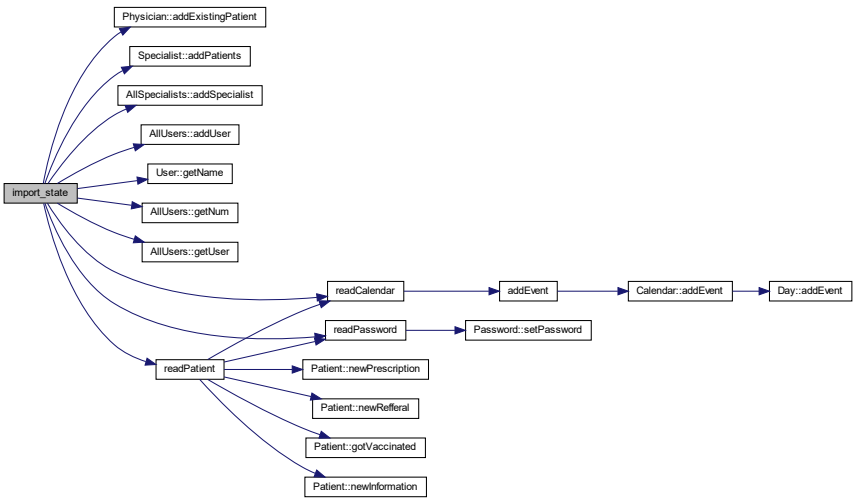
```
void import_state (
            AllUsers & allUsers,
            AllSpecialists & allSpecialists,
            const std::string & file )
```

A paraméterként megadott string segítségével beolvassa a rendszer állapotát.

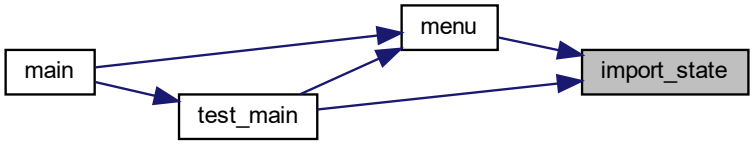
Paraméterek

allUsers	a felhasználókat tároló objektum.
allSpecialists	a szakorvosokat tároló objektum.
file	a beolvasandó file elérési útja.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.11.2.3. patient_sign_in()

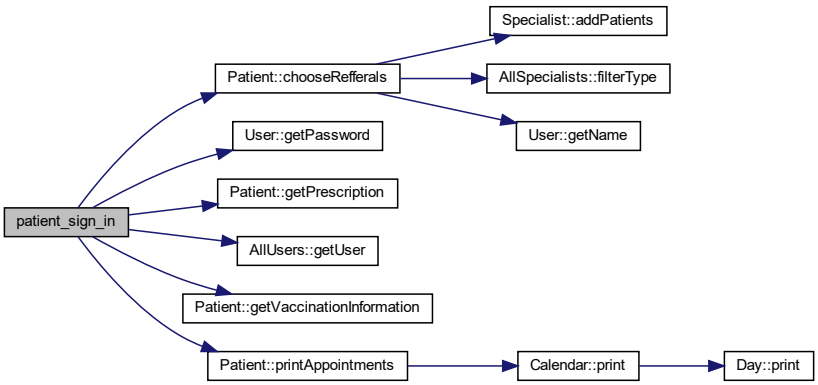
```
void patient_sign_in (
            AllUsers & Users,
            AllSpecialists & Specialists )
```

A pácienskent történő belépést, cselekmény-végrehajtást vezérli, kontrollálja. A menu() függvény hívja meg, ő hív meg további függvényeket.

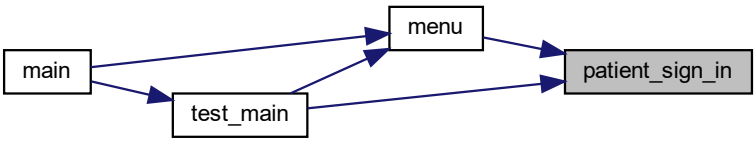
Paraméterek

Users	az összes profilt tartalmazó heterogén kollekció, a menüben lett létrehozva.
Specialists	az sszes szakorvosi profilt tartalmazó heterogén kollekció, a menüben lett létrehozva.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.11.2.4. physician_sign_in()

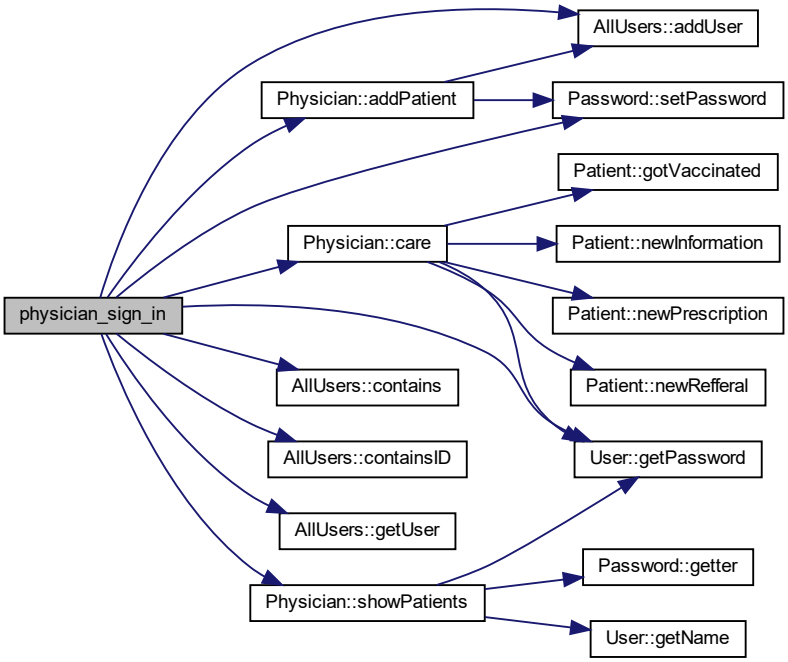
```
void physician_sign_in (
            AllUsers & Users )
```

A háziorvosként történő belépést, cselekmény-végrehajtást vezérli, kontrollálja. A menu() függvény hívja meg, ő hív meg további függvényeket.

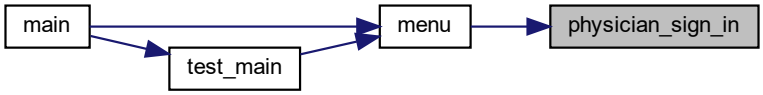
Paraméterek

Users	az összes profilt tartalmazó heterogén kollekció, a menüben lett létrehozva.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.11.2.5. specialist_sign_in()

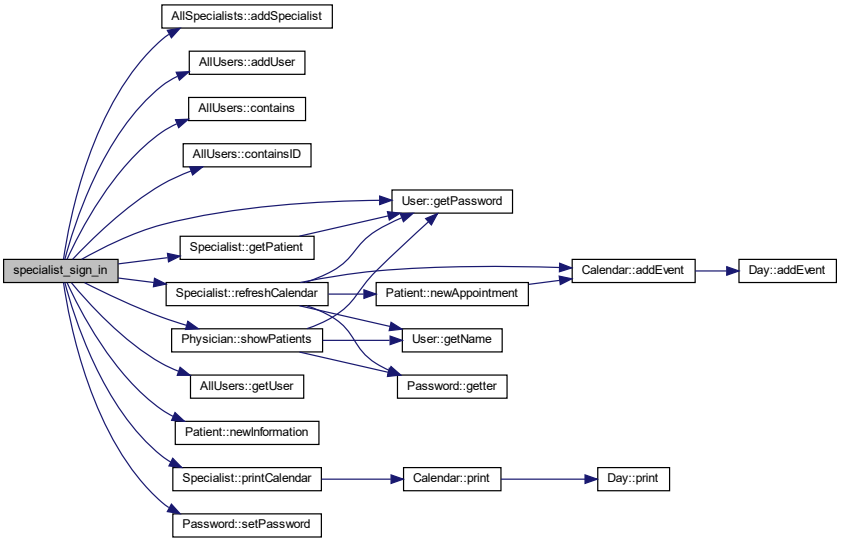
```
void specialist_sign_in (
            AllUsers & Users,
            AllSpecialists & Specialists )
```

A szakorvosként történő belépést, cselekmény-végrehajtást vezérli, kontrollálja. A menu() függvény hívja meg,ő hív meg további függvényeket.

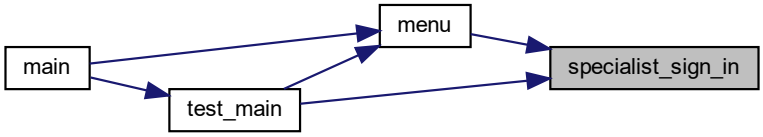
Paraméterek

Users	az összes profilt tartalmazó heterogén kollekció, a menüben lett létrehozva.
Specialists	az sszes szakorvosi profilt tartalmazó heterogén kollekció, a menüben lett létrehozva.

A függvény hívási gráfja:



A függvény hívó gráfja:



5.12. C:/Users/Lili/Desktop/Prog2/nhf/controller.h

Ugrás a fájl dokumentációjához.

