HÁZI FELADAT

Specifikáció

Feladat:

A https://infocpp.iit.bme.hu/hf_otlet oldalon található feladatok közül a **Vonatjegy** nevűt választottam. A feladatleírás másolata:

Vonatjegy
Tervezze meg egy vonatjegy eladó rendszer egyszerűsített objektummodelljét, majd valósítsa azt meg! A vonatjegy a feladatban mindig jegyet és helyjegyet jelent együtt. Így egy jegyen minimum a következőket kell feltüntetni:

- vonatszám, kocsiszám, hely
- indulási állomás, indulási idő
- érkezési állomás, érkezési idő

A rendszerrel minimum a következő műveleteket kívánjuk elvégezni:

- vonatok felvétele
- jegy kiadása

A rendszer később lehet bővebb funkcionalitású (pl. késések kezelése, vonat törlése, menetrend, stb.), ezért nagyon fontos, hogy jól határozza meg az objektumokat és azok felelősségét. Valósítsa meg a jeggyel végezhető összes értelmes műveletet operátor átdefiniálással (overload), de nem kell ragaszkodni az összes operátor átdefiniálásához! A megoldáshoz **ne** használjon STL tárolót!

Specifikáció:

A vonatjegy eladó rendszer legfontosabb alapeleme maga a vonatjegy. Egy jegyen a leírásban foglaltakon kívül a következőket fogom feltüntetni:

- jegy ára, kedvezmények (ha van)
- kocsiosztály
- retúr jegy-e
- melyik állomáson adták el

A rendszer képes továbbá a feladatleírásban szereplő műveleteken felül a felvett vonatokból menetrendet generálni, vonatokat törölni és az eddigi jegyeladásokat összesíteni.

A feladat ezenfelül előírja, hogy operátorokkal valósítsak meg jegyeken értelmezhető műveleteket. Erre a következőket tervezem implementálni:

- operator== és operator!= : két jegy minden attribútumának egyenlőségét ellenőrzik
- operator> és operator< : két jegy árát hasonlítja össze
- operator<< és operator>> : jegy adatainak kiírásához vagy betöltéséhez
- operator[] : a jegy valamilyen tulajdonságának lekérdezéséhez
  (pl.: std::cout << jegy[“ár”] → 200 ft)

A laboron egyeztetettek alapján egy osztály sablon dinamikus tömbben fogom majd tárolni többek között a vonatokat és az eladott jegyek adatait.

Teszteléshez a leírt funkciókat átfogóan kihasználó tesztprogramot fogok írni. Hibakezelést az értelemszerűen elvárható helyeken fogok alkalmazni és ezt a tesztprogram is ellenőrizni fogja.

Program használata:

A program konzolos felületen használható, az elérhető funkciók számozott menüben jelennek meg és a megfelelő szám begépelésével érhetőek el.

Amennyiben egy platformon rendelkezésre áll C++ fordító, a programnak működőképesnek kell lennie.

HÁZI FELADAT

Terv

Objektumok:

A **Vonatjegy** feladat alapeleme maga a vonatjegy. Ezt a *Jegy* osztályban fogom megvalósítani. Továbbá a *Vonat,* a gyakorlatokon elkészített *String* és megegyezés alapján a *dyn_array<T>* osztályokból fog állni a programom.

Jegy osztály:

A *Jegy* osztály mezői és metódusai a feladatleírásnak megfelelően lettek létrehozva. A vonatszám, indulási állomás, indulási idő, érkezési állomás és érkezési idő egy *Vonat* objektumra mutató pointeren keresztül érhető el mivel ezeket az információkat a *Vonat* osztály tartalmazza. A Kedvezményeknek a *Jegy* osztály csupán a megnevezését tartalmazza, ezek megadása és figyelembe vétele a jegy kiadása során történik. A mezők elérésére az operator[] használható ez azonban írásvédett, mezők módosításához a get<T>(const String&) metódus használható az igényelt attribútum nevének megadásával.

A csatolt osztálydigramon konstruktorok nem lettek feltüntetve. Az osztály rendelkezik másoló konstruktorral és egy olyan konstruktorral amely minden attribútumának (felülírhatóan) ad értelemszerű alapértéket.

Vonat osztály:

A *Vonat* osztály tartalmazza a következő információkat: vonatszám, indulási állomás, indulási idő, érkezési állomás és érkezési idő. A mezők elérése a *Jegy* osztályban ismertetett módon történik.

A csatolt osztálydigramon konstruktorok nem lettek feltüntetve. Az osztály rendelkezik másoló konstruktorral és egy olyan konstruktorral amely minden attribútumának (felülírhatóan) ad értelemszerű alapértéket.

