Nagyházi

Generated by Doxygen 1.9.6

Chapter 1

Nagyházi feladat Prog2-ből:

Ez a féléves beadandó házim. A feladat pontos leírása `itt` található Sportegyesület néven.

1.1 Futtatás:

A fordítás után egy `nagyhazi` vagy `nagyhazi.exe` file fog keletkezni. Fordítani a következő féle képpen lehet:

- Windows alatt:
  `Makefile_WIN.cmd`
- Linux, unix alatt:
  `make`

1.2 Dokumentáció:

A feladato PDF dokumentációja a `docs/Docs.pdf` helyen érhető el, a HTML dokumentáció a `html/` mappában az `index.html` futtatásával tekinthető meg.

1.3 Specifikáció:

A feladathoz tartozó specifikáció a `spec/Spec.pdf`-ben található.

1.4 Állapot:

Kész. A feladat a Laborvezető elbírálására vár, utólagos javítások azután valószínűek.

1.5 Changelog:

- Heterogén kollekció inplementálása (*2023.05.01*)
  - A Class-okban
  - A Nyilvántartás API-ban
  - A Menüben
  - Dokumentációban, Specifikációban

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

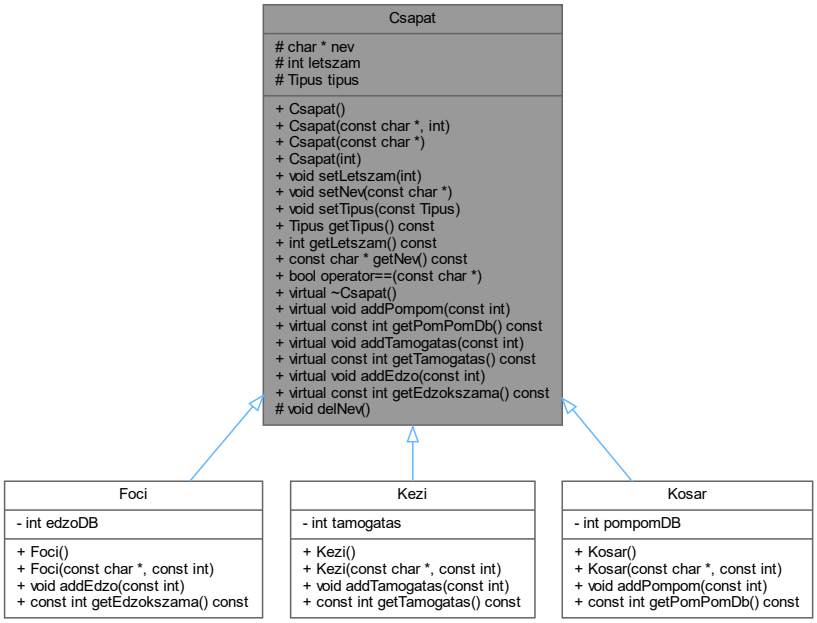
Class Documentation

5.1 Csapat Class Reference

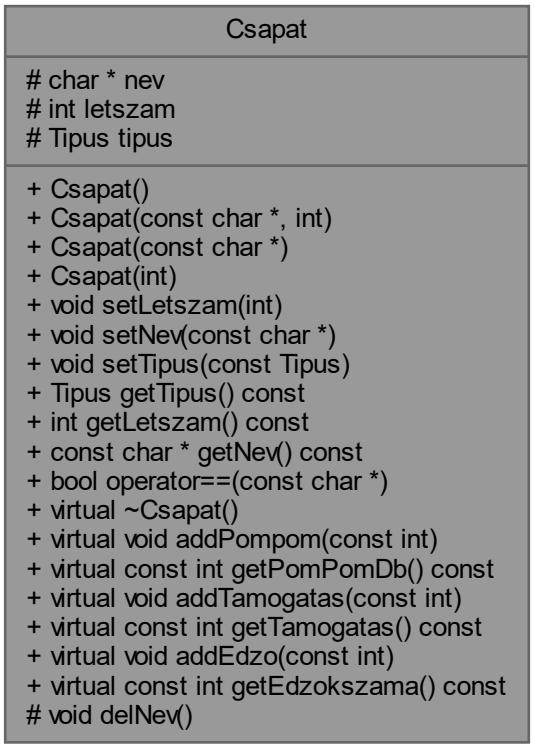
A csapatok szülőobjektuma. Ez írja je a csapatok közös viselkedését, tulajdonságait.

`#include <csapat.h>`

Inheritance diagram for Csapat:



Collaboration diagram for Csapat:



Public Member Functions

- Csapat ()

  Default konstruktor, amely létrehoz egy NINCS típusú, 0 létszámú, semmilyen nevű csapatot.
- Csapat (const char ∗, int)

  Név és létszámos konstruktor, amely létrehozza az adadtoknak megfelelő csapatot.
- Csapat (const char ∗)

  A csapat nevét létrehozó konstruktor, amely az a adoktnak megfelelő csapatot hoz létre, és a létszámot nullázza.
- Csapat (int)

  A csapat létszámot is létrehozó konstruktor, amely nem hoz létre csapatnevet.
- void setLetszam (int)

  A csapat létszámát átállító, beállító függvény.
- void setNev (const char ∗)

  A csapat nevét átállító, beállító függvény.
- void setTipus (const Tipus)

  A csapat típusát beállító függvény (többször átállítani nincs értelme, mert úgyis öröklődik és az örökös tulajdonságai mások).
- Tipus getTipus () const

  Viszaadja a csapat típusát a Tipus enum segítségével.
- int getLetszam () const

Visszaadja a csapat létszámát.

- const char * getNev () const

  Visszaadja a csapat nevét.
- bool operator== (const char *)

  Lehetővé teszi a csapatok közti gyors keresést a == operátor túlterhelésével.
- virtual ~Csapat ()

  Virtuális destruktor (mert öröklődik majd a class).
- virtual void addPompom (const int)

  Kosárcsapatra vonatkozó függvénypointer.
- virtual const int getPomPomDb () const

  Kosárcsapatra vonatkozó függvénypointer.
- virtual void addTamogatas (const int)

  Kézicsapatra vonatkozó függvénypointer.
- virtual const int getTamogatas () const

  Kézicsapatra vonatkozó függvénypointer.
- virtual void addEdzo (const int)

  Focicsapatokra vonatkozó függvénypointer.
- virtual const int getEdzokszama () const

  Focicsapatokra vonatkozó függvénypointer.

Protected Member Functions

- void delNev ()

  Segédfunkció, amely kitörli, felszabadítja a nevet, ha az nem üres.

Protected Attributes

- char * nev

  A csapat neve.
- int letszam

  A csapat létszáma.
- Tipus tipus

  A csapat típusa a Típus enum segítségével.

5.1.1 Detailed Description

A csapatok szülőobjektuma. Ez írja je a csapatok közös viselkedését, tulajdonságait.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Csapat() [1/4]

```
Csapat::Csapat ( )  [inline]
```

Default konstruktor, amely létrehoz egy NINCS típusú, 0 létszámú, semmilyen nevű csapatot.

5.1.2.2 Csapat() [2/4]

```
Csapat::Csapat (
            const char * p,
            int n )
```

Név és létszámos konstruktor, amely létrehozza az adadtoknak megfelelő csapatot.

Parameters

csapat_nev	ez a const char* paraméter a csapat neve lesz.
letszam	ez az int paraméter a csapat létszáma lesz.

5.1.2.3 Csapat() [3/4]

```
Csapat::Csapat (
            const char * p )
```

A csapat nevét létrehozó konstruktor, amely az a adoktnak megfelelő csapatot hoz létre, és a létszámot nullázza.

Parameters

csapat_nev	ez a const char* paraméter a csapat neve lesz.

5.1.2.4 Csapat() [4/4]

```
Csapat::Csapat (
            int n )
```

A csapat létszámot is létrehozó konstruktor, amely nem hoz létre csapatnevet.

Parameters

letszam	ez az int paraméter a csapatlétszám lesz.

5.1.2.5 ~Csapat()

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

Virtuális destruktor (mert öröklődik majd a class).

Here is the call graph for this function:

5.1.3 Member Function Documentation

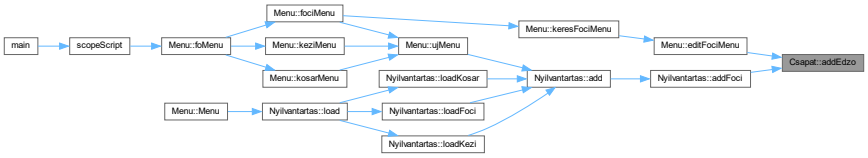
5.1.3.1 addEdzo()

```
virtual void Csapat::addEdzo (
            const int  )  [inline], [virtual]
```

Focicsapatokra vonatkozó függvénypointer.

Reimplemented in Foci.

Here is the caller graph for this function:



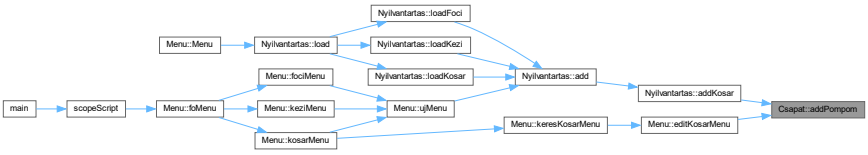
5.1.3.2 addPompom()

```
virtual void Csapat::addPompom (
            const int  )  [inline], [virtual]
```

Kosárcsapatra vonatkozó függvénypointer.

Reimplemented in Kosar.

Here is the caller graph for this function:



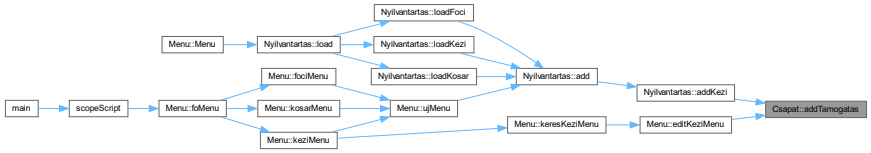
5.1.3.3 addTamogatas()

```
virtual void Csapat::addTamogatas (
            const int  )  [inline], [virtual]
```

Kézicsapatra vonatkozó függvénypointer.

Reimplemented in Kezi.

Here is the caller graph for this function:

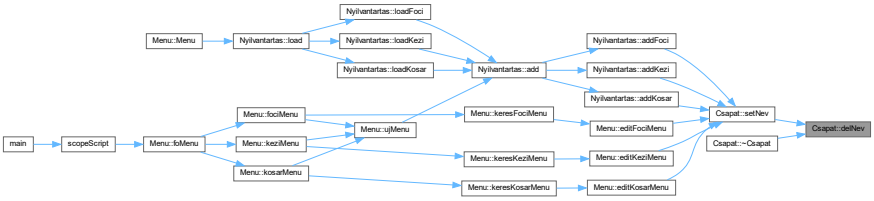


5.1.3.4 delNev()

```
void Csapat::delNev ( )  [inline], [protected]
```

Segédfunkció, amely kitörli, felszabadítja a nevet, ha az nem üres.

Here is the caller graph for this function:



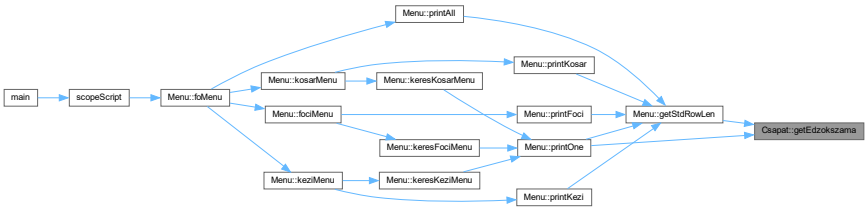
5.1.3.5 getEdzokszama()

```
virtual const int Csapat::getEdzokszama ( ) const  [inline], [virtual]
```

Focicsapatokra vonatkozó függvénypointer.

Reimplemented in Foci.

Here is the caller graph for this function:



5.1.3.6 getLetszam()

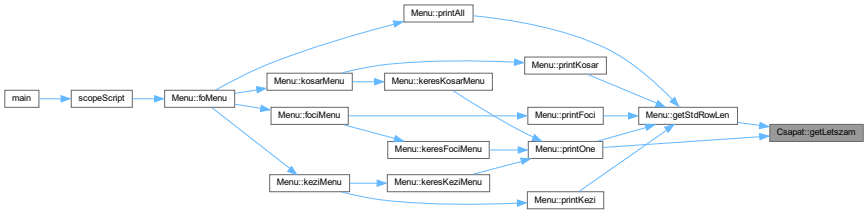
```
int Csapat::getLetszam ( ) const
```

Visszaadja a csapat létszámát.

Returns

A csapat létszáma, int.

Here is the caller graph for this function:



5.1.3.7 getNev()

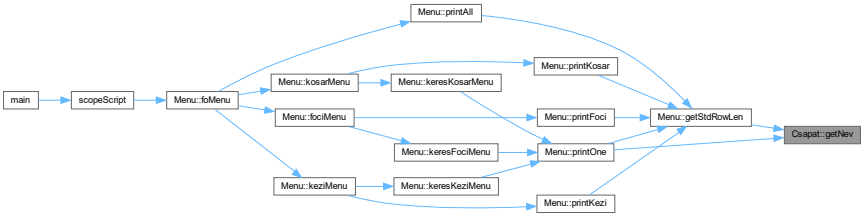
```
const char * Csapat::getNev ( ) const
```

Visszaadja a csapat nevét.

Returns

A csapat neve, const char*.

Here is the caller graph for this function:



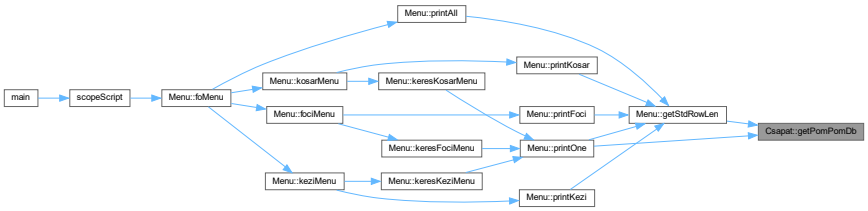
5.1.3.8 getPomPomDb()

```
virtual const int Csapat::getPomPomDb ( ) const  [inline], [virtual]
```

Kosárcsapatra vonatkozó függvénypointer.

Reimplemented in Kosar.

Here is the caller graph for this function:



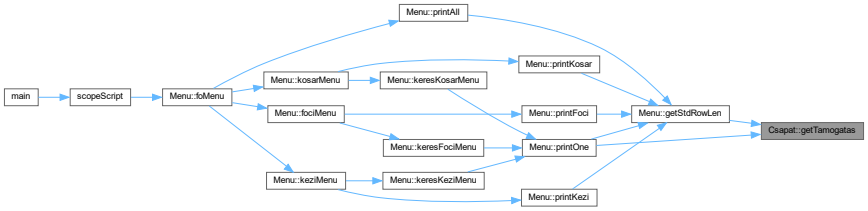
5.1.3.9 getTamogatas()

```
virtual const int Csapat::getTamogatas ( ) const  [inline], [virtual]
```

Kézicsapatra vonatkozó függvénypointer.

Reimplemented in Kezi.

Here is the caller graph for this function:



5.1.3.10 getTipus()

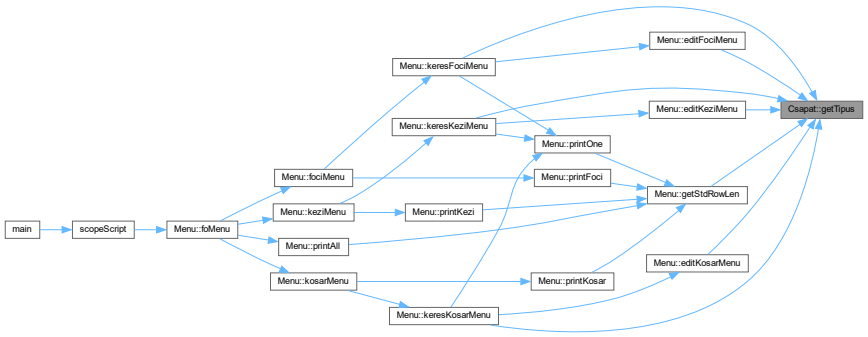
```
Tipus Csapat::getTipus ( ) const
```

Viszaadja a csapat típusát a Tipus enum segítségével.

Returns

A csapat típusa a Tipus enum-ban.

Here is the caller graph for this function:



5.1.3.11 operator==()

```
bool Csapat::operator== (
            const char * str )
```

Lehetővé teszi a csapatok közti gyors keresést a == operátor túlterhelésével.

Megvizsgálja, hogy az adott névvel egyezik-e a csapat neve és ennek megfelelő

bool (igaz/hamis) értéket dob vissza.

Parameters

keresett_csapat_nev	A keresett csapatnév const char∗ paraméter.

Returns

Egy igaz hamis érték, hogy egyezik-e a csapatnév.

5.1.3.12 setLetszam()

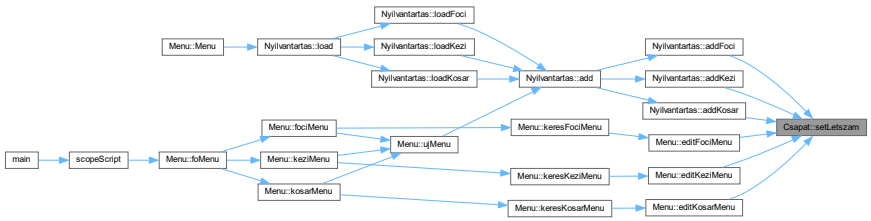
```
void Csapat::setLetszam (
            int n )
```

A csapat létszámát átállító, beállító függvény.

Parameters

uj_letszam	Ez az int parameter lesz az új létszáma a csapatnak.

Here is the caller graph for this function:



5.1.3.13 setNev()

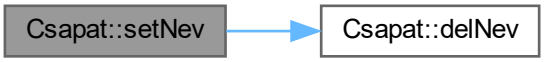
```
void Csapat::setNev (
            const char * p )
```

A csapat nevét átállító, beállító függvény.

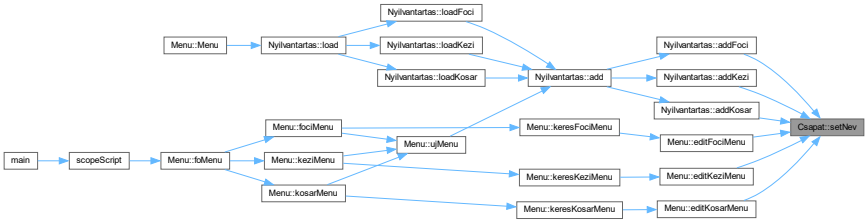
Parameters

uj_nev	Ez a const char∗ paraméter lesz a csapat új neve.