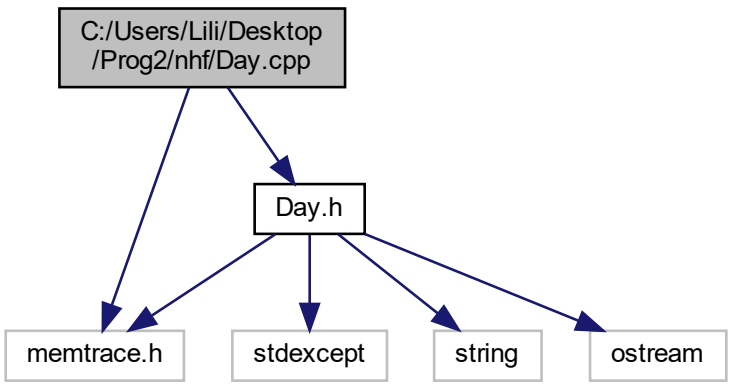
```
1
5 #ifndef NHF_CONTROLLER_H
6 #define NHF_CONTROLLER_H
7 #define MEMTRACE
8
9 #include <string>
10 #include <iostream>
11 #include <fstream>
12
13 #include "memtrace.h"
14
15 #include "AllUsers.h"
```

```
16 #include "AllSpecialists.h"
17
18
24 void physician_sign_in (AllUsers& Users);
25
32 void patient_sign_in (AllUsers& Users, AllSpecialists& Specialists);
33
40 void specialist_sign_in (AllUsers& Users, AllSpecialists& Specialists);
41
48  void import_state(AllUsers& allUsers, AllSpecialists& allSpecialists, const std::string& file);
49
55 void export_state(AllUsers& users, AllSpecialists& specialists);
56 #endif //NHF_CONTROLLER_H
```

5.13. C:/Users/Lili/Desktop/Prog2/nhf/Day.cpp fájlreferencia

```
#include "memtrace.h"
#include "Day.h"
```

A Day.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

5.13.1. Makródefiníciók dokumentációja

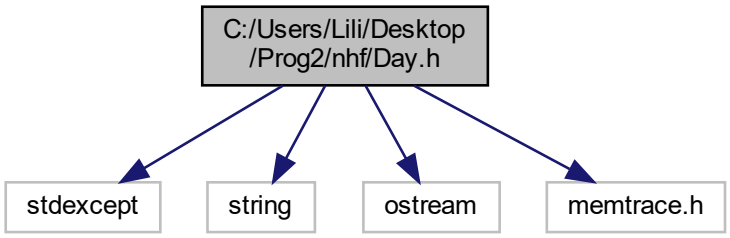
5.13.1.1. MEMTRACE

```
#define MEMTRACE
```

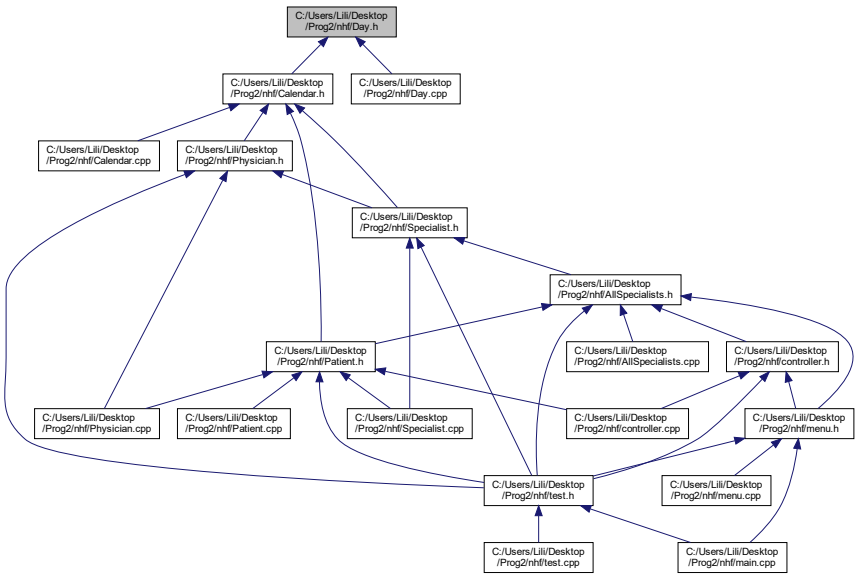
5.14. C:/Users/Lili/Desktop/Prog2/nhf/Day.h fájlreferencia

```
#include <stdexcept>
#include <string>
#include <ostream>
#include "memtrace.h"
```

A Day.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class Day

Makródefiníciók

- #define MEMTRACE

5.14.1. Makródefiníciók dokumentációja

5.14.1.1. MEMTRACE

```
#define MEMTRACE
```

5.15. C:/Users/Lili/Desktop/Prog2/nhf/Day.h

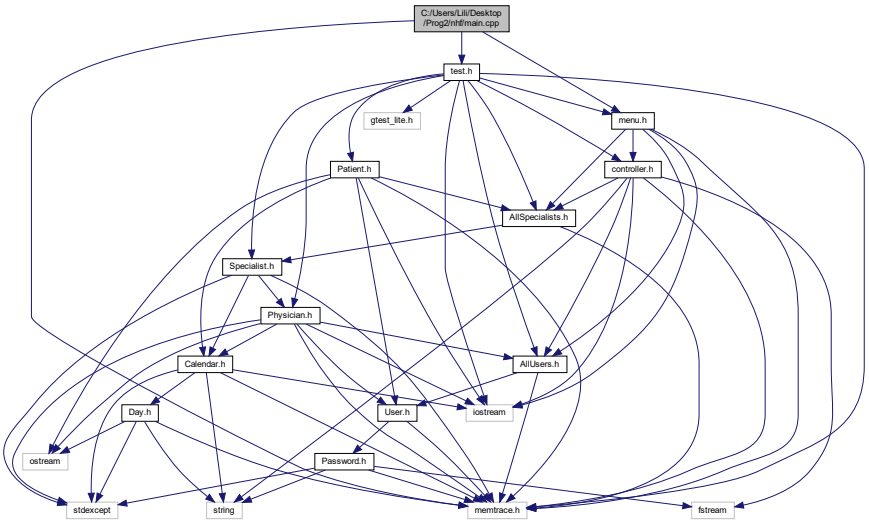
Ugrás a fájl dokumentációjához.

```
1
5 #if !defined(_DAY_H)
6 #define _DAY_H
7 #define MEMTRACE
8
9 #include <stdexcept>
10 #include <string>
11 #include <ostream>
12
13 #include "memtrace.h"
14
18 class Day
19 {
20 private:
21     std::string sections [4];
22 public:
26     Day()
27     {
28         for(int i=0; i<4; i++)
29         {
30             sections[i]="0";
31         }
32     }
39     void addEvent(int section, const std::string& who);
44     std::string exporter();
49     void print(std::ostream& os);
53     ~Day () = default;
54 };
55
56 #endif  //_DAY_H
```

5.16. C:/Users/Lili/Desktop/Prog2/nhf/main.cpp fájlreferencia

```
#include "memtrace.h"
#include "menu.h"
#include "test.h"
```

A main.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

Függvények

- int main ()

5.16.1. Makródefiníciók dokumentációja

5.16.1.1. MEMTRACE

```
#define MEMTRACE
```

5.16.2. Függvények dokumentációja

5.16.2.1. main()