String osztály:
A *String* osztály az ötödik laborfeladatban elkészítettel teljes mértékben megegyezik. *Részletek itt*

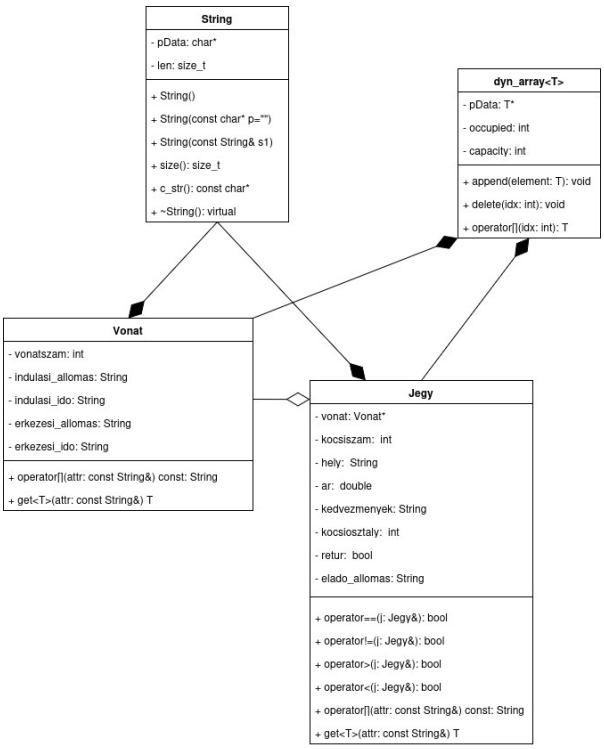
A csatolt osztálydiagramon nem minden tagfüggvény lett feltüntetve.

dyn_array<T>:
Ez egy dinamikus tömböt megvalósító sablon osztály. Három metódusa van:

- void append(T element); hozzáad egy elemet a tömb végéhez
- void delete(int idx); törli az idx indexű elemet
- T operator[](int idx); az idx indexű elemet teszi elérhetővé

Az append és delete hatására történő méret változásokat a tömb automatikusan kezeli. A lefoglalt terület méretét minden esetben a kétszeresére növeli ha több helyre van szükség. Törlésnél pedig akkor csökkenti a foglalt terület méretét a háromnegyedére ha az addig foglalt területnek kevesebb mint a fele van ténylegesen adattárolásra használva. Mindezt az occupied és a capacity mezők segítségével követi nyomon. Törlésnél az n-edik elem törlése után az n+1-edik lesz az n-edik, n+2-edik lesz az n+1-edik stb.

A csatolt osztálydiagramon konstruktorok nem lettek feltüntetve. Az osztály rendelkezik egy alapértelmezetten egy hosszúságú tömböt létrehozó konstruktorral amelynek megadható, hogy milyen hosszú tömböt szeretnénk.



Menüszerkezet:

A program parancssorból használható menürendszer által biztosítja az ígért funkciókat. Indítás után a főmenüben az alábbi opciók érhetőek el:

1. Vonat adatbázis betöltése/mentése
2. Jegy adatbázis betöltése/mentése
3. Vonat felvétele
4. Vonat törlése
5. Jegy kiadása
6. Menetrend generálása
7. Eladások összesítése
8. Kilépés

Ekkor a program nem tárol semmilyen információt de példa adatbázis (vonatokkal és jegyekkel) elérhető lesz. A megfelelő számú gomb megnyomása után az adott funkció hasonló menükben kéri be a folytatáshoz szükséges információkat.

Az első két lehetőség választásánál a fájlnév megadása után a felhasználó visszajelzést kap a művelet sikerességéről és a főmenü újra láthatóvá válik. A fájlnév megadása előtt dönteni kell arról, hogy az adatbázist betöltenénk vagy elmentenénk. Betöltésnél eldönthető, hogy a fájlból olvasott adatbázis a jelenleg tárolt mellé kerüljön a memóriában vagy felülírja azt. A program nem vizsgálja, hogy így egyezések létrejönnek-e.

Vonat felvételénél a *Vonat* osztály mezőinek értékét kell egyesével megadni majd itt is visszajelzést kapunk és a főmenü lesz újra látható. Ugyanígy történik a jegy kiadása is.

Vonat törlése esetén először kiválasztható, hogy a *Vonat* osztály mely mezője alapján szeretnénk törölni majd a találatokról egyesével eldönthető, hogy törölni szeretnénk-e vagy sem. Ekkor lehetőség van a művelet megszakítására vagy az összes találat törlésére is. A végén a program visszatér a főmenübe.

Menetrend generálásánál és Eladás összesítésnél a kimenet megjelenítésre kerül majd dönthetünk arról, hogy szeretnénk-e ezt fájlba menteni. Ha igen akkor meg kell adni a fájlnevet és utánna a főmenü lesz látható.

Készítette:
Pavlisinec Tamás

EVER - Egyszerű Vonatjegy Eladó Rendszer

Generated by Doxygen 1.9.8

Chapter 1

Prog2-NHF

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 dyn_array< T > Class Template Reference

automatikusan méretet változtató tömb tetszőleges típus tárolására

`#include <dyn_array.hpp>`

Public Member Functions

- dyn_array (int c=1)
  Konstruktor.
- void append (T element)
  Egy elem hozzáadása a tömb végére.
- void arr_delete (int idx)
  adott indexű elem törlése (hibás index esetén kivételt dob)
- T & operator[ ] (int idx)
  tárolt adatok elérése, túlindexelés esetén kivételt dob
- int occ () const
  tömbben foglalt helyek száma
- ~**dyn_array** ()
  Destruktor (felszabadítja a memóriát)

4.1.1 Detailed Description

template<typename T>
class dyn_array< T >

automatikusan méretet változtató tömb tetszőleges típus tárolására

Template Parameters

T	- tárolt adat típusa

4.1.2 Constructor & Destructor Documentation

4.1.2.1 dyn_array()

```
template<typename T >
dyn_array< T >::dyn_array (
            int c = 1 )  [inline]
```

Konstruktor.