Here is the call graph for this function:



Here is the caller graph for this function:



5.1.3.14 setTipus()

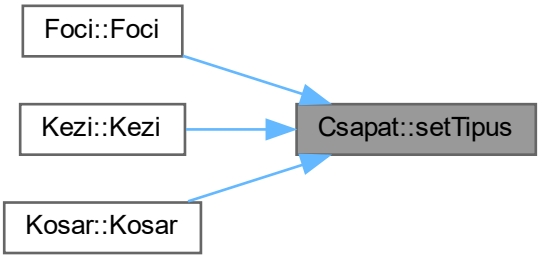
```
void Csapat::setTipus (
            const Tipus t )
```

A csapat típusát beállító függvény (többször átállítani nincs értelme, mert úgyis öröklődik és az örökös tulajdonságai mások).

Parameters

uj_tipus	Ez a Tipus parameter a csapat típusát adja meg a Tipus enum segítségével.

Here is the caller graph for this function:



5.1.4 Member Data Documentation

5.1.4.1 letszam

```
int Csapat::letszam  [protected]
```

A csapat létszáma.

5.1.4.2 nev

```
char* Csapat::nev  [protected]
```

A csapat neve.

5.1.4.3 tipus

```
Tipus Csapat::tipus  [protected]
```

A csapat típusa a Típus enum segítségével.

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

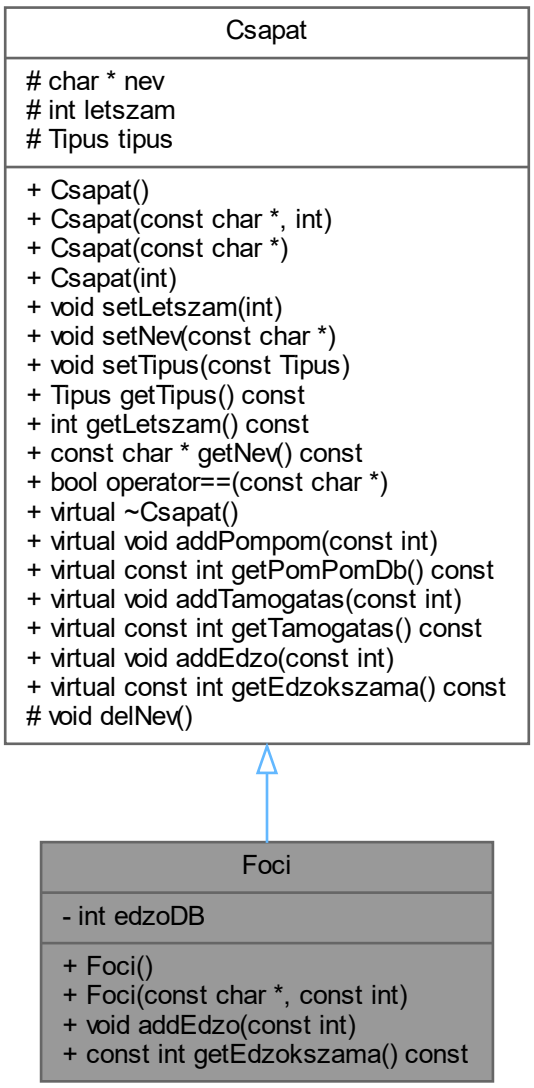
- csapat.h
- csapat.cpp

5.2 Foci Class Reference

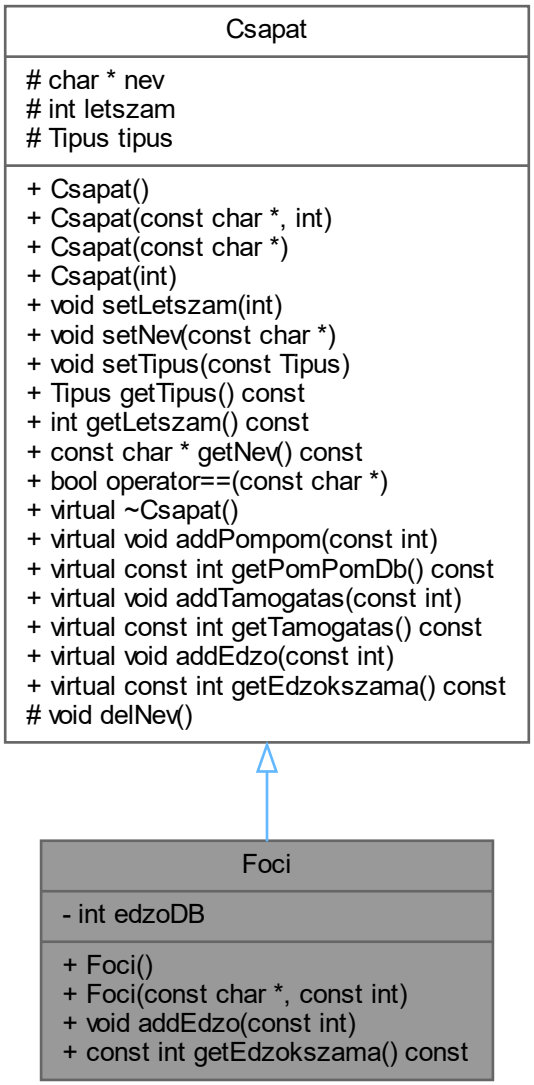
A focicsapat objektumja, amely örökli a Csapat objektum tulajdonságait.

`#include <foci.h>`

Inheritance diagram for Foci:



Collaboration diagram for Foci:



Public Member Functions

- Foci ()

  A default konstruktor, amely a Csapat default konstruktorát használja.
- Foci (const char ∗, const int)

  A konstruktor, amely létrehoz az adatoknak megfelelő Csapat objektumot.
- void addEdzo (const int)

  Sok darab új edzot ad hozzá a csapathoz. Növeli az edzoDB countert a megfelelő számmal.
- const int getEdzokszama () const

  Visszaadja az edzők számát.

Public Member Functions inherited from Csapat

- Csapat ()

  Default konstruktor, amely létrehoz egy NINCS típusú, 0 létszámú, semmilyen nevű csapatot.
- Csapat (const char *, int)

  Név és létszámos konstruktor, amely létrehozza az adadtoknak megfelelő csapatot.
- Csapat (const char *)

  A csapat nevét létrehozó konstruktor, amely az a adoktnak megfelelő csapatot hoz létre, és a létszámot nullázza.
- Csapat (int)

  A csapat létszámot is létrehozó konstruktor, amely nem hoz létre csapatnevet.
- void setLetszam (int)

  A csapat létszámát átállító, beállító függvény.
- void setNev (const char *)

  A csapat nevét átállító, beállító függvény.
- void setTipus (const Tipus)

  A csapat típusát beállító függvény (többször átállítani nincs értelme, mert úgyis öröklődik és az örökös tulajdonságai mások).
- Tipus getTipus () const

  Viszaadja a csapat típusát a Tipus enum segítségével.
- int getLetszam () const

  Visszaadja a csapat létszámát.
- const char * getNev () const

  Visszaadja a csapat nevét.
- bool operator== (const char *)

  Lehetővé teszi a csapatok közti gyors keresést a == operátor túlterhelésével.
- virtual ~Csapat ()

  Virtuális destruktor (mert öröklődik majd a class).
- virtual void addPompom (const int)

  Kosárcsapatra vonatkozó függvénypointer.
- virtual const int getPomPomDb () const

  Kosárcsapatra vonatkozó függvénypointer.
- virtual void addTamogatas (const int)

  Kézicsapatra vonatkozó függvénypointer.
- virtual const int getTamogatas () const

  Kézicsapatra vonatkozó függvénypointer.
- virtual void addEdzo (const int)

  Focicsapatokra vonatkozó függvénypointer.
- virtual const int getEdzokszama () const

  Focicsapatokra vonatkozó függvénypointer.

Private Attributes

- int edzoDB

  A focicsapatra specifikus Edzőket számláló változó.

Additional Inherited Members

Protected Member Functions inherited from Csapat

- void delNev ()

  Segédfunkció, amely kitörli, felszabadítja a nevet, ha az nem üres.

Protected Attributes inherited from Csapat

- char * nev

  A csapat neve.
- int letszam

  A csapat létszáma.
- Tipus tipus

  A csapat típusa a Típus enum segítségével.

5.2.1 Detailed Description

A focicsapat objektumja, amely örökli a Csapat objektum tulajdonságait.

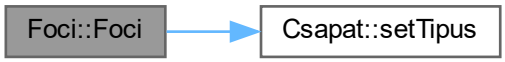
5.2.2 Constructor & Destructor Documentation

5.2.2.1 Foci() [1/2]

```
Foci::Foci ( )
```

A default konstruktor, amely a Csapat default konstruktorát használja.

tehát létrehoz egy üres csapatot (csak a típusát FOCI-ra rakja). Here is the call graph for this function:



5.2.2.2 Foci() [2/2]

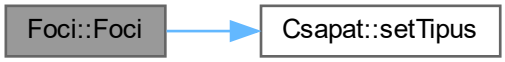
```
Foci::Foci (
            const char * p = "",
            const int n = 0 )
```

A konstruktor, amely létrehoz az adatoknak megfelelő Csapat objektumot.

Parameters

csapat_nev	a csapat beállítandó neve.
letszam	a csapat beállítandó létszáma.

Here is the call graph for this function:



5.2.3 Member Function Documentation

5.2.3.1 addEdzo()

```
void Foci::addEdzo (
            const int n )  [virtual]
```

Sok darab új edzot ad hozzá a csapathoz. Növeli az edzoDB countert a megfelelő számmal.

Parameters

edzok_szama	ennyivel növeli az edzoDB countert.

Reimplemented from Csapat.

5.2.3.2 getEdzokszama()

```
const int Foci::getEdzokszama ( ) const  [virtual]
```

Visszaadja az edzők számát.

Returns

Az edzők száma, int.

Reimplemented from Csapat.

5.2.4 Member Data Documentation

5.2.4.1 edzoDB

```
int Foci::edzoDB  [private]
```

A focicsapatra specifikus Edzőket számláló változó.

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

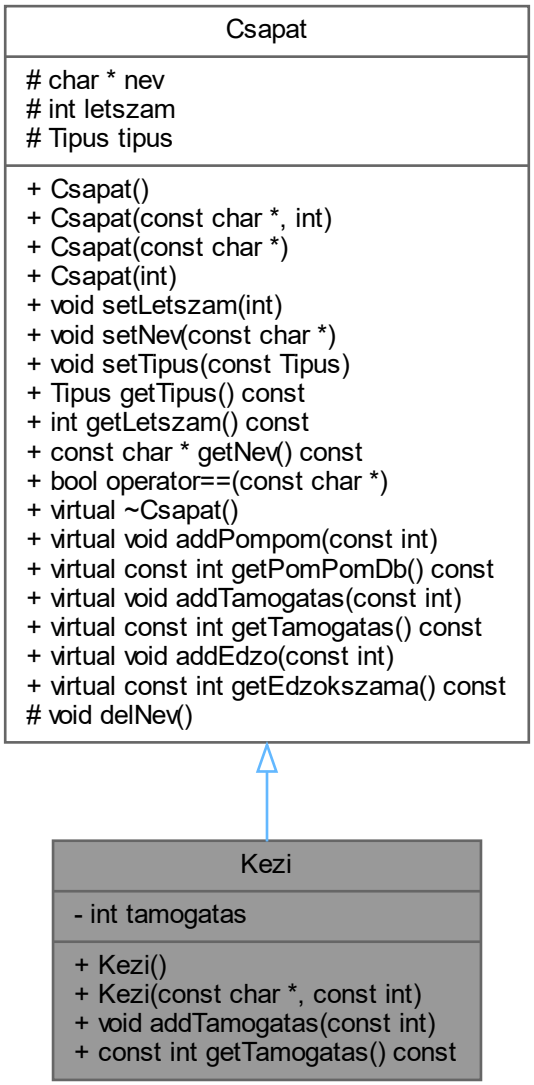
- foci.h
- foci.cpp

5.3 Kezi Class Reference

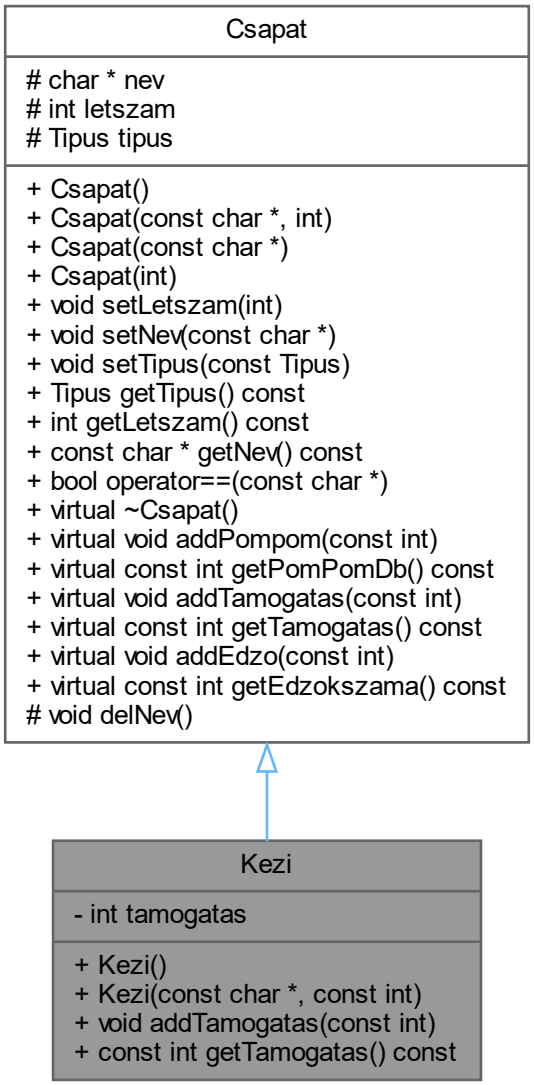
A kézilabda csapat objektumja, amely örökli a Csapat objektum tulajdonságait.

```
#include <kezi.h>
```

Inheritance diagram for Kezi:



Collaboration diagram for Kezi:



Public Member Functions

- Kezi ()

  A default konstruktor, amely a Csapat default konstruktorát hívja, csak a.
- Kezi (const char ∗, const int)

  A konstruktor, amely a megadott adatoknake megfelelően állítja be a csapatot.
- void addTamogatas (const int)

  A megadott számmal növeli a tamogatas változót.
- const int getTamogatas () const

  Visszaadja a támogatások számát.

Public Member Functions inherited from Csapat

- Csapat ()

  Default konstruktor, amely létrehoz egy NINCS típusú, 0 létszámú, semmilyen nevű csapatot.
- Csapat (const char *, int)

  Név és létszámos konstruktor, amely létrehozza az adadtoknak megfelelő csapatot.
- Csapat (const char *)

  A csapat nevét létrehozó konstruktor, amely az a adoktnak megfelelő csapatot hoz létre, és a létszámot nullázza.
- Csapat (int)

  A csapat létszámot is létrehozó konstruktor, amely nem hoz létre csapatnevet.
- void setLetszam (int)

  A csapat létszámát átállító, beállító függvény.
- void setNev (const char *)

  A csapat nevét átállító, beállító függvény.
- void setTipus (const Tipus)

  A csapat típusát beállító függvény (többször átállítani nincs értelme, mert úgyis öröklődik és az örökös tulajdonságai mások).
- Tipus getTipus () const

  Viszaadja a csapat típusát a Tipus enum segítségével.
- int getLetszam () const

  Visszaadja a csapat létszámát.
- const char * getNev () const

  Visszaadja a csapat nevét.
- bool operator== (const char *)

  Lehetővé teszi a csapatok közti gyors keresést a == operátor túlterhelésével.
- virtual ~Csapat ()

  Virtuális destruktor (mert öröklődik majd a class).
- virtual void addPompom (const int)

  Kosárcsapatra vonatkozó függvénypointer.
- virtual const int getPomPomDb () const

  Kosárcsapatra vonatkozó függvénypointer.
- virtual void addTamogatas (const int)

  Kézicsapatra vonatkozó függvénypointer.
- virtual const int getTamogatas () const

  Kézicsapatra vonatkozó függvénypointer.
- virtual void addEdzo (const int)

  Focicsapatokra vonatkozó függvénypointer.
- virtual const int getEdzokszama () const

  Focicsapatokra vonatkozó függvénypointer.

Private Attributes

- int tamogatas

  Az éves támogatásokat tároló int.

Additional Inherited Members

Protected Member Functions inherited from Csapat

- void delNev ()

  Segédfunkció, amely kitörli, felszabadítja a nevet, ha az nem üres.

Protected Attributes inherited from Csapat

- char * nev

  A csapat neve.
- int letszam

  A csapat létszáma.
- Tipus tipus

  A csapat típusa a Típus enum segítségével.

5.3.1 Detailed Description

A kézilabda csapat objektumja, amely örökli a Csapat objektum tulajdonságait.

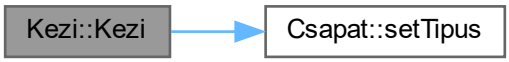
5.3.2 Constructor & Destructor Documentation

5.3.2.1 Kezi() [1/2]

```
Kezi::Kezi ( )
```

A default konstruktor, amely a Csapat default konstruktorát hívja, csak a.

típust KEZI-re rakja a Típus enum segítségével. Here is the call graph for this function:



5.3.2.2 Kezi() [2/2]

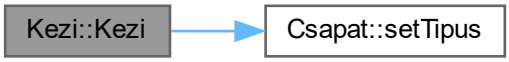
```
Kezi::Kezi (
            const char * p,
            const int n )
```

A konstruktor, amely a megadott adatoknake megfelelően állítja be a csapatot.

Parameters

uj_csapat_nev	a csapat uj neve.
letszam	a csapat létszáma.

Here is the call graph for this function:



5.3.3 Member Function Documentation

5.3.3.1 addTamogatas()

```
void Kezi::addTamogatas (
            const int t )  [virtual]
```

A megadott számmal növeli a tamogatas változót.

Parameters

uj_tamogatas	ennyivel növeli a támogatás változót.

Reimplemented from Csapat.

5.3.3.2 getTamogatas()

```
const int Kezi::getTamogatas ( ) const  [virtual]
```

Visszaadja a támogatások számát.

Returns

A támogatások száma.

Reimplemented from Csapat.

5.3.4 Member Data Documentation

5.3.4.1 tamogatas

```
int Kezi::tamogatas  [private]
```

Az éves támogatásokat tároló int.