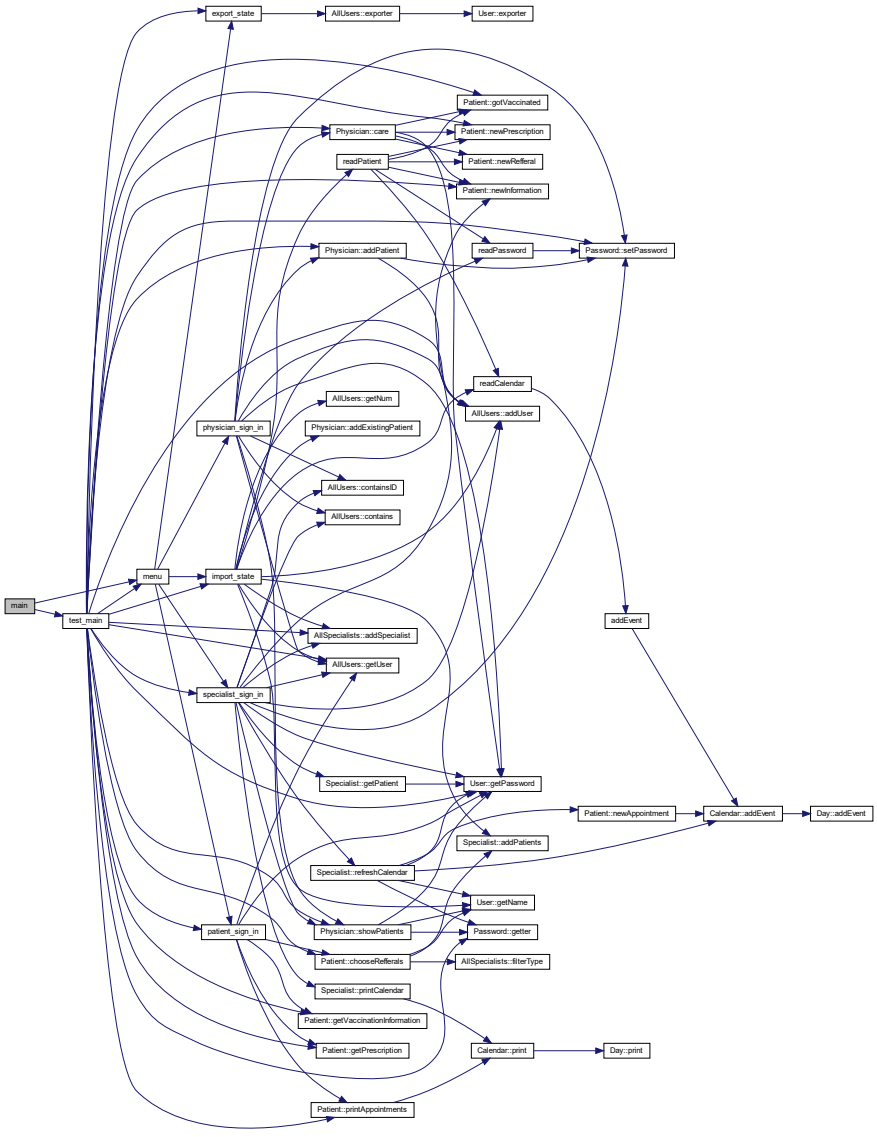
```
int main ( )
```

A program belépési pontja.

Visszatérési érték

kilépési kód.

A függvény hívási gráfja:

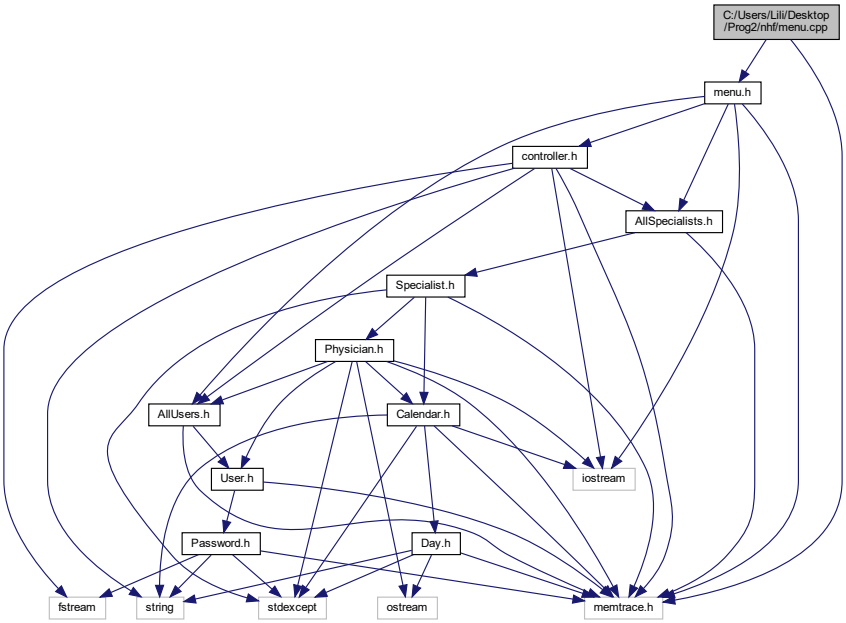


5.17. C:/Users/Lili/Desktop/Prog2/nhf/menu.cpp fájlreferencia

```
#include "memtrace.h"
```

```
#include "menu.h"
```

A menu.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

Függvények

- void menu ()

5.17.1. Makródefiníciók dokumentációja

5.17.1.1. MEMTRACE

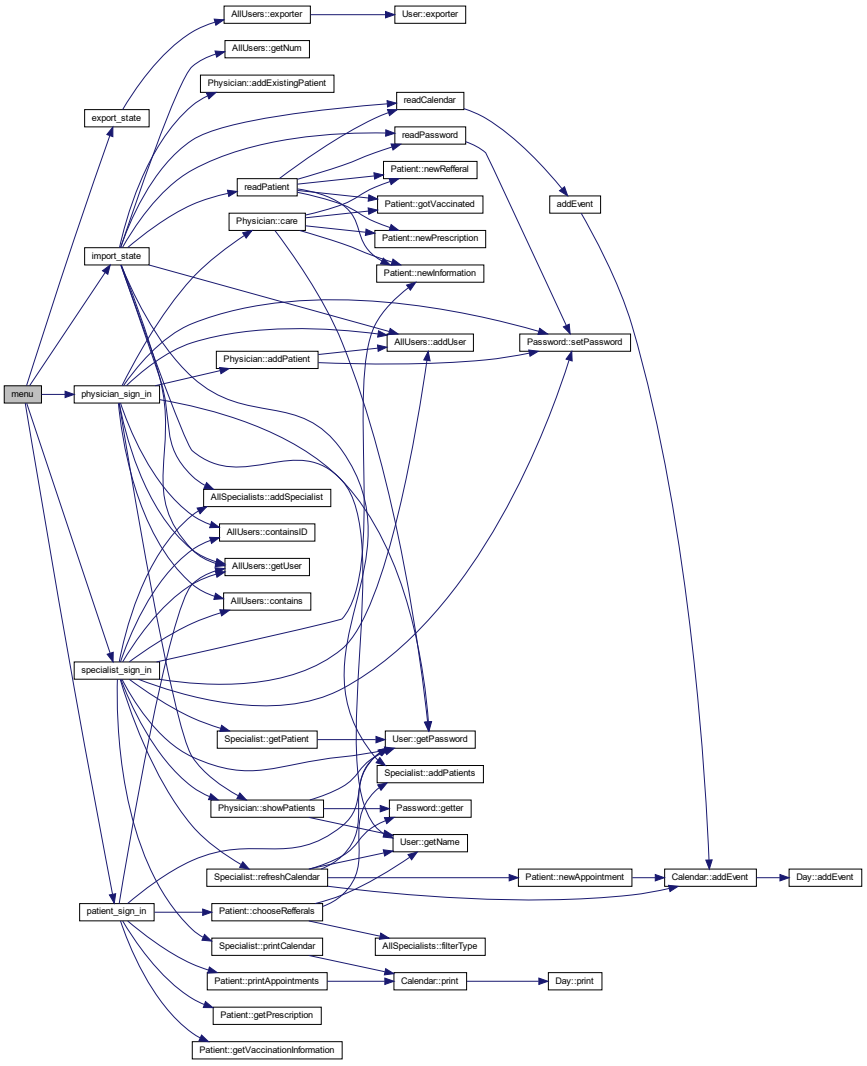
```
#define MEMTRACE
```

5.17.2. Függvények dokumentációja

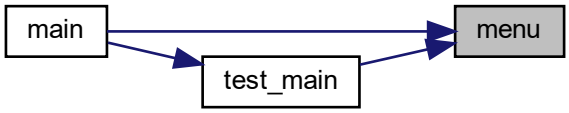
5.17.2.1. menu()

```
void menu ( )
```

A főmenűt kiíró függvény. A controller modulba tartozó függvényeket hív meg attól függően, hogy a felhasználó melyik lehetőséget választja. Itt jön létre a program alapjául szolgálü két heterogén kollekció, az AllUsers és az AllSpecialists. A menü a specifikációnak, egfelelően jön létre, kód beírása alapján lehet váltogatni a funkciókat. Amennyiben nem a felsorolt funkciók egyikét (1-5) írja be a felhasználó, a program kilép a menüből és befejeződik. A függvény hívási gráfja:



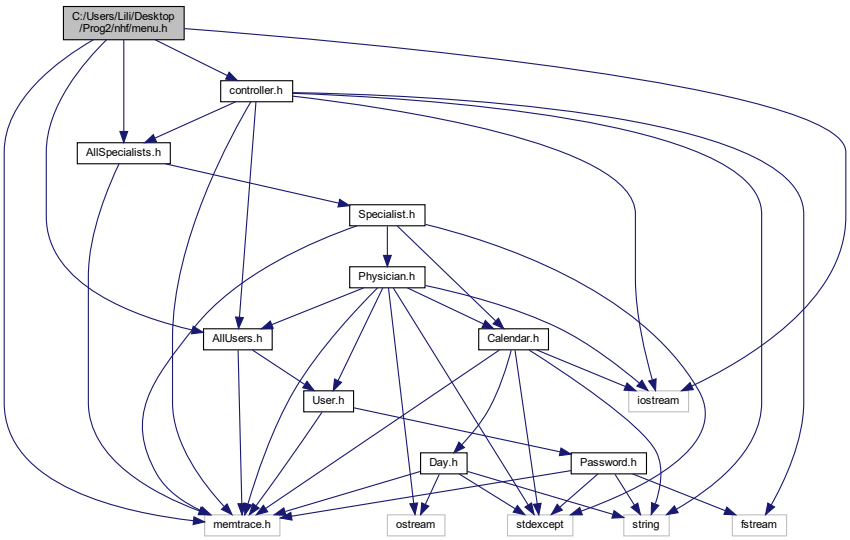
A függvény hívó gráfja:



