# Nagyházi

Készítette Doxygen 1.9.6

# 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

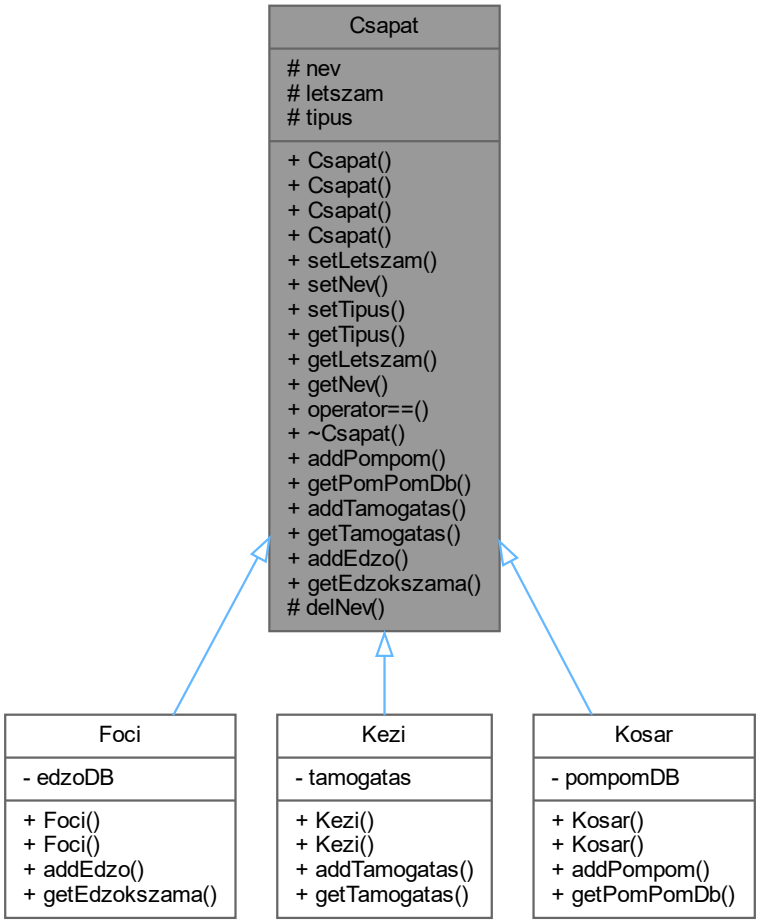
# Osztályok dokumentációja

## 4.1. Csapat osztályreferencia

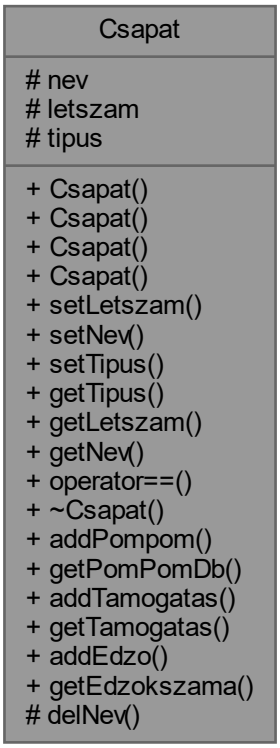
A csapatok szülőobjektuma. Ez írja je a csapatok közös viselkedését, tulajdonságait.

`#include <csapat.h>`

A Csapat osztály származási diagramja:



A Csapat osztály együttműködési diagramja:




## Publikus tagfüggvények

- 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.*

## Védett tagfüggvények

- void delNev ()

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

## Védett attribútumok

- char * nev

  *A csapat neve.*
- int letszam

  *A csapat létszáma.*
- Tipus tipus

  *A csapat típusa a Típus enum segítségével.*

### 4.1.1. Részletes leírás

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

### 4.1.2. Konstruktorok és destruktorok dokumentációja

#### 4.1.2.1. Csapat() [1/4]

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

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

#### 4.1.2.2. Csapat() [2/4]

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

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

**Paraméterek**

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

#### 4.1.2.3. 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.

**Paraméterek**

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

#### 4.1.2.4. Csapat() [4/4]

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

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

**Paraméterek**

| | |
|---|---|
| *letszam* | ez az int paraméter a csapatlétszám lesz. |

#### 4.1.2.5. ~Csapat()

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

Virtuális destruktor (mert öröklődik majd a class).

A függvény hívási gráfja:


## 4.1.3. Tagfüggvények dokumentációja

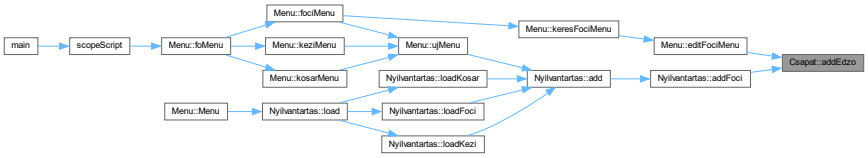
### 4.1.3.1. addEdzo()

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

Focicsapatokra vonatkozó függvénypointer.

Újraimplementáló leszármazottak: Foci.

A függvény hívó gráfja:



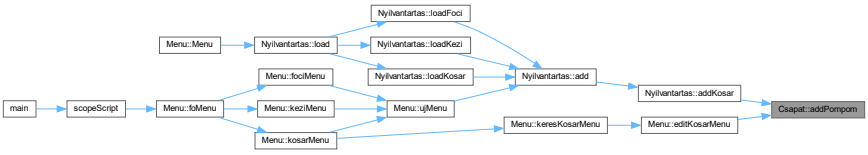

### 4.1.3.2. addPompom()

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

Kosárcsapatra vonatkozó függvénypointer.

Újraimplementáló leszármazottak: Kosar.

A függvény hívó gráfja:



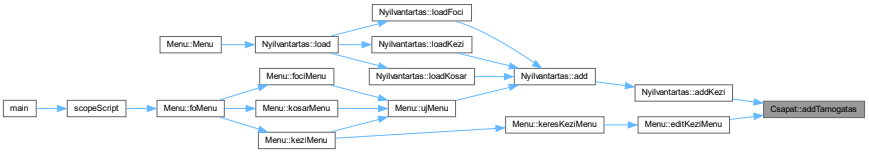
#### 4.1.3.3. addTamogatas()

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

Kézicsapatra vonatkozó függvénypointer.

Újraimplementáló leszármazottak: Kezi.

A függvény hívó gráfja:

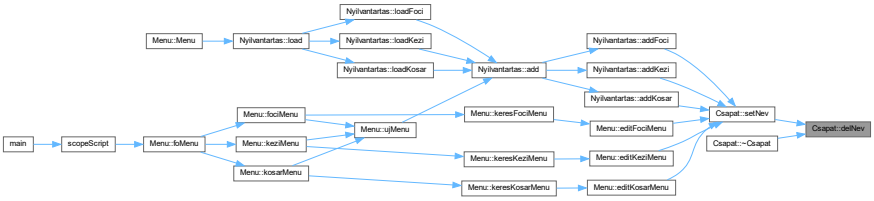



#### 4.1.3.4. delNev()

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

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

A függvény hívó gráfja:



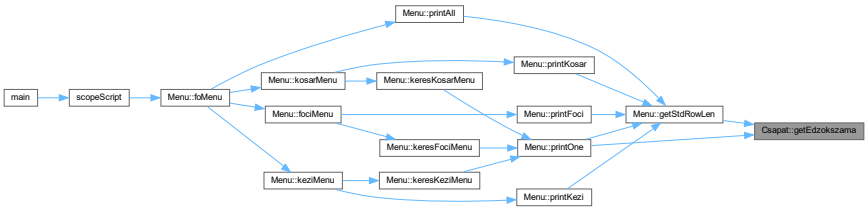

#### 4.1.3.5. getEdzokszama()

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

Focicsapatokra vonatkozó függvénypointer.

Újraimplementáló leszármazottak: Foci.

A függvény hívó gráfja:




### 4.1.3.6. getLetszam()

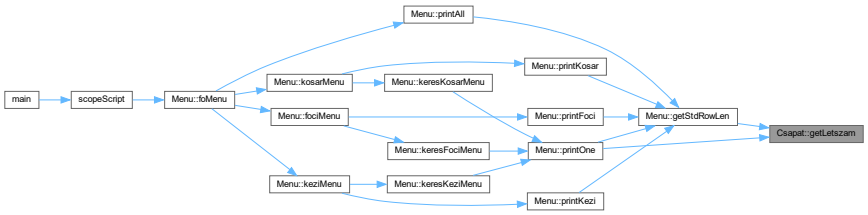
```
int Csapat::getLetszam ( ) const
```

Visszaadja a csapat létszámát.

#### Visszatérési érték

A csapat létszáma, int.

A függvény hívó gráfja:




### 4.1.3.7. getNev()

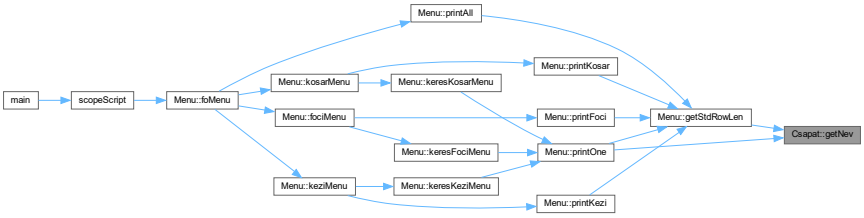
```
const char * Csapat::getNev ( ) const
```

Visszaadja a csapat nevét.

**Visszatérési érték**

A csapat neve, const char∗.

A függvény hívó gráfja:



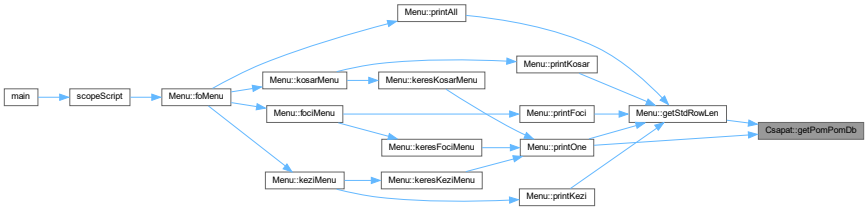

### 4.1.3.8. getPomPomDb()

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

Kosárcsapatra vonatkozó függvénypointer.

Újraimplementáló leszármazottak: Kosar.

A függvény hívó gráfja:



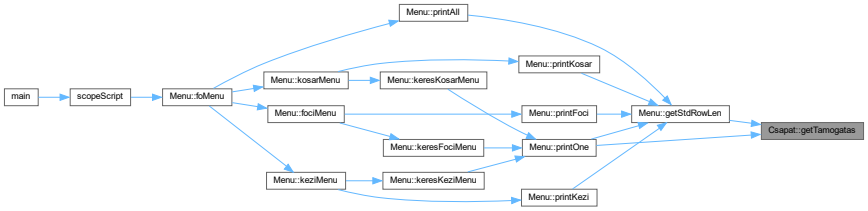

### 4.1.3.9. getTamogatas()

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

Kézicsapatra vonatkozó függvénypointer.

Újraimplementáló leszármazottak: Kezi.

A függvény hívó gráfja:




#### 4.1.3.10. getTipus()

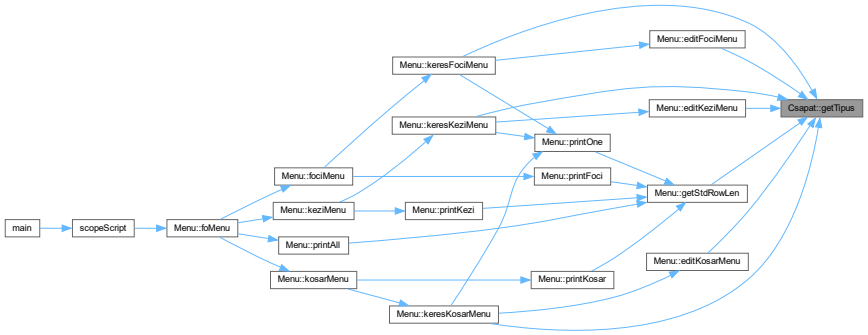
```
Tipus Csapat::getTipus ( ) const
```

Viszaadja a csapat típusát a Tipus enum segítségével.

**Visszatérési érték**

A csapat típusa a Tipus enum-ban.

A függvény hívó gráfja:




#### 4.1.3.11. 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.

**Paraméterek**

| | |
|---|---|
| *keresett_csapat_nev* | A keresett csapatnév const char∗ paraméter. |

**Visszatérési érték**

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

#### 4.1.3.12. setLetszam()

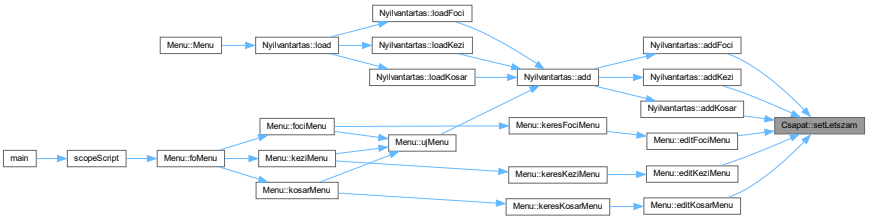
```
void Csapat::setLetszam (
            int n )
```

A csapat létszámát átállító, beállító függvény.

**Paraméterek**

| | |
|---|---|
| *uj_letszam* | Ez az int parameter lesz az új létszáma a csapatnak. |

A függvény hívó gráfja:




#### 4.1.3.13. setNev()

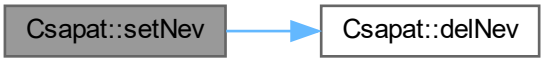
```
void Csapat::setNev (
            const char * p )
```

A csapat nevét átállító, beállító függvény.

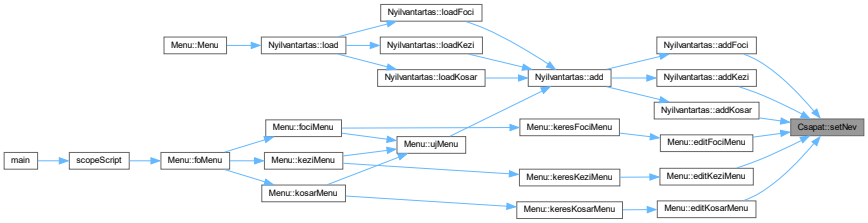
**Paraméterek**

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

A függvény hívási gráfja:




A függvény hívó gráfja:




#### 4.1.3.14. setTipus()

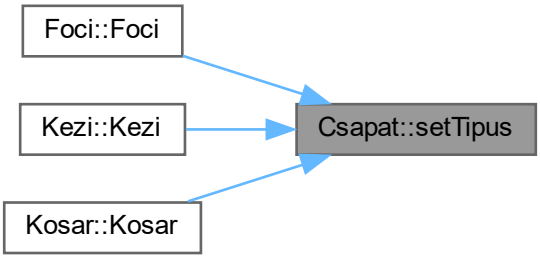
```
void Csapat::setTipus (
            const Tipus t )
```

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

**Paraméterek**

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

A függvény hívó gráfja:




### 4.1.4. Adattagok dokumentációja

#### 4.1.4.1. letszam

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

A csapat létszáma.

