

# Házi feladat

1.0

Generated by Doxygen 1.10.0

<b>1 Hierarchical Index</b>	<b>1</b>
1.1 Class Hierarchy	1
<b>2 Data Structure Index</b>	<b>3</b>
2.1 Data Structures	3
<b>3 File Index</b>	<b>5</b>
3.1 File List	5
<b>4 Data Structure Documentation</b>	<b>7</b>
4.1 Allomas Class Reference	7
4.1.1 Detailed Description	7
4.1.2 Constructor & Destructor Documentation	8
4.1.2.1 Allomas() [1/2]	8
4.1.2.2 Allomas() [2/2]	8
4.1.3 Member Function Documentation	8
4.1.3.1 getAllomas()	8
4.1.4 Field Documentation	9
4.1.4.1 allomasNev	9
4.2 Diak Class Reference	9
4.2.1 Detailed Description	11
4.2.2 Constructor & Destructor Documentation	11
4.2.2.1 Diak()	11
4.2.3 Member Function Documentation	11
4.2.3.1 getTipus()	11
4.2.3.2 nyomtat()	12
4.2.4 Field Documentation	12
4.2.4.1 tipusNev	12
4.3 Ido Class Reference	13
4.3.1 Detailed Description	13
4.3.2 Constructor & Destructor Documentation	13
4.3.2.1 Ido() [1/2]	13
4.3.2.2 Ido() [2/2]	14
4.3.3 Member Function Documentation	14
4.3.3.1 getOra()	14
4.3.3.2 getPerc()	15
4.3.4 Field Documentation	15
4.3.4.1 ora	15
4.3.4.2 perc	15
4.4 Jegy Class Reference	16
4.4.1 Detailed Description	17
4.4.2 Constructor & Destructor Documentation	17
4.4.2.1 Jegy()	17

4.4.3 Member Function Documentation	17
4.4.3.1 fizetendo()	17
4.4.3.2 getAr()	18
4.4.3.3 getSzaz()	18
4.4.3.4 nyomtat()	18
4.4.4 Field Documentation	19
4.4.4.1 ar	19
4.4.4.2 szazalek	19
4.5 Kiosk Class Reference	20
4.5.1 Detailed Description	20
4.5.2 Member Function Documentation	21
4.5.2.1 init()	21
4.5.2.2 inputCheck()	21
4.5.2.3 jegyValt()	22
4.5.2.4 listaz()	23
4.5.2.5 userInput()	24
4.5.2.6 vonatHozza()	24
4.6 Kutya Class Reference	25
4.6.1 Detailed Description	27
4.6.2 Constructor & Destructor Documentation	27
4.6.2.1 Kutya()	27
4.6.3 Member Function Documentation	27
4.6.3.1 getKov()	27
4.6.3.2 getTipus()	27
4.6.3.3 nyomtat()	28
4.6.4 Field Documentation	29
4.6.4.1 kov	29
4.6.4.2 tipusNev	29
4.7 Mav Class Reference	29
4.7.1 Detailed Description	31
4.7.2 Constructor & Destructor Documentation	31
4.7.2.1 Mav()	31
4.7.2.2 ~Mav()	31
4.7.3 Member Function Documentation	31
4.7.3.1 add()	31
4.7.3.2 addTrain()	32
4.7.3.3 beolvas()	33
4.7.3.4 getSize()	34
4.7.3.5 getVonatAt()	35
4.7.3.6 kiir()	35
4.7.3.7 kiirAt()	36
4.7.3.8 operator[]()	37

4.7.4 Field Documentation	37
4.7.4.1 si	37
4.7.4.2 vonatok	37
4.8 Seged Struct Reference	38
4.8.1 Detailed Description	38
4.8.2 Field Documentation	38
4.8.2.1 buffSize	38
4.8.2.2 erkezes	39
4.8.2.3 indulas	39
4.8.2.4 indulo	39
4.8.2.5 kocsidb	39
4.8.2.6 szam	39
4.8.2.7 veg	39
4.9 Teljes Class Reference	40
4.9.1 Detailed Description	42
4.9.2 Constructor & Destructor Documentation	42
4.9.2.1 Teljes()	42
4.9.3 Member Function Documentation	42
4.9.3.1 getTipus()	42
4.9.3.2 nyomtat()	42
4.9.4 Field Documentation	43
4.9.4.1 tipusNev	43
4.10 Vonat Class Reference	43
4.10.1 Detailed Description	45
4.10.2 Constructor & Destructor Documentation	45
4.10.2.1 Vonat() [1/2]	45
4.10.2.2 Vonat() [2/2]	46
4.10.3 Member Function Documentation	46
4.10.3.1 getErkezes()	46
4.10.3.2 getIndulas()	46
4.10.3.3 getIndulo()	46
4.10.3.4 getKocsidb()	46
4.10.3.5 getSzam()	47
4.10.3.6 getVeg()	47
4.10.3.7 kiir()	47
4.10.3.8 setErkezes()	48
4.10.3.9 setIndulas()	49
4.10.3.10 setIndulo()	49
4.10.3.11 setKocsidb()	49
4.10.3.12 setSzam()	50
4.10.3.13 setVeg()	50
4.10.4 Field Documentation	50

4.10.4.1 erkezes	50
4.10.4.2 indulas	51
4.10.4.3 indulo	51
4.10.4.4 kocsidb	51
4.10.4.5 szam	51
4.10.4.6 veg	51
<b>5 File Documentation</b>	<b>53</b>
5.1 allomas.cpp File Reference	53
5.1.1 Function Documentation	54
5.1.1.1 operator<<()	54
5.2 allomas.cpp	54
5.3 allomas.h File Reference	54
5.3.1 Function Documentation	55
5.3.1.1 operator<<()	55
5.4 allomas.h	56
5.5 ido.cpp File Reference	56
5.5.1 Function Documentation	57
5.5.1.1 operator<<()	57
5.6 ido.cpp	57
5.7 ido.h File Reference	57
5.7.1 Function Documentation	58
5.7.1.1 operator<<()	58
5.8 ido.h	59
5.9 jegy.h File Reference	60
5.10 jegy.h	61
5.11 kiosk.cpp File Reference	63
5.12 kiosk.cpp	63
5.13 kiosk.h File Reference	65
5.13.1 Enumeration Type Documentation	66
5.13.1.1 menulitem	66
5.14 kiosk.h	67
5.15 main.cpp File Reference	67
5.15.1 Function Documentation	68
5.15.1.1 main()	68
5.16 main.cpp	70
5.17 mav.cpp File Reference	70
5.18 mav.cpp	71
5.19 mav.h File Reference	72
5.20 mav.h	74
5.21 mav_test.cpp File Reference	75
5.21.1 Function Documentation	75

---

5.21.1.1 main()	75
5.21.1.2 testMavAdd()	76
5.21.1.3 testMavAddTrain()	77
5.21.1.4 testMavBeolvas()	78
5.21.1.5 testMavKiir()	79
5.21.1.6 testMavKiirAt()	80
5.22 mav_test.cpp	81
5.23 seged.h File Reference	83
5.24 seged.h	84
5.25 vonat.h File Reference	84
5.26 vonat.h	86

# Chapter 1

## Hierarchical Index

### 1.1 Class Hierarchy

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

Allomas . . . . .	7
Ido . . . . .	13
Jegy . . . . .	16
Diak . . . . .	9
Kutya . . . . .	25
Teljes . . . . .	40
Kiosk . . . . .	20
Mav . . . . .	29
Seged . . . . .	38
Vonat . . . . .	43

## Chapter 2

# Data Structure Index

### 2.1 Data Structures

Here are the data structures with brief descriptions:

Allomas	7
Diak	9
Ido	13
Jegy	16
Kiosk	20
Kutya	25
Mav	29
Seged	38
Teljes	40
Vonat	43



## Chapter 3

# File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">allomas.cpp</a>	53
<a href="#">allomas.h</a>	54
<a href="#">ido.cpp</a>	56
<a href="#">ido.h</a>	57
<a href="#">jegy.h</a>	60
<a href="#">kiosk.cpp</a>	63
<a href="#">kiosk.h</a>	65
<a href="#">main.cpp</a>	67
<a href="#">mav.cpp</a>	70
<a href="#">mav.h</a>	72
<a href="#">mav_test.cpp</a>	75
<a href="#">seged.h</a>	83
<a href="#">vonat.h</a>	84

## Chapter 4

# Data Structure Documentation

### 4.1 Allomas Class Reference

```
#include <allomas.h>
```

Collaboration diagram for Allomas:

Allomas
- allomasNev
+ Allomas()
+ Allomas()
+ getAllomas()

#### Public Member Functions

- [Allomas](#) (const char \*allomas)  
*Allomas nevét tartalmazo string.*
- [Allomas](#) (char \*allomas)
- std::string [getAllomas](#) () const

#### Private Attributes

- std::string [allomasNev](#)

#### 4.1.1 Detailed Description

STRINGGEL ÚJRAÍRVA

Definition at line 12 of file [allomas.h](#).

## 4.1.2 Constructor & Destructor Documentation

### 4.1.2.1 Allomas() [1/2]

```
Allomas::Allomas (
    const char * allomas ) [inline]
```

[Allomas](#) nevét tartalmazó string.

[Allomas](#) konstruktor

#### Parameters

<i>allomas</i>	itt egy char* egyértelműen. a másik verzióban const, hogy fel lehessen közvetlen tölteni értsd: "Álloms_neve" mint paraméter.
----------------	---

Definition at line 22 of file [allomas.h](#).

```
00022 : allomasNev(allomas) {}
```

### 4.1.2.2 Allomas() [2/2]

```
Allomas::Allomas (
    char * allomas ) [inline]
```

Definition at line 23 of file [allomas.h](#).

```
00023 : allomasNev(allomas) {}
```

## 4.1.3 Member Function Documentation

### 4.1.3.1 getAllomas()

```
std::string Allomas::getAllomas ( ) const [inline]
```

Definition at line 27 of file [allomas.h](#).

```
00027 { return allomasNev; }
```

Here is the caller graph for this function:



## 4.1.4 Field Documentation

### 4.1.4.1 allomasNev

```
std::string Allomas::allomasNev [private]
```

Definition at line 14 of file [allomas.h](#).

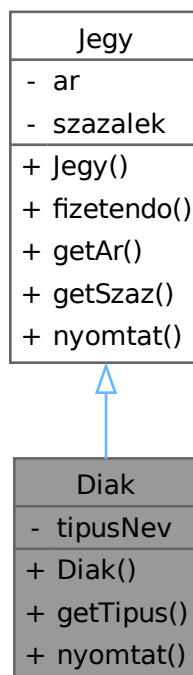
The documentation for this class was generated from the following file:

- [allomas.h](#)

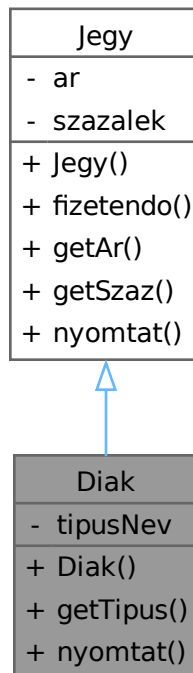
## 4.2 Diak Class Reference

```
#include <jegy.h>
```

Inheritance diagram for Diak:



Collaboration diagram for Diak:



### Public Member Functions

- **Diak** (int `ar`=1200, const double `szazalek`=0.7)
- std::string `getTipus` ()  
*Getterek.*
- void `nyomtat` (**Vonat** &`v`)

### Public Member Functions inherited from **Jegy**

- **Jegy** (int `ar`=1200, double `szazalek`=0.7)  
*Kedvezmény százaléka, pontosabban mennyi az amit fizet százalék.*
- virtual int `fizetendo` ()  
*Fizetendő összeg, ár és a kedvezmény szorzata.*
- int `getAr` ()  
*Getterek.*
- double `getSzaz` ()

### Private Attributes

- const std::string `tipusNev` = "Diák"

### 4.2.1 Detailed Description

Definition at line 48 of file [jegy.h](#).

### 4.2.2 Constructor & Destructor Documentation

#### 4.2.2.1 Diak()

```
Diak::Diak (
    int ar = 1200,
    const double szazalek = 0.7 ) [inline]
```

Konstruktor

Parameters

<i>ar</i>	megadja a jegy árát
<i>szazalek</i>	a kedvezmény értékét adja meg Meghívja a anya-class konstruktorát.

Definition at line 59 of file [jegy.h](#).

```
00059 : Jegy(ar, szazalek) {}
```

### 4.2.3 Member Function Documentation

#### 4.2.3.1 getTipus()

```
std::string Diak::getTipus ( ) [inline]
```

Getterek.

Definition at line 62 of file [jegy.h](#).

```
00062 { return tipusNev; }
```

Here is the caller graph for this function:



### 4.2.3.2 nyomtat()

```
void Diak::nyomtat (
    Vonat & v ) [inline], [virtual]
```

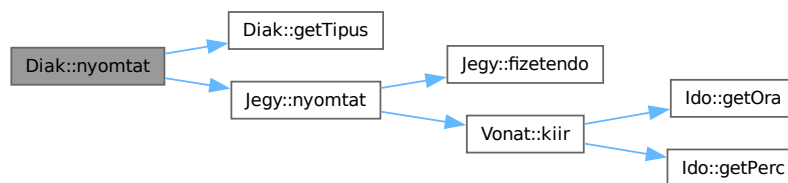
megörökölt nyomtat fv overwrite-ja meghívja az "eredei", főosztály nyomtatját

Reimplemented from [Jegy](#).

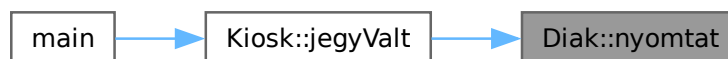
Definition at line 66 of file [jegy.h](#).

```
00066 {
00067     std::cout << "### Jegy adatai ###\n";
00068     std::cout << "Jegy típusa: " << getTipus() << std::endl;
00069     Jegy::nyomtat(v);
00070 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.2.4 Field Documentation

### 4.2.4.1 tipusNev

```
const std::string Diak::tipusNev = "Diák" [private]
```

Definition at line 49 of file [jegy.h](#).

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

- [jegy.h](#)

## 4.3 Ido Class Reference

```
#include <ido.h>
```

Collaboration diagram for Ido:

Ido
- ora
- perc
+ Ido()
+ Ido()
+ getOra()
+ getPerc()

### Public Member Functions

- [Ido](#) (const char \*ido)
- [Ido](#) (int [ora](#), int [perc](#))
- int [getOra](#) () const
- int [getPerc](#) () const

### Private Attributes

- int [ora](#)
- int [perc](#)

### 4.3.1 Detailed Description

Definition at line [12](#) of file [ido.h](#).

## 4.3.2 Constructor & Destructor Documentation

### 4.3.2.1 Ido() [1/2]

```
Ido::Ido (
    const char * ido ) [inline]
```

Konstruktor



## Parameters

<i>ido</i>	mivel beolvasáskor char* típusokba olvasok bele char*-ot kap paraméterként és azt kezeli megfelelően konvertálva.
------------	---

Definition at line 22 of file [ido.h](#).

```
00022     {
00023         ora = (ido[0] - '0') * 10 + (ido[1] - '0');
00024         perc = (ido[2] - '0') * 10 + (ido[3] - '0');
00025     }
```

## 4.3.2.2 Ido() [2/2]

```
Ido::Ido (
    int ora,
    int perc ) [inline]
```

Definition at line 27 of file [ido.h](#).

```
00027 : ora(ora), perc(perc) {}
```

## 4.3.3 Member Function Documentation

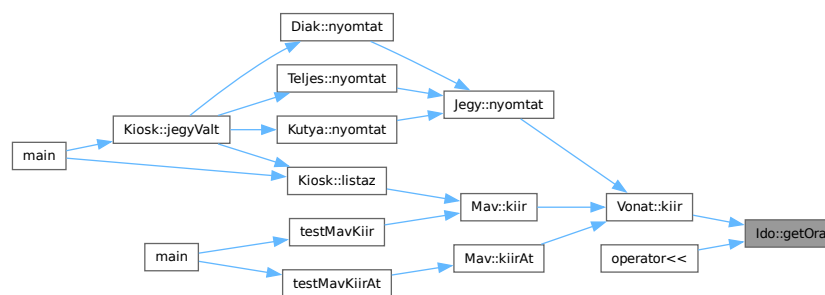
## 4.3.3.1 getOra()

```
int Ido::getOra ( ) const [inline]
```

Definition at line 36 of file [ido.h](#).

```
00036 { return ora; };
```

Here is the caller graph for this function:



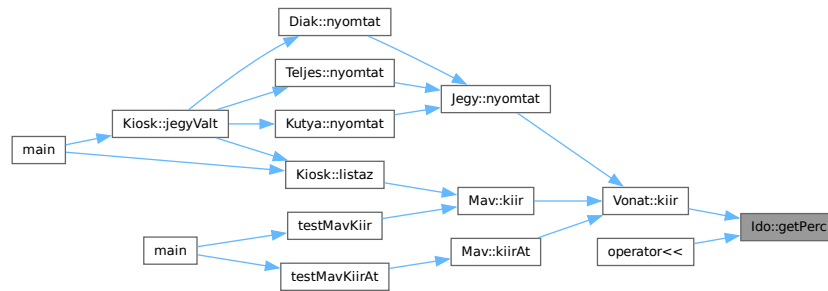
### 4.3.3.2 getPerc()

```
int IdO::getPerc ( ) const [inline]
```

Definition at line 37 of file [ido.h](#).