Parameters

c	- tömb kapacitása (1)

4.1.3 Member Function Documentation

4.1.3.1 append()

```
template<typename T >
void dyn_array< T >::append (
            T element )  [inline]
```

Egy elem hozzáadása a tömb végére.

Parameters

element	- elem amit hozzáadunk

4.1.3.2 arr_delete()

```
template<typename T >
void dyn_array< T >::arr_delete (
            int idx )  [inline]
```

adott indexű elem törlése (hibás index esetén kivételt dob)

Parameters

idx	- index amit törlünk

4.1.3.3 occ()

```
template<typename T >
int dyn_array< T >::occ ( ) const  [inline]
```

tömbben foglalt helyek száma

Returns

hány elem van a tömbben

4.1.3.4 operator[]()

```
template<typename T >
T & dyn_array< T >::operator[] (
             int idx )  [inline]
```

tárolt adatok elérése, túlindexelés esetén kivételt dob

Parameters

idx	- index

Returns

referencia adott indexű tömbelemre

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

- dyn_array.hpp

4.2 Jegy Class Reference

```
#include <Jegy.hpp>
```

Public Types

- enum class JegyAttr {
  vonat , **kocsiszam** , **hely** , **ar** ,
  kedvezmenyek , **kocsiosztaly** , **retur** , **elado_allomas** ,
  ismeretlen }

  Jegy mezőit azonosító típusok.

Public Member Functions

- Jegy (Vonat ∗v=NULL, int ksz=0, const char ∗h="", double ar=0.0, const char ∗k="", int ko=3, bool r=false, const char ∗ea="")

  Konstruktor ami alapértéket ad minden mezőnek.
- Jegy (const Jegy &j)

  Másoló konstruktor.
- bool operator== (const Jegy &j) const

  Két jegy minden mezőjének egynelőségét vizsgálja.
- bool operator!= (const Jegy &j) const

két jegy minden mezőjének egyenlőségét vizsgálja

- bool operator> (const Jegy &j) const

  két jegy árának összehasonlítása
- bool operator< (const Jegy &j) const

  két jegy árának összehasonlítása
- String operator[ ] (const String &attr) const

  Jegy osztály egy mezőjének elérése.
- JegyAttr toJegyAttr (const String &a) const

  mezőt nevesítő stringből megfelelő típust hoz létre
- template<typename T >
  T & get (const String &attr)

  mezőt nevesítő String alapján referenciát ad arra a mezőre
- template<> Vonat *& **get** (const String &attr)

  get<T> template specializáció Vonat típusra*
- template<> String & **get** (const String &attr)

  get<T> template specializáció String típusra
- template<> int & **get** (const String &attr)

  get<T> template specializáció int típusra
- template<> double & **get** (const String &attr)

  get<T> template specializáció double típusra
- template<> bool & **get** (const String &attr)

  get<T> template specializáció bool típusra
- template<> Vonat *& **get** (const String &attr)

  get<T> template specializáció Vonat típusra*
- template<> String & **get** (const String &attr)

  get<T> template specializáció String típusra
- template<> int & **get** (const String &attr)

  get<T> template specializáció int típusra
- template<> double & **get** (const String &attr)

  get<T> template specializáció double típusra
- template<> bool & **get** (const String &attr)

  get<T> template specializáció bool típusra

Public Attributes

- std::invalid_argument **hibas_tipus** = std::invalid_argument("Hibas template parameter")

  kivételek a get<T> templatenak
- std::invalid_argument **nincs_mezo** = std::invalid_argument("A Jegy osztalynak nincs ilyen mezoje")

  kivételek a get<T> templatenak

4.2.1 Detailed Description

Jegy osztály.

A mezők a specifikációnak megfelelően tartalmazzák az adatokat.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Jegy() [1/2]

```
Jegy::Jegy (
            Vonat * v = NULL,
            int ksz = 0,
            const char * h = "",
            double ar = 0.0,
            const char * k = "",
            int ko = 3,
            bool r = false,
            const char * ea = "" )  [inline]
```

Konstruktor ami alapértéket ad minden mezőnek.

Parameters

v	- vonat pointer (NULL)
ksz	- kocsiszám (0)
h	- hely ("")
ar	- ár (0.0)
k	- kedvezmények ("")
ko	- kocsiosztály (3)
r	- retur jegy-e (false)
ea	- állomás ahol a jegyet eladták ("")

4.2.2.2 Jegy() [2/2]

```
Jegy::Jegy (
            const Jegy & j )  [inline]
```

Másoló konstruktor.

Parameters

j	- jegy amit másolunk

4.2.3 Member Function Documentation

4.2.3.1 get()

```
template<typename T >
T & Jegy::get (
            const String & attr )  [inline]
```

mezőt nevesítő String alapján referenciát ad arra a mezőre

Template Parameters

T	- a mező típusa

Parameters

attr	- mezőt nevesítő String

Returns

referencia a nevesített mezőre helytelen típus vagy nem létező mezőnév megadása esetén kivételt dob

4.2.3.2 operator"!=()

```
bool Jegy::operator!= (
            const Jegy & j ) const
```

két jegy minden mezőjének egyenlőségét vizsgálja

Parameters

j	- jegy amivel összehasonlítunk

Returns

igaz ha a két jegy bármely mezője nem egyenlő

4.2.3.3 operator<()

```
bool Jegy::operator< (
            const Jegy & j ) const
```

két jegy árának összehasonlítása

Parameters

j	- jegy amivel összehasonlítunk

Returns

igaz ha ennek a jegynek nagyobb az ára mint j jegynek

4.2.3.4 operator==()

```
bool Jegy::operator== (
            const Jegy & j ) const
```

Két jegy minden mezőjének egynelőségét vizsgálja.