#### 4.1.4.2. nev

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

A csapat neve.

#### 4.1.4.3. tipus

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

A csapat típusa a Típus enum segítségével.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

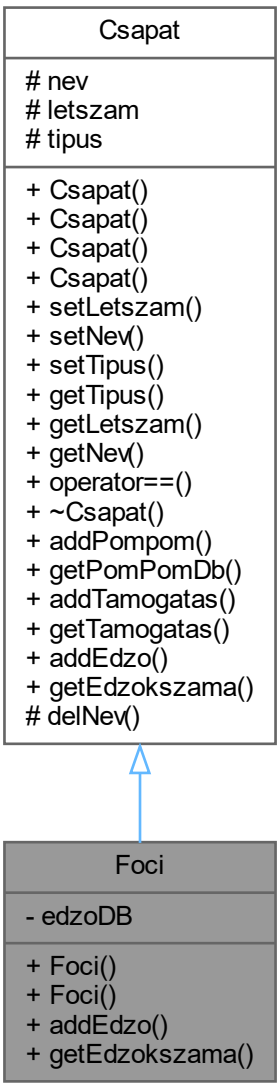
- csapat.h
- csapat.cpp

## 4.2. Foci osztályreferencia

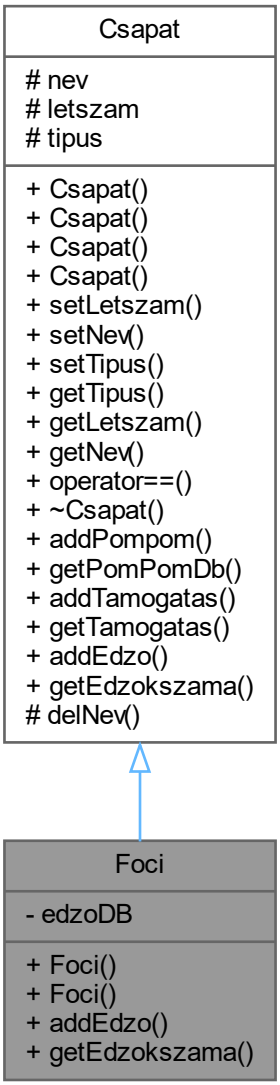
A focicsapat objektumja, amely örökli a Csapat objektum tulajdonságait.

```
#include <foci.h>
```

A Foci osztály származási diagramja:



A Foci osztály együttműködési diagramja:




## Publikus tagfüggvények

- 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.*

### Publikus tagfüggvények a(z) Csapat osztályból származnak

- 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.*

## Privát attribútumok

- int edzoDB

  *A focicsapatra specifikus Edzőket számláló változó.*

## További örökölt tagok

### Védett tagfüggvények a(z) Csapat osztályból származnak

- void delNev ()

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

**Védett attribútumok a(z) Csapat osztályból származnak**

- char * nev

  *A csapat neve.*
- int letszam

  *A csapat létszáma.*
- Tipus tipus

  *A csapat típusa a Típus enum segítségével.*

### 4.2.1. Részletes leírás

A focicsapat objektumja, amely örökli a Csapat objektum tulajdonságait.

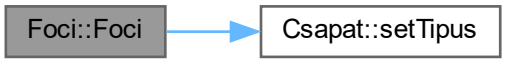
### 4.2.2. Konstruktorok és destruktorok dokumentációja

#### 4.2.2.1. 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). A függvény hívási gráfja:




#### 4.2.2.2. Foci() [2/2]

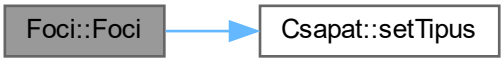
```
Foci::Foci (
            const char * p = "",
            const int n = 0 )
```

A konstruktor, amely létrehoz az adatoknak megfelelő Csapat objektumot.

**Paraméterek**

| | |
|---|---|
| *csapat_nev* | a csapat beállítandó neve. |
| *letszam* | a csapat beállítandó létszáma. |

A függvény hívási gráfja:




### 4.2.3. Tagfüggvények dokumentációja

#### 4.2.3.1. addEdzo()

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

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

**Paraméterek**

| | |
|---|---|
| *edzok_szama* | ennyivel növeli az edzoDB countert. |

Újraimplementált ősök: Csapat.

#### 4.2.3.2. getEdzokszama()

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

Visszaadja az edzők számát.

**Visszatérési érték**

Az edzők száma, int.

Újraimplementált ősök: Csapat.

### 4.2.4. Adattagok dokumentációja

#### 4.2.4.1. edzoDB

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

A focicsapatra specifikus Edzőket számláló változó.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

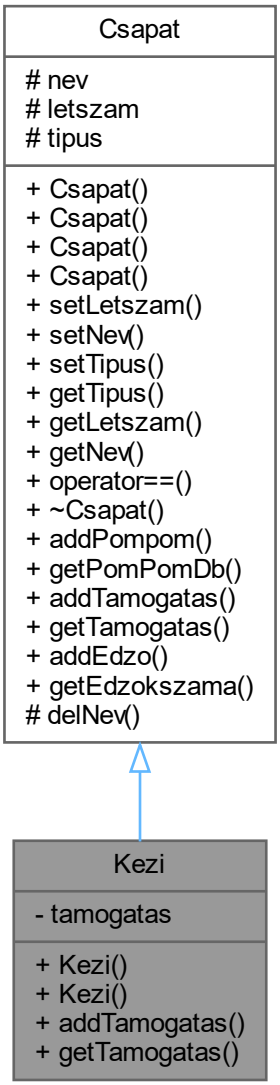
- foci.h
- foci.cpp

## 4.3. Kezi osztályreferencia

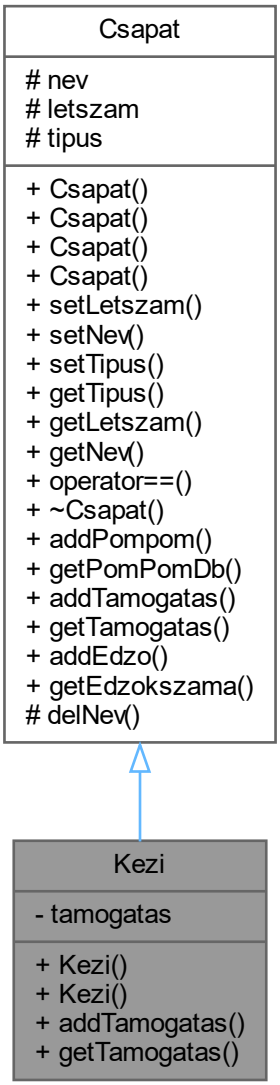
A kézilabda csapat objektumja, amely örökli a Csapat objektum tulajdonságait.

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

A Kezi osztály származási diagramja:



A Kezi osztály együttműködési diagramja:




## Publikus tagfüggvények

- 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.*

### Publikus tagfüggvények a(z) Csapat osztályból származnak

- 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.*

## Privát attribútumok

- int tamogatas

  *Az éves támogatásokat tároló int.*

## További örökölt tagok

### Védett tagfüggvények a(z) Csapat osztályból származnak

- void delNev ()

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

**Védett attribútumok a(z) Csapat osztályból származnak**

- char * nev

  *A csapat neve.*
- int letszam

  *A csapat létszáma.*
- Tipus tipus

  *A csapat típusa a Típus enum segítségével.*

### 4.3.1. Részletes leírás

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

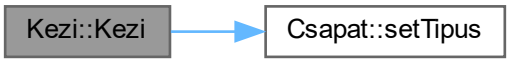
### 4.3.2. Konstruktorok és destruktorok dokumentációja

#### 4.3.2.1. Kezi() [1/2]

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

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

típust KEZI-re rakja a Típus enum segítségével. A függvény hívási gráfja:




#### 4.3.2.2. Kezi() [2/2]

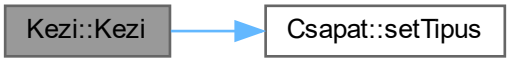
```
Kezi::Kezi (
            const char * p,
            const int n )
```

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

**Paraméterek**

| | |
|---|---|
| *uj_csapat_nev* | a csapat uj neve. |
| *letszam* | a csapat létszáma. |

A függvény hívási gráfja:




## 4.3.3. Tagfüggvények dokumentációja

### 4.3.3.1. addTamogatas()

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

A megadott számmal növeli a tamogatas változót.

**Paraméterek**

| | |
|---|---|
| *uj_tamogatas* | ennyivel növeli a támogatás változót. |

Újraimplementált ősök: Csapat.

### 4.3.3.2. getTamogatas()

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

Visszaadja a támogatások számát.

**Visszatérési érték**

A támogatások száma.

Újraimplementált ősök: Csapat.

## 4.3.4. Adattagok dokumentációja

#### 4.3.4.1. tamogatas

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

Az éves támogatásokat tároló int.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

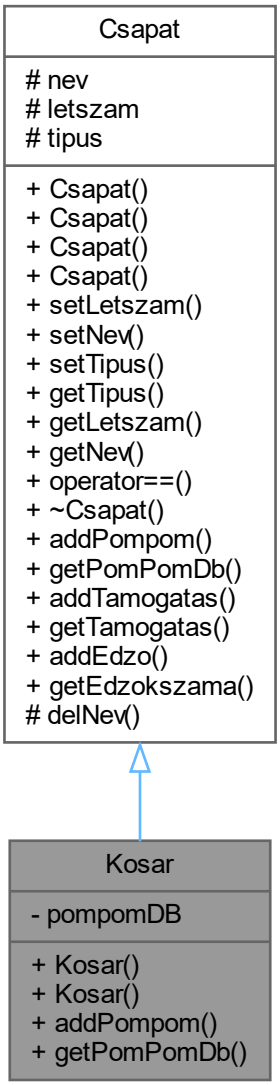
- kezi.h
- kezi.cpp

## 4.4. Kosar osztályreferencia

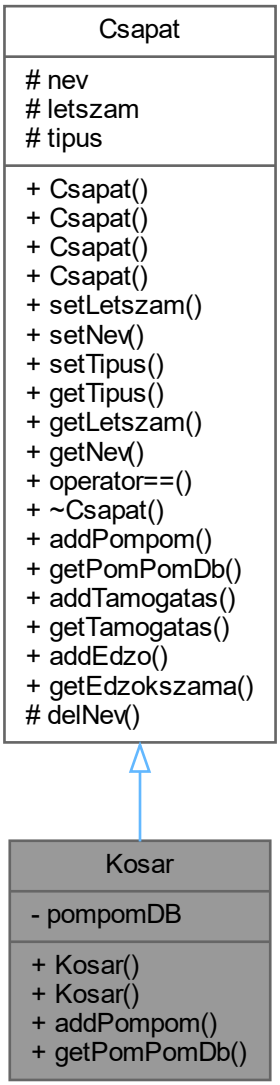
A kosárlabda csapat objektumja, amely örökli a Csapat objektum tulajdonságait.

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

A Kosar osztály származási diagramja:



A Kosar osztály együttműködési diagramja:




## Publikus tagfüggvények

- 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.*

### Publikus tagfüggvények a(z) Csapat osztályból származnak

- 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.*

## Privát attribútumok

- int pompomDB

  *A pompomlányok száma.*

## További örökölt tagok

### Védett tagfüggvények a(z) Csapat osztályból származnak

- void delNev ()

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

**Védett attribútumok a(z) Csapat osztályból származnak**

- char * nev

  *A csapat neve.*
- int letszam

  *A csapat létszáma.*
- Tipus tipus

  *A csapat típusa a Típus enum segítségével.*

### 4.4.1. Részletes leírás

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

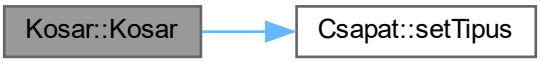
### 4.4.2. Konstruktorok és destruktorok dokumentációja

#### 4.4.2.1. Kosar() [1/2]

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

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

típust átállítja KOSAR-ra a Tipus enum segítségével. A függvény hívási gráfja:




#### 4.4.2.2. Kosar() [2/2]

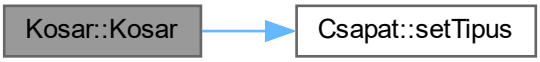
```
Kosar::Kosar (
            const char * p = "",
            const int n = 0 )
```

Az adatoknak megfelelő csapatot hoz létre.

**Paraméterek**

| | |
|---|---|
| *csapatnev* | az uj csapatnév. |
| *letszam* | az új létszám. |

A függvény hívási gráfja:




## 4.4.3. Tagfüggvények dokumentációja

### 4.4.3.1. addPompom()

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

Az adatoknak megfelelő mennyiségű pompomlányt ad a csapathoz.

**Paraméterek**

| | |
|---|---|
| *darabSzam* | ennyi pompomlány lesz a csapatban a csapathoz. |

Újraimplementált ősök: Csapat.

### 4.4.3.2. getPomPomDb()

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

Visszaadja a pompomlányok számát.

**Visszatérési érték**

A pompomlányok száma.

Újraimplementált ősök: Csapat.

## 4.4.4. Adattagok dokumentációja

#### 4.4.4.1. pompomDB

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

A pompomlányok száma.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

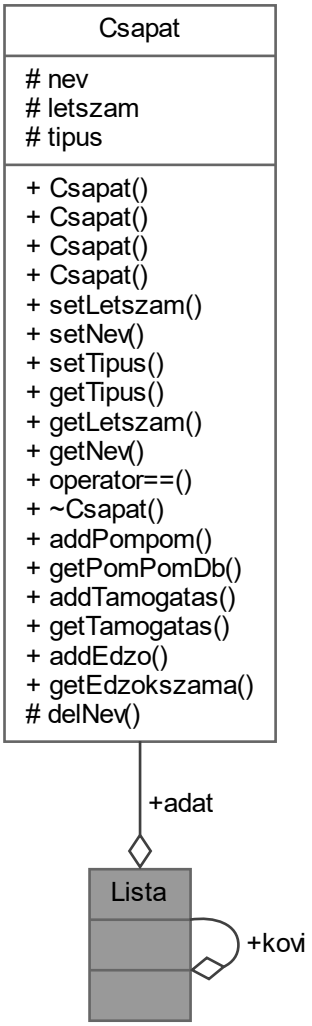
- kosar.h
- kosar.cpp

## 4.5. Lista struktúrareferencia

Láncolt listaelem.