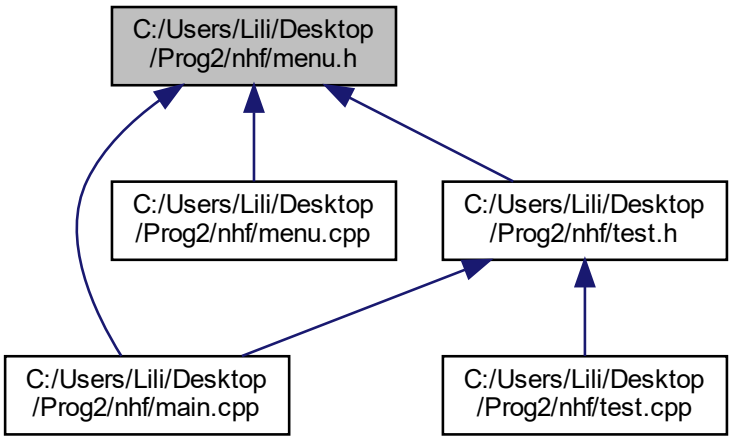
5.18. C:/Users/Lili/Desktop/Prog2/nhf/menu.h fájlreferencia

```
#include <iostream>
#include "memtrace.h"
#include "controller.h"
#include "AllUsers.h"
#include "AllSpecialists.h"
```

A menu.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Makródefiníciók

- #define MEMTRACE

Függvények

- void menu ()

5.18.1. Makródefiníciók dokumentációja

5.18.1.1. MEMTRACE

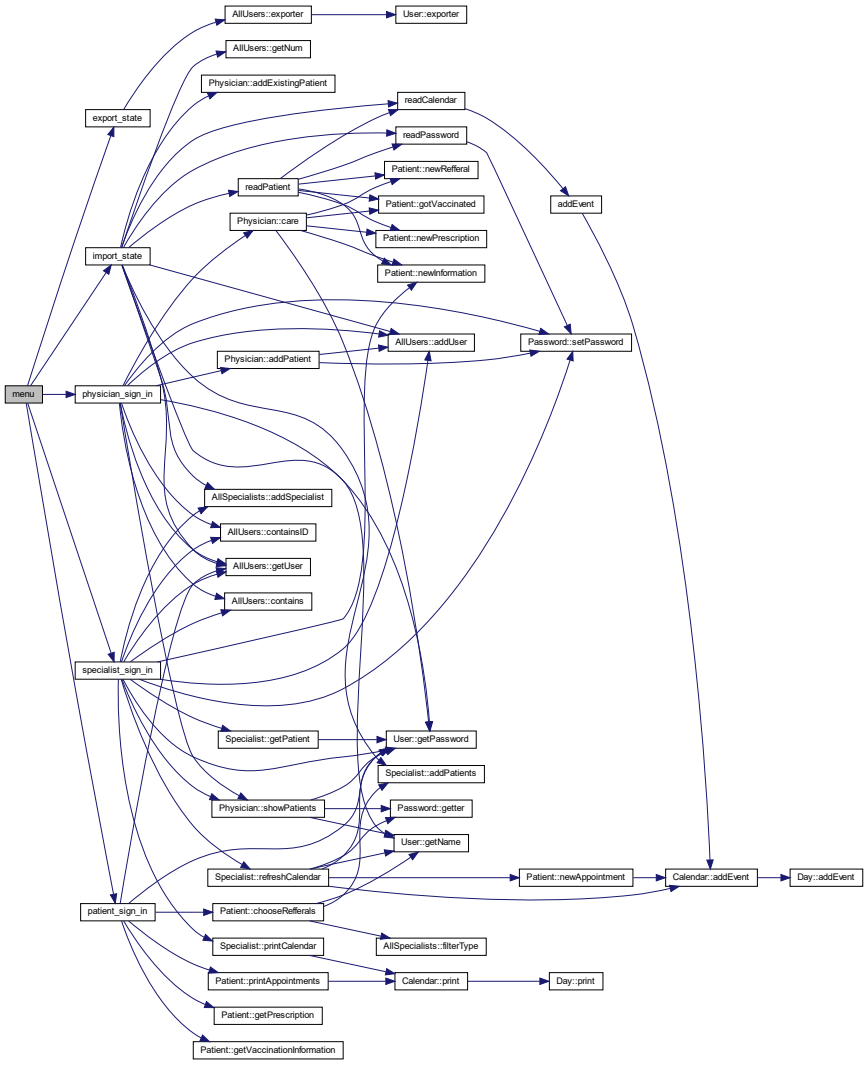
```
#define MEMTRACE
```

5.18.2. Függvények dokumentációja

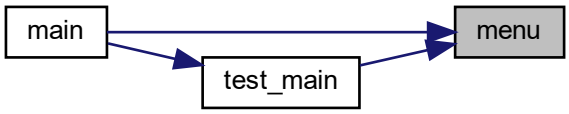
5.18.2.1. menu()

```
void menu ( )
```

A főmenűt kiíró függvény. A controller modulba tartozó függvényeket hív meg attól függően, hogy a felhasználó melyik lehetőséget választja. Itt jön létre a program alapjául szolgálü két heterogén kollekció, az AllUsers és az AllSpecialists. A menü a specifikációnak, egfelelően jön létre, kód beírása alapján lehet váltogatni a funkciókat. Amennyiben nem a felsorolt funkciók egyikét (1-5) írja be a felhasználó, a program kilép a menüből és befejeződik. A függvény hívási gráfja:



A függvény hívó gráfja:



5.19. C:/Users/Lili/Desktop/Prog2/nhf/menu.h

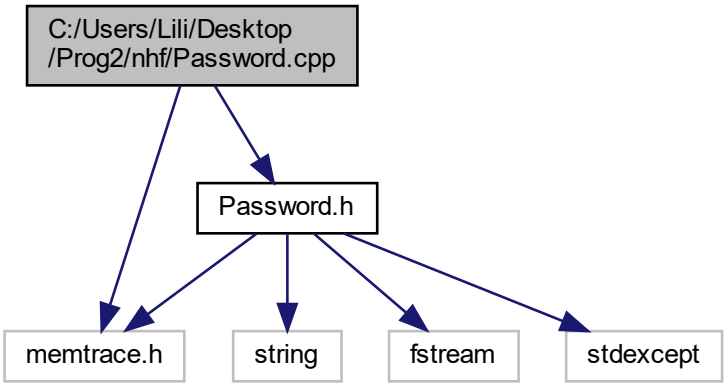
Ugrás a fájl dokumentációjához.

```
1
5 #ifndef NHF_MENU_H
6 #define NHF_MENU_H
7 #define MEMTRACE
8
9
10 #include <iostream>
11
12 #include "memtrace.h"
13
14 #include "controller.h"
15 #include "AllUsers.h"
16 #include "AllSpecialists.h"
17
24 void menu();
25
26 #endif //NHF_MENU_H
27
```

5.20. C:/Users/Lili/Desktop/Prog2/nhf/Password.cpp fájlreferencia

```
#include "memtrace.h"
#include "Password.h"
```

A Password.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

5.20.1. Makródefiníciók dokumentációja

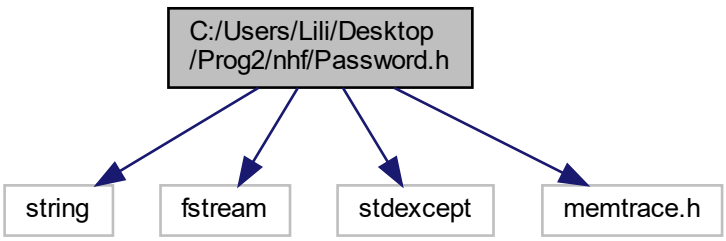
5.20.1.1. MEMTRACE

```
#define MEMTRACE
```

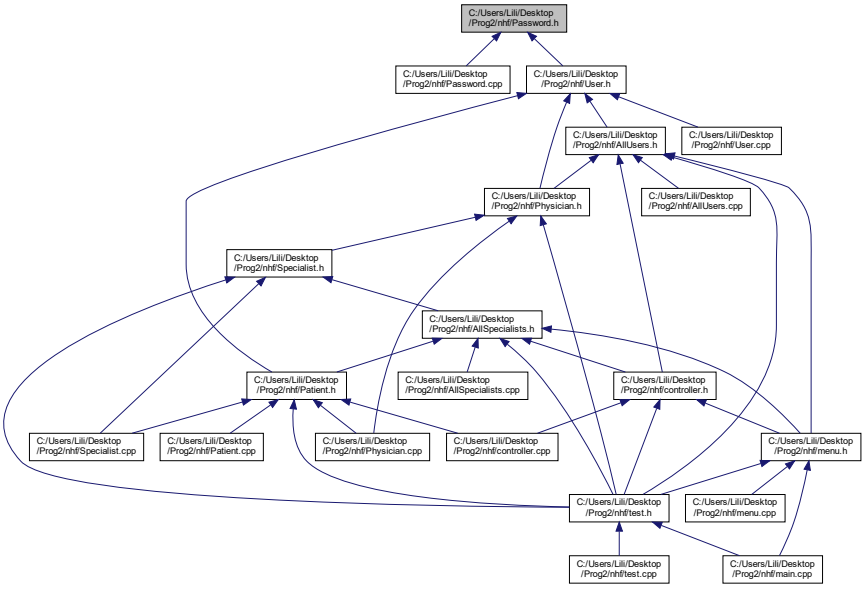
5.21. C:/Users/Lili/Desktop/Prog2/nhf/Password.h fájlreferencia

```
#include <string>
#include <fstream>
#include <stdexcept>
#include "memtrace.h"
```

A Password.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class Password

Makródefiníciók

- #define MEMTRACE

5.21.1. Makródefiníciók dokumentációja

5.21.1.1. MEMTRACE

```
#define MEMTRACE
```

5.22. C:/Users/Lili/Desktop/Prog2/nhf/Password.h

Ugrás a fájl dokumentációjához.