```
00037 {return perc; };
```

Here is the caller graph for this function:



## 4.3.4 Field Documentation

### 4.3.4.1 ora

```
int IdO::ora [private]
```

Definition at line 13 of file [ido.h](#).

### 4.3.4.2 perc

```
int IdO::perc [private]
```

Definition at line 14 of file [ido.h](#).

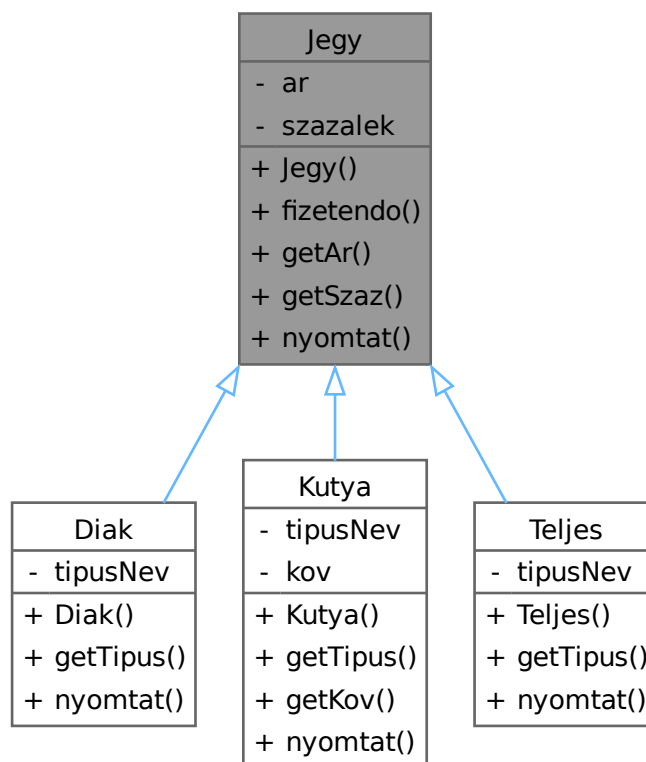
The documentation for this class was generated from the following file:

- [ido.h](#)

## 4.4 Jegy Class Reference

```
#include <jegy.h>
```

Inheritance diagram for Jegy:



Collaboration diagram for Jegy:



## Public Member Functions

- [Jegy](#) (int [ar](#)=1200, double [szazalek](#)=0.7)  
*Kedvezmény százaléka, pontosabban mennyi az amit fizet százalék.*
- virtual int [fizetendo](#) ()  
*Fizetendő összeg, ár és a kedvezmény szorzata.*
- int [getAr](#) ()  
*Getterek.*
- double [getSzaz](#) ()
- virtual void [nyomtat](#) ([Vonat](#) &[v](#))

## Private Attributes

- int [ar](#)
- double [szazalek](#)  
*A jegy ára.*

### 4.4.1 Detailed Description

FILE JEGY\_H

Definition at line 11 of file [jegy.h](#).

### 4.4.2 Constructor & Destructor Documentation

#### 4.4.2.1 Jegy()

```
Jegy::Jegy (
    int ar = 1200,
    double szazalek = 0.7 ) [inline]
```

Kedvezmény százaléka, pontosabban mennyi az amit fizet százalék.

Anya osztály Konstruktora

#### Parameters

<i>ar</i>	megadja a jegy árát
<i>szazalek</i>	a kedvezmény értékét adja meg

Definition at line 21 of file [jegy.h](#).

```
00021 : ar(ar), szazalek(szazalek) {}
```

### 4.4.3 Member Function Documentation

#### 4.4.3.1 fizetendo()

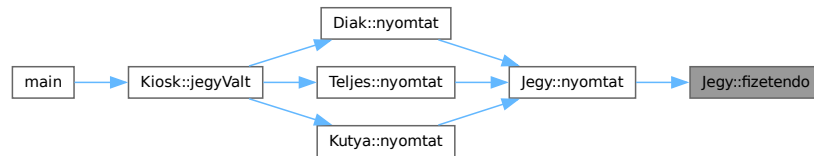
```
virtual int Jegy::fizetendo ( ) [inline], [virtual]
```

Fizetendő összeg, ár és a kedvezmény szorzata.

Definition at line 24 of file [jegy.h](#).

```
00024 { return ar * szazalek; }
```

Here is the caller graph for this function:



#### 4.4.3.2 getAr()

```
int Jegy::getAr ( ) [inline]
```

Getterek.

Definition at line 27 of file [jegy.h](#).

```
00027 { return ar; }
```

#### 4.4.3.3 getSzaz()

```
double Jegy::getSzaz ( ) [inline]
```

Definition at line 28 of file [jegy.h](#).

```
00028 { return szazalek; }
```

#### 4.4.3.4 nyomtat()

```
virtual void Jegy::nyomtat (
    Vonat & v ) [inline], [virtual]
```

##### Parameters

<b>Vonat&amp;</b>	megadott vonatra vonatkozik Kíírja a jegy adatainak egy felét. Azt a felét ami minden leszármazottál azonos. A felső pár so. virtual mivel a leszármazottak megöröklük és felülírják.
-------------------	---

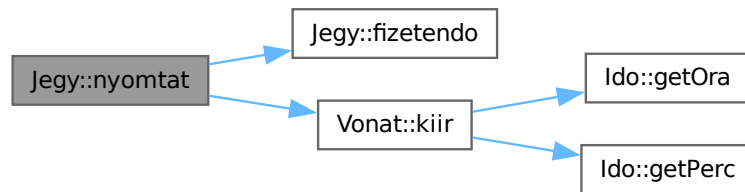
Reimplemented in [Diak](#), [Teljes](#), and [Kutya](#).

Definition at line 36 of file [jegy.h](#).

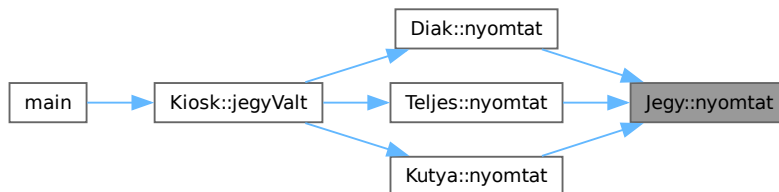
```

00036     {
00037         std::cout << "Fizetendő: " << fizetendo() << "JMF\n";
00038         std::cout << "### Vonatod adatai ###" << std::endl;
00039         v.kiir();
00040         std::cout << std::endl;
00041     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.4.4 Field Documentation

### 4.4.4.1 ar

```
int Jegy::ar [private]
```

Definition at line 12 of file [jegy.h](#).

### 4.4.4.2 szazalek

```
double Jegy::szazalek [private]
```

A jegy ára.

Definition at line 13 of file [jegy.h](#).

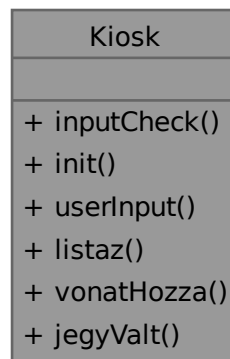
The documentation for this class was generated from the following file:

- [jegy.h](#)

## 4.5 Kiosk Class Reference

```
#include <kiosk.h>
```

Collaboration diagram for Kiosk:



### Public Member Functions

- bool [inputCheck](#) ()  
*Hibakezelés. Érvényes input esetén.*
- void [init](#) ()  
*Inicializálja a menüt, ergo kiírja a lehetőségeket.*
- [menuItem](#) [userInput](#) ()
- void [listaz](#) ([Mav](#) &mav)  
*Liszázza a vonatokat amiket beolvasott a fájlból.*
- void [vonatHozza](#) ([Mav](#) &mav, std::streampos currPos)  
*Hozzá ad a user vonatokat a fájlhoz.*
- void [jegyValt](#) ([Mav](#) &mav)  
*jegyet vált a kiválasztott vonatra*

### 4.5.1 Detailed Description

Definition at line 23 of file [kiosk.h](#).

## 4.5.2 Member Function Documentation

### 4.5.2.1 init()

```
void Kiosk::init ( )
```

Inicializálja a menüt, ergo kiírja a lehetőségeket.

Definition at line 26 of file [kiosk.cpp](#).

```
00026     {
00027     std::cout << "### Dönts ###" << std::endl;
00028     std::cout << "1. Vonat hozzáadása " << std::endl;
00029     std::cout << "2. Vonatok listázása " << std::endl;
00030     std::cout << "3. Jegy nyomtatása" << std::endl;
00031     std::cout << "4. Kilépés" << std::endl;
00032 }
```

Here is the caller graph for this function:



### 4.5.2.2 inputCheck()

```
bool Kiosk::inputCheck ( )
```

Hibakezelés. Érvényes input esetén.

Érvényes-e az input mikor számot kellene megadnom.

Returns

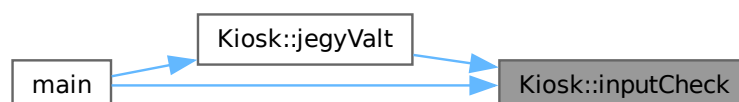
true, különben

false

Definition at line 15 of file [kiosk.cpp](#).

```
00015     {
00016     // Kb innen lopva
00017     https://stackoverflow.com/questions/12721911/c-how-to-verify-if-the-data-input-is-of-the-correct-datatype
00017     if (std::cin.fail()) {
00018         std::cout << "Nem sikerült érvényes inputot megadni... PRÓBÁLD ÚJRA" << std::endl;
00019         std::cin.clear();
00020         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00021         return false;
00022     }
00023     else return true;
00024 }
```

Here is the caller graph for this function:





## 4.5.2.3 jegyVált()

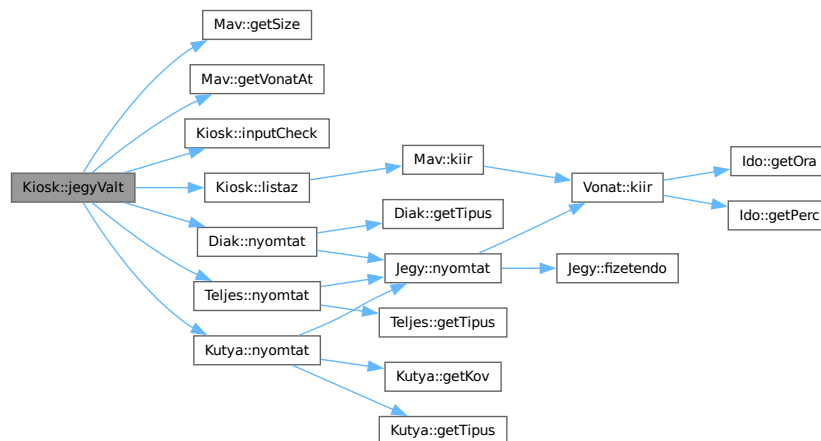
```
void Kiosk::jegyVált (
    Mav & mav )
```

jegyet vált a kiválasztott vonatra

Definition at line 65 of file kiosk.cpp.

```
00065 {
00066     int idx, buf;
00067     listaz(mav);
00068     std::cout << "Valasz vonatot: ";
00069     std::cin >> idx;
00070     // Hibakezelés
00071     if(size_t(idx) > mav.getSize()){ std::cout << "Túlinindexelés\n"; return; }
00072
00073     if(!(Kiosk::inputCheck())) return; // Érvényes inputot szűrök.
00074
00075     std::cout << std::endl; // szép kiírás miatt.
00076     // end if Hibakezelés
00077
00078
00079     std::cout << "1.Teljes, 2.Diak, 3.Kutya\n ";
00080     std::cin >> buf;
00081     // Hibakezelés
00082     if(buf > 3){ std::cout << "Túlinindexelés"; return; }
00083
00084     if(!(Kiosk::inputCheck())) return;
00085
00086     std::cout << std::endl;
00087     // end of Hibakezelés
00088
00089     switch (buf) {
00090     case 1: {
00091         Teljes t;
00092         t.nyomtat(mav.getVonatAt(idx));
00093         break;
00094     }
00095
00096     case 2: {
00097         Diak d;
00098         d.nyomtat(mav.getVonatAt(idx));
00099         break;
00100     }
00101
00102     case 3: {
00103         Kutya k;
00104         k.nyomtat(mav.getVonatAt(idx));
00105         break;
00106     }
00107
00108     } // end of switch
00109
00110 } // end of jegyVált
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.5.2.4 listaz()

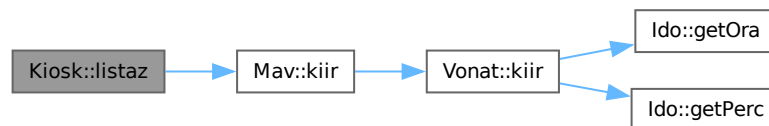
```
void Kiosk::listaz (  
    Mav & mav )
```

Liszázza a vonatokat amiket beolvasott a fájlból.

Definition at line 43 of file [kiosk.cpp](#).

```
00043 {  
00044     mav.kiir();  
00045 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.5.2.5 userInput()

```
menuItem Kiosk::userInput ( )
```

Felhasználótól bemenetet kér.

##### Returns

menultem fent említett enum class elemet ad vissza

Definition at line 35 of file [kiosk.cpp](#).

```
00035     {
00036         int valasz;
00037         std::cout << "Választott lehetőség: ";
00038         std::cin >> valasz;
00039         return static_cast<menuItem>(valasz);
00040     }
```

Here is the caller graph for this function:



#### 4.5.2.6 vonatHozza()

```
void Kiosk::vonatHozza (
    Mav & mav,
    std::streampos currPos )
```

Hozzá ad a user vonatokat a fájlhoz.

Definition at line 48 of file [kiosk.cpp](#).

```
00048     {
00049         Seged s;
00050
00051         std::cin >> s.szam;
00052         // if(*(s.szam) == '\n') std::cout << "FASZ";
00053         std::cin >> s.indulo;
00054         std::cin >> s.veg;
00055         std::cin >> s.kocsidb;
00056         std::cin >> s.indulas;
00057         std::cin >> s.erkezes;
00058
00059         mav.addTrain(currPos, atoi(s.szam), Allomas(s.indulo), Allomas(s.veg),
00060                     atoi(s.kocsidb), Ido(s.indulas), Ido(s.erkezes));
00061     } // End of vonatHozza
```

Here is the call graph for this function:



Here is the caller graph for this function:



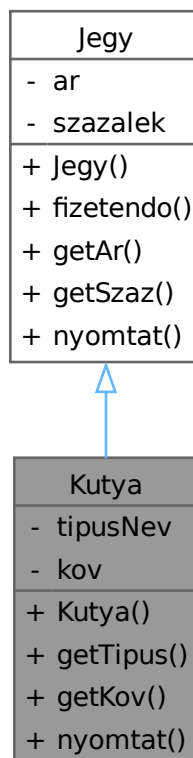
The documentation for this class was generated from the following files:

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

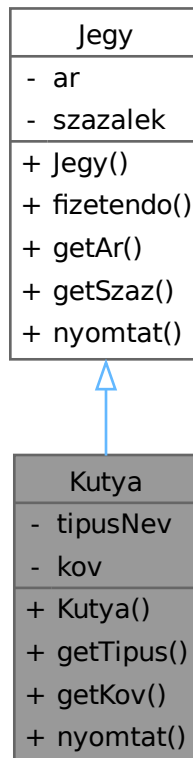
## 4.6 Kutya Class Reference

```
#include <jegy.h>
```

Inheritance diagram for Kutya:



Collaboration diagram for Kutya:



### Public Member Functions

- **Kutya** (int `ar`=1200, const double `szazalek`=0.2)
- std::string `getTipus` ()
- std::string `getKov` ()
- void `nyomtat` (`Vonat` &`v`)

### Public Member Functions inherited from **Jegy**

- **Jegy** (int `ar`=1200, double `szazalek`=0.7)  
*Kedvezmény százaléka, pontosabban mennyi az amit fizet százalék.*
- virtual int `fizetendo` ()  
*Fizetendő összeg, ár és a kedvezmény szorzata.*
- int `getAr` ()  
*Getterek.*
- double `getSzaz` ()

### Private Attributes

- const std::string `tipusNev` = "Kutya"
- const std::string `kov` = "Szájkosár"

### 4.6.1 Detailed Description

Definition at line 101 of file [jegy.h](#).

### 4.6.2 Constructor & Destructor Documentation

#### 4.6.2.1 Kutya()

```
Kutya::Kutya (
    int ar = 1200,
    const double szazalek = 0.2 ) [inline]
```

Konstruktor

Parameters

<i>ar</i>	megadja a jegy árát
<i>szazalek</i>	a kedvezmény értékét adja meg Meghívja a anya-class konstruktorát.

Definition at line 112 of file [jegy.h](#).

```
00112 : Jegy(ar, szazalek) {}
```

### 4.6.3 Member Function Documentation

#### 4.6.3.1 getKov()

```
std::string Kutya::getKov ( ) [inline]
```

Definition at line 116 of file [jegy.h](#).

```
00116 { return kov; }
```

Here is the caller graph for this function:



#### 4.6.3.2 getTipus()

```
std::string Kutya::getTipus ( ) [inline]
```

Definition at line 115 of file [jegy.h](#).

```
00115 { return tipusNev; }
```

Here is the caller graph for this function:



#### 4.6.3.3 nyomtat()

```

void Kutya::nyomtat (
    Vonat & v ) [inline], [virtual]
  
```

megörökölt nyomtat fv overwrite-ja meghívja az "eredei", főosztály nyomtatját

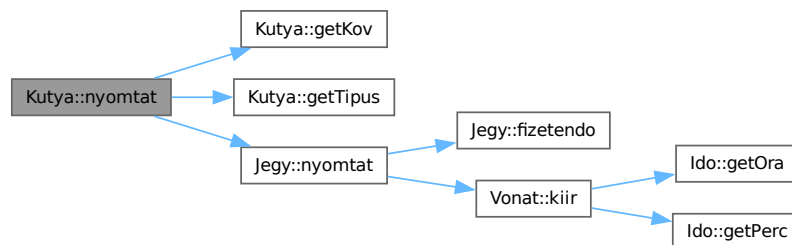
Reimplemented from [Jegy](#).

Definition at line 120 of file [jegy.h](#).