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

A Lista osztály együttműködési diagramja:



### Publikus attribútumok

- Csapat * adat
- Lista * kovi

### 4.5.1. Részletes leírás

Láncolt listaelem.

### 4.5.2. Adattagok dokumentációja

#### 4.5.2.1. adat

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

#### 4.5.2.2. kovi

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

Ez a dokumentáció a struktúráról a következő fájl alapján készült:

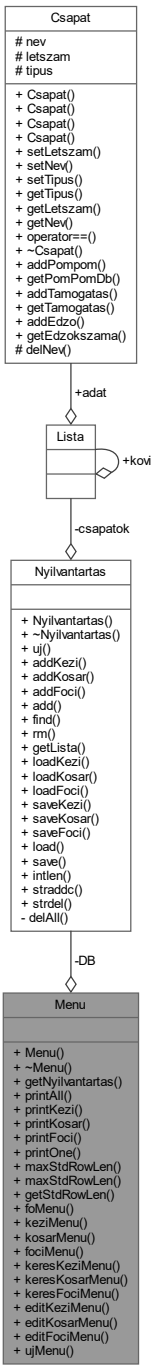
- nyilvantartas.h

## 4.6. Menu osztályreferencia

A futó programot irányító menürendszer objektuma.

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

A Menu osztály együttműködési diagramja:




## Publikus tagfüggvények

- 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.*

## Privát attribútumok

- Nyilvantartas DB

  *A nyilvántartás adatbázis.*

### 4.6.1. Részletes leírás

A futó programot irányító menürendszer objektuma.

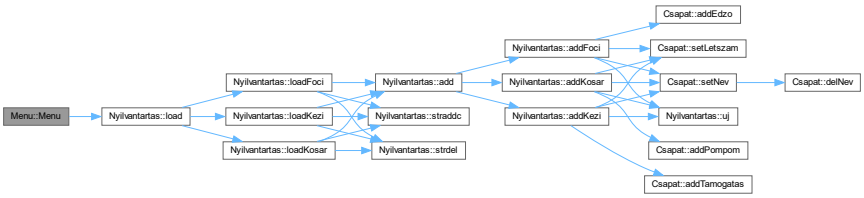
### 4.6.2. Konstruktorok és destruktorok dokumentációja

#### 4.6.2.1. Menu()

`Menu::Menu ( )`

Default konstruktor, betölti a nyilvántartás adatait fileból.

A függvény hívási gráfja:




#### 4.6.2.2. ~Menu()

`Menu::~Menu ( )`

Destruktor, lementi a nyilvántartás adatait fileokba.

A függvény hívási gráfja:


### 4.6.3. Tagfüggvények dokumentációja

#### 4.6.3.1. editFociMenu()

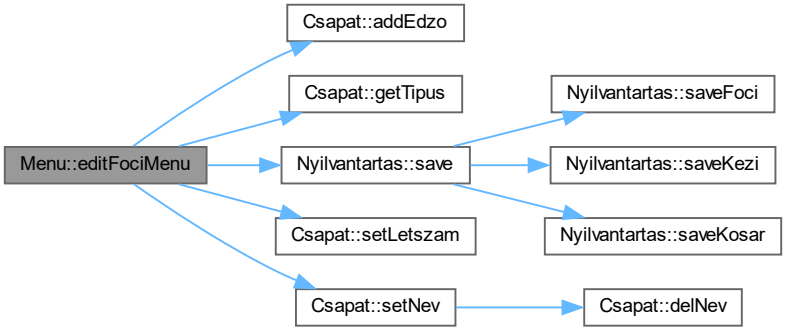
```
void Menu::editFociMenu (
            Lista * p )
```

A láncolt lista egy elemét módosító almenü.

**Paraméterek**

| | |
|---|---|
| *listaelem* | Ennek módosításában segít a menü. |

A függvény hívási gráfja:




A függvény hívó gráfja:


#### 4.6.3.2. editKeziMenu()

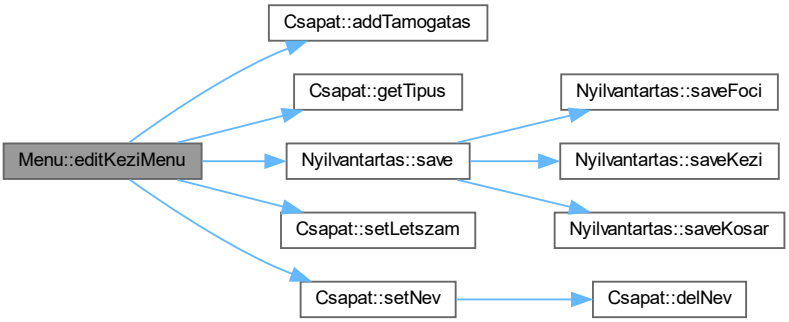
```
void Menu::editKeziMenu (
            Lista * p )
```

A láncolt lista egy elemét módosító almenü.

**Paraméterek**

| | |
|---|---|
| *listaelem* | Ennek módosításában segít a menü. |

A függvény hívási gráfja:




A függvény hívó gráfja:


#### 4.6.3.3. editKosarMenu()

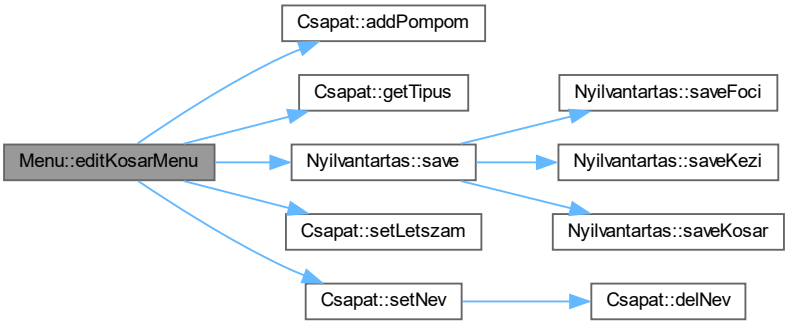
```
void Menu::editKosarMenu (
            Lista * p )
```

A láncolt lista egy elemét módosító almenü.

**Paraméterek**

| | |
|---|---|
| *listaelem* | Ennek módosításában segít a menü. |

A függvény hívási gráfja:



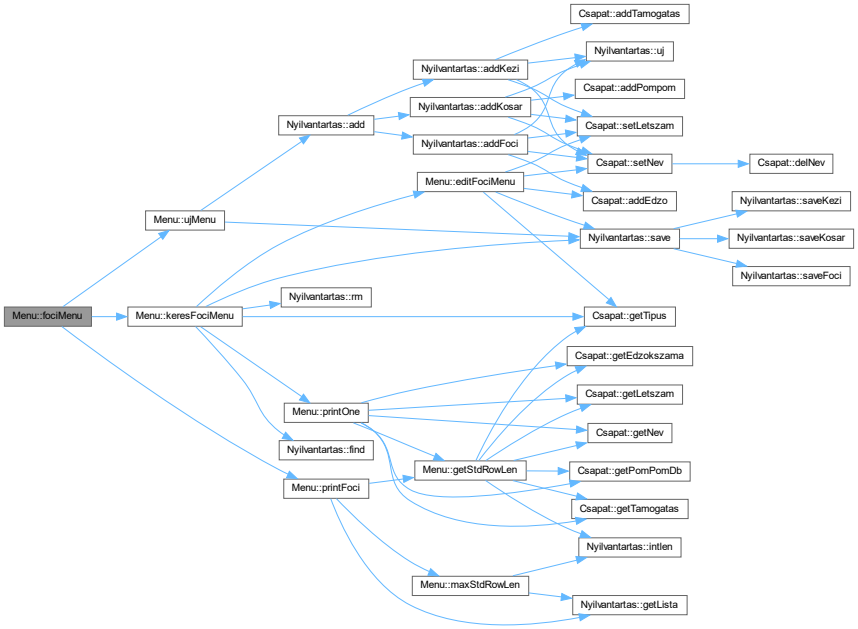

A függvény hívó gráfja:


#### 4.6.3.4. fociMenu()

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

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

A függvény hívási gráfja:



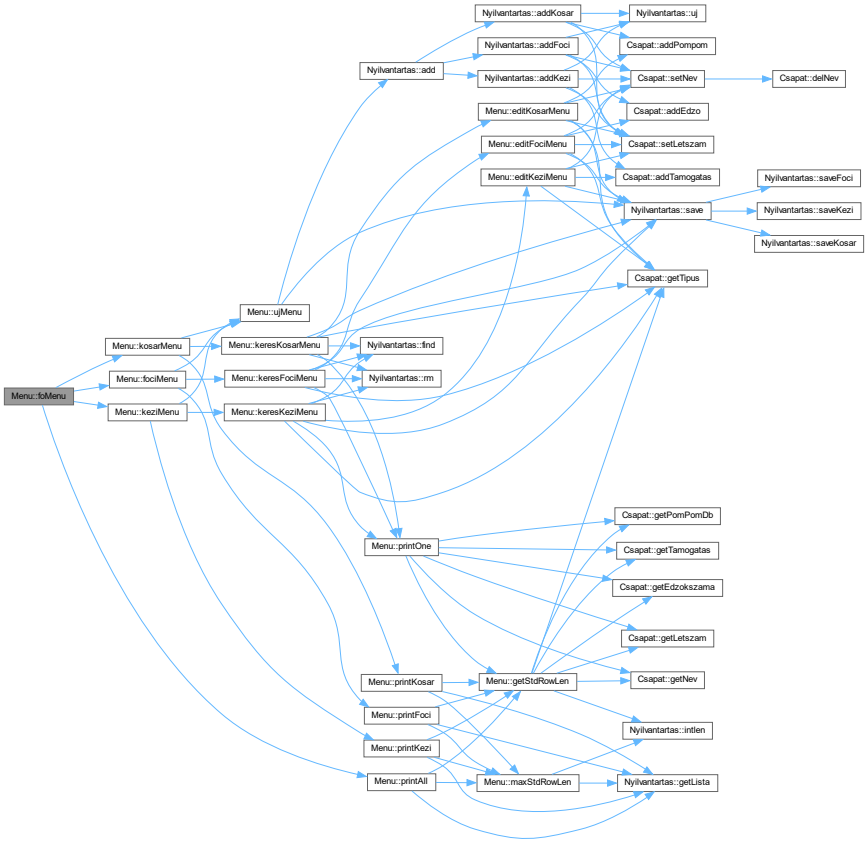
A függvény hívó gráfja:


#### 4.6.3.5. foMenu()

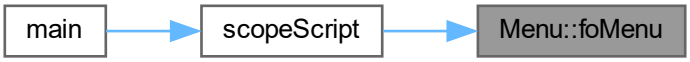
```
void Menu::foMenu ( )
```

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

átveszi az irányítást, és automata menürendszerként üzemel. A függvény hívási gráfja:



A függvény hívó gráfja:




#### 4.6.3.6. getNyilvantartas()

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

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

**Visszatérési érték**

Nyilvántartás adatbázis.

#### 4.6.3.7. getStdRowLen()

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

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

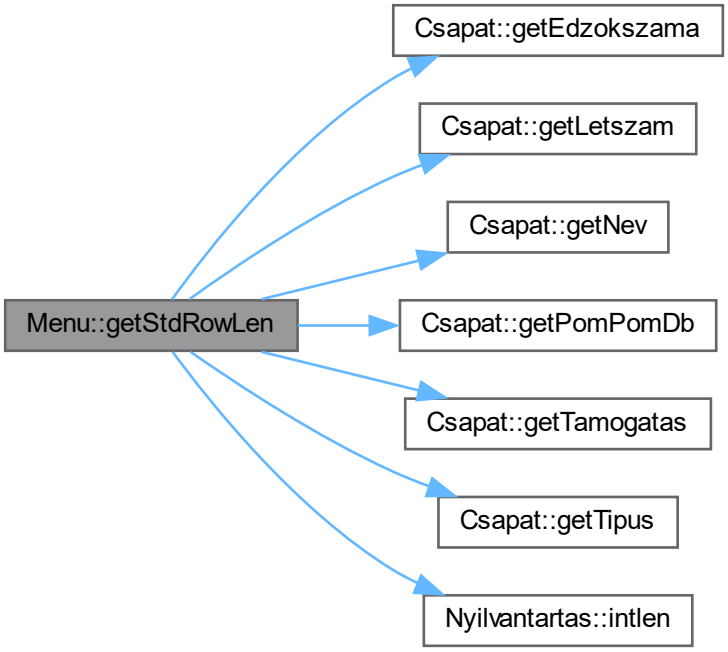
**Paraméterek**

| | |
|---|---|
| *listaelem* | Ennek a sorhosszára vagyunk kíváncsiak. |

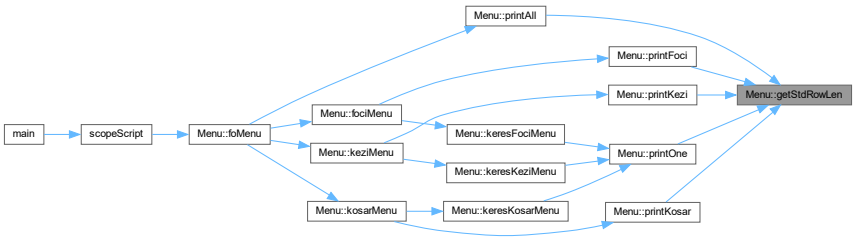
**Visszatérési érték**

A listaelem sorának hossza.