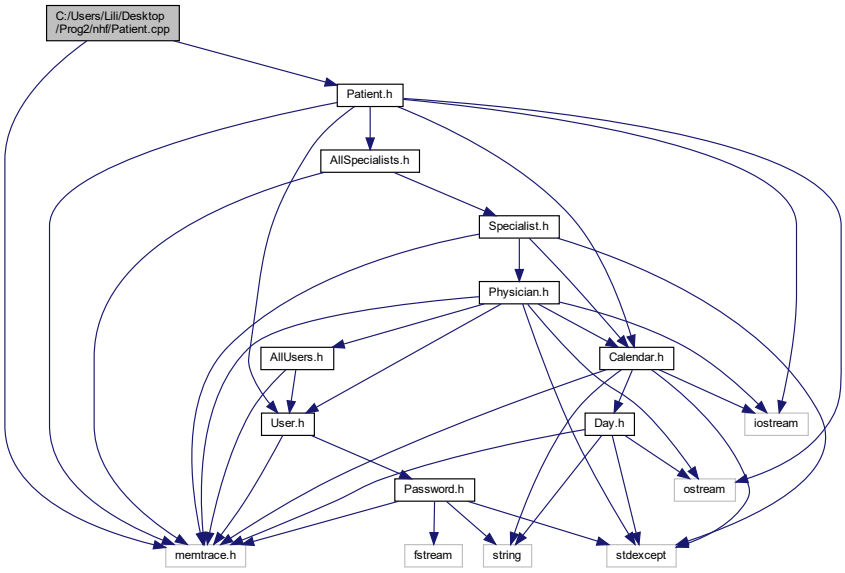
```
1
5 #if !defined(_PASSWORD_H)
6 #define _PASSWORD_H
7 #define MEMTRACE
8
9 #include <string>
10 #include <fstream>
11 #include <stdexcept>
12
```

```
13 #include "memtrace.h"
14
18 class Password
19 {
20     std::string pass;
21 public:
25     Password();
30     std::string getter ()
31     {
32         return pass;
33     }
40     bool setPassword(const std::string& p);
48     bool operator==(const std::string& p);
53     std::string exporter();
57     ~Password() = default;
58 };
59
60 #endif  //_PASSWORD_H
```

5.23. C:/Users/Lili/Desktop/Prog2/nhf/Patient.cpp fájlreferencia

```
#include "memtrace.h"
#include "Patient.h"
```

A Patient.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

5.23.1. Makródefiníciók dokumentációja

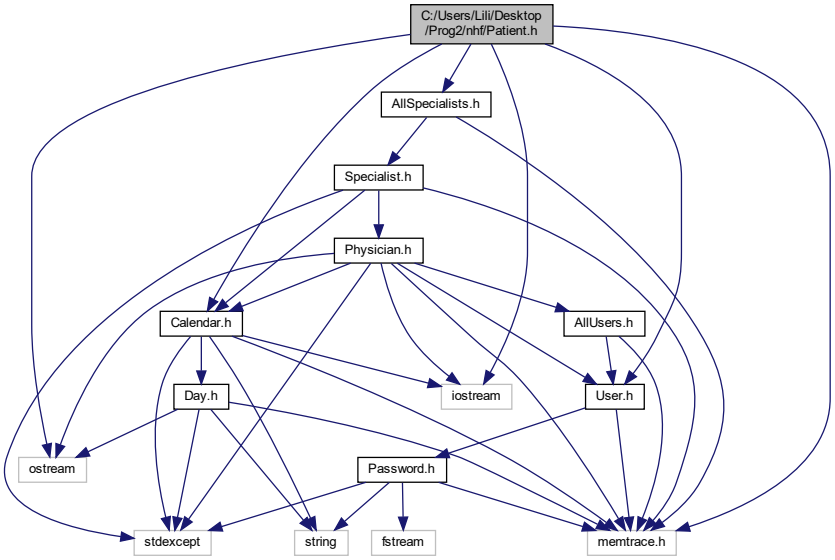
5.23.1.1. MEMTRACE

```
#define MEMTRACE
```

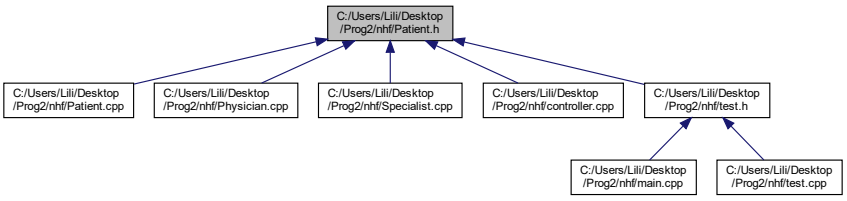
5.24. C:/Users/Lili/Desktop/Prog2/nhf/Patient.h fájlreferencia

```
#include <ostream>
#include <iostream>
#include "memtrace.h"
#include "User.h"
#include "Calendar.h"
#include "AllSpecialists.h"
```

A Patient.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class Patient

Makródefiníciók

- #define MEMTRACE

5.24.1. Makródefiníciók dokumentációja

5.24.1.1. MEMTRACE

```
#define MEMTRACE
```

5.25. C:/Users/Lili/Desktop/Prog2/nhf/Patient.h

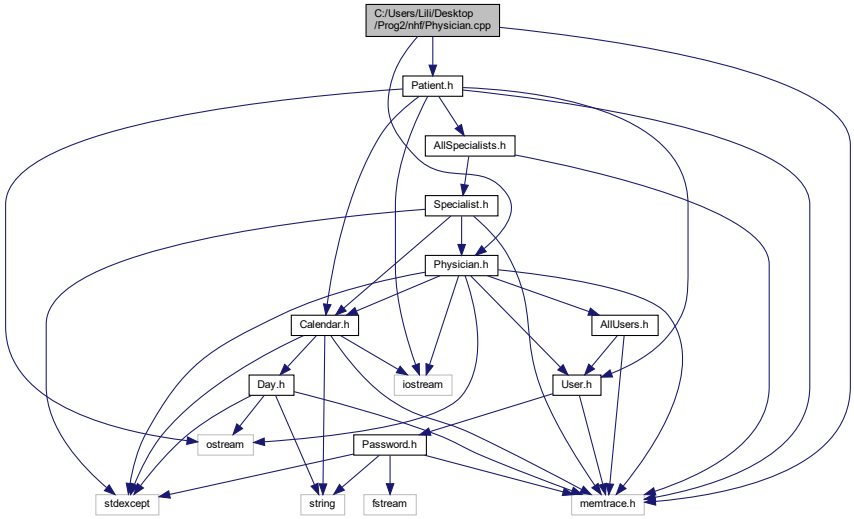
Ugrás a fájl dokumentációjához.

```
1
5 #if !defined(_PATIENT_H)
6 #define _PATIENT_H
7 #define MEMTRACE
8
9 #include <ostream>
10 #include <iostream>
11
12 #include "memtrace.h"
13
14 #include "User.h"
15 #include "Calendar.h"
16 #include "AllSpecialists.h"
17
18
22 class Patient : public User
23 {
24 private:
25     bool vaccination;
26     int* refferals;
27     size_t numOfRefferals;
28     std::string* prescriptions;
29     size_t numOfPrescriptions;
30     std::string* additionalInformation;
31     size_t numOfAdditional;
32     Calendar appointments;
33 public:
40     Patient(const std::string& Name, Password pass, Calendar a) : User(Name, pass)
41     {
42         vaccination= false;
43         numOfRefferals=0;
44         refferals=new int[0];
45         numOfPrescriptions=0;
46         prescriptions=new std::string[0];
47         numOfAdditional=0;
48         additionalInformation=new std::string[0];
49         appointments=a;
50     }
54     void gotVaccinated();
59     void newRefferal(int n);
64     void newPrescription(const std::string& p);
69     void newInformation(const std::string& I);
74     void getPrescription();
79     void getVaccinationInformation();
86     void chooseRefferals(AllSpecialists& list, std::ostream& os);
91     void printAppointments(std::ostream& os);
99     void newAppointment(int day, int sec, const std::string& who);
100
105     std::string exporter() override;
109     ~Patient() override
110     {
111         delete[] refferals;
112         delete[] prescriptions;
113         delete[] additionalInformation;
114     };
115
116
117 };
118
119 #endif  //_PATIENT_H
```

5.26. C:/Users/Lili/Desktop/Prog2/nhf/Physician.cpp fájlreferencia

```
#include "memtrace.h"
#include "Physician.h"
#include "Patient.h"
```