Parameters

j	- jegy amivel összehasonlítunk

Returns

igaz ha a két jegy minden mezője egyenlő

4.2.3.5 operator>()

```
bool Jegy::operator> (
            const Jegy & j ) const
```

két jegy árának összehasonlítása

Parameters

j	- jegy amivel összehasonlítunk

Returns

igaz ha ennek a jegynek nagyobb az ára mint j jegynek

4.2.3.6 operator[]()

```
String Jegy::operator[] (
            const String & attr ) const
```

Jegy osztály egy mezőjének elérése.

Jegy egy mezőjének elérése.

Parameters

attr	- mező neve

Returns

Jegy osztály egy mezőjének Stringbe sorosított értéke ha nincs adott nevű mező a jegy osztályban akkor std::invalid_argument kivételt dob

4.2.3.7 toJegyAttr()

```
Jegy::JegyAttr Jegy::toJegyAttr (
            const String & a ) const
```

mezőt nevesítő stringből megfelelő típust hoz létre

Parameters

a	- mezőt nevesítő String

Returns

JegyAttr a megfelelő mezőhőz vagy JegyAttr::ismeretlen ha nincs ilyen mező

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

- Jegy.hpp
- Jegy.cpp

4.3 Menu Class Reference

A Felhasználói felületet megvalósító objektum.

`#include <Menu.h>`

Public Member Functions

- Menu (std::istream &ci=std::cin, std::ostream &co=std::cout)
  konstruktor
- bool active () const
  Menü állapot lekérdezéséhez.
- void check_cin (bool dob)
  EOF-et és input stream hibákat kezel.
- void **nextState** ()
  A menü megfelelő állapotába lép a bemenet alapján.

4.3.1 Detailed Description

A Felhasználói felületet megvalósító objektum.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Menu()

```
Menu::Menu (
            std::istream & ci = std::cin,
            std::ostream & co = std::cout )  [inline]
```

konstruktor

Parameters

ci	- input stream
co	- output stream

4.3.3 Member Function Documentation

4.3.3.1 active()

```
bool Menu::active ( ) const  [inline]
```

Menü állapot lekérdezéséhez.

Returns

igaz ha a user nem kért kilépést

4.3.3.2 check_cin()

```
void Menu::check_cin (
            bool dob = false )
```

EOF-et és input stream hibákat kezel.

Parameters

dob	- dobjon-e kivételt ha valami gond van

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

- Menu.h
- Menu.cpp

4.4 String Class Reference

```
#include <String.h>
```

Public Member Functions

- String (char c)

  konstrktor karakterből
- String (const char *p="")

  Konstruktor egy nullával lezárt karaktersorozatból.
- String (const String &s1)

Másoló konstruktor.

- size_t size () const
- const char * c_str () const
- String & operator= (const String &rhs_s)

  Értékadó operátor.
- String operator+ (const String &rhs_s) const

  Két stringet összefűz.
- String operator+ (char c) const

  Stringhez karaktert fűz.
- char & operator[ ] (size_t idx)

  A String adott indexű karakterét adja vissza.
- const char & operator[ ] (size_t idx) const

  A String adott indexű karakterét adja vissza.
- ~**String** ()

  Destruktor.

4.4.1 Detailed Description

String osztály A pData tartalmazza a karaktereket a lezáró nullával együtt hossz: a String hossz a lezáró nulla nélkül

4.4.2 Constructor & Destructor Documentation

4.4.2.1 String() [1/3]

```
String::String (
            char c )
```

konstrktor karakterből

konstrktor karakterből

Parameters

c	- karakter

4.4.2.2 String() [2/3]

```
String::String (
            const char * p = "" )
```

Konstruktor egy nullával lezárt karaktersorozatból.

Konstruktor egy nullával lezárt karaktersorozatból

Parameters

p	- pointer egy C stringre

4.4.2.3 String() [3/3]

```
String::String (
            const String & s1 )
```

Másoló konstruktor.

Másoló konstruktor

Parameters

s1	- String amiből másolunk

4.4.3 Member Function Documentation

4.4.3.1 c_str()

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

C-stringet ad vissza

Returns

pointer egy C-stringre

4.4.3.2 operator+() [1/2]

```
String String::operator+ (
            char c ) const
```

Stringhez karaktert fűz.

Stringhez karaktert fűz

Parameters

c	- jobboldali karakter

Returns

új String ami a régi String + karakter

4.4.3.3 operator+() [2/2]

```
String String::operator+ (
            const String & rhs_s ) const
```

Két stringet összefűz.

Két stringet összefűz

Parameters

rhs_s	- jobboldali String

Returns

új String ami a két stringet tartalmazza egymás után

4.4.3.4 operator=()

```
String & String::operator= (
            const String & rhs_s )
```

Értékadó operátor.

Értékadó operátor

Parameters

rhs_s	- jobboldali String

Returns

modosított string referenciája

4.4.3.5 operator[]() [1/2]

```
char & String::operator[] (
            size_t idx )
```

A String adott indexű karakterét adja vissza.