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

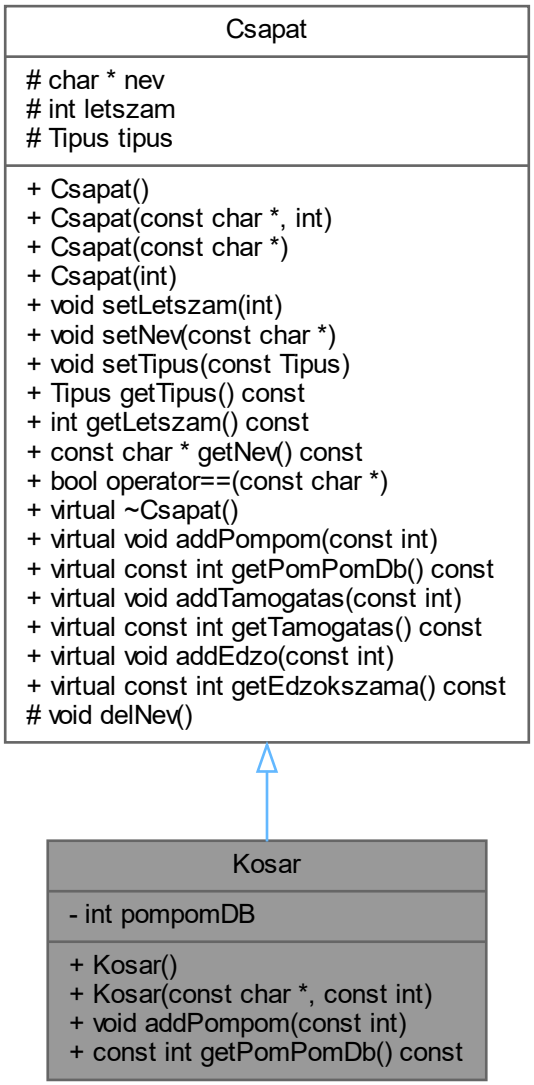
- kezi.h
- kezi.cpp

5.4 Kosar Class Reference

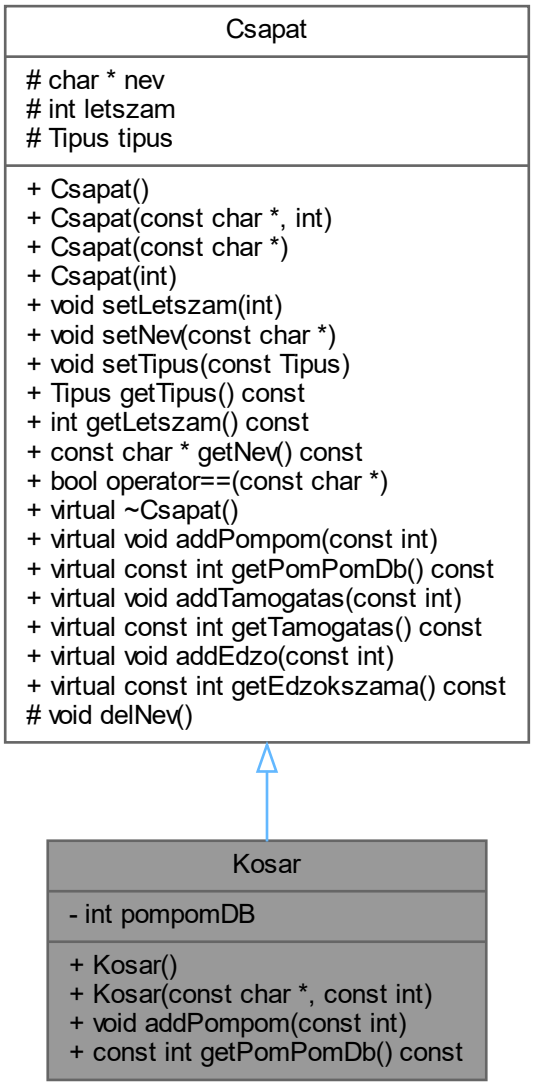
A kosárlabda csapat objektumja, amely örökli a Csapat objektum tulajdonságait.

```
#include <kosar.h>
```

Inheritance diagram for Kosar:



Collaboration diagram for Kosar:



Public Member Functions

- Kosar ()

  A default konstruktor, amely a Csapat default konstruktorát hívja, csak a.
- Kosar (const char ∗, const int)

  Az adatoknak megfelelő csapatot hoz létre.
- void addPompom (const int)

  Az adatoknak megfelelő mennyiségű pompomlányt ad a csapathoz.
- const int getPomPomDb () const

  Visszaadja a pompomlányok számát.

Public Member Functions inherited from Csapat

- Csapat ()

  Default konstruktor, amely létrehoz egy NINCS típusú, 0 létszámú, semmilyen nevű csapatot.
- Csapat (const char *, int)

  Név és létszámos konstruktor, amely létrehozza az adadtoknak megfelelő csapatot.
- Csapat (const char *)

  A csapat nevét létrehozó konstruktor, amely az a adoktnak megfelelő csapatot hoz létre, és a létszámot nullázza.
- Csapat (int)

  A csapat létszámot is létrehozó konstruktor, amely nem hoz létre csapatnevet.
- void setLetszam (int)

  A csapat létszámát átállító, beállító függvény.
- void setNev (const char *)

  A csapat nevét átállító, beállító függvény.
- void setTipus (const Tipus)

  A csapat típusát beállító függvény (többször átállítani nincs értelme, mert úgyis öröklődik és az örökös tulajdonságai mások).
- Tipus getTipus () const

  Viszaadja a csapat típusát a Tipus enum segítségével.
- int getLetszam () const

  Visszaadja a csapat létszámát.
- const char * getNev () const

  Visszaadja a csapat nevét.
- bool operator== (const char *)

  Lehetővé teszi a csapatok közti gyors keresést a == operátor túlterhelésével.
- virtual ~Csapat ()

  Virtuális destruktor (mert öröklődik majd a class).
- virtual void addPompom (const int)

  Kosárcsapatra vonatkozó függvénypointer.
- virtual const int getPomPomDb () const

  Kosárcsapatra vonatkozó függvénypointer.
- virtual void addTamogatas (const int)

  Kézicsapatra vonatkozó függvénypointer.
- virtual const int getTamogatas () const

  Kézicsapatra vonatkozó függvénypointer.
- virtual void addEdzo (const int)

  Focicsapatokra vonatkozó függvénypointer.
- virtual const int getEdzokszama () const

  Focicsapatokra vonatkozó függvénypointer.

Private Attributes

- int pompomDB

  A pompomlányok száma.

Additional Inherited Members

Protected Member Functions inherited from Csapat

- void delNev ()

  Segédfunkció, amely kitörli, felszabadítja a nevet, ha az nem üres.

Protected Attributes inherited from Csapat

- char * nev

  A csapat neve.
- int letszam

  A csapat létszáma.
- Tipus tipus

  A csapat típusa a Típus enum segítségével.

5.4.1 Detailed Description

A kosárlabda csapat objektumja, amely örökli a Csapat objektum tulajdonságait.

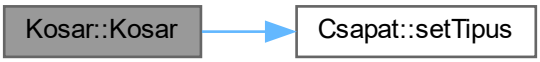
5.4.2 Constructor & Destructor Documentation

5.4.2.1 Kosar() [1/2]

```
Kosar::Kosar ( )
```

A default konstruktor, amely a Csapat default konstruktorát hívja, csak a.

típust átállítja KOSAR-ra a Tipus enum segítségével. Here is the call graph for this function:



5.4.2.2 Kosar() [2/2]

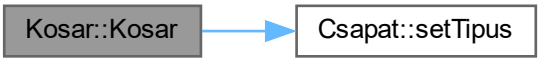
```
Kosar::Kosar (
            const char * p = "",
            const int n = 0 )
```

Az adatoknak megfelelő csapatot hoz létre.

Parameters

csapatnev	az uj csapatnév.
letszam	az új létszám.

Here is the call graph for this function:



5.4.3 Member Function Documentation

5.4.3.1 addPompom()

```
void Kosar::addPompom (
            const int n )  [virtual]
```

Az adatoknak megfelelő mennyiségű pompomlányt ad a csapathoz.

Parameters

darabSzam	ennyi pompomlány lesz a csapatban a csapathoz.

Reimplemented from Csapat.

5.4.3.2 getPomPomDb()

```
const int Kosar::getPomPomDb ( ) const  [virtual]
```

Visszaadja a pompomlányok számát.

Returns

A pompomlányok száma.

Reimplemented from Csapat.

5.4.4 Member Data Documentation

5.4.4.1 pompomDB

```
int Kosar::pompomDB  [private]
```

A pompomlányok száma.

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

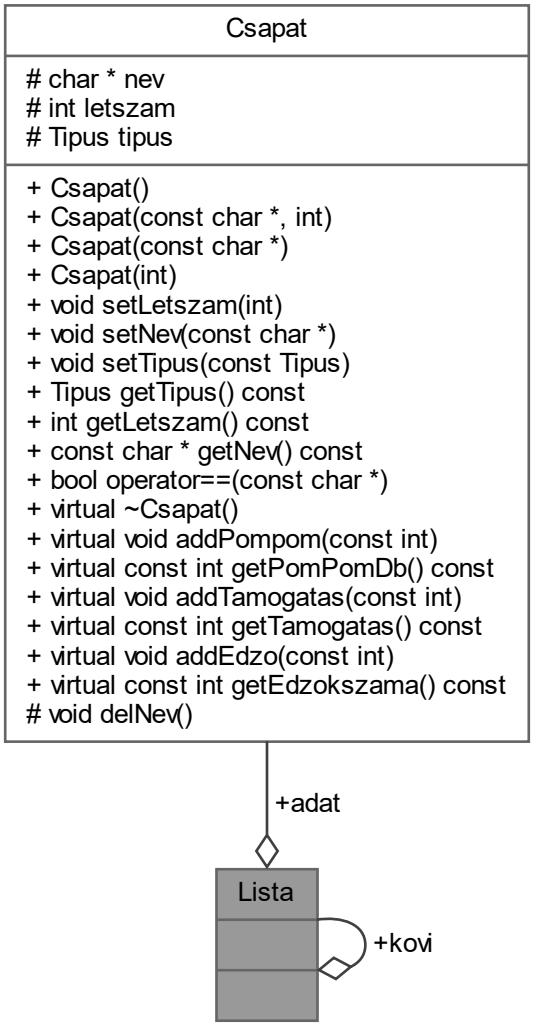
- kosar.h
- kosar.cpp

5.5 Lista Struct Reference

Láncolt listaelem.

```
#include <nyilvantartas.h>
```

Collaboration diagram for Lista:



Public Attributes

- Csapat * adat
- Lista * kovi

5.5.1 Detailed Description

Láncolt listaelem.

5.5.2 Member Data Documentation

5.5.2.1 adat

```
Csapat* Lista::adat
```

5.5.2.2 kovi

```
Lista* Lista::kovi
```

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

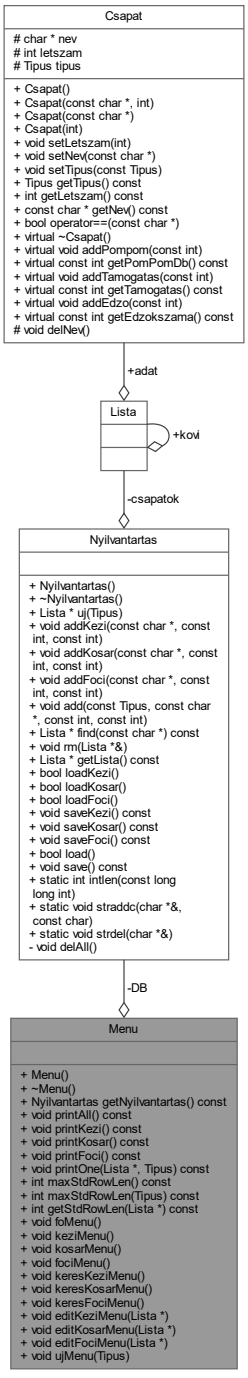
- nyilvantartas.h

5.6 Menu Class Reference

A futó programot irányító menürendszer objektuma.

```
#include <menu.h>
```

Collaboration diagram for Menu:



Public Member Functions

- Menu ()

  Default konstruktor, betölti a nyilvántartás adatait fileból.
- ~Menu ()

  Destruktor, lementi a nyilvántartás adatait fileokba.
- Nyilvantartas getNyilvantartas () const

Getter. Visszaadja az egész nyilvántartás osztályt (Debug célokra főképp, mivel nincs értelme a classal kommunikálni kívülről.)

- void printAll () const

  Kilistázza megformázva az összes adatot.
- void printKezi () const

  Kilistázza megformázva a Kézilabda (Kezi) csapatokat.
- void printKosar () const

  Kilistázza megformázva a Kárlabda (Kosar) csapatokat.
- void printFoci () const

  Kilistázza megformázva a Focilabda (Foci) csapatokat.
- void printOne (Lista ∗, Tipus) const

  Egy listaelemet ír ki megformázva.
- int maxStdRowLen () const

  Kiszámolja a leghoszabb sor hosszát a nyilvántartásban. Erre a TAB-ok és a kinézet miatt van szügség.
- int maxStdRowLen (Tipus) const

  Kiszámolja, hogy a nyílvántartás típus szerinti láncolt listájában mennyi a leghoszabb sor. Erre a TAB-ok és a kinézet miatt van szüggség.
- int getStdRowLen (Lista ∗) const

  Egy adott listaelem sorának hosszát adja vissza. Design felhasználási céllal.
- void foMenu ()

  Főmenü. Innen indul minden. Ez tujajdonképpen az entrypoint, ahonnan a class.
- void keziMenu ()

  A kézilabda csapatokkal foglalkozó almenü. Innen lehet kézi sprecifikus dolgokat csinálni.
- void kosarMenu ()

  A kosárlabda csapatokkal foglalkozó almenü. Innen lehet kosár sprecifikus dolgokat csinálni.
- void fociMenu ()

  A focicsapatokkal foglalkozó almenü. Innen lehet foci sprecifikus dolgokat csinálni.
- void keresKeziMenu ()

  Egy kézilabda csapat keresési menüje.
- void keresKosarMenu ()

  Egy kosárlabda csapat keresési menüje.
- void keresFociMenu ()

  Egy focilabda csapat keresési menüje.
- void editKeziMenu (Lista ∗)

  A láncolt lista egy elemét módosító almenü.
- void editKosarMenu (Lista ∗)

  A láncolt lista egy elemét módosító almenü.
- void editFociMenu (Lista ∗)

  A láncolt lista egy elemét módosító almenü.
- void ujMenu (Tipus)

  Egy új tipus típusú csapat lérehozására létező almenü.

Private Attributes

- Nyilvantartas DB

  A nyilvántartás adatbázis.

5.6.1 Detailed Description

A futó programot irányító menürendszer objektuma.

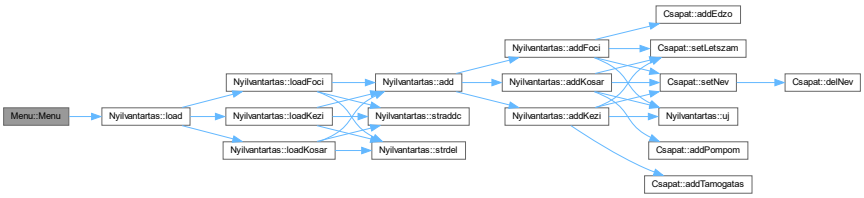
5.6.2 Constructor & Destructor Documentation

5.6.2.1 Menu()

```
Menu::Menu ( )
```

Default konstruktor, betölti a nyilvántartás adatait fileból.

Here is the call graph for this function:



5.6.2.2 ~Menu()

```
Menu::~Menu ( )
```

Destruktor, lementi a nyilvántartás adatait fileokba.

Here is the call graph for this function:

5.6.3 Member Function Documentation

5.6.3.1 editFociMenu()

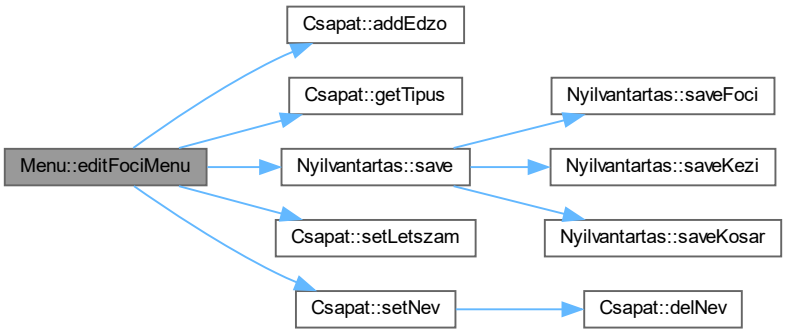
```
void Menu::editFociMenu (
            Lista * p )
```

A láncolt lista egy elemét módosító almenü.

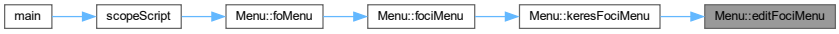
Parameters

listaelem	Ennek módosításában segít a menü.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.2 editKeziMenu()

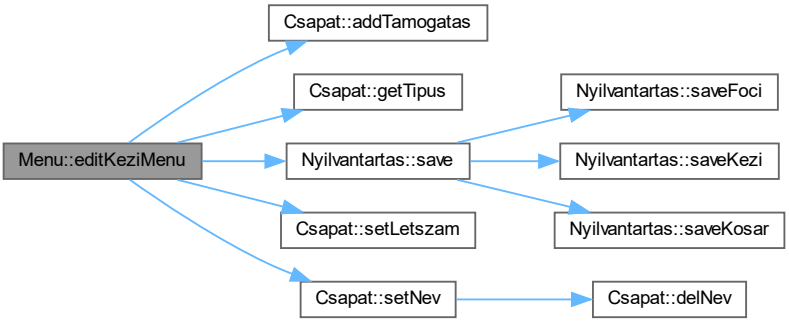
```
void Menu::editKeziMenu (
            Lista * p )
```

A láncolt lista egy elemét módosító almenü.

Parameters

listaelem	Ennek módosításában segít a menü.

Here is the call graph for this function:



Here is the caller graph for this function:

5.6.3.3 editKosarMenu()

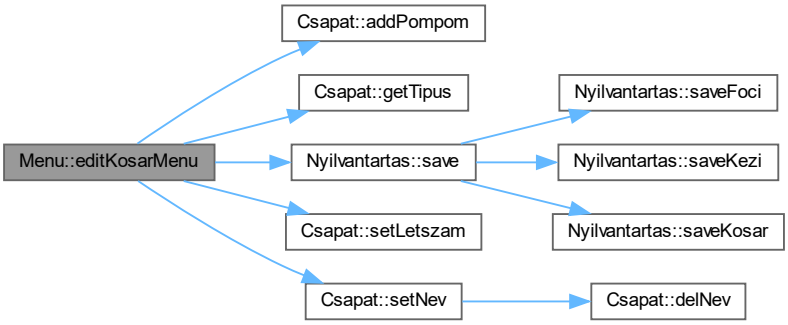
```
void Menu::editKosarMenu (
            Lista * p )
```

A láncolt lista egy elemét módosító almenü.

Parameters

listaelem	Ennek módosításában segít a menü.