A Physician.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

5.26.1. Makródefiníciók dokumentációja

5.26.1.1. MEMTRACE

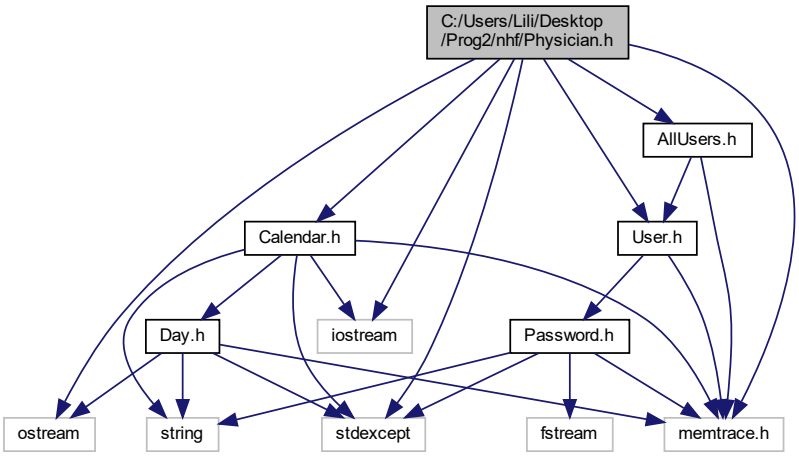
```
#define MEMTRACE
```

5.27. C:/Users/Lili/Desktop/Prog2/nhf/Physician.h fájlreferencia

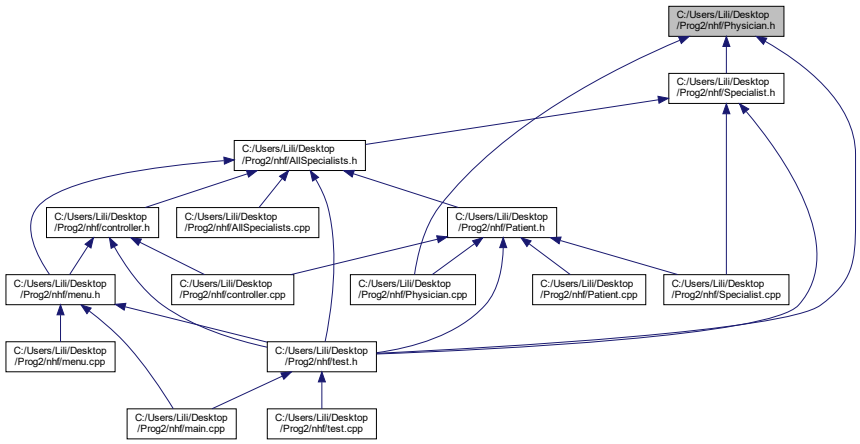
```
#include <ostream>
#include <iostream>
#include <stdexcept>
#include "memtrace.h"
#include "User.h"
#include "AllUsers.h"
```

```
#include "Calendar.h"
```

A Physician.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class Physician

Makródefiníciók

- #define MEMTRACE

5.27.1. Makródefiníciók dokumentációja

5.27.1.1. MEMTRACE

```
#define MEMTRACE
```

5.28. C:/Users/Lili/Desktop/Prog2/nhf/Physician.h

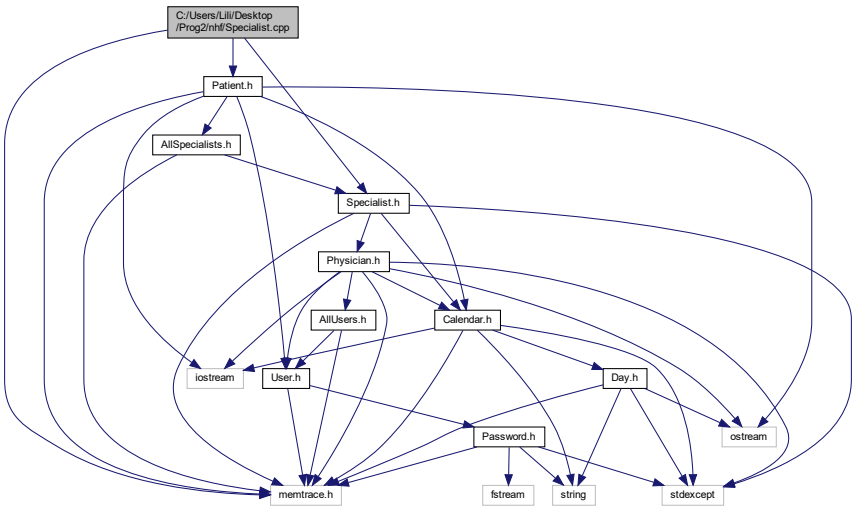
Ugrás a fájl dokumentációjához.

```
1
5 #if !defined(_PHYSICIAN_H)
6 #define _PHYSICIAN_H
7 #define MEMTRACE
8
9 #include <ostream>
10 #include <iostream>
11 #include <stdexcept>
12
13 #include "memtrace.h"
14
15 #include "User.h"
16 #include "AllUsers.h"
17 #include "Calendar.h"
18
19 class Patient;
20
24 class Physician : public User
25 {
26 protected:
27     int numOfPatients;
28     Patient** patients;
29 public:
35     Physician(const std::string &n, Password p) : User(n, p), numOfPatients(0)
36     {
37         patients=new Patient* [0];
38     }
39
44     std::string exporter() override;
45
52     void addPatient(const std::string& taj,const std::string& name, AllUsers& users);
58     void care(const std::string& taj, std::ostream& os);
63     void showPatients(std::ostream& os) const;
69     void addExistingPatient(Patient *pPatient);
73     ~Physician()
74     {
75         delete[] patients;
76     }
77 };
78
79 #endif  //_PHYSICIAN_H
```

5.29. C:/Users/Lili/Desktop/Prog2/nhf/Specialist.cpp fájlreferencia

```
#include "memtrace.h"
#include "Specialist.h"
#include "Patient.h"
```

A Specialist.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

5.29.1. Makródefiníciók dokumentációja

5.29.1.1. MEMTRACE

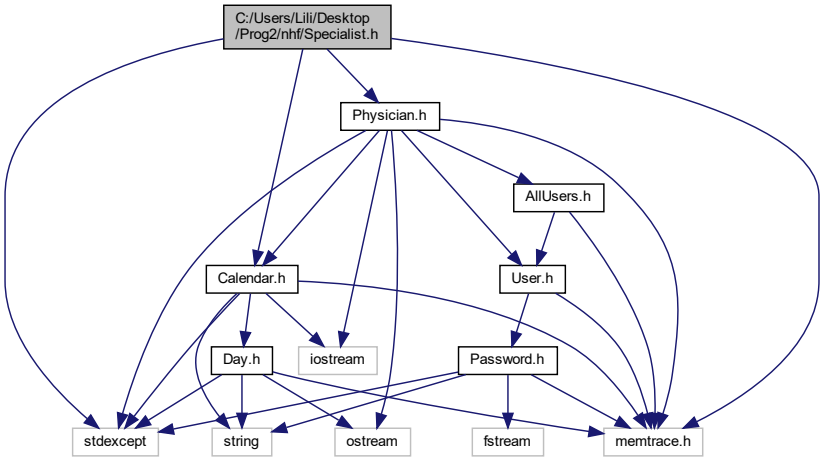
```
#define MEMTRACE
```

5.30. C:/Users/Lili/Desktop/Prog2/nhf/Specialist.h fájlreferencia

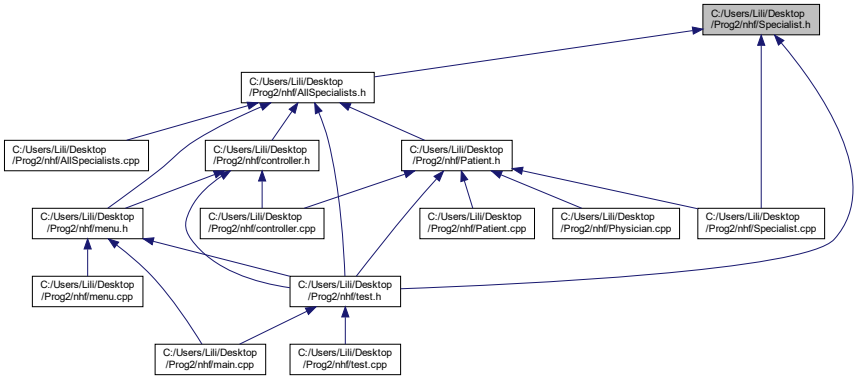
```
#include <stdexcept>
#include "memtrace.h"
#include "Physician.h"
```

```
#include "Calendar.h"
```

A Specialist.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class Specialist