A String adott indexű karakterét adja vissza

Parameters

idx	- karakter indexe

Returns

karakter referencia Indexelési hiba esetén const char* kivételt dob

4.4.3.6 operator[]() [2/2]

```
const char & String::operator[] (
            size_t idx ) const
```

A String adott indexű karakterét adja vissza.

A String adott indexű karakterét adja vissza

Parameters

idx	- karakter indexe

Returns

karakter referencia Indexelési hiba esetén const char* kivételt dob

4.4.3.7 size()

```
size_t String::size ( ) const  [inline]
```

Hossz lekérdezése

Returns

String hossza

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

- String.h
- String.cpp

4.5 Vonat Class Reference

A vonatokat modellező osztály.

```
#include <Vonat.hpp>
```

Public Types

- enum class VonatAttr {
  vonatszam = 1 , **indulasi_allomas** = 2 , **indulasi_ido** = 3 , **erkezesi_allomas** = 4 ,
  erkezesi_ido = 5 , **ismeretlen** = 6 }

  Vonat mezőit azonosító típusok.

Public Member Functions

- Vonat (int vszam=0, const char *ind_a="", const char *ind_i="", const char *erk_a="", const char *erk_i="")
  Konstruktor, alapértékeket ad felülírhatóan.
- Vonat (const Vonat &v)
  Másoló konstruktor.
- String operator[ ] (const String &attr) const
  Vonat osztály egy mezőjének elérése.
- VonatAttr toVonatAttr (const String &a) const
  mezőt nevesítő Stringből megfelelő típust hoz létre
- template<typename T >
  T & get (const String &attr)
  mezőt nevesítő String alapján referenciát ad arra a mezőre
- template<> String & **get** (const String &attr)
  get<T> template specializáció String típusra
- template<> int & **get** (const String &attr)
  get<T> template specializáció int típusra
- template<> String & **get** (const String &attr)
  get<T> template specializáció String típusra
- template<> int & **get** (const String &attr)
  get<T> template specializáció int típusra

Public Attributes

- std::exception **hibas_tipus** = std::invalid_argument("Hibas template parameter")
  kivételek a get<T> templatenak
- std::exception **nincs_mezo** = std::invalid_argument("A Vonat osztalynak nincs ilyen mezoje")
  kivételek a get<T> templatenak

4.5.1 Detailed Description

A vonatokat modellező osztály.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Vonat() [1/2]

```
Vonat::Vonat (
            int vszam = 0,
            const char * ind_a = "",
            const char * ind_i = "",
            const char * erk_a = "",
            const char * erk_i = "" )  [inline]
```

Konstruktor, alapértékeket ad felülírhatóan.

Parameters

vszam	- Vonatszám
ind_a	- Állomás ahonnan a vonat indul ("")
ind_i	- Indulási idő ("")
erk_a	- Állomás ahová a vonat érkezik ("")
erk_i	- Érkezési idő ("")

4.5.2.2 Vonat() [2/2]

```
Vonat::Vonat (
            const Vonat & v )  [inline]
```

Másoló konstruktor.

Parameters

v	- vonat amit másolunk

4.5.3 Member Function Documentation

4.5.3.1 get()

```
template<typename T >
T & Vonat::get (
            const String & attr )  [inline]
```

mezőt nevesítő String alapján referenciát ad arra a mezőre

Template Parameters

T	- a mező típusa

Parameters

attr	- mezőt nevesítő String

Returns

referencia a nevesített mezőre helytelen típus vagy nem létező mezőnév megadása esetén kivételt dob

4.5.3.2 operator[]()

```
String Vonat::operator[] (
            const String & attr ) const
```

Vonat osztály egy mezőjének elérése.

Parameters

attr	- mező neve

Returns

Vonat osztály egy mezőjének Stringbe sorosított értéke

4.5.3.3 toVonatAttr()

```
Vonat::VonatAttr Vonat::toVonatAttr (
            const String & a ) const
```