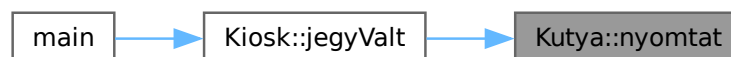
```

00120     {
00121         std::cout << "### Jegy adatai ###\n";
00122         std::cout << "Jegy típusa: " << getTipus() << std::endl;
00123         std::cout << "Követelmény: " << getKov() << std::endl;
00124         Jegy::nyomtat(v);
00125     }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.6.4 Field Documentation

### 4.6.4.1 kov

```
const std::string Kutya::kov = "Szájkosár" [private]
```

Definition at line 103 of file [jegy.h](#).

### 4.6.4.2 tipusNev

```
const std::string Kutya::tipusNev = "Kutya" [private]
```

Definition at line 102 of file [jegy.h](#).

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

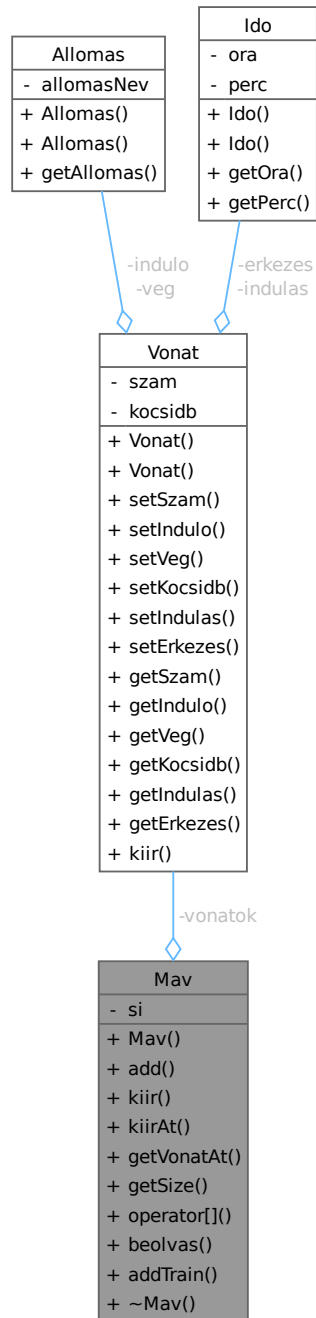
- [jegy.h](#)

## 4.7 Mav Class Reference

```
#include <mav.h>
```



Collaboration diagram for Mav:



### Public Member Functions

- **Mav** (`size_t si=0`)
- void **add** (**Vonat** &vonat)
- void **kiir** ()
- void **kiirAt** (int i)
- **Vonat** & **getVonatAt** (int i)

- `size_t` [getSize\(\)](#)
- `Vonat` & `operator[]` (int idx)
- void `beolvas()`
- void `addTrain` (std::streampos &currPos, int szam, `Allomas` indulo, `Allomas` veg, int kocsidb, `Ido` indulas, `Ido` erkezes)
- `~Mav()`

### Private Attributes

- `Vonat` \* `vonatok`
- `size_t` `si`

## 4.7.1 Detailed Description

Definition at line 16 of file [mav.h](#).

## 4.7.2 Constructor & Destructor Documentation

### 4.7.2.1 Mav()

```
Mav::Mav (
    size_t si = 0 ) [inline]
```

Definition at line 22 of file [mav.h](#).  
00022 : `vonatok`(`nullptr`), `si`(`si`) {}

### 4.7.2.2 ~Mav()

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

Definition at line 70 of file [mav.h](#).  
00070 {  
00071 `delete[]` `vonatok`;  
00072 }

## 4.7.3 Member Function Documentation

### 4.7.3.1 add()

```
void Mav::add (
    Vonat & vonat ) [inline]
```

Elem hozzáfűzése Vonatok kollekcióhoz

#### Parameters

<code>vonat</code>	vonat referencia amit hozzáad
--------------------	-------------------------------

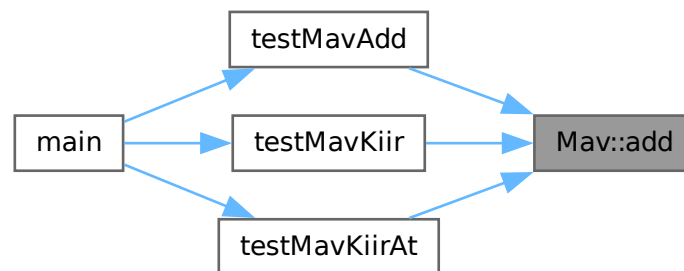
Definition at line 28 of file [mav.h](#).

```

00028     {
00029         Vonat* temp = new Vonat[si + 1]; // Lefoglalok helyet
00030
00031         for (size_t i = 0; i < si; ++i) { // Átmásolom a tömb elemeit
00032             temp[i] = vonatok[i];
00033         }
00034         temp[si++] = vonat; // Beleteszem a vonatot
00035
00036         delete[] vonatok; // "Régi" tömböt törlöm
00037         vonatok = temp; //
00038     } // End of feltolt

```

Here is the caller graph for this function:



#### 4.7.3.2 addTrain()

```

void Mav::addTrain (
    std::streampos & currPos,
    int szam,
    Allomas indulo,
    Allomas veg,
    int kocsidb,
    Ido indulas,
    Ido erkezes )

```

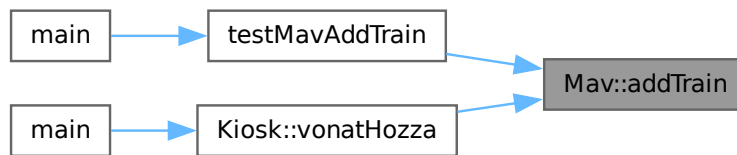
Definition at line 45 of file [mav.cpp](#).

```

00046     {
00047         // Inicializálás
00048         const char* vonatok = "./vonatok.txt";
00049         std::ofstream file (vonatok, std::ios::app); // Hozzáfüzésesen nyitom meg.
00050
00051         // HIBAKEZELES IDE
00052         if (!file.is_open()) {
00053             std::cout << "Megyinitással van baj " << vonatok << std::endl;
00054             return;
00055         }
00056         // end of HIBAKEZELES
00057
00058         if (file.is_open()) {
00059             file.seekp(currPos); // Megfelelő helyre ugrok
00060
00061             file << szam << " " << indulo << " " << veg << " " << kocsidb << " "
00062                 << indulas << " " << erkezes << std::endl;
00063         }
00064
00065         file.close();
00066     } // END OF addTrain

```

Here is the caller graph for this function:



#### 4.7.3.3 beolvas()

```
void Mav::beolvas ( )
```

Beolvaásás a file-ből egy segéd strukturába, onnan pedig egy [Vonat](#) objektum feltöltés

##### Parameters

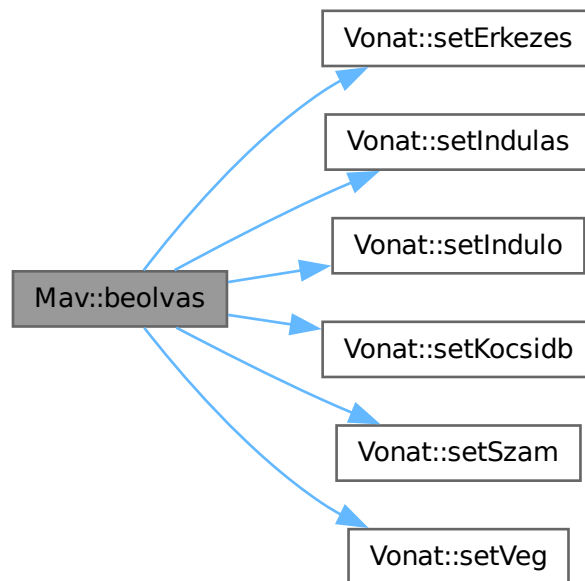
<b>Mav&amp;</b>	mav class feltöltéséhez szükséges
-----------------	-----------------------------------

Definition at line 9 of file [mav.cpp](#).

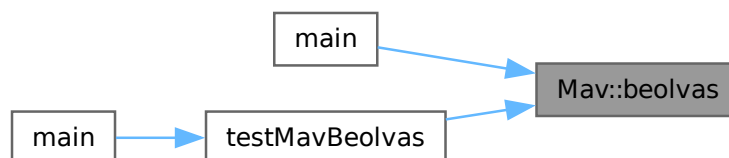
```

00009      {
00010          // Inicializálás
00011          const char* vonatok_file = "./vonatok.txt";
00012          std::ifstream file(vonatok_file);
00013          Vonat v;
00014          Seged seged;
00015
00016          if (!file.is_open()) {
00017              std::cout << "Megnyitással van baj " << vonatok_file << std::endl;
00018              return;
00019          }
00020
00021          // Kiírom a vonatok kollekciót, hogy ne legyen ráolvasás
00022          delete[] vonatok;
00023          vonatok = nullptr;
00024          si = 0;
00025
00026          while (file >> seged.szam >> seged.indulo >> seged.veg >> seged.kocsidb >> seged.indulas >> seged.erkezes)
00027          {
00028              v.setSzam(std::atoi(seged.szam));
00029              v.setIndulo(seged.indulo);
00030              v.setVeg(seged.veg);
00031              v.setKocsidb(std::atoi(seged.kocsidb));
00032              v.setIndulas(seged.indulas);
00033              v.setErkezes(seged.erkezes);
00034
00035              // feltöltöm a MAV "mgmt" class Vonat* tömbjét
00036              this->add(v);
00037          }
00038          file.close();
00039      } // END OF BEOLVAS
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



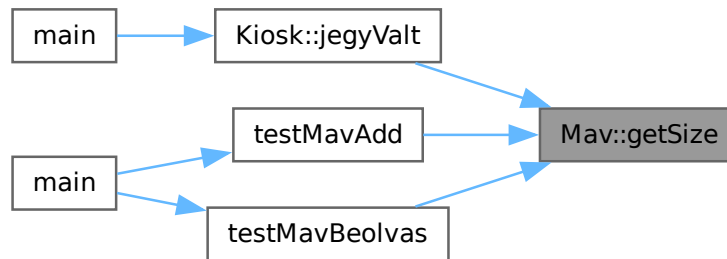
#### 4.7.3.4 getSize()

```
size_t Mav::getSize ( ) [inline]
```

Definition at line 57 of file [mav.h](#).

```
00057 { return static_cast<int>(si); }
```

Here is the caller graph for this function:



#### 4.7.3.5 getVonatAt()

```

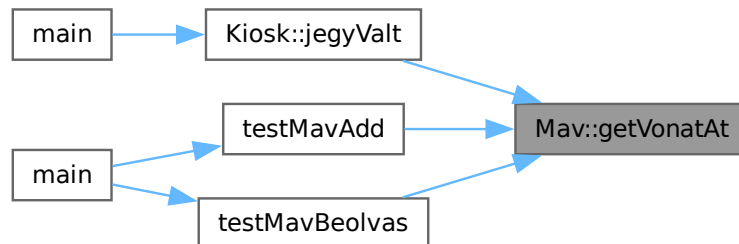
Vonat & Mav::getVonatAt (
    int i ) [inline]
  
```

Definition at line 56 of file [mav.h](#).

```

00056 { return vonatok[i-1]; }
  
```

Here is the caller graph for this function:



#### 4.7.3.6 kiir()

```

void Mav::kiir ( ) [inline]
  
```

Definition at line 41 of file [mav.h](#).

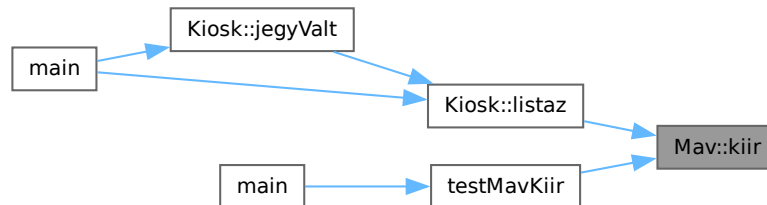
```

00041     {
00042         for (size_t i = 0; i < si; ++i) {
00043             std::cout << "--- " << i+1 << ".vonat ---" << std::endl;
00044             vonatok[i].kiir();
00045             std::cout << std::endl;
00046         }
00047     } // end of kiir
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.7.3.7 kiirAt()

```
void Mav::kiirAt (
    int i ) [inline]
```

Definition at line 50 of file [mav.h](#).

```
00050 {
00051     // HIBAKEZEZÉS
00052     std::cout << i+1 << ".vonat" << std::endl;
00053     vonatok[i].kiir();
00054 } // end of kiirAt
```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 4.7.3.8 operator[]()

```
Vonat & Mav::operator[] (
    int idx ) [inline]
```

Definition at line 59 of file [mav.h](#).

```
00059 { return vonatok[idx]; }
```

### 4.7.4 Field Documentation

#### 4.7.4.1 si

```
size_t Mav::si [private]
```

Definition at line 18 of file [mav.h](#).

#### 4.7.4.2 vonatok

```
Vonat* Mav::vonatok [private]
```

Definition at line 17 of file [mav.h](#).

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

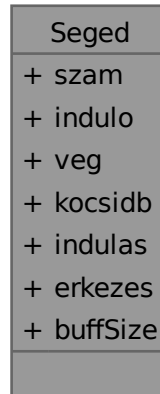
- [mav.h](#)
- [mav.cpp](#)



## 4.8 Seged Struct Reference

```
#include <seged.h>
```

Collaboration diagram for Seged:



### Data Fields

- char [szam](#) [[buffSize](#)]
- char [indulo](#) [[buffSize](#)]
- char [veg](#) [[buffSize](#)]
- char [kocsidb](#) [[buffSize](#)]
- char [indulas](#) [[buffSize](#)]
- char [erkezes](#) [[buffSize](#)]

### Static Public Attributes

- static const int [buffSize](#) = 25

### 4.8.1 Detailed Description

Magyarország leghosszabb településneve 15 karakter 25 karakterbe bele kell férnia

Definition at line [20](#) of file [seged.h](#).

### 4.8.2 Field Documentation

#### 4.8.2.1 buffSize

```
const int Seged::buffSize = 25 [static]
```

Definition at line [21](#) of file [seged.h](#).

#### 4.8.2.2 erkezes

```
char Seged::erkezes[buffSize]
```

Definition at line 27 of file [seged.h](#).

#### 4.8.2.3 indulas

```
char Seged::indulas[buffSize]
```

Definition at line 26 of file [seged.h](#).

#### 4.8.2.4 indulo

```
char Seged::indulo[buffSize]
```

Definition at line 23 of file [seged.h](#).

#### 4.8.2.5 kocsidb

```
char Seged::kocsidb[buffSize]
```

Definition at line 25 of file [seged.h](#).

#### 4.8.2.6 szam

```
char Seged::szam[buffSize]
```

Definition at line 22 of file [seged.h](#).

#### 4.8.2.7 veg

```
char Seged::veg[buffSize]
```

Definition at line 24 of file [seged.h](#).

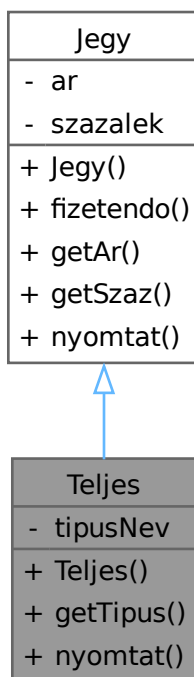
The documentation for this struct was generated from the following file:

- [seged.h](#)

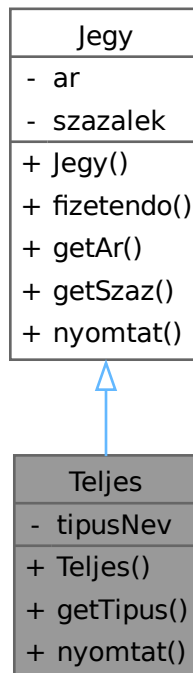
## 4.9 Teljes Class Reference

```
#include <jegy.h>
```

Inheritance diagram for Teljes:



Collaboration diagram for Teljes:



### Public Member Functions

- `Teljes` (int `ar`=1200, const double `szazalek`=1.0)
- `std::string getTipus ()`
- `void nyomtat (Vonat &v)`

### Public Member Functions inherited from `Jegy`

- `Jegy` (int `ar`=1200, double `szazalek`=0.7)  
*Kedvezmény százaléka, pontosabban mennyi az amit fizet százalék.*
- `virtual int fizetendo ()`  
*Fizetendő összeg, ár és a kedvezmény szorzata.*
- `int getAr ()`  
*Getterek.*
- `double getSzaz ()`

### Private Attributes

- `const std::string tipusNev = "Teljes"`

## 4.9.1 Detailed Description

Definition at line 75 of file [jegy.h](#).

## 4.9.2 Constructor & Destructor Documentation

### 4.9.2.1 Teljes()

```
Teljes::Teljes (
    int ar = 1200,
    const double szazalek = 1.0 ) [inline]
```

Konstruktor

Parameters

<i>ar</i>	megadja a jegy árát
<i>szazalek</i>	a kedvezmény értékét adja meg Meghívja a anya-class konstruktorát.

Definition at line 86 of file [jegy.h](#).

```
00086 : Jegy(ar, szazalek) {}
```

## 4.9.3 Member Function Documentation

### 4.9.3.1 getTipus()

```
std::string Teljes::getTipus ( ) [inline]
```

Definition at line 88 of file [jegy.h](#).

```
00088 { return tipusNev; }
```

Here is the caller graph for this function:



### 4.9.3.2 nyomtat()

```
void Teljes::nyomtat (
    Vonat & v ) [inline], [virtual]
```

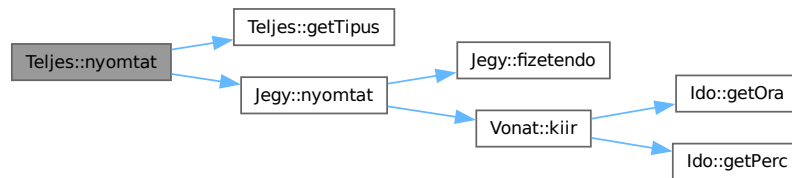
megörökölt nyomtat fv overwrite-ja meghívja az "eredei", főosztály nyomtatját

Reimplemented from [Jegy](#).

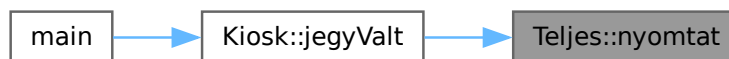
Definition at line 92 of file [jegy.h](#).