Makródefiníciók

- #define MEMTRACE

5.30.1. Makródefiníciók dokumentációja

5.30.1.1. MEMTRACE

```
#define MEMTRACE
```

5.31. C:/Users/Lili/Desktop/Prog2/nhf/Specialist.h

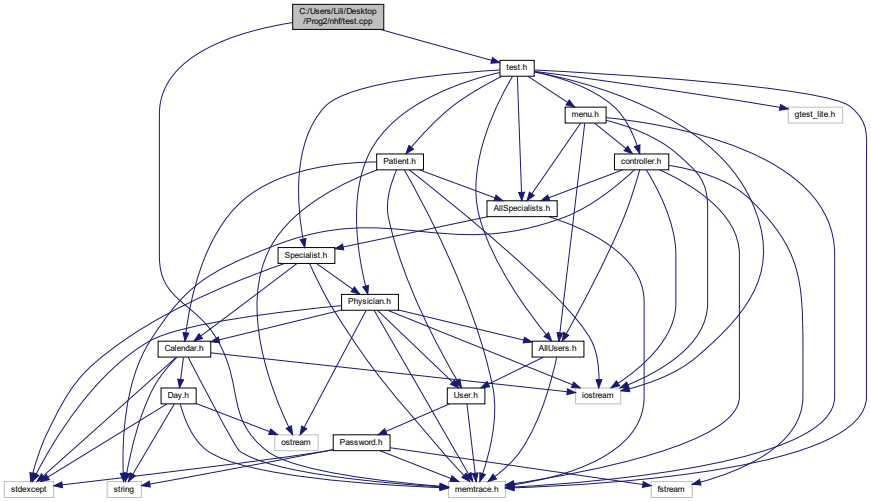
Ugrás a fájl dokumentációjához.

```
1
5 #if !defined(_SPECIALIST_H)
6 #define _SPECIALIST_H
7 #define MEMTRACE
8
9 #include <stdexcept>
10
11 #include "memtrace.h"
12
13 #include "Physician.h"
14 #include "Calendar.h"
15
16
17 class Patient;
18
22 class Specialist : public Physician{
23 private:
24     int type;
25     Calendar cal;
26 public:
34     Specialist(const std::string &n, Password p, Calendar a, int t) : Physician(n, p), type(t), cal(a) {}
35
42     void addPatients(Patient* p);
43
51     Patient* getPatient(const std::string& taj);
52
61     void refreshCalendar(int day, int section, Patient* who);
62
67     std::string exporter();
68
73     void printCalendar(std::ostream& os);
74
79     int getType() const;
80
81 };
82
83 #endif  //_SPECIALIST_H
```

5.32. C:/Users/Lili/Desktop/Prog2/nhf/test.cpp fájlreferencia

```
#include "memtrace.h"
#include "test.h"
```

A test.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

Függvények

- void test_main ()

5.32.1. Makródefiníciók dokumentációja

5.32.1.1. MEMTRACE

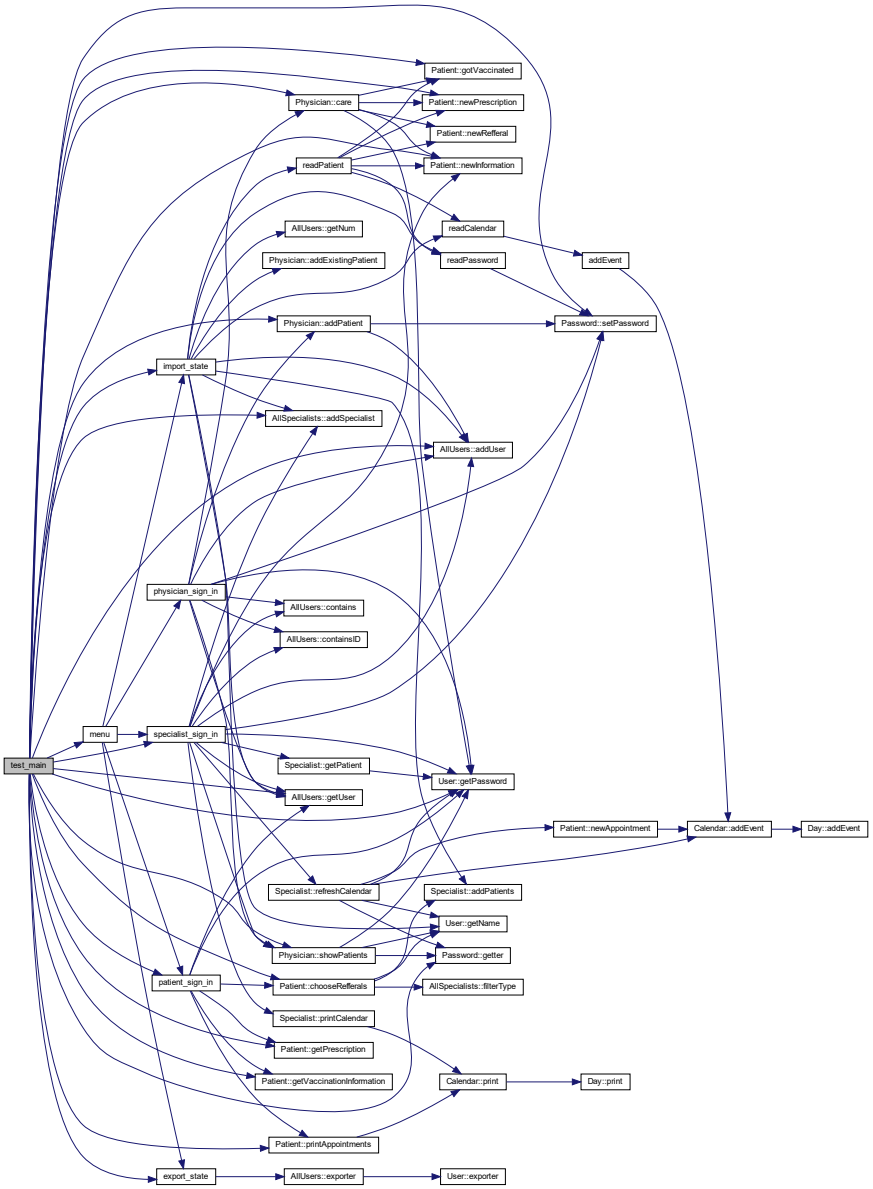
```
#define MEMTRACE
```

5.32.2. Függvények dokumentációja

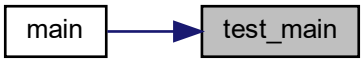
5.32.2.1. test_main()

```
void test_main ( )
```

A tesztelést megvalósító függvény. A függvény hívási gráfja:



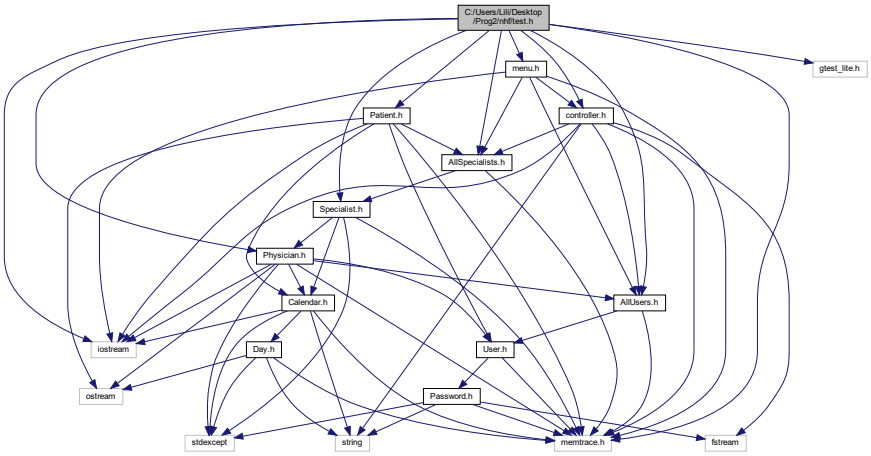
A függvény hívó gráfja:



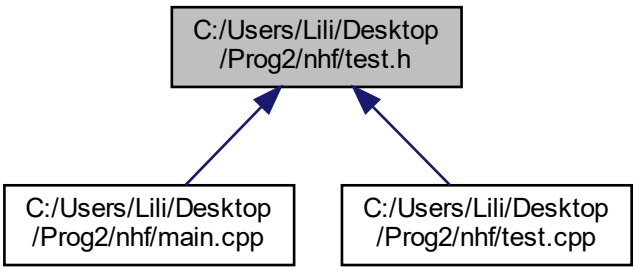
5.33. C:/Users/Lili/Desktop/Prog2/nhf/test.h fájlreferencia

```
#include <iostream>
#include "memtrace.h"
#include "AllUsers.h"
#include "Physician.h"
#include "Specialist.h"
#include "AllSpecialists.h"
#include "Patient.h"
#include "controller.h"
#include "menu.h"
#include "gtest_lite.h"
```

A test.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Makródefiníciók

- #define MEMTRACE

Függvények

- void test_main ()