mezőt nevesítő Stringből megfelelő típust hoz létre

Parameters

a	- mezőt nevesítő String

Returns

VonatAttr a megfelelő mezőhoz vagy VonatAttr::ismeretlen ha nincs ilyen mező

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

- Vonat.hpp
- Vonat.cpp

Chapter 5

File Documentation

5.1 dyn_array.hpp

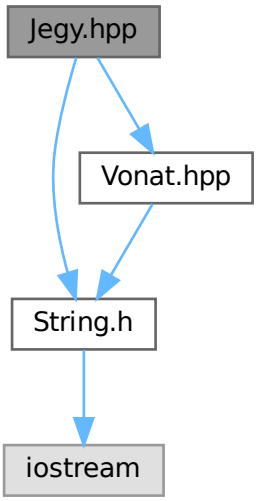
```
00001 #ifndef DYN_ARRAY_HPP
00002 #define DYN_ARRAY_HPP
00009 #include <stdexcept>
00010
00011
00014 template <typename T>
00015 class dyn_array {
00016     T* pData;
00017     int occupied;
00018     int capacity;
00019 public:
00022     dyn_array(int c = 1): pData(new T[c]), occupied(0), capacity(c) {};
00023
00026     void append(T element) {
00027         // ha van hely hozzáadjuk
00028         if (occupied < capacity) {
00029             pData[occupied++] = element;
00030         } else {
00031             // ha nincs növeljük a foglalt terület méretét
00032             capacity = capacity * 2;
00033             T* pData_new = new T[capacity];
00034             for (int i = 0; i < occupied; i++) {
00035                 pData_new[i] = pData[i];
00036             }
00037             delete[] pData;
00038             pData = pData_new;
00039             pData[occupied++] = element;
00040         }
00041     }
00042
00045     void arr_delete(int idx)  {
00046         if (idx >= 0 && idx < occupied) {
00047             for (int i = idx; i < occupied-1; i++) {
00048                 pData[i] = pData[i + 1];
00049             }
00050             occupied--;
00051             // ha nagyon kevés elem van már csak a tömbben akkor csökkentjük a foglalt terület méretét
00052             if (((double)occupied / capacity) <= 0.5 && capacity > 10 && occupied > 0) {
00053                 capacity = (int)(0.75 * capacity) + 1;
00054                 T* pData_new = new T[capacity];
00055                 for (int i = 0; i < occupied; i++) {
00056                     pData_new[i] = pData[i];
00057                 }
00058                 delete[] pData;
00059                 pData = pData_new;
00060             }
00061         } else {
00062             throw std::out_of_range("dyn_array: hibas index");
00063         }
00064     }
00065
00069     T& operator[](int idx) {
00070         if (idx >= 0 && idx < occupied) {
00071             return pData[idx];
00072         } else {
00073             throw std::out_of_range("dyn_array: hibas index");
00074         }
00075     }
```

```
00076
00079     int occ() const {
00080         return occupied;
00081     }
00082
00084     ~dyn_array() {
00085         delete[] pData;
00086     };
00087 };
00088
00089
00090 #endif
```

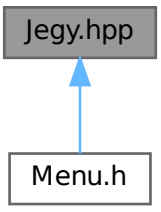
5.2 Jegy.hpp File Reference

```
#include "String.h"
#include "Vonat.hpp"
```

Include dependency graph for Jegy.hpp:



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



Classes

- class Jegy

5.2.1 Detailed Description

Ez a fájl tartalmazza a Jegy osztály deklarációját.

5.3 Jegy.hpp

Go to the documentation of this file.

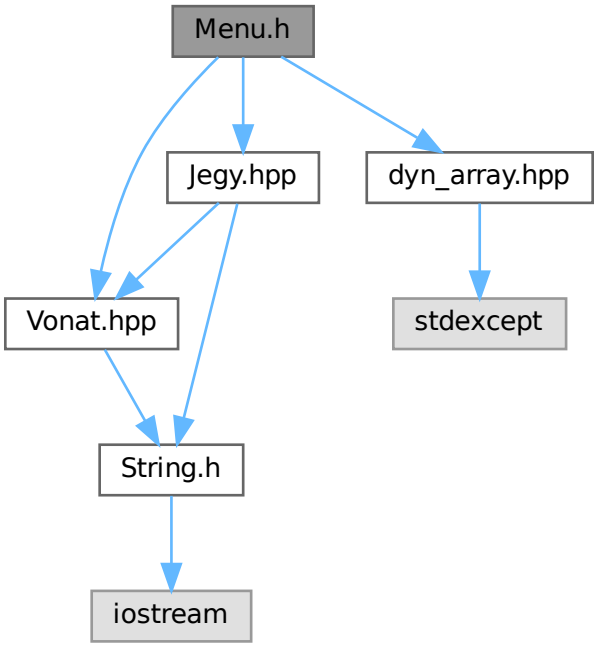
```
00001 #ifndef JEGY_H
00002 #define JEGY_H
00003
00010 #include "String.h"
00011 #include "Vonat.hpp"
00012
00018 class Jegy {
00019     Vonat* vonat;
00020     int kocsiszam;
00021     String hely;
00022     double ar;
00023     String kedvezmenyek;
00024     int kocsiosztaly;
00025     bool retur;
00026     String elado_allomas;
00027 public:
00028
00038     Jegy(Vonat* v = NULL,
00039         int ksz = 0,
00040         const char* h = "",
00041         double ar = 0.0,
00042         const char* k = "",
00043         int ko = 3,
00044         bool r = false,
00045         const char* ea = ""): vonat(v), kocsiszam(ksz), hely(h),
00046         ar(ar), kedvezmenyek(k), kocsiosztaly(ko), retur(r), elado_allomas(ea) {}
00047
00050     Jegy(const Jegy& j): vonat(j.vonat), kocsiszam(j.kocsiszam), hely(j.hely),
00051         ar(j.ar), kedvezmenyek(j.kedvezmenyek), kocsiosztaly(j.kocsiosztaly), retur(j.retur),
      elado_allomas(j.elado_allomas) {}
00052
00056     bool operator==(const Jegy& j) const;
00057
00061     bool operator!=(const Jegy& j) const;
00062
00066     bool operator>(const Jegy& j)  const;
00067
00071     bool operator<(const Jegy& j)  const;
00072
00077     String operator[](const String& attr) const;
00078
00080     enum class JegyAttr {
00081         vonat,
00082         kocsiszam,
00083         hely,
00084         ar,
00085         kedvezmenyek,
00086         kocsiosztaly,
00087         retur,
00088         elado_allomas,
00089         ismeretlen
00090     };
00091
00093     std::invalid_argument hibas_tipus = std::invalid_argument("Hibas template parameter");
00095     std::invalid_argument nincs_mezo = std::invalid_argument("A Jegy osztalynak nincs ilyen mezoje");
00096
00100     JegyAttr toJegyAttr(const String& a) const;
00101
00107     template<typename T>
00108     T& get(const String& attr) {
00109         JegyAttr jegyA = this->toJegyAttr(attr);
00110
00111         if (jegyA == JegyAttr::ismeretlen) throw nincs_mezo;
00112         throw hibas_tipus;
```

```
00113     }
00114 };
00115
00117 template<>
00118 Vonat*& Jegy::get<Vonat*>(const String& attr);
00119
00121 template<>
00122 String& Jegy::get<String>(const String& attr);
00123
00125 template<>
00126 int& Jegy::get<int>(const String& attr);
00127
00129 template<>
00130 double& Jegy::get<double>(const String& attr);
00131
00133 template<>
00134 bool& Jegy::get<bool>(const String& attr);
00135
00136 #endif
```