Here is the call graph for this function:



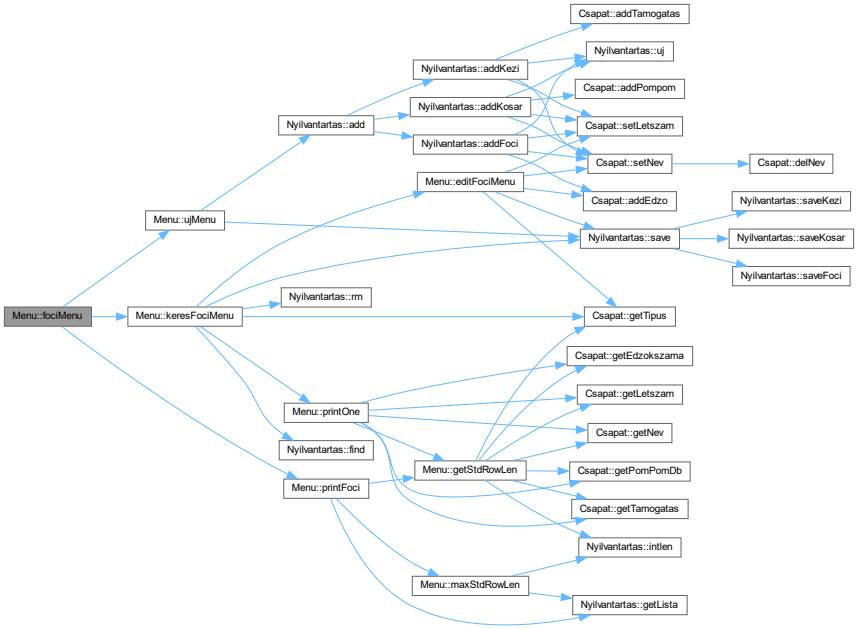
Here is the caller graph for this function:

5.6.3.4 fociMenu()

```
void Menu::fociMenu ( )
```

A focicsapatokkal foglalkozó almenü. Innen lehet foci sprecifikus dolgokat csinálni.

Here is the call graph for this function:



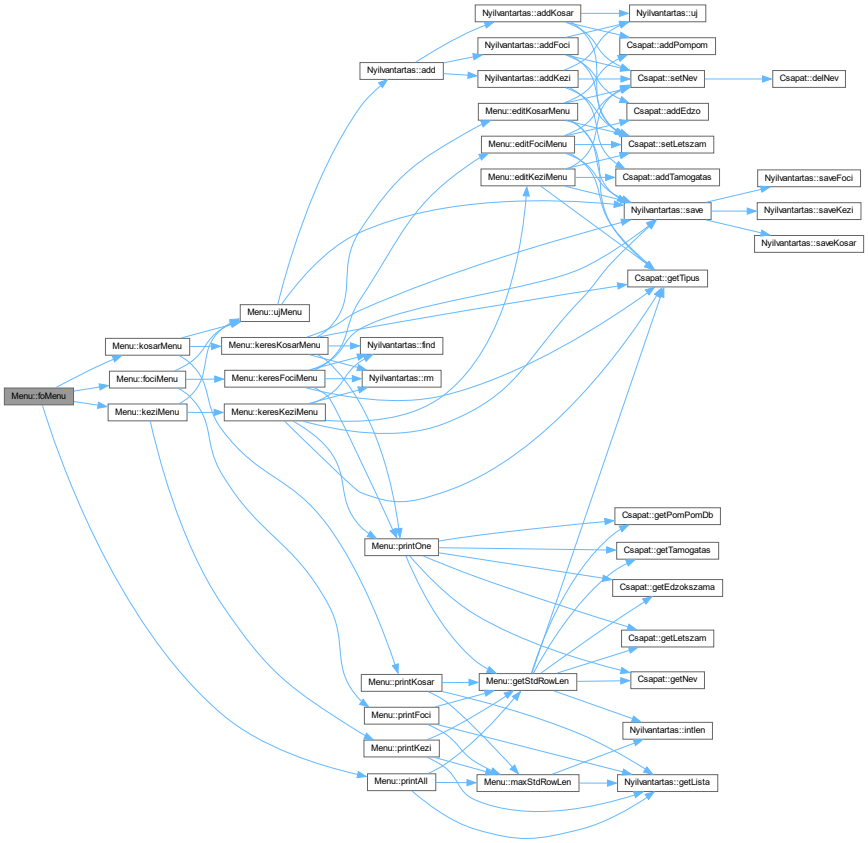
Here is the caller graph for this function:

5.6.3.5 foMenu()

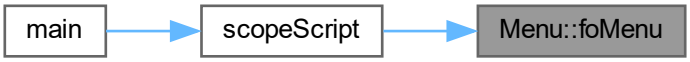
```
void Menu::foMenu ( )
```

Főmenü. Innen indul minden. Ez tujajdonképpen az entrypoint, ahonnan a class.

átveszi az irányítást, és automata menürendszerként üzemel. Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.6 getNyilvantartas()

```
Nyilvantartas Menu::getNyilvantartas ( ) const  [inline]
```

Getter. Visszaadja az egész nyilvántartás osztályt (Debug célokra főképp, mivel nincs értelme a classal kommunikálni kívülről.)

Returns

Nyilvántartás adatbázis.

5.6.3.7 getStdRowLen()

```
int Menu::getStdRowLen (
            Lista * l ) const
```

Egy adott listaelem sorának hosszát adja vissza. Design felhasználási céllal.

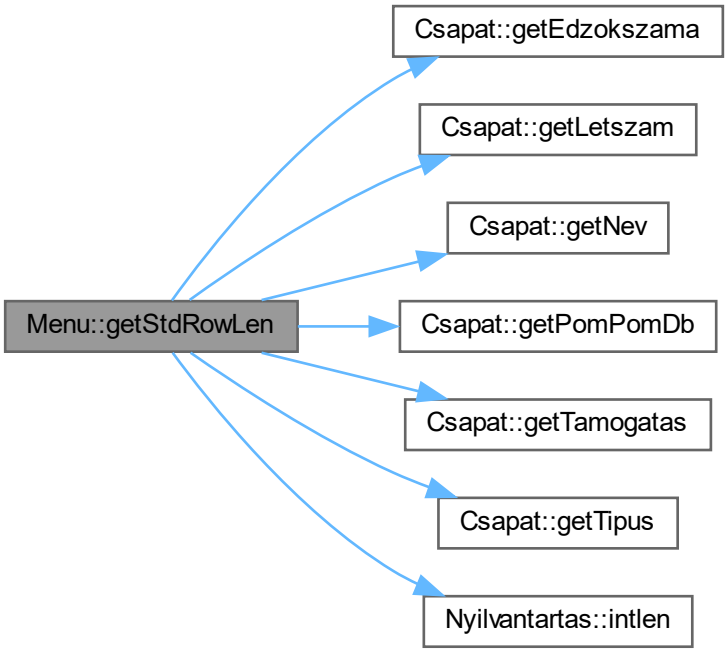
Parameters

listaelem	Ennek a sorhosszára vagyunk kíváncsiak.

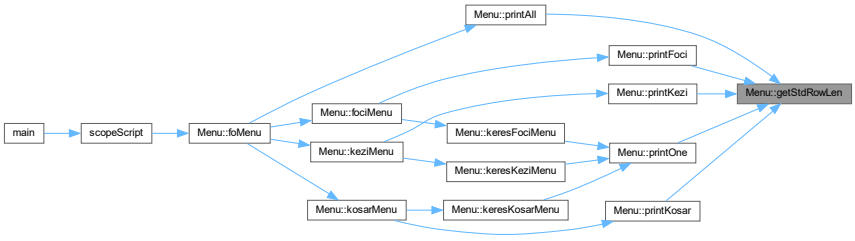
Returns

A listaelem soránk hossza.

Here is the call graph for this function:



Here is the caller graph for this function:

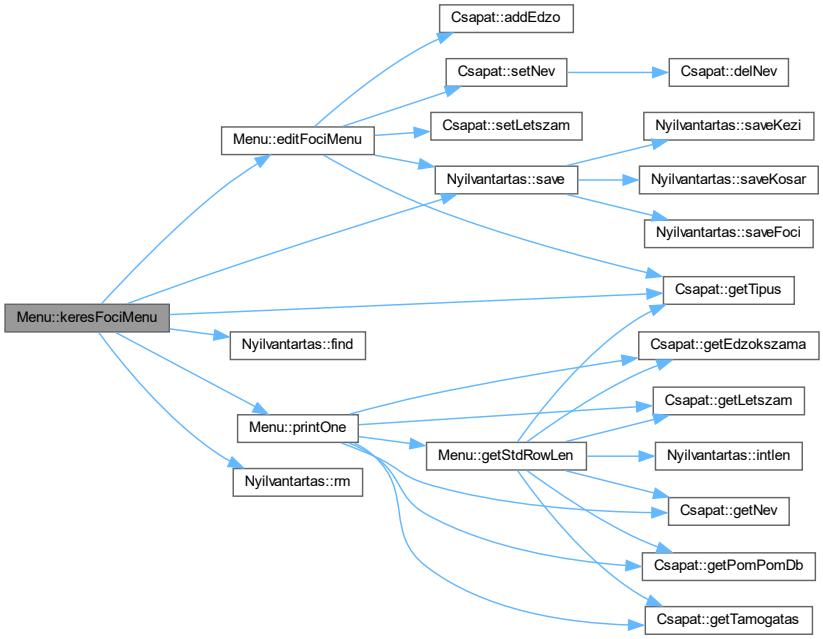


5.6.3.8 keresFociMenu()

```
void Menu::keresFociMenu ( )
```

Egy focilabda csapat keresési menüje.

Here is the call graph for this function:



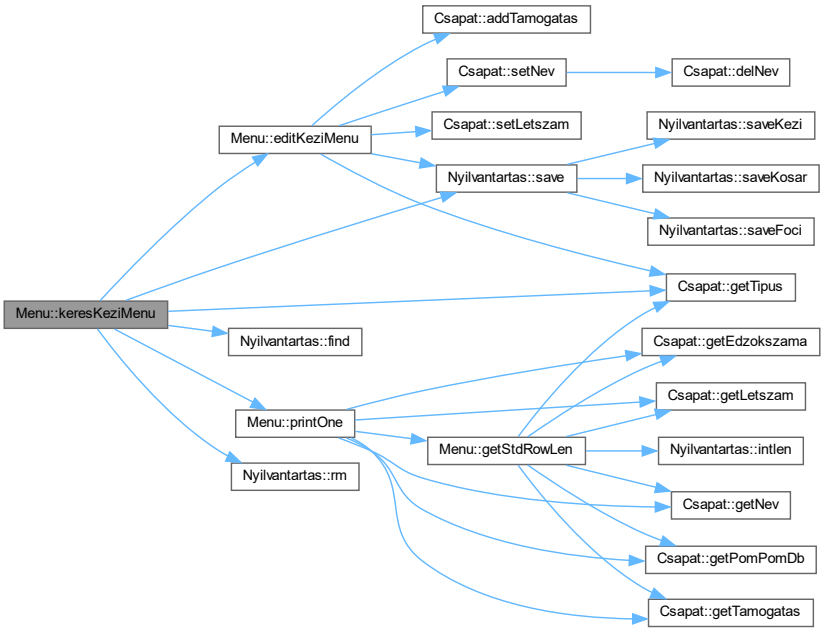
Here is the caller graph for this function:

5.6.3.9 keresKeziMenu()

```
void Menu::keresKeziMenu ( )
```

Egy kézilabda csapat keresési menüje.

Here is the call graph for this function:



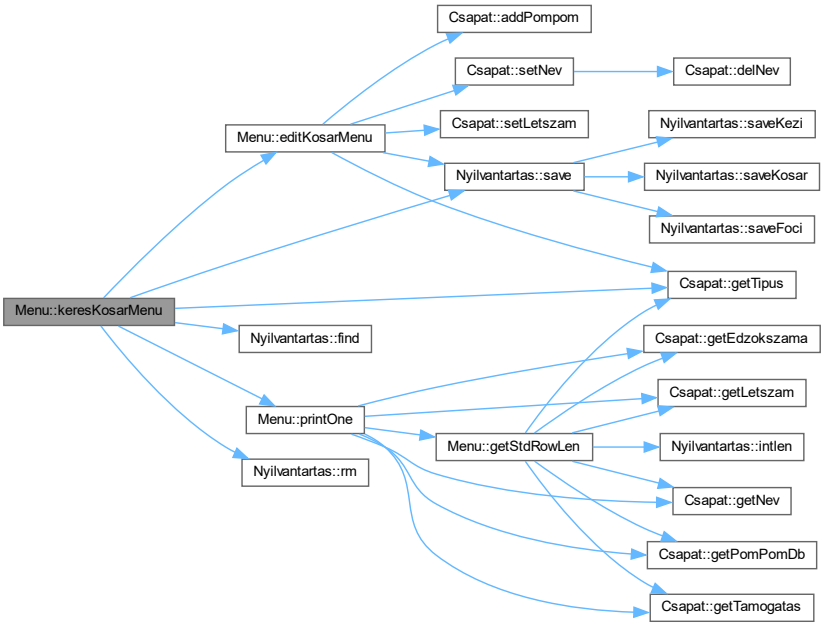
Here is the caller graph for this function:

5.6.3.10 keresKosarMenu()

```
void Menu::keresKosarMenu ( )
```

Egy kosárlabda csapat keresési menüje.

Here is the call graph for this function:



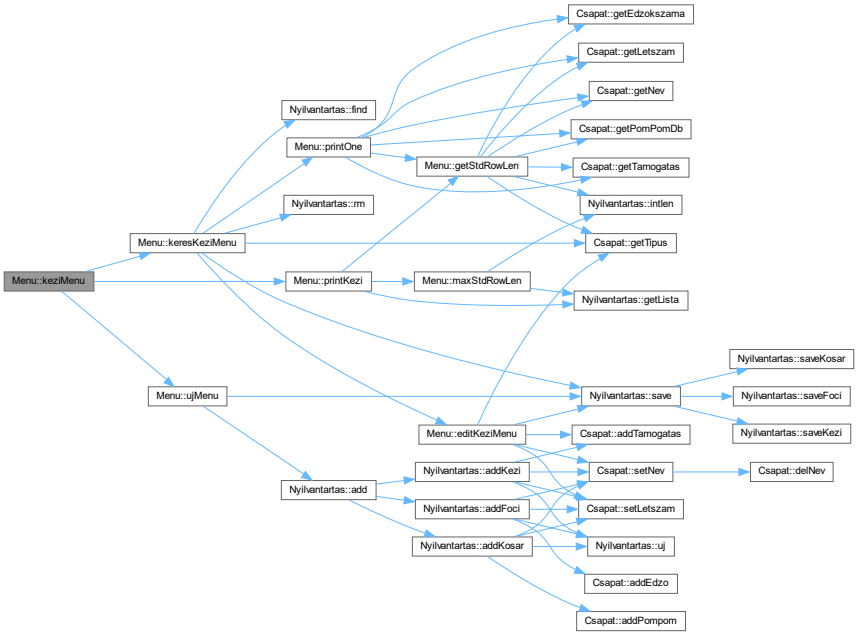
Here is the caller graph for this function:

5.6.3.11 keziMenu()

```
void Menu::keziMenu ( )
```

A kézilabda csapatokkal foglalkozó almenü. Innen lehet kézi sprecifikus dolgokat csinálni.

Here is the call graph for this function:



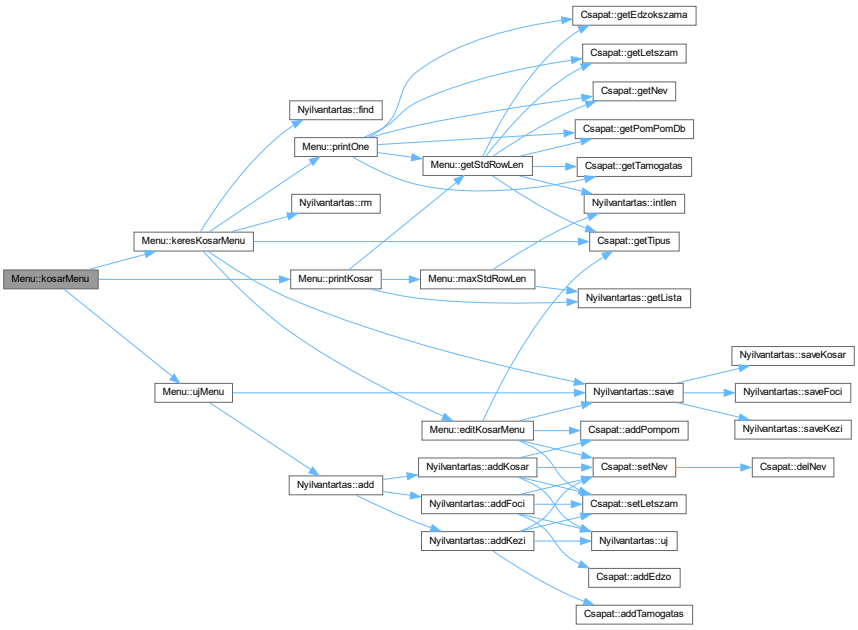
Here is the caller graph for this function:

5.6.3.12 kosarMenu()

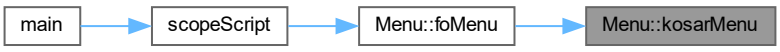
```
void Menu::kosarMenu ( )
```

A kosárlabda csapatokkal foglalkozó almenü. Innen lehet kosár sprecifikus dolgokat csinálni.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.13 maxStdRowLen() [1/2]

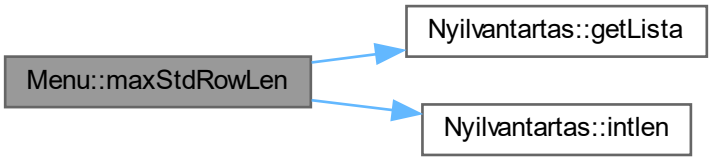
```
int Menu::maxStdRowLen ( ) const
```

Kiszámolja a leghoszabb sor hosszát a nyilvántartásban. Erre a TAB-ok és a kinézet miatt van szügség.

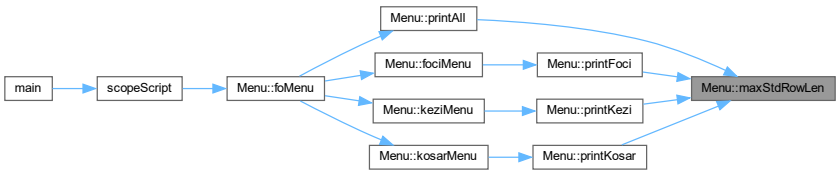
Returns

A leghoszabb sor hossza az adatbáisban.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.14 maxStdRowLen() [2/2]

```
int Menu::maxStdRowLen (
            Tipus t ) const
```

Kiszámolja, hogy a nyílvántartás típus szerinti láncolt listájában mennyi a leghoszabb sor. Erre a TAB-ok és a kinézet miatt van szüggség.

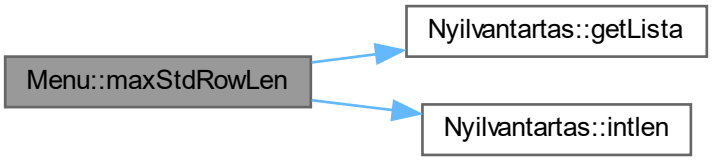
Parameters

tipus	A láncolt lista típusa, hogy melyikben keresse a leghoszabb sort.

Returns

A maximális sor hossz.