5.33.1. Makródefiníciók dokumentációja

5.33.1.1. MEMTRACE

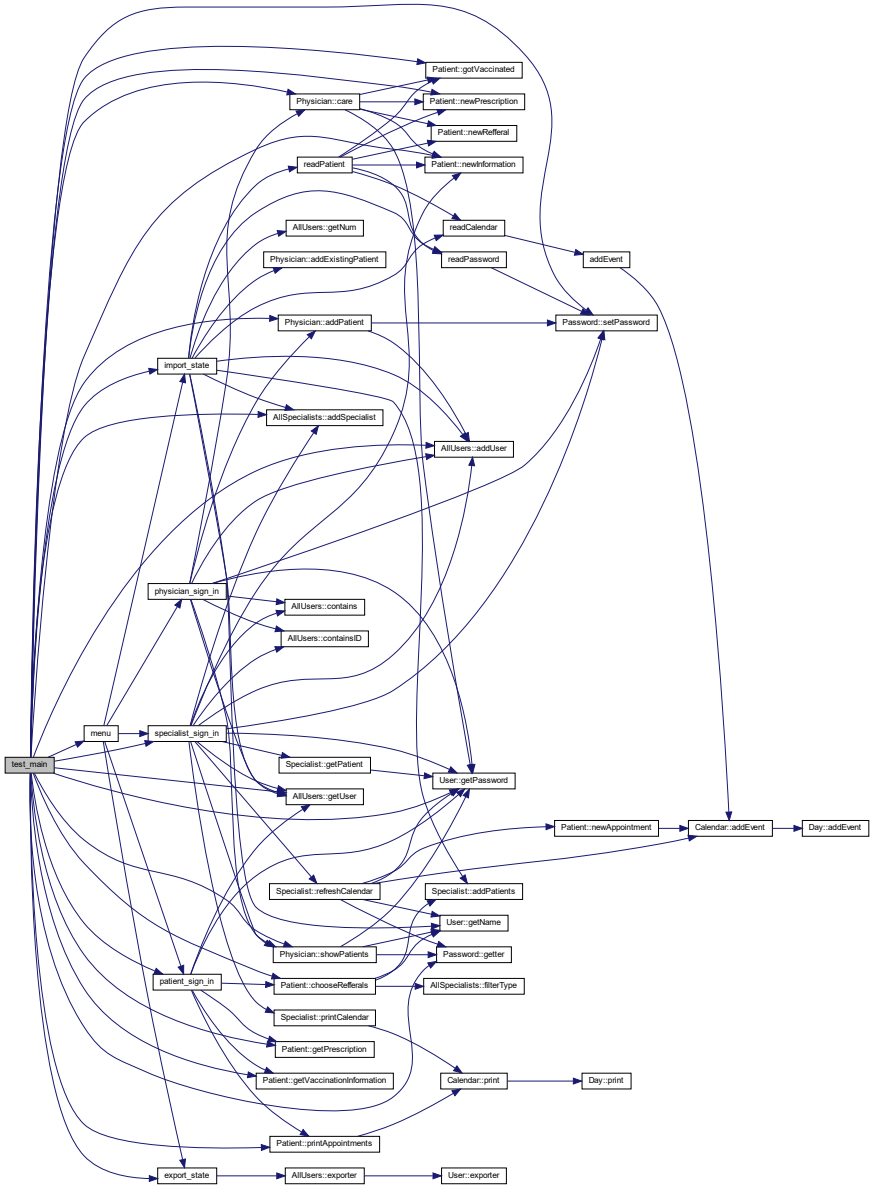
```
#define MEMTRACE
```

5.33.2. Függvények dokumentációja

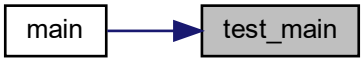
5.33.2.1. test_main()

```
void test_main ( )
```

A tesztelést megvalósító függvény. A függvény hívási gráfja:



A függvény hívó gráfja:



5.34. C:/Users/Lili/Desktop/Prog2/nhf/test.h

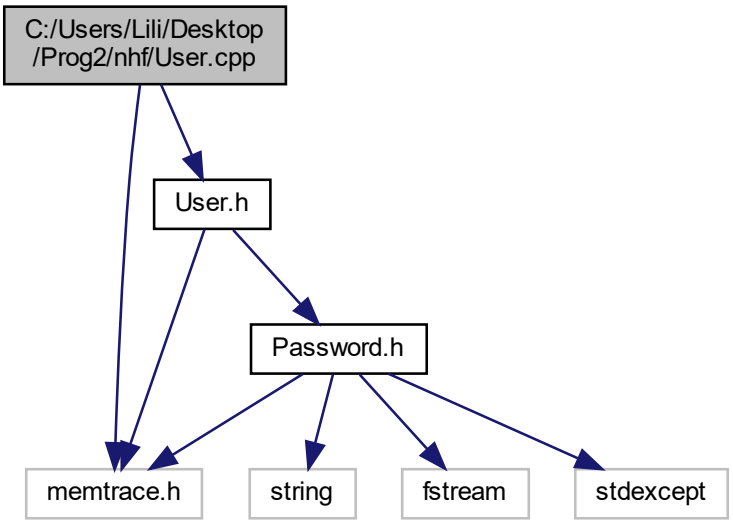
Ugrás a fájl dokumentációjához.

```
1
5 #ifndef NHF_TEST_H
6 #define NHF_TEST_H
7
8 #include <iostream>
9
10 #include "memtrace.h"
11
12 #include "AllUsers.h"
13 #include "Physician.h"
14 #include "Specialist.h"
15 #include "AllSpecialists.h"
16 #include "Patient.h"
17 #include "controller.h"
18 #include "menu.h"
19 #include "gtest_lite.h"
20
21 #define MEMTRACE
22
23 //a tesztelést végző függvény
24 void test_main();
25 #endif //NHF_TEST_H
```

5.35. C:/Users/Lili/Desktop/Prog2/nhf/User.cpp fájlreferencia

```
#include "memtrace.h"
#include "User.h"
```

Az User.cpp definíciós fájl függési gráfja:



Makródefiníciók

- #define MEMTRACE

5.35.1. Makródefiníciók dokumentációja

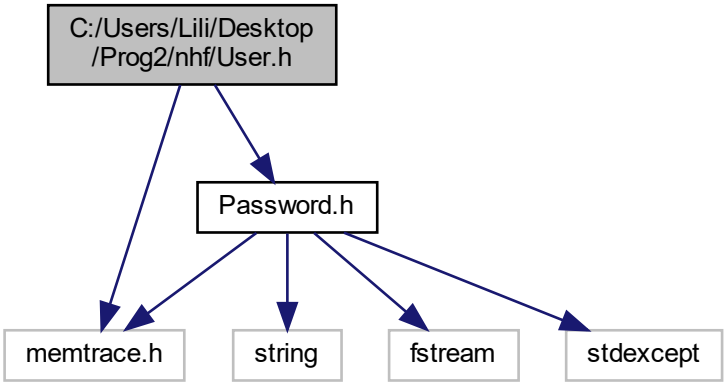
5.35.1.1. MEMTRACE

```
#define MEMTRACE
```

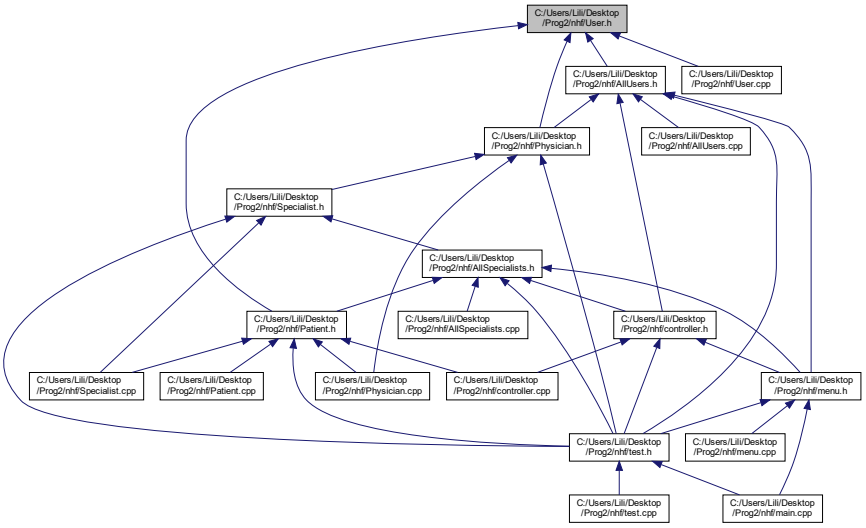
5.36. C:/Users/Lili/Desktop/Prog2/nhf/User.h fájlreferencia

```
#include "memtrace.h"
#include "Password.h"
```

Az User.h definíciós fájl függési gráfja:



Ez az ábra azt mutatja, hogy mely fájlok ágyazzák be közvetve vagy közvetlenül ezt a fájlt:



Osztályok

- class User

Makródefiníciók

- #define MEMTRACE

5.36.1. Makródefiníciók dokumentációja

5.36.1.1. MEMTRACE

```
#define MEMTRACE
```

5.37. C:/Users/Lili/Desktop/Prog2/nhf/User.h

Ugrás a fájl dokumentációjához.

```
1
5 #if !defined(_USER_H)
6 #define _USER_H
7 #define MEMTRACE
8
9 #include "memtrace.h"
10
11 #include "Password.h"
12
16 class User
17 {
18 protected:
19     std::string name;
20     Password passW;
21 public:
27     User(const std::string& n, Password p)
28     {
29         name=n;
30         passW=p;
31     }
36     Password getPassword();
37
41     virtual std::string exporter()=0;
42
47     std::string getName();
48
52     virtual ~User(){}
53 };
54
55 #endif  //_USER_H
```