A függvény hívási gráfja:




A függvény hívó gráfja:

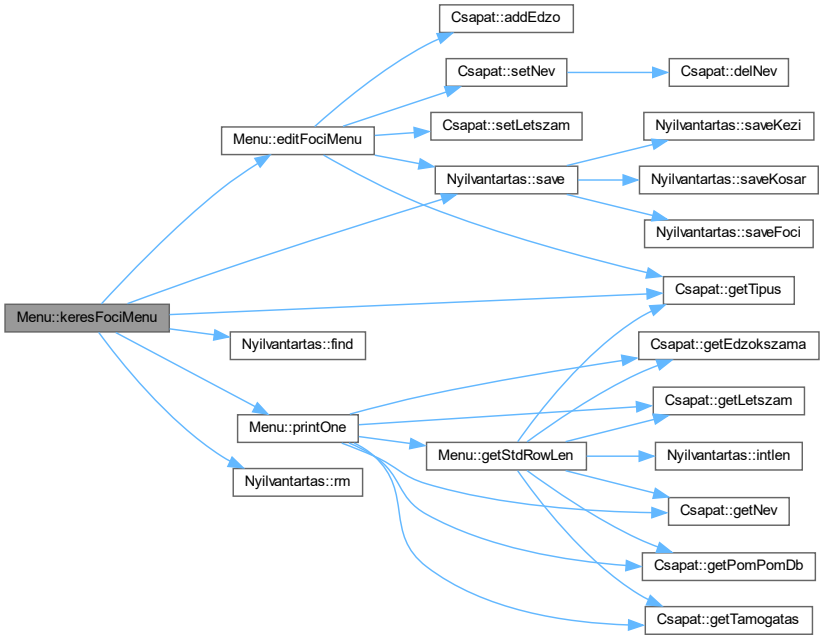



#### 4.6.3.8. keresFociMenu()

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

Egy focilabda csapat keresési menüje.

A függvény hívási gráfja:



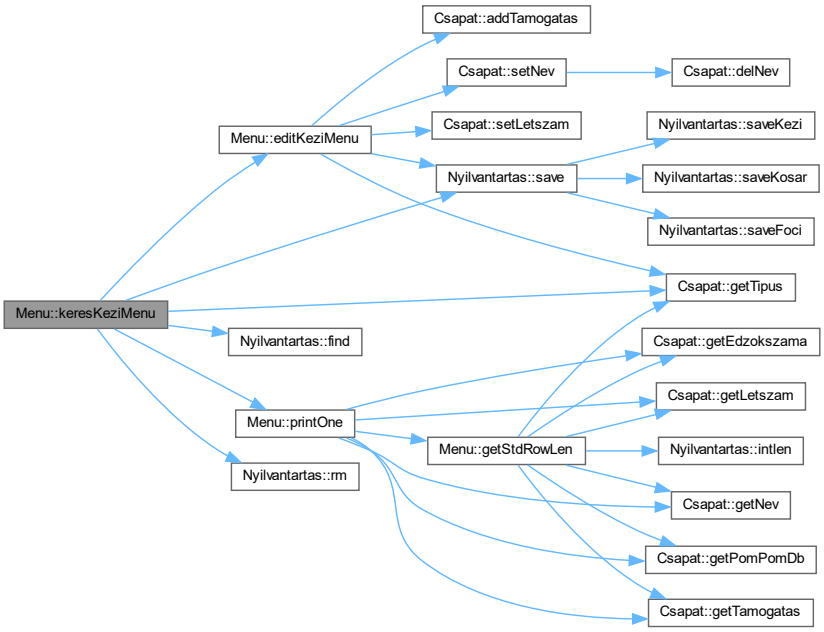

A függvény hívó gráfja:


#### 4.6.3.9. keresKeziMenu()

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

Egy kézilabda csapat keresési menüje.

A függvény hívási gráfja:



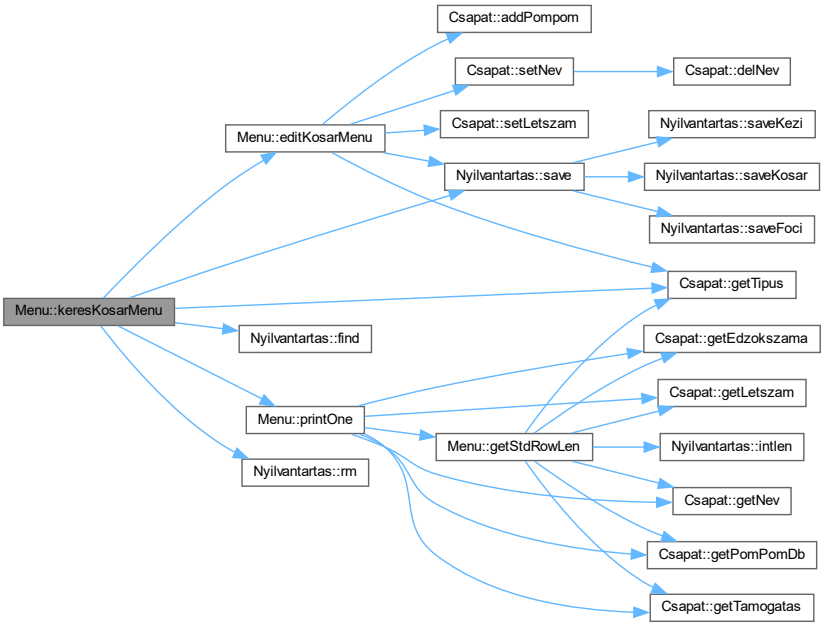

A függvény hívó gráfja:


#### 4.6.3.10. keresKosarMenu()

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

Egy kosárlabda csapat keresési menüje.

A függvény hívási gráfja:



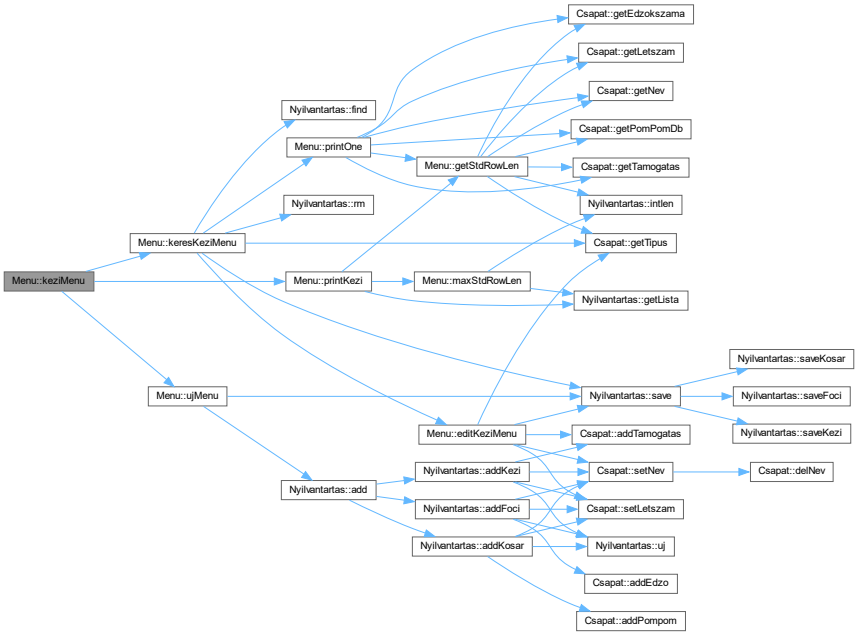

A függvény hívó gráfja:


#### 4.6.3.11. keziMenu()

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

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

A függvény hívási gráfja:



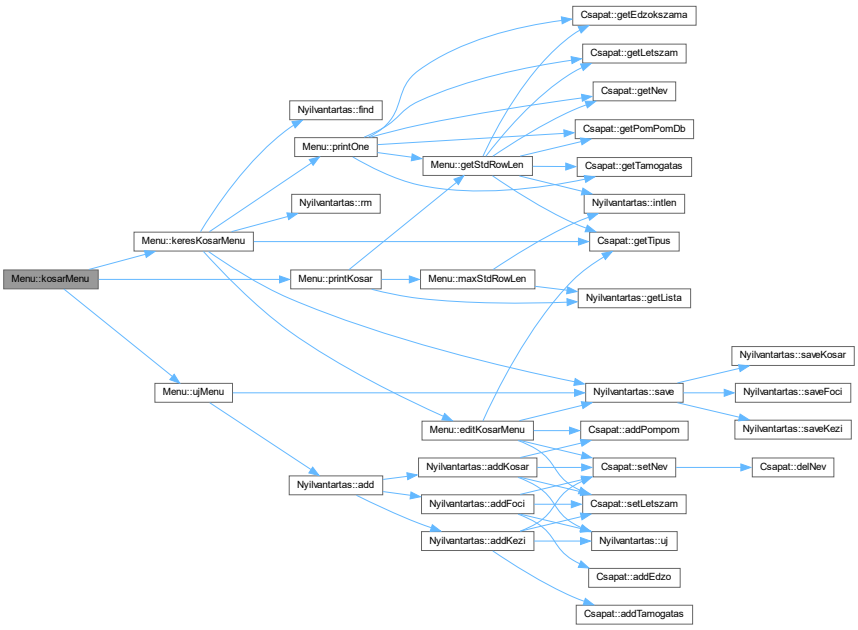

A függvény hívó gráfja:


#### 4.6.3.12. kosarMenu()

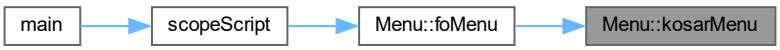
```
void Menu::kosarMenu ( )
```

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

A függvény hívási gráfja:




A függvény hívó gráfja:




#### 4.6.3.13. maxStdRowLen() [1/2]

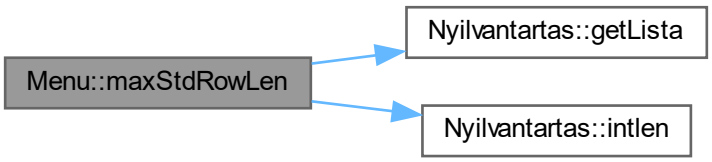
```
int Menu::maxStdRowLen ( ) const
```

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

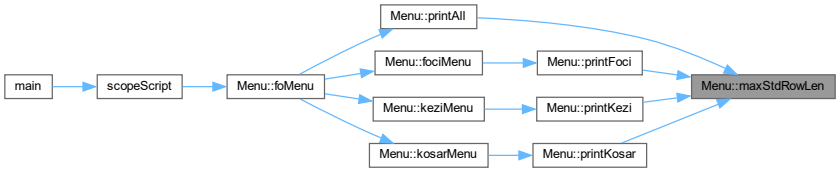
**Visszatérési érték**

A leghoszabb sor hossza az adatbáisban.

A függvény hívási gráfja:




A függvény hívó gráfja:




#### 4.6.3.14. 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.

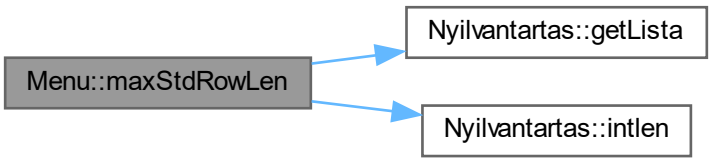
**Paraméterek**

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

**Visszatérési érték**

A maximális sor hossz.

A függvény hívási gráfja:

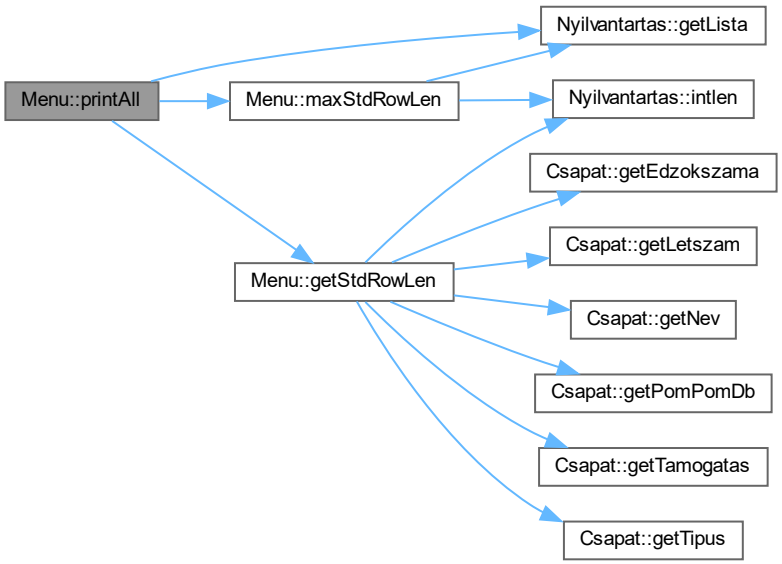



#### 4.6.3.15. printAll()

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

Kilistázza megformázva az összes adatot.

A függvény hívási gráfja:



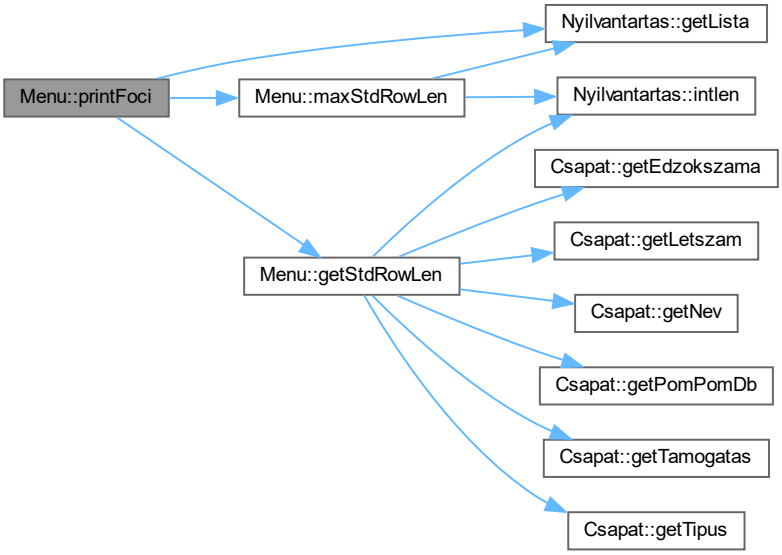
A függvény hívó gráfja:


#### 4.6.3.16. printFoci()

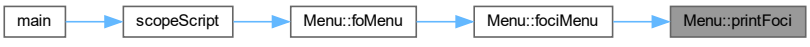
```
void Menu::printFoci ( ) const
```

Kilistázza megformázva a Focilabda (Foci) csapatokat.