```
00092     {
00093         std::cout << "### Jegy adatai ###\n";
00094         std::cout << "Jegy típusa: " << getTipus() << std::endl;
00095         Jegy::nyomtat(v);
00096     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



## 4.9.4 Field Documentation

### 4.9.4.1 tipusNev

```
const std::string Teljes::tipusNev = "Teljes" [private]
```

Definition at line 76 of file [jegy.h](#).

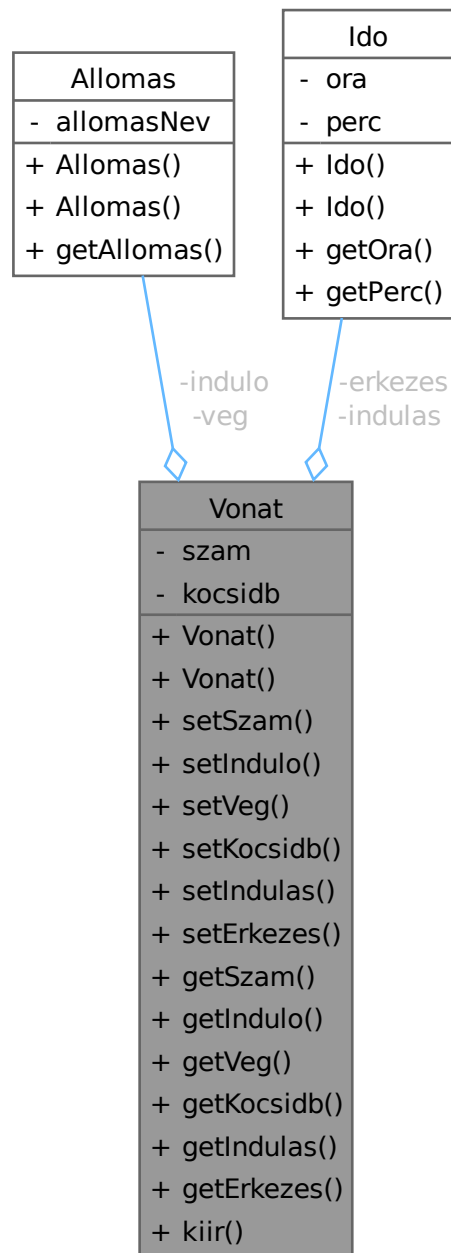
The documentation for this class was generated from the following file:

- [jegy.h](#)

## 4.10 Vonat Class Reference

```
#include <vonat.h>
```

Collaboration diagram for Vonat:



## Public Member Functions

- [Vonat](#) ()  
*Érkezési idp.*
- [Vonat](#) (int vszam, [Allomas](#) indulop, [Allomas](#) vegp, int [kocsidb](#), [Ido](#) indulasp, [Ido](#) erkezesp)  
*Paraméteres Konstruktor.*
- void [setSzam](#) (int [szam](#))

*Setterek.*

- void [setIndulo](#) (const char \*[indulo](#))
- void [setVeg](#) (const char \*[veg](#))
- void [setKocsidb](#) (int [kocsidb](#))
- void [setIndulas](#) (const char \*[ido](#))
- void [setErkezes](#) (const char \*[ido](#))
- int [getSzam](#) () const

*Getterek.*

- [Allomas](#) [getIndulo](#) () const
- [Allomas](#) [getVeg](#) () const
- int [getKocsidb](#) () const
- [Ido](#) [getIndulas](#) () const
- [Ido](#) [getErkezes](#) () const
- void [kiir](#) () const

## Private Attributes

- int [szam](#)
- [Allomas](#) [indulo](#)

*Vonatszám.*

- [Allomas](#) [veg](#)

*Kiinduló állomás.*

- int [kocsidb](#)

*Végállomás.*

- [Ido](#) [indulas](#)

*Kocsik darabszáma, basically semmit sem csinál.*

- [Ido](#) [erkezes](#)

*Indulási idő*

## 4.10.1 Detailed Description

Definition at line 15 of file [vonat.h](#).

## 4.10.2 Constructor & Destructor Documentation

### 4.10.2.1 Vonat() [1/2]

```
Vonat::Vonat ( ) [inline]
```

Érkezési idp.

Paraméter nélküli Konstruktork

Definition at line 26 of file [vonat.h](#).

```
00026 : szam(0), indulo(""), veg(""), kocsidb(0), indulas(0,0), erkezes(0,0) {}
```



### 4.10.2.2 Vonat() [2/2]

```
Vonat::Vonat (
    int vszam,
    Allomas indulop,
    Allomas vegp,
    int kocsidb,
    Ido indulasp,
    Ido erkezesp ) [inline]
```

Paraméteres Konstruktor.

Definition at line 29 of file [vonat.h](#).

```
00030      : szam(vszam), indulo(indulop), veg(vegp), kocsidb(kocsidb), indulas(indulasp),
      erkezes(erkezesp) {}
```

## 4.10.3 Member Function Documentation

### 4.10.3.1 getErkezes()

```
Ido Vonat::getErkezes ( ) const [inline]
```

Definition at line 46 of file [vonat.h](#).

```
00046 { return erkezes; }
```

### 4.10.3.2 getIndulas()

```
Ido Vonat::getIndulas ( ) const [inline]
```

Definition at line 45 of file [vonat.h](#).

```
00045 { return indulas; }
```

### 4.10.3.3 getIndulo()

```
Allomas Vonat::getIndulo ( ) const [inline]
```

Definition at line 42 of file [vonat.h](#).

```
00042 { return indulo; }
```

### 4.10.3.4 getKocsidb()

```
int Vonat::getKocsidb ( ) const [inline]
```

Definition at line 44 of file [vonat.h](#).

```
00044 { return kocsidb; }
```

#### 4.10.3.5 getSzam()

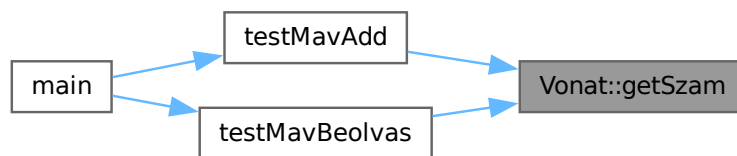
```
int Vonat::getSzam ( ) const [inline]
```

Getterek.

Definition at line 41 of file [vonat.h](#).

```
00041 { return szam; }
```

Here is the caller graph for this function:



#### 4.10.3.6 getVeg()

```
Allomas Vonat::getVeg ( ) const [inline]
```

Definition at line 43 of file [vonat.h](#).

```
00043 { return veg; }
```

#### 4.10.3.7 kiir()

```
void Vonat::kiir ( ) const [inline]
```

Kiírja a vonat adatait Főképp a jegyváltáskor van meghívva + listázás.

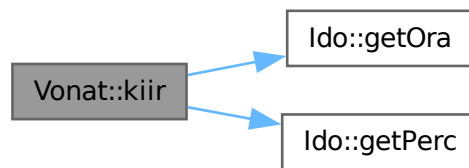
Definition at line 52 of file [vonat.h](#).

```

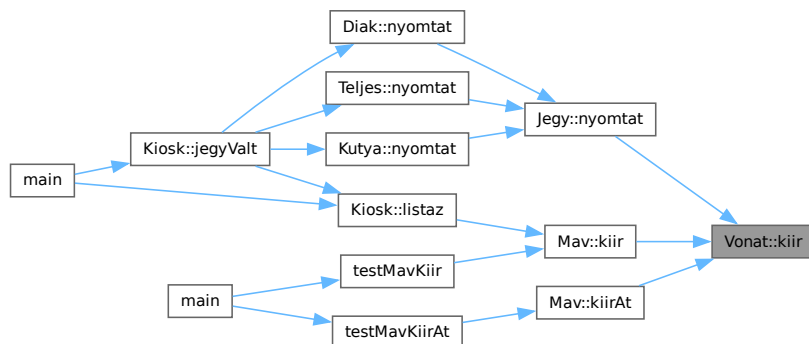
00052     {
00053         std::cout << "Vonat szama: " << szam << std::endl;
00054         std::cout << "Indulasi allomas: " << indulo << std::endl;
00055         std::cout << "Erkezesi allomas: " << veg << std::endl;
00056         std::cout << "Kocsi darabszam: " << kocsidb << std::endl;
00057
00058         std::cout << "Indulas idopontja: ";
00059         std::cout << std::setw(2) << std::setfill('0') << indulas.getOra();
00060         std::cout << ":";
00061         std::cout << std::setw(2) << std::setfill('0') << indulas.getPerc() << std::endl; // Nagyon ronda,
00062         tudom...
00063         std::cout << "Erkezés idopontja: ";
00064         std::cout << std::setw(2) << std::setfill('0') << erkezes.getOra();
00065         std::cout << ":";
00066         std::cout << std::setw(2) << std::setfill('0') << erkezes.getPerc() << std::endl; // Nagyon ronda,
00067         tudom...
00068     }

```

Here is the call graph for this function:



Here is the caller graph for this function:



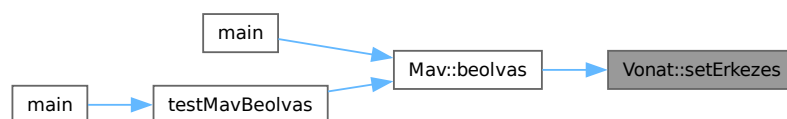
#### 4.10.3.8 setErkezes()

```
void Vonat::setErkezes (
    const char * ido ) [inline]
```

Definition at line 38 of file [vonat.h](#).

```
00038 { this->erkezes = Ido(ido); }
```

Here is the caller graph for this function:



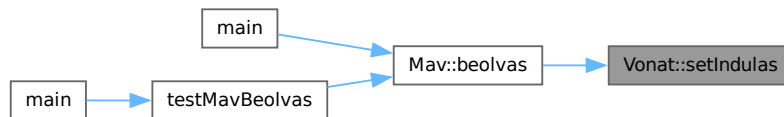
#### 4.10.3.9 setIndulas()

```
void Vonat::setIndulas (
    const char * ido ) [inline]
```

Definition at line 37 of file [vonat.h](#).

```
00037 { this->indulas = Idó(ido); }
```

Here is the caller graph for this function:



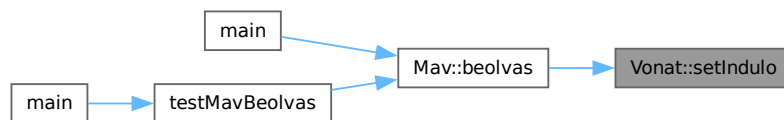
#### 4.10.3.10 setIndulo()

```
void Vonat::setIndulo (
    const char * indulo ) [inline]
```

Definition at line 34 of file [vonat.h](#).

```
00034 { this-> indulo = Allomas(indulo); }
```

Here is the caller graph for this function:



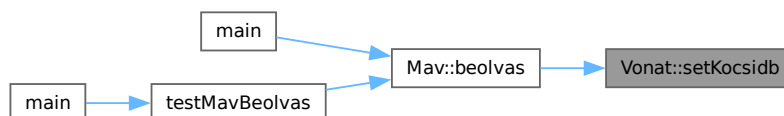
#### 4.10.3.11 setKocsidb()

```
void Vonat::setKocsidb (
    int kocsidb ) [inline]
```

Definition at line 36 of file [vonat.h](#).

```
00036 { this->kocsidb = kocsidb; }
```

Here is the caller graph for this function:



#### 4.10.3.12 setSzam()

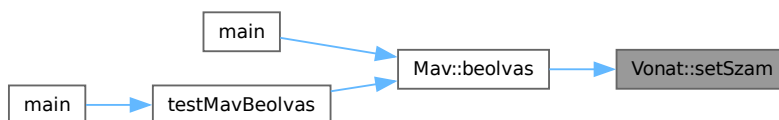
```
void Vonat::setSzam (
    int szam ) [inline]
```

Setterek.

Definition at line 33 of file [vonat.h](#).

```
00033 { this->szam = szam; }
```

Here is the caller graph for this function:



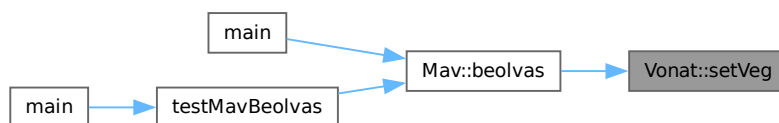
#### 4.10.3.13 setVeg()

```
void Vonat::setVeg (
    const char * veg ) [inline]
```

Definition at line 35 of file [vonat.h](#).

```
00035 { this->veg = Allomas(veg); }
```

Here is the caller graph for this function:



### 4.10.4 Field Documentation

#### 4.10.4.1 erkezes

```
Ido Vonat::erkezes [private]
```

Indulási idő

Definition at line 22 of file [vonat.h](#).

#### 4.10.4.2 indulas

```
Ido Vonat::indulas [private]
```

Kocsik darabszáma, basically semmit sem csinál.

Definition at line 21 of file [vonat.h](#).

#### 4.10.4.3 indulo

```
Allomas Vonat::indulo [private]
```

Vonatszám.

Definition at line 18 of file [vonat.h](#).

#### 4.10.4.4 kocsidb

```
int Vonat::kocsidb [private]
```

Végállomás.

Definition at line 20 of file [vonat.h](#).

#### 4.10.4.5 szam

```
int Vonat::szam [private]
```

Definition at line 17 of file [vonat.h](#).

#### 4.10.4.6 veg

```
Allomas Vonat::veg [private]
```

Kiinduló állomás.

Definition at line 19 of file [vonat.h](#).

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

- [vonat.h](#)

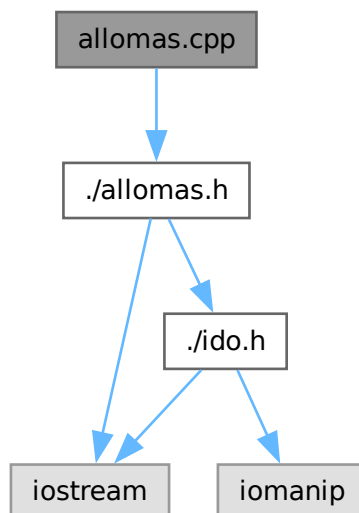
## Chapter 5

# File Documentation

### 5.1 allomas.cpp File Reference

```
#include "../allomas.h"
```

Include dependency graph for allomas.cpp:



#### Functions

- `std::ostream & operator<< (std::ostream &os, const Allomas &allomas)`  
*Allomas* *op<< overload.*

## 5.1.1 Function Documentation

### 5.1.1.1 operator<<()

```
std::ostream & operator<< (
    std::ostream & os,
    const Allomas & allomas )
```

Allomas op<< overload.

Definition at line 4 of file [allomas.cpp](#).

```
00004 { return os << allomas.getAllomas(); }
```

Here is the call graph for this function:



## 5.2 allomas.cpp

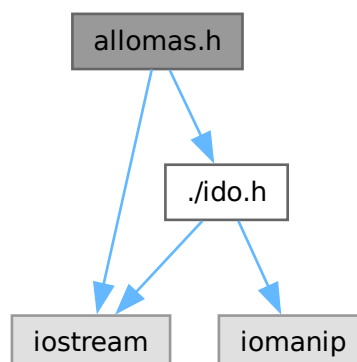
[Go to the documentation of this file.](#)

```
00001 #include "allomas.h"
00002
00003 // Másképp nem ette meg a compiler.
00004 std::ostream& operator<<(std::ostream& os, const Allomas& allomas) { return os << allomas.getAllomas(); }
```

## 5.3 allomas.h File Reference

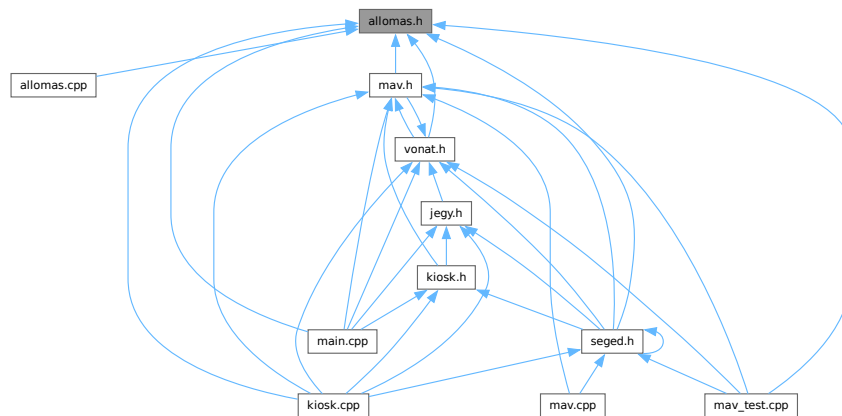
```
#include <iostream>
#include "ido.h"
```

Include dependency graph for allomas.h:





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



## Data Structures

- class [Allomas](#)

## Functions

- `std::ostream & operator<< (std::ostream &os, const Allomas &allomas)`  
[Allomas](#) *op*<< *overload*.

## 5.3.1 Function Documentation

### 5.3.1.1 operator<<()

```
std::ostream & operator<< (
    std::ostream & os,
    const Allomas & allomas )
```

[Allomas](#) *op*<< *overload*.