Here is the call graph for this function:

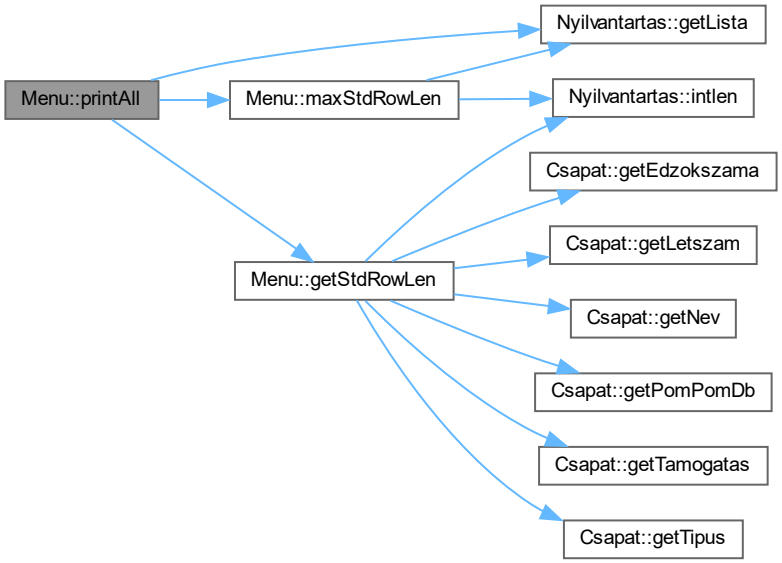


5.6.3.15 printAll()

```
void Menu::printAll ( ) const
```

Kilistázza megformázva az összes adatot.

Here is the call graph for this function:



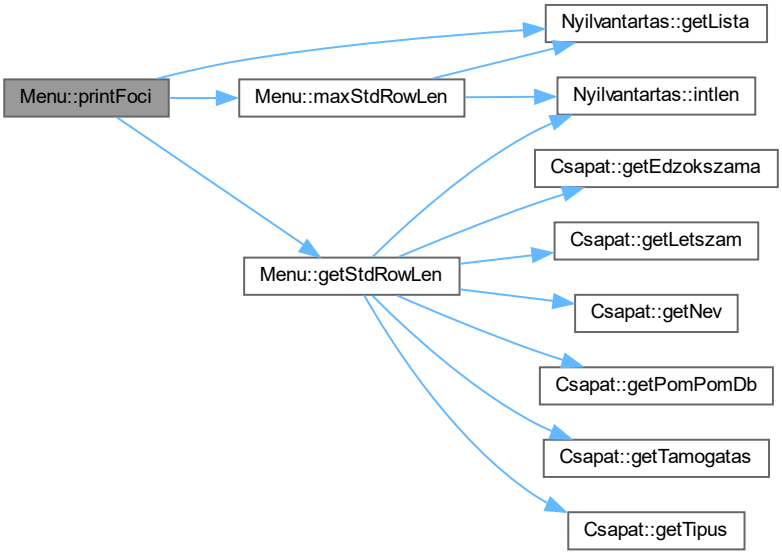
Here is the caller graph for this function:

5.6.3.16 printFoci()

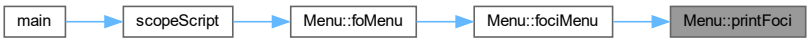
```
void Menu::printFoci ( ) const
```

Kilistázza megformázva a Focilabda (Foci) csapatokat.

Here is the call graph for this function:



Here is the caller graph for this function:

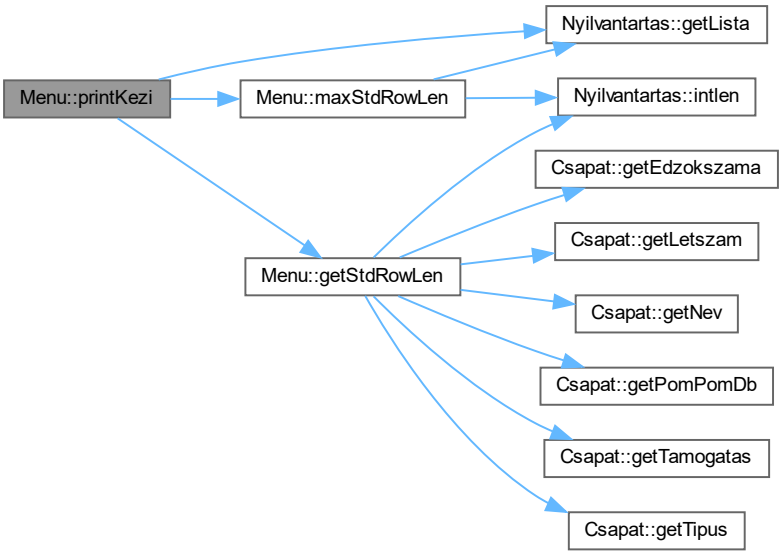


5.6.3.17 printKezi()

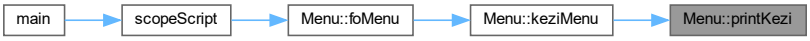
```
void Menu::printKezi ( ) const
```

Kilistázza megformázva a Kézilabda (Kezi) csapatokat.

Here is the call graph for this function:



Here is the caller graph for this function:

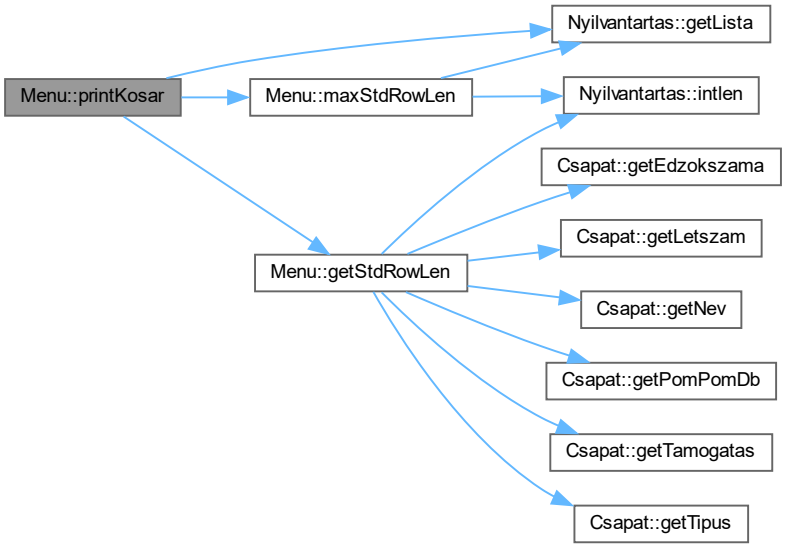


5.6.3.18 printKosar()

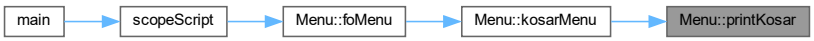
```
void Menu::printKosar ( ) const
```

Kilistázza megformázva a Kárlabda (Kosar) csapatokat.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.19 printOne()

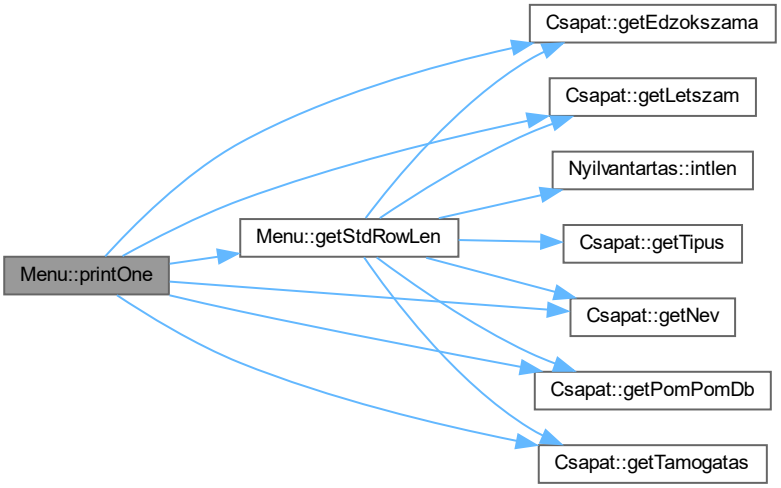
```
void Menu::printOne (
            Lista * p,
            Tipus t ) const
```

Egy listaelemet ír ki megformázva.

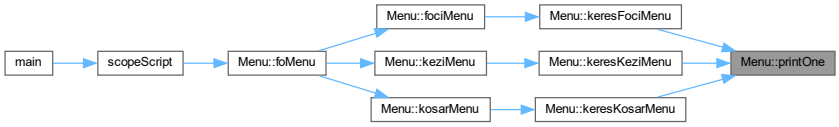
Parameters

listaelem	A lista láncszemére mutató pointer.
t	A típus, ami szerint kiírjuk a listaelemet

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.3.20 ujMenu()

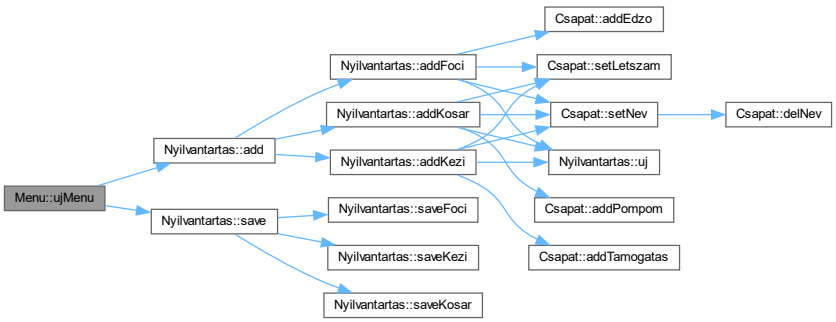
```
void Menu::ujMenu (
            Tipus T )
```

Egy új tipus típusú csapat lérehozására létező almenü.

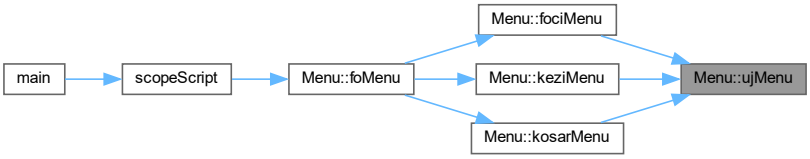
Parameters

tipus	Az új csapat típusa.

Here is the call graph for this function:



Here is the caller graph for this function:



5.6.4 Member Data Documentation

5.6.4.1 DB

`Nyilvantartas Menu::DB [private]`

A nyilvántartás adatbázis.

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

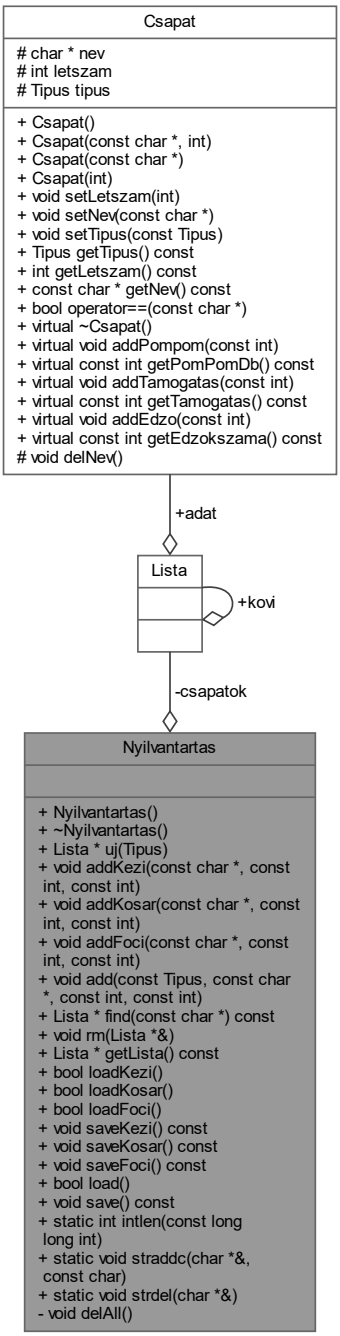
- menu.h
- menu.cpp

5.7 Nyilvantartas Class Reference

A nyilvántartás osztály. Ez tárolja a csapatokat (Kosar, Foci, Kezi) láncolt listákban.

`#include <nyilvantartas.h>`

Collaboration diagram for Nyilvantartas:



Public Member Functions

- Nyilvantartas ()

  A default konstruktor, amely 'nem csinál semmit'. Magyarul inicializála a.
- ~Nyilvantartas ()

  A destruktor, felszabadítja a láncolt listákat a segédfüggvények segítségével.
- Lista * uj (Tipus)

  Létrehoz egy új láncolt lista elemet, beláncolja a listába, majd.
- void addKezi (const char *, const int, const int)

  Beleláncol a listába a paramétereknek megfelelő Kézilabda csapatot.
- void addKosar (const char *, const int, const int)

  Beleláncol a listába a paramétereknek megfelelő Kosárlabda csapatot.
- void addFoci (const char *, const int, const int)

  Beleláncol a listába a paramétereknek megfelelő Focilabda csapatot.
- void add (const Tipus, const char *, const int, const int)

  Beleláncol a listába a paramétereknek megfelelő, Típustól függő csapatot csapatot.
- Lista * find (const char *) const

  A paraméternek megfelelő nevű csapatra mutató pointert ad vissza. Kikeresi a láncolt listából.
- void rm (Lista *&)

  Kitörli a láncolt listából a paraméterben megadott listaelemet.
- Lista * getLista () const

  Visszaadja a csapatok láncolt listáját.
- bool loadKezi ()

  Betölti a kezi.txt fileból a kézilabda csapat adatait a keziCS listába.
- bool loadKosar ()

  Betölti a kosar.txt fileból a kosárlabda csapat adatait a kosarCS listába.
- bool loadFoci ()

  Betölti a foci.txt fileból a kosárlabda csapat adatait a fociCS listába.
- void saveKezi () const

  Elmenti a kezi.txt fileba a keziCS adatait.
- void saveKosar () const

  Elmenti a kosar.txt fileba a kosarCS adatait.
- void saveFoci () const

  Elmenti a foci.txt fileba a fociCS adatait.
- bool load ()

  Betölti az összes fileból (kezi.txt, kosar.txt, foci.txt) az adatokat a.
- void save () const

  Elmenti az összes listát (keziCS, kosarCS, fociCS) a megfelelő fileokba.

Static Public Member Functions

- static int intlen (const long long int)

  Kiszamolja egy szám legnagyobb helyiértékét (tehát, milyen hosszú a szám). Statikus függvény.
- static void straddc (char *&, const char)

  Statikus. Hozzáfűz egy karakterpointerhez egy betűt. (VIGYÁZAT UTÁNNA DELETE[]-ELNI KELL).
- static void strdel (char *&)

  Felszabadít és nullptr-é tesz egy karakterpointert. Statikus.

Private Member Functions

- void delAll ()

  A kézilabda csapatokat tároló láncolt listát felszabadító segédfüggvény.

Private Attributes

- Lista * csapatok

  A csapatokat tároló lista.

5.7.1 Detailed Description

A nyilvántartás osztály. Ez tárolja a csapatokat (Kosar, Foci, Kezi) láncolt listákban.

Képes ezeket fileokból beolvasni, lementeni. Hozzáadni csapatokat, törölni csapatokat.

A program futása során ebből több mint egyet létrehozni nem kell (nem értelmes).

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Nyilvantartas()

```
Nyilvantartas::Nyilvantartas ( )  [inline]
```

A default konstruktor, amely 'nem csinál semmit'. Magyarul inicializála a.

láncolt listákat. (mindegyiket nullptr-re rakja).

5.7.2.2 ~Nyilvantartas()

```
Nyilvantartas::~Nyilvantartas ( )
```

A destruktor, felszabadítja a láncolt listákat a segédfüggvények segítségével.

Here is the call graph for this function:

5.7.3 Member Function Documentation

5.7.3.1 add()

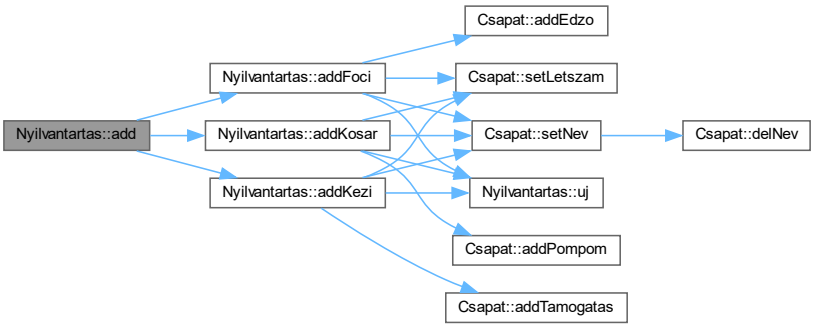
```
void Nyilvantartas::add (
            const Tipus T,
            const char * csnev,
            const int letszam,
            const int vari )
```

Beleláncol a listába a paramétereknek megfelelő, Típustól függő csapatot csapatot.

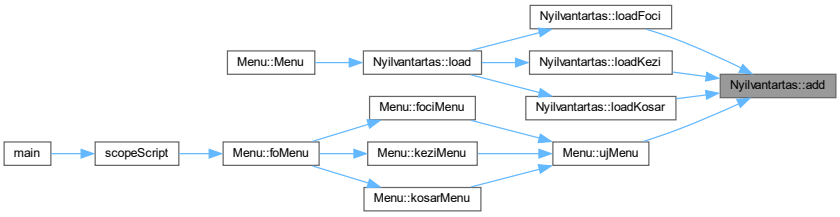
Parameters

csapat_tipus	Megadja a csapat típusát a Tipus enum segítségével.
csapatnev	a csapat neve.
letszam	a csapat létszáma.
csapat_speicfikus_szam	a csapatokra külön vonatkozó specifikus szám (támogatás, pompomlányok, edzők).

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.3.2 addFoci()

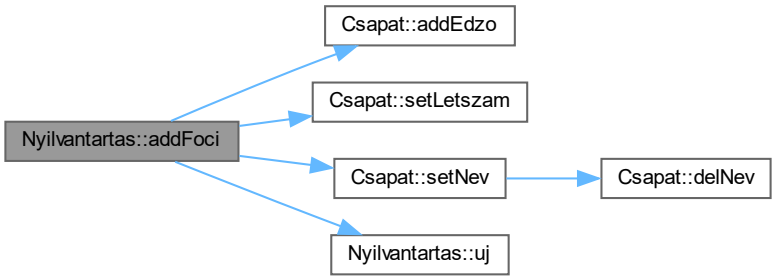
```
void Nyilvantartas::addFoci (
            const char * csnev,
            const int letszam,
            const int edzok )
```

Beleláncol a listába a paramétereknek megfelelő Focilabda csapatot.

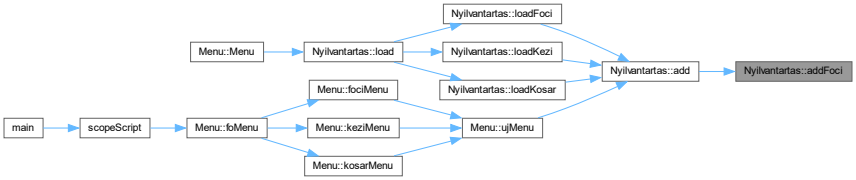
Parameters

csapatnev	a csapat neve.
letszam	a csapat létszáma.
edzok	a csapat edzőinek száma.

