LiteCal

Készítette Doxygen 1.9.4

1. Hierarchikus mutató

1.1. Osztályhierarchia

Majdnem (de nem teljesen) betűrendbe szedett leszármazási lista:

2. Osztálymutató

2.1. Osztálylista

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

3. Fájlmutató

3.1. Fájllista

Az összes dokumentált fájl listája rövid leírásokkal:

4. Osztályok dokumentációja

4.1. Date osztályreferencia

Dátumokat nyílvántartó osztály.

`#include <date.h>`

Publikus tagfüggvények

- **Date** ()

  Default konstruktor Alapértelmezett dátum 1970.01.01. (Unix time)
- Date (int year, int month, int day)

  Paraméteres konstruktor Érvényesség ellenőrzéssel.
- int getYear () const

  Getter függvény.
- int getMonth () const

  Getter függvény.
- int getDay () const

  Getter függvény.
- void setYear (int y)

  Setter függvény.
- void setMonth (int m)

  Setter függvény.
- void setDay (int d)

  Setter függvény.
- bool isValid () const

  Megállapítja egy dátumról, hogy érvényes-e.
- int dateInDays () const

  Egy dátum reprezentációja napokban.
- bool isLeapYear () const

  Megállapítja egy évről, hogy az szökőév-e.
- bool isLeapYear (int y) const

  Megállapítja egy évről, hogy az szökőév-e.
- int daysInMonth (int m) const

  Megállapítja, hány napos egy hónap.
- const char * getWeekDay () const

  Megállapítja, hogy egy dátum milyen napra esik. Ehhez a "Zeller's congruence" nevű algoritmust használtam. Forrás: https://www.geeksforgeeks.org/zellers-congruence-find-day-date/.
- size_t getWeekDayIdx () const

  Megállapítja, hogy egy dátum milyen napra esik. Ehhez a "Zeller's congruence" nevű algoritmust használtam. Forrás: https://www.geeksforgeeks.org/zellers-congruence-find-day-date/.
- bool operator> (const Date &rhs) const

  Két dátum összehasonlítása.
- bool operator< (const Date &rhs) const

  Két dátum összehasonlítása.
- bool operator== (const Date &rhs) const

  Két dátum összehasonlítása.
- bool operator>= (const Date &rhs) const

  Két dátum összehasonlítása.

- bool operator<= (const Date &rhs) const

  Két dátum összehasonlítása.
- Date operator+ (int rhs) const

  Milyen dátum lesz adott nap elteltével.
- int operator- (const Date &rhs) const

  Két dátum között eltelt napok száma.

4.1.1. Részletes leírás

Dátumokat nyílvántartó osztály.

4.1.2. Konstruktorok és destruktorok dokumentációja

4.1.2.1. Date()

```
Date::Date (
        int year,
        int month,
        int day )  [inline]
```

Paraméteres konstruktor Érvényesség ellenőrzéssel.

Paraméterek

year	év
month	hónap
day	nap

A függvény hívási gráfja:

4.1.3. Tagfüggvények dokumentációja

4.1.3.1. getYear()

```
int Date::getYear ( ) const  [inline]
```

Getter függvény.

Visszatérési érték

év

4.1.3.2. getMonth() `int Date::getMonth ( ) const [inline]`

Getter függvény.

Visszatérési érték

hónap

4.1.3.3. getDay() `int Date::getDay ( ) const [inline]`

Getter függvény.

Visszatérési érték

nap

4.1.3.4. setYear() `void Date::setYear ( int y ) [inline]`

Setter függvény.

Paraméterek

y	év

A függvény hívási gráfja:

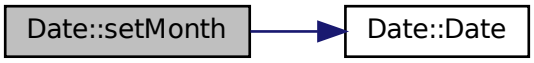
4.1.3.5. setMonth() `void Date::setMonth ( int m ) [inline]`

Setter függvény.

Paraméterek

m	hónap

A függvény hívási gráfja:



4.1.3.6. setDay() `void Date::setDay ( int d ) [inline]`

Setter függvény.

Paraméterek

d	nap

A függvény hívási gráfja:

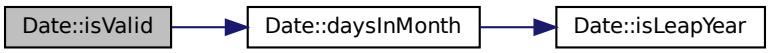
4.1.3.7. isValid() `bool Date::isValid ( ) const`

Megállapítja egy dátumról, hogy érvényes-e.

Visszatérési érték

igen/nem

A függvény hívási gráfja:



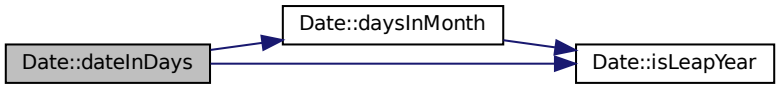
4.1.3.8. dateInDays() `int Date::dateInDays ( ) const`

Egy dátum reprezentációja napokban.

Visszatérési érték

1970.01.01. (unix epoch) óta eltelt napok száma

A függvény hívási gráfja:



4.1.3.9. isLeapYear() [1/2] `bool Date::isLeapYear ( ) const`

Megállapítja egy évről, hogy az szökőév-e.

Visszatérési érték

igen/nem

4.1.3.10. isLeapYear() [2/2] `bool Date::isLeapYear ( int y ) const`

Megállapítja egy évről, hogy az szökőév-e.

Paraméterek

y	év

Visszatérési érték

igen/nem

4.1.3.11. daysInMonth() `int Date::daysInMonth ( int m ) const`

Megállapítja, hány napos egy hónap.

Visszatérési érték

napok száma

A függvény hívási gráfja:

4.1.3.12. getWeekDay() `const char * Date::getWeekDay ( ) const`

Megállapítja, hogy egy dátum milyen napra esik. Ehhez a "Zeller's congruence" nevű algoritmust használtam. Forrás: https://www.geeksforgeeks.org/zellers-congruence-find-day-date/.

Visszatérési érték

A hét egy adott napja

4.1.3.13. getWeekDayIdx() `size_t Date::getWeekDayIdx ( ) const`

Megállapítja, hogy egy dátum milyen napra esik. Ehhez a "Zeller's congruence" nevű algoritmust használtam. Forrás: https://www.geeksforgeeks.org/zellers-congruence-find-day-date/.

Visszatérési érték

A hét egy adott napjának indexe (Hétfő = 0, Vasárnap = 6)

4.1.3.14. operator>() `bool Date::operator> ( const Date & rhs ) const`

Két dátum összehasonlítása.

Paraméterek

rhs	összehasonlítás jobboldala

Visszatérési érték

lhs frisebb dátum-e mint rhs

4.1.3.15. operator<() `bool Date::operator< ( const Date & rhs ) const`

Két dátum összehasonlítása.

Paraméterek

rhs	összehasonlítás jobboldala

Visszatérési érték

lhs régebbi dátum-e mint rhs

4.1.3.16. operator==() `bool Date::operator== ( const Date & rhs ) const`

Két dátum összehasonlítása.

Paraméterek

rhs	összehasonlítás jobboldala

Visszatérési érték

lhs és rhs megegyező dátum-e

4.1.3.17. operator>=() `bool Date::operator>= ( const Date & rhs ) const`

Két dátum összehasonlítása.

Paraméterek

rhs	összehasonlítás jobboldala

Visszatérési érték

lhs frisebb dátum-e mint rhs, vagy megegyeznek

4.1.3.18. operator<=() `bool Date::operator<= ( const Date & rhs ) const`

Két dátum összehasonlítása.

Paraméterek

rhs	összehasonlítás jobboldala

Visszatérési érték

lhs régebbi dátum-e mint rhs, vagy megegyeznek

4.1.3.19. operator+() `Date Date::operator+ ( int rhs ) const`

Milyen dátum lesz adott nap elteltével.

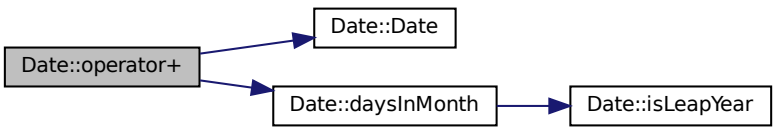
Paraméterek

rhs	+ hány nap

Visszatérési érték

Az új dátum

A függvény hívási gráfja:



4.1.3.20. operator-() `int Date::operator- ( const Date & rhs ) const`

Két dátum között eltelt napok száma.

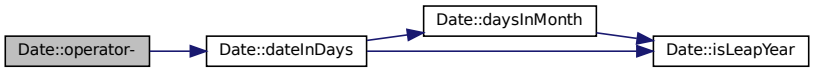
Paraméterek

rhs	másik dátum

Visszatérési érték

eltelt napok száma

A függvény hívási gráfja:



Ez a dokumentáció az osztályról a következő fájlok alapján készült:

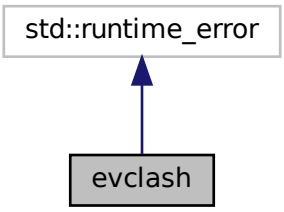
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/date.h
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/date.cpp

4.2. evclash osztályreferencia

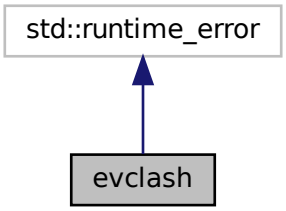
Saját kivételosztály arra, ha két esemény ütközik és ez kivételt jelent. Az std::runtime_error osztályból származik le. Más osztály nem származhat le belőle.

`#include <except.h>`

Az evclash osztály származási diagramja:



Az evclash osztály együttműködési diagramja:



4.2.1. Részletes leírás

Saját kivételosztály arra, ha két esemény ütközik és ez kivételt jelent. Az std::runtime_error osztályból származik le. Más osztály nem származhat le belőle.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/except.h

4.3. Event osztályreferencia

Eseményekhez dátumot, időt, és leírást párosító osztály.

`#include <event.h>`

Publikus tagfüggvények

- **Event** ()
  Default konstruktor.
- Event (const Date &evDate, const Time &evTime, const String &evDesc)
  Paraméteres konstruktor.
- Event (int y, int mo, int d, int h, int mi, const String &evDesc)
  Paraméteres konstruktor.
- Date getEvDate () const
  Getter függvény.
- Time getEvTime () const
  Getter függvény.
- String getEvDesc () const
  Getter függvény.
- void setEvDate (Date date)
  Setter függvény.
- void setEvTime (Time time)
  Setter függvény.

- void setEvDesc (String desc)

  Setter függvény.
- Event & operator= (const Event &rhs)

  Értékadó operátor.
- bool operator> (const Event &rhs) const

  Esemény összehasonlítása dátum alapján.
- bool operator< (const Event &rhs) const

  Esemény összehasonlítása dátum alapján.
- bool operator== (const Event &rhs) const

  Két esemény megegyezősége.

4.3.1. Részletes leírás

Eseményekhez dátumot, időt, és leírást párosító osztály.

4.3.2. Konstruktorok és destruktorok dokumentációja

4.3.2.1. Event() [1/2]

```
Event::Event (
        const Date & evDate,
        const Time & evTime,
        const String & evDesc )  [inline]
```

Paraméteres konstruktor.

Paraméterek

evDate	Dátum objektum
evTime	Idő objektum
evDesc	Leírás sztring objektum

4.3.2.2. Event() [2/2]

```
Event::Event (
        int y,
        int mo,
        int d,
        int h,
        int mi,
        const String & evDesc )  [inline]
```

Paraméteres konstruktor.

Paraméterek

y	Év
mo	Hónap

Paraméterek

d	Nap
h	Óra
mi	Perc
evDesc	Leírás sztring objektum

4.3.3. Tagfüggvények dokumentációja

4.3.3.1. getEvDate() `Date Event::getEvDate ( ) const [inline]`

Getter függvény.

Visszatérési érték

Dátum objektum

4.3.3.2. getEvTime() `Time Event::getEvTime ( ) const [inline]`

Getter függvény.

Visszatérési érték

Idő objektum

4.3.3.3. getEvDesc() `String Event::getEvDesc ( ) const [inline]`

Getter függvény.

Visszatérési érték

String objektum

4.3.3.4. setEvDate() `void Event::setEvDate ( Date date ) [inline]`

Setter függvény.

Paraméterek

date	dátum

4.3.3.5. setEvTime() `void Event::setEvTime ( Time time ) [inline]`

Setter függvény.

Paraméterek

time	időpont

4.3.3.6. setEvDesc() `void Event::setEvDesc ( String desc ) [inline]`

Setter függvény.

Paraméterek

desc	leírás

4.3.3.7. operator=() `Event & Event::operator= ( const Event & rhs )`

Értékadó operátor.

Paraméterek

rhs	Új érték

Visszatérési érték

Megváltozott értékű objektum

4.3.3.8. operator>() `bool Event::operator> ( const Event & rhs ) const`

Esemény összehasonlítása dátum alapján.

Paraméterek

rhs	Összehasonlítás jobb oldala

Visszatérési érték

lhs frisebb dátumú-e mint rhs

4.3.3.9. operator<() `bool Event::operator< ( const Event & rhs ) const`

Esemény összehasonlítása dátum alapján.

Paraméterek

rhs	Összehasonlítás jobb oldala

Visszatérési érték

lhs régebbi dátumú-e mint rhs

4.3.3.10. operator==() `bool Event::operator== ( const Event & rhs ) const`

Két esemény megegyezősége.

Paraméterek

rhs	Összehasonlítás jobb oldala

Visszatérési érték

lhs és rhs megegyező dátumúak-e

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

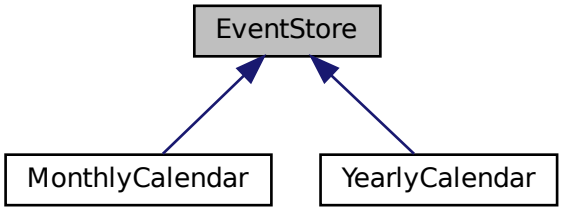
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/event.h
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/event.cpp

4.4. EventStore osztályreferencia

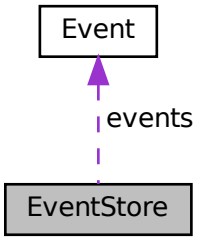
A többi (éves, havi) naptár alaposztálya.

`#include <calendar.h>`

Az EventStore osztály származási diagramja:



Az EventStore osztály együttműködési diagramja:



Publikus tagfüggvények

- **EventStore** ()
  Default konstruktor.
- EventStore (const Event &event)
  Paraméteres konstruktor.
- EventStore (const EventStore &rhs)
  Másoló konstruktor.
- EventStore & operator= (const EventStore &rhs)
  Értékadó operátor.
- virtual ~**EventStore** ()
  Destruktor (virtuális)
- Event * getEvents () const
  Getter függvény
- size_t getNEvents () const
  Getter függvény.

- Event & operator[ ] (int i)

  Indexelő operátor Módosíthatja a hívó objektumot. Végez indexellenőrzést, kivétellel tér vissza érvénytelen indexelés esetén.
- Event operator[ ] (int i) const

  Indexelő operátor NEM módosíthatja a hívó objektumot. Végez indexellenőrzést, kivétellel tér vissza érvénytelen indexelés esetén.
- Event & find (const Event &searchEv)

  Kereső függvény az eseménytárolóban.
- const Event & find (const Event &searchEv) const

  Kereső függvény az eseménytárolóban.
- void eventClash (const Event &checked) const

  Kivételt dob (except::evclash) ha a listában van a vizsgálttal ütköző esemény.
- EventStore & operator+ (const Event &rhs)

  Az események listájához hozzáfűz egy új eseményt.
- EventStore & operator- (const Event &rhs)

  Az események listájából töröl egy specifikus eseményt.
- void **sort** ()

  Az események listáját dátum szerint csökkenő sorrendbe rendezi. Ehhez az std::sort() STL függvényt használja.
- EventStore filterBy (int year=NOPARAM, int month=NOPARAM, int day=NOPARAM)

  Szűrés egy adott paraméter szerint az eseménytárolóban.
- const EventStore filterBy (int year=NOPARAM, int month=NOPARAM, int day=NOPARAM) const

  Szűrés egy adott paraméter szerint az eseménytárolóban.
- Event * begin ()

  Iterátor kezdete.
- const Event * begin () const
- Event * end ()

  Iterátor vége.
- const Event * end () const

  Iterátor vége.

Védett attribútumok

- size_t **nEvents**

  események száma
- Event * **events**

  események dinamikus tömbje

4.4.1. Részletes leírás

A többi (éves, havi) naptár alaposztálya.

4.4.2. Konstruktorok és destruktorok dokumentációja

4.4.2.1. EventStore() [1/2] EventStore::EventStore (const Event & *event*) [inline]

Paraméteres konstruktor.

Paraméterek

event	Egy esemény

4.4.2.2. EventStore() [2/2] EventStore::EventStore (const EventStore & *rhs*)

Másoló konstruktor.

Paraméterek

rhs	Másolt objektum

4.4.3. Tagfüggvények dokumentációja

4.4.3.1. operator=() EventStore & EventStore::operator= (const EventStore & *rhs*)

Értékadó operátor.

Paraméterek

rhs	Átadott érték

4.4.3.2. getEvents() Event * EventStore::getEvents ( ) const [inline]

Getter függvény

Visszatérési érték

Tároló

4.4.3.3. getNEvents() size_t EventStore::getNEvents ( ) const [inline]

Getter függvény.

Visszatérési érték

Tárolt események száma

4.4.3.4. operator[]() [1/2] Event & EventStore::operator[] (int *i*)

Indexelő operátor Módosíthatja a hívó objektumot. Végez indexellenőrzést, kivétellel tér vissza érvénytelen indexelés esetén.

Paraméterek

i	index

Visszatérési érték

indexelt elem referenciája

4.4.3.5. operator[]() [2/2] Event EventStore::operator[] (int *i*) const

Indexelő operátor NEM módosíthatja a hívó objektumot. Végez indexellenőrzést, kivétellel tér vissza érvénytelen indexelés esetén.

Paraméterek

i	index

Visszatérési érték

indexelt elem értéke

4.4.3.6. find() [1/2] Event & EventStore::find (const Event & *searchEv*)

Kereső függvény az eseménytárolóban.

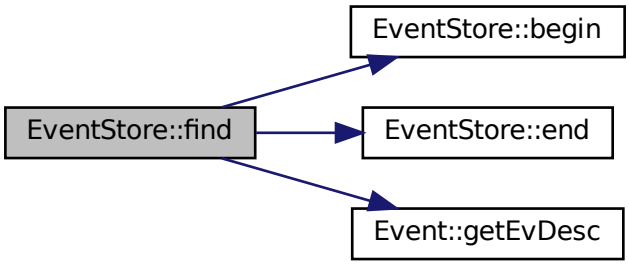
Paraméterek

searchEv	Az esemény ami a keresés kritériuma

Visszatérési érték

Az esemény referenciája (ha nincs találat, akkor except::nofind kivételt dob)

A függvény hívási gráfja:



4.4.3.7. find() [2/2] const Event & EventStore::find (const Event & *searchEv*) const

Kereső függvény az eseménytárolóban.

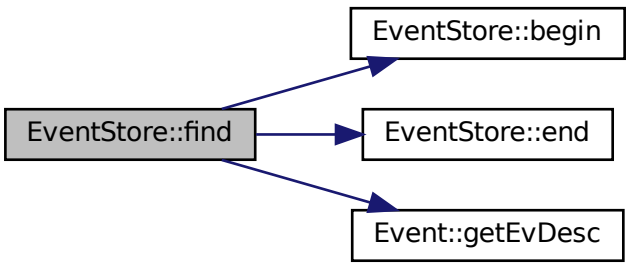
Paraméterek

searchEv	Az esemény ami a keresés kritériuma

Visszatérési érték

Az esemény referenciája (konstans) (ha nincs találat, akkor except::nofind kivételt dob)

A függvény hívási gráfja:



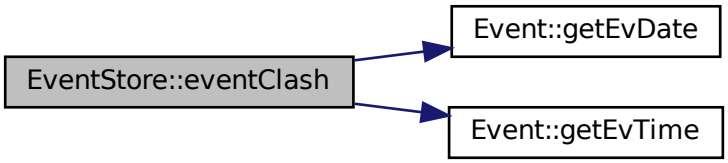
4.4.3.8. eventClash() `void EventStore::eventClash ( const Event & checked ) const`

Kivételt dob (except::evclash) ha a listában van a vizsgálttal ütköző esemény.

Paraméterek

checked	Vizsgált esemény

A függvény hívási gráfja:



4.4.3.9. operator+() `EventStore & EventStore::operator+ ( const Event & rhs )`

Az események listájához hozzáfűz egy új eseményt.

Paraméterek

rhs	A hozzáfűzött esemény

Visszatérési érték

∗this pointer

A függvény hívási gráfja:

4.4.3.10. operator-() `EventStore & EventStore::operator- ( const Event & rhs )`

Az események listájából töröl egy specifikus eseményt.

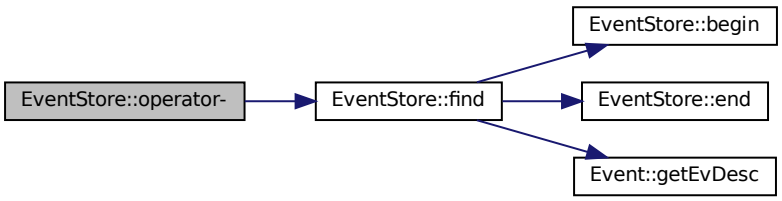
Paraméterek

rhs	A törlendő esemény

Visszatérési érték

∗this pointer

A függvény hívási gráfja:



4.4.3.11. filterBy() `[1/2]` `EventStore EventStore::filterBy ( int year = NOPARAM, int month = NOPARAM, int day = NOPARAM )`

Szűrés egy adott paraméter szerint az eseménytárolóban.

Egy paraméter explicit elhagyásához a "NOPARAM" makró használható.

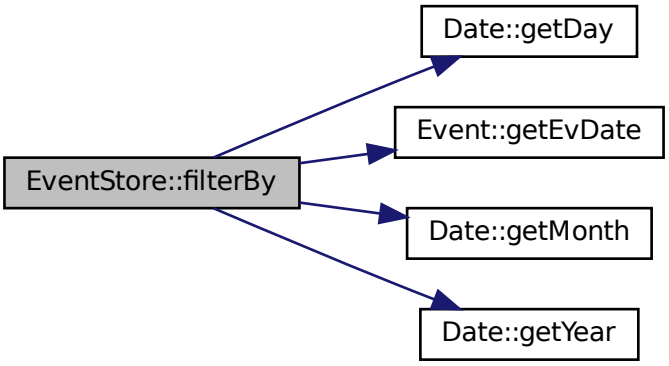
Paraméterek

year	év paraméter (elhagyható)
month	hónap paraméter (elhagyható)
day	nap paraméter (elhagyható)

Visszatérési érték

Új eseménytároló

A függvény hívási gráfja:



4.4.3.12. filterBy() [2/2]

```
const EventStore EventStore::filterBy (
        int year = NOPARAM,
        int month = NOPARAM,
        int day = NOPARAM ) const
```

Szűrés egy adott paraméter szerint az eseménytárolóban.

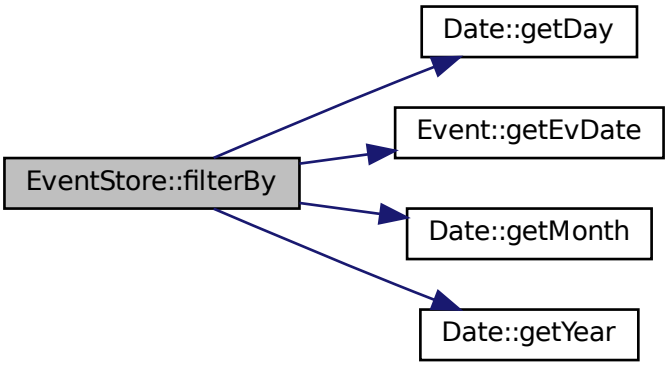
Paraméterek

year	év paraméter (elhagyható)
month	hónap paraméter (elhagyható)
day	nap paraméter (elhagyható)

Visszatérési érték

Új eseménytároló (konstans)

A függvény hívási gráfja:



4.4.3.13. begin() [1/2] Event * EventStore::begin ( ) [inline]

Iterátor kezdete.

Visszatérési érték

Első esemény pointer

4.4.3.14. begin() [2/2] const Event * EventStore::begin ( ) const [inline]

Visszatérési érték

Első esemény pointer

4.4.3.15. end() [1/2] Event * EventStore::end ( ) [inline]

Iterátor vége.

Utolsó esemény utáni pointer

4.4.3.16. end() [2/2] const Event * EventStore::end ( ) const [inline]

Iterátor vége.

Utolsó esemény utáni pointer

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

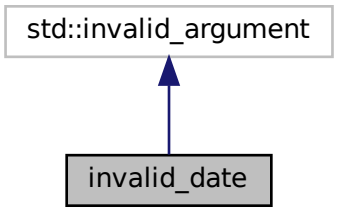
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/calendar.h
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/calendar.cpp

4.5. invalid_date osztályreferencia

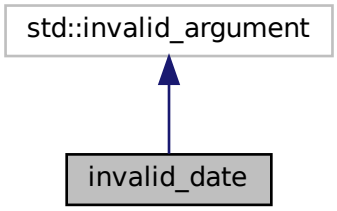
Saját kivételosztály az érvénytelen dátumokra. Az std::invalid_argument osztályból származik le. Más osztály nem származhat le belőle.

`#include <except.h>`

Az invalid_date osztály származási diagramja:



Az invalid_date osztály együttműködési diagramja:



4.5.1. Részletes leírás

Saját kivételosztály az érvénytelen dátumokra. Az std::invalid_argument osztályból származik le. Más osztály nem származhat le belőle.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

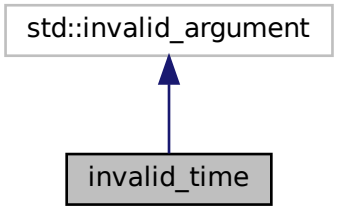
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/except.h

4.6. invalid_time osztályreferencia

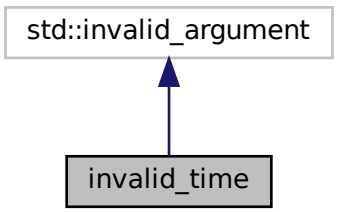
Saját kivételosztály az érvénytelen időpontokra. Az std::invalid_argument osztályból származik le. Más osztály nem származhat le belőle.

`#include <except.h>`

Az invalid_time osztály származási diagramja:



Az invalid_time osztály együttműködési diagramja:



4.6.1. Részletes leírás

Saját kivételosztály az érvénytelen időpontokra. Az std::invalid_argument osztályból származik le. Más osztály nem származhat le belőle.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

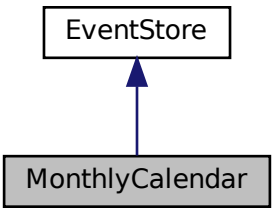
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/except.h

4.7. MonthlyCalendar osztályreferencia

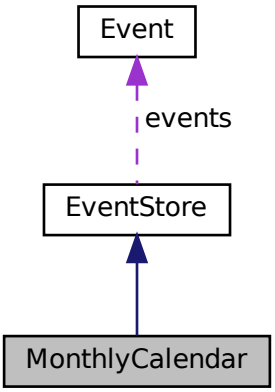
Egy származtatott adapterosztálya az eseménytárolónak.

`#include <calendar.h>`

A MonthlyCalendar osztály származási diagramja:



A MonthlyCalendar osztály együttműködési diagramja:



Publikus tagfüggvények

- **MonthlyCalendar** ()

  Default konstruktor.
- MonthlyCalendar (const EventStore &src, int selYear=1970, int selMonth=1)

  Paraméteres konstruktor.
- void printCalendar (std::ostream &os=std::cout) const

  Kiírja a naptárat havi nézetben.

További örökölt tagok

4.7.1. Részletes leírás

Egy származtatott adapterosztálya az eseménytárolónak.

4.7.2. Konstruktorok és destruktorok dokumentációja

4.7.2.1. MonthlyCalendar() `MonthlyCalendar::MonthlyCalendar (`

```
const EventStore & src,
int selYear = 1970,
int selMonth = 1 )  [inline]
```

Paraméteres konstruktor.

Paraméterek

src	forrástároló
selYear	kiválasztott év (default = 1970)
selMonth	kiválasztott hónap (default = 1)

4.7.3. Tagfüggvények dokumentációja

4.7.3.1. printCalendar() `void MonthlyCalendar::printCalendar (`

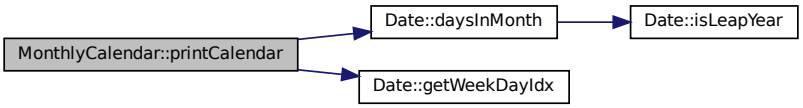
```
std::ostream & os = std::cout ) const
```

Kiírja a naptárat havi nézetben.

Paraméterek

os	output stream

A függvény hívási gráfja:



Ez a dokumentáció az osztályról a következő fájlok alapján készült:

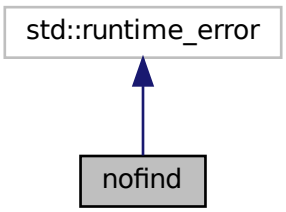
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/calendar.h
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/calendar.cpp

4.8. nofind osztályreferencia

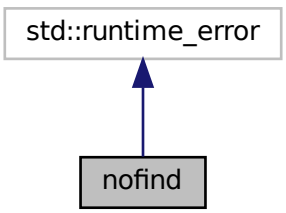
Saját kivételosztály arra, ha a keresés nem talált értéket. Az std::runtime_error osztályból származik le. Más osztály nem származhat le belőle.

`#include <except.h>`

A nofind osztály származási diagramja:



A nofind osztály együttműködési diagramja:



4.8.1. Részletes leírás

Saját kivételosztály arra, ha a keresés nem talált értéket. Az std::runtime_error osztályból származik le. Más osztály nem származhat le belőle.

Ez a dokumentáció az osztályról a következő fájl alapján készült:

- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/except.h

4.9. String osztályreferencia

Egy dinamikus sztringkezelő osztály RAII elven működő, automatikusan memóriafoglalást és felszabadítást megvalósító osztály, felüldefiniált operátorokkal.

`#include <string.h>`

Publikus tagfüggvények

- **String** ()
  Paraméter nélküli konstruktor Egy üres sztringet reprezentál.
- String (char c)
  Paraméteres konstruktor Karakterből String objektumot készít.
- String (const char ∗str)
  Paraméteres konstruktor C-stringből String objektumot készít.
- String (const String &rhs)
  Másoló konstruktor.
- ∼**String** ()
  Destruktor.
- size_t getLen () const
  Getter fügvény.
- char const ∗ c_str () const
  Getter függvény.
- String & operator= (const String &rhs)
  Értékadó operátor Fontos értékadó operátort definiálni a helyes memóriakezelés érdekében.
- String operator+ (const String &rhs) const
  Sztring hozzáfűzés Az "strcat" C függvényt alkalmazza.
- String operator+ (char c) const
  Sztring hozzáfűzés Az "strcat" C függvényt alkalmazza.
- String & operator+= (const String &rhs)
  Értékadó hozzáfűzés.
- String & operator+= (char c)
  Értékadó hozzáfűzés.
- char & operator[ ] (int i)
  Indexelő operátor Módosíthatja a hívó objektumot. Végez indexellenőrzést, kivétellel tér vissza érvénytelen indexelés esetén.
- char operator[ ] (int i) const
  Indexelő operátor NEM módosíthatja a hívó objektumot. Végez indexellenőrzést, kivétellel tér vissza érvénytelen indexelés esetén.
- bool operator> (const String &rhs) const
  ABC sorrend szerinti összehasonlítása két sztringnek.
- bool operator< (const String &rhs) const
  ABC sorrend szerinti összehasonlítása két sztringnek.
- bool operator== (const String &rhs) const
  Két sztring megegyező-e.
- bool operator!= (const String &rhs) const
  Két sztring nem megegyező-e.

4.9.1. Részletes leírás

Egy dinamikus sztringkezelő osztály RAII elven működő, automatikusan memóriafoglalást és felszabadítást megvalósító osztály, felüldefiniált operátorokkal.

4.9.2. Konstruktorok és destruktorok dokumentációja

4.9.2.1. String() [1/3] String::String (char c) [inline]

Paraméteres konstruktor Karakterből String objektumot készít.

Paraméterek

c	Karakter

4.9.2.2. String() [2/3] String::String (const char * str)

Paraméteres konstruktor C-stringből String objektumot készít.

Paraméterek

str	C-string ('\0'-val lezárt)

4.9.2.3. String() [3/3] String::String (const String & rhs)

Másoló konstruktor.

Paraméterek

rhs	A másolandó String objektum

4.9.3. Tagfüggvények dokumentációja

4.9.3.1. getLen() `size_t String::getLen ( ) const [inline]`

Getter fügvény.

Visszatérési érték

String objektum hossza

4.9.3.2. c_str() `char const * String::c_str ( ) const [inline]`

Getter függvény.

Visszatérési érték

C-sztring (read only)

4.9.3.3. operator=() `String & String::operator= ( const String & rhs )`

Értékadó operátor Fontos értékadó operátort definiálni a helyes memóriakezelés érdekében.

Paraméterek

rhs	Értékként adott String objektum

Visszatérési érték

∗this pointer

4.9.3.4. operator+() [1/2] `String String::operator+ ( const String & rhs ) const`

Sztring hozzáfűzés Az "strcat" C függvényt alkalmazza.

Paraméterek

rhs	Jobbról hozzáfűzött String

Visszatérési érték

Új String objektum

4.9.3.5. operator+() [2/2] String String::operator+ (char c) const

Sztring hozzáfűzés Az "strcat" C függvényt alkalmazza.

Paraméterek

c	Jobbról hozzáfűzött karakter

Visszatérési érték

Új String objektum

4.9.3.6. operator+=() [1/2] String & String::operator+= (const String & rhs)

Értékadó hozzáfűzés.

Paraméterek

rhs	A hozzáfűzendő String

Visszatérési érték

Hozzáfűzött String

4.9.3.7. operator+=() [2/2] String & String::operator+= (char c)

Értékadó hozzáfűzés.

Paraméterek

c	A hozzáfűzendő karakter

Visszatérési érték

Hozzáfűzött String

4.9.3.8. operator[]() [1/2] char & String::operator[] (int i)

Indexelő operátor Módosíthatja a hívó objektumot. Végez indexellenőrzést, kivétellel tér vissza érvénytelen indexelés esetén.

Paraméterek

i	index

Visszatérési érték

indexelt elem referenciája

4.9.3.9. operator[]() [2/2] char String::operator[] (int *i*) const

Indexelő operátor NEM módosíthatja a hívó objektumot. Végez indexellenőrzést, kivétellel tér vissza érvénytelen indexelés esetén.

Paraméterek

i	index

Visszatérési érték

indexelt elem értéke

4.9.3.10. operator>() bool String::operator> (const String & *rhs*) const

ABC sorrend szerinti összehasonlítása két sztringnek.

Paraméterek

rhs	A hívóval összehasonlítandó String

Visszatérési érték

lhs előrébb van-e az ABC-ben, mint rhs

4.9.3.11. operator<() bool String::operator< (const String & *rhs*) const

ABC sorrend szerinti összehasonlítása két sztringnek.

Paraméterek

rhs	A hívóval összehasonlítandó String

Visszatérési érték

lhs hátrébb van-e az ABC-ben, mint rhs

4.9.3.12. operator==() `bool String::operator== ( const String & rhs ) const`

Két sztring megegyező-e.

Paraméterek

rhs	A hívóval összehasonlítandó String

Visszatérési érték

lhs ugyanaz-e, mint rhs

4.9.3.13. operator"!=() `bool String::operator!= ( const String & rhs ) const`

Két sztring nem megegyező-e.

Paraméterek

rhs	A hívóval összehasonlítandó String

Visszatérési érték

lhs nem ugyanaz-e, mint rhs

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/string.h
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/string.cpp

4.10. Time osztályreferencia

Időpontokat nyílvántartó osztály.

`#include <time.h>`

Publikus tagfüggvények

- **Time** ()

  Default konstruktor Alapértelmezett időpont: 00:00 (éjfél)
- Time (int hour, int minute)

  Paraméteres konstruktor Érvényesség ellenőrzéssel.
- bool isValid () const

  Megállapítja egy időpontról, hogy érvényes-e.
- int getHour () const

  Getter függvény.
- int getMinute () const

  Getter függvény.
- void setHour (int h)

  Setter függvény.
- void setMinute (int m)

  Setter függvény.
- bool operator> (const Time &rhs) const

  Két időpont összehasonlítása.
- bool operator< (const Time &rhs) const

  Két időpont összehasonlítása.
- bool operator== (const Time &rhs) const

  Két időpont összehasonlítása.
- bool operator>= (const Time &rhs) const

  Két időpont összehasonlítása.
- bool operator<= (const Time &rhs) const

  Két időpont összehasonlítása.

4.10.1. Részletes leírás

Időpontokat nyílvántartó osztály.

4.10.2. Konstruktorok és destruktorok dokumentációja

4.10.2.1. Time()

```
Time::Time (
        int hour,
        int minute )  [inline]
```

Paraméteres konstruktor Érvényesség ellenőrzéssel.

Paraméterek

hour	óra
minute	perc

A függvény hívási gráfja:

4.10.3. Tagfüggvények dokumentációja

4.10.3.1. isValid() `bool Time::isValid ( ) const`

Megállapítja egy időpontról, hogy érvényes-e.

Visszatérési érték

igen/nem

4.10.3.2. getHour() `int Time::getHour ( ) const [inline]`

Getter függvény.

Visszatérési érték

óra

4.10.3.3. getMinute() `int Time::getMinute ( ) const [inline]`

Getter függvény.

Visszatérési érték

perc

4.10.3.4. setHour() `void Time::setHour ( int h ) [inline]`

Setter függvény.

Paraméterek

h	óra

A függvény hívási gráfja:

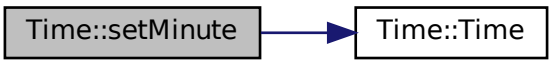
4.10.3.5. setMinute() `void Time::setMinute ( int m ) [inline]`

Setter függvény.

Paraméterek

m	perc

A függvény hívási gráfja:



4.10.3.6. operator>() `bool Time::operator> ( const Time & rhs ) const`

Két időpont összehasonlítása.

Paraméterek

rhs	összehasonlítás jobboldala

Visszatérési érték

lhs frisebb időpont-e mint rhs

4.10.3.7. operator<() `bool Time::operator< ( const Time & rhs ) const`

Két időpont összehasonlítása.

Paraméterek

rhs	összehasonlítás jobboldala

Visszatérési érték

lhs régebbi időpont-e mint rhs

4.10.3.8. operator==() `bool Time::operator== ( const Time & rhs ) const`

Két időpont összehasonlítása.

Paraméterek

rhs	összehasonlítás jobboldala

Visszatérési érték

lhs és rhs megegyező időpont-e

4.10.3.9. operator>=() `bool Time::operator>= ( const Time & rhs ) const`

Két időpont összehasonlítása.

Paraméterek

rhs	összehasonlítás jobboldala

Visszatérési érték

lhs frisebb időpont-e mint rhs, vagy megegyeznek

4.10.3.10. operator<=() `bool Time::operator<= ( const Time & rhs ) const`

Két időpont összehasonlítása.

Paraméterek

rhs	összehasonlítás jobboldala

Visszatérési érték

lhs régebbi időpont-e mint rhs, vagy megegyeznek

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

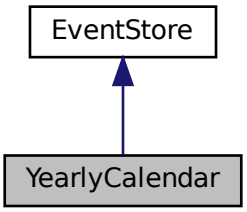
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/time.h
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/time.cpp

4.11. YearlyCalendar osztályreferencia

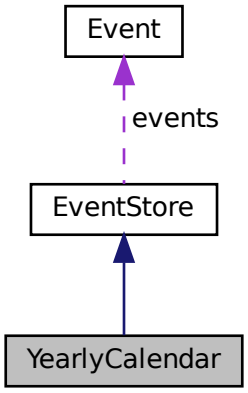
Egy származtatott adapterosztálya az eseménytárolónak.

`#include <calendar.h>`

Az YearlyCalendar osztály származási diagramja:



Az YearlyCalendar osztály együttműködési diagramja:



Publikus tagfüggvények

- **YearlyCalendar** ()

  Default konstruktor.
- YearlyCalendar (const EventStore &src, int selYear=1970)

  Paraméteres konstruktor.
- void printCalendar (std::ostream &os=std::cout) const

  Kiírja a naptárat éves nézetben.

További örökölt tagok

4.11.1. Részletes leírás

Egy származtatott adapterosztálya az eseménytárolónak.

4.11.2. Konstruktorok és destruktorok dokumentációja

4.11.2.1. YearlyCalendar() `YearlyCalendar::YearlyCalendar ( const EventStore & src, int selYear = 1970 ) [inline]`

Paraméteres konstruktor.

Paraméterek

src	forrástároló
selYear	kiválasztott év (default = 1970)

4.11.3. Tagfüggvények dokumentációja

4.11.3.1. printCalendar() `void YearlyCalendar::printCalendar ( std::ostream & os = std::cout ) const`

Kiírja a naptárat éves nézetben.

Paraméterek

os	output stream

A függvény hívási gráfja:

Ez a dokumentáció az osztályról a következő fájlok alapján készült:

- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/calendar.h
- /home/tomin/Documents/PROG2/NHF/Calendar_NHF/calendar.cpp

5. Fájlok dokumentációja

5.1. calendar.h

```
1 // calendar.h naptárkezelő osztály (deklarációk, inlineok) - SAXHSH
2
3 #ifndef CALENDAR_H
4 #define CALENDAR_H
5
6 #include "event.h"
7 #include "string.h"
8 #include "date.h"
9 #include "time.h"
10
11 // Látszatos makró a szűrés tetszőleges elhagyható paramétereire
12 #define NOPARAM -1
13
16 class EventStore {
17 protected:
18     size_t nEvents;
19     Event *events;
20 public:
22     EventStore() :nEvents(0), events(new Event[0]) {}
23
26     EventStore(const Event& event) :nEvents(1), events(new Event[1]) { events[0] = event; }
27
30     EventStore(const EventStore& rhs);
31
34     EventStore& operator=(const EventStore& rhs);
35
37     virtual ~EventStore() { delete[] events; }
38
41     Event* getEvents()const { return events; }
```

```
42
45      size_t getNEvents()const { return nEvents; };
46
51      Event& operator[](int i);
52
57      Event operator[](int i) const;
58
62      Event& find(const Event& searchEv);
63
67      const Event& find(const Event& searchEv) const;
68
71      void eventClash(const Event& checked) const;
72
76      EventStore& operator+(const Event& rhs);
77
81      EventStore& operator-(const Event& rhs);
82
85      void sort();
86
93      EventStore filterBy(int year = NOPARAM, int month = NOPARAM, int day = NOPARAM);
94
100      const EventStore filterBy(int year = NOPARAM, int month = NOPARAM, int day = NOPARAM) const;
101
104      Event *begin() { return events; }
105
106      // @brief Iterátor kezdete (konstans)
108      const Event *begin()const { return events; }
109
112      Event *end() { return events + nEvents; }
113
116      const Event *end()const { return events + nEvents; }
117 };
118
123 std::ostream& operator«(std::ostream& os, const EventStore &rhs);
124
127 class YearlyCalendar :public EventStore {
128 private:
129     int selYear;
130 public:
132     YearlyCalendar():  selYear(1970) {}
133
137     YearlyCalendar(const EventStore& src, int selYear = 1970) :EventStore(src.filterBy(selYear)),
      selYear(selYear) {}
138
141     void printCalendar(std::ostream& os = std::cout) const;
142 };
143
146 class MonthlyCalendar :public EventStore {
147 private:
148     int selYear;
149     int selMonth;
150 public:
152     MonthlyCalendar():  selYear(1970), selMonth(1) {}
153
158     MonthlyCalendar(const EventStore& src, int selYear = 1970, int selMonth = 1)
159                     :EventStore(src.filterBy(selYear, selMonth)), selYear(selYear), selMonth(selMonth)
     {}
160
163     void printCalendar(std::ostream& os = std::cout) const;
164 };
165
166 #endif
```

5.2. date.h

```
1 // date.h dátumkezelő osztály (deklarációk/inlineok) - SAXHSH
2
3 #ifndef DATE_H
4 #define DATE_H
5
6 #include <iostream>
7 #include "except.h"
8
11 class Date {
12 private:
13     int year;
14     int month;
15     int day;
16     static const char* weekDays[7];
17 public:
20     Date() :year(1970), month(1), day(1) {}
21
27     Date(int year, int month, int day) :year(year), month(month), day(day) {
28         if (!isValid()) throw invalid_date("Érvénytelen dátum!");
```

```
29      }
30
33      int getYear()const { return year; }
34
37      int getMonth()const { return month; }
38
41      int getDay()const { return day; }
42
43      /*Megjegyzés:  a settereknél a konstruktort újrafuttatom, hogy megmaradjon a hibaellenőrzés az új
        dátumokra.*/
44
47      void setYear(int y) { *this = Date(y, month, day); }
48
51      void setMonth(int m) { *this = Date(year, m, day); }
52
55      void setDay(int d) { *this = Date(year, month, d); }
56
59      bool isValid() const;
60
63      int dateInDays() const;
64
67      bool isLeapYear() const;
68
72      bool isLeapYear(int y) const;
73
76      int daysInMonth(int m) const;
77
82      const char* getWeekDay() const;
83
88      size_t getWeekDayIdx() const;
89
93      bool operator>(const Date& rhs) const;
94
98      bool operator<(const Date& rhs) const;
99
103      bool operator==(const Date& rhs) const;
104
108      bool operator>=(const Date& rhs) const;
109
113      bool operator<=(const Date& rhs) const;
114
118      Date operator+(int rhs) const;
119
123      int operator-(const Date& rhs) const;
124 };
125
128 std::ostream& operator«(std::ostream& os, const Date& rhs);
129
132 std::istream& operator»(std::istream& is, Date& rhs);
133
134 #endif
```

5.3. event.h

```
1  // event.h eseménykezelő osztály (deklarációk/inlineok) - SAXHSH
2
3  #ifndef EVENT_H
4  #define EVENT_H
5
6  #include "string.h"
7  #include "date.h"
8  #include "time.h"
9
12 class Event {
13 private:
14      Date evDate;
15      Time evTime;
16      String evDesc;
17 public:
19      Event() :evDate(), evTime(), evDesc() {}
20
25      Event(const Date& evDate, const Time& evTime, const String& evDesc) :evDate(evDate), evTime(evTime),
        evDesc(evDesc) {}
26
34      Event(int y, int mo, int d, int h, int mi, const String& evDesc) :evDate(y, mo, d), evTime(h, mi),
        evDesc(evDesc) {}
35
38      Date getEvDate()const { return evDate; }
39
42      Time getEvTime()const { return evTime; }
43
46      String getEvDesc()const { return evDesc; }
47
50      void setEvDate(Date date) { evDate = date; }
```

```
51
54      void setEvTime(Time time) { evTime = time; }
55
58      void setEvDesc(String desc) { evDesc = desc; }
59
63      Event& operator=(const Event& rhs);
64
68      bool operator>(const Event& rhs) const;
69
73      bool operator<(const Event& rhs) const;
74
78      bool operator==(const Event& rhs) const;
79  };
80
83  std::ostream& operator«(std::ostream& os, const Event& rhs);
84
87  std::istream& operator»(std::istream& is, Event& rhs);
88
89  #endif
```

5.4. except.h

```
1  // except.h kivételkezelő osztályok - SAXHSH
2
3  #ifndef EXCEPT_H
4  #define EXCEPT_H
5
6  #include <stdexcept>
7
11 struct invalid_date final :  public std::invalid_argument {
12      using std::invalid_argument::invalid_argument;
13 };
14
18 struct invalid_time final :  public std::invalid_argument {
19      using std::invalid_argument::invalid_argument;
20 };
21
25 struct nofind final :  public std::runtime_error {
26      using std::runtime_error::runtime_error;
27 };
28
32 struct evclash final :  public std::runtime_error {
33      using std::runtime_error::runtime_error;
34 };
35
36 #endif
```

5.5. menu.h

```
1  // menu.h - menükezelő függvényeinek deklarációi - SAXHSH
2
3  #include "date.h"
4  #include "time.h"
5  #include "event.h"
6  #include "calendar.h"
7
8  #ifndef MENU_H
9  #define MENU_H
10
12 enum class MenuState {
13      MAIN_MENU,
14      MANAGE_MENU,
15      CAL_OPS_MENU,
16      MONTHLY,
17      YEARLY,
18      FILTER_MENU,
19      EXIT
20 };
21
23 enum class FilterOpts {
24      BYDATE,
25      BYYEARANDMONTH,
26      BYMONTHANDDAY,
27      BYYEAR,
28      BYMONTH,
29      BYDAY
30 };
31
33 enum class CalOpOpts {
34      WHATDAY,
35      NUMBETWEEN,
```

```
36     NUMTILL,
37     DAYINX
38 };
39
41 void waitForReturn();
42
44 void clearScreen();
45
47 void showMainMenu();
48
50 void showManageMenu();
51
53 void showCalOpsMenu();
54
56 void showFilterMenu();
57
60 void showMonthly(const EventStore& mainStore);
61
64 void showYearly(const EventStore& es);
65
68 void handleAddEvent(EventStore& mainStore);
69
72 void handleDelEvent(EventStore& mainStore);
73
76 void handleSearch(const EventStore& mainStore);
77
81 void handleCalOps(EventStore& mainStore, CalOpOpts option);
82
86 void handleFilter(EventStore& mainStore, FilterOpts option);
87
88 #endif
```

5.6. string.h

```
1 // string.h dinamikus sztringkezelő osztály (deklarációk/inlineok) - SAXHSH
2
3 #ifndef STRING_H
4 #define STRING_H
5
6 #include <cstddef>
7 #include <iostream>
8
12 class String {
13 private:
14     size_t len;
15     char *pData;
16 public:
19     String() :  len(0), pData(new char[1]{'\0'}) {}
20
24     String(char c) :len(1), pData(new char[2]{c, '\0'}) {}
25
29     String(const char* str);
30
33     String(const String& rhs);
34
36     ~String() { delete[] pData; }
37
40     size_t getLen()const { return len; }
41
44     char const* c_str()const { return pData; }
45
50     String& operator=(const String& rhs);
51
56     String operator+(const String& rhs) const;
57
62     String operator+(char c) const;
63
67     String& operator+=(const String& rhs);
68
72     String& operator+=(char c);
73
78     char& operator[](int i);
79
84     char operator[](int i) const;
85
89     bool operator>(const String& rhs) const;
90
94     bool operator<(const String& rhs) const;
95
99     bool operator==(const String& rhs) const;
100
104     bool operator!=(const String& rhs) const;
105 };
106
```

```
109 std::ostream& operator«(std::ostream& os, const String& rhs);
110
113 std::istream& operator»(std::istream& is, String& rhs);
114
115 #endif
```

5.7. time.h

```
1 // time.h időkezelő osztály (deklarációk/inlineok) - SAXHSH
2
3 #ifndef TIME_H
4 #define TIME_H
5
6 #include <iostream>
7 #include "except.h"
8
11 class Time {
12 private:
13     int hour;
14     int minute;
15 public:
18     Time() :hour(0), minute(0) {}
19
24     Time(int hour, int minute) :hour(hour), minute(minute) {
25         if (!isValid()) throw invalid_time("Érvénytelen idő!");
26     }
27
30     bool isValid() const;
31
34     int getHour()const { return hour; }
35
38     int getMinute()const { return minute; }
39
40     /*Megjegyzés:  a settereknél a konstruktort újrafuttatom, hogy megmaradjon a hibaellenőrzés az új
      időpontokra.*/
41
44     void setHour(int h) { *this = Time(h, minute); }
45
48     void setMinute(int m) { *this = Time(hour, m); }
49
53     bool operator>(const Time& rhs) const;
54
58     bool operator<(const Time& rhs) const;
59
63     bool operator==(const Time& rhs) const;
64
68     bool operator>=(const Time& rhs) const;
69
73     bool operator<=(const Time& rhs) const;
74 };
75
78 std::ostream& operator«(std::ostream& os, const Time& rhs);
79
82 std::istream& operator»(std::istream& is, Time& rhs);
83
84 #endif
85
```

Tárgymutató