Definition at line 4 of file [allomas.cpp](#).

```
00004 { return os << allomas.getAllomas(); }
```

Here is the call graph for this function:



## 5.4 allomas.h

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

```

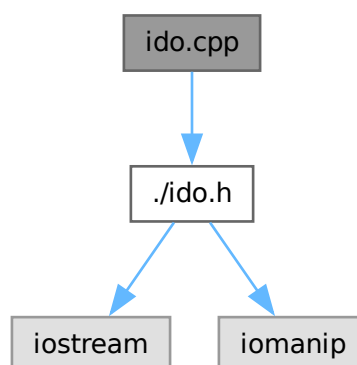
00001 #ifndef ALLOMASH_H
00002 #define ALLOMASH_H
00003
00004 /* file ALLOMASH_H */
00005
00006 // STD::STRING-GEL MEGY A CUCCOKS
00007
00008 #include <iostream>
00009 #include "../ido.h"
00010
00011 /** STRINGGEL ÚJRAÍRVA */
00012 class Allomas{
00013 private:
00014     std::string allomasNev; /// Allomas nevét tartalmazó string
00015 public:
00016     /**
00017      * Allomas konstruktor
00018      * @param allomas itt egy char* egyértelműen.
00019      * a másik verzióban const, hogy fel lehessen közvetlen tölteni
00020      * értsd: "Álloms_neve" mint paraméter.
00021      */
00022     Allomas(const char* allomas) : allomasNev(allomas) {}
00023     Allomas(char* allomas) : allomasNev(allomas) {}
00024     // Allomas(const std::string& allomas) : allomasNev(allomas) {}
00025
00026     // Getterek
00027     std::string getAllomas() const { return allomasNev; }
00028
00029 }; // End of ALLOMAS
00030 /// Allomas op« overload
00031 std::ostream& operator<<(std::ostream& os, const Allomas& allomas);
00032
00033 #endif // !ALLOMASH_H

```

## 5.5 ido.cpp File Reference

```
#include "../ido.h"
```

Include dependency graph for ido.cpp:



### Functions

- `std::ostream & operator<< (std::ostream &os, const Ido &ido)`  
*Ido class op<<() overwreje, a fájlba beíráskor hívódik meg. (addTrain)*

## 5.5.1 Function Documentation

### 5.5.1.1 operator<<()

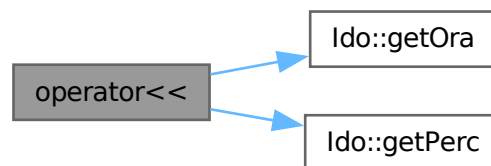
```
std::ostream & operator<< (
    std::ostream & os,
    const Ido & ido )
```

Ido class op<<() overwrireja, a fájlba beíráskor hívódik meg. (addTrain)

Definition at line 3 of file ido.cpp.

```
00003 {
00004     os << std::setw(2) << std::setfill('0') << ido.getOra();
00005     os << std::setw(2) << std::setfill('0') << ido.getPerc();
00006     return os;
00007 } // operator<<()
```

Here is the call graph for this function:



## 5.6 ido.cpp

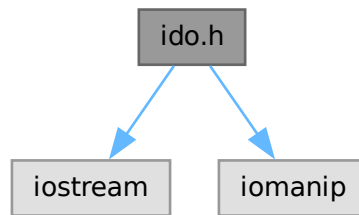
[Go to the documentation of this file.](#)

```
00001 #include "../ido.h"
00002
00003 std::ostream& operator<<(std::ostream& os, const Ido& ido){
00004     os << std::setw(2) << std::setfill('0') << ido.getOra();
00005     os << std::setw(2) << std::setfill('0') << ido.getPerc();
00006     return os;
00007 } // operator<<()
00008
```

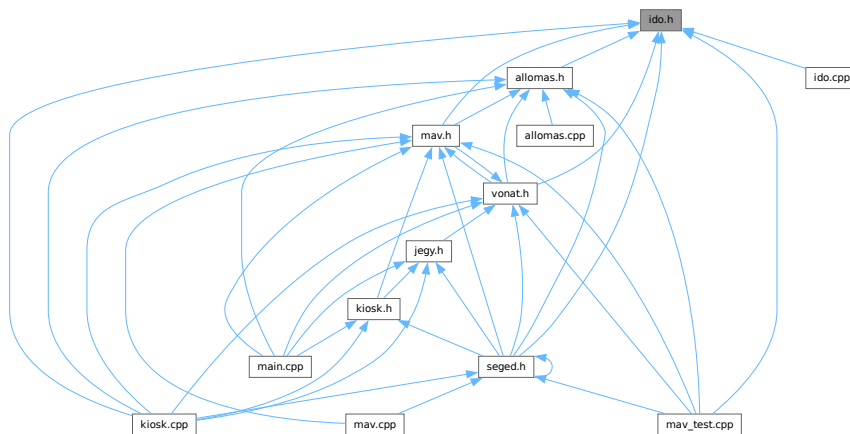
## 5.7 ido.h File Reference

```
#include <iostream>
#include <iomanip>
```

Include dependency graph for ido.h:



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



## Data Structures

- class `Ido`

## Functions

- `std::ostream & operator<< (std::ostream &os, const Ido &ido)`  
*Ido class op<<() overwreija, a fájlba beíráskor hívódik meg. (addTrain)*

## 5.7.1 Function Documentation

### 5.7.1.1 operator<<()

```

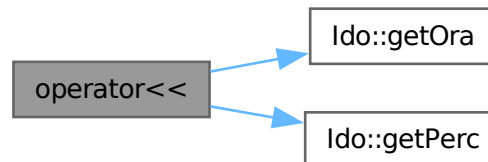
std::ostream & operator<< (
    std::ostream & os,
    const Ido & ido )
  
```

`Ido` class `op<<()` overwrireja, a fájlba beíráskor hívódik meg. (`addTrain`)

Definition at line 3 of file `ido.cpp`.

```
00003 {
00004     os << std::setw(2) << std::setfill('0') << ido.getOra();
00005     os << std::setw(2) << std::setfill('0') << ido.getPerc();
00006     return os;
00007 } // operator<<()
```

Here is the call graph for this function:



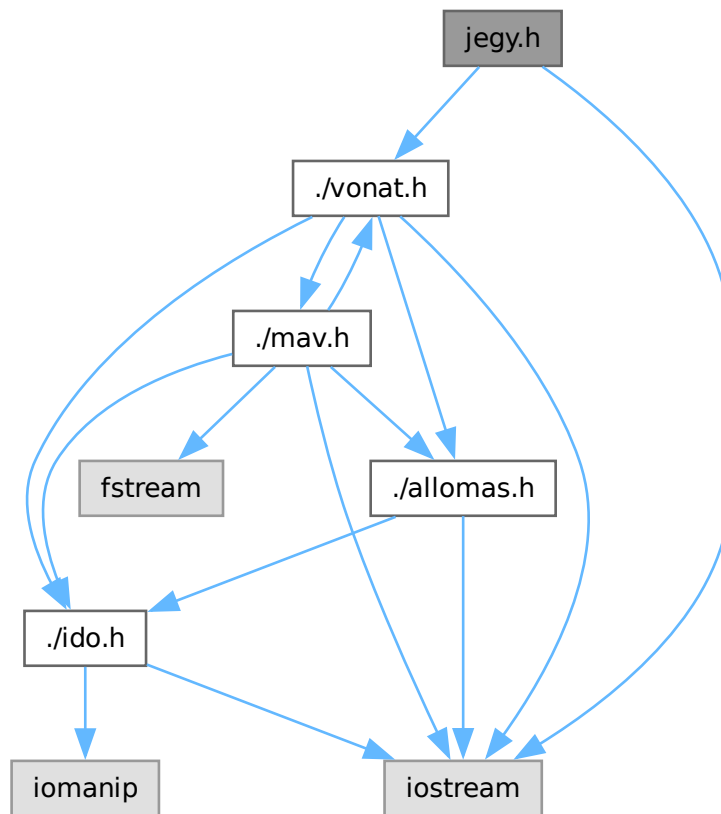
## 5.8 ido.h

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

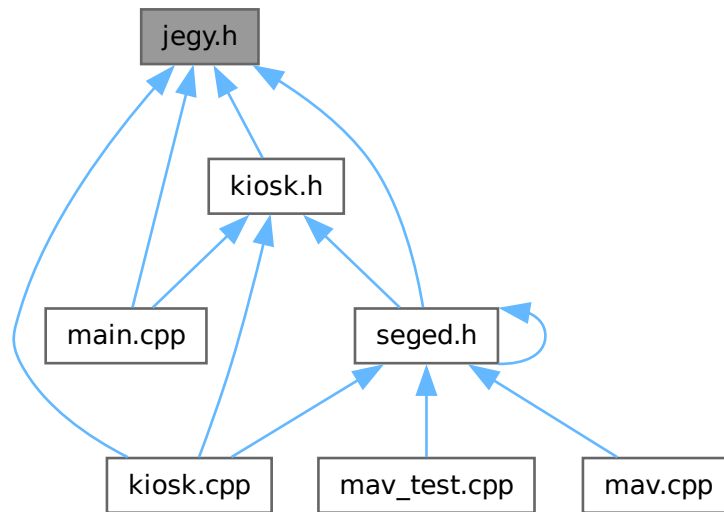
```
00001 #ifndef IDO_H
00002 #define IDO_H
00003
00004 /*
00005 * FILE IDO_H
00006 */
00007
00008 #include <iostream>
00009 #include <iomanip>
00010
00011
00012 class Ido{
00013     int ora;
00014     int perc;
00015 public:
00016     // Ido(int ora = 0, int perc = 0) : ora(ora), perc(perc) {}
00017     /**
00018      * Konstruktor
00019      * @param ido mivel beolvasáskor char* típusokba olvasok bele
00020      * char*-ot kap paraméterként és azt kezeli megfelelően konvertálva.
00021      */
00022     Ido(const char* ido) {
00023         ora = (ido[0] - '0') * 10 + (ido[1] - '0');
00024         perc = (ido[2] - '0') * 10 + (ido[3] - '0');
00025     }
00026
00027     Ido(int ora, int perc) : ora(ora), perc(perc) {}
00028
00029     // Setter
00030     void setIdo(char* ido){
00031         ora = (ido[0] - '0') * 10 + (ido[1] - '0');
00032         perc = (ido[2] - '0') * 10 + (ido[3] - '0');
00033     }
00034
00035     // Gettere
00036     int getOra() const { return ora; };
00037     int getPerc() const { return perc; };
00038
00039
00040
00041 }; // END OF IDO
00042
00043 /// Ido class op<<() overwrireja, a fájlba beíráskor hívódik meg. (addTrain)
00044 std::ostream& operator<<(std::ostream& os, const Ido& ido);
00045
00046
00047
00048 #endif // !IDO_H
```

## 5.9 jegy.h File Reference

```
#include <iostream>
#include "../vonat.h"
Include dependency graph for jegy.h:
```



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



## Data Structures

- class [Jegy](#)
- class [Diak](#)
- class [Teljes](#)
- class [Kutya](#)

## 5.10 jegy.h

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

```

00001 #ifndef JEGY_H
00002 #define JEGY_H
00003
00004 /**
00005  * FILE JEGY_H
00006  */
00007
00008 #include <iostream>
00009 #include "../vonat.h"
00010
00011 class Jegy {
00012     int ar; /// A jegy ára
00013     double szazalek; /// Kedvezmény százaléka, pontosabban mennyi az amit fizet százalék.
00014
00015 public:
00016     /**
00017      * Anya osztály Konstruktora
00018      * @param ar megadja a jegy árát
00019      * @param szazalek a kedvezmény értékét adja meg
00020      */
00021     Jegy(int ar = 1200, double szazalek = 0.7) : ar(ar), szazalek(szazalek) {}
00022
00023     /// Fizetendő összeg, ár és a kedvezmény szorzata
00024     virtual int fizetendo() { return ar * szazalek; }
00025
00026     /// Getterek
00027     int getAr() { return ar; }
  
```

```

00028 double getSzasz() { return szazalek; }
00029
00030 /**
00031  * @param Vonat& megadott vonatra vonatkozik
00032  * Kiírja a jegy adatainak egy felét. Azt a felét ami minden leszármazottál
00033  * azonos. A felső pár so.
00034  * virtual mivel a leszármazottak megöröklík és felülírják.
00035  */
00036 virtual void nyomtat(Vonat& v) {
00037     std::cout << "Fizetendő: " << fizetendo() << "JMF\n";
00038     std::cout << "### Vonatod adatai ###" << std::endl;
00039     v.kiir();
00040     std::cout << std::endl;
00041 }
00042 // virtual void abstract() = 0 {}
00043
00044 }; // END OF JEGY
00045
00046
00047
00048 class Diak : public Jegy {
00049     const std::string tipusNev = "Diák";
00050     // const double szazalek = 0.7;
00051
00052 public:
00053     /**
00054      * Konstruktor
00055      * @param ar megadja a jegy árát
00056      * @param szazalek a kedvezmény értékét adja meg
00057      * Meghívja a anya-class konstruktorát.
00058      */
00059     Diak(int ar = 1200, const double szazalek = 0.7) : Jegy(ar, szazalek) {}
00060
00061     /// Getterek
00062     std::string getTipus() { return tipusNev; }
00063
00064     /// megörökölt nyomtat fv overwrite-ja
00065     /// meghívja az "eredei", főosztály nyomtatját
00066     void nyomtat(Vonat &v) {
00067         std::cout << "### Jegy adatai ###\n";
00068         std::cout << "Jegy típusa: " << getTipus() << std::endl;
00069         Jegy::nyomtat(v);
00070     }
00071
00072 }; // end of diak
00073
00074
00075 class Teljes : public Jegy {
00076     const std::string tipusNev = "Teljes";
00077     // const double szazalek = 1.0;
00078
00079 public:
00080     /**
00081      * Konstruktor
00082      * @param ar megadja a jegy árát
00083      * @param szazalek a kedvezmény értékét adja meg
00084      * Meghívja a anya-class konstruktorát.
00085      */
00086     Teljes(int ar = 1200, const double szazalek = 1.0) : Jegy(ar, szazalek) {}
00087
00088     std::string getTipus() { return tipusNev; }
00089
00090     /// megörökölt nyomtat fv overwrite-ja
00091     /// meghívja az "eredei", főosztály nyomtatját
00092     void nyomtat(Vonat &v) {
00093         std::cout << "### Jegy adatai ###\n";
00094         std::cout << "Jegy típusa: " << getTipus() << std::endl;
00095         Jegy::nyomtat(v);
00096     }
00097
00098 }; // end of Teljes
00099
00100
00101 class Kutya : public Jegy {
00102     const std::string tipusNev = "Kutya";
00103     const std::string kov = "Szájkosár";
00104
00105 public:
00106     /**
00107      * Konstruktor
00108      * @param ar megadja a jegy árát
00109      * @param szazalek a kedvezmény értékét adja meg
00110      * Meghívja a anya-class konstruktorát.
00111      */
00112     Kutya(int ar = 1200, const double szazalek = 0.2) : Jegy(ar, szazalek) {}
00113
00114     // Getterek

```



```

00115     std::string getTipus() { return tipusNev; }
00116     std::string getKov() { return kov; }
00117
00118     /// megörökölt nyomtat fv overwrite-ja
00119     /// meghívja az "eredei", főosztály nyomtatját
00120     void nyomtat(Vonat &v) {
00121         std::cout << "### Jegy adatai ###\n";
00122         std::cout << "Jegy típusa: " << getTipus() << std::endl;
00123         std::cout << "Követelmény: " << getKov() << std::endl;
00124         Jegy::nyomtat(v);
00125     }
00126
00127 }; // end of Kutya
00128
00129
00130 #endif // !JEGY_H

```

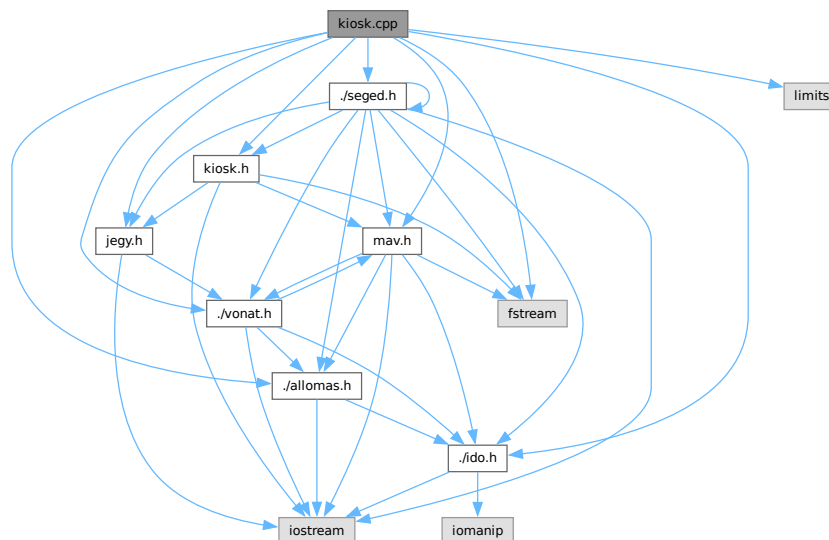
## 5.11 kiosk.cpp File Reference

```

#include "kiosk.h"
#include "jegy.h"
#include "../seged.h"
#include "../mav.h"
#include "../vonat.h"
#include "allomas.h"
#include "ido.h"
#include <fstream>
#include <limits>

```

Include dependency graph for kiosk.cpp:



## 5.12 kiosk.cpp

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

```

00001
00002 #include "kiosk.h"
00003 #include "jegy.h"
00004 #include "../seged.h"
00005