A függvény hívási gráfja:




A függvény hívó gráfja:

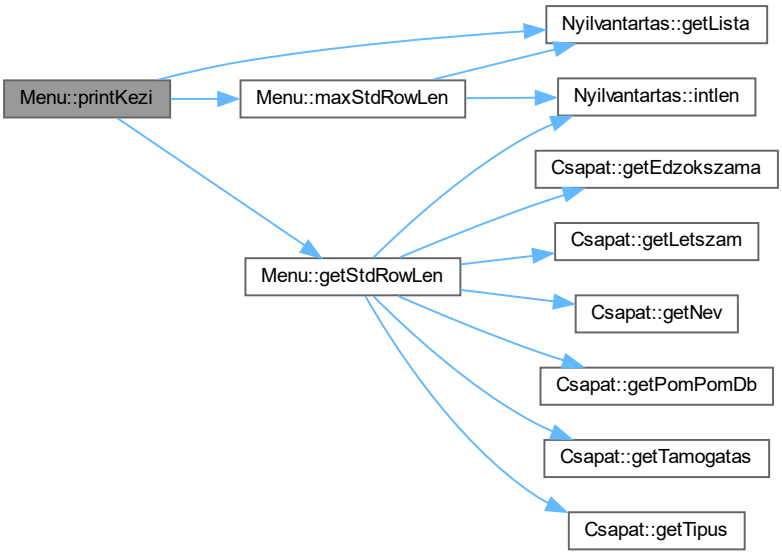


#### 4.6.3.17. printKezi()

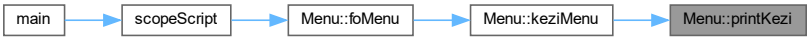
```
void Menu::printKezi ( ) const
```

Kilistázza megformázva a Kézilabda (Kezi) csapatokat.

A függvény hívási gráfja:




A függvény hívó gráfja:

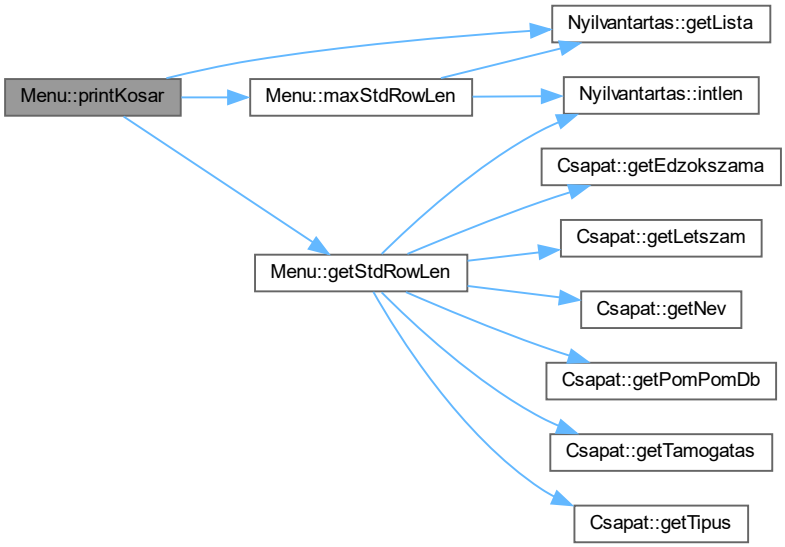



#### 4.6.3.18. printKosar()

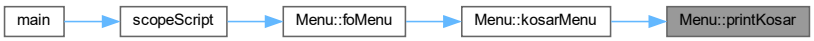
```
void Menu::printKosar ( ) const
```

Kilistázza megformázva a Kárlabda (Kosar) csapatokat.

A függvény hívási gráfja:




A függvény hívó gráfja:




#### 4.6.3.19. printOne()

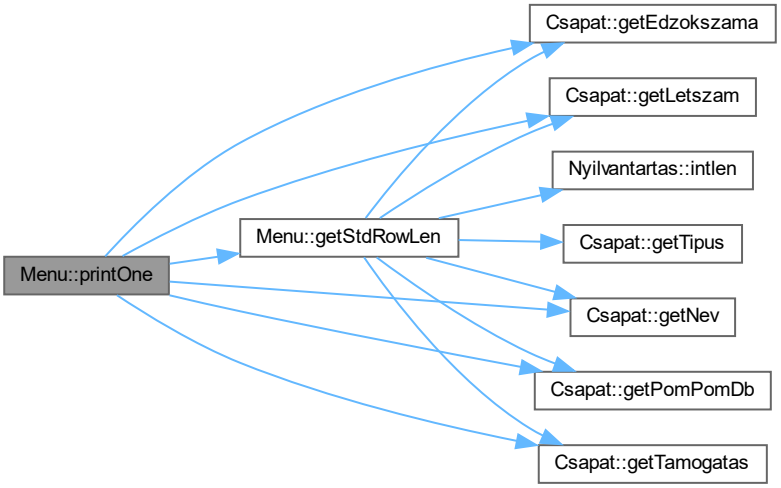
```
void Menu::printOne (
            Lista * p,
            Tipus t ) const
```

Egy listaelemet ír ki megformázva.

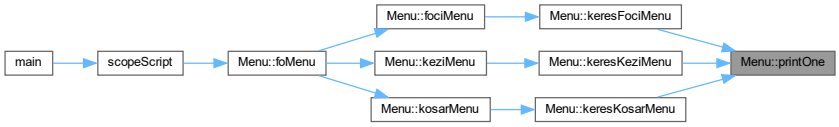
**Paraméterek**

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

A függvény hívási gráfja:




A függvény hívó gráfja:




#### 4.6.3.20. ujMenu()

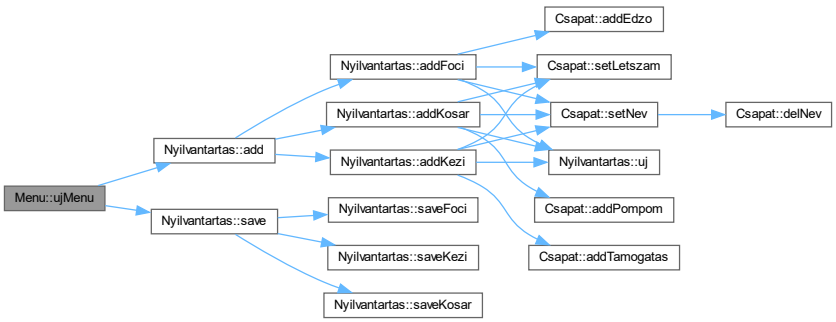
```
void Menu::ujMenu (
            Tipus T )
```

Egy új tipus típusú csapat lérehozására létező almenü.

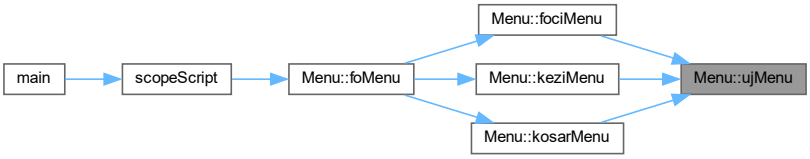
**Paraméterek**

| | |
|---|---|
| *tipus* | Az új csapat típusa. |

A függvény hívási gráfja:




A függvény hívó gráfja:




## 4.6.4. Adattagok dokumentációja

### 4.6.4.1. DB

```
Nyilvantartas Menu::DB  [private]
```

A nyilvántartás adatbázis.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

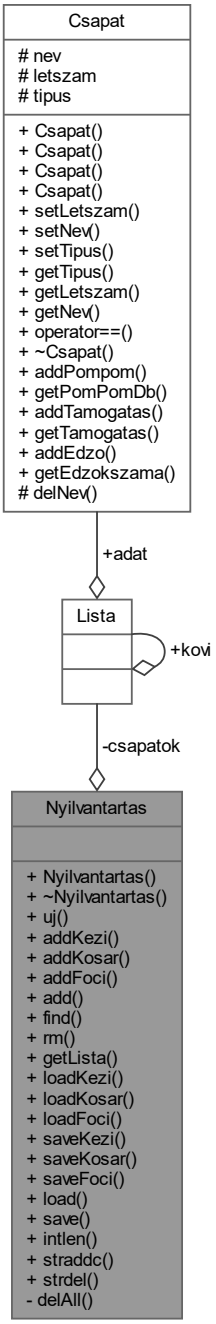
- menu.h
- menu.cpp

# 4.7. Nyilvantartas osztályreferencia

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

`#include <nyilvantartas.h>`

A Nyilvantartas osztály együttműködési diagramja:



## Publikus tagfüggvények

- 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.*

## Statikus publikus tagfüggvények

- 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.*

## Privát tagfüggvények

- void delAll ()

  *A kézilabda csapatokat tároló láncolt listát felszabadító segédfüggvény.*

## Privát attribútumok

- Lista * csapatok

  *A csapatokat tároló lista.*

### 4.7.1. Részletes leírás

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).

### 4.7.2. Konstruktorok és destruktorok dokumentációja

#### 4.7.2.1. Nyilvantartas()

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

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

láncolt listákat. (mindegyiket nullptr-re rakja).

#### 4.7.2.2. ~Nyilvantartas()

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

A destruktor, felszabadítja a láncolt listákat a segédfüggvények segítségével.

A függvény hívási gráfja:

## 4.7.3. Tagfüggvények dokumentációja

### 4.7.3.1. add()

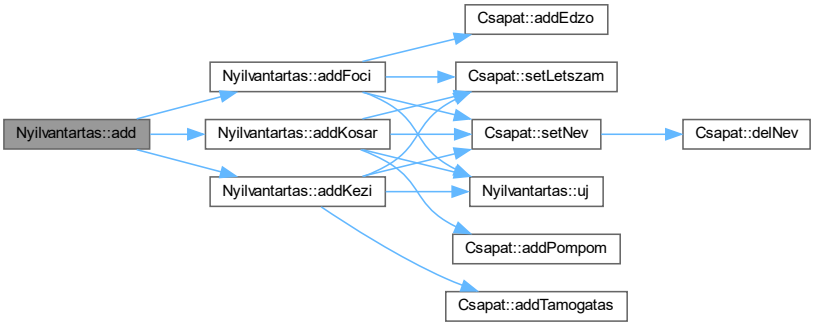
```
void Nyilvantartas::add (
            const Tipus T,
            const char * csnev,
            const int letszam,
            const int vari )
```

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

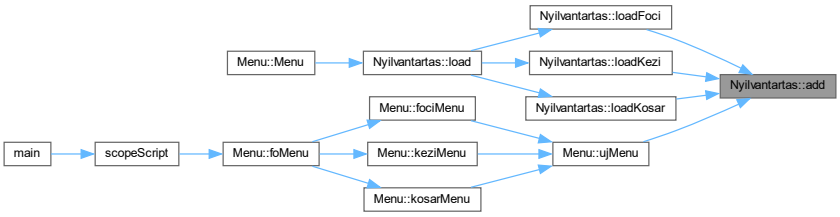
**Paraméterek**

| | |
|---|---|
| *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). |

A függvény hívási gráfja:




A függvény hívó gráfja:



#### 4.7.3.2. addFoci()

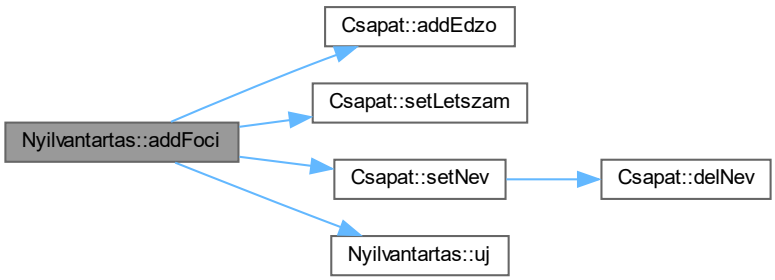
```
void Nyilvantartas::addFoci (
            const char * csnev,
            const int letszam,
            const int edzok )
```

Beleláncol a listába a paramétereknek megfelelő Focilabda csapatot.

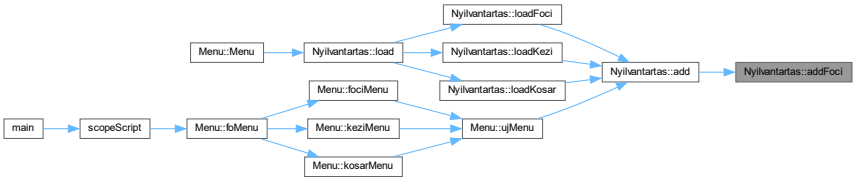
**Paraméterek**

| | |
|---|---|
| *csapatnev* | a csapat neve. |
| *letszam* | a csapat létszáma. |
| *edzok* | a csapat edzőinek száma. |

A függvény hívási gráfja:




A függvény hívó gráfja:




#### 4.7.3.3. addKezi()

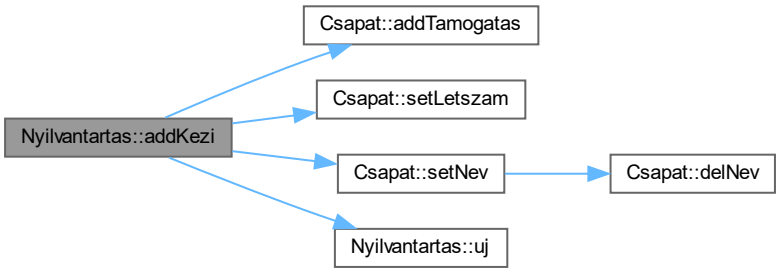
```
void Nyilvantartas::addKezi (
            const char * csnev,
            const int letszam,
            const int tamogatas )
```

Beleláncol a listába a paramétereknek megfelelő Kézilabda csapatot.

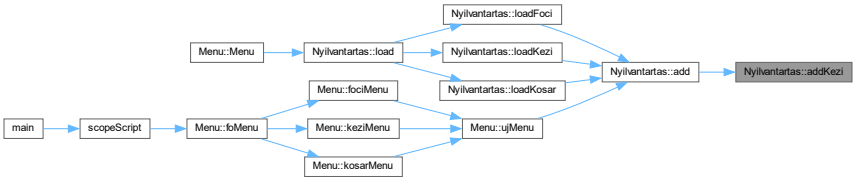
**Paraméterek**

| | |
|---|---|
| *csapatnev* | a csapat neve. |
| *letszam* | a csapat létszáma. |
| *tamogatas* | a csapat támogatásai. |

A függvény hívási gráfja:




A függvény hívó gráfja:




#### 4.7.3.4. addKosar()

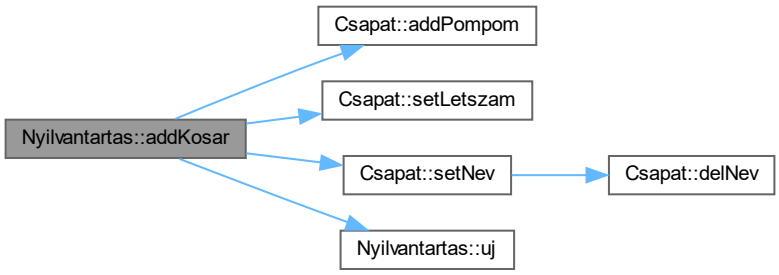
```
void Nyilvantartas::addKosar (
            const char * csnev,
            const int letszam,
            const int pompomn )
```

Beleláncol a listába a paramétereknek megfelelő Kosárlabda csapatot.