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.3.3 addKezi()

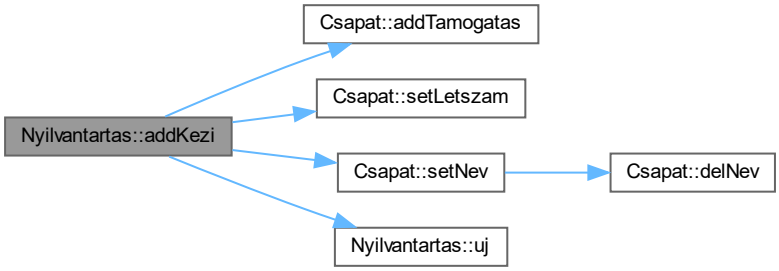
```
void Nyilvantartas::addKezi (
            const char * csnev,
            const int letszam,
            const int tamogatas )
```

Beleláncol a listába a paramétereknek megfelelő Kézilabda csapatot.

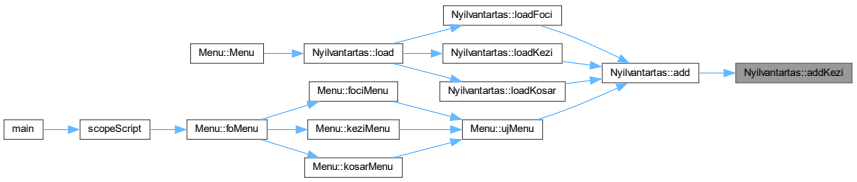
Parameters

csapatnev	a csapat neve.
letszam	a csapat létszáma.
tamogatas	a csapat támogatásai.

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.3.4 addKosar()

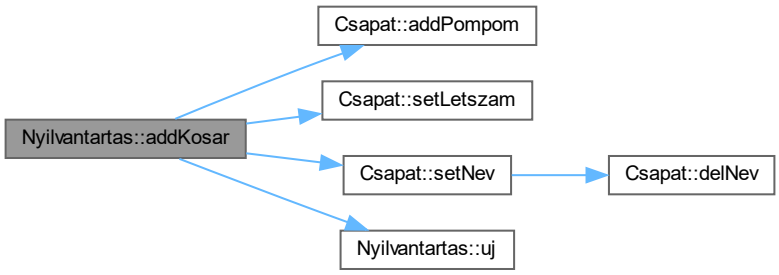
```
void Nyilvantartas::addKosar (
            const char * csnev,
            const int letszam,
            const int pompomn )
```

Beleláncol a listába a paramétereknek megfelelő Kosárlabda csapatot.

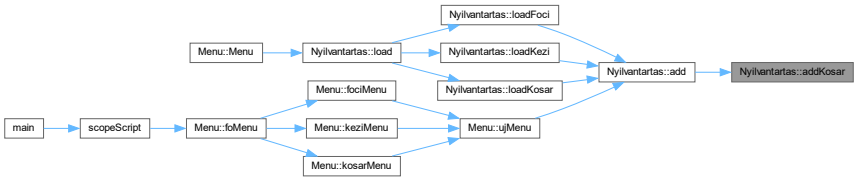
Parameters

csapatnev	a csapat neve.
letszam	a csapat létszáma.
pompom_lanyok	a csapat pompom lányainak száma.

Here is the call graph for this function:



Here is the caller graph for this function:

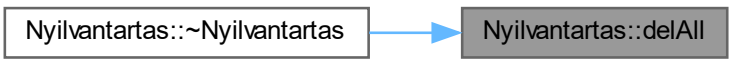


5.7.3.5 delAll()

```
void Nyilvantartas::delAll ( )  [inline], [private]
```

A kézilabda csapatokat tároló láncolt listát felszabadító segédfüggvény.

Here is the caller graph for this function:



5.7.3.6 find()

```
Lista * Nyilvantartas::find (
            const char * csapatnev ) const
```

A paraméternek megfelelő nevű csapatra mutató pointert ad vissza. Kikeresi a láncolt listából.

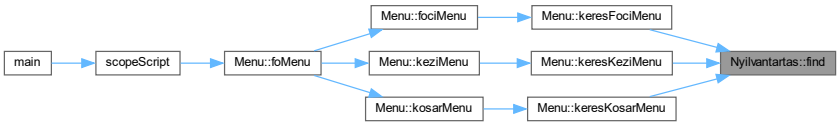
Parameters

csapat_nev	a keresendő csapat neve.

Returns

A keresendő csapatra mutató pointer, VAGY nullptr ha nem található ilyen csapat.

Here is the caller graph for this function:



5.7.3.7 getLista()

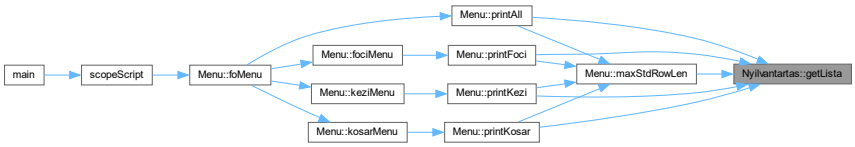
```
Lista * Nyilvantartas::getLista ( ) const
```

Visszaadja a csapatok láncolt listáját.

Returns

Az első listaelemre mutató pointer, vagy nullptr ha üres a lista.

Here is the caller graph for this function:



5.7.3.8 intlen()

```
int Nyilvantartas::intlen (
            const long long int szam )  [static]
```

Kiszamolja egy szám legnagyobb helyiértékét (tehát, milyen hosszú a szám). Statikus függvény.

Nem működik tökéletesen, de 63 számjegyig működik (egyébként sem reális 64 számjegyű támogatás,

vagy pompomlányok szóval most jó lesz...)

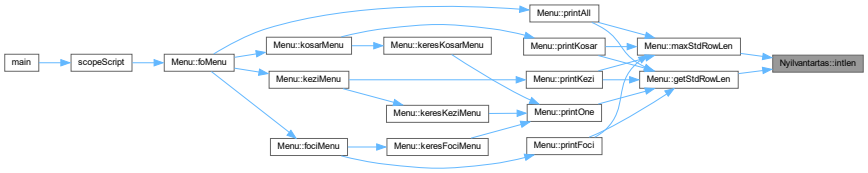
Parameters

szam	a kiszámolandó szám.

Returns

A szám legnagyobb helyiértéke.

Here is the caller graph for this function:



5.7.3.9 load()

```
bool Nyilvantartas::load ( )
```

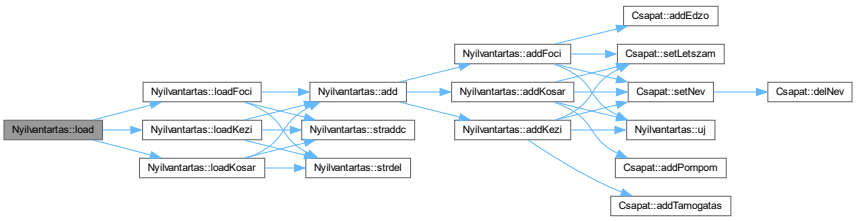
Betölti az összes fileból (kezi.txt, kosar.txt, foci.txt) az adatokat a.

listákba (keziCS, kosarCS, fociCS) a segédfüggvények segítségével.

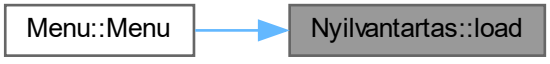
Returns

igaz ha sikeres, hamis ha nem sikeres.

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.3.10 loadFoci()

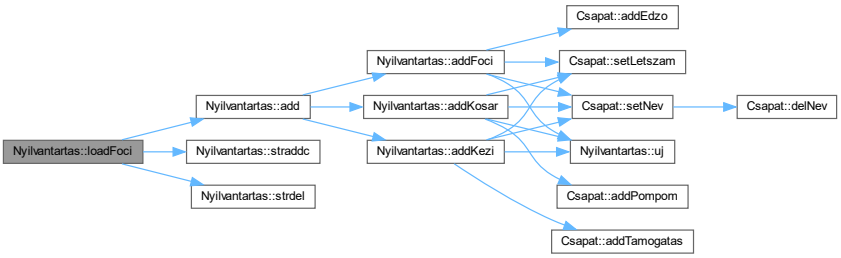
```
bool Nyilvantartas::loadFoci ( )
```

Betölti a foci.txt fileból a kosárlabda csapat adatait a fociCS listába.

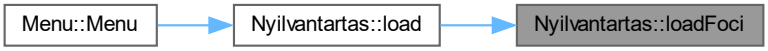
Returns

Igaz, ha sikerült betölteni, hamis ha nem.

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.3.11 loadKezi()

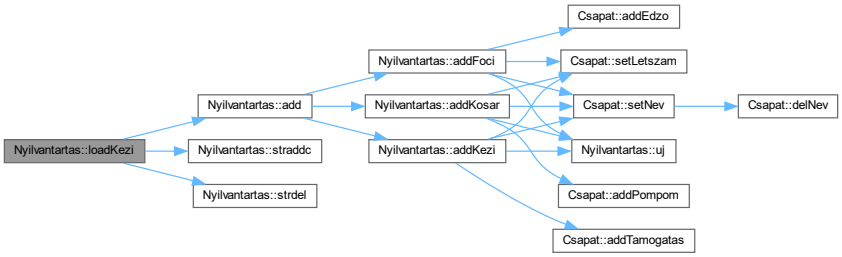
```
bool Nyilvantartas::loadKezi ( )
```

Betölti a kezi.txt fileból a kézilabda csapat adatait a keziCS listába.

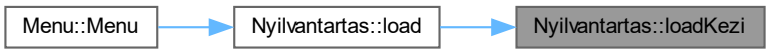
Returns

Igaz, ha sikerült betölteni, hamis ha nem.

Here is the call graph for this function:



Here is the caller graph for this function:



5.7.3.12 loadKosar()

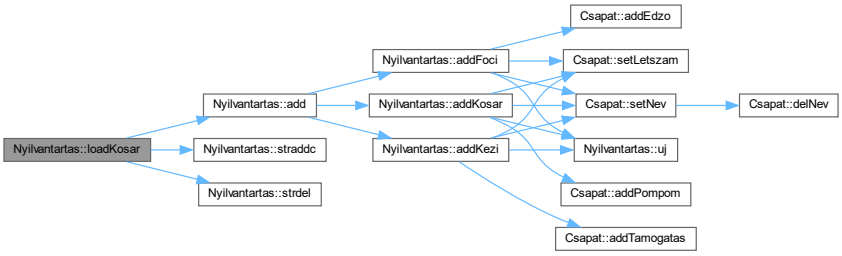
```
bool Nyilvantartas::loadKosar ( )
```

Betölti a kosar.txt fileból a kosárlabda csapat adatait a kosarCS listába.

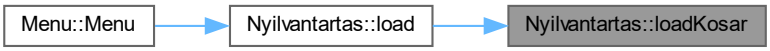
Returns

Igaz, ha sikerült betölteni, hamis ha nem.

File leírás:Here is the call graph for this function:



Here is the caller graph for this function:



5.7.3.13 rm()

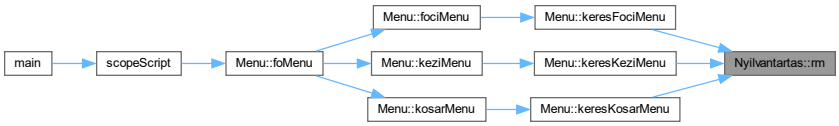
```
void Nyilvantartas::rm (
            Lista *& torlendo )
```

Kitörli a láncolt listából a paraméterben megadott listaelemet.

Parameters

lista_elem	a láncolt listában egy elem-re mutató pointer.

Here is the caller graph for this function:

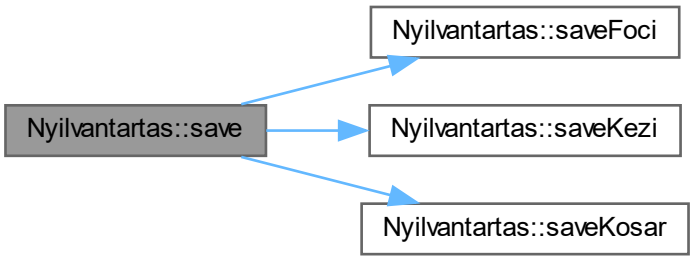


5.7.3.14 save()

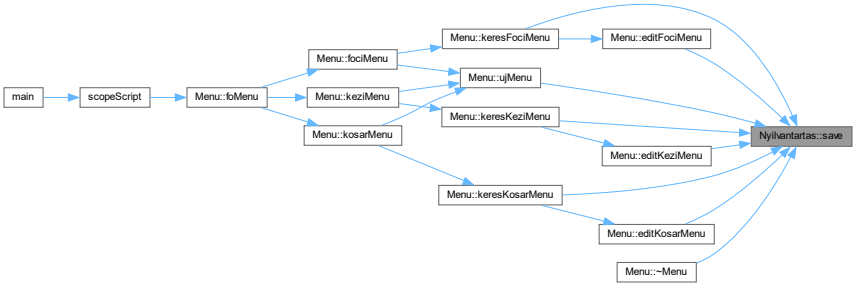
```
void Nyilvantartas::save ( ) const
```

Elmenti az összes listát (keziCS, kosarCS, fociCS) a megfelelő fileokba.

(kezi.txt, kosar.txt, foci.txt) a segédfüggvények segítségével. Here is the call graph for this function:



Here is the caller graph for this function:

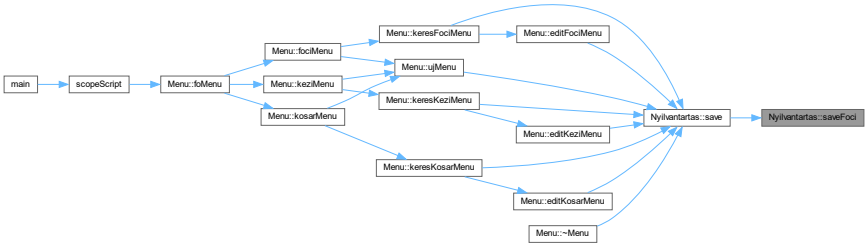


5.7.3.15 saveFoci()

```
void Nyilvantartas::saveFoci ( ) const
```

Elmenti a foci.txt fileba a fociCS adatait.

Here is the caller graph for this function:

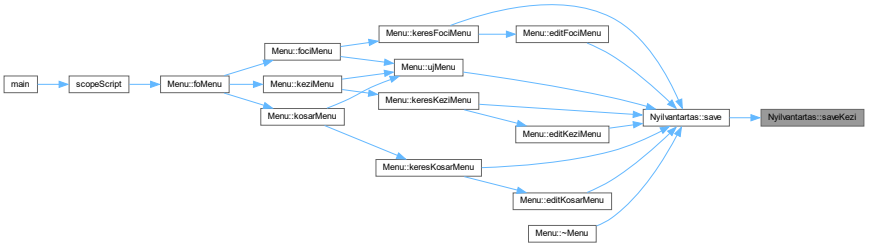


5.7.3.16 saveKezi()

```
void Nyilvantartas::saveKezi ( ) const
```

Elmenti a kezi.txt fileba a keziCS adatait.

Here is the caller graph for this function:

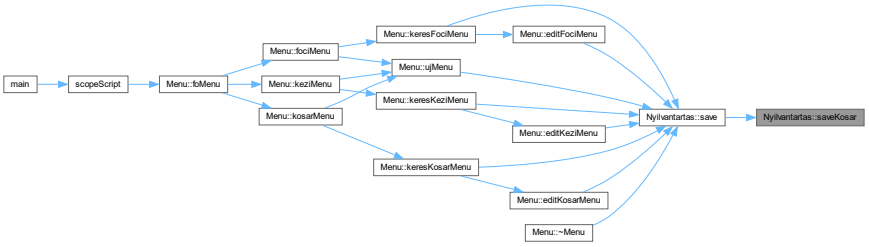


5.7.3.17 saveKosar()

```
void Nyilvantartas::saveKosar ( ) const
```

Elmenti a kosar.txt fileba a kosarCS adatait.

Here is the caller graph for this function:



5.7.3.18 straddc()

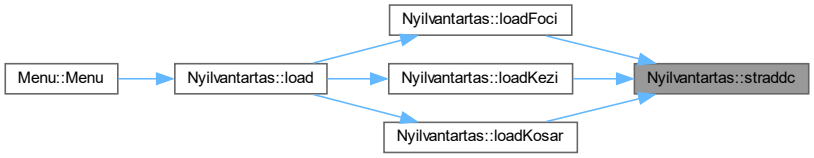
```
void Nyilvantartas::straddc (
            char *& str,
            const char c )  [static]
```

Statikus. Hozzáfűz egy karakterpointerhez egy betűt. (VIGYÁZAT UTÁNNA DELETE[]-ELNI KELL).

Parameters

str	A karakterpointer.
c	A hozzáadandó betű.

Here is the caller graph for this function:



5.7.3.19 strdel()

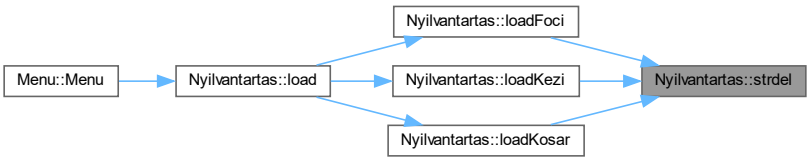
```
void Nyilvantartas::strdel (
            char *& str )  [static]
```

Felszabadít és nullptr-é tesz egy karakterpointert. Statikus.

Parameters

str	A karakertpointer.

Here is the caller graph for this function:



5.7.3.20 uj()

```
Lista * Nyilvantartas::uj (
            Tipus t )
```

Létrehoz egy új láncolt lista elemet, beláncolja a listába, majd.

visszaadja az erre mutató pointer (azért, hogy lehessen vele dolgozni.)

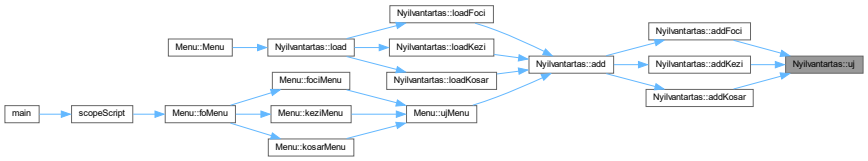
Parameters

tipus	Az új csapat típusa

Returns

Az újjonnan létrehozott láncolt listaelem pointere.

Here is the caller graph for this function:



5.7.4 Member Data Documentation

5.7.4.1 csapatok

```
Lista* Nyilvantartas::csapatok  [private]
```

A csapatokat tároló lista.