```

```

00006 #include "./mav.h"
00007 #include "./vonat.h"
00008 #include "allomas.h"
00009 #include "ido.h"
00010
00011 #include <fstream>
00012 #include <limits>
00013
00014 /// Érvényes-e az input mikor számot kellene megadnom
00015 bool Kiosk::inputCheck(){
00016     // Kb innen lopva
00017     https://stackoverflow.com/questions/12721911/c-how-to-verify-if-the-data-input-is-of-the-correct-datatype
00018     if (std::cin.fail()) {
00019         std::cout << "Nem sikerült érvényes inputot megadni... PRÓBÁLD ÚJRA" << std::endl;
00020         std::cin.clear();
00021         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
00022         return false;
00023     }
00024     else return true;
00025 }
00026 void Kiosk::init() {
00027     std::cout << "### Dönts ###" << std::endl;
00028     std::cout << "1. Vonat hozzáadása " << std::endl;
00029     std::cout << "2. Vonatok listázása " << std::endl;
00030     std::cout << "3. Jegy nyomtatása" << std::endl;
00031     std::cout << "4. Kilépés" << std::endl;
00032 }
00033
00034 // User input az input streamről enum-má konvertálva
00035 menuItem Kiosk::userInput() {
00036     int valasz;
00037     std::cout << "Választott lehetőség: ";
00038     std::cin > valasz;
00039     return static_cast<menuItem>(valasz);
00040 }
00041
00042 // Liszázza az összes vonatot
00043 void Kiosk::listaz(Mav& mav) {
00044     mav.kiir();
00045 }
00046
00047
00048 void Kiosk::vonatHozza(Mav& mav, std::streampos currPos){
00049     Seged s;
00050
00051     std::cin > s.szam;
00052     // if(*(s.szam) == '\n') std::cout << "FASZ";
00053     std::cin > s.indulo;
00054     std::cin > s.veg;
00055     std::cin > s.kocsidb;
00056     std::cin > s.indulas;
00057     std::cin > s.erkezes;
00058
00059     mav.addTrain(currPos, atoi(s.szam), Allomas(s.indulo), Allomas(s.veg),
00060     atoi(s.kocsidb), Ido(s.indulas), Ido(s.erkezes));
00061 } // End of vonatHozza
00062
00063
00064 // Jegy vásárlás
00065 void Kiosk::jegyValt(Mav& mav){
00066     int idx, buf;
00067     listaz(mav);
00068     std::cout << "Valasz vonatot: ";
00069     std::cin > idx;
00070     // Hibakezelés
00071     if(size_t(idx) > mav.getSize()){ std::cout << "Túlinindexelés\n"; return; }
00072
00073     if(!(Kiosk::inputCheck())) return; // Érvényes inputot szűrök.
00074
00075     std::cout << std::endl; // szép kiírás miatt.
00076     // end if Hibakezelés
00077
00078
00079     std::cout << "1.Teljes, 2.Diak, 3.Kutya\n ";
00080     std::cin > buf;
00081     // Hibakezelés
00082     if(buf > 3){ std::cout << "Túlinindexelés"; return; }
00083
00084     if(!(Kiosk::inputCheck())) return;
00085
00086     std::cout << std::endl;
00087     // end of Hibakezelés
00088
00089     switch (buf) {
00090     case 1: {
00091         Teljes t;

```

```

00092     t.nyomtAt (mav.getVonatAt (idx));
00093     break;
00094 }
00095
00096 case 2: {
00097     Diak d;
00098     d.nyomtAt (mav.getVonatAt (idx));
00099     break;
00100 }
00101
00102 case 3: {
00103     Kutya k;
00104     k.nyomtAt (mav.getVonatAt (idx));
00105     break;
00106 }
00107
00108 } // end of switch
00109
00110 } // end of jegyValt
00111
00112
00113
00114
00115
00116
00117

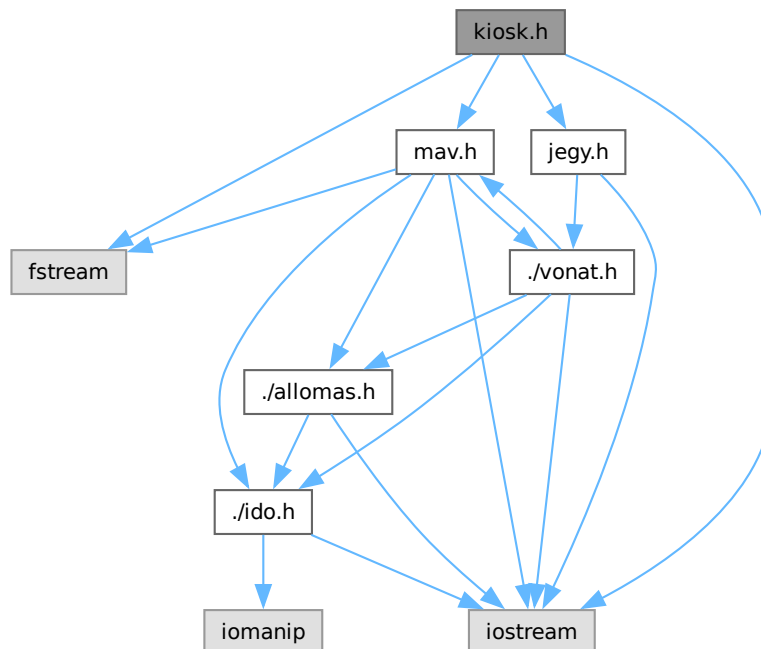
```

## 5.13 kiosk.h File Reference

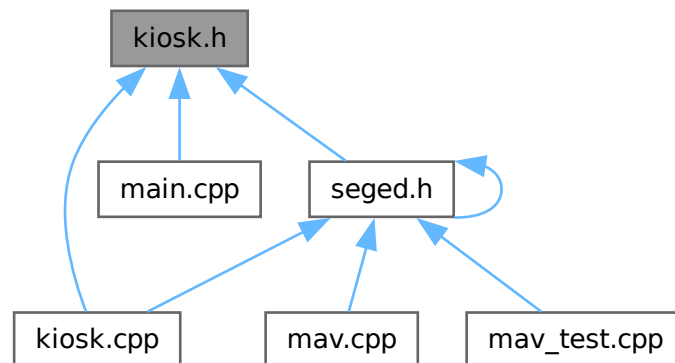
```

#include <iostream>
#include "mav.h"
#include "jegy.h"
#include <fstream>
Include dependency graph for kiosk.h:

```



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



## Data Structures

- class [Kiosk](#)

## Enumerations

- enum [menuItem](#) { [add](#) , [list](#) , [nyomtat](#) , [kilep](#) }

## 5.13.1 Enumeration Type Documentation

### 5.13.1.1 menuItem

```
enum menuItem
```

Segéd "class" A menü kezelését könnyíti meg.

#### Enumerator

add	
list	
nyomtat	
kilep	

Definition at line 16 of file [kiosk.h](#).

```

00016      {
00017          add,
00018          list,
00019          nyomtat,
00020          kilep
00021      };
  
```

## 5.14 kiosk.h

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

```

00001 #ifndef KIOSK_H
00002 #define KIOSK_H
00003
00004 /*
00005  * FILE KIOSK_H
00006  */
00007
00008 #include <iostream>
00009
00010 #include "mav.h"
00011 #include "jegy.h"
00012 #include <fstream>
00013
00014 /// Segéd "class"
00015 /// A menü kezelését könnyíti meg.
00016 enum menuItem{
00017     add,
00018     list,
00019     nyomtat,
00020     kilep
00021 };
00022
00023 class Kiosk {
00024 public:
00025
00026     /// Hibakezelés. Érvényes input esetén @return true, különben @return false
00027     bool inputCheck();
00028     /// Inicializálja a menüt, ergo kiírja a lehetőségeket
00029     void init();
00030     /// Felhasználótól bemenetet kér.
00031     /// @return menuItem fent említett enum class elemet ad vissza
00032     menuItem userInput();
00033
00034     /// Liszázza a vonatokat amiket beolvasott a fájlból
00035     void listaz(Mav& mav);
00036     /// Hozzá ad a user vonatokat a fájlhoz
00037     void vonatHozza(Mav& mav, std::streampos currPos);
00038     /// jegyet vált a kiválasztott vonatra
00039     void jegyValt(Mav& mav);
00040
00041 }; // end of Kiosk
00042
00043
00044
00045
00046
00047
00048 #endif // !KIOSK_H

```

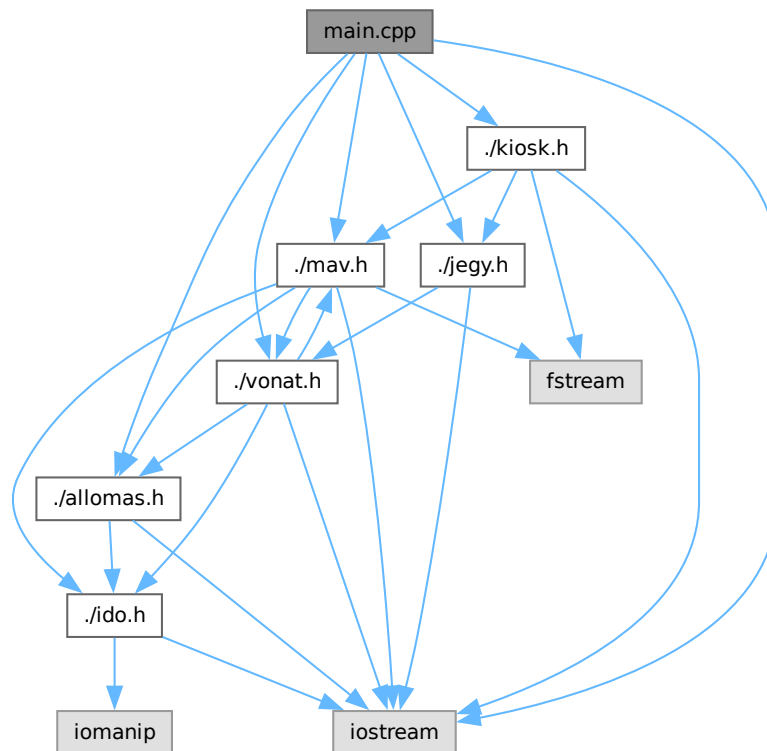
## 5.15 main.cpp File Reference

```

#include <iostream>
#include "../mav.h"
#include "../vonat.h"
#include "../allomas.h"
#include "../jegy.h"
#include "../kiosk.h"

```

Include dependency graph for main.cpp:



## Functions

- int [main](#) ()

## 5.15.1 Function Documentation

### 5.15.1.1 main()

```
int main ( )
```

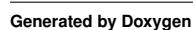
Definition at line 10 of file [main.cpp](#).

```

00010     {
00011         // Itt tárolon hol vagyok a file-ban
00012         // Mind a beolvasás mind pedig az íráshoz KELL!
00013         std::streampos currPos;
00014         Mav mav;
00015         mav.beolvas();
00016         /* KIOSK TEST */
00017
00018
00019         Kiosk k;
00020         k.init(); // Kiírja a lehetőségeket
00021
00022         int choice = 0;
00023         while (choice != 4) {
00024             choice = k.userInput(); // Get user input once per loop iteration
00025             switch (choice) {
00026                 // Vonat hozzáadása

```

Here is the call graph for this function:



## 5.16 main.cpp

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

```

00001 #include <iostream>
00002
00003 #include "../mav.h"
00004 #include "../vonat.h"
00005 #include "../allomas.h"
00006 #include "../jegy.h"
00007 #include "../kiosk.h"
00008
00009
00010 int main(){
00011     // Itt tárolon hol vagyok a file-ban
00012     // Mind a beolvasás mind pedig az íráshoz KELL!
00013     std::streampos currPos;
00014     Mav mav;
00015     mav.beolvas();
00016     /* KIOSK TEST */
00017
00018
00019     Kiosk k;
00020     k.init(); // Kiírja a lehetőségeket
00021
00022     int choice = 0;
00023     while (choice != 4) {
00024         choice = k.userInput(); // Get user input once per loop iteration
00025         switch (choice) {
00026             // Vonat hozzáadása
00027             case 1: {
00028                 std::cout << "Beolvas: " << std::endl;
00029                 std::cout << "Szám Indulo Vég kocsidb Indulás érkezés" << std::endl;
00030                 k.vonatHozza(mav, currPos);
00031                 // system("clear"); // Tábla törlés
00032                 std::cout << "\n" << "Vonat hozzáadva\n\n";
00033                 mav.beolvas();
00034                 k.init(); // Kiírja a lehetőségeket
00035                 break;
00036             }
00037             // Vonatok listázása
00038             case 2: {
00039                 std::cout << std::endl;
00040                 mav.beolvas();
00041                 k.listaz(mav);
00042                 std::cout << std::endl;
00043                 k.init(); // Kiírja a lehetőségeket
00044                 break;
00045             }
00046             // Jegy vásárlás
00047             case 3: {
00048                 std::cout << std::endl;
00049                 k.jegyValt(mav);
00050                 std::cout << std::endl;
00051                 k.init(); // Kiírja a lehetőségeket
00052                 break;
00053             }
00054             // Kilépés
00055             case 4: {
00056                 std::cout << "Kilépés" << std::endl;
00057                 break;
00058             }
00059             default: {
00060                 // std::cout << "Érvénytelen lehetőség" << std::endl;
00061                 // Hibakezelés
00062                 if(!k.inputCheck()) { std::cout << std::endl; k.init(); }
00063             }
00064         } // end of switch
00065     } // end of while
00066
00067     return 0;
00068 } // end of main

```

## 5.17 mav.cpp File Reference

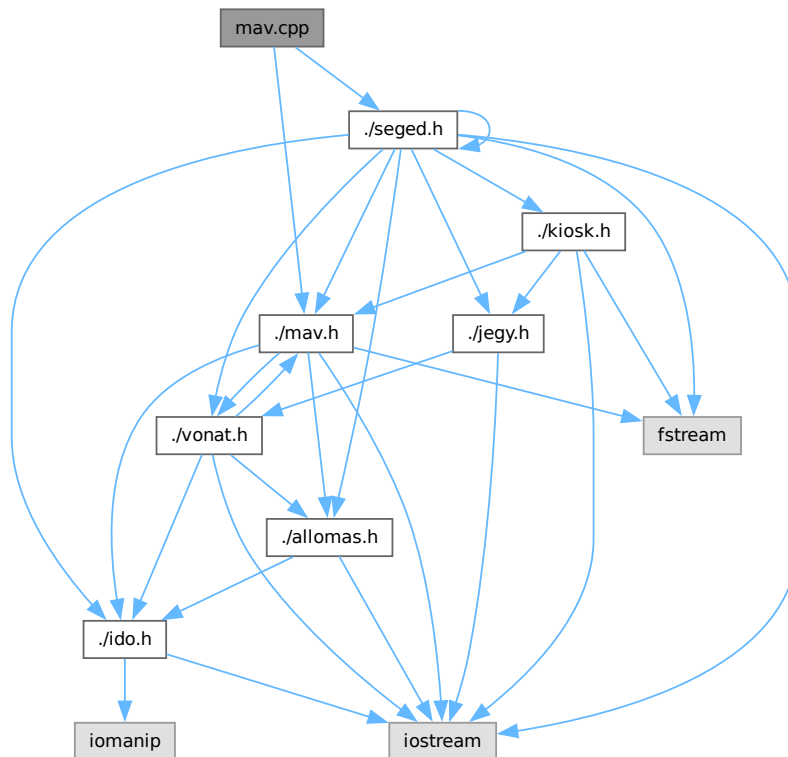
```

#include "../mav.h"
#include "../seged.h"

```



Include dependency graph for mav.cpp:



## 5.18 mav.cpp

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

```

00001 #include "mav.h"
00002 #include "seged.h"
00003
00004
00005 /**
00006  * Beolvasás a file-ból egy segéd strukturába, onnan pedig egy Vonat objektum feltöltés
00007  * @param Mav& mav class feltöltéséhez szükséges
00008  */
00009 void Mav::beolvas() {
00010     // Inicializálás
00011     const char* vonatok_file = "vonatok.txt";
00012     std::ifstream file(vonatok_file);
00013     Vonat v;
00014     Seged seged;
00015
00016     if (!file.is_open()) {
00017         std::cout << "Megnyitással van baj " << vonatok_file << std::endl;
00018         return;
00019     }
00020
00021     // Kiürítem a vonatok kollekciót, hogy ne legyen ráolvasás
00022     delete[] vonatok;
00023     vonatok = nullptr;
00024     sz = 0;
00025
00026     while (file >> seged.szam >> seged.indulo >> seged.veg >> seged.kocsidb >> seged.indulas >> seged.erkezes)
00027     {
00028         v.setSzam(std::atoi(seged.szam));
00029         v.setIndulo(seged.indulo);
00030         v.setVeg(seged.veg);
00031         v.setKocsidb(std::atoi(seged.kocsidb));
00032     }
00033 }

```

```

00031     v.setIndulas(seged.indulas);
00032     v.setErkezes(seged.erkezes);
00033
00034     // feltöltöm a MAV "mgmt" class Vonat* tömbjét
00035     this->add(v);
00036 }
00037 file.close();
00038 } // END OF BEOLVAS
00039
00040 /*
00041 * Vonatok hozzáadása
00042 * @param currPos a streamben elfoglalt helyem
00043 * @param minden-más a vonat class feltöltéséhez szükséges adatok
00044 */
00045 void Mav::addTrain(std::streampos& currPos, int szam, Allomas indulo,
00046                   Allomas veg, int kocsidb, Ido indulas, Ido erkezes){
00047     // Inicializálás
00048     const char* vonatok = "./vonatok.txt";
00049     std::ofstream file (vonatok, std::ios::app); // Hozzáfűzésesen nyitom meg.
00050
00051     // HIBAKEZEELÉS IDE
00052     if (!file.is_open()) {
00053         std::cout << "Megynitással van baj " << vonatok << std::endl;
00054         return;
00055     }
00056     // end of HIBAKEZEELÉS
00057
00058     if(file.is_open()){
00059         file.seekp(currPos); // Megfelelő helyre ugrok
00060
00061         file << szam << " " << indulo << " " << veg << " " << kocsidb << " "
00062             << indulas << " " << erkezes << std::endl;
00063     }
00064
00065     file.close();
00066 } // END OF addTrain
00067
00068