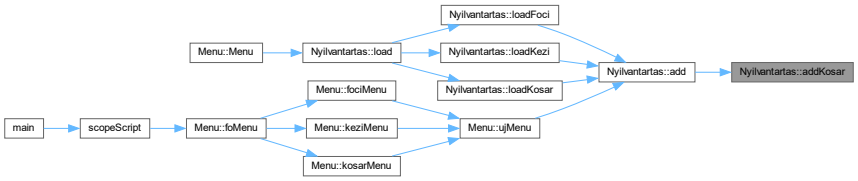
**Paraméterek**

| | |
|---|---|
| *csapatnev* | a csapat neve. |
| *letszam* | a csapat létszáma. |
| *pompom_lanyok* | a csapat pompom lányainak száma. |

A függvény hívási gráfja:




A függvény hívó gráfja:

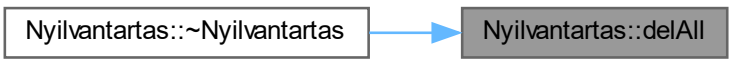



#### 4.7.3.5. delAll()

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

A kézilabda csapatokat tároló láncolt listát felszabadító segédfüggvény.

A függvény hívó gráfja:




#### 4.7.3.6. find()

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

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

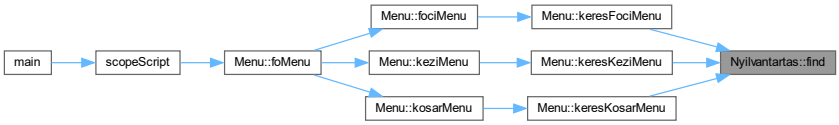
**Paraméterek**

| | |
|---|---|
| *csapat_nev* | a keresendő csapat neve. |

**Visszatérési érték**

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

A függvény hívó gráfja:




#### 4.7.3.7. getLista()

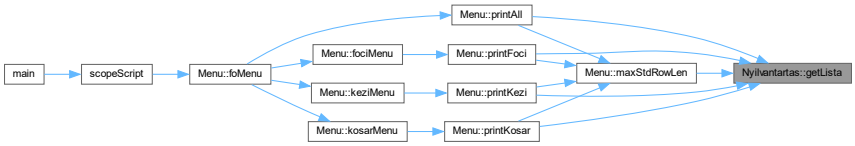
```
Lista * Nyilvantartas::getLista ( ) const
```

Visszaadja a csapatok láncolt listáját.

**Visszatérési érték**

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

A függvény hívó gráfja:




#### 4.7.3.8. 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...)

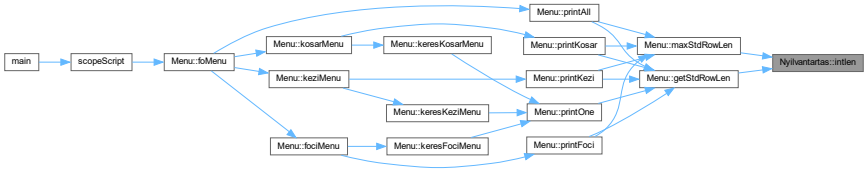
**Paraméterek**

| | |
|---|---|
| *szam* | a kiszámolandó szám. |

**Visszatérési érték**

A szám legnagyobb helyiértéke.

A függvény hívó gráfja:




### 4.7.3.9. load()

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

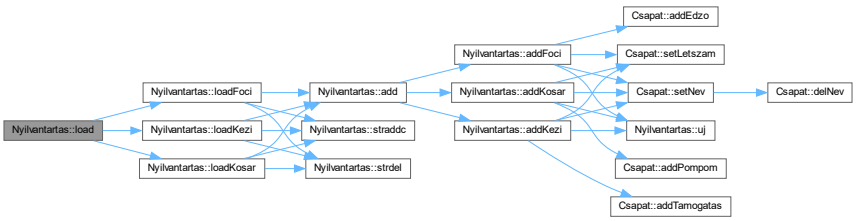
Betölti az összes fileból (kezi.txt, kosar.txt, foci.txt) az adatokat a.

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

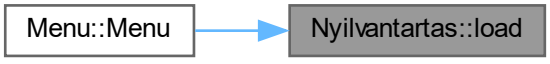
**Visszatérési érték**

igaz ha sikeres, hamis ha nem sikeres.

A függvény hívási gráfja:




A függvény hívó gráfja:



#### 4.7.3.10. loadFoci()

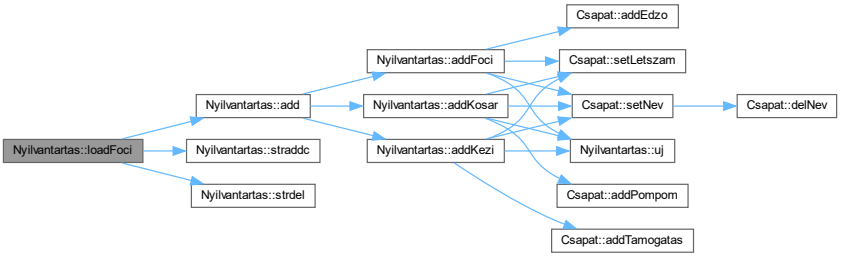
```
bool Nyilvantartas::loadFoci ( )
```

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

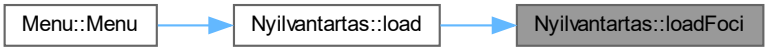
**Visszatérési érték**

Igaz, ha sikerült betölteni, hamis ha nem.

A függvény hívási gráfja:




A függvény hívó gráfja:




#### 4.7.3.11. loadKezi()

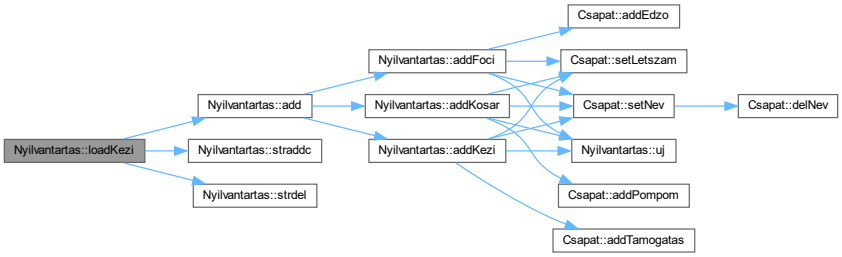
```
bool Nyilvantartas::loadKezi ( )
```

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

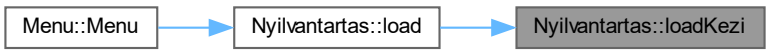
**Visszatérési érték**

Igaz, ha sikerült betölteni, hamis ha nem.

A függvény hívási gráfja:




A függvény hívó gráfja:




#### 4.7.3.12. loadKosar()

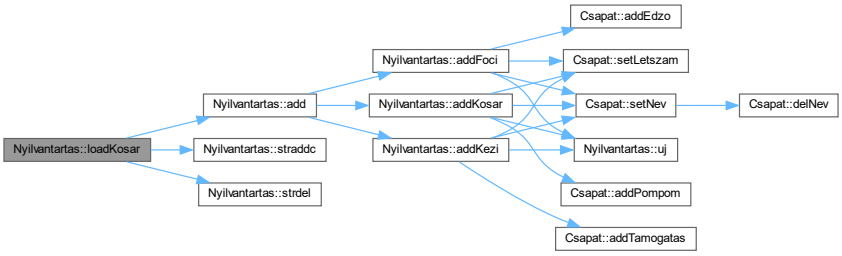
```
bool Nyilvantartas::loadKosar ( )
```

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

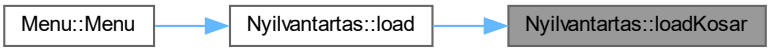
**Visszatérési érték**

Igaz, ha sikerült betölteni, hamis ha nem.

File leírás:A függvény hívási gráfja:



A függvény hívó gráfja:




#### 4.7.3.13. rm()

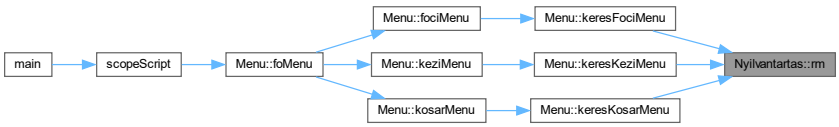
```
void Nyilvantartas::rm (
            Lista *& torlendo )
```

Kitörli a láncolt listából a paraméterben megadott listaelemet.

**Paraméterek**

| | |
|---|---|
| *lista_elem* | a láncolt listában egy elem-re mutató pointer. |

A függvény hívó gráfja:

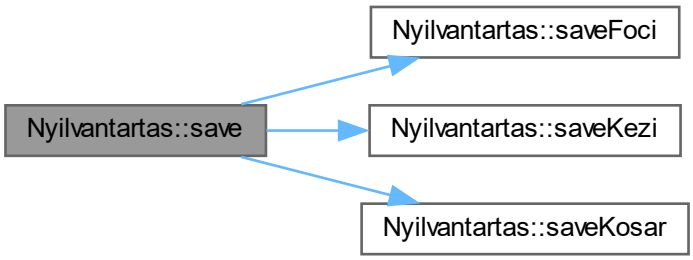



#### 4.7.3.14. save()

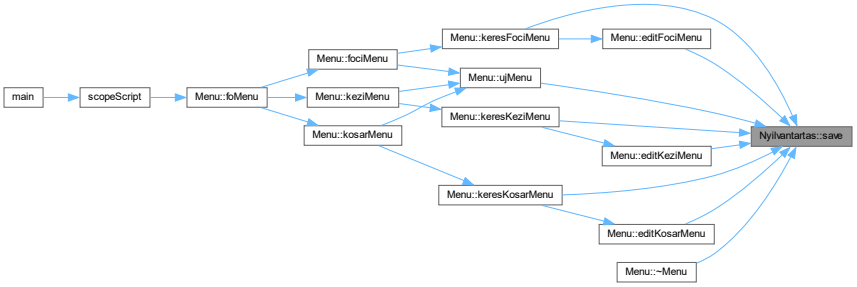
```
void Nyilvantartas::save ( ) const
```

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

(kezi.txt, kosar.txt, foci.txt) a segédfüggvények segítségével. A függvény hívási gráfja:




A függvény hívó gráfja:

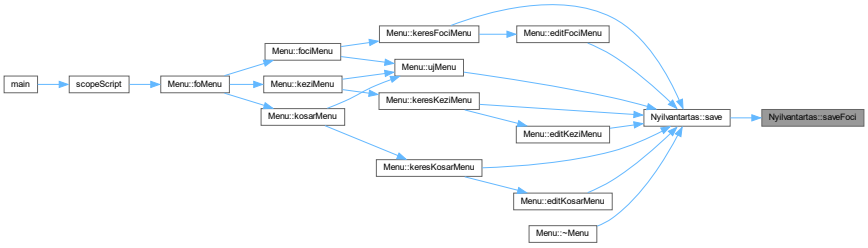



#### 4.7.3.15. saveFoci()

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

Elmenti a foci.txt fileba a fociCS adatait.

A függvény hívó gráfja:

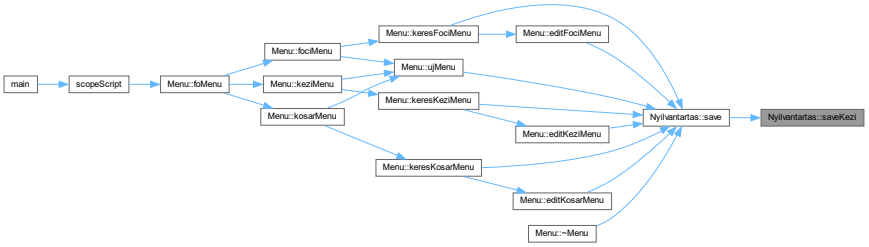


#### 4.7.3.16. saveKezi()

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

Elmenti a kezi.txt fileba a keziCS adatait.

A függvény hívó gráfja:

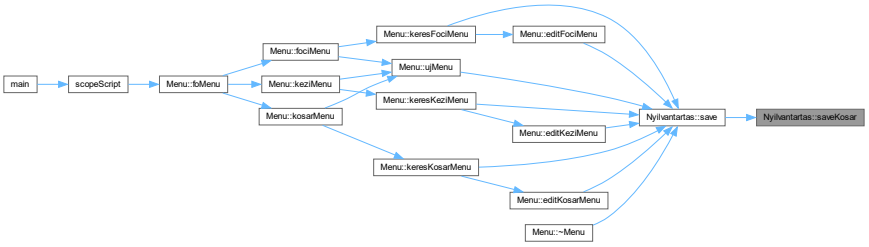



#### 4.7.3.17. saveKosar()

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

Elmenti a kosar.txt fileba a kosarCS adatait.

A függvény hívó gráfja:




#### 4.7.3.18. straddc()

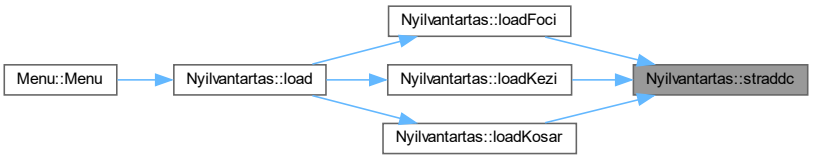
```
void Nyilvantartas::straddc (
            char *& str,
            const char c )  [static]
```

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

**Paraméterek**

| | |
|---|---|
| *str* | A karakterpointer. |
| *c* | A hozzáadandó betű. |

A függvény hívó gráfja:




#### 4.7.3.19. strdel()

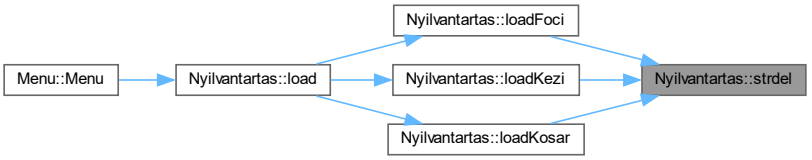
```
void Nyilvantartas::strdel (
            char *& str )  [static]
```

Felszabadít és nullptr-é tesz egy karakterpointert. Statikus.