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

- nyilvantartas.h
- nyilvantartas.cpp

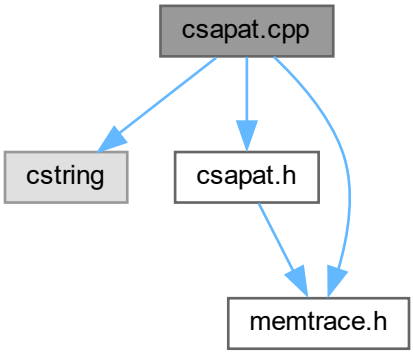
Chapter 6

File Documentation

6.1 csapat.cpp File Reference

```
#include <cstring>
#include "csapat.h"
#include "memtrace.h"
```

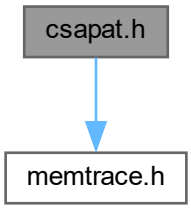
Include dependency graph for csapat.cpp:



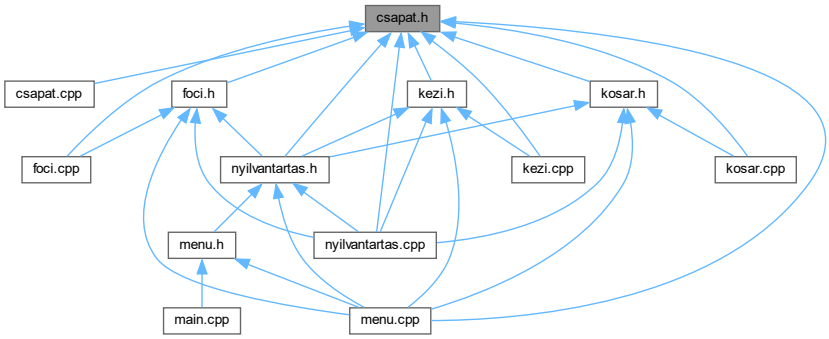
6.2 csapat.h File Reference

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

Include dependency graph for csapat.h:



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



Classes

- class Csapat

  A csapatok szülőobjektuma. Ez írja je a csapatok közös viselkedését, tulajdonságait.

Enumerations

- enum Tipus { NINCS , KEZI , FOCI , KOSAR }

  A csapatok típusait tartalmazó enum.

6.2.1 Enumeration Type Documentation

6.2.1.1 Tipus

`enum Tipus`

A csapatok típusait tartalmazó enum.

Enumerator

NINCS	Nincs típus.
KEZI	A kézilabda csapat típusa.
FOCI	A focicsapat típusa.
KOSAR	A kosárcsapat típusa.

6.3 csapat.h

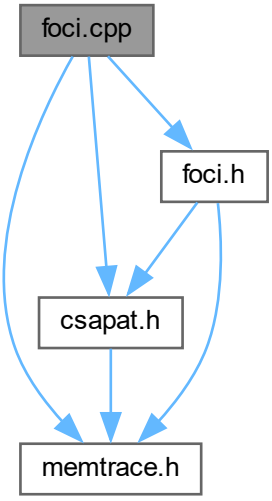
Go to the documentation of this file.

```
00001 #ifndef CSAPAT_H
00002 #define CSAPAT_H
00003
00004 #include "memtrace.h"
00005
00007 enum Tipus {
00009   NINCS,
00010
00012   KEZI,
00013
00015   FOCI,
00016
00018   KOSAR
00019 };
00020
00023 class Csapat {
00024   protected:
00026     char *nev;
00027
00029     int letszam;
00030
00032     Tipus tipus;
00033
00035     void delNev() { if (nev != nullptr) { delete[] nev; } };
00036
00037   public:
00039     Csapat() : nev(nullptr), letszam(0), tipus(NINCS) {};
00040
00044     Csapat(const char *, int);
00045
00048     Csapat(const char *);
00049
00052     Csapat(int);
00053
00056     void setLetszam(int);
00057
00060     void setNev(const char *);
00061
00064     void setTipus(const Tipus);
00065
00068     Tipus getTipus() const;
00069
00072     int getLetszam() const;
00073
00076     const char *getNev() const;
00077
00083     bool operator==(const char*);
00084
00086     virtual ~Csapat();
00087
00089     virtual void addPompom(const int) {};
00090
00092     virtual const int getPomPomDb() const {return -1;}
00093
00095     virtual void addTamogatas(const int) {}
00096
00098     virtual const int getTamogatas() const {return -1;}
00099
00101     virtual void addEdzo(const int) {}
00102
00104     virtual const int getEdzokszama() const {return -1;}
00105 };
00106
00107 #endif
```

6.4 foci.cpp File Reference

```
#include "csapat.h"
#include "foci.h"
#include "memtrace.h"
```

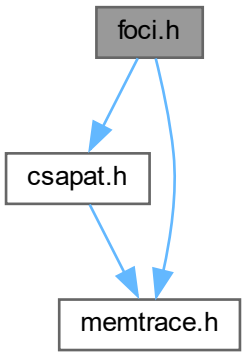
Include dependency graph for foci.cpp:



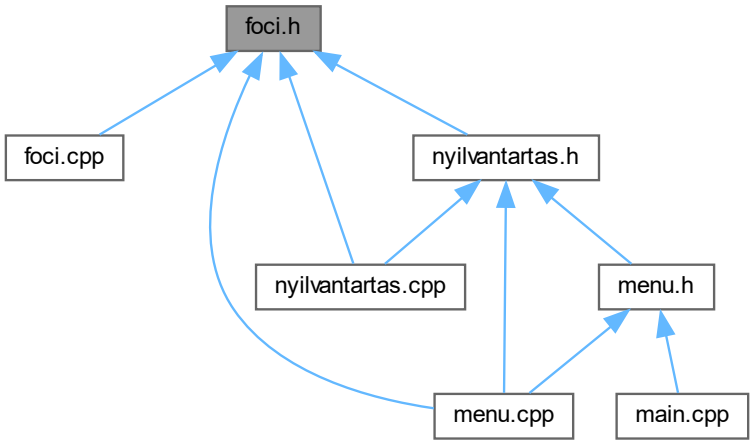
6.5 foci.h File Reference

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

Include dependency graph for foci.h:



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



Classes

- class Foci

  A focicsapat objektumja, amely örökli a Csapat objektum tulajdonságait.

6.6 foci.h

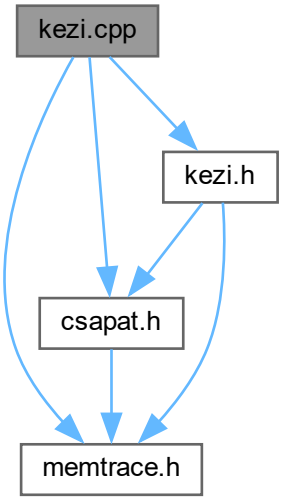
Go to the documentation of this file.

```
00001 #ifndef FOCI_H
00002 #define FOCI_H
00003
00004 #include "csapat.h"
00005 #include "memtrace.h"
00006
00008 class Foci : public Csapat {
00009   private:
00011     int edzoDB;
00012
00013   public:
00016     Foci();
00017
00021     Foci(const char *, const int);
00022
00025     void addEdzo(const int);
00026
00029     const int getEdzokszama() const;
00030
00031 };
00032
00033 #endif
```

6.7 kezi.cpp File Reference

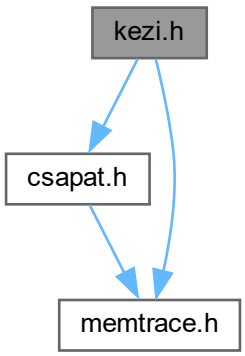
```
#include "csapat.h"
#include "kezi.h"
```

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

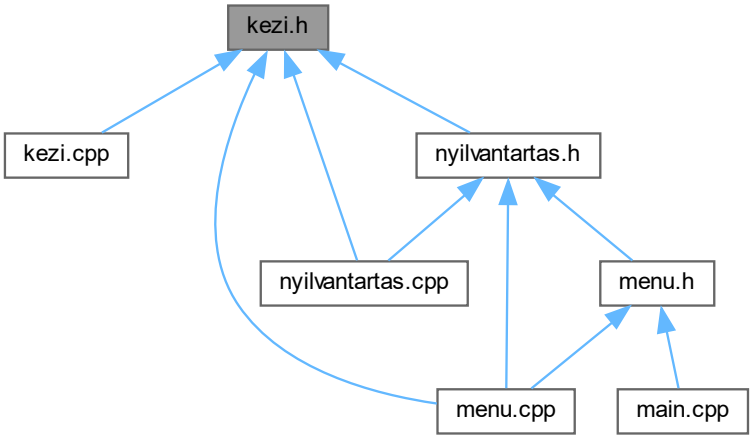


6.8 kezi.h File Reference

```
#include "csapat.h"
#include "memtrace.h"
```
Include dependency graph for kezi.h:



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



Classes

- class Kezi

  A kézilabda csapat objektumja, amely örökli a Csapat objektum tulajdonságait.

6.9 kezi.h

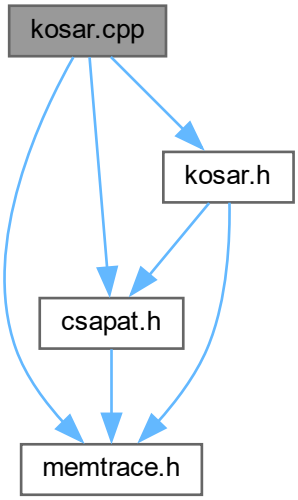
Go to the documentation of this file.

```
00001 #ifndef KEZI_H
00002 #define KEZI_H
00003
00004 #include "csapat.h"
00005 #include "memtrace.h"
00006
00008 class Kezi : public Csapat {
00009   private:
00011     int tamogatas;
00012
00013   public:
00016     Kezi();
00017
00021     Kezi(const char*, const int);
00022
00025     void addTamogatas(const int);
00026
00029     const int getTamogatas() const;
00030 };
00031
00032 #endif
```

6.10 kosar.cpp File Reference

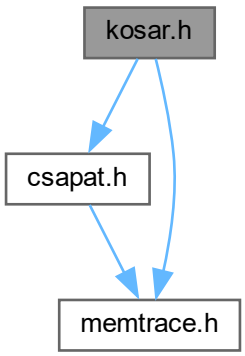
```
#include "csapat.h"
#include "kosar.h"
```

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

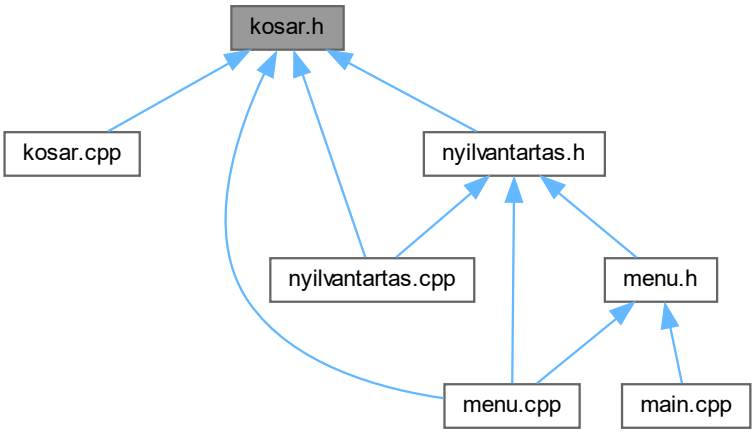


6.11 kosar.h File Reference

```
#include "csapat.h"
#include "memtrace.h"
```
Include dependency graph for kosar.h:



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



Classes

- class Kosar

  A kosárlabda csapat objektumja, amely örökli a Csapat objektum tulajdonságait.

6.12 kosar.h

Go to the documentation of this file.

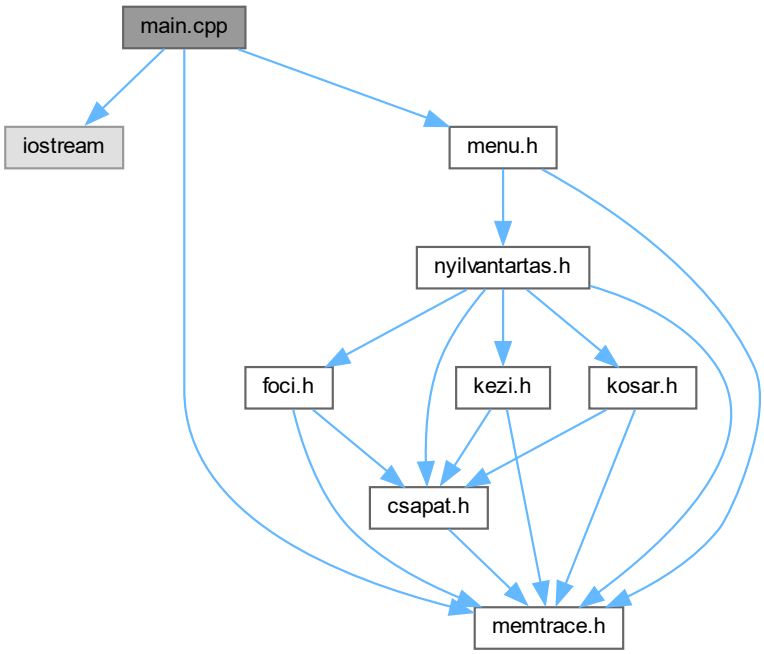
```
00001 #ifndef KOSAR_H
00002 #define KOSAR_H
00003
00004 #include "csapat.h"
00005 #include "memtrace.h"
00006
00008 class Kosar : public Csapat {
00009   private:
00011     int pompomDB;
00012
00013   public:
00016     Kosar();
00017
00021     Kosar(const char*, const int);
00022
00025     void addPompom(const int);
00026
00029     const int getPomPomDb() const;
00030
00031 };
00032
00033 #endif
```

6.13 main.cpp File Reference

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

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

Include dependency graph for main.cpp:



Functions

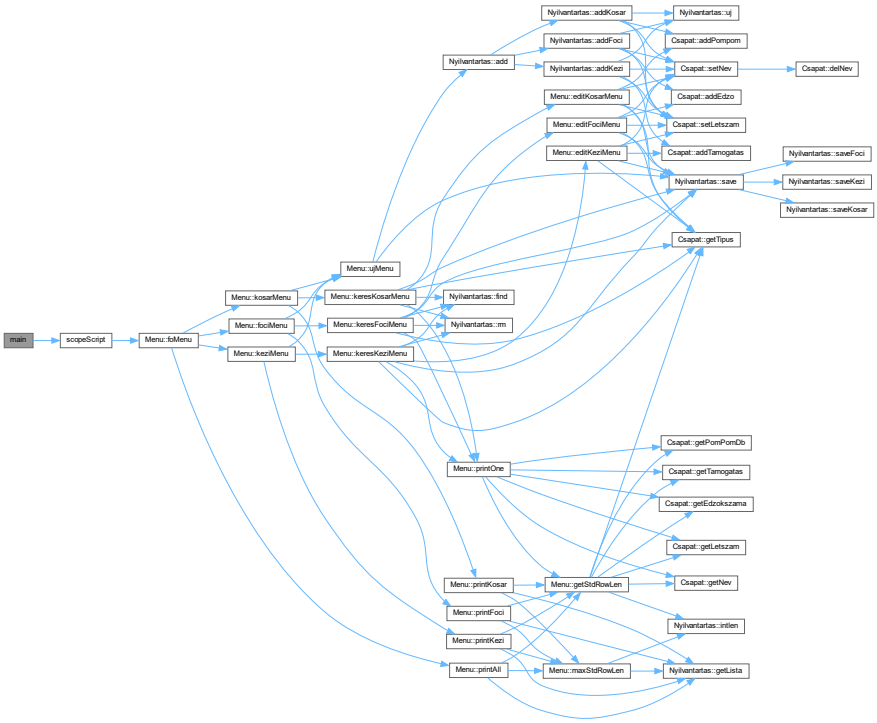
- void scopeScript ()
- int main ()

6.13.1 Function Documentation

6.13.1.1 main()

```
int main ( )
```

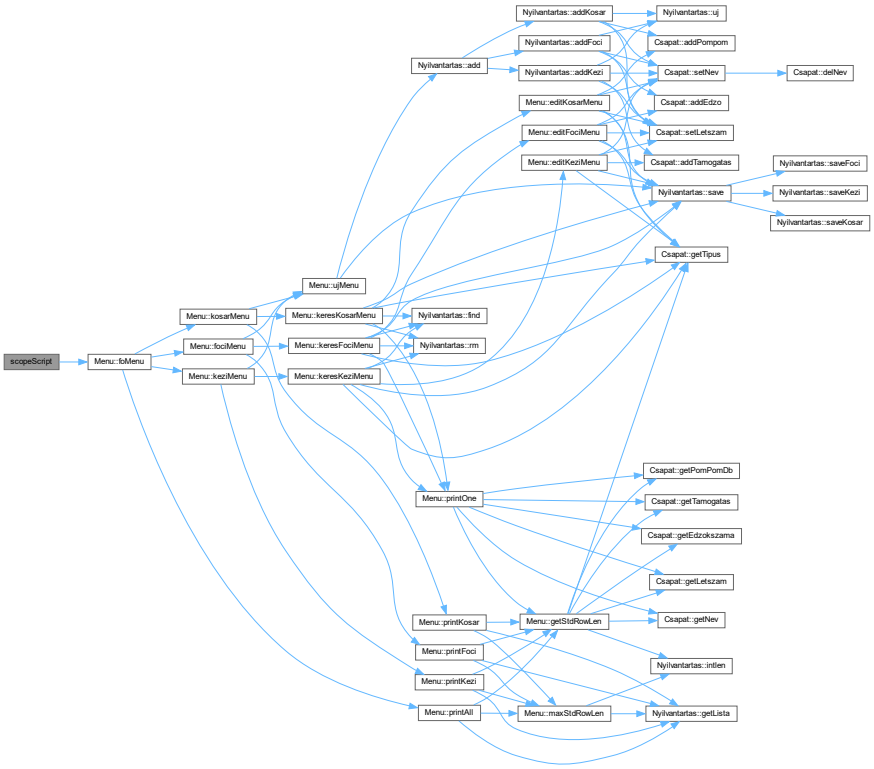
Here is the call graph for this function:



6.13.1.2 scopeScript()

```
void scopeScript ( )
```

Here is the call graph for this function:

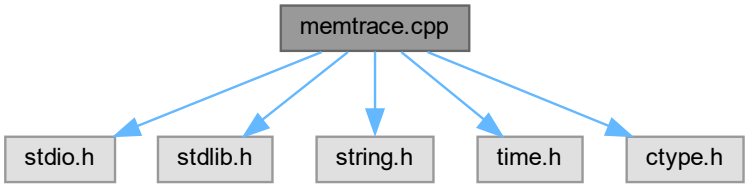


Here is the caller graph for this function:

6.14 memtrace.cpp File Reference

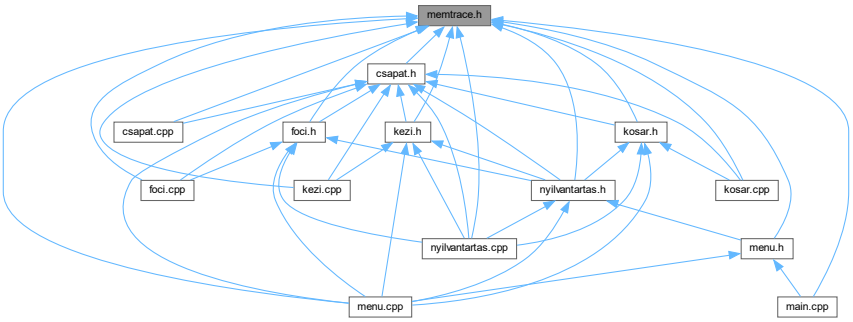
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
```

Include dependency graph for memtrace.cpp:



6.15 memtrace.h File Reference

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



6.16 memtrace.h

Go to the documentation of this file.