```

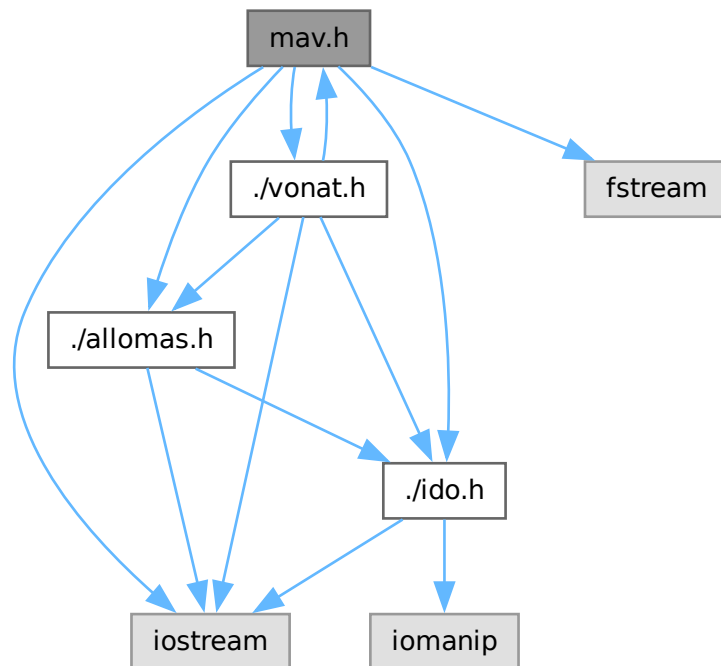
## 5.19 mav.h File Reference

```

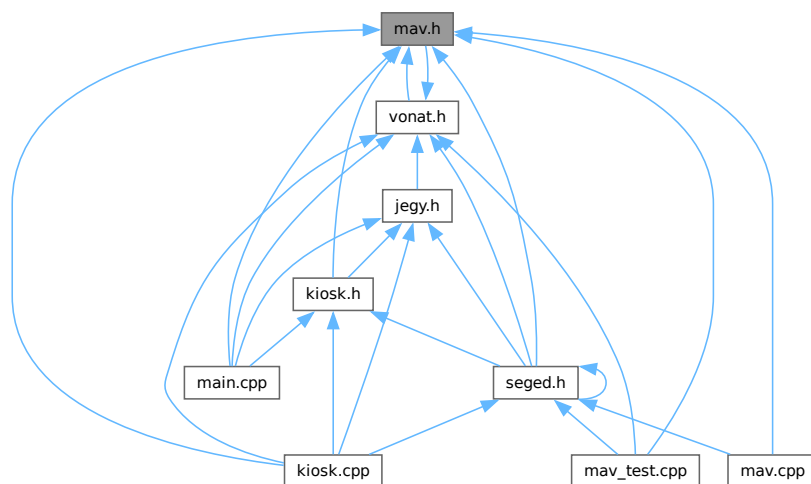
#include <iostream>
#include <fstream>
#include " ./vonat.h"
#include " ./allomas.h"
#include " ./ido.h"

```

Include dependency graph for mav.h:



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



## Data Structures

- class [Mav](#)

## 5.20 mav.h

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

```

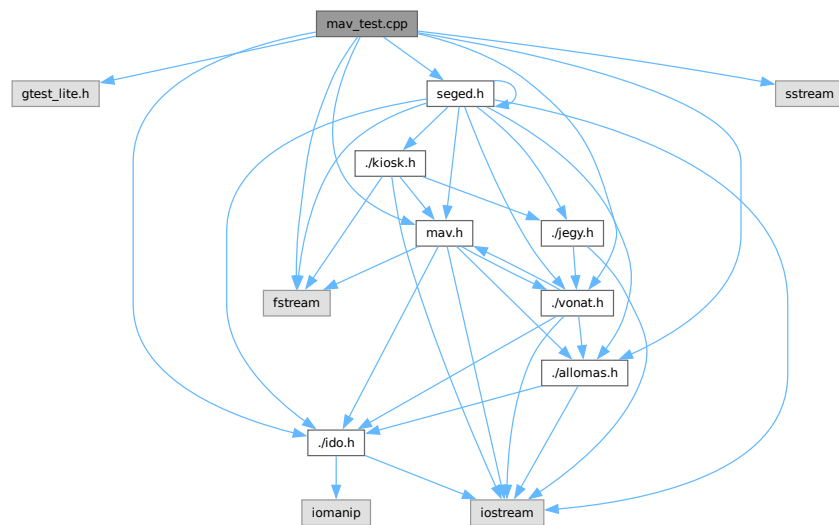
00001 #ifndef MAV_H
00002 #define MAV_H
00003
00004 /*
00005  * FILE MAV_H
00006  */
00007
00008 #include <iostream>
00009 #include <fstream>
00010
00011 #include "../vonat.h"
00012 #include "../allomas.h"
00013 #include "../ido.h"
00014
00015
00016 class Mav {
00017     Vonat* vonatok; // Vonatok kollekcio
00018     size_t si;      // Vonatok kollekcio merete db szamban
00019
00020 public:
00021     // Konstruktor
00022     Mav(size_t si = 0) : vonatok(nullptr), si(si) {}
00023
00024     /**
00025      * Elem hozzafuzese Vonatok kollekciohoz
00026      * @param vonat vonat referencia amit hozzAAD
00027      */
00028     void add(Vonat& vonat){
00029         Vonat* temp = new Vonat[si + 1]; // Lefoglalok helyet
00030
00031         for (size_t i = 0; i < si; ++i) { // Atmasolom a tomb elemeit
00032             temp[i] = vonatok[i];
00033         }
00034         temp[si++] = vonat; // Beleteszem a vonatot
00035
00036         delete[] vonatok; // "Regi" tombot torlom
00037         vonatok = temp; //
00038     } // End of feltolt
00039
00040     // Vonatok tomb kiirasa
00041     void kiir() {
00042         for (size_t i = 0; i < si; ++i) {
00043             std::cout << "--- " << i+1 << ".vonat ---" << std::endl;
00044             vonatok[i].kiir();
00045             std::cout << std::endl;
00046         }
00047     } // end of kiir
00048
00049     // i.edik elem kiirasa
00050     void kiirAt(int i){
00051         // HIBAEKEZELES
00052         std::cout << i+1 << ".vonat" << std::endl;
00053         vonatok[i].kiir();
00054     } // end of kiirAt
00055
00056     Vonat& getVonatAt(int i){ return vonatok[i-1]; }
00057     size_t getSize() { return static_cast<int>(si); }
00058
00059     Vonat& operator[] (int idx){ return vonatok[idx]; }
00060
00061     /**
00062      * @param fasz geci
00063      * @param fasz geic
00064      */
00065     void beolvas();
00066     void addTrain(std::streampos& currPos, int szam, Allomas indulo,
00067                 Allomas veg, int kocsidb, Idó indulas, Idó érkezés);
00068
00069     // Destruktor
00070     ~Mav(){
00071         delete[] vonatok;
00072     }
00073
00074 }; // END OF MAV
00075
00076
00077
00078
00079
00080 #endif // !MAV_H
00081

```

## 5.21 mav\_test.cpp File Reference

```
#include "gtest_lite.h"
#include "mav.h"
#include "vonat.h"
#include "allomas.h"
#include "ido.h"
#include "seged.h"
#include <sstream>
#include <fstream>
```

Include dependency graph for mav\_test.cpp:



### Functions

- void `testMavAdd` ()
- void `testMavKiir` ()
- void `testMavKiirAt` ()
- void `testMavBeolvas` ()
- void `testMavAddTrain` ()
- int `main` ()

### 5.21.1 Function Documentation

#### 5.21.1.1 main()

```
int main ( )
```

Definition at line 122 of file `mav_test.cpp`.

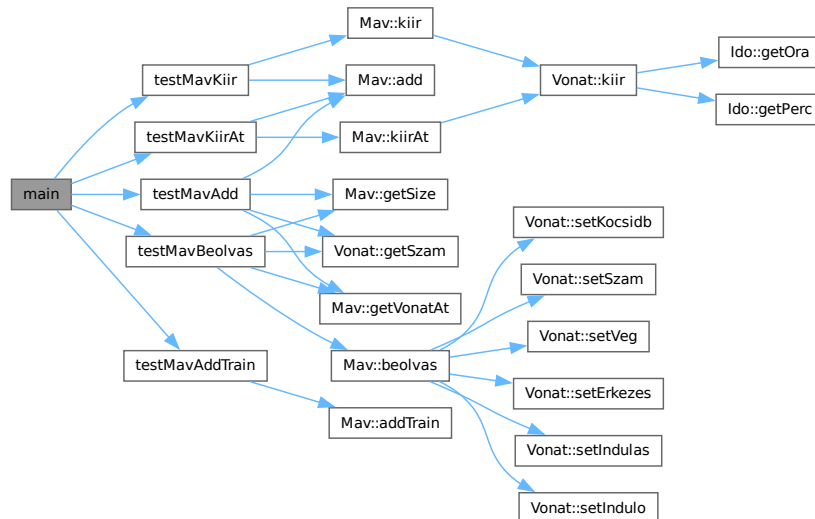
```
00122     {
00123         testMavAdd();
00124         testMavKiir();
00125         testMavKiirAt();
00126         testMavBeolvas();
00127         testMavAddTrain();
00128     }
```

```

00129     GTEND(std::cerr); // Csak C(J)PORTA működéséhez kell
00130     return 0;
00131 }

```

Here is the call graph for this function:



### 5.21.1.2 testMavAdd()

```
void testMavAdd ( )
```

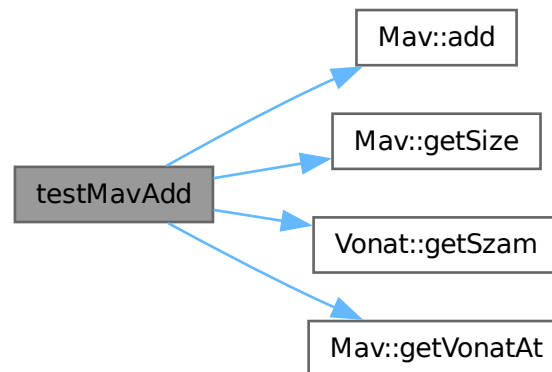
Definition at line 10 of file [mav\\_test.cpp](#).

```

00010     {
00011         Mav mav;
00012         Vonat vonat1(1, "Budapest", "Szeged", 10, Ido(8, 0), Ido(10, 0));
00013         Vonat vonat2(2, "Szeged", "Debrecen", 8, Ido(12, 0), Ido(14, 0));
00014
00015         mav.add(vonat1);
00016         mav.add(vonat2);
00017
00018         TEST(Mav, AddFunction) {
00019             EXPECT_EQ(mav.getSize(), 2);
00020             EXPECT_EQ(mav.getVonatAt(1).getSzam(), 1);
00021             EXPECT_EQ(mav.getVonatAt(2).getSzam(), 2);
00022         } END
00023     }

```

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.21.1.3 testMavAddTrain()

```
void testMavAddTrain ( )
```

Definition at line 90 of file [mav\\_test.cpp](#).

```

00090     {
00091         // Create a temporary test file
00092         const char* testFilename = "./test_vonatok.txt";
00093         std::ofstream outFile(testFilename);
00094         outFile << "1 Budapest Szeged 10 08:00 10:00\n";
00095         outFile.close();
00096
00097         // Change the file to be read by the Mav class to the test file
00098         std::rename(testFilename, "./vonatok.txt");
00099
00100         Mav mav;
00101         std::streampos currPos = 0;
00102
00103         TEST(Mav, AddTrainFunction) {
00104             Allomas indulo("Szeged"), veg("Debrecen");
00105             Ido indulas(12, 0), erkezes(14, 0);
00106             mav.addTrain(currPos, 2, indulo, veg, 8, indulas, erkezes);
00107
00108             std::ifstream inFile("./vonatok.txt");
00109             std::stringstream buffer;
00110             buffer << inFile.rdbuf();
00111             inFile.close();
00112         }
  
```

```

00113         std::string expectedContent = "1 Budapest Szeged 10 08:00 10:00\n"
00114                                         "2 Szeged Debrecen 8 12:00 14:00\n";
00115         EXPECT_EQ(buffer.str(), expectedContent);
00116     } END
00117
00118     // Clean up the test file
00119     std::remove("./vonatok.txt");
00120 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



#### 5.21.1.4 testMavBeolvas()

```
void testMavBeolvas ( )
```

Definition at line 67 of file [mav\\_test.cpp](#).

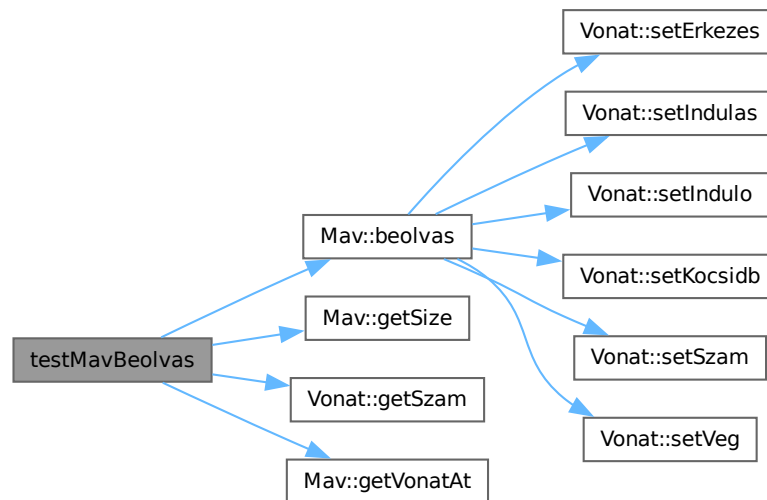
```

00067     {
00068         // Create a temporary test file
00069         const char* testFilename = "./test_vonatok.txt";
00070         std::ofstream outFile(testFilename);
00071         outFile << "1 Budapest Szeged 10 08:00 10:00\n"
00072                 << "2 Szeged Debrecen 8 12:00 14:00\n";
00073         outFile.close();
00074
00075         // Change the file to be read by the Mav class to the test file
00076         Mav mav;
00077         std::rename(testFilename, "./vonatok.txt");
00078
00079         TEST(Mav, BeolvasFunction) {
00080             mav.beolvas();
00081             EXPECT_EQ(mav.getSize(), 2);
00082             EXPECT_EQ(mav.getVonatAt(1).getSzam(), 1);
00083             EXPECT_EQ(mav.getVonatAt(2).getSzam(), 2);
00084         } END
00085
00086         // Clean up the test file
00087         std::remove("./vonatok.txt");
00088     }

```



Here is the call graph for this function:



Here is the caller graph for this function:



### 5.21.1.5 testMavKiir()

```
void testMavKiir ( )
```

Definition at line 25 of file `mav_test.cpp`.

```

00025     {
00026         Mav mav;
00027         Vonat vonat1(1, "Budapest", "Szeged", 10, Ido(8, 0), Ido(10, 0));
00028         Vonat vonat2(2, "Szeged", "Debrecen", 8, Ido(12, 0), Ido(14, 0));
00029
00030         mav.add(vonat1);
00031         mav.add(vonat2);
00032
00033         TEST(Mav, KiirFunction) {
00034             std::stringstream output;
00035             std::streambuf* oldCoutBuffer = std::cout.rdbuf(output.rdbuf());
00036             mav.kiir();
00037             std::cout.rdbuf(oldCoutBuffer);
00038
00039             std::string expectedOutput = "--- 1.vonat ---\n"
00040                                         "1: Budapest -> Szeged, 10 kocsival, Indulás: 08:00, Érkezés:
00041                                         "10:00\n\n"
00042                                         "--- 2.vonat ---\n"

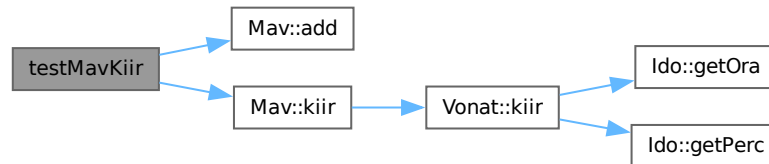
```

```

00042                                     "2: Szeged -> Debrecen, 8 kocsival, Indulás: 12:00, Érkezés:
00043     14:00\n\n";
00043     EXPECT_EQ(output.str(), expectedOutput);
00044 } END
00045 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



### 5.21.1.6 testMavKiirAt()

```
void testMavKiirAt ( )
```

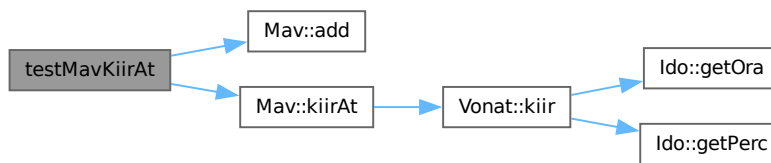
Definition at line 47 of file [mav\\_test.cpp](#).

```

00047     {
00048         Mav mav;
00049         Vonat vonat1(1, "Budapest", "Szeged", 10, Ido(8, 0), Ido(10, 0));
00050         Vonat vonat2(2, "Szeged", "Debrecen", 8, Ido(12, 0), Ido(14, 0));
00051
00052         mav.add(vonat1);
00053         mav.add(vonat2);
00054
00055         TEST(Mav, KiirAtFunction) {
00056             std::stringstream output;
00057             std::streambuf* oldCoutBuffer = std::cout.rdbuf(output.rdbuf());
00058             mav.kiirAt(0);
00059             std::cout.rdbuf(oldCoutBuffer);
00060
00061             std::string expectedOutput = "1.vonat\n"
00062                                     "1: Budapest -> Szeged, 10 kocsival, Indulás: 08:00, Érkezés:
00063     10:00\n\n";
00063     EXPECT_EQ(output.str(), expectedOutput);
00064 } END
00065 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



## 5.22 mav\_test.cpp

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