**Paraméterek**

| | |
|---|---|
| *str* | A karakertpointer. |

A függvény hívó gráfja:




#### 4.7.3.20. 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.)

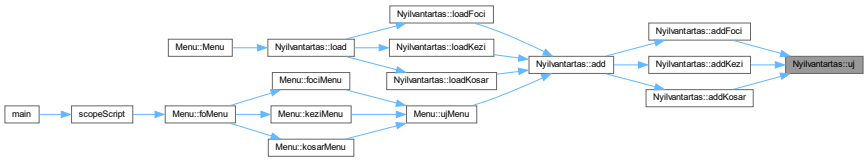
**Paraméterek**

| | |
|---|---|
| *tipus* | Az új csapat típusa |

**Visszatérési érték**

Az újjonnan létrehozott láncolt listaelem pointere.

A függvény hívó gráfja:




## 4.7.4. Adattagok dokumentációja

### 4.7.4.1. csapatok

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

A csapatokat tároló lista.

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- nyilvantartas.h
- nyilvantartas.cpp

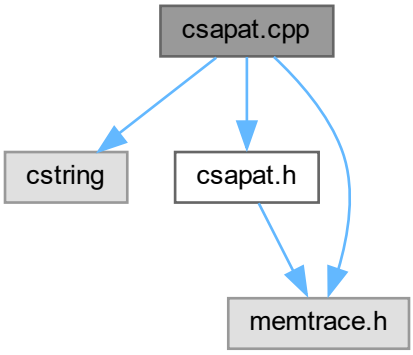
# 5. fejezet

# Fájlok dokumentációja

## 5.1. csapat.cpp fájlreferencia

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

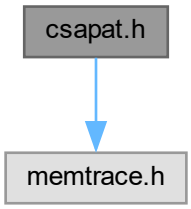
A csapat.cpp definíciós fájl függési gráfja:



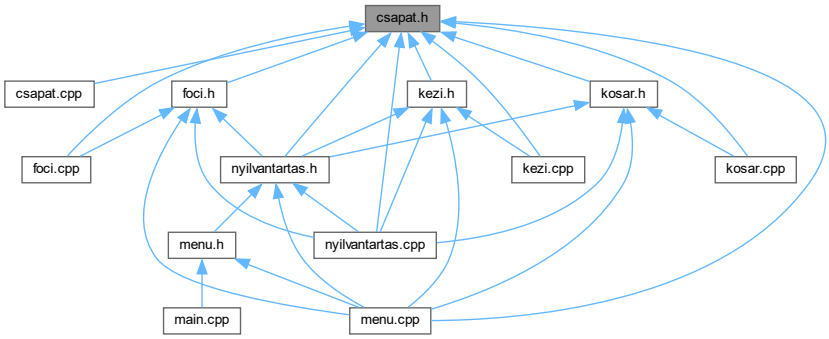

## 5.2. csapat.h fájlreferencia

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

A csapat.h definíciós fájl függési gráfja:




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




## Osztályok

- class Csapat

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

## Enumerációk

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

  *A csapatok típusait tartalmazó enum.*

## 5.2.1. Enumerációk dokumentációja

#### 5.2.1.1. Tipus

`enum Tipus`

A csapatok típusait tartalmazó enum.

**Enumeráció-értékek**

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

## 5.3. csapat.h

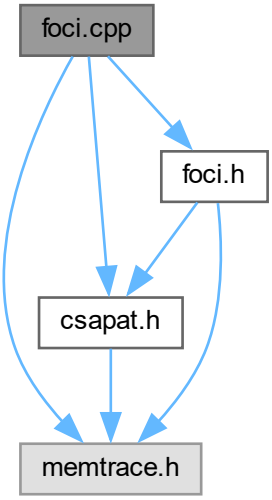
Ugrás a fájl dokumentációjához.

```
00001
00002
00003 #ifndef CSAPAT_H
00004 #define CSAPAT_H
00005
00006 #include "memtrace.h"
00007
00009 enum Tipus {
00011   NINCS,
00012
00014   KEZI,
00015
00017   FOCI,
00018
00020   KOSAR
00021 };
00022
00025 class Csapat {
00026   protected:
00028     char *nev;
00029
00031     int letszam;
00032
00034     Tipus tipus;
00035
00037     void delNev() { if (nev != nullptr) { delete[] nev; } };
00038
00039   public:
00041     Csapat() : nev(nullptr), letszam(0), tipus(NINCS) {};
00042
00046     Csapat(const char *, int);
00047
00050     Csapat(const char *);
00051
00054     Csapat(int);
00055
00058     void setLetszam(int);
00059
00062     void setNev(const char *);
00063
00066     void setTipus(const Tipus);
00067
00070     Tipus getTipus() const;
00071
00074     int getLetszam() const;
00075
00078     const char *getNev() const;
00079
00085     bool operator==(const char*);
00086
00088     virtual ~Csapat();
00089
00091     virtual void addPompom(const int) {};
00092
00094     virtual const int getPomPomDb() const {return -1;}
00095
00097     virtual void addTamogatas(const int) {}
00098
00100     virtual const int getTamogatas() const {return -1;}
00101
00103     virtual void addEdzo(const int) {}
00104
00106     virtual const int getEdzokszama() const {return -1;}
00107 };
00108
00109 #endif
```

## 5.4. foci.cpp fájlreferencia

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

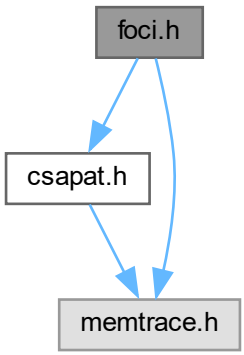
A foci.cpp definíciós fájl függési gráfja:



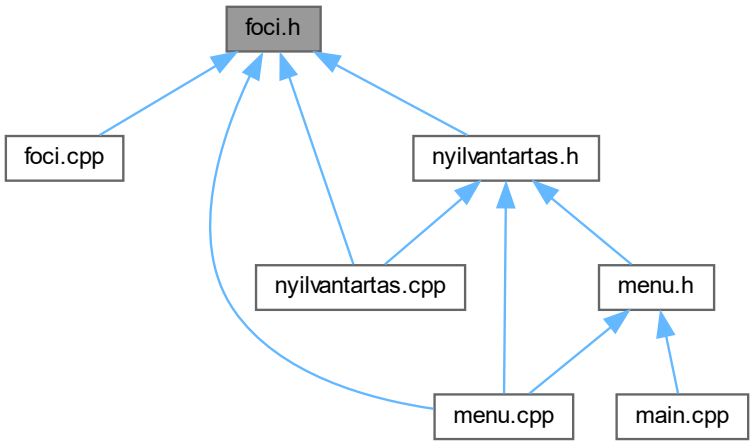

## 5.5. foci.h fájlreferencia

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

A foci.h definíciós fájl függési gráfja:



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




### Osztályok

- class Foci

  *A focicsapat objektumja, amely örökli a Csapat objektum tulajdonságait.*

## 5.6. foci.h

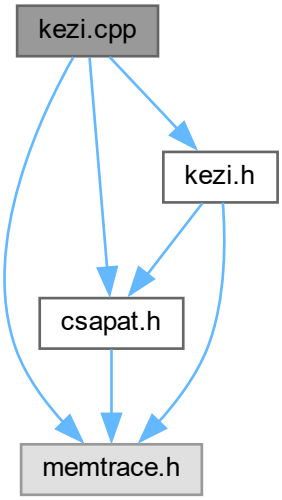
Ugrás a fájl dokumentációjához.

```
00001
00002
00003 #ifndef FOCI_H
00004 #define FOCI_H
00005
00006 #include "csapat.h"
00007 #include "memtrace.h"
00008
00010 class Foci : public Csapat {
00011   private:
00013     int edzoDB;
00014
00015   public:
00018     Foci();
00019
00023     Foci(const char *, const int);
00024
00027     void addEdzo(const int);
00028
00031     const int getEdzokszama() const;
00032
00033 };
00034
00035 #endif
```

## 5.7. kezi.cpp fájlreferencia

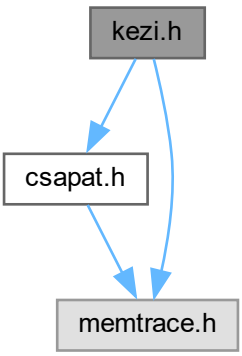
```
#include "csapat.h"
#include "kezi.h"
```

```
#include "memtrace.h"
```
A kezi.cpp definíciós fájl függési gráfja:

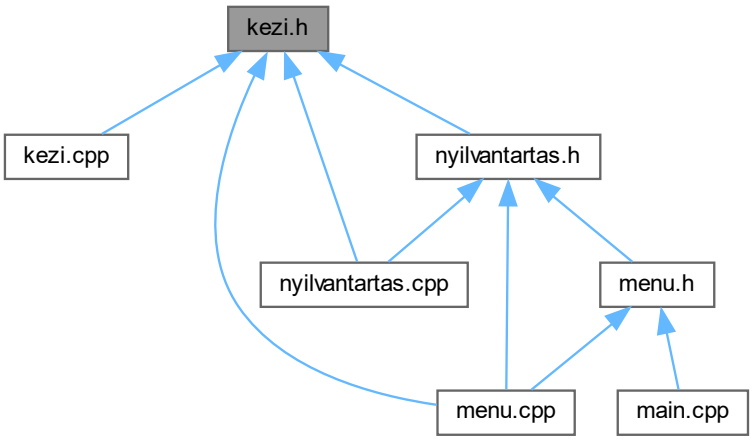



## 5.8. kezi.h fájlreferencia

```
#include "csapat.h"
#include "memtrace.h"
```
A kezi.h definíciós fájl függési gráfja:



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




### Osztályok

- class Kezi

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

## 5.9. kezi.h

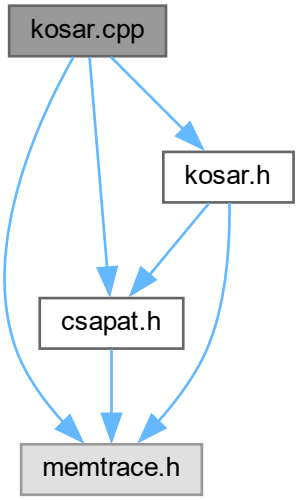
Ugrás a fájl dokumentációjához.

```
00001
00002
00003 #ifndef KEZI_H
00004 #define KEZI_H
00005
00006 #include "csapat.h"
00007 #include "memtrace.h"
00008
00010 class Kezi : public Csapat {
00011   private:
00013     int tamogatas;
00014
00015   public:
00018     Kezi();
00019
00023     Kezi(const char*, const int);
00024
00027     void addTamogatas(const int);
00028
00031     const int getTamogatas() const;
00032 };
00033
00034 #endif
```

## 5.10. kosar.cpp fájlreferencia

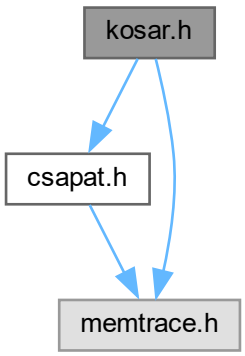
```
#include "csapat.h"
#include "kosar.h"
```

```
#include "memtrace.h"
```
A kosar.cpp definíciós fájl függési gráfja:

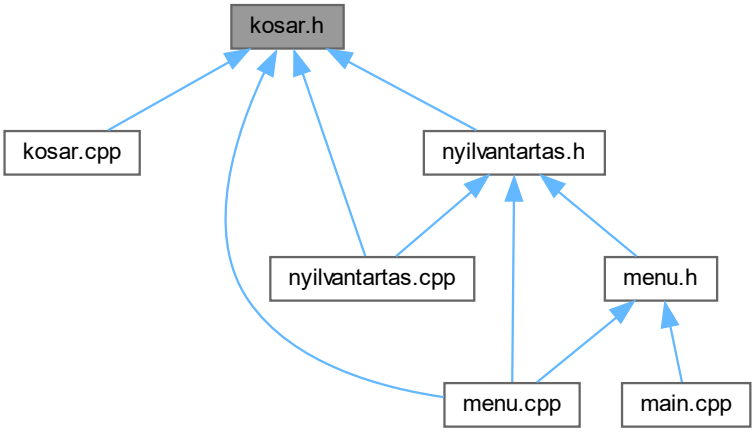



## 5.11. kosar.h fájlreferencia

```
#include "csapat.h"
#include "memtrace.h"
```
A kosar.h definíciós fájl függési gráfja:



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




### Osztályok

- class Kosar

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

## 5.12. kosar.h

Ugrás a fájl dokumentációjához.

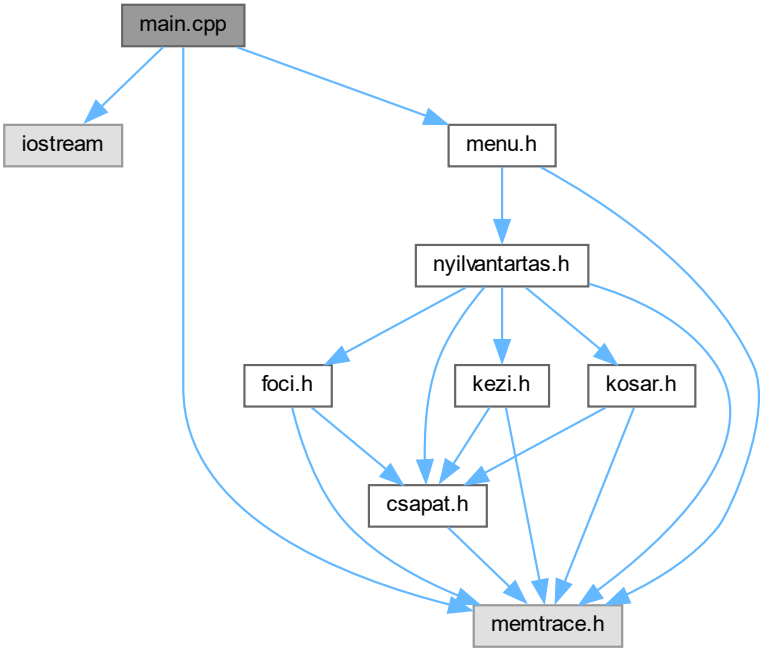
```
00001
00002
00003 #ifndef KOSAR_H
00004 #define KOSAR_H
00005
00006 #include "csapat.h"
00007 #include "memtrace.h"
00008
00010 class Kosar : public Csapat {
00011   private:
00013     int pompomDB;
00014
00015   public:
00018     Kosar();
00019
00023     Kosar(const char*, const int);
00024
00027     void addPompom(const int);
00028
00031     const int getPomPomDb() const;
00032
00033 };
00034
00035 #endif
```

## 5.13. main.cpp fájlreferencia

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

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

A main.cpp definíciós fájl függési gráfja:




## Függvények

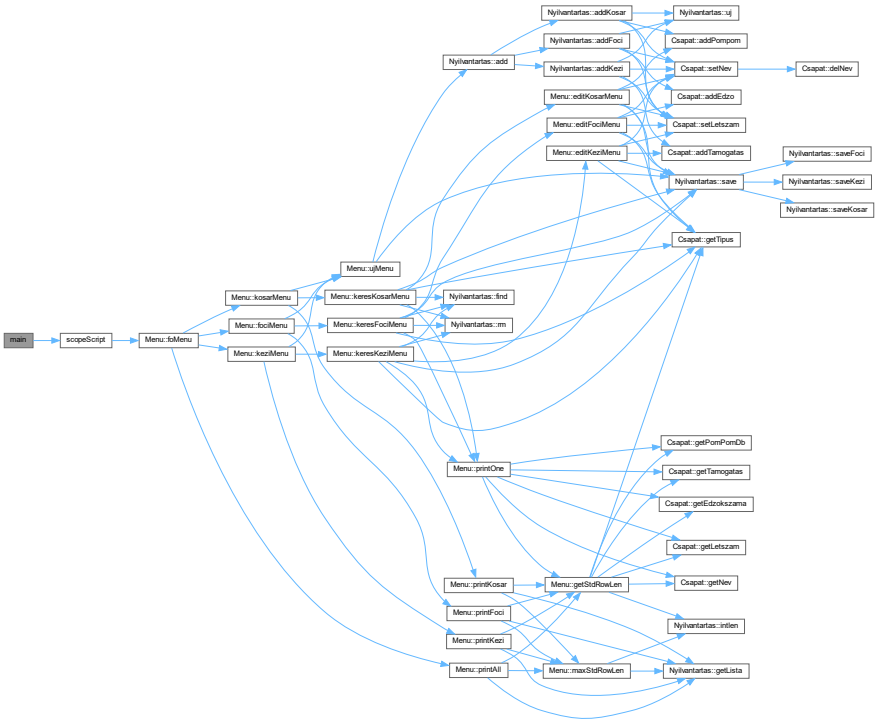
- void scopeScript ()
- int main ()

### 5.13.1. Függvények dokumentációja

#### 5.13.1.1. main()

```
int main ( )
```

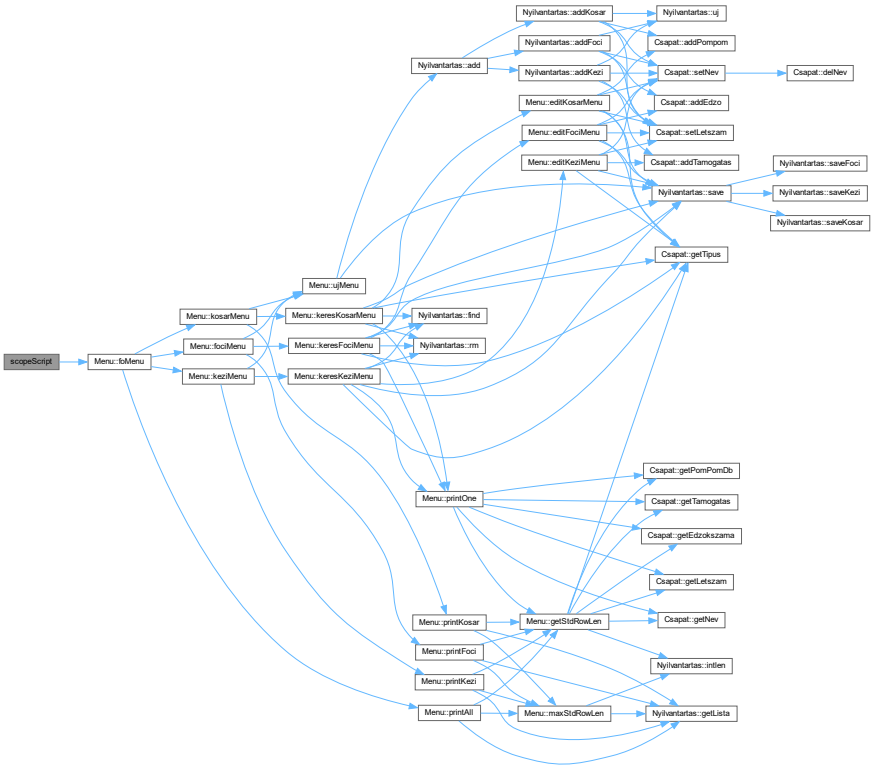
A függvény hívási gráfja:




### 5.13.1.2. scopeScript()

```
void scopeScript ( )
```

A függvény hívási gráfja:




A függvény hívó gráfja:


## 5.14. main_test.cpp fájlreferencia

### Makródefiníciók

- #define TESZT_ESET 0

### 5.14.1. Makródefiníciók dokumentációja

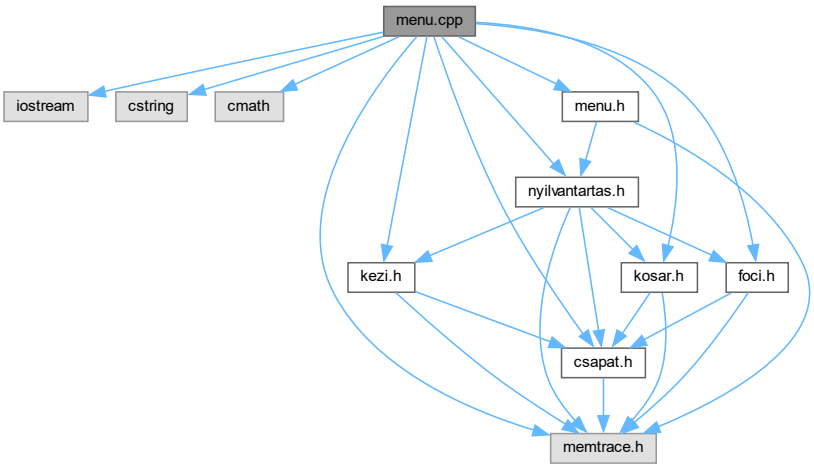
#### 5.14.1.1. TESZT_ESET

```
#define TESZT_ESET 0
```

## 5.15. menu.cpp fájlreferencia

```
#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"
```

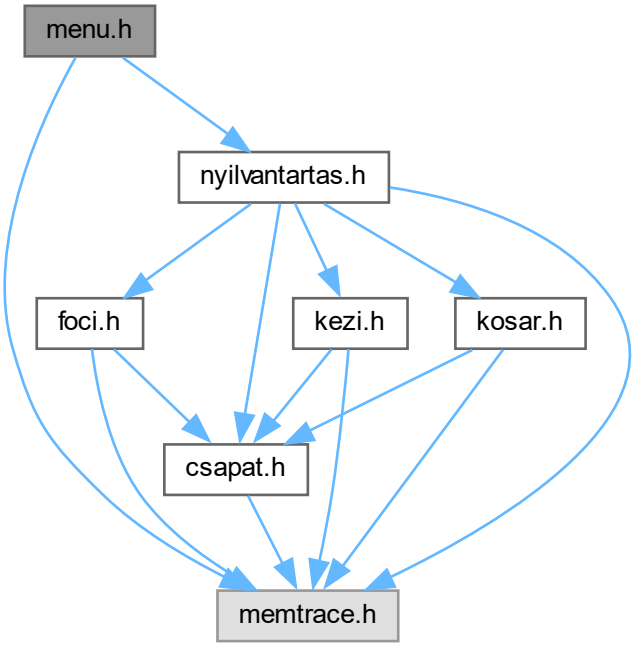
A menu.cpp definíciós fájl függési gráfja:



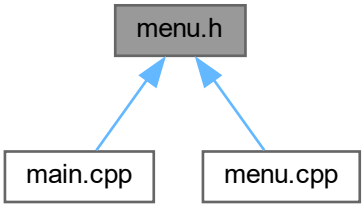

## 5.16. menu.h fájlreferencia

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

A menu.h definíciós fájl függési gráfja:




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




## Osztályok

- class Menu

  *A futó programot irányító menürendszer objektuma.*

## 5.17. menu.h

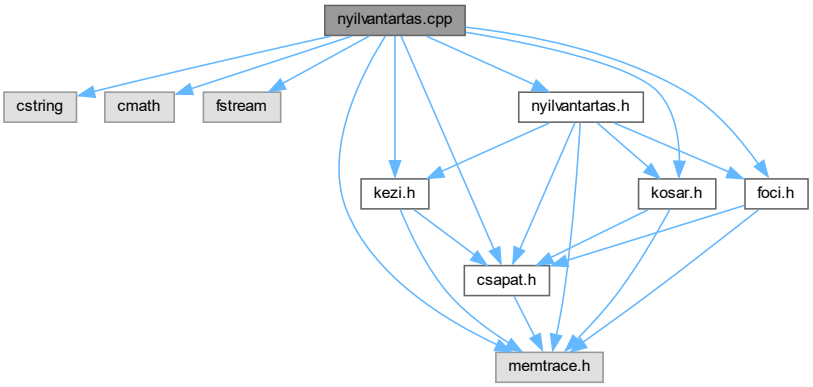
Ugrás a fájl dokumentációjához.

```
00001
00002
00003 #ifndef MENU_H
00004 #define MENU_H
00005
00006 #include "nyilvantartas.h"
00007 #include "memtrace.h"
00008
00010 class Menu {
00011   private:
00013     Nyilvantartas DB;
00014   public:
00015
00017     Menu();
00018
00020     ~Menu();
00021
00024     Nyilvantartas getNyilvantartas() const { return DB;}
00025
00027     void printAll() const;
00028
00030     void printKezi() const;
00031
00033     void printKosar() const;
00034
00036     void printFoci() const;
00037
00041     void printOne(Lista *, Tipus) const;
00042
00045     int maxStdRowLen() const;
00046
00050     int maxStdRowLen(Tipus) const;
00051
00055     int getStdRowLen(Lista *) const;
00056
00059     void foMenu();
00060
00062     void keziMenu();
00063
00065     void kosarMenu();
00066
00068     void fociMenu();
00069
00071     void keresKeziMenu();
00072
00074     void keresKosarMenu();
00075
00077     void keresFociMenu();
00078
00081     void editKeziMenu(Lista *);
00082
00085     void editKosarMenu(Lista*);
00086
00089     void editFociMenu(Lista *);
00090
00093     void ujMenu(Tipus);
00094 };
00095
00096 #endif
```

## 5.18. nyilvantartas.cpp fájlreferencia

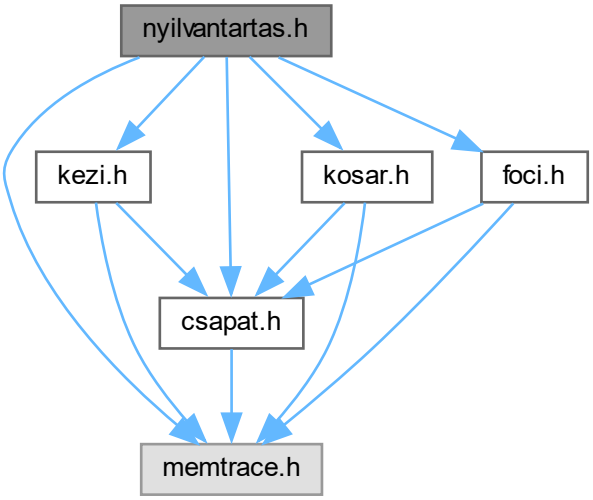
```
#include <cstring>
#include <cmath>
#include <fstream>
#include "nyilvantartas.h"
#include "csapat.h"
#include "kezi.h"
#include "kosar.h"
#include "foci.h"
```

```
#include "memtrace.h"
```
A nyilvantartas.cpp definíciós fájl függési gráfja:

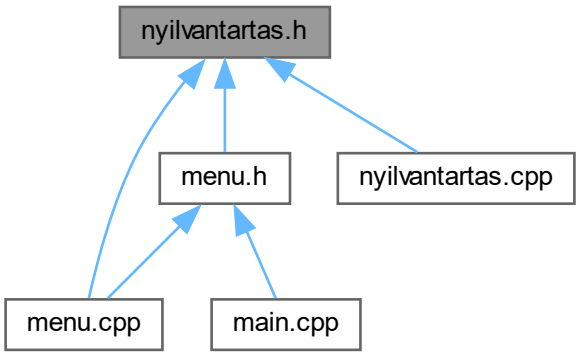



## 5.19. nyilvantartas.h fájlreferencia

```
#include "csapat.h"
#include "kezi.h"
#include "kosar.h"
#include "foci.h"
#include "memtrace.h"
```
A nyilvantartas.h definíciós fájl függési gráfja:



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




## Osztályok

- struct Lista

  *Láncolt listaelem.*
- class Nyilvantartas

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

# 5.20. nyilvantartas.h

Ugrás a fájl dokumentációjához.

```
00001
00002
00003 #ifndef NYILVANTARTAS_H
00004 #define NYILVANTARTAS_H
00005
00006 #include "csapat.h"
00007 #include "kezi.h"
00008 #include "kosar.h"
00009 #include "foci.h"
00010 #include "memtrace.h"
00011
00013 struct Lista {
00014   Csapat *adat;
00015   Lista *kovi;
00016 };
00017
00021 class Nyilvantartas {
00022   private:
00024     Lista *csapatok;
00025
00027     void delAll(){
00028       if (csapatok != nullptr) {
00029         Lista* i = csapatok;
00030         while (i != nullptr) {
00031           Lista *kov = i->kovi;
00032           delete i->adat;
00033           delete i;
00034           i = kov;
00035         }
00036       }
00037     }
00038
00039   public:
00040
```

```
00046     static int intlen(const long long int);
00047
00051     static void straddc(char *&, const char);
00052
00055     static void strdel(char *&);
00056
00059     Nyilvantartas() : csapatok(nullptr) {}
00060
00062     ~Nyilvantartas();
00063
00068     Lista *uj(Tipus);
00069
00074     void addKezi(const char*, const int, const int);
00075
00080     void addKosar(const char*, const int, const int);
00081
00086     void addFoci(const char*, const int, const int);
00087
00093     void add(const Tipus, const char*, const int, const int);
00094
00098     Lista *find(const char*) const;
00099
00102     void rm(Lista *&);
00103
00106     Lista *getLista() const;
00107
00110     bool loadKezi();
00111
00114     bool loadKosar();
00115
00118     bool loadFoci();
00119
00121     void saveKezi() const;
00122
00124     void saveKosar() const;
00125
00127     void saveFoci() const;
00128
00132     bool load();
00133
00136     void save() const;
00137 };
00138
00139 #endif
```

# Tárgymutató