5.4 Menu.h File Reference

```
#include "Vonat.hpp"
#include "Jegy.hpp"
#include "dyn_array.hpp"
```

Include dependency graph for Menu.h:



Classes

- class Menu

  A Felhasználói felületet megvalósító objektum.

5.4.1 Detailed Description

A Specifikáció szerinti felhasználói felületet megvalósító objektum.

5.5 Menu.h

Go to the documentation of this file.

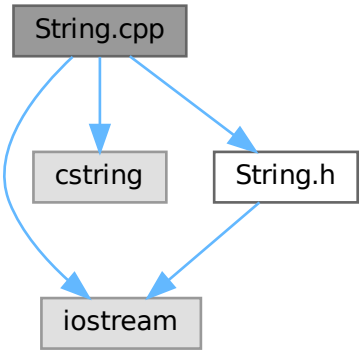
```
00001 #ifndef MENU_H
00002 #define MENU_H
00003
00010 #include "Vonat.hpp"
00011 #include "Jegy.hpp"
00012 #include "dyn_array.hpp"
00013
00015 class Menu {
00016     bool running = true;
00017     dyn_array<Vonat> vonatok;
00018     dyn_array<Jegy> jegyek;
00019     std::istream& cinput;
00020     std::ostream& coutput;
00021
00023     enum class MenuState {
00024         Alap,
00025         VonatAdat,
00026         JegyAdat,
00027         VonatFel,
00028         VonatTor,
00029         JegyKi,
00030         Menetrend,
00031         Eladasok,
00032         Kilep
00033     };
00034
00036     MenuState current = MenuState::Alap;
00037
00039     void alap();
00041     void vonatadat();
00043     void jegyadat();
00045     void vonatfel();
00047     void vonattor();
00049     void jegyki();
00051     void menetrend();
00053     void eladasok();
00055     Vonat* find_vonat(const int vsz);
00056
00060     bool str_to_bool(const String& s) const;
00061
00062 public:
00066     Menu(std::istream& ci = std::cin, std::ostream& co = std::cout): vonatok(dyn_array<Vonat>()),
      jegyek(dyn_array<Jegy>()), cinput(ci), coutput(co) {};
00067
00070     inline bool active() const {return running;}
00071
00074     void check_cin(bool dob);
00075
00077     void nextState();
00078 };
00079
00080 #endif
```

5.6 String.cpp File Reference

```
#include <iostream>
#include <cstring>
```

```
#include "String.h"
```

Include dependency graph for String.cpp:



Functions

- std::ostream & operator<< (std::ostream &os, const String &str)

  Stringet ír az ostream-re.
- std::istream & operator>> (std::istream &is, String &s0)

  istream-ről egy szót olvas be Stringbe
- String operator+ (char c, const String &str)

  Karakterhez Stringet fűz.

5.6.1 Function Documentation

5.6.1.1 operator+()

```
String operator+ (
            char ch,
            const String & str )
```

Karakterhez Stringet fűz.

Karakterhez Stringet fűz

Parameters

ch	- karakter
str	- String

Returns

új String, karakter + String

5.6.1.2 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const String & str )
```

Stringet ír az ostream-re.

Stringet ír az ostream-re

Parameters

os	- ostream
str	- String

Returns

os

5.6.1.3 operator>>()

```
std::istream & operator>> (
            std::istream & is,
            String & s0 )
```

istream-ről egy szót olvas be Stringbe

istream-ről egy szót olvas be Stringbe

Parameters

is	- istream
s0	- String

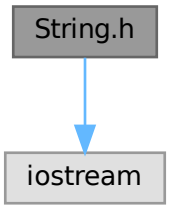
Returns

is

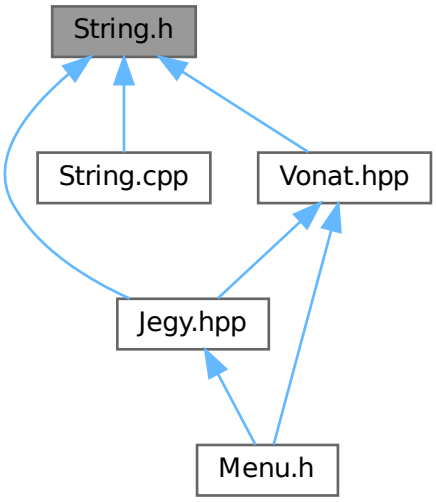
5.7 String.h File Reference