```

00001 #include "gtest_lite.h"
00002 #include "mav.h"
00003 #include "vonat.h"
00004 #include "allomas.h"
00005 #include "ido.h"
00006 #include "seged.h"
00007 #include <sstream>
00008 #include <fstream>
00009
00010 void testMavAdd() {
00011     Mav mav;
00012     Vonat vonat1(1, "Budapest", "Szeged", 10, Ido(8, 0), Ido(10, 0));
00013     Vonat vonat2(2, "Szeged", "Debrecen", 8, Ido(12, 0), Ido(14, 0));
00014
00015     mav.add(vonat1);
00016     mav.add(vonat2);
00017
00018     TEST(Mav, AddFunction) {
00019         EXPECT_EQ(mav.getSize(), 2);
00020         EXPECT_EQ(mav.getVonatAt(1).getSzam(), 1);
00021         EXPECT_EQ(mav.getVonatAt(2).getSzam(), 2);
00022     } END
00023 }
00024
00025 void testMavKiir() {
00026     Mav mav;
00027     Vonat vonat1(1, "Budapest", "Szeged", 10, Ido(8, 0), Ido(10, 0));
00028     Vonat vonat2(2, "Szeged", "Debrecen", 8, Ido(12, 0), Ido(14, 0));
00029
00030     mav.add(vonat1);
00031     mav.add(vonat2);
00032
00033     TEST(Mav, KiirFunction) {
00034         std::stringstream output;
00035         std::streambuf* oldCoutBuffer = std::cout.rdbuf(output.rdbuf());
00036         mav.kiir();
00037         std::cout.rdbuf(oldCoutBuffer);
00038     }
  
```

```

00039         std::string expectedOutput = "--- 1.vonat ---\n"
00040                                     "1: Budapest -> Szeged, 10 kocsiival, Indulás: 08:00, Érkezés:
10:00\n\n"
00041                                     "--- 2.vonat ---\n"
00042                                     "2: Szeged -> Debrecen, 8 kocsiival, Indulás: 12:00, Érkezés:
14:00\n\n";
00043         EXPECT_EQ(output.str(), expectedOutput);
00044     } END
00045 }
00046
00047 void testMavKiirAt() {
00048     Mav mav;
00049     Vonat vonat1(1, "Budapest", "Szeged", 10, Ido(8, 0), Ido(10, 0));
00050     Vonat vonat2(2, "Szeged", "Debrecen", 8, Ido(12, 0), Ido(14, 0));
00051
00052     mav.add(vonat1);
00053     mav.add(vonat2);
00054
00055     TEST(Mav, KiirAtFunction) {
00056         std::stringstream output;
00057         std::streambuf* oldCoutBuffer = std::cout.rdbuf(output.rdbuf());
00058         mav.kiirAt(0);
00059         std::cout.rdbuf(oldCoutBuffer);
00060
00061         std::string expectedOutput = "1.vonat\n"
00062                                     "1: Budapest -> Szeged, 10 kocsiival, Indulás: 08:00, Érkezés:
10:00\n\n";
00063         EXPECT_EQ(output.str(), expectedOutput);
00064     } END
00065 }
00066
00067 void testMavBeolvas() {
00068     // Create a temporary test file
00069     const char* testFilename = "./test_vonatok.txt";
00070     std::ofstream outFile(testFilename);
00071     outFile << "1 Budapest Szeged 10 08:00 10:00\n"
00072              << "2 Szeged Debrecen 8 12:00 14:00\n";
00073     outFile.close();
00074
00075     // Change the file to be read by the Mav class to the test file
00076     Mav mav;
00077     std::rename(testFilename, "./vonatok.txt");
00078
00079     TEST(Mav, BeolvasFunction) {
00080         mav.beolvas();
00081         EXPECT_EQ(mav.getSize(), 2);
00082         EXPECT_EQ(mav.getVonatszamlas(1).getSzamlas(), 1);
00083         EXPECT_EQ(mav.getVonatszamlas(2).getSzamlas(), 2);
00084     } END
00085
00086     // Clean up the test file
00087     std::remove("./vonatok.txt");
00088 }
00089
00090 void testMavAddTrain() {
00091     // Create a temporary test file
00092     const char* testFilename = "./test_vonatok.txt";
00093     std::ofstream outFile(testFilename);
00094     outFile << "1 Budapest Szeged 10 08:00 10:00\n";
00095     outFile.close();
00096
00097     // Change the file to be read by the Mav class to the test file
00098     std::rename(testFilename, "./vonatok.txt");
00099
00100     Mav mav;
00101     std::streampos currPos = 0;
00102
00103     TEST(Mav, AddTrainFunction) {
00104         Allomas indulo("Szeged"), veg("Debrecen");
00105         Ido indulas(12, 0), erkezes(14, 0);
00106         mav.addTrain(currPos, 2, indulo, veg, 8, indulas, erkezes);
00107
00108         std::ifstream inFile("./vonatok.txt");
00109         std::stringstream buffer;
00110         buffer << inFile.rdbuf();
00111         inFile.close();
00112
00113         std::string expectedContent = "1 Budapest Szeged 10 08:00 10:00\n"
00114                                     "2 Szeged Debrecen 8 12:00 14:00\n";
00115         EXPECT_EQ(buffer.str(), expectedContent);
00116     } END
00117
00118     // Clean up the test file
00119     std::remove("./vonatok.txt");
00120 }
00121
00122 int main() {

```

```

00123     testMavAdd();
00124     testMavKiir();
00125     testMavKiirAt();
00126     testMavBeolvas();
00127     testMavAddTrain();
00128
00129     GTEND(std::cerr); // Csak C(J)PORTA működéséhez kell
00130     return 0;
00131 }
00132

```

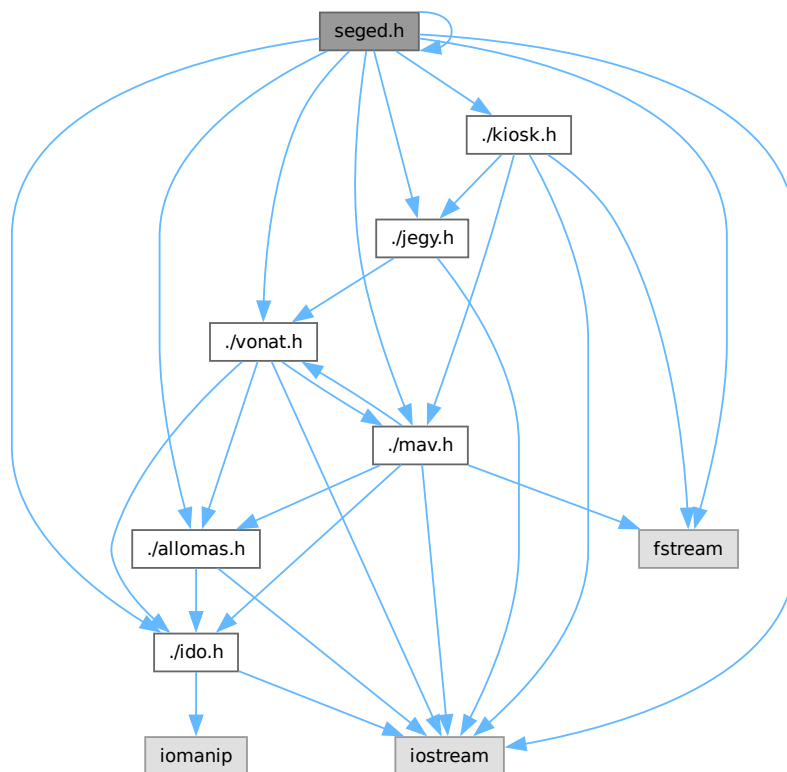
## 5.23 seged.h File Reference

```

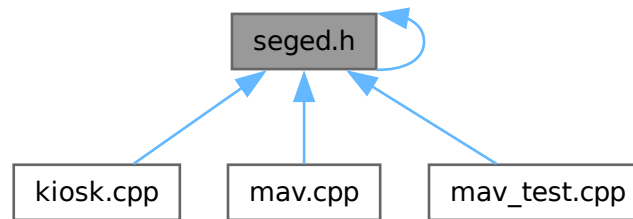
#include <iostream>
#include <fstream>
#include "../allomas.h"
#include "../ido.h"
#include "../jegy.h"
#include "../kiosk.h"
#include "../mav.h"
#include "../seged.h"
#include "../vonat.h"

```

Include dependency graph for seged.h:



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



## Data Structures

- struct [Seged](#)

## 5.24 seged.h

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

```

00001 #ifndef SEGED_H
00002 #define SEGED_H
00003
00004 #include <iostream>
00005 #include <fstream>
00006
00007 #include "../allomas.h"
00008 #include "../ido.h"
00009 #include "../jegy.h"
00010 #include "../kiosk.h"
00011 #include "../mav.h"
00012 #include "../seged.h"
00013 #include "../vonat.h"
00014
00015
00016 /**
00017  * Magyarország leghosszabb településneve 15 karakter
00018  * 25 karakterbe bele kell férnie
00019  */
00020 struct Seged {
00021     static const int bufferSize = 25;
00022     char szam[bufferSize];
00023     char indulo[bufferSize];
00024     char veg[bufferSize];
00025     char kocsidb[bufferSize];
00026     char indulas[bufferSize];
00027     char erkezes[bufferSize];
00028 };
00029
00030
00031
00032 #endif // !SEGED_H
00033
  
```

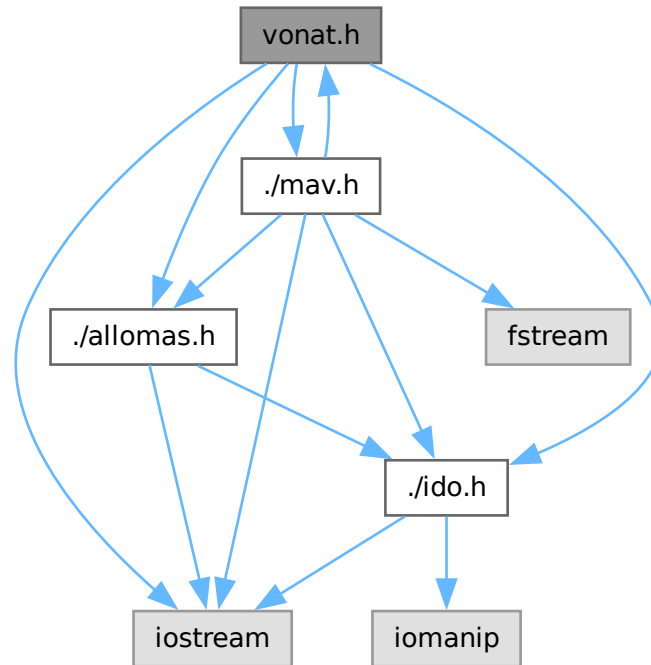
## 5.25 vonat.h File Reference

```

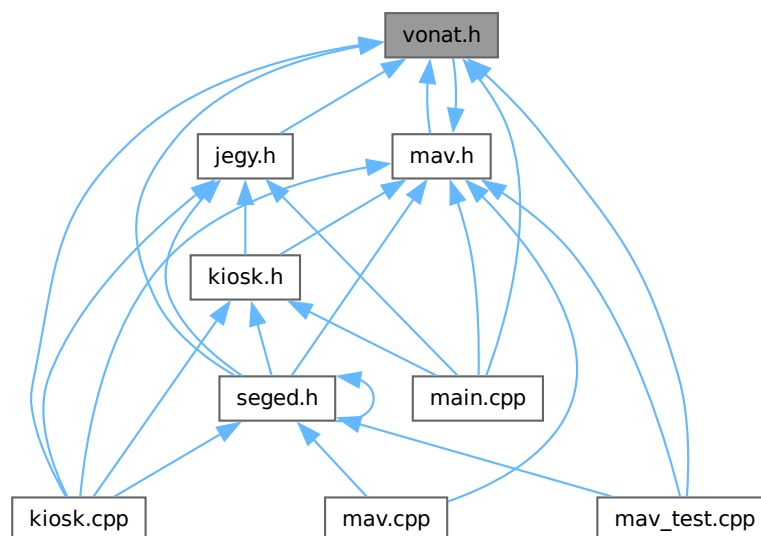
#include <iostream>
#include "../mav.h"
#include "../allomas.h"
  
```

```
#include "../ido.h"
```

Include dependency graph for vonat.h:



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



## Data Structures

- class [Vonat](#)

## 5.26 vonat.h

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

```

00001 #ifndef VONAT_H
00002 #define VONAT_H
00003
00004 /*
00005  * file VONAT_H
00006 */
00007 #include <iostream>
00008
00009 #include "../mav.h"
00010 #include "../allomas.h"
00011 #include "../ido.h"
00012
00013 class Mav; // Nem ette meg a headerből, nem értem miért.
00014
00015 class Vonat {
00016 private:
00017     int szam; // Vonatszám
00018     Allomas indulo; // Kiinduló állomás
00019     Allomas veg; // Végállomás
00020     int kocsidb; // Kocsik darabszáma, basically semmit sem csinál
00021     Ido indulas; // Indulási idő
00022     Ido erkezes; // Érkezési idő
00023
00024 public:
00025     // Paraméter nélküli Konstruktor
00026     Vonat() : szam(0), indulo(""), veg(""), kocsidb(0), indulas(0,0), erkezes(0,0) {}
00027
00028     // Paraméteres Konstruktor
00029     Vonat(int vszam, Allomas indulop, Allomas vegp, int kocsidb, Ido indulasp, Ido erkezesp)
00030         : szam(vszam), indulo(indulop), veg(vegp), kocsidb(kocsidb), indulas(indulasp),
00031           erkezes(erkezesp) {}
00032
00033     // Setterek
00034     void setSzam(int szam) { this->szam = szam; }
00035     void setIndulo(const char* indulo) { this->indulo = Allomas(indulo); }
00036     void setVeg(const char* veg) { this->veg = Allomas(veg); }
00037     void setKocsidb(int kocsidb) { this->kocsidb = kocsidb; }
00038     void setIndulas(const char* ido) { this->indulas = Ido(ido); }
00039     void setErkezes(const char* ido) { this->erkezes = Ido(ido); }
00040
00041     // Getterek
00042     int getSzam() const { return szam; }
00043     Allomas getIndulo() const { return indulo; }
00044     Allomas getVeg() const { return veg; }
00045     int getKocsidb() const { return kocsidb; }
00046     Ido getIndulas() const { return indulas; }
00047     Ido getErkezes() const { return erkezes; }
00048
00049     /**
00050      * Kiírja a vonat adatait
00051      * Főképp a jegyváltáskor van meghívva + listázás.
00052      */
00053     void kiir() const {
00054         std::cout << "Vonat szama: " << szam << std::endl;
00055         std::cout << "Indulasi allomas: " << indulo << std::endl;
00056         std::cout << "Erkezesi allomas: " << veg << std::endl;
00057         std::cout << "Kocsi darabszam: " << kocsidb << std::endl;
00058
00059         std::cout << "Indulas idopontja: ";
00060         std::cout << std::setw(2) << std::setfill('0') << indulas.getOra();
00061         std::cout << ":";
00062         std::cout << std::setw(2) << std::setfill('0') << indulas.getPerc() << std::endl; // Nagyon ronda,
00063         tudom...
00064
00065         std::cout << "Erkezés idopontja: ";
00066         std::cout << std::setw(2) << std::setfill('0') << erkezes.getOra();
00067         std::cout << ":";
00068         std::cout << std::setw(2) << std::setfill('0') << erkezes.getPerc() << std::endl; // Nagyon ronda,
00069         tudom...
00070     }
00071 }
00072
00073
00074
00075
00076
00077
00078
00079
00080
00081
00082
00083
00084
00085
00086
00087
00088
00089
00090
00091
00092
00093
00094
00095
00096
00097
00098
00099
00100
00101
00102
00103
00104
00105
00106
00107
00108
00109
00110
00111
00112
00113
00114
00115
00116
00117
00118
00119
00120
00121
00122
00123
00124
00125
00126
00127
00128
00129
00130
00131
00132
00133
00134
00135
00136
00137
00138
00139
00140
00141
00142
00143
00144
00145
00146
00147
00148
00149
00150
00151
00152
00153
00154
00155
00156
00157
00158
00159
00160
00161
00162
00163
00164
00165
00166
00167
00168
00169
00170
00171
00172
00173
00174
00175
00176
00177
00178
00179
00180
00181
00182
00183
00184
00185
00186
00187
00188
00189
00190
00191
00192
00193
00194
00195
00196
00197
00198
00199
00200
00201
00202
00203
00204
00205
00206
00207
00208
00209
00210
00211
00212
00213
00214
00215
00216
00217
00218
00219
00220
00221
00222
00223
00224
00225
00226
00227
00228
00229
00230
00231
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264
00265
00266
00267
00268
00269
00270
00271
00272
00273
00274
00275
00276
00277
00278
00279
00280
00281
00282
00283
00284
00285
00286
00287
00288
00289
00290
00291
00292
00293
00294
00295
00296
00297
00298
00299
00300
00301
00302
00303
00304
00305
00306
00307
00308
00309
00310
00311
00312
00313
00314
00315
00316
00317
00318
00319
00320
00321
00322
00323
00324
00325
00326
00327
00328
00329
00330
00331
00332
00333
00334
00335
00336
00337
00338
00339
00340
00341
00342
00343
00344
00345
00346
00347
00348
00349
00350
00351
00352
00353
00354
00355
00356
00357
00358
00359
00360
00361
00362
00363
00364
00365
00366
00367
00368
00369
00370
00371
00372
00373
00374
00375
00376
00377
00378
00379
00380
00381
00382
00383
00384
00385
00386
00387
00388
00389
00390
00391
00392
00393
00394
00395
00396
00397
00398
00399
00400
00401
00402
00403
00404
00405
00406
00407
00408
00409
00410
00411
00412
00413
00414
00415
00416
00417
00418
00419
00420
00421
00422
00423
00424
00425
00426
00427
00428
00429
00430
00431
00432
00433
00434
00435
00436
00437
00438
00439
00440
00441
00442
00443
00444
00445
00446
00447
00448
00449
00450
00451
00452
00453
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999
01000

```



```
00071 }; // end of VONAt
00072
00073
00074 #endif // !VONAT_H
```