```
00001 /*********************************
00002 Memoriaszivargas-detektor
00003 Keszitette: Peregi Tamas, BME IIT, 2011
00004             petamas@iit.bme.hu
00005 Kanari:     Szeberenyi Imre, 2013.,
00006 VS 2012:    Szeberényi Imre, 2015.,
00007 mem_dump:   2016.
00008 inclue-ok:  2017., 2018. 2019.
00009 *********************************/
00010
00011 #ifndef MEMTRACE_H
00012 #define MEMTRACE_H
00013
00014 #if defined(MEMTRACE)
00015
00016 /*ha definiálva van, akkor a hibakat ebbe a fajlba írja, egyébkent stderr-re*/
00017 /*#define MEMTRACE_ERRFILE MEMTRACE.ERR*/
00018
00019 /*ha definialva van, akkor futas kozben lancolt listat epit. Javasolt a hasznalata*/
00020 #define MEMTRACE_TO_MEMORY
00021
00022 /*ha definialva van, akkor futas kozben fajlba irja a foglalasokat*/
00023 /*ekkor nincs ellenorzes, csak naplozas*/
00024 /*#define MEMTRACE_TO_FILE*/
```

```
00025
00026 /*ha definialva van, akkor a megallaskor automatikus riport keszul */
00027 #define MEMTRACE_AUTO
00028
00029 /*ha definialva van, akkor malloc()/calloc()/realloc()/free() kovetve lesz*/
00030 #define MEMTRACE_C
00031
00032 #ifdef MEMTRACE_C
00033     /*ha definialva van, akkor free(NULL) nem okoz hibat*/
00034     #define ALLOW_FREE_NULL
00035 #endif
00036
00037 #ifdef __cplusplus
00038     /*ha definialva van, akkor new/delete/new[]/delete[] kovetve lesz*/
00039     #define MEMTRACE_CPP
00040 #endif
00041
00042 #if defined(__cplusplus) && defined(MEMTRACE_TO_MEMORY)
00043     /*ha definialva van, akkor atexit helyett objektumot hasznal*/
00044     /*ajanlott bekapcsolni*/
00045     #define USE_ATEXIT_OBJECT
00046 #endif
00047
00048 /*****************************************/
00049 /* INNEN NE MODOSITSD                    */
00050 /*****************************************/
00051 #ifdef NO_MEMTRACE_TO_FILE
00052     #undef MEMTRACE_TO_FILE
00053 #endif
00054
00055 #ifdef NO_MEMTRACE_TO_MEMORY
00056     #undef MEMTRACE_TO_MEMORY
00057 #endif
00058
00059 #ifndef MEMTRACE_AUTO
00060     #undef USE_ATEXIT_OBJECT
00061 #endif
00062
00063 #ifdef __cplusplus
00064     #define START_NAMESPACE namespace memtrace {
00065     #define END_NAMESPACE } /*namespace*/
00066     #define TRACEC(func) memtrace::func
00067     #include <new>
00068 #else
00069     #define START_NAMESPACE
00070     #define END_NAMESPACE
00071     #define TRACEC(func) func
00072 #endif
00073
00074 // THROW deklaráció változatai
00075 #if defined(_MSC_VER)
00076   // VS rosszul kezeli az __cplusplus makrot
00077   #if _MSC_VER < 1900
00078     // * nem biztos, hogy jó így *
00079     #define THROW_BADALLOC
00080     #define THROW_NOTHING
00081   #else
00082     // C++11 vagy újabb
00083     #define THROW_BADALLOC noexcept(false)
00084     #define THROW_NOTHING noexcept
00085   #endif
00086 #else
00087   #if __cplusplus < 201103L
00088     // C++2003 vagy régebbi
00089     #define THROW_BADALLOC throw (std::bad_alloc)
00090     #define THROW_NOTHING throw ()
00091   #else
00092     // C++11 vagy újabb
00093     #define THROW_BADALLOC noexcept(false)
00094     #define THROW_NOTHING noexcept
00095   #endif
00096 #endif
00097
00098 START_NAMESPACE
00099     int allocated_blocks();
00100 END_NAMESPACE
00101
00102 #if defined(MEMTRACE_TO_MEMORY)
00103 START_NAMESPACE
00104         int mem_check(void);
00105 END_NAMESPACE
00106 #endif
00107
00108 #if defined(MEMTRACE_TO_MEMORY) && defined(USE_ATEXIT_OBJECT)
00109 #include <cstdio>
00110 START_NAMESPACE
00111     class atexit_class {
```

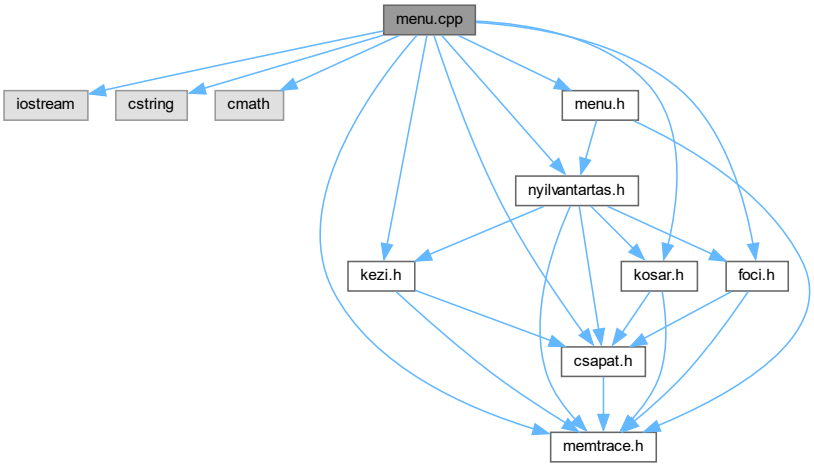
```
00112         private:
00113             static int counter;
00114             static int err;
00115         public:
00116             atexit_class() {
00117 #if defined(CPORTA) && !defined(CPORTA_NOSETBUF)
00118                 if (counter == 0) {
00119                         setbuf(stdout, 0);
00120                         setbuf(stderr, 0);
00121                 }
00122 #endif
00123             counter++;
00124             }
00125 
00126             int check() {
00127                 if(--counter == 0)
00128                     err = mem_check();
00129                 return err;
00130             }
00131 
00132             ~atexit_class() {
00133                 check();
00134             }
00135     };
00136 
00137 static atexit_class atexit_obj;
00138 
00139 END_NAMESPACE
00140 #endif/*MEMTRACE_TO_MEMORY && USE_ATEXIT_OBJECT*/
00141 
00142 /*Innentol csak a "normal" include eseten kell, kulonben osszezavarja a mukodest*/
00143 #ifndef FROM_MEMTRACE_CPP
00144 #include <stdlib.h>
00145 #ifdef __cplusplus
00146     #include <iostream>
00147 /* ide gyűjtjük a nemtrace-vel összeakadó headereket, hogy előbb legyenek */
00148 
00149     #include <fstream>  // VS 2013 headerjében van deleted definició
00150     #include <sstream>
00151     #include <vector>
00152     #include <list>
00153     #include <map>
00154     #include <algorithm>
00155     #include <functional>
00156 #endif
00157 #ifdef MEMTRACE_CPP
00158     namespace std {
00159         typedef void (*new_handler)();
00160 }
00161 #endif
00162 
00163 #ifdef MEMTRACE_C
00164 START_NAMESPACE
00165     #undef malloc
00166     #define malloc(size) TRACEC(traced_malloc)(size,#size,__LINE__,__FILE__)
00167     void * traced_malloc(size_t size, const char *size_txt, int line, const char * file);
00168 
00169     #undef calloc
00170     #define calloc(count,size) TRACEC(traced_calloc)(count, size, #count","#size,__LINE__,__FILE__)
00171     void * traced_calloc(size_t count, size_t size, const char *size_txt, int line, const char *
    file);
00172 
00173     #undef free
00174     #define free(p) TRACEC(traced_free)(p, #p,__LINE__,__FILE__)
00175     void traced_free(void * p, const char *size_txt, int line, const char * file);
00176 
00177     #undef realloc
00178     #define realloc(old,size) TRACEC(traced_realloc)(old,size,#size,__LINE__,__FILE__)
00179     void * traced_realloc(void * old, size_t size, const char *size_txt, int line, const char * file);
00180 
00181     void mem_dump(void const *mem, size_t size, FILE* fp);
00182 
00183 
00184 END_NAMESPACE
00185 #endif/*MEMTRACE_C*/
00186 
00187 #ifdef MEMTRACE_CPP
00188 START_NAMESPACE
00189     #undef set_new_handler
00190     #define set_new_handler(f) TRACEC(_set_new_handler)(f)
00191     void _set_new_handler(std::new_handler h);
00192 
00193     void set_delete_call(int line, const char * file);
00194 END_NAMESPACE
00195 
00196 void * operator new(size_t size, int line, const char * file) THROW_BADALLOC;
00197 void * operator new[](size_t size, int line, const char * file) THROW_BADALLOC;
```

```
00198 void * operator new(size_t size) THROW_BADALLOC;
00199 void * operator new[](size_t size) THROW_BADALLOC;
00200 void operator delete(void * p)  THROW_NOTHING;
00201 void operator delete[](void * p) THROW_NOTHING;
00202
00203 /* Visual C++ 2012 miatt kell, mert háklis, hogy nincs megfelelő delete, bár senki sem használja */
00204 void operator delete(void *p, int, const char *) THROW_NOTHING;
00205 void operator delete[](void *p, int, const char *) THROW_NOTHING;
00206
00207
00208 #define new new(__LINE__, __FILE__)
00209 #define delete memtrace::set_delete_call(__LINE__, __FILE__),delete
00210
00211 #ifdef CPORTA
00212 #define system(...)  // system(__VA_ARGS__)
00213 #endif
00214
00215 #endif /*MEMTRACE_CPP*/
00216
00217 #endif /*FROM_MEMTRACE_CPP*/
00218 #endif /*MEMCHECK*/
00219 #endif /*MEMTRACE_H*/
```

6.17 menu.cpp File Reference

```
#include <iostream>
#include <cstring>
#include <cmath>
#include "nyilvantartas.h"
#include "menu.h"
#include "csapat.h"
#include "kezi.h"
#include "kosar.h"
#include "foci.h"
#include "memtrace.h"
```

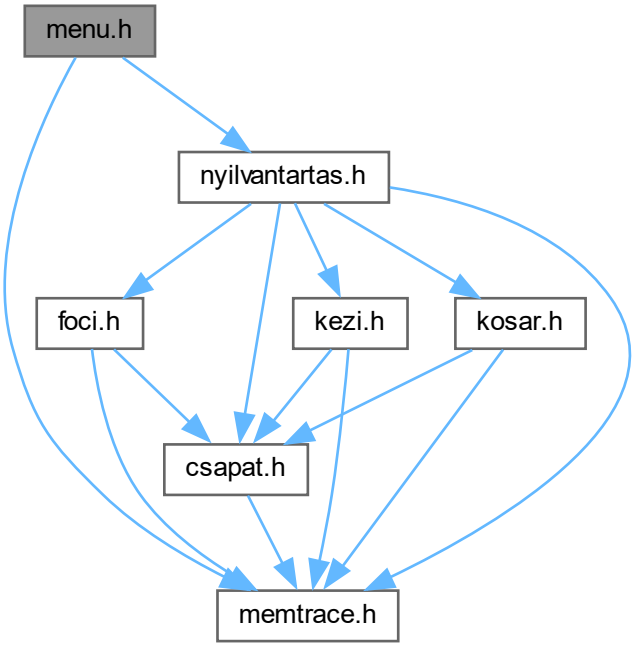
Include dependency graph for menu.cpp:



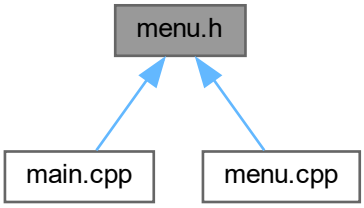
6.18 menu.h File Reference

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

Include dependency graph for menu.h:



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



Classes

- class Menu

  A futó programot irányító menürendszer objektuma.

6.19 menu.h

Go to the documentation of this file.

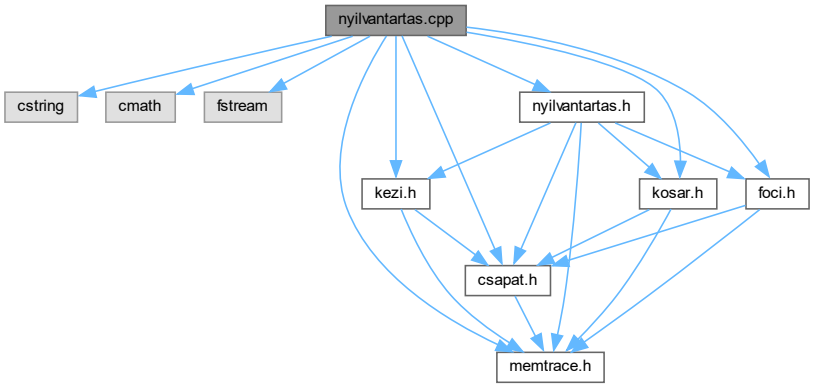
```
00001 #ifndef MENU_H
00002 #define MENU_H
00003
00004 #include "nyilvantartas.h"
00005 #include "memtrace.h"
00006
00008 class Menu {
00009   private:
00011     Nyilvantartas DB;
00012   public:
00013
00015     Menu();
00016
00018     ~Menu();
00019
00022     Nyilvantartas getNyilvantartas() const { return DB;}
00023
00025     void printAll() const;
00026
00028     void printKezi() const;
00029
00031     void printKosar() const;
00032
00034     void printFoci() const;
00035
00039     void printOne(Lista *, Tipus) const;
00040
00043     int maxStdRowLen() const;
00044
00048     int maxStdRowLen(Tipus) const;
00049
00053     int getStdRowLen(Lista *) const;
00054
00057     void foMenu();
00058
00060     void keziMenu();
00061
00063     void kosarMenu();
00064
00066     void fociMenu();
00067
00069     void keresKeziMenu();
00070
00072     void keresKosarMenu();
00073
00075     void keresFociMenu();
00076
00079     void editKeziMenu(Lista *);
00080
00083     void editKosarMenu(Lista*);
00084
00087     void editFociMenu(Lista *);
00088
00091     void ujMenu(Tipus);
00092 };
00093
00094 #endif
```

6.20 nyilvantartas.cpp File Reference

```
#include <cstring>
#include <cmath>
#include <fstream>
#include "nyilvantartas.h"
#include "csapat.h"
#include "kezi.h"
#include "kosar.h"
#include "foci.h"
```

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

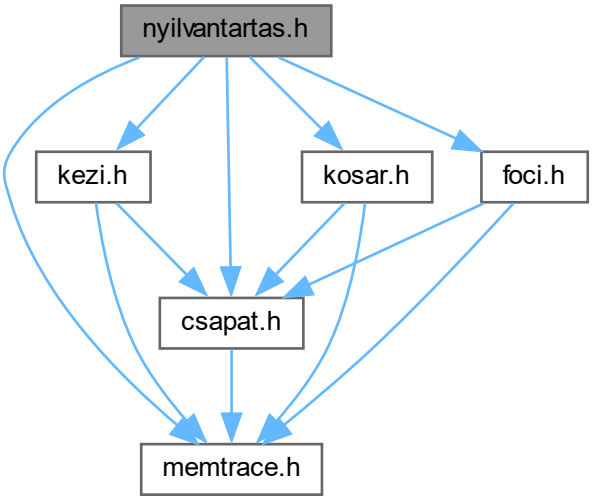
Include dependency graph for nyilvantartas.cpp:



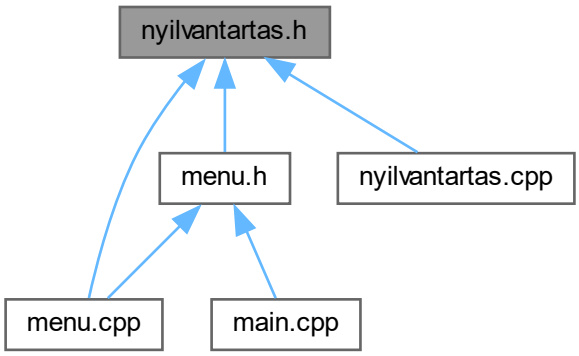
6.21 nyilvantartas.h File Reference

```
#include "csapat.h"
#include "kezi.h"
#include "kosar.h"
#include "foci.h"
#include "memtrace.h"
```

Include dependency graph for nyilvantartas.h:



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



Classes

- struct Lista

  Láncolt listaelem.
- class Nyilvantartas

  A nyilvántartás osztály. Ez tárolja a csapatokat (Kosar, Foci, Kezi) láncolt listákban.

6.22 nyilvantartas.h

Go to the documentation of this file.

```
00001 #ifndef NYILVANTARTAS_H
00002 #define NYILVANTARTAS_H
00003
00004 #include "csapat.h"
00005 #include "kezi.h"
00006 #include "kosar.h"
00007 #include "foci.h"
00008 #include "memtrace.h"
00009
00011 struct Lista {
00012   Csapat *adat;
00013   Lista *kovi;
00014 };
00015
00019 class Nyilvantartas {
00020   private:
00022     Lista *csapatok;
00023
00025     void delAll(){
00026       if (csapatok != nullptr) {
00027         Lista* i = csapatok;
00028         while (i != nullptr) {
00029           Lista *kov = i->kovi;
00030           delete i->adat;
00031           delete i;
00032           i = kov;
00033         }
00034       }
00035     }
00036
00037   public:
00038
00044     static int intlen(const long long int);
00045
```

```
00049     static void straddc(char *&, const char);
00050
00053     static void strdel(char *&);
00054
00057     Nyilvantartas() : csapatok(nullptr) {}
00058
00060     ~Nyilvantartas();
00061
00066     Lista *uj(Tipus);
00067
00072     void addKezi(const char*, const int, const int);
00073
00078     void addKosar(const char*, const int, const int);
00079
00084     void addFoci(const char*, const int, const int);
00085
00091     void add(const Tipus, const char*, const int, const int);
00092
00096     Lista *find(const char*) const;
00097
00100     void rm(Lista *&);
00101
00104     Lista *getLista() const;
00105
00108     bool loadKezi();
00109
00112     bool loadKosar();
00113
00116     bool loadFoci();
00117
00119     void saveKezi() const;
00120
00122     void saveKosar() const;
00123
00125     void saveFoci() const;
00126
00130     bool load();
00131
00134     void save() const;
00135 };
00136
00137 #endif
```

6.23 README.md File Reference

Index