```
#include <iostream>
```

Include dependency graph for String.h:



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



Classes

- class String

Functions

- std::ostream & operator<< (std::ostream &os, const String &str)

  Stringet ír az ostream-re.
- std::istream & operator>> (std::istream &is, String &s0)

  istream-ről egy szót olvas be Stringbe
- String operator+ (char ch, const String &str)

  Karakterhez Stringet fűz.

5.7.1 Detailed Description

Ez a fájl tartalmazza az ötödik laborfeladatnak megfelelő string osztály implementációt.

5.7.2 Function Documentation

5.7.2.1 operator+()

```
String operator+ (
            char ch,
            const String & str )
```

Karakterhez Stringet fűz.

Karakterhez Stringet fűz

Parameters

ch	- karakter
str	- String

Returns

új String, karakter + String

5.7.2.2 operator<<()

```
std::ostream & operator<< (
            std::ostream & os,
            const String & str )
```

Stringet ír az ostream-re.

Stringet ír az ostream-re

Parameters

os	- ostream
str	- String

Returns

os

5.7.2.3 operator>>()

```
std::istream & operator>> (
            std::istream & is,
            String & s0 )
```

istream-ről egy szót olvas be Stringbe

istream-ről egy szót olvas be Stringbe

Parameters

is	- istream
s0	- String

Returns

is

5.8 String.h

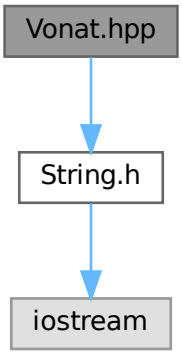
Go to the documentation of this file.

```
00001 #ifndef STRING_H
00002 #define STRING_H
00009 #include <iostream>
00010
00011
00017 class String {
00018     char *pData;
00019     size_t hossz;
00020 public:
00023     String(char c);
00024
00027     String(const char *p = "");
00028
00031     String(const String& s1);
00032
00035     inline size_t size() const { return hossz; }
00036
00039     inline const char* c_str() const { return pData; }
00040
00044     String& operator=(const String& rhs_s);
00045
00049     String operator+(const String& rhs_s) const;
00050
00054     String operator+(char c) const;
00055
00060     char& operator[](size_t idx);
00061
00066     const char& operator[](size_t idx) const;
00067
00069     ~String() { delete[] pData; }
00070 };
00071
00076 std::ostream& operator«(std::ostream& os, const String& str);
00077
00082 std::istream& operator»(std::istream& is, String& s0);
00083
00088 String operator+(char ch, const String& str);
00089
00090 #endif
```

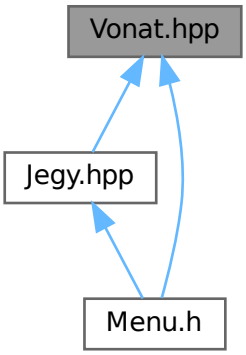
5.9 Vonat.hpp File Reference

```
#include "String.h"
```

Include dependency graph for Vonat.hpp:



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



Classes

- class Vonat

  A vonatokat modellező osztály.

5.9.1 Detailed Description

Ez a fájl tartalmazza a Vonat osztály deklarációját

5.10 Vonat.hpp

Go to the documentation of this file.

```
00001 #ifndef VONAT_H
00002 #define VONAT_H
00008 #include "String.h"
00009
00011 class Vonat {
00012     int vonatszam;
00013     String indulasi_allomas;
00014     String indulasi_ido;
00015     String erkezesi_allomas;
00016     String erkezesi_ido;
00017 public:
00024     Vonat(int vszam = 0,
00025         const char* ind_a = "",
00026         const char* ind_i = "",
00027         const char* erk_a = "",
00028         const char* erk_i = ""): vonatszam(vszam), indulasi_allomas(ind_a), indulasi_ido(ind_i),
00029         erkezesi_allomas(erk_a), erkezesi_ido(erk_i) {}
00030
00033     Vonat(const Vonat& v): vonatszam(v.vonatszam), indulasi_allomas(v.indulasi_allomas),
      indulasi_ido(v.indulasi_ido),
00034         erkezesi_allomas(v.erkezesi_ido), erkezesi_ido(v.erkezesi_ido) {}
00035
00039     String operator[](const String& attr) const;
00040
00042     enum class VonatAttr {
00043         vonatszam = 1,
00044         indulasi_allomas = 2,
00045         indulasi_ido = 3,
00046         erkezesi_allomas = 4,
00047         erkezesi_ido = 5,
00048         ismeretlen = 6
00049     };
00050
00052     std::exception hibas_tipus = std::invalid_argument("Hibas template parameter");
00054     std::exception nincs_mezo = std::invalid_argument("A Vonat osztalynak nincs ilyen mezoje");
00055
00059     VonatAttr toVonatAttr(const String& a) const;
00060
00066     template <typename T>
00067     T& get(const String& attr){
00068         VonatAttr vonatA = this->toVonatAttr(attr);
00069
00070         if (vonatA == VonatAttr::ismeretlen) throw nincs_mezo;
00071         throw hibas_tipus;
00072     }
00073 };
00074
00076 template<>
00077 String& Vonat::get<String>(const String& attr);
00078
00080 template<>
00081 int& Vonat::get<int>(const String& attr);
00082 #endif